BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC HÀNG HẢI
BỘ MÔN: KHOA HỌC MÁY TÍNH
KHOA: CÔNG NGHỆ THÔNG TIN

BÀI GIẢNG
LẬP TRÌNH WINDOWS

TÊN HỌC PHẦN : **Lập trình Windows**
MÃ HỌC PHẦN : **17214**
TRÌNH ĐỘ ĐÀO TẠO : **ĐẠI HỌC CHÍNH QUY**
DÙNG CHO SV NGÀNH : **CÔNG NGHỆ THÔNG TIN**

HẢI PHÒNG - 2010

Bài giảng môn học: Lập trình Windows

Tên học phần: Lập trình Windows **Loại học phần:** 2

Bộ môn phụ trách giảng dạy: Khoa học Máy tính

Khoa phụ trách: CNTT

Mã học phần: 17214 **Tổng số TC**: 3

TS tiết	Lý thuyết	Thực hành/Xemina	Tự học	Bài tập lớn	Đồ án môn học
60	30	30	0	0	0

Điều kiện tiên quyết:

Sinh viên phải học xong các học phần sau mới được đăng ký học phần này:

Lập trình hướng đối tượng, Cấu trúc dữ liệu

Mục tiêu của học phần:

- Cung cấp các kiến thức cơ bản về lập trình trực quan trên hệ điều hành Windows
- Cung cấp các kiến thức về truy cập và can thiệp vào các t hành phần của hệ điều hành Windows

Nội dung chủ yếu

Các kiến thức về thao tác với file và thư mục , cơ sở dữ liệu registry , các luồng, tiến trình, dịch vụ, các thư viện liên kết động và lập trình sockets trên Windows.

Nội dung chi tiết của học phần:

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	**LT**	**TH/Xemina**	**BT**	**KT**
Chương I. Các khái niệm cơ bản		**3**	**0**	**0**	**0**
1.1. Giới thiệu về môi trường lập trình trên Windows		1			
1.1.1. Cở sở về hệ điều hành Windows					
1.1.2. Các phiên bản của hệ điều hành Windows		1			
1.1.3. Vai trò của Windows trên thị trường phần mềm					
1.2. Thư viện Win32 và Win64		1			
1.2.1. Win32 API					
1.2.2. Win64 API					
1.3. Giới thiệu về bộ công cụ Visual Studio 2005					
Chương II. Hệ thống file và thư mục		**4**	**4**	**0**	
2.1. Truy cập và sử dụng hệ thống file trên môi trường Windows		2	2		
2.1.1. Hệ thống file và thư mục của Windows					
2.1.2. Các thao tác với file và thư mục trên Windows					
2.1.3. Các vấn đề liên quan tới Unicode		1	1		
2.2. Các ví dụ về thao tác với file					
2.2.1. Tạo file và xử lý các lỗi liên quan					

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	**LT**	**TH/Xemina**	**BT**	**KT**
2.2.2. Copy file 2.2.3. Hiển thị danh sách các file trong thư mục hiện thời 2.3. Quản lý file và thư mục nâng cao 2.3.1. Con trỏ file 2.3.2. Truy cập tới các thuộc tính của file và thư mục		1	1		
Chương III. Hệ thống cơ sở dữ liệu Registry		**4**	**6**	**0**	**1**
3.1. Khái niệm và vai trò của CSDL Registry 3.1.1. Các khóa, các hive 3.1.2. Các kiểu dữ liệu 3.2. Quản lý CSDL Registry 3.2.1. Thay đổi khóa 3.2.2. Thêm mới khóa 3.2.3. Liệt kê các khóa 3.3. Can thiệp Windows qua Registry 3.3.1. Thay đổi giao diện 3.3.2. Thay đổi các thiết lập đối với các ổ đĩa 3.3.3. Thay đổi các thiết lập với người dùng		1 1 2	1 2 3		 1
Chương IV. Quản lý các tiến trình và luồng		**4**	**6**	**0**	**1**
4.1. Các tiến trình và luồng trên Windows 4.2. Các thao tác với tiến trình 4.2.1. Tạo tiến trình 4.2.2. Kết thúc và thoát khỏi một tiến trình 4.2.3. Các thao tác với biến môi trường của Windows 4.2.4. Ví dụ: Ghi nhật ký thời gian thực hiện của các tiến trình 4.3. Quản lý luồng (thread) trên Windows 4.3.1. Các khái niệm cơ bản 4.3.2. Mô hình Boss /Worker và các mô hình khác 4.3.3. Bộ nhớ dành cho luồng 4.3.4. Độ ưu tiên và các trạng thái của luồng 4.4. Một số ví dụ về tiến trình và luồng 4.4.1. Tìm kiếm song song với các tiến trình 4.4.2. Thuật toán sắp xếp trộn bằng đa luồng		2 1 1	2 2 2		 1
Chương V. Các dịch vụ của Windows		**4**	**6**	**0**	**1**
5.1. Tổng quan về dịch vụ trên Windows 5.2. Các thành phần của một dịch vụ 5.2.1. Hàm main() 5.2.2. Hàm ServiceMain() 5.2.3. Kiểm soát dịch vụ qua các Handler 5.3. Ví du: dịch vụ đơn giản trên Windows		1 1	2 2		

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	**LT**	**TH/Xemina**	**BT**	**KT**
5.4. Quản lý các dịch vụ của Windows 5.4.1. Các phương pháp kiểm soát các dịch vụ của Windows 5.4.2. Ví dụ : Điều khiển các dịch vụ của Windows		2	2		 1
Chương VI. Lập trình mạng với Sockets		**4**	**4**	**0**	**0**
6.1. Khái niệm sockets trên Windows 6.2. Các hàm sockets phía server 6.3. Các hàm sockets phía client 6.4. Ứng dụng mạng đơn giản 6.4.1. Phía server 6.4.2. Phía client 6.5. Windows Sockets 2.0		0,5 0,5 0,5 2 0,5	1 0,5 0,5 2		
Chương VII. Thư viện liên kết động		**4**	**4**	**0**	**0**
7.1. Khái niệm và ứng dụng của thư viện liên kết động 7.2. Hệ thống thư viện DLL của Windows 7.3. Các bước tạo một thư viện DLL 7.3.1. Tạo thư viện DLL 7.3.2. Viết ứng dụng gọi tới thư viện DLL		1 1 2	0,5 1 2,5		

Nhiệm vụ của sinh viên :

Tham dự các buổi thuyết trình của giáo viên, tự học, tự làm bài tập do giáo viên giao, tham dự các bài kiểm tra định kỳ và cuối kỳ.

Tài liệu học tập :

- Lê Hữu Đạt. *Lập trình Windows*. NXB Giáo dục.
- Charles Petzold. *Programming Windows, fifth edition*. Microsoft Press. 1998.
- Johnson M. Hart. *Windows System Programming Third Edition*. Addison Wesley Professional. 2004.

Hình thức và tiêu chuẩn đánh giá sinh viên:

- Hình thức thi cuối kỳ : Thi vấn đáp.
- Sinh viên phải đảm bảo các điều kiện theo Quy chế của Nhà trường và của Bộ

Thang điểm: Thang điểm chữ A, B, C, D, F

Điểm đánh giá học phần: Z = 0,3X + 0,7Y.

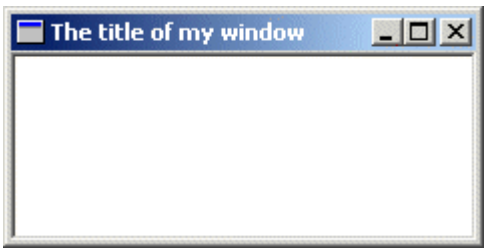
MỤC LỤC

Tôi luôn muốn làm trước và học kỹ càng sa u vì thế ở đây tôi sẽ trình bày đoạn mã chương trình tạo ra một cửa sổ trước và giải thích sau:

```
#include <windows.h>


const char g_szClassName[] = "myWindowClass"; /* tên lớp cửa sổ */


// Step 4: the Window Procedure
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
   switch(msg)
   {
      case WM_CLOSE:
         DestroyWindow(hwnd);
      break;
      case WM_DESTROY:
         PostQuitMessage(0);
      break;
      default:
        // để windows xử lý các thông điệp còn lại
         return DefWindowProc(hwnd, msg, wParam, lParam);
   }
   return 0;
}


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmdLine, int nCmdShow)
{
   WNDCLASSEX wc;
   HWND hwnd;
   MSG Msg;


   //Step 1: Registering the Window Class
   wc.cbSize        = sizeof(WNDCLASSEX);
```

```
wc.style         = 0;
wc.lpfnWndProc   = WndProc; /* hàm xử lý thông điệp cửa sổ */
wc.cbClsExtra    = 0;/* không cần thông tin thêm cho cửa sổ */
wc.cbWndExtra    = 0;
wc.hInstance     = hInstance;
wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1); /* màu nền */
/*wc.hbrBackground = GetStockObject(WHITE_BRUSH); màu nền trắng */
wc.lpszMenuName  = NULL; /* không có hệ thống thực đơn */
wc.lpszClassName = g_szClassName; /* tên lớp cửa sổ */
wc.hIconSm       = LoadIcon(NULL, IDI_APPLICATION);
/* đăng ký lớp cửa sổ */
if(!RegisterClassEx(&wc))
{
   MessageBox(NULL, "Window Registration Failed!", "Error!",
      MB_ICONEXCLAMATION | MB_OK);
   return 0;
}

// Step 2: Creating the Window
// Tạo ra một thể nghiệm của lớp cửa sổ cho ứng dụng
hwnd = CreateWindowEx(
   WS_EX_CLIENTEDGE,
   g_szClassName,
   "The title of my window",
   WS_OVERLAPPEDWINDOW,
   CW_USEDEFAULT, CW_USEDEFAULT, 240, 120,
   NULL, NULL, hInstance, NULL);

if(hwnd == NULL)
{
   MessageBox(NULL, "Window Creation Failed!", "Error!",
      MB_ICONEXCLAMATION | MB_OK);
```

```
        return 0;
    }
    // Hiển thị cửa sổ
    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    // Step 3: The Message Loop
    // Step 3: Tạo vòng lặp xử lý thông điệp
    while(GetMessage(&Msg, NULL, 0, 0) > 0)
    {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }
    return Msg.wParam;
}
```



Hình 1 – Cửa sổ của chương trình khi chạy

Đây có lẽ là một chương trình windows đơn giản nhất mà bạn có thể viết và chức năng của chương trình đơn giản là tạo ra một cửa sổ cũng rất đơn giản . Bạn nên gõ chương trình và biên dịch để chạy thử và hãy đảm bảo là không có lỗi gì xảy ra.

Bước 1: Đăng ký (Registering) lớp cửa sổ (Window).

Một lớp cửa sổ (Window Class) chứa các thông tin về kiểu cửa sổ , bao gồm thủ tục Window của nó (thủ tục này kiểm soát cửa sổ khi nó được tạo ra và đáp ứng lại các sự kiện người dùng tác động lên cửa sổ), các icon lớn và nhỏ của cửa sổ, và màu nền của cửa sổ. Theo cách này bạn có thể đăng ký một lớp và sau đó tạo ra bao nhiêu cửa sổ tùy thích mà không cần chỉ rõ tất cả các thuộc tính đó thêm một lần nào nữa . Hầu hết các thuộc tính mà bạn thiết lập trong lớp cửa sổ (window) có thể thay đổi được theo một cách rất cơ bản (thô mộc) nếu như bạn thích. Và cần chú ý là lớp cửa sổ này không liên quan gì tới khái niệm các lớp trong C++.

```
const char g_szClassName[] = "myWindowClass";
```

Biến trên lưu tên của lớp cửa sổ của chúng ta , chúng ta sẽ sử dụng nó để đăng ký lớp cửa sổ của chúng ta với hệ thống.

```
WNDCLASSEX wc;
wc.cbSize        = sizeof(WNDCLASSEX);
wc.style         = 0;
wc.lpfnWndProc   = WndProc;
wc.cbClsExtra    = 0;
wc.cbWndExtra    = 0;
wc.hInstance     = hInstance;
wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
wc.lpszMenuName  = NULL;
wc.lpszClassName = g_szClassName;
wc.hIconSm       = LoadIcon(NULL, IDI_APPLICATION);

if(!RegisterClassEx(&wc))
{
   MessageBox(NULL, "Window Registration Failed!", "Error!",
      MB_ICONEXCLAMATION | MB_OK);
   return 0;
}
```

Đó là đoạn mã chương trình chúng ta sử dụng trong hàm WinMain() để đăng ký lớp cửa sổ của chúng ta. Chúng ta sẽ điền đầy đủ các thành viên của cấu trúc WNDCLASSEX và gọi tới hàm RegisterClassEx().

Các thành viên của cấu trúc ảnh hưởng đến lớp cửa sổ gồm có:

cbSize: kích thước của cấu trúc

style: Class Style (CS_*), để không nhầm lẫn với Window Style (WS_*). Thông thường thuộc tính này được gán bằng 0.

lpfnWndProc: con trỏ trỏ tới thủ tục quản lý cửa sổ cho lớp cửa sổ này.

cbClsExtra: Lượng dữ liệu thêm được cấp phát trong bộ nhớ cho mỗi cửa sổ thuộc loại này. Thông thường cũng được gán bằng 0.

hInstance: Quản lý một thể nghiệm (instance) của ứng dụng (tham số đầu tiên của hàm WinMain()).

hIcon: Icon lớn (thường là 32x32) hiển thị khi chúng ta nhấn tổ hợp phím Alt+TAB.

hCursor: Con trỏ sẽ xuất hiện khi con trỏ chuột di chuyển qua vùng cửa sổ của chương trình.

hbrBackground: Tham số sử dụng để thiết lập mầu nền của cửa sổ.

lpszMenuName: tên của tài nguyên menu được sử dụng cho các cửa sổ của lớp.

lpszClassName: Tên để định danh lớp cửa sổ

hIconSm: Icon nhỏ (16x16) hiển thị trên thanh task bar và góc trên bên trái của cửa sổ.

Bạn không nên lo lắng nếu chưa hiểu về các thuộc tính này, chúng sẽ được giải thích ở các phần tiếp theo của tài liệu này. Một điều khác cần phải nhớ là không nên cố nhớ những thuộc tính này. Tôi rất ít khi (không muốn nói là không bao giờ) nhớ các thuộc tính cũng như các tham số của các hàm, điều đó hoàn toàn là một cố gắng lãng phí thời gian và công sức. Nếu như bạn biết các hàm bạn cần sử dụng trong một chương trình nào đó, hãy tra trong các tài liệu help. Nếu bạn không có các file help hãy down chúng về, các file kiểu này trên mạng không hề thiếu. Cuối cùng thì bạn sẽ biết các tham số của các hàm mà bạn thường xuyên sử dụng.

Tiếp đến chúng ta gọi tới hàm RegisterClassEx() và kiểm tra xem hàm này có thành công hay không, nếu như hàm trả về sai chúng ta sẽ hiển thị một thông báo thất bại và dừng chương trình.

Bước 2: Tạo ra cửa sổ chương trình

Khi đã đăng ký lớp cửa sổ, chúng ta có thể tạo ra các cửa sổ trong chương trình. Các bạn nên tra các tham số của hàm CreateWindowEx() (bạn luôn nên làm vậy khi làm việc với các hàm API mới), nhưng tôi sẽ giải thích một chút ở đây:

```
HWND hwnd;
  hwnd = CreateWindowEx(
  WS_EX_CLIENTEDGE,
    g_szClassName,
    "The title of my window",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, 240, 120,
    NULL, NULL, hInstance, NULL);
```

Tham số đầu tiên (WS_EX_CLIENTEDGE) là kiểu cửa sổ mở rộng (extended), trong trường hợp này tham số này được sử dụng để tạo thành một đường viền chìm bên trong xung quanh cửa sổ. Có thể đặt bằng 0 nếu chúng ta muốn khác. Cũng nên thay đổi các giá trị khác để xem chúng làm việc như thế nào.

Tiếp theo chúng ta cần có tên lớp (g_szClassName), tham số này báo cho hệ thống biết loại cửa sổ mà chúng ta muốn tạo ra. Vì chúng ta muốn tạo ra một cửa sổ từ lớp mà chúng ta vừa đăng ký nên chúng ta sử dụng tên của lớp cửa sổ đó. Sau đó chúng ta chỉ rõ tên của cửa sổ hoặc tiêu đề (xâu ký tự) sẽ xuất hiện trong vai trò là tham số Caption hoặc Title Bar của cửa sổ của chúng ta.

Tham số chúng ta sử dụng WS_OVERLAPPEDWINDOW trong vai trò tham số Window Style. Có khá ít các tham số kiểu này do đó bạn nên kiểm tra và thay đổi giá trị của chúng để xem kết quả hoạt động như thế nào. Chúng ta sẽ bàn thêm về các tham số này sau.

Bốn tham số tiếp theo (CW_USEDEFAULT, CW_USEDEFAULT, 320, 240) là tọa độ X và Y của góc trên bên trái của cửa sổ và các kích thước dài rộng của nó. Tôi đã thiết lập giá trị của X và Y là CW_USEDEFAULT để windows tự chọn vị trí để đặt cửa sổ chương trình khi chạy. Hãy nhớ là bên trái của màn hình là một giá trị 0 của X và nó tăng dần khi sang phải. Đỉnh của màn hình tương ứng với giá trị 0 của Y tăng dần theo chiều đi xuống. Các đơn vị là các điểm ảnh, là đơn vị nhỏ nhất mà màn hình có thể hiển thị được tại độ phân giải hiện tại.

Tiếp theo (NULL, NULL, g_hInst, NULL) chúng ta có một handle quản lý cửa sổ cha của cửa sổ được sinh ra, một handle quản lý menu, một handle cho thể nghiệm của ứng dụng và một con trỏ trỏ tới dữ liệu cửa sổ. Trong windows các cửa sổ trên màn hình được sắp xếp theo một cấu trúc phân cấp theo kiểu các cửa sổ cha và con. Khi chúng ta nhìn thấy một nút trên một cửa sổ, nút đó là con và nó được chứa trong cửa sổ là cha của nó. Trong ví dụ này của chúng ta handle của cửa sổ cha là NULL vì chúng ta không có cửa sổ cha nào cả, đây là cửa sổ chính hoặc cửa sổ ở cấp cao nhất của chúng ta. Menu cũng là NULL vì chúng ta chưa có menu chương trình nào. Handle dành cho thể nghiệm của chương trình được gán là giá trị được truyền cho tham số đầu tiên của hàm WinMain(). Dữ liệu cửa sổ (thường là chúng ta chẳng bao giờ sử dụng chúng) có thể được sử dụng để gửi các dữ liệu khác tới cửa sổ đang được tạo ra và cũng là NULL.

Nếu bạn đang băn khoăn về giá trị NULL bí hiểm này thì hãy yên tâm vì nó đơn giản chỉ là 0 (zero). Thực sự trong C nó được định nghĩa là ((void*)0) vì nó được sử dụng với các con trỏ. Vì thế cho nên bạn có thể sẽ gặp các cảnh báo khi biên dịch chương trình nếu sử dụng NULL cho các giá trị kiểu nguyên, tùy thuộc vào trình biên dịch và cấp độ cảnh báo được thiết lập với trình biên dịch. Bạn có thể bỏ qua các cảnh báo hoặc đơn giản là sử dụng giá trị 0 để thay thế.

Nguyên nhân số 1 mà mọi người không biết tại sao chương trình của mình lại có lỗi là do họ không kiểm tra các giá trị trả về của lời gọi hàm mà họ thực hiện trong chương trình của mình. CreateWindow() sẽ trả về sai trong một số trường hợp ngay cả khi bạn là một lập trình viên có nhiều kinh nghiệm, chỉ đơn giản bởi vì có rất nhiều lỗi mà chúng ta rất dễ tạo ra chúng. Cho tới khi bạn học được cánh nhanh chóng xác định các lỗi kiểu như vậy, hãy tự cho mình một cơ hội để có thể tìm ra các lỗi trong chương trình, hãy luôn kiểm tra các giá trị trả về khi gọi tới các hàm.

```
if(hwnd == NULL)
{
   MessageBox(NULL, "Window Creation Failed!", "Error!",
      MB_ICONEXCLAMATION | MB_OK);
   return 0;
}
```

Sau khi chúng ta đã tạo ra cửa sổ và kiểm tra để đảm bảo là chúng ta có một handle hợp lệ, chúng ta sẽ hiển thị cửa sổ lên màn hình của windows (hệ điều hành) bằng cách sử dụng

tham số cuối cùng của hàm WinMain() và sau đó cập nhật nó để đảm bảo là tự nó được vẽ lại một cách hợp lệ trên màn hình.

```
ShowWindow(hwnd, nCmdShow);
UpdateWindow(hwnd);
```

Tham số nCmdShow là tùy chọn, bạn có thể đơn giản gán nó là SW_SHOWNORMAL mỗi khi chúng ta làm việc với nó.Tuy nhiên sử dụng tham số được truyền vào hàm WinMain() sẽ tạo cơ hội cho các người dùng sử dụng chương trình của chúng ta có thể chỉ định họ muốn hoặc không muốn của sổ của chúng ta bắt đầu với kích thước cửa sổ lớn nhất hoặc nhỏ nhất … Bạn sẽ thấy các tùy chọn này trong các tham số của các shortcut của tới các cửa sổ.

Bước 3: Vòng lặp xử lý thông điệp

Đây chính là trái tim của toàn bộ chương trình, hầu hết những thứ mà chương trình của chúng ta xử lý là nằm ở đây.

```
while(GetMessage(&Msg, NULL, 0, 0) > 0)
{
   TranslateMessage(&Msg);
   DispatchMessage(&Msg);
}
return Msg.wParam;
```

Hàm GetMessage() lấy một thông điệp từ hàng đợi thông điệp của ứng dụng của bạn. Bất kỳ thời điểm nào người dùng di chuyển con chuột, gõ bàn phím, click chuột trên menu của cửa sổ chương trình, hoặc làm bất cứ một thao tác nào đại loại như vậy, các thông điệp sẽ được sinh ra và thêm chúng vào hàng đợi thông điệp của chương trình. Bằng cách gọi tới hàm GetMessage() bạn đang yêu cầu thông điệp tiếp theo loại bỏ khỏi hàng đợi và trả nó về cho bạn xử lý. Nếu như không có thông điệp nào hàm GetMesage() sẽ chuyển sang blocks. Nếu bạn không quen với thuật ngữ, điều đó có nghĩa là nó sẽ đợi cho tới khi có một thông điệp tiếp theo để xử lý (lấy về cho chương trình).

TranslateMessage() thực hiện thêm một số xử lý trên các sự kiện bàn phím chẳng hạn như sinh ra các thông điệp WM_CHAR cùng với các thông điệp WM_KEYDOWN. Cuối cùng DispatchMessage() gửi thông điệp tới cửa sổ mà thông điệp cần được gửi tới (xử lý theo kiểu hướng sự kiện). Đó có thể là cửa sổ chính của chương trình hoặc có thể là một cửa sổ khác, hoặc là một điều khiển, và trong một số trường hợp là một cửa sổ được tạo ra ở hậu trường bởi hệ thống hoặc một chương trình khác. Đây không phải là điều mà các bạn cần phải lo lắng vì tất cả những gì chúng ta bàn tới là lấy thông điệp và gửi nó đi, hệ thống sẽ xử lý để đảm bảo thông điệp đó được gửi đến đúng nơi sẽ nhận nó.

Bước 4: Thủ tục cửa sổ.

Nếu như vòng lặp thông điệp là trái tim của chương trình thì thủ tục cửa sổ là bộ não của chương trình. Đây là nơi mà tất cả các thông điệp được gửi tới cửa sổ của chúng ta được xử lý.

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
```

```
{
   switch(msg)
   {
      case WM_CLOSE:
         DestroyWindow(hwnd);
      break;
      case WM_DESTROY:
         PostQuitMessage(0);
      break;
      default:
         return DefWindowProc(hwnd, msg, wParam, lParam);
   }
   return 0;
}
```

Thủ tục cửa sổ được gọi tới để xử lý các thông điệp , tham số HWND là để quản lý cửa sổ của bạn, cũng chính là đối tượng của thông điệp . Điều này là quan trọng bởi vì bạn có thể có hai hoặc nhiều hơn nữa các cửa sổ cùng lớp và chúng sẽ sử dụng cùng một thủ tục cửa sổ (WndProc()). Sự khác nhau ở đây là tham số hwnd sẽ thay đổi tùy thuộc vào cửa sổ là cửa sổ nào. Chẳng hạn khi chúng ta lấy về một thông điệp WM _CLOSE chúng ta sẽ hủy cửa sổ. Và vì chúng ta sử dụng handle nhận được để quản lý cửa sổ qua tham số thứ nhất nên các cửa sổ khác sẽ không bị ảnh hưởng, chỉ có cửa sổ nhận thông điệp là sẽ bị hủy.

WM_CLOSE được gửi đến khi người dùng nhấn chu ột vào nút đóng cửa sổ hoặc nhấn tổ hợp phím Alt +F4. Thao tác này sẽ làm cho cửa sổ bị hủy theo mặc định , nhưng tôi muốn xử lý nó một cách dứt khoát , vì điều này có thể liên quan tới một số thao tác khác chẳng hạn như đóng các file dữ liệu đang mở , ngắt các kết nối cơ sở dữ liệu , các kết nối mạng vân vân trước khi thoát khỏi chương trình.

Khi chúng ta gọi tới hàm DestroyWindow () hệ thống sẽ gửi thông điệp WM_DESTROY tới cửa sổ sẽ bị hủy bỏ, trong trường hợp này là cửa sổ của chúng ta , và sau đó hủy tất cả các cửa sổ con trước khi loại bỏ cửa sổ của chúng ta khỏi hệ thống . Vì chỉ có một cửa sổ trong chương trình của chúng ta nên cũng không có nhiề u việc phải làm và chúng ta muốn thoát khỏi chương trình, vì thế hàm PostQuitMessage() được gọi tới. Hàm này sẽ gửi thông điệp WM _QUIT tới vòng lặp thông điệp . Chúng ta không bao nhận được thông điệp này vì nó làm cho hàm GetMessage() trả về FALSE, và như bạn sẽ thấy trong đoạn mã vòng lặp xử lý thông điệp của chúng ta , khi điều đó xảy ra chúng ta dừng việc xử lý các thông điệp và trả về giá trị cuối cùng , tham số wParam của thông điệp WM _QUIT là giá trị được truyền qua hàm PostQuitMessage (). Giá trị trả về chỉ thực sự có ích nếu chương trình của chúng ta được thiết kế để một chương trình khác gọi và chúng ta muốn trả về một giá trị cụ thể.

Bước 5: không có bước 5

Đến đây là hết , chúng ta không có bước 5 và tôi hy vọng các bạn đã hiểu được ít nhiều cấu trúc cơ bản của một chương trình trên Windows.

```
                    // <-    we just added this stuff
      break;              // <-
      case WM_CLOSE:
        DestroyWindow(hwnd);
      break;
      case WM_DESTROY:
        PostQuitMessage(0);
      break;
      default:
        return DefWindowProc(hwnd, msg, wParam, lParam);
    }
    return 0;
}
```

Thứ tự xử lý các thông điệp là quan trọng và cần nhớ là đối với các thông điệp khác (ngoài WM_DESTROY và WM_QUIT) cần có thêm câu lệnh break sau khi xử lý xong thông điệp.

Trước tiên tôi sẽ trình bày đoạn mã lệnh mà chúng ta sẽ thêm vào (hiển thị tên của chương trình của chúng ta) và sau đó tôi sẽ tích hợp đoạn mã đó vào chương trình của chúng ta. Trong các phần sau của chương trình t ôi sẽ chỉ cho các bạn đoạn mã và để các bạn tự tích hợp đoạn mã đó vào các chương trình. Điều này vừa tốt cho tôi : tôi sẽ không phải gõ đi gõ lại các đoạn mã lệnh và vừa tốt cho các bạn : các bạn có cơ hội thực hàn h những hiểu biết của mình để nâng cao kỹ năng thực hành . Còn nếu như bạn không chắc chắn hãy tra trong mã nguồn chương trình đi kèm với tài liệu này.

```
GetModuleFileName(hInstance, szFileName, MAX_PATH);
```

```
MessageBox(hwnd, szFileName, "This program is:", MB_OK | MB_ICONINFORMATION);
```

Hàm WndProc() của chúng ta bây giờ sẽ như sau:

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
  switch(msg)
  {
    case WM_LBUTTONDOWN:
   // BEGIN NEW CODE
    {
      char szFileName[MAX_PATH];
      HINSTANCE hInstance = GetModuleHandle(NULL);
```

```
            GetModuleFileName(hInstance, szFileName, MAX_PATH);
            MessageBox(hwnd, szFileName, "This program is:", MB_OK |
MB_ICONINFORMATION);
        }
    // END NEW CODE
        break;
        case WM_CLOSE:
            DestroyWindow(hwnd);
        break;
        case WM_DESTROY:
            PostQuitMessage(0);
        break;
        default:
            return DefWindowProc(hwnd, msg, wParam, lParam);
      }
      return 0;
    }
```

Hãy chú ý tới cặp dấu { và } mới. Các cặp dấu này là bắt buộc khi chúng ta khai báo các biến trong câu lệnh switch. Bước tiếp theo tất nhiên là dịch chương trình, chạy thử và xem kết quả của chương trình.

Chúng ta có thể chú ý là ở đây tôi đã sử dụng hai biến khi gọi hàm GetModuleFileName(), tham số thứ nhất là một handle tham chiếu tới mộ t chương trình đang chạy, nhưng chúng ta có thể lấy handle đó ở đâu ra ? Ở đây một lân nữa chúng ta lại sử dụng một hàm API khác GetModuleHandle(), thật may mắn là đối với hàm này nếu chúng ta truyền tham số là NULL vào thì kết quả trả về sẽ là handle trỏ tới file được sử dụng để tạo ra tiến trình đã gọi hàm, đó chính xác là cái mà chúng ta cần. Và do đó chúng ta có câu lệnh:

```
HINSTANCE hInstance = GetModuleHandle(NULL);
```

Tham số thứ hai khi gọ i hàm GetModuleFileName() là một con trỏ xâu để chứa đường dẫn (kết quả của hàm) tới chương trình có handle là tham số thứ nhất , kiểu của nó là LPTRSTR (hoặc LPSTR) và do LPSTR hoàn toàn tương tự như char * nên chúng ta khao báo một xâu như sau:

```
char szFileName[MAX_PATH];
```

MAX_PATH là một macro thuộc windows .h định nghĩa độ dài tối đa của một xâu để chứa độ dài đường dẫn tới một file trên windows.

lParam đều bằng 0 trong trường hợp này. Điều này là do như chúng ta đã nói trước chúng không có ý nghĩa gì đối với thông điệp WM_CLOSE.

Các hộp thoại (Dialog).

Khi bạn đã bắt đầu sử dụng các hộp thoại, bạn sẽ cần gửi các thông điệp tới các điều khiển để có thể truyền thông với chúng. Bạn có thể làm điều này bằng cách trước hết lấy handle quản lý điều khiển bằng hàm GetDlgItem() và sau đó sử dụng hàm SendMessage() hoặc có thể sử dụng hàm SendDlgItemMessage() (hàm này kết hợp công việc của cả hai hàm trên). Bạn sẽ cung cấp cho hàm một handle của một cửa sổ và một ID con và hàm sẽ lấy handle con của hộp thoại và gửi thông điệp tới cho nó. SendDlgItemMessage() và một vài hàm API tương tự khác chẳng hạn như GetDlgItemText() sẽ làm việc trên tất cả các cửa sổ chứ không chỉ với các hộp thoại.

Thế nào là hàng đợi thông điệp?

Giả sử bạn đang bận túi bụi với việc xử lý thông điệp WM_PAINT và đột nhiên người dùng thực hiện hàng loạt thao tác trên bàn phím của máy tính. Điều gì sẽ xảy ra? Bạn sẽ bị ngắt việc đang vẽ để xử lý các thao tác bàn phím của người dùng hoặc các thao tác đó sẽ bị bỏ qua? Tất cả các cách giải quyết như vậy đều có vấn đề, giải pháp ở đây là sử dụng một hàng đợi để lưu các thông điệp cần xử lý, khi các thông điệp được gửi đến chúng sẽ được thêm vào hàng đợi và khi các thông điệp được xử lý chúng sẽ được loại bỏ khỏi hàng đợi. Và để đảm bảo các thông điệp không bị bỏ qua nếu như bạn đang bận xử lý một thông điệp nào đó, các thông điệp khác sẽ được chờ trong hàng đợi cho tới khi tới lượt chúng được xử lý.

Thế nào là một vòng lặp thông điệp (vòng lặp xử lý thông điệp – Message Loop)

```
while(GetMessage(&Msg, NULL, 0, 0) > 0)
{
   TranslateMessage(&Msg);
   DispatchMessage(&Msg);
}
```

1. Vòng lặp thông điệp gọi tới hàm GetMessage(), hàm này sẽ kiểm tra hàng đợi thông điệp của bạn. Nếu như hàng đợi thông điệp là rỗng chương trình của bạn về cơ bản sẽ dừng và đợi cho tới khi có một thông điệp mới (trạng thái Block).

2. Khi một sự kiện xảy ra làm cho một thông điệp được thêm vào hàng đợi (chẳng hạn như hệ thống ghi nhận một sự kiện nhấn chuột) hàm GetMessage() sẽ trả về một giá trị nguyên dương chỉ ra rằng có một thông điệp cần xử lý, và các thông tin về thông điệp đó sẽ được điền vào cấu trúc MSG truyền cho hàm. Hàm sẽ trả về 0 nếu như thông điệp là WM_QUIT và là một giá trị âm nếu như có lỗi xảy ra.

3. Chúng ta nhận được thông điệp (qua biến Msg) và truyền cho hàm TranslateMessage(), hàm này thực hiện xử lý thêm một chút, dịch các thông tin của thông điệp thành các thông điệp ký tự. Bước này thực sự không bắt buộc nhưng một số thông điệp sẽ không làm việc được nếu không có bước này.

4. Sau đó chúng ta chuyển thông điệp cho hàm DispatchMessage(). Hàm này sẽ nhận thông điệp kiểm tra cửa sổ đích của thông điệp và tìm hàm xử lý thông điệp

(Window Procedure) của cửa sổ đó. Sau đó nó sẽ gọi tới hàm xử lý thông điệp của cửa sổ , gửi các tham số: handle của cửa sổ, thông điệp và các tham số wParam, lParam.

5. Trong hàm xử lý thông điệp bạn sẽ kiểm tra thông điệp và các tham số của nó và làm bất cứ điều gì mà bạn thích với chúng . Nếu bạn không muốn xử lý các thông điệp một cách chi tiết cụ thể, bạn hầu như chỉ việc gọi tới hàm DefWindowProc (), hàm này sẽ thực hiện các hành động mặc định cho bạn (điều này thường có nghĩa là chẳng làm gì cả).

6. Sau khi bạn đã kết thúc việc xử lý thông điệp, hàm xử lý thông điệp của bạn trả về, hàm DispatchMessage() trả về và chúng ta qua trở lại đầu vòng lặp.

Đây là một khái niệm cực kỳ quan trọng của các chương trình trên Windows . Thủ tục xử lý thông điệp của bạn không được gọi một c ách thần bí bởi hệ thống, mà chính bạn đã gọi tới chúng một cách gián tiếp thông qua việc gọi tới hàm DitpatchMessage (). Nếu như bạn muốn, bạn có thể sử dụng hàm GetWindowLong () trên handle của cửa sổ đích của thông điệp để tìm ra thủ tục xử lý cửa sổ của nó và gọi tới hàm đó một cách trực tiếp:

```
while(GetMessage(&Msg, NULL, 0, 0) > 0)
{
  WNDPROC fWndProc = (WNDPROC)GetWindowLong(Msg.hwnd, GWL_WNDPROC);
  fWndProc(Msg.hwnd, Msg.message, Msg.wParam, Msg.lParam);
}
```

Tôi đã thử cách đó với đoạn mã chương trình trước của chúng ta và nó hoạt động tốt , tuy nhiên có rất nhiều vấn đề chẳng hạn như các chuyển đổi Unicode /ANSI, các lời gọi tới các điều khiển thời gian vân vân m à hàm này không phù hợp , và khả năng rất cao là nó sẽ break với hầu hết các chương trình trừ các chương trình đơn giản . Vì thế không nên thử dùng hàm này, trừ khi bạn chỉ muốn thử nó.

Chú ý là chúng ta sử dụng hàm Get WindowLong() để lấy thủ tục xử lý cửa sổ của cửa sổ. Tại sao chúng ta không đơn giản là gọi tới hàm WndProc () một cách trực tiếp ? Vòng lặp các thông điệp của chúng ta chịu trách nhiệm đáp ứng cho tất cả các cửa sổ tron g chương trình của chúng ta , điều này bao gồm cả các thứ chẳng hạn như các nút (button) và các hộp danh sách với các hàm xử lý thông điệp của chúng , vì thế chúng ta cần đảm bảo là chúng ta gọi đến đúng hàm xử lý cửa sổ của các thành phần đó (đây thực sự là một ví dụ của khái niệm đa thể trong lập trình hướng đối tượng). Vì các cửa sổ có thể sử dụng chung một hàm xử lý thông điệp nên tham số đầu tiên (handle của cửa sổ) được dùng để chỉ cho hàm xử lý thông điệp biết cửa sổ nào là dành cho thông điệp nào.

Như bạn có thể thấy ứng dụng của bạn dành phần lớn thời gian của nó cho vòng lặp xử lý thông điệp, nơi mà bạn có thể hân hoan gửi các thông điệp tới các cửa sổ sẽ xử lý chúng . Nhưng bạn cần làm gì khi muốn thoát khỏi chương trình ? Vì chúng ta sử dụng một vòng lặp while(), nếu như hàm GetMessage () trả về FALSE , vòng lặp sẽ thoát và chúng ta sẽ tới cuối hàm WinMain() và thoát khỏi chương trình . Đó chính xác là những gì mà hàm PostQuitMessage() đã làm. Nó đặt một thông điệp WM _QUIT vào hàng đợi thônmg điệp và thay vì trả về một giá trị dương, hàm GetMesage() điền đầy đủ các thông tin cho cấu trúc Msg và trả về 0. Tại thời điểm này thành viên wParam của cấu trúc Msg chứa giá trị mà bạn đã truyền cho hàm PostQuitMessage () và bạn cũng có thể bỏ qua nó , hoặc trả về qua hàm WinMain(), giá trị đó sẽ được dùng như là mã thoát chương trình khi tiến trình kết thúc.

cần thiết phải làm việc với một trong các lo ại file này chúng ta cần phải viết thêm các đoạn mã chương trình cần thiết để có thể làm việc với các loại file khác nhau.

2.1.2 Hàm Serialize

Lớp CArchive được sử dụng trong hàm Serialize trên các đối tượng document và dữ liệu trong các ứng dụng Visual C ++. Khi một ứng dụng đọc hoặc ghi lên một file hàm Serialize của đối tượng document sẽ được gọi đến , truyền đối tượng CArchive được sử dụng để ghi hoặc đọc từ file. Trong hàm Serialize thứ tự logic được tuân theo để xác định xem đó là thao tác đọc hay ghi bằng cách gọi tới các hàm IsStoring hoặc IsLoading . Giá trị trả về từ hai hàm trên sẽ xác định xem ứng dụng cần ghi hay đọc từ luồng vào ra của lớp CArchive . Một hàm Serialize điển hình sẽ có cấu trúc như sau:

```
void CAppDoc::Serialize(CArchive& ar)
{
  // Is the archive being written to?
  if (ar.IsStoring())
  {
        // Yes, write my variable
        ar << m_MyVar;
  }
  else
  {
        // No, read my variable
        ar >> m_MyVar;
  }
}
```

Chúng ta có thể đặt một hàm Serialize trong bất cứ lớp nào mà chúng ta tạo ra để sau đó gọ đến hàm Serialize từ hàm Serialize của lớp document . Nếu chúng ta đặt các đối tượng do chúng ta tạo ra vào một mảng các đối tượng chẳng hạn như CObArray như trong ứng dụng trước đây 3 ngày chúng ta có thể gọi tới hàm Serialize của mảng từ hàm Serialize của lớp document. Đối tượng mảng sẽ , khi tới lượt nó , gọi tới hàm Serialize của các đối tượng được chứa bên trong nó.

2.1.3 Các đối tượng có thể Serialize

Khi chúng ta tạo ra lớp CLine trong bài thực hành số 10 chúng ta cần phải thêm 2 macro trước khi chúng ta có thể ghi lại hoặc nạp dữ liệu từ các file . Hai macro này , DECLARE_SERIAL và IMPLEMENT _SERIAL sẽ bao gói chức năng cần thiết trong các lớp của chúng ta để hàm Serialize có thể hoạt động đúng đắn.

Macro thứ nhất DECLARE _SERIAL được đặt ngay dòng lệnh đầu tiên trong khai báo lớp (khái niệm này giống như khái niệm giao diện của Java), nó nhận một tham số là tên của lớp, tác dụng của nó là sẽ thêm vào lớp của chúng ta các khai báo toán tử và hàm cần thiết để chức năng serialization hoạt động đúng đắn.

Ví dụ như sau:

```
class CMyClass : public CObject
{
  DECLARE_SERIAL (CMyClass)
  public:
        virtual void Serialize(CArchive &ar);
        CMyClass();
        virtual ~CMyClass();

};
```

Macro thứ hai IMPLEMENTATION _SERIAL được đặt trong phần cài đặt của lớp do chúng ta tạo ra. Macro này cần phải đặt bên ngoài bất cứ các hàm thành viên nào của lớp vì nó sẽ thêm mã của các hàm cần thiết cho lớp tương ứng với khai báo của macro DECLARE_SERIAL.

Macro này nhận 3 tham số. Tham số thứ nhất là tên lớp, giống như macro thứ nhất. Tham số thứ hai là tên của lớp cơ sở (lớp cha). Tham số thứ ba là một số version có thể được sử dụng để xác định một file có là đúng version để thực hiện thao tác đọc với ứng dụng của chúng ta hay không. Số version này phải là một số dương nên được tăng lên mỗi lần phương thức serializeation của lớp được thay đổi thay bất cứ cách thức nào làm thay đổi dữ liệu được ghi hoặc đọc từ file. Ví dụ về khai báo của macro thứ hai này như sau:

```
// MyClass.cpp: implementation of the CMyClass class.

#include "stdafx.h"
#include "MyClass.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

IMPLEMENT_SERIAL (CMyClass, CObject, 1)
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
```

3. Phần Advanced features bỏ dấu check ActiveX control

4. Phần Generated Classes chọn lớp cơ sở của lớp CSerialzeView là CFormView

5. Nhấn Finish , chúng ta sẽ nhận được thông báo là chương trình sẽ không có chức năng in ấn, nhấn Yes để tiếp tục

Sau khi tạo một ứng dụng SDI hoặc MDI trong đó lớp view kế thừa từ lớp CFormView chúng ta cần thiết kế form view của chương trình trình , quá trình này cũng giống như thiết kế các hộp thoại trong các ứng dụng dạng Dialog based nhưng chúng ta không cần có các nút để thoát khỏi chương trình hoặc hủy bỏ quá trình thực hiện như trên các hộp thoại thông thường . Với một ứng dụng SDI hoặc MDI chức năng ghi và thoát cử a sổ được đặt trên các hệ thống menu chương trình hoặc trên các thanh công cụ.

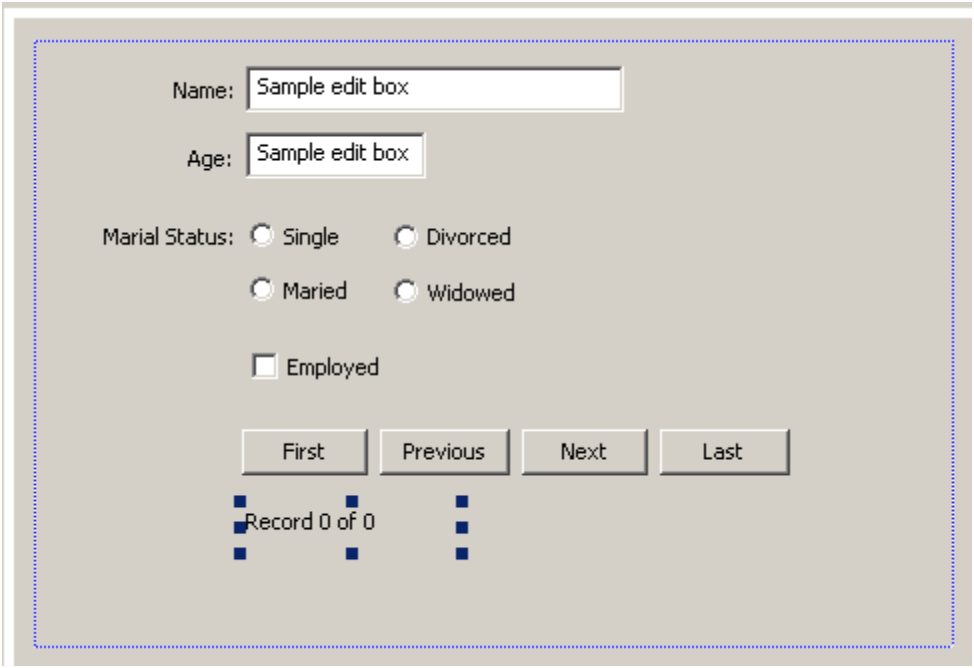
Chú ý: Nếu chúng ta làm việc với các ứng dụng dạng Dialog based App Wizard sẽ không cung cấp các đoạn mã serialization cho ứng dụng chúng ta cần p hải tự thực hiện điều này nếu muốn.

Thiết kế form của chương trình gồm các điều khiển như bảng sau:

Object	*Property*	*Setting*
Static Text	**ID**	**IDC_STATIC**
	Caption	&Name:
Edit Box	ID	IDC_ENAME
Static Text	ID	IDC_STATIC
	Caption	&Age
Edit Box	ID	IDC_EAGE
Static Text	ID	IDC_STATIC
	Caption	Marital Status:
Radio Button	ID	IDC_RSINGLE
	Caption	&Single
	Group	Checked
Radio Button	ID	IDC_RMARRIED
	Caption	&Married
Radio Button	ID	IDC_RDIVORCED
	Caption	&Divorced
Radio Button	ID	IDC_RWIDOW
	Caption	&Widowed
Check Box	ID	IDC_CBEMPLOYED
	Caption	&Employed
Button	ID	IDC_BFIRST
	Caption	&First

Button	ID	IDC_BPREV
	Caption	&Previous
Button	ID	IDC_BNEXT
	Caption	Nex&t
Button	ID	IDC_BLAST
	Caption	&Last
Static Text	ID	IDC_SPOSITION
	Caption	Record 0 of 0

Giao diện chương trình sau khi thiết kế trong như sau:



Khi chúng ta phát triển các ứng dụng hoặc cửa sổ kiểu dialog -based chúng ta sẽ gắn các biến cho các điều khiển trên cửa sổ của lớp hộp thoại . Tuy nhiên đối với một ứng dụng dạng SDI hoặc MDI chúng ta sẽ gắn cho lớp nào ? Vì hàm UpdateData là một hàm của lớp CWnd và lớp view là một hậu duệ của lớp CWnd, mặc dù lớp document thì không phải , nên lớp view sẽ là nơi logic nhất để đặt các biến gắn với các điều khiển trên cửa sổ chương trình.

Để gán các biến cho các điều khiển của chương trình chúng ta mở Class Wizard v à gắn các biến cho các điều khiển và chỉ định lớp chứa chúng là lớp view cụ thể ở ứng dụng này là CSerializeView. Các biến được gán cho các điều khiển như sau:

Object	*Name*	*Category*	*Type*
IDC_CBEMPLOYED	m_bEmployed	Value	BOOL
IDC_EAGE	m_iAge	Value	int
IDC_ENAME	m_sName	Value	CString
IDC_RSINGLE	m_iMaritalStatus	Value	int

IDC_SPOSITION	m_sPosition	Value	CString

Nếu như chúng ta kiểm tra mã chương trình của lớp view chúng ta sẽ thấy rằng không có hàm OnDraw. Nếu chúng ta sử dụng lớp tổ tiên CFormView cho ứng dụng SDI hoặc MDI chún g ta không cần lo lắng về hàm OnDraw. Thay vào đó chúng ta sẽ đối xử với lớp view rất giống với lớp hộp thoại như trong một cửa s ổ hộp thoại hoặc một ứng dụng dạng dialog - based. Sự khác nhau chủ yếu ở đây là dữ liệu chương trình mà chúng ta cần sử dụng để gán cho các điều khiển của cửa sổ chương trình không phải là của lớp view mà là của lớp document. Do đó chúng ta cần xâu dựng các tương tác giữa các lớp này để truyền các dữ liệu giữa chúng.

2.2.2 Tạo lớp Serializable

Khi chúng ta tạo ra một ứng dụng dạng form-based chúng ta sẽ giả sử rằng chương trình sẽ chứa nhiều bản ghi trong form và người dùng có thể duyệt qua các bản ghi này để thực hiện các thay đổi và ghi lại. Người dùng cũng có thể thêm vào các bản ghi mới và loại bỏ bản ghi nào đó. Thách thức đối với chúng ta trong việc xây dựng ứng dụng là làm thế nào để biểu diễn các bản ghi đó để có thể hỗ trợ tất cả các chức năng của chương trình.

Một cách tiếp cận là tạo ra một lớp bao gói tất cả các bản ghi và sau đó chứa tất cả các bản ghi trong một mảng giống như cách mà chúng ta đã thực hiện với các ứng dụng trong ba bài thực hành trước đây. Lớp này sẽ cần kế thừa từ lớp CObject và cần chứa các biến cho tất cả các biến tương ứng với các điều k hiển được thêm vào lớp view cùng với phương thức đọc và ghi tất cả các biến này. Cùng với việc thêm vào các phương thức đọc và ghi các biến chúng ta cần phải làm cho lớp này là một lớp serialiable bằng các thêm vào hàm Serialize cho lớp cùng với hai macro đã mô tả ở trên.

Tạo một lớp cơ bản

Như các bạn có thể nhớ từ bài thực hành số 10 khi chúng ta muốn tạo ra một lớp mới chúng ta cần chọn tab Class View và nhấn chuột phải chọn Add | Add Class. Sau đó trong hộp thoại tiếp theo chọn đó là một Generic class , chúng ta sẽ học nhiều hơn về các lớp này trong bài thực hành số 16. Tiếp theo chúng ta gõ tên lớp là CPerson , lớp cơ sở của lớp là lớp CObject. Sau khi tạ o xong lớp chúng ta tiếp tục thêm các biến thành viên cho lớp , các biến thành viên này cần phải khớp với các biến được gán cho các điều khiển trên cửa sổ chương trình trong lớp view như trong bảng sau:

Name	**Type**
m_bEmployed	BOOL
m_iAge	int
m_sName	CString
m_iMaritalStatus	int

Cài đặt phương thức đọc và ghi cho lớp CPerson

Sau khi đã tạo ra lớp CPerson chúng ta sẽ cung cấp một phương tiện để đọc và ghi các biến cho lớp. Cách dễ nhất để cung cấp các chức năng này là thêm các hàm inline trong định nghĩa của lớp, chúng ta có thể thêm một tập cá c hàm inline để thiết lập và lấy giá trị của các

biến (riêng biệt). Ví dụ chúng ta có thể mở file header (Person.h) và sửa lại khai báo của lớp CPerson như sau:

```
class CPerson : public CObject
{
  public:
        // Functions for setting the variables
        void SetEmployed(BOOL bEmployed) { m_bEmployed = bEmployed;}
        void SetMaritalStat(int iStat) { m_iMaritalStatus = iStat;}
        void SetAge(int iAge) { m_iAge = iAge;}
        void SetName(CString sName) { m_sName = sName;}
        // Functions for getting the current settings of the variables
        BOOL GetEmployed() { return m_bEmployed;}
        int GetMaritalStatus() { return m_iMaritalStatus;}
        int GetAge() {return m_iAge;}
        CString GetName() {return m_sName;}
        CPerson();
        virtual ~CPerson();
  private:
        BOOL m_bEmployed;
        int m_iMaritalStatus;
        int m_iAge;
        CString m_sName;
};
```

Hàm cấu tử mặc định được Visual C++ tự động cung cấp và chúng ta không cần sửa đổi (với Visual 6.0 thì vẫn cần thêm hàm này vào).

2.2.3 Cài đặt hàm Serialize

Tiếp đến chúng ta sẽ thêm hàm Serialize cho lớp bắt đầu bằng việc thêm 2 macro vào đúng vị trí của chúng trong các file khai báo và cài đặt của lớp CPerson , sau đó thêm hàm virtual, public Serialize(CArchive &ar) cho lớp CPerson như sau:

```
void CPerson::Serialize(CArchive &ar)
{
  // Call the ancestor function
  CObject::Serialize(ar);

  // Are we writing?
```

```
    if (ar.IsStoring())
        // Write all of the variables, in order
        ar << m_sName << m_iAge << m_iMaritalStatus << m_bEmployed;
    else
        // Read all of the variables, in order
        ar >> m_sName >> m_iAge >> m_iMaritalStatus >> m_bEmployed;
}
```

Về bản chất hàm này cũng khá đơn giản , ban đầu nó gọi tới phương thức Serialize của lớp cha để xác định xem đó là thao tác đọc hay ghi dữ liệu, sau đó nó gọi tới hàm IsStoring để xác định nếu là ghi dữ liệu thì thực hiện ghi các biến thành viên của lớp còn ngược lại thì thực hiện thao tác đọc (chú ý thứ tự của các biến là quan trọng).

Chú ý: chúng ta biết là một ứng dụng không phải khi nào cũng có thể đọc hoặc ghi dữ liệu thành công nên trên thực tế khi chúng ta tiến hành thực hiện hàm trên sẽ có một Exception được throw để kiểm soát lỗi nhưng ở đây chúng ta tạm thời bỏ qua vấn đề này (xem phụ lục A cuốn Teach yourself Visual C++ 6.0 in 21 days để biết thêm chi tiết).

Cài đặt các chức năng cho lớp document

Khi chúng ta xây dựng một ứng dụng dạng form-based trong đó form nằm trên cửa sổ là không gian chính để người dùng có thể tương tác với ứng dụng có một giả sử không được đề cập rõ ràng là ứng dụng của chúng ta sẽ cho phép người dùng làm việc với nhiều bản ghi . Điều này có nghĩa là chúng ta cần hỗ trợ các tính năng lưu và duyệt qua các bản ghi này . Việc lưu các bản ghi có thể thực hiện dễ dàng bằng cách sử dụng một mảng như chúng ta đã từng làm trong bài thực hành số 10. Cách làm này cho phép chúng ta có thể thêm vào một số lượng bản ghi không hạn chế (không biết có đúng không nữa). Việc duyệt qua các bản ghi được thực hiện qua bốn thao tác là First (duyệt bản ghi đầu tiên), Last (bản ghi cuối cùng), Previous (bản ghi trước) và Next (bản ghi tiếp theo). Chúng ta cần một chức năng thông báo để xác định bản ghi nào đang được hiển thị.

Để lưu trữ và hỗ trợ các tính năng này lớp document cần hai biến : một mảng và một chỉ số bản ghi hiện tại như bảng sau:

Name	**Type**
m_iCurPosition	int
m_oaPeople	CObArray

Một việc khác chúng ta cần làm là include file header của lớp CPerson vào file cài đặt của lớp document (vị trí trước các file header của lớp document và view) như sau:

```
#include "stdafx.h"
#include "Serialize.h"
#include "Person.h"
```

```
#include "SerializeDoc.h"
#include "SerializeView.h"
```

Thêm một bản ghi mới

Trước khi chúng ta có thể duyệt qua các bản ghi trong chương trình chúng ta cần xây dựng chức năng thêm một bản ghi mới cho mảng các đối tượng . Cách tiếp cận tương tự như bài thực hành sẽ được sử dụng , và vì các bản ghi mặc định đều có các trường dữ liệu là rỗng nên chúng ta chỉ cần sử dụng hàm cấu tử mặc định do Visual C ++ cung cấp là đủ , đồng thời mỗi khi thêm vào một bản ghi mới chúng ta sẽ gán bản ghi hiện tại là bản ghi mới đó (hàm này là private).

```
CPerson* CSerializeDoc::AddNewRecord(void)
{
  // Create a new CPerson object
  CPerson *pPerson = new CPerson();
  try
  {
        // Add the new person to the object array
        m_oaPeople.Add(pPerson);
        // Mark the document as dirty
        SetModifiedFlag();
        // Set the new position mark
        m_iCurPosition = (m_oaPeople.GetSize() - 1);
  }
  // Did we run into a memory exception?
  catch (CMemoryException* perr)
  {
        // Display a message for the user, giving them the
        // bad news
        AfxMessageBox("Out of memory", MB_ICONSTOP | MB_OK);
        // Did we create a line object?
        if (pPerson)
        {
              // Delete it
              delete pPerson;
              pPerson = NULL;
        }
```

```
    // Delete the exception object
        perr->Delete();
    }
    return pPerson;
}
```

Tương tự như bài thực hành số 10 chúng ta cần các hàm lấy tổng số bản ghi , số thứ tự bà đối tượng tương ứng với bản ghi hiện tại như sau (các hàm này là public):

```
int CSerializeDoc::GetTotalRecords(void)
{
    return m_oaPeople.GetCount();
}

int CSerializeDoc::GetCurRecordNbr(void)
{
    return m_iCurPosition + 1;
}

CPerson* CSerializeDoc::GetCurRecord(void)
{
    // Are we editing a valid record number?
    if (m_iCurPosition >= 0)
        // Yes, return the current record
        return (CPerson*)m_oaPeople[m_iCurPosition];
    else
        // No, return NULL
    return NULL;
}
```

Các chức năng tiếp theo cần được cài đặt là các hàm cho phép thực hiện các thao tác lấy các bản ghi của mảng một cách tương đối (đầu, cuối, trước, sau):

```
CPerson* CSerializeDoc::GetFirstRecord(void)
{
    // Are there any records in the array?
    if (m_oaPeople.GetSize() > 0)
    {
```

```
            // Yes, move to position 0
            m_iCurPosition = 0;
            // Return the record in position 0
            return (CPerson*)m_oaPeople[0];
  }else
            // No records, return NULL
            return NULL;
}

CPerson* CSerializeDoc::GetNextRecord(void)
{
  // After incrementing the position marker, are we
  // past the end of the array?
  if (++m_iCurPosition < m_oaPeople.GetSize())
            // No, return the record at the new current position
            return (CPerson*)m_oaPeople[m_iCurPosition];
  else
            // Yes, add a new record
  return AddNewRecord();
}
CPerson* CSerializeDoc::GetPrevRecord(void)
{
  // Are there any records in the array?
  if (m_oaPeople.GetSize() > 0)
  {
            // Once we decrement the current position,
            // are we below position 0?
            if (--m_iCurPosition < 0)
                        // If so, set the record to position 0
                        m_iCurPosition = 0;
                        // Return the record at the new current position
            return (CPerson*)m_oaPeople[m_iCurPosition];
  }else
            // No records, return NULL
```

```
  return NULL;
}

CPerson* CSerializeDoc::GetLastRecord(void)
{
  // Are there any records in the array?
  if (m_oaPeople.GetSize() > 0)
  {
        // Move to the last position in the array
        m_iCurPosition = (m_oaPeople.GetSize() - 1);
        // Return the record in this position
        return (CPerson*)m_oaPeople[m_iCurPosition];
  }else
        // No records, return NULL
        return NULL;
}
```

Tiếp đến là hàm Serialize cho mảng các đối tượng của lớp document (CSerializeDoc):

```
void CSerializeDoc::Serialize(CArchive& ar)
{
  // Pass the serialization on to the object array
  m_oaPeople.Serialize(ar);
}
```

Hàm làm công tác môi trường, dọn dẹp tất cả mọi thứ trước khi bắt đầu một tài liệu mới (hàm này được gọi tới khi chương trình kết thúc hoặc trước khi một tài liệu mới được mở):

```
void CSerializeDoc::DeleteContents()
{
  // TODO: Add your specialized code here and/or call the base class
  // Get the number of lines in the object array
  int liCount = m_oaPeople.GetSize();
  int liPos;

  // Are there any objects in the array?
  if (liCount)
  {
```

```
            // Loop through the array, deleting each object
            for (liPos = 0; liPos < liCount; liPos++)
                  delete m_oaPeople[liPos];
            // Reset the array
            m_oaPeople.RemoveAll();
      }
      CDocument::DeleteContents();
}
```

Chúng ta có thể thấy các bước thực hiện hoàn toàn giống với bài thực hành số 10, và cũng cần nhắc lại một chú ý đó là : cần phải chuyển đổi tượng lấy từ mảng CObArray thành kiểu CPerson vì đó là một biến kiểu CObject.

Tiếp theo cần phải sửa lại hàm tương ứng với sự kiện OnNewDocument:

```
BOOL CSerializeDoc::OnNewDocument()
{
  if (!CDocument::OnNewDocument())
            return FALSE;

  // TODO: add reinitialization code here
  // (SDI documents will reuse this document)
  // If unable to add a new record, return FALSE
  if (!AddNewRecord())
            return FALSE;

  // Get a pointer to the view
  POSITION pos = GetFirstViewPosition();
  CSerializeView* pView = (CSerializeView*)GetNextView(pos);
  // Tell the view that it's got a new data set
  if (pView)
            pView->NewDataSet();
  return TRUE;
}
```

Khi một tài liệu mới bắt đầu chương trình sẽ đưa ra một form rỗng sẵn sàng để nhập thông tin mới, và để bản ghi này có thể sẵn sàng nhận thông tin chúng ta thêm vào một bản ghi trong mảng các đối tượng và khi một bản ghi mới được thêm vào mảng chúng ta cần thay đổi việc hiển thị để chỉ ra rằng bản ghi mới đó tồn tại ngược lại các hiển thị sẽ tiếp tục với bản

ghi cuối cùng từ tập bản ghi trước (và người dùng có thể băn khoăn tại sao ứng dụng của chúng ta không bắt đầu với một tập bản ghi mới).

Khi mở một tập dữ liệu sẵn có chúng ta không cần thêm vào bất cứ bản ghi mới nào nhưng vẫn cần phải cho đối tượng view biết rằng nó cần phải làm tươi bản ghi được hiển thị cho người dùng. Do đó chúng ta có thể thêm đoạn mã tương tự cho hàm OnOpenDocument như sau (bỏ phần đầu có chức năng thêm vào một bản ghi mới) như sau:

```
BOOL CSerializeDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
  if (!CDocument::OnOpenDocument(lpszPathName))
        return FALSE;

  // TODO:  Add your specialized creation code here
        // Get a pointer to the view
  POSITION pos = GetFirstViewPosition();
  CSerializeView* pView = (CSerializeView*)GetNextView(pos);
  // Tell the view that it's got a new data set
  if (pView)
        pView->NewDataSet();

  return TRUE;
}
```

Đó là tất cả các công việc chuẩn bị, tổ chức và xử lý dữ liệu của lớp document, tiếp đến chúng ta sẽ làm việc với lớp view để tương tác với người dùng.

Điều đầu tiên cần chú ý là các include trong các file mã nguồn cần theo đúng thứ tự (giống bài thực hành số 10): lớp CPerson trước, sau đó tới lớp document và cuối cùng là lớp view và các chỉ thị include này chỉ thực hiện trong các file cài đặt lớp (khác với C/C++ thông thường)

```
#include "stdafx.h"
#include "Serialize.h"

#include "Person.h"
#include "SerializeDoc.h"
#include "SerializeView.h"
```

Vì số lượng các thao tác đối với các bản ghi là khá nhiều nên cũng giống như bài thực hành 10 (sử dụng 1 biến lưu điểm hiện tại của con trỏ chuột) trong bài thực hành này để cho tiện chúng ta thêm một biến thành viên kiểu CPerson * có tên là m_pCurPerson cho lớp View.

Hàm đầu tiên mà chúng ta sẽ thực hiện là hàm hiển thị dữ liệu , nhưng chức năng này được sử dụng trong hầu hết các tương tác nên chúng ta sẽ l àm một hàm riêng để sau đó gọi đến hàm này (giống hàm Draw của lớp CLine trong bài thực hành 10 về chức năng) (hàm private):

```
void CSerializeView::PopulateView(void)
{
  // Get a pointer to the current document
  CSerializeDoc* pDoc = GetDocument();
  if (pDoc)
  {
        // Display the current record position in the set
        m_sPosition.Format("Record %d of %d", pDoc->GetCurRecordNbr(),
              pDoc->GetTotalRecords());
  }
  // Do we have a valid record object?
  if (m_pCurPerson)
  {
        // Yes, get all of the record values
        m_bEmployed = m_pCurPerson->GetEmployed();
        m_iAge = m_pCurPerson->GetAge();
        m_sName = m_pCurPerson->GetName();
        m_iMaritalStatus = m_pCurPerson->GetMaritalStatus();
  }
  // Update the display
  UpdateData(FALSE);
}
```

Tiếp đến là các hàm duyệt qua các bản ghi , đồng thời cũng là các hàm xử lý các sự kiện tương ứng với các nút lệnh:

```
void CSerializeView::OnBnClickedBfirst()
{
  // TODO: Add your control notification handler code here
  // Get a pointer to the current document
  CSerializeDoc * pDoc = GetDocument();
  if (pDoc)
  {
```

```
            // Get the first record from the document
            m_pCurPerson = pDoc->GetFirstRecord();
            if (m_pCurPerson)
            {
                  // Display the current record
                  PopulateView();
            }
  }
}


void CSerializeView::OnBnClickedBlast()
{
  // TODO: Add your control notification handler code here
  // Get a pointer to the current document
  CSerializeDoc * pDoc = GetDocument();
  if (pDoc)
  {
            // Get the last record from the document
            m_pCurPerson = pDoc->GetLastRecord();
            if (m_pCurPerson)
            {
                  // Display the current record
                  PopulateView();
            }
  }
}
void CSerializeView::OnBnClickedBprev()
{
  // TODO: Add your control notification handler code here
  // Get a pointer to the current document
  CSerializeDoc * pDoc = GetDocument();
  if (pDoc)
  {
```

```
            // Get the last record from the document
            m_pCurPerson = pDoc->GetPrevRecord();
            if (m_pCurPerson)
            {
                  // Display the current record
                  PopulateView();
            }
    }
}


void CSerializeView::OnBnClickedBnext()
{
    // TODO: Add your control notification handler code here
    // Get a pointer to the current document
    CSerializeDoc * pDoc = GetDocument();
    if (pDoc)
    {
            // Get the last record from the document
            m_pCurPerson = pDoc->GetNextRecord();
            if (m_pCurPerson)
            {
                  // Display the current record
                  PopulateView();
            }
    }
}
```

Tiếp đến chúng ta cần một hàm reset lại lớp view mỗi khi một bản ghi mới được bắt đầu hoặc được mở để người không tiếp tục nhìn thấy tập bản ghi cũ . Chúng ta có thể gọi tới hàm xử lý sự kiện của nút First để buộc lớp view đưa ra bản ghi đầu tiên trong tập bản ghi . Để làm điều này chúng ta thêm một hàm void (pubic) tên là NewDataSet như sau:

```
void CSerializeView::NewDataSet(void)
{
    OnBnClickedBfirst();
}
```

Đến đây chúng ta có thể dịch và chạy chương trình nhưng các bạn sẽ thấy chỉ các các nút duyệt qua các bản ghi là có tác dụng còn các điều khiển khác của form là không có tác dụng gì. Điều này là do chúng ta chưa có các hàm xử lý các điều khiển trên form . Cần thêm các hàm xử lý các sự kiện với các điều khiển trên form chương trình như sau:

Hàm xử lý dấu check Employed:

```
void CSerializeView::OnBnClickedCbemployed()
{
  // TODO: Add your control notification handler code here
  UpdateData(TRUE);
  // If we have a valid person object, pass the data changes to it
  if (m_pCurPerson)
        m_pCurPerson->SetEmployed(m_bEmployed);
}
```

hàm xử lý các sự kiện cho các nút Radio:

```
void CSerializeView::OnBnClickedMaritalstatus()
{
  // TODO: Add your control notification handler code here
  UpdateData(TRUE);
  // If we have a valid person object, pass the data changes to it
  if (m_pCurPerson)
        m_pCurPerson->SetMaritalStat(m_iMaritalStatus);
}
```

Đối với các trường tên và tuổi chúng ta cần xử lý sự kiện EN _CHANGE và gọi tới các hàm SetName, SetAge tương ứng của lớp CPerson như sau:

```
void CSerializeView::OnEnChangeEname()
{
  // TODO:  Add your control notification handler code here
  UpdateData(TRUE);
  // If we have a valid person object, pass the data changes to it
  if (m_pCurPerson)
        m_pCurPerson->SetName(m_sName);
}


void CSerializeView::OnEnChangeEage()
{
```

```
GetVersionEx (&OSVer);
IsNT = (OSVer.dwPlatformId == VER_PLATFORM_WIN32_NT);
/* NT (all versions) returns VER_PLATFORM_WIN32_NT. */
GetStartupInfo (&StartUp);
GetSystemTime (&StartTimeSys);

/* Execute the command line; wait for process to complete. */
CreateProcess (NULL, targv, NULL, NULL, TRUE,
    NORMAL_PRIORITY_CLASS, NULL, NULL, &StartUp, &ProcInfo);

/* Assure that we have all REQUIRED access to the process. */
DuplicateHandle (GetCurrentProcess (), ProcInfo.hProcess,
    GetCurrentProcess (), &hProc,
    PROCESS_QUERY_INFORMATION | SYNCHRONIZE, FALSE, 0);
WaitForSingleObject (hProc, INFINITE);
GetSystemTime (&ExitTimeSys);

if (IsNT) { /* W NT. Elapsed, Kernel, & User times. */
  GetProcessTimes (hProc, &CreateTime.ft,
      &ExitTime.ft, &KernelTime, &UserTime);
  ElapsedTime.li = ExitTime.li - CreateTime.li;
  FileTimeToSystemTime (&ElapsedTime.ft, &ElTiSys);
  FileTimeToSystemTime (&KernelTime, &KeTiSys);
  FileTimeToSystemTime (&UserTime, &UsTiSys);
  _tprintf (_T ("Real Time: %02d:%02d:%02d:%03d\n"),
      ElTiSys.wHour, ElTiSys.wMinute, ElTiSys.wSecond,
      ElTiSys.wMilliseconds);
  _tprintf (_T ("User Time: %02d:%02d:%02d:%03d\n"),
      UsTiSys.wHour, UsTiSys.wMinute, UsTiSys.wSecond,
      UsTiSys.wMilliseconds);
  _tprintf (_T ("Sys Time: %02d:%02d:%02d:%03d\n"),
      KeTiSys.wHour, KeTiSys.wMinute, KeTiSys.wSecond,
      KeTiSys.wMilliseconds);
} else {
```

```
        CreateFile (ProcFile [iProc].TempFile,
          GENERIC_WRITE,
          FILE_SHARE_READ | FILE_SHARE_WRITE, &StdOutSA,
          CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
      StartUpSearch.dwFlags = STARTF_USESTDHANDLES;
      StartUpSearch.hStdOutput = hTempFile;
      StartUpSearch.hStdError = hTempFile;
      StartUpSearch.hStdInput = GetStdHandle (STD_INPUT_HANDLE);

      /* Create a process to execute the command line. */
      CreateProcess (NULL, CommandLine, NULL, NULL,
          TRUE, 0, NULL, NULL, &StartUpSearch, &ProcessInfo);
      /* Close unwanted handles. */
      CloseHandle (hTempFile); CloseHandle (ProcessInfo.hThread);
      hProc [iProc] = ProcessInfo.hProcess;
    }

    /* Processes are all running. Wait for them to complete. */
    for (iProc = 0; iProc < argc - 2; iProc += MAXIMUM_WAIT_OBJECTS)
      WaitForMultipleObjects ( /* Allows a large # of processes */
          min (MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc),
          &hProc [iProc], TRUE, INFINITE);
    /* Result files sent to std output using "cat." */
    for (iProc = 0; iProc < argc - 2; iProc++) {
      if (GetExitCodeProcess(hProc [iProc], &ExCode) && ExCode==0) {
        /* Pattern was detected -- List results. */
        if (argc > 3) _tprintf (_T ("%s:\n"), argv [iProc + 2]);
        fflush (stdout); /* Multiple processes use stdout. */
        _stprintf (CommandLine, _T ("%s%s"),
            _T ("cat "), ProcFile [iProc].TempFile);
        CreateProcess (NULL, CommandLine, NULL, NULL,
            TRUE, 0, NULL, NULL, &StartUp, &ProcessInfo);
        WaitForSingleObject (ProcessInfo.hProcess, INFINITE);
        CloseHandle (ProcessInfo.hProcess);
```

```
DWORD FsLow, nRead, LowRecNo, nRecTh, NPr, ThId, iTh;
BOOL NoPrint;
int iFF, iNP;
PTHREADARG ThArg;
LPTSTR StringEnd;

iNP = Options (argc, argv, _T ("n"), &NoPrint, NULL);
iFF = iNP + 1;
NPr = _ttoi (argv [iNP]); /* Number of threads. */
hFile = CreateFile (argv [iFF], GENERIC_READ | GENERIC_WRITE,
      0, NULL, OPEN_EXISTING, 0, NULL);
FsLow = GetFileSize (hFile, NULL);
nRec = FsLow / RECSIZE; /* Total number of records. */
nRecTh = nRec / NPr; /* Records per thread. */

/* Allocate thread args and handle array
   and space for the file. Read the complete file. */

ThArg = malloc (NPr * sizeof (THREADARG)); /* Thread args. */
ThreadHandle = malloc (NPr * sizeof (HANDLE));
pRecords = malloc (FsLow + sizeof (TCHAR));
ReadFile (hFile, pRecords, FsLow, &nRead, NULL);
CloseHandle (hFile);

LowRecNo = 0; /* Create the sorting threads. */
for (iTh = 0; iTh < NPr; iTh++) {
   ThArg [iTh].iTh = iTh;
   ThArg [iTh].LowRec = pRecords + LowRecNo;
   ThArg [iTh].HighRec = pRecords + (LowRecNo + nRecTh);
   LowRecNo += nRecTh;
   ThreadHandle [iTh] = (HANDLE) _beginthreadex (NULL, 0,
        ThSort, &ThArg [iTh], CREATE_SUSPENDED, &ThId);
}
```

```
    for (iTh = 0; iTh < NPr; iTh++) /* Run all sort threads. */
      ResumeThread (ThreadHandle [iTh]);
    WaitForSingleObject (ThreadHandle [0], INFINITE);
    for (iTh = 0; iTh < NPr; iTh++) CloseHandle (ThreadHandle [iTh]);

    StringEnd = (LPTSTR) pRecords + FsLow;
    *StringEnd = '\0';
    if (!NoPrint) printf ("\n%s", (LPCTSTR) pRecords);
    free (pRecords);
    free (ThArg);
    free (ThreadHandle);
    return 0;
  }  /* End of _tmain. */

  static VOID MergeArrays (LPRECORD, LPRECORD);
  DWORD WINAPI ThSort (PTHREADARG pThArg)
  {
    DWORD GrpSize = 2, RecsInGrp, MyNumber, TwoToI = 1;
    LPRECORD First;

    MyNumber = pThArg->iTh;
    First = pThArg->LowRec;
    RecsInGrp = pThArg->HighRec - First;
    qsort (First, RecsInGrp, RECSIZE, KeyCompare);
    while ((MyNumber % GrpSize) == 0 && RecsInGrp < nRec) {
            /* Merge with the adjacent sorted array. */
      WaitForSingleObject (
          ThreadHandle [MyNumber + TwoToI], INFINITE);
      MergeArrays (First, First + RecsInGrp);
      RecsInGrp *= 2;
      GrpSize *= 2;
      TwoToI *= 2;
    }
    _endthreadex (0);
```

```
{
  DWORD i, Context = 1;
  /* Set the current directory and open a log file, appending to
    an existing file. */

  /* Set all server status data members. */
  hServStatus.dwServiceType = SERVICE_WIN32_OWN_PROCESS;
  hServStatus.dwCurrentState = SERVICE_START_PENDING;
  hServStatus.dwControlsAccepted = SERVICE_ACCEPT_STOP |
      SERVICE_ACCEPT_SHUTDOWN |
SERVICE_ACCEPT_PAUSE_CONTINUE;
  hServStatus.dwWin32ExitCode = ERROR_SERVICE_SPECIF0C_ERROR;
  hServStatus.dwServiceSpecificExitCode = 0;
  hServStatus.dwCheckPoint = 0;
  hServStatus.dwWaitHint = 2 * CS_TIMEOUT;

  hSStat = RegisterServiceCtrlHandlerEx (ServiceName,
    ServerCtrlHandler, &Context);
  SetServiceStatus (hSStat, &hServStatus);

  /* Start service-specific work; generic work is complete. */
  if (ServiceSpecific (argc, argv) != 0) {
    hServStatus.dwCurrentState = SERVICE_STOPPED;
    hServStatus.dwServiceSpecificExitCode = 1;
        /* Server initialization failed. */
    SetServiceStatus (hSStat, &hServStatus);
    return;
  }
  /* We will only return here when the ServiceSpecific function
    completes, indicating system shutdown. */
  UpdateStatus (SERVICE_STOPPED, 0);
  return;
}
```

```
void UpdateStatus (int NewStatus, int Check)
/* Set a new service status and checkpoint --
   either specific value or increment. */
{
  if (Check < 0) hServStatus.dwCheckPoint++;
  else hServStatus.dwCheckPoint = Check;
  if (NewStatus >= 0) hServStatus.dwCurrentState = NewStatus;
  SetServiceStatus (hSStat, &hServStatus);
  return;
}

/* Control handler function, invoked by the SCM to run */
/* in the same thread as the main program. */
/* The last three parameters are not used, and the pre-NT5 */
/* handlers would also work in this example. */
VOID WINAPI ServerCtrlHandlerEx (DWORD Control, DWORD EventType,
        LPVOID lpEventData, LPVOID lpContext)
{
  swsitch (Control) {
    case SERVICE_CONTROL_SHUTDOWN:
    case SERVICE_CONTROL_STOP:
      ShutDown = TRUE; /* Set the global shutdown flag. */
      UpdateStatus (SERVICE_STOP_PENDING, -1);
      break;
    case SERVICE_CONTROL_PAUSE:
      PauseFlag = TRUE; /* Interrogated periodically. */
      break;
    case SERVICE_CONTROL_CONTINUE:
      PauseFlag = FALSE;
      break;
    case SERVICE_CONTROL_INTERROGATE:
      break;
    default:
      if (Control > 127 && Control < 256) /* User defined. */
```

```
  DWORD i, LocArgc; /* Local argc. */
  TCHAR argstr [MAX_ARG] [MAX_COMMAND_LINE];
  LPTSTR pArgs [MAX_ARG];

  /* Prepare the local "argv" array as pointers to strings. */
  for (i = 0; i < MAX_ARG; i++) pArgs [i] = argstr [i];

  /* Open the SC Control Manager on the local machine. */
  hScm = OpenSCManager (NULL, NULL, SC_MANAGER_ALL_ACCESS);

  /* Main command processing loop. */
  _tprintf (_T ("\nWindows Service Management"));
  while (!Exit) {
    _tprintf (_T ("\nSM$"));
    _fgetts (Command, MAX_COMMAND_LINE, stdin);
    ... Similar to JobShell ...
    if (_tcscmp (argstr [0], _T ("create")) == 0) {
      Create (LocArgc, pArgs, Command);
    }
    ... Similarly for all commands ...
  }
  CloseServiceHandle (hScm);
  return 0;
}

int Create (int argc, LPTSTR argv [], LPTSTR Command)
{
  /* Create a new service as a "demand start" service:
    argv [1]: service Name
    argv [2]: display Name
    argv [3]: binary executable */
  SC_HANDLE hSc;
  TCHAR CurrentDir [MAX_PATH + 1], Executable [MAX_PATH + 1];
```

```
  hSc = CreateService (hScm, argv [1], argv [2],
      SERVICE_ALL_ACCESS, SERVICE_WIN32_OWN_PROCESS,
      SERVICE_DEMAND_START, SERVICE_ERROR_NORMAL,
      Executable, NULL, NULL, NULL, NULL, NULL);
  return 0;
}

/* Delete a service -- argv [1]: ServiceName to delete. */

int Delete (int argc, LPTSTR argv [], LPTSTR Command)
{
  SC_HANDLE hSc;
  hSc = OpenService (hScm, argv [1], DELETE);
  DeleteService (hSc);
  CloseServiceHandle (hSc);
  return 0;
}

/* Start a named service -- argv [1]: service name to start. */
int Start (int argc, LPTSTR argv [], LPTSTR Command)
{
  SC_HANDLE hSc;
  TCHAR WorkingDir [MAX_PATH + 1];
  LPTSTR pWorkingDir = WorkingDir;
  LPTSTR argvStart [] = {argv [1], WorkingDir};

  GetCurrentDirectory (MAX_PATH + 1, WorkingDir);
  hSc = OpenService(hScm, argv [1], SERVICE_ALL_ACCESS);
  /* Start the service with one arg, the working directory. */
  /* Note: The service name agrees, by default, with the name */
  /*    associated with the handle, hSc, by OpenService. */
  /*    But, the ServiceMain function does not verify this. */
  StartService (hSc, 2, argvStart);
  CloseServiceHandle (hSc);
```

```
    return 0;
  }

  /* Control a named service. argv [1]: service name to control.
    argv [2]: Control command: stop, pause, resume, interrogate. */
    static LPCTSTR Commands [] =
      {"stop," "pause," "resume," "interrogate," "user"};
    static DWORD Controls [] = {
      SERVICE_CONTROL_STOP, SERVICE_CONTROL_PAUSE,
      SERVICE_CONTROL_CONTINUE,     SERVICE_CONTROL_INTERROGATE,
128};

  int Control (int argc, LPTSTR argv [], LPTSTR Command)
  {
    SC_HANDLE hSc;
    SERVICE_STATUS ServiceStatus;
    DWORD dwControl, i;
    BOOL Found = FALSE;

    for (i= 0; i < sizeof (Controls)/sizeof (DWORD) && !Found; i++)
      Found = (_tcscmp (Commands [i], argv [2]) == 0);
    if (!Found) {
      _tprintf (_T ("\nIllegal Control Command %s"), argv [1]);
      return 1;
    }
    dwControl = Controls [i - 1];
    hSc = OpenService(hScm, argv [1],
      SERVICE_INTERROGATE | SERVICE_PAUSE_CONTINUE |
      SERVICE_STOP | SERVICE_USER_DEFINED_CONTROL |
      SERVICE_QUERY_STATUS);
    ControlService (hSc, dwControl, &ServiceStatus);

    if (dwControl == SERVICE_CONTROL_INTERROGATE) {
      QueryServiceStatus (hSc, &ServiceStatus);
```

```
/* Server's socket address structure. */
struct sockaddr_in ConnectSAddr; /* Connected socket. */
WSADATA WSStartData; /* Socket library data structure. */

typedef struct SERVER_ARG_TAG { /* Server thread arguments. */
  volatile DWORD number;
  volatile SOCKET sock;
  volatile DWORD status;
  /* Explained in main thread comments. */
  volatile HANDLE srv_thd;
  HINSTANCE dlhandle; /* Shared library handle. */
} SERVER_ARG;
volatile static ShutFlag = FALSE;
static SOCKET SrvSock, ConnectSock;

int _tmain (DWORD argc, LPCTSTR argv [])
{
  /* Server listening and connected sockets. */
  BOOL Done = FALSE;
  DWORD ith, tstatus, ThId;
  SERVER_ARG srv_arg [MAX_CLIENTS];
  HANDLE hAcceptTh = NULL;
  HINSTANCE hDll = NULL;

  /* Initialize the WSA library, Ver 2.0, although 1.1 will work. */
  WSAStartup (MAKEWORD (2, 0), &WSStartData);

  /* Open command library DLL if specified on command line. */
  if (argc > 1) hDll = LoadLibrary (argv [1]);
  /* Initialize thread arg array. */
  for (ith = 0; ith < MAX_CLIENTS; ith++) {
    srv_arg [ith].number = ith;
    srv_arg [ith].status = 0; srv_arg [ith].sock = 0;
    srv_arg [ith].dlhandle = hDll; srv_arg [ith].srv_thd = NULL;
```

```
}
/* Follow standard server socket/bind/listen/accept sequence. */
SrvSock = socket (AF_INET, SOCK_STREAM, 0);
SrvSAddr.sin_family = AF_INET;
SrvSAddr.sin_addr.s_addr = htonl ( INADDR_ANY );
SrvSAddr.sin_port = htons ( SERVER_PORT );
bind (SrvSock, (struct sockaddr *) &SrvSAddr,
    sizeof SrvSAddr);
listen (SrvSock, MAX_CLIENTS);

/* Main thread becomes listening/connecting/monitoring thread. */
/* Find an empty slot in the server thread arg array. */
/* status values: 0 -- slot is free; 1 -- thread stopped;
  2 -- thread running; 3 -- stop entire system. */
while (!ShutFlag) {
  for (ith = 0; ith < MAX_CLIENTS && !ShutFlag; ) {
    if (srv_arg [ith].status==1 || srv_arg [ith].status==3) {
      /* Thread stopped, normally or by shutdown request. */
      WaitForSingleObject (srv_arg[ith].srv_thd INFINITE);
      CloseHandle (srv_arg[ith].srv_thd);
      if (srv_arg [ith].status == 3) ShutFlag = TRUE;
      else srv_arg [ith].status = 0;
            /* Free thread slot. */
    }
    if (srv_arg [ith].status == 0 || ShutFlag) break;
    ith = (ith + 1) % MAX_CLIENTS;
    if (ith == 0) Sleep (1000);
    /* Break the polling loop. */
    /* Alternative: use an event to signal a free slot. */
  }
  /* Wait for a connection on this socket. */
  /* Separate thread so we can poll the ShutFlag flag. */
  hAcceptTh = (HANDLE)_beginthreadex (NULL, 0, AcceptTh,
      &srv_arg [ith], 0, &ThId);
```

```
    while (!ShutFlag) {
      tstatus = WaitForSingleObject (hAcceptTh, CS_TIMEOUT);
      if (tstatus == WAIT_OBJECT_0) break;
      /* Connection made. */
    }
    CloseHandle (hAcceptTh);
    hAcceptTh = NULL; /* Prepare for next connection. */
  }

  _tprintf (_T ("Server shutdown. Wait for all srvr threads\n"));
  /* Terminate the accept thread if it is still running.
   * See the Web site for more detail on this shutdown logic. */
  if (hDll != NULL) FreeLibrary (hDll);
  if (hAcceptTh != NULL) TerminateThread (hAcceptTh, 0);
  /* Wait for any active server threads to terminate. */
  for (ith = 0; ith < MAX_CLIENTS; ith++)
    if (srv_arg [ith].status != 0) {
      WaitForSingleObject (srv_arg[ith].srv_thd, INFINITE);
      CloseHandle (srv_arg[ith].srv_thd);
    }
  shutdown (SrvSock, 2);
  closesocket (SrvSock);
  WSACleanup ();
  return 0;
}

static DWORD WINAPI AcceptTh (SERVER_ARG * pThArg)
{
  /* Accepting thread that allows the main thread to poll the */
  /* shutdown flag. This thread also creates the server thread. */
  LONG AddrLen, ThId;

  AddrLen = sizeof (ConnectSAddr);
  pThArg->sock = accept (SrvSock, /* This is a blocking call. */
```

```
      (struct sockaddr *) &ConnectSAddr, &AddrLen);
      /* A new connection. Create a server thread. */
    pThArg->status = 2;
    pThArg->srv_thd =
        (HANDLE) _beginthreadex (NULL, 0, Server, pThArg, 0, &ThId);
    return 0; /* Server thread remains running. */
}
static DWORD WINAPI Server (SERVER_ARG * pThArg)
/* Server thread function. Thread created on demand. */
{
    /* Each thread keeps its own request, response,
       and bookkeeping data structures on the stack. */
    /* ... Standard declarations from serverNP omitted ... */
    SOCKET ConnectSock;
    int Disconnect = 0, i;
    int (*dl_addr)(char *, char *);
    char *ws = " \0\t\n"; /* White space. */

    GetStartupInfo (&StartInfoCh);
    ConnectSock = pThArg->sock;
    /* Create a temp file name. */
    sprintf (TempFile, "%s%d%s", "ServerTemp",
        pThArg->number, ".tmp");

    while (!Done && !ShutFlag) { /* Main command loop. */
      Disconnect = ReceiveRequestMessage (&Request, ConnectSock);
      Done = Disconnect || (strcmp (Request.Record, "$Quit") == 0)
          || (strcmp (Request.Record, "$ShutDownServer") == 0);
      if (Done) continue;
      /* Stop this thread on "$Quit" or "$ShutDownServer". */
      hTmpFile = CreateFile (TempFile,
          GENERIC_READ | GENERIC_WRITE,
          FILE_SHARE_READ | FILE_SHARE_WRITE, &TempSA,
          CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
```

```
  /* Check for a DLL command. For simplicity, shared */
  /* library commands take precedence over process
    commands. First, extract the command name. */

  i = strcspn (Request.Record, ws); /* Length of token. */
  memcpy (sys_command, Request.Record, i);
  sys_command [i] = '\0';

  dl_addr = NULL; /* Will be set if GetProcAddress succeeds. */
  if (pThArg->dlhandle != NULL) { /* Try server "in process." */
    dl_addr = (int (*)(char *, char *))
        GetProcAddress (pThArg->dlhandle, sys_command);
    if (dl_addr != NULL) __try {
      /* Protect server process from exceptions in DLL. */
      (*dl_addr) (Request.Record, TempFile);
    } __except (EXCEPTION_EXECUTE_HANDLER {
      ReportError (_T ("Exception in DLL"), 0, FALSE);
    }
  }
}
  if (dl_addr == NULL) { /* No in-process support. */
  /* Create a process to carry out the command. */
  /* ... Same as in serverNP ... */
  }
  /* ... Same as in serverNP ... */

} /* End of main command loop. Get next command. */

/* End of command loop. Free resources; exit from the thread. */

_tprintf (_T ("Shutting down server# %d\n"), pThArg->number);
shutdown (ConnectSock, 2);
closesocket (ConnectSock);
```

```
/*   Initialize the WSA library, Ver 2.0, although 1.1 will work. */
WSAStartup (MAKEWORD (2, 0), &WSStartData);

/* Connect to the server. */
/* Follow the standard client socket/connect sequence. */
ClientSock = socket (AF_INET, SOCK_STREAM, 0);
memset (&ClientSAddr, 0, sizeof (ClientSAddr));
ClientSAddr.sin_family = AF_INET;
if (argc >= 2)
  ClientSAddr.sin_addr.s_addr = inet_addr (argv [1]);
else
  ClientSAddr.sin_addr.s_addr = inet_addr (DefaultIPAddr);
ClientSAddr.sin_port = htons (SERVER_PORT);
        /* Defined as 1070. */
connect (ClientSock,
  (struct sockaddr *) &ClientSAddr, sizeof (ClientSAddr));
/* Main loop to prompt user, send request, receive response. */
while (!Quit) {
  _tprintf (_T ("%s"), PromptMsg);
  /* Generic input, but command to server must be ASCII. */
  _fgetts (Req, MAX_RQRS_LEN-1, stdin);
  for (j = 0; j <= _tcslen (Req); j++)
    Request.Record [j] = Req [j];
  /* Get rid of the new line at the end. */
  Request.Record [strlen (Request.Record) - 1] = '\0';
  if (strcmp (Request.Record, QuitMsg) == 0 ||
      strcmp (Request.Record, ShutMsg) == 0) Quit = TRUE;
  SendRequestMessage (&Request, ClientSock);
  ReceiveResponseMessage (&Response, ClientSock);
}

shutdown (ClientSock, 2); /* Disallow sends and receives. */
closesocket (ClientSock);
WSACleanup ();
```

Chúng ta sẽ tạo một một thư viện liên kết động là edrlib .dll (Easy drawing routine). Hàm thư viện này sẽ chỉ chứa một hàm đơn giản để thực hiện công việc vẽ ra một xâu trong ứng dụng demo của chúng ta.

Để tạo ra một thư viện liên kết động chúng ta cần có một cách tiếp cận khác s o với cách mà chúng ta vẫn dùng để viết các ứng dụng. VC++ phân biệt giữa các khái niệm "workspace" và "project". Một project thường là một ứng dụng hoặc một thư viện liên kết động . Một workspace có thể gồm nhiều project . Cho đến thời điểm này chúng ta mới chỉ viết các workspace chỉ có 1 project. Trong phần này chúng ta sẽ tạo một workspace có hai project , một cho việc tạo dll và một cho việc gọi tới file dll đó.

Các bước tạo ứng dụng với Visual Studio .NET 2003 như sau:

1. Tạo 1 Solution (Dll1 chẳng hạn) rỗng (Blank Solution)
2. Thêm một project chứa file dll cho ứng dụng (chọn New → Project → Win32 Project) và gõ tên của Project là SimpleDll.
3. Trong hộp thoại Application Setting chọn DLL (Application Type) và chọn mục Empty.
4. Thêm file mã nguồn cho Project và gõ nội dung của file dll vào:

```
/*----------------------
  simpledll.h header file

  ----------------------*/

#ifdef __cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif

EXPORT BOOL CALLBACK EdrCenterTextA (HDC, PRECT, PCSTR) ;
EXPORT BOOL CALLBACK EdrCenterTextW (HDC, PRECT, PCWSTR) ;

#ifdef UNICODE
#define EdrCenterText EdrCenterTextW
#else
#define EdrCenterText EdrCenterTextA
#endif
```

```
/*--------------------------------------------
  simpledll.c -- Easy Drawing Routine Library module

        (c) Charles Petzold, 1998
 --------------------------------------------*/

#include <windows.h>
#include "simpledll.h"

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, PVOID
pvReserved)
{
    return TRUE ;
}

EXPORT BOOL CALLBACK EdrCenterTextA (HDC hdc, PRECT prc, PCSTR
pString)
{
    int  iLength ;
    SIZE size ;

    iLength = lstrlenA (pString) ;
    GetTextExtentPoint32A (hdc, pString, iLength, &size) ;
    return TextOutA (hdc, (prc->right - prc->left - size.cx) / 2,
                     (prc->bottom - prc->top - size.cy) / 2,
                pString, iLength) ;
}

EXPORT BOOL CALLBACK EdrCenterTextW (HDC hdc, PRECT prc, PCWSTR
pString)
{
    int  iLength ;
    SIZE size ;
```

```
    iLength = lstrlenW (pString) ;
    GetTextExtentPoint32W (hdc, pString, iLength, &size) ;
    return TextOutW (hdc, (prc->right - prc->left - size.cx) / 2,
                          (prc->bottom - prc->top - size.cy) / 2,
                     pString, iLength) ;
}
```

5. Tạo một Project Win 32 Project, sau đó chọn kiểu Project là Application, tạo file .c cho ứng dụng và gõ nội dung ứng dụng vào như sau:

```
#include <windows.h>
#include "..\\SimpleDll\\simpledll.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR szAppName[] = TEXT ("StrProg") ;
    HWND         hwnd ;
    MSG          msg ;
    WNDCLASS     wndclass ;

    wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc   = WndProc ;
    wndclass.cbClsExtra    = 0 ;
    wndclass.cbWndExtra    = 0 ;
    wndclass.hInstance     = hInstance ;
    wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName  = NULL ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
```

```
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("DLL Demonstration Program"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
    HDC         hdc ;

    PAINTSTRUCT ps ;
    RECT        rect ;

    switch (message)
    {
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
                    GetClientRect (hwnd, &rect) ;
        EdrCenterText (hdc, &rect,
                       TEXT ("This string was displayed by a DLL")) ;
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
    }
```

6. Đặt chế độ dịch là Project thứ hai phụ thuộc (dependencies) vào Project thứ nhất và thư mục Output cùng tạm của hai Project là ../Release.

7. Dịch và chạy chương trình.

Ở đây chúng ta có hai phiên bản của một hàm trong thư viện liên kết động được sử dụng, điều này cho phép dùng các hàm có hỗ trợ Unicode trong trường hợp hệ thống có hỗ trợ và ngược lại sử dụng một hàm không có Unicode , thường tên của hàm sẽ có thêm chữ W nếu có hỗ trợ Unicode và A nếu không.

Đồng thời chúng ta cũng t hấy trong mã của thư viện có một hàm DllMain , hàm này có vai trò tương tự như hàm WinMain trong một chương trình . Tác dụng của hàm DllMain là khởi tạo và thu hồi bộ nhớ và những thứ liên quan khác , chúng ta sẽ bàn tới vấn đề này ở cuối chương, hiện tại chỉ cần return TRUE là ổn.

Điều bí ẩn còn lại có lẽ là ở định danh EXPORT . Các hàm trong một thư viện liên kết động được sử dụng bởi các ứng dụng khác phải được xuất khẩu . Điều này không liên quan tới các vấn đề thương mại thông thường mà chỉ là một chỉ thị đảm bảo trình biên dịch sẽ thêm tên hàm vào thư viện import simpledll .lib để trình liên kết có thể đưa các thông tin phù hợp vào chương trình (file *.exe) để có thể nạp các thư viện dll khi chương trình chạy . Định danh EXPORT còn bao gồm chỉ định lớp chứa __declspec(dllexport) và một chỉ thị tiền xử lý if extern “C” để đề phòng trường hợp file header được biên dịch theo kiểu C++. Điều này ngăn chặn trình biên dịch khỏi các lỗi trùng tên của các hàm C ++ và cho phép các thư viện liên kết động có thể được sử dụng bởi cả các chương trình C và C++.

Điểm vào và điểm thoát của thư viện (Entry and Exit Point)

Hàm DllMain được gọi đến khi thư viện liên kết động lần đầu tiên được nạp vào bộ nhớ để thực hiện và khi nó kết thúc nhiệm vụ (bị loại khỏi bộ nhớ). Tham số đầu tiên của hàm DllMain là handle tới instance của th ư viện. Nếu như thư viện có sử dụng các tài nguyên đòi hỏi một handle instance (chẳng hạn như các hộp thoại), chúng ta nên lưu lại hInstance vào một biến toàn cục . Tham số cuối cùng của hàm DllMain được dữ trự dành cho hệ t hống sử dụng.

Tham số fdwReason có thể là một trong 4 giá trị chỉ ra tại sao Windows lại gọi tới hàm DllMain. Trong các mục tiếp theo chúng ta nên nhớ rằng một chương trình đơn có thể được

nạp nhiều lần và chạy đồng thời t rên Windows. Mỗi lần chương trình được nạp nó được xem như là một tiến trình (process) riêng rẽ.

Giá trị của tham số fdwReason bằng DLL _PROCESS_ATTACH chỉ ra rằng thư viện liên kết động đã được ánh xạ vào vùng địa chỉ của mộ t tiến trình. Đây là một đầu mối cho phép thư viện thực hiện bất cứ khởi tạo nào đòi hỏi được phục vụ cho các yêu cầu tiếp theo của tiến trình. Các khởi tạo kiểu này có thể là cấp phát bộ nhớ chẳng hạn. Trong thời gian tiến trình đang chạy, DllMain được gọi với một tham số DLL _PROCESS_ATTACH chỉ một lần trong cả thời gian tồn tại của process đó . Bất cứ một tiến trình nào khác sử dụng cùng file DLL sẽ gọi đến hàm DllMain với một giá trị tham số DLL_PROCESS_ATTACH.

Nếu như việc khởi tạo là thành công DllMain sẽ trả về một giá trị khác 0, giá trị trả về là 0 sẽ làm cho Windows không chạy chương trình.

Nếu giá trị của fdwReason bằng DLL _PROCESS_DETACH thì có nghĩa là chương trình không cần file DLL nữa, và đây là một cơ hội để thư viện thực hiện các công việc dọn dẹp của nó. Trên các hệ điều hành 32 bit của Windows điều này không thực sự cần thiết nhưng là một thói quen lập trình tốt.

Tương tự khi hàm DllMain được gọi với một giá trị tham số là DLL_THREAD_ATTACH thì có nghĩa là một tiến trình sử dụng thư viện đã tạo ra một luồng (thread) mới. Khi luồng kết thúc Windows lại gọi tới hàm DllMain với tham số là DLL_THREAD_DETACH. Cũng có thể xảy ra trường hợp Windows thực hiện lời gọi tới hàm DllMain với giá trị của tham số fdwReason bằng DLL _THREAD_DETACH mà không thực hiện lời gọi với giá trị DLL _THREAD_ATTACH trước đó nếu như thư viện liên kết động được gắn với một tiến trình sau khi luồng đã được tạo ra.

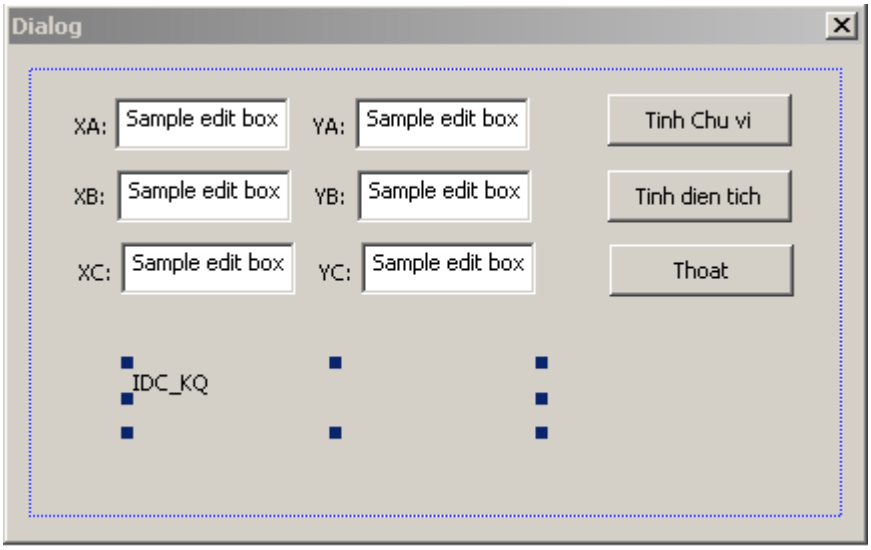
Luồng vẫn tồn tại khi hàm DllMain được gọi đến với tham số DLL_THREAD_DETACH. Nó thậm chí có thể gửi các thông điệp trong tiến trì nh. Nhưng các luồng không nên sử dụng hàm PostMessage () vì luồng có thể kết thúc trước khi thông điệp đến đích.

Chương trình để test thư viện liên kết động là một chương trình đơn giản và là một Project khác (thuộc loại Win 32 Application). Chúng ta có thể để các file của 2 Project vào cùng một thư mục hoặc riêng rẽ trong 2 thư mục.

Trong quá trình dịch chương trình file simpledll .dll và simpledll .lib sẽ được sinh ra trước, file simpledll .lib sẽ được tự động liên kết với chương trình thử nghiệm và file simpledll.dll sẽ được nạp vào bộ nhớ khi chương trình chạy . Cần chú ý là chương trình usedll.exe không chứa mã của hàm sử dụng trong thư viện, mã của hàm chỉ được nạp vào b ộ nhớ khi chương trình chạy.

Việc include file simpledll .h cũng giống như chúng ta include file windows .h, liên kết với file simpledll .lib cũng tương tự như liên kết với file user 32.lib và liên kết với file simpledll.dll cũng giống như chương trình liên kết với file user32.dll.

Mặc dù chúng ta xếp một file DLL là một mở rộng của Windows nhưng nó cũng là một mở rộng của chương trình ứng dụng của chúng ta . Tất cả những gì file DLL thực hiện đều là thay mặt cho ứng dụng sử dụng nó. Chẳng hạn tất cả các thao tác cấp phát bộ nhớ được kiểm soát bởi chương trình. Bất cứ cửa sổ nào nó tạo ra đều sở hữu bởi chương trình và bất cứ file nào được mở cũng được kiểm soát bởi ch ương trình. Nhiều chương trình có thể sử dụng cùng



Các điều khiển XA , YA, XB, YB, XC, YC là tọa độ 3 đỉnh trên mặt phẳng tọa độ (nguyên). Khi nhấn nút "Tinh chu vi" hãy tính chu vi của tam giác tạo thành bởi 3 đỉnh A, B, C và hiển thị lên IDC _KQ. Nhấn "Thoat" để thóat khỏi hộp thoại và nhấn "Tinh dien tich" sẽ tính diện tích và hiển thị như trong phần tính chu vi

Bài số 2

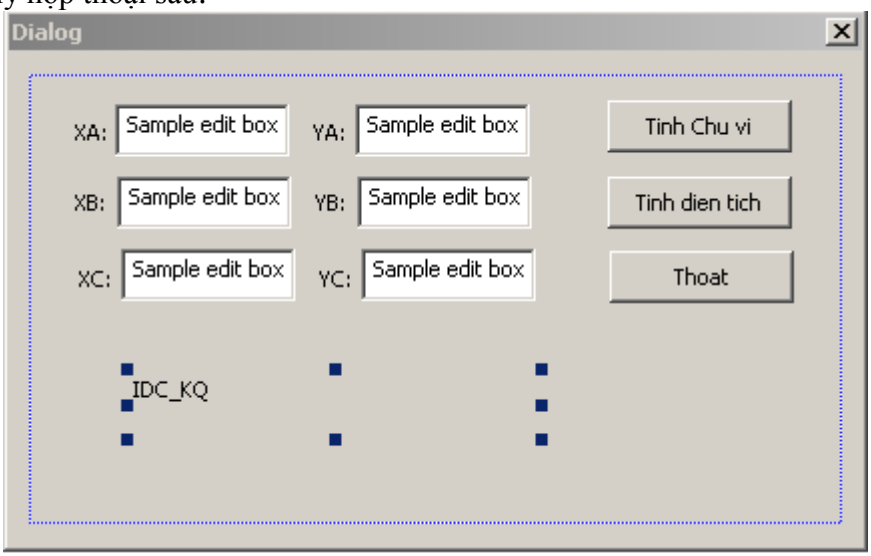
a) Hãy trình bày (đưa ra) dạng đơn giản nhất của một hàm xử lý thông điệp cửa sổ ch o một hộp thoại.

b) Hãy viết hàm WndProc cho một chương trình có hệ thống menu gồm 1 mục File , trong đó có các mục con với các chức năng sau : Menu1, Menu2, khi người dùng nhấn vào các mục này chỉ cần đưa ra thông báo đơn giản , mục Exit để thoát khỏi chương trình.

Đề số 3:

Bài số 1

Viết hàm xử lý hộp thoại sau:



Các điều khiển XA , YA, XB, YB, XC, YC là tọa độ 3 đỉnh trên mặt phẳng tọa độ (nguyên). Khi nhấn nút "Tinh chu vi" hãy tính chu vi của tam giác tạo thành bởi 3 đỉnh A, B, C và hiển thị lên IDC _KQ. Nhấn "Thoat" để thóat khỏi hộp thoại và nhấn "Tinh dien tich" sẽ tính diện tích và hiển thị như trong phần tính chu vi

Bài số 2

a) Hãy trình bày (đưa ra) dạng đơn giản nhất của một hàm xử lý thông điệp cửa sổ cho một hộp thoại.

b) Hãy viết hàm WndProc cho một chương trình có hệ thống menu gồm 1 mục File, trong đó có các mục con với các chức năng sau: Menu1, Menu2, khi người dùng nhấn vào các mục này chỉ cần đưa ra thông báo đơn giản, mục Exit để thoát khỏi chương trình.

Bài giảng môn lập trình Windows 3

BÀI 1: TUYẾN TRÌNH.

Mục tiêu của bài:

Nhằm trang bị cho người học:

- Kiến thức và kỹ năng lập trình với các tuyến trình.
- Kiến thức và kỹ năng tạo các chương trình đa tuyến trình.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

1. Ứng dụng đa tuyến trình:

Xử lý đa tuyến- Khái niệm:

Những hệ điều hành như Windows và Linux là những hệ thống đa nhiệm cho phép bạn chạy nhiều chương trình cùng lúc. Như bạn biết, một chương trình đơn giản là một danh sách các chỉ thị lệnh **CPU** để thực thi một nhiệm vụ đặc biệt.

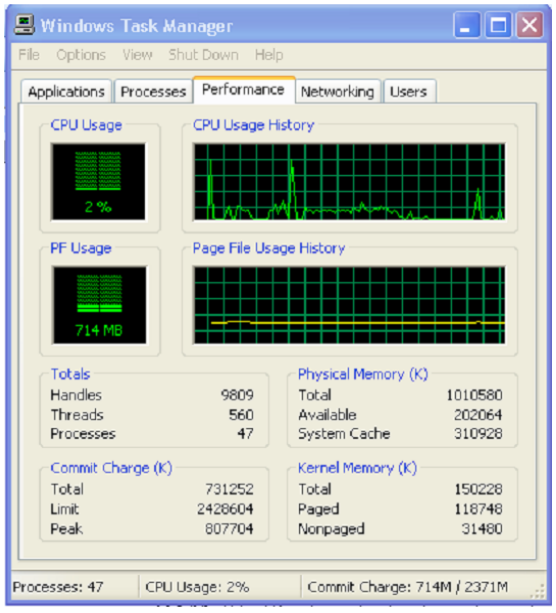
Trước khi **CPU** có thể thực thi một chương trình, chỉ thị lệnh phải cư trú bên trong bộ nhớ truy cập ngẫu nhiên của máy tính(**RAM**).

Để chạy nhiều chương trình, hệ điều hành tải chỉ lệnh của từng chương trình vào **RAM**. Khi Windows tải hai hay nhiều chương trình vào trong kí ức, Windows phải có cách theo dõi và quản lý kí ức(và những thành phần khác như file nào đang mở, chương trình đang thực hiện nhiệm vụ ở mặt tiền hay mặt hậu...). Windows theo dõi những tài nguyên của mỗi chương trình bằng cách gán từng chương trình cho một tiến trình quản lý duy nhất.

Mỗi khi bạn chạy một chương trình, Windows tạo ra một tiến trình mới để giữ thông tin tài nguyên và trạng thái của chương trình. Bên trong một PC, CPU thực thi từng chỉ thị lệnh của mỗi chương trình.

Ta thấy **PC** có vẻ thực thi các chương trình cùng một lúc nhưng thực ra **CPU** chỉ có thể thực thi mỗi lúc chỉ một chỉ thị lệnh.

Ta có thể theo dõi Windows trao đổi **CPU** giữa những tiến trình và thời gian **CPU** mà mỗi tiến trình đã tiêu thụ trong **Windows Task Manager** khi nhấn **CTRL+ALT+DELETE** :



Windows là một hệ điều hành đa nhiệm được ưu tiên, có nghĩa là chính Windows xác định khi nào thì một tiến trình sẽ bắt đầu tới thời gian thực thi chiếm lĩnh CPU và khi nào thời gian thực thi sẽ chấm dứt.

Windows không quan tâm đến các chỉ thị lệnh mà tiến trình thực thi khi nó chuyển điều khiển của CPU cho tiến trình.

Để theo dõi mỗi tiến trình về trạng thái thực thi(như nội dung con trỏ và thanh ghi), Windows gán thông tin cho đối tượng tiến trình và lưu lại trước khi nó bắt đầu thực thi tiến trình tiếp theo để có thể sẵn sàng thực thi lại tiến trình trong lần tới. Lúc đó Windows sẽ khôi phục thông tin về tiến trình cùng những chỉ thị CPU.

Lập trình viên thường gọi tập chỉ thị thực thi chương trình là một thread.

Ví dụ: trong xử lý văn bản ứng dụng có thể dùng một thread kiểm tra chính tả, còn một thread khác kiểm soát việc gõ văn bản. Để cho chương trình có thể xử lý thao tác cùng lúc bạn có thể tạo ra nhiều thread trong một chương trình.

Khi một chương trình sử dụng nhiều thread , Windows sẽ điều khiển thời gian chuyển đổi CPU của các thread trong một chương trình.

2. Thao tác với các tuyến trình(xử lý Thread):

2.1. Tạo và chạy nhiều thread:

*** Khái niệm:** Một thread là một chuỗi liên tiếp những sự thực thi trong chương trình.

Thread tương ứng với một đơn vị thực thi, để sử dụng thread trong một chương trình C# trước hết chúng ta tạo ra một đối tượng thread cho mỗi thread mà chúng ta lập kế hoạch sử dụng. Thread được thao tác bằng cách dùng lớp Thread nằm trong namespaces System.threading, một thể hiện của thread đại diện cho một thread, ta có thể tạo các thread khác bằng cách khởi tạo một đối tượng thread.

Một thread thực thi một tập các chỉ thị lệnh trong chương trình của chúng ta. Khi chúng ta tạo ra một thể hiện của thread chúng ta phải chỉ rõ địa chỉ những phát biểu lệnh đầu tiên mã thread sẽ thực thi và thường là địa chỉ của một chương trình con hay thủ tục.

Ví dụ:Tạo một đối tượng thread như sau:

```
Thread A = new Thread(entryPoint);
```

Đoạn mã trên biểu diễn một hàm dựng của Thread một thông số chỉ định điểm nhập của một Thread, đó là phương thức nơi Thread bắt đầu thi hành.trong tình huống này ta dùng thông số là delegate, một delegate đã được định nghĩa trong System.Threading gọi là ThreadStart, chữ kí của nó như sau:

```
Public delegate void ThreadStart();
```

Thông số ta truyền cho hàm dựng phải là một delegate kiểu này.

Ta bắt đầu thread bằng cách gọi phương thức Thread.Start(), giả sử rằng ta có phương thức Display() như sau:

```
static void Display()
{
   for (int i = 0; i < 100; i++)
   {
      Console.Write("A");
   }
}
```

Vậy sắp xếp lại ta có đoạn mã sau đây:

```
        ThreadStart entryPoint = new ThreadStart(Display);
        Thread A = new Thread(entryPoint);
        A.Name = "Thread Demo";
        A.Start();
```

Trong mã này ta đăng kí tên cho thread bằng cách dùng thuộc tính Thread.Name không cần thiết làm điều này nhưng nó có thể hữu ích.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo1
{
   class Program
   {
      static void Display()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
         }
      }
      static void Main(string[] args)
      {
         ThreadStart entryPoint = new ThreadStart(Display);
         Thread A = new Thread(entryPoint);
         A.Name = " Thread Demo ";
         A.Start();
      }
   }
}
```

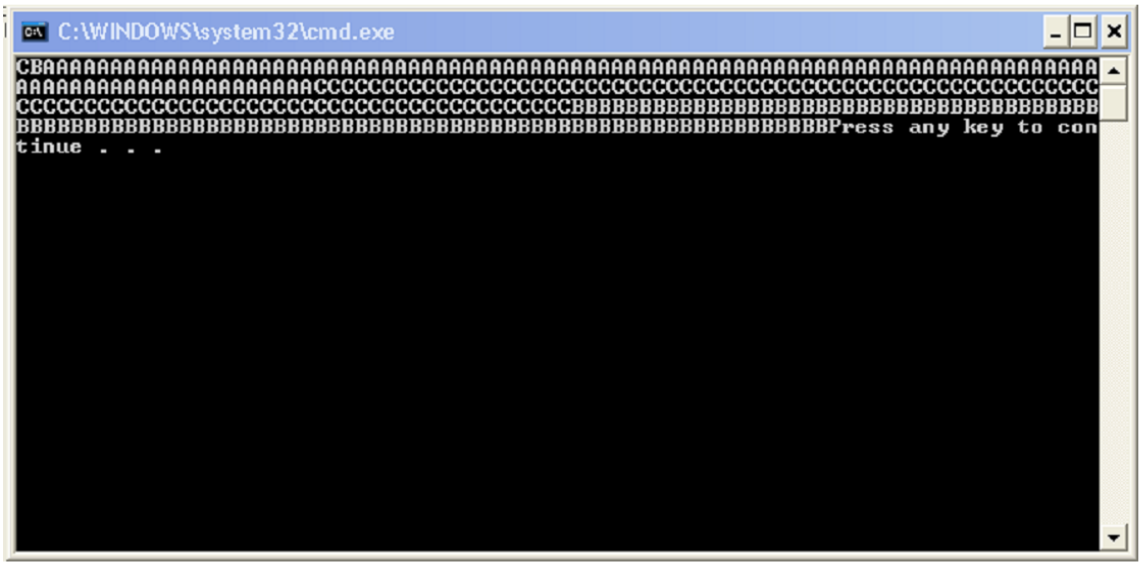
Ở chương trình trên để khởi động thực thi Thread, mã chương trình phải gọi phương thức Start() của lớp Thread.

Tương tự như trên nhưng giờ ta tạo và chạy ba Thread A,B,C như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo2
{
   class Program
   {
      static void DisplayA()
      {
         for (int i = 0; i < 100; i++)
```

```
        {
            Console.Write("A");
        }
    }
    static void DisplayB()
    {
        for (int i = 0; i < 100; i++)
        {
            Console.Write("B");
        }
    }
    static void DisplayC()
    {
        for (int i = 0; i < 100; i++)
        {
            Console.Write("C");
        }
    }
    static void Main(string[] args)
    {
      ThreadStart entryPointA = new ThreadStart(DisplayA);
      ThreadStart entryPointB = new ThreadStart(DisplayB);
      ThreadStart entryPointC = new ThreadStart(DisplayC);
      Thread A = new Thread(entryPointA);
      Thread B = new Thread(entryPointB);
      Thread C = new Thread(entryPointC);
       A.Name = " Thread Demo A";
       B.Name = " Thread Demo B";
       C.Name = " Thread Demo C";
       A.Start();
       B.Start();
       C.Start();
    }
  }
}
```

Sau khi chúng ta biên dịch và thực thi chương trình thì mà hình kết xuất như sau:



Như chúng ta cũng thấy trên hình thì các Thread chạy một cách tuần tự chứ không song hành. Trật tự này có thể thay đổi tùy vào thời điểm mà Windows điều phối các Thread.

2.2. Đặt Thread vào trạng thái chờ ngủ(sleep):

Nếu như chúng ta muốn đình chỉ sự thực thi của Thread trong một khoảng thời gian nào đó. Ví dụ chúng ta tạo Thread để theo dõi tài nguyên sử dụng của Server. Bên trong chương trình, chúng ta có thể sử dụng thời gian định kỳ để kiểm tra ký ức sẵn có, 1 giây để khảo sát không gian đĩa còn trống và 3 giây kiểm tra kết nối của người dùng từ xa đến Server. Tùy theo nhu cầu, chúng ta có thể muốn"đánh thức" thread và thực thi xử lý mỗi phút hoặc có thể là 10 phút một lần.

Để dừng Thread trong một khoảng thời gian, chúng ta có thể sử dụng phương thức Sleep do lớp thread cung cấp. Ví dụ: Phát biểu sau sẽ đặt dừng(hay ngủ) khoảng 7000(1 giây = 1000 mili-giây).

```
Thread.Sleep(7000);
```

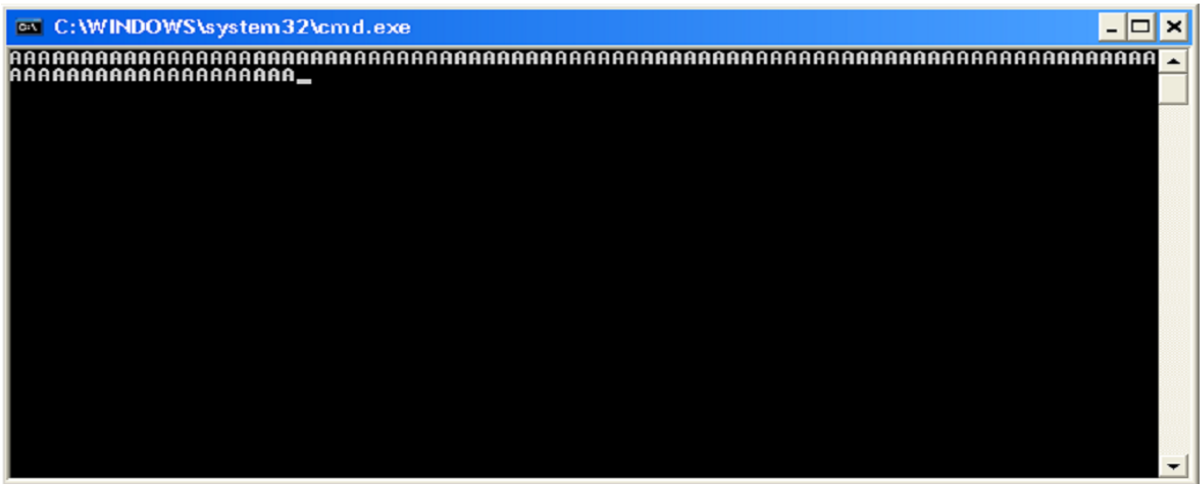
Khi chúng ta đặt Thread vào trạng thái ngủ bằng phương thức Sleep thì thread sẽ không thực thi và tiêu thụ tài nguyên **CPU.**

Chúng ta sẽ xét ví dụ tiếp sau đây để minh họa cho điều này:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo3
{
   class Program
   {
      static void Display()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
```

```
        }
    }
    static void Main(string[] args)
    {
        ThreadStart entryPoint = new ThreadStart(Display);
        Thread A = new Thread(entryPoint);
        A.Name = " Thread Demo ";
        A.Start();
       Thread.Sleep(7000);
    }
  }
}
```

Khi biên dịch và thực thi chương trình màn hình sẽ hiển thị kết xuất sau:

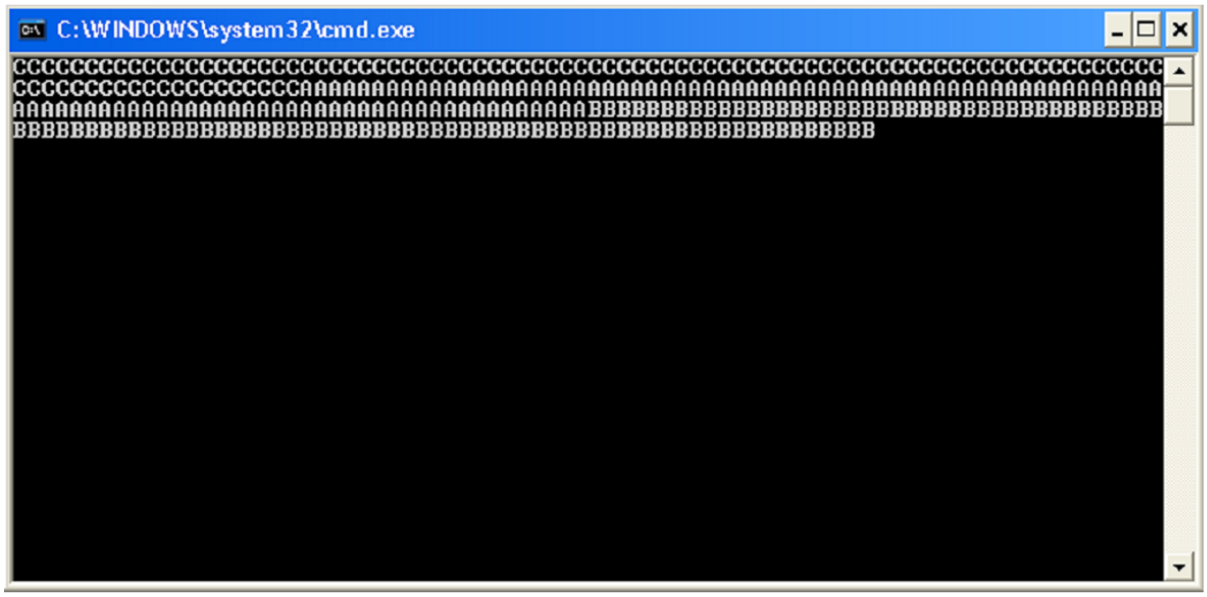


Bây giờ ta tạo ra một chương trình như sau: Trong chương trình ta sẽ tạo và khởi động 3 Thread. Sau đó sẽ sử dụng phương thức Sleep để đình chỉ một Thread trong vòng 0.3 giây, 0.1 giây và 0.8 giây.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo4
{
  class Program
  {
    static void DisplayA()
    {
            //Dừng Thread trong vòng 0.3 giây.
            Thread.Sleep(300);
      for (int i = 0; i < 100; i++)
      {
        Console.Write("A");
```

```
        }
    }
    static void DisplayB()
    {
            //Dừng Thread trong vòng 0.8 giây.
            Thread.Sleep(800);
        for (int i = 0; i < 100; i++)
        {
            Console.Write("B");
        }
    }
    static void DisplayC()
    {
            //Dừng Thread trong vòng 0.1 giây.
            Thread.Sleep(100);
        for (int i = 0; i < 100; i++)
        {
            Console.Write("C");
        }
    }

    static void Main(string[] args)
    {
        ThreadStart entryPointA = new ThreadStart(DisplayA);
        ThreadStart entryPointB = new ThreadStart(DisplayB);
        ThreadStart entryPointC = new ThreadStart(DisplayC);
        Thread A = new Thread(entryPointA);
        Thread B = new Thread(entryPointB);
        Thread C = new Thread(entryPointC);
        A.Name = " Thread Demo A";
        B.Name = " Thread Demo B";
        C.Name = " Thread Demo C";
        A.Start();
        B.Start();
        C.Start();
        Console.ReadLine();
    }
  }
}
```

Khi biên dịch và thực thi chương trình màn hình sẽ hiển thị kết xuất sau:



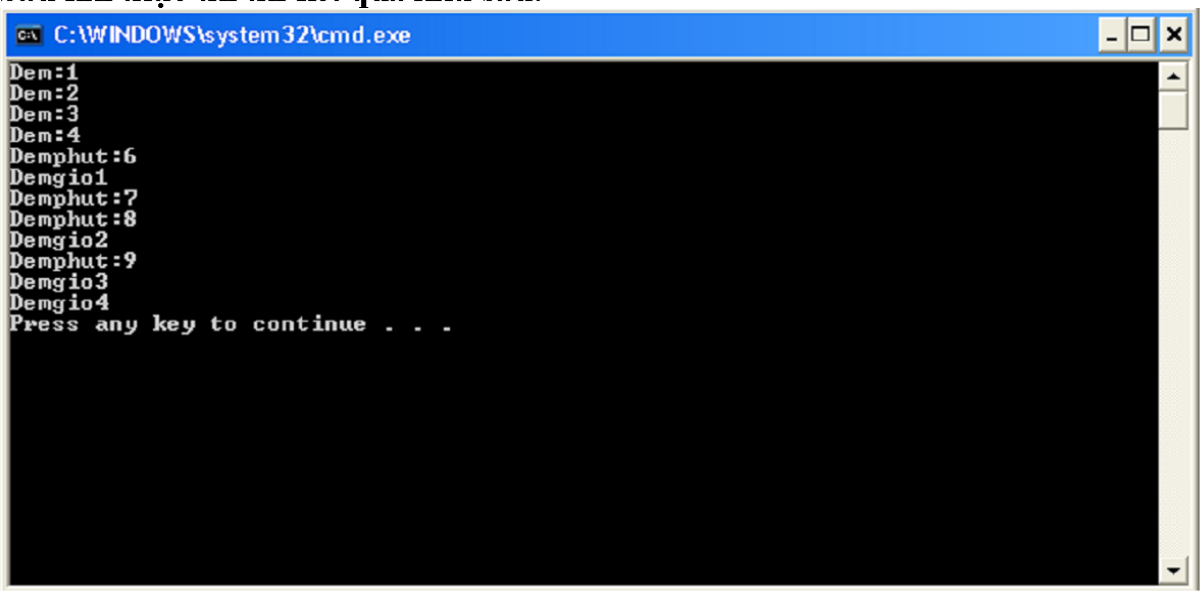
Do thread C có thời gian ngủ ít hơn, nó sẽ là Thread làm việc đầu tiên và do không có yêu cầu nhiều về thời gian nên Thread C có thể chấm dứt trước khi Thread B thực thi. Và vì Thread B cũng không yêu cầu nhiều về thời gian xử lý nên nó có thể chấm dứt trước khi Thread A thức dậy hoạt động.

Ta xem thêm một ví dụ tiếp theo để minh họa cho vấn đề này:

```
using System;
using System.Threading;
public class Example
{
   static void Demgio()
   {
      int i;
      for (i = 1; i < 5; i++)
      {
         Thread.Sleep(1000);
         Console.WriteLine("Demgio"+i);
      }
   }
   static void Demphut()
   {
      int i;
      for (i = 6; i < 10; i++)
      {
         Thread.Sleep(500);
         Console.WriteLine("Demphut:" + i);
      }
   }
   static void Dem()
   {
      int i;
```

```
        for (i = 1; i < 5; i++)
        {

            Console.WriteLine("Dem:"+i);
        }
    }
    public static void Main()
    {
        Thread A = new Thread(new ThreadStart(Demgio));
        Thread B = new Thread(new ThreadStart(Demphut));
        A.Start();
        B.Start();
        Dem();
    }
}
```

Sau khi thực thi thì kết quả như sau:



2.3. Dừng, khởi động lại, hủy bỏ thread:

Trong chương trình của chúng ta sử dụng thread thực thi xử lý những tác vụ đặc biệt. Chúng ta sử dụng phương thức Sleep để dừng thread trong một thời gian biết trước. Tuy nhiên, nếu không biết được thời gian dừng là bao nhiêu lâu bạn có thể gọi phương thức Suspend để dừng vô thời hạn một thread.

A.Suspend();

Tương tự như phương thức Sleep(), Suspend đưa thread vào trạng thái ngủ nhưng không có thời gian tự động thức dậy. Để đánh thức thread dậy hoạt động trở lại ta gọi phương thức Resum() như sau:

A.Resum();

Nếu chúng ta không muốn sử dụng thread nữa chúng ta có thể hủy thread để hệ thống giải phóng tài nguyên bằng cách gọi Abort() như sau:

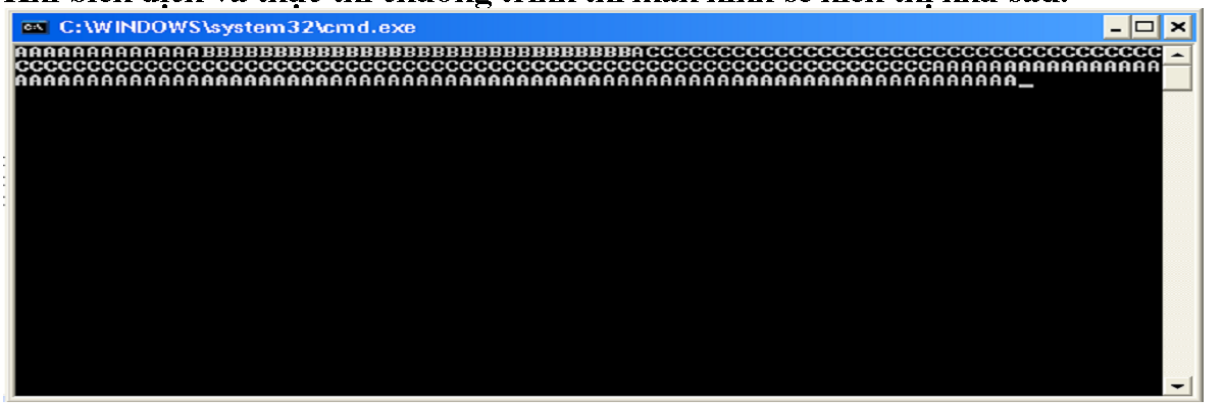
A.Abort();

Chúng ta sẽ xét 3 phương thức ở trên thông qua một thí dụ bên dưới đây:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo5
{
   class Program
   {
      static void DisplayA()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
         }

      }
      static void DisplayB()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("B");
         }
      }
      static void DisplayC()
      {

         for (int i = 0; i < 100; i++)
         {
            Console.Write("C");
         }
      }

      static void Main(string[] args)
      {
         ThreadStart entryPointA = new ThreadStart(DisplayA);
         ThreadStart entryPointB = new ThreadStart(DisplayB);
         ThreadStart entryPointC = new ThreadStart(DisplayC);
         Thread A = new Thread(entryPointA);
         Thread B = new Thread(entryPointB);
         Thread C = new Thread(entryPointC);
         A.Name = " Thread Demo A";
         B.Name = " Thread Demo B";
         C.Name = " Thread Demo C";
         A.Start();
         A.Suspend();
```

```
            B.Start();
            A.Resume();
            C.Start();
            B.Abort();
            Console.ReadLine();
        }
    }
}
```

Khi biên dịch và thực thi chương trình thì màn hình sẽ hiển thị như sau:



Từ ví dụ trên chúng ta thấy, khi thread A bắt đầu nó ngay lập tức bị dừng lại bằng phương thức Suspend. Sau đó thread B được khởi động khi thread B được khởi động thì ta lại đánh thức hoạt động của thread A bằng phương thức A.Resum(), tiếp đến chúng ta lại khởi động thread C và cuối cùng chúng ta hủy thead B để hệ thống giải phóng tài nguyên bằng cách gọi B.Abort().

Ở đây chúng ta phải chú ý thêm một điều là: Nếu chương trình của chúng ta cố hủy thread bằng Abort mà thread không đang trong trạng thái thực thi thì lỗi ngoai lệ sẽ xuất hiện. Thường thì chương trình sẽ sử dụng try...catch...filnally để kiểm tra xem nó có bị dừng chưa.

Ví dụ:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo
{
   class Program
   {
      static void DisplayA()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
         }
      }
```

```
        static void DisplayB()
        {
            try
            {
                for (int i = 0; i < 100; i++)
                {
                    Console.Write("B");
                }
            }
            catch(ThreadAbortException ex)
            {
                Console.Write("Thread B the end");
            }
        }
        static void DisplayC()
        {

            for (int i = 0; i < 100; i++)
            {
                Console.Write("C");
            }
        }

        static void Main(string[] args)
        {
            ThreadStart entryPointA = new ThreadStart(DisplayA);
            ThreadStart entryPointB = new ThreadStart(DisplayB);
            ThreadStart entryPointC = new ThreadStart(DisplayC);
            Thread A = new Thread(entryPointA);
            Thread B = new Thread(entryPointB);
            Thread C = new Thread(entryPointC);
            A.Name = " Thread Demo A";
            B.Name = " Thread Demo B";
            C.Name = " Thread Demo C";
            A.Start();
            A.Suspend();
            B.Start();
            A.Resume();
            C.Start();
            B.Abort();
            Console.ReadLine();
        }
    }
}
```

Phương thức Suspend() có thể không làm cho thread bị đình chỉ tức thời mà có thể là sau một vài lệnh, điều này là để thread được đình chỉ an toàn.

Đối với phương thức Abort() nó làm việc bằng cách ném ra biệt lệ **ThreadAbortException. ThreadAbortException** là một lớp biệt lệ đặc biệt mà không bao giờ được xử lí. Nếu thread đó thực thi mã bên trong khối try, bất kì khối finally sẽ được thực thi trước khi thread bị huỷ.

2.4. Lấy thông tin về thread:

Nếu chúng ta muốn biết thông tin chi tiết về thread đang hoạt động thì lớp Thread cung cấp cho chúng ta rất nhiều phương thức như IsAlive xác định thread còn hoạt động hay không, Priority xác định độ ưu tiên, threadState xác định trạng thái của thread.

Chúng ta sẽ trình bày những điều này thông qua ví dụ phía dưới đây:

Chúng ta xem thêm một ví dụ **ThreadPlayaround** minh họa sau đây để hiểu rõ phần này:

Ví dụ: ThreadPlayaround:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo6
{
   class EntryPoint
   {
      static int interval;
      static void DisplayNumbers()
      {
         Thread thisThread = Thread.CurrentThread;
         string name = thisThread.Name;
         Console.WriteLine("Starting thread: " + name);
         Console.WriteLine(name + ": Current Culture = " +
                           thisThread.CurrentCulture);
         for (int i = 1; i <= 8 * interval; i++)
         {
            if (i % interval == 0)
               Console.WriteLine(name + ": count has reached " + i);
         }
      }
      static void StartMethod()
      {
         DisplayNumbers();
         Console.WriteLine("Worker Thread Finished");
      }
      static void Main()
      {
         Console.Write("Interval to display results at?> ");
         interval = int.Parse(Console.ReadLine());
```

```
        Thread thisThread = Thread.CurrentThread;
        thisThread.Name = "Main Thread";

        ThreadStart workerStart = new ThreadStart(StartMethod);
        Thread workerThread = new Thread(workerStart);
        workerThread.Name = "Worker";
        workerThread.Start();
        DisplayNumbers();
        Console.WriteLine("Main Thread Finished");
        Console.ReadLine();
      }
   }
}
```

Phần chính của ví dụ này là 1 phương thức ngắn, DisplayNumber(), phương thức này cũng bắt đầu bằng việc trình bày tên và bản địa (culture) của thread mà phương thức đang chạy :

```
static void DisplayNumbers()
{
        Thread thisThread = Thread.CurrentThread;
        string name = thisThread.Name;
        Console.WriteLine("Starting thread: " + name);
        Console.WriteLine(name + ": Current Culture = " +
                    thisThread.CurrentCulture);
        for (int i = 1; i <= 8 * interval; i++)
        {
           if (i % interval == 0)
              Console.WriteLine(name + ": count has reached " + i);
        }
}
```

Việc trình bày số tuỳ thuộc vào interval, là một trường mà giá trị được gõ bởi người dùng. Nếu người dùng gõ 100 sẽ trình bày đến 800, trình bày giá trị 100, 200, 300,4 00, 500, 600, 700, 800.

Nếu người dùng gõ 1000 thì sẽ trình bày đến 8000, trình bày giá trị 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000.

Những gì ThreadPlayaround() làm là bắt đầu thread công việc thứ hai, mà chạy DisplayNumber(), nhưng ngay sau khi bắt đầu thread công việc, thread chính bắt đầu thi hành cùng phương thức, nghĩa là ta thấy cả hai việc đếm xảy ra cùng lúc.
Phương thức main() cho ThreadPlayaround và lớp chứa đựng của nó là :

```
class EntryPoint
  {
    static int interval;

    static void Main()
    {
      Console.Write("Interval to display results at?> ");
```

```
        interval = int.Parse(Console.ReadLine());

        Thread thisThread = Thread.CurrentThread;
        thisThread.Name = "Main Thread";

        ThreadStart workerStart = new ThreadStart(StartMethod);
        Thread workerThread = new Thread(workerStart);
        workerThread.Name = "Worker";
        workerThread.Start();

        DisplayNumbers();
        Console.WriteLine("Main Thread Finished");

        Console.ReadLine();

    }
```

Trong phương thức main() đầu tiên ta hỏi người dùng interval, sau đó ta lấy 1 tham chiếu đến đối tượng thread mà đại diện cho thread chính .

Kế tiếp ta tạo ra thread công việc, đặt cùng tên, và bắt đầu nó, truyền cho nó 1 delegate mà chỉ định phương thức mà nó phải bắt đầu trong đó, phương thức workerStart.

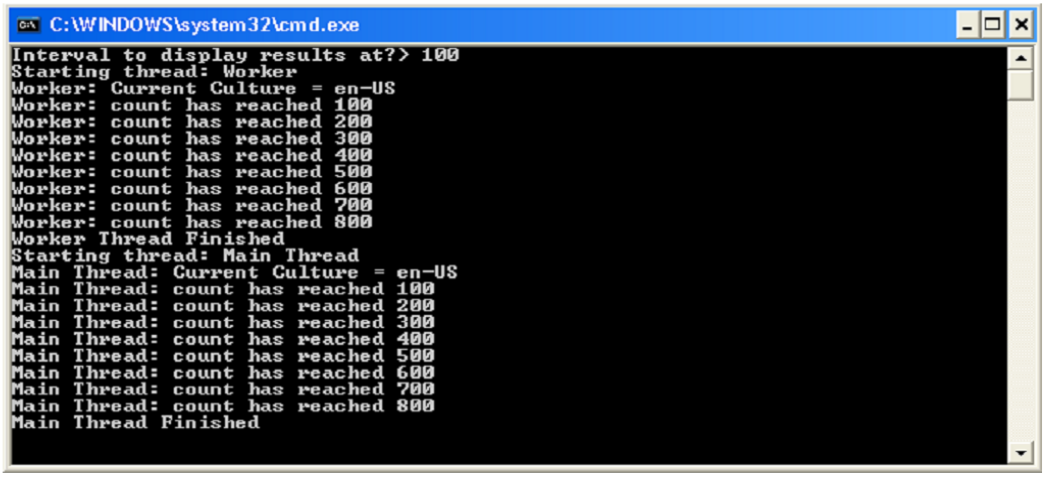
Cuối cùng, ta gọi phương thức DisplayNumber() để bắt đầu đếm, điểm đột nhập(bắt đầu) của thread này là :

```
static void StartMethod()
{
        DisplayNumbers();
        Console.WriteLine("Worker Thread Finished");
}
```

Lưu ý rằng tất cả các phương thức này đều là phương thức static trong cùng một lớp, EntryPoint. Biến i trong phương thức DisplayNumber() được dùng để đếm là một biến cục bộ.

Biến này không chỉ có phạm vi đối với phương thức được định nghĩa mà còn khả kiến đối với thread đang thực thi phương thức đó.

Nếu một thread khác bắt đầu thi hành cùng một phương thức, thì thread đó sẽ lấy một bản sao chép riêng của biến địa phương. Ta sẽ bắt đầu bằng việc chạy đoạn mã cho giá trị interval là 100 thì kết quả như sau:



2.5. Gán tên cho thread:

Khi chương trình của chúng ta sử dụng thread để thực thi xử lý, đôi khi chúng ta muốn biết thread nào đang thực thi. Một cách phân biệt các thread là gán tên duy nhất cho mỗi thread qua thuộc tính name.

Ví dụ:

```
workerThread.Name = "Worker";
```

Ta sẽ minh họa thêm việc gán tên cho thread thông qua ví dụ sau đây:

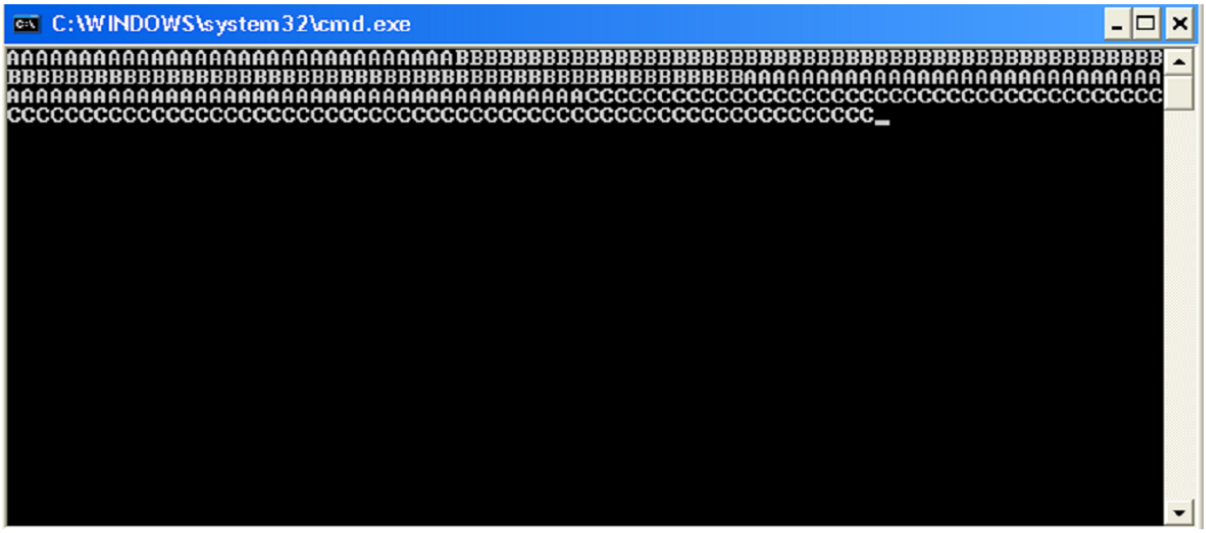
Ví dụ:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
namespace ThreadDemo6
{
   class EntryPoint
   {
      static void Display()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write(Thread.CurrentThread.Name);
         }
      }
      static void Main()
      {
         ThreadStart entryPoint;
         entryPoint = new ThreadStart(Display);
         Thread A = new Thread(entryPoint);
         Thread B = new Thread(entryPoint);
         Thread C = new Thread(entryPoint);
         A.Name = "A";
         A.Start();
         B.Name = "B";
```

```
            B.Start();
            C.Name = "C";
            C.Start();
            Console.ReadLine();
        }
    }
}
```

Ví dụ ở trên minh họa cho việc sử dụng thuộc tính Name của thread. Trong thủ tục Display ở trên đã sử dụng vòng lặp For để hiển thị tên của thread, chúng ta có thể hiển thị ra tên của 3 hay nhiều hơn 3 thread mà chỉ sử dụng một thủ tục.

Khi biên dịch và kết xuất chúng ta có kết quả như sau:



Ở đây chúng ta gặp một vấn đề là khi sử dụng thuộc tính Name của thread để phân biệt các thread đang hoạt động chúng ta có thể gán trùng tên cho hai hoặc nhiều thread.

Ví dụ chúng ta có thể đặt tên như sau:

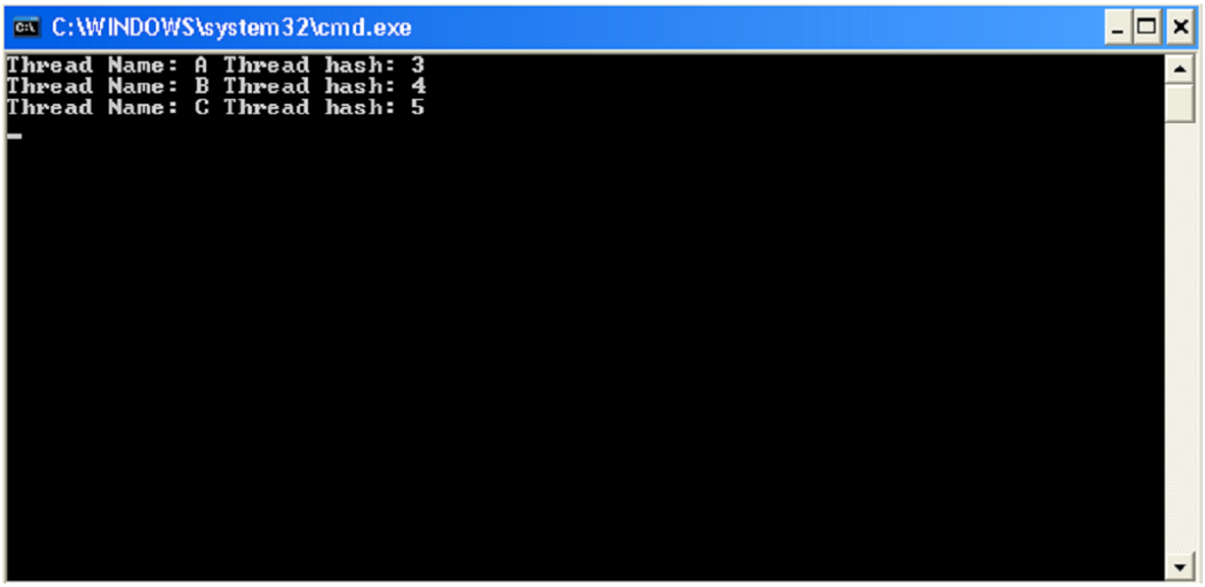
```
A.Name =”A”;
workerThread.Name = “A”;
```

Vì vậy, khi chúng ta phải phân biệt các thread bằng tên chuong trình thì chúng ta nên gọi phương thức GetHashCode, hàm này trả về một đối tượng duy nhất tương ứng với một tên thread. Chương trình ThreadHash.cs sau đây sẽ tạo ra 3 thread, sau đó hiển thị các thread thông qua dùng hàm GethashCode để lấy về thể hiện(instance) của thread.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo7
{
   class EntryPoint
   {
```

```
        static void DisplayHash()
        {
            Thread T;
            int hash;
            T = Thread.CurrentThread;
            hash = T.GetHashCode();
            Console.Write("Thread Name: {0}",Thread.CurrentThread.Name);
            Console.WriteLine("Thread hash: {0}",hash.ToString());
        }
        static void Main()
        {
            ThreadStart entryPoint = new ThreadStart(DisplayHash);
            Thread A = new Thread(entryPoint);
            Thread B = new Thread(entryPoint);
            Thread C = new Thread(entryPoint);
            A.Name = "A";
            A.Start();
            B.Name = "B";
            B.Start();
            C.Name = "C";
            C.Start();
            Console.ReadLine();
        }
    }
}
```

Sau khi biên dịch và thực thi chương trình chúng ta có kết quả như sau:



2.6. Chờ thread khác hoàn thành tác vụ xử lý:

Khi chương trình của chúng ta sử dụng thread để thực hiện xử lý, sẽ có lúc chúng ta cần chờ thread khác hoạt động xong thì thread nào đó mới tiếp tục xử lý.

Ví dụ: Nếu chúng ta không muốn thread thực hiện xem tài liệu khi thread khác mang tên Dowloading đang tải dữ liệu về từ Internet chưa hoàn thành xong xử lý của nó. Để dừng thực thi của thread cho đến khi thread khác hoàn thành xong tác vụ xử lý của nó, lớp thread cung cấp cho chúng ta phương thức Join. Phát biểu sau sẽ định hướng các thread khác chờ cho đến khi thread Dowloading hoàn tất.

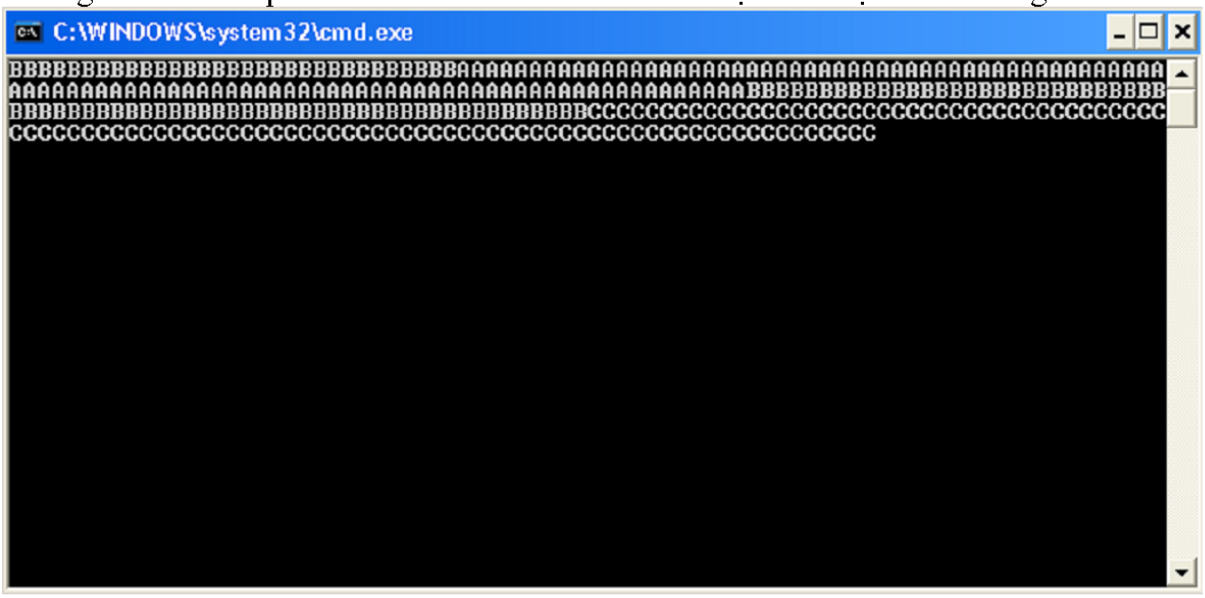
Dowloading.Join();

Ta xét một ví dụ sau đây để thấy điều này:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo8
{
   class Program
   {
      static void DisplayA()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
         }
      }
      static void DisplayB()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("B");
         }
      }
      static void DisplayC()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("C");
         }
      }
      static void Main(string[] args)
      {
                ThreadStart entryPointA = new ThreadStart(DisplayA);
                ThreadStart entryPointB = new ThreadStart(DisplayB);
                ThreadStart entryPointC = new ThreadStart(DisplayC);
        Thread A = new Thread(entryPointA);
        Thread B = new Thread(entryPointB);
        Thread C = new Thread(entryPointC);
         B.Start();
```

```
            A.Start();
            B.Join();
            A.Join();
            C.Start();
            Console.ReadLine();
        }
    }
}
```

Trong chương trình trên thread C sẽ dừng và đợi cho đến khi thread A và B hoàn tất Chúng ta xem kết quả ở hình bên dưới sau khi biên dịch và thực thi chương trình:



2.7. Điều khiển quyền ưu tiên thread:

.Khi có nhiều thread thực thi thì hệ thống sẽ gán thời gian chiếm giữ CPU cho mỗi thread. Khi thread đang chạy thì thông thường CPU sẽ dành cho thread 33% thời gian CPU. Tùy thuộc vào xử lý mà thread thực thi trong ứng dụng của chúng ta sẽ có lúc thread thực thi này quan trọng hơn những thread khác. Trong những trường hợp như vậy ta có thể đặt cho thread một quyền ưu tiên cao hơn, khiến cho thread có nhiều thời gian chiêm lĩnh CPU hơn. Ta dùng thuộc tính Priority của lớp Thread để làm điều này.

Ta có NameSpace **System.Thread.Thread.Priority** cung cấp các hằng số giá trị ưu tiên bao gồm: Lowest(thấp nhất), BellowNormal(dưới mức bình thường), Normal(Bình thường), AboveNormal(trên mức bình thường), Highest(cao nhất).

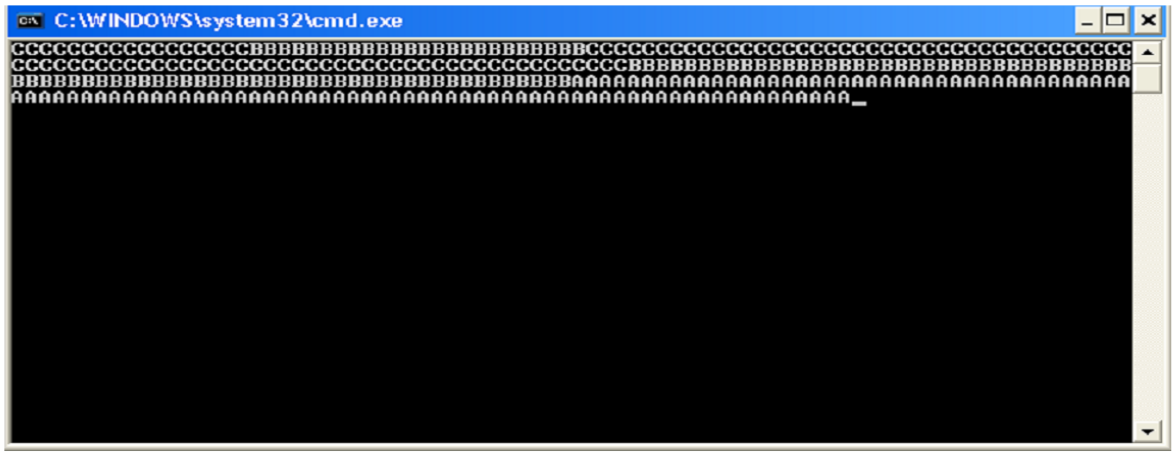
Ví dụ: Ta sẽ gán quyền ưu tiên cao hơn cho thread A và quyền ưu tiên thấp nhất cho thread B như sau:

```
A.Priority = ThreadPriority.Highest;
B.Priority = ThreadPriority.Lowest;
```

Ta xét một ví dụ sau đây:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
```

```
namespace ThreadDemo10
{
   class Program
   {
      static void DisplayA()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("A");
         }
      }
      static void DisplayB()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("B");
         }
      }
      static void DisplayC()
      {
         for (int i = 0; i < 100; i++)
         {
            Console.Write("C");
         }
      }
      static void Main(string[] args)
      {
         ThreadStart entryPointA = new ThreadStart(DisplayA);
         ThreadStart entryPointB = new ThreadStart(DisplayB);
         ThreadStart entryPointC = new ThreadStart(DisplayC);
         Thread A = new Thread(entryPointA);
         Thread B = new Thread(entryPointB);
         Thread C = new Thread(entryPointC);
         A.Name = " Thread Demo A";
         B.Name = " Thread Demo B";
         C.Name = " Thread Demo C";
         A.Priority = ThreadPriority.BelowNormal;
         C.Priority = ThreadPriority.AboveNormal;
         A.Start();
         B.Start();
         C.Start();
         Console.ReadLine();
      }
   }
}
```



Ta thấy rằng vì thread C có quyền ưu tiên cao nhất trong 3 thread nên thread này hoàn thành quá trình xử lý của nó trước. Vì thread A có quyền ưu tiên thấp nhất nên xử lý của nó sẽ kết thúc sau cùng. Chúng ta chú ý rằng thread có quyền ưu tiên thấp hơn thì nó sẽ chạy ít hơn.

2.8. Chia sẻ thread(Thread Pool):

Ta có thể tổ chức một nhóm các thread nằm chờ phía sau hậu trường của ứng dụng và gán công việc cho chúng thực thi. Các thread có thể luân phiên thực hiện nhiều nhiệm vụ khác nhau mà không cần phải hủy và tạo lại các đối tượng thread mới.

Thay vì phải tạo ra từng thread xử lý từng sự kiện riêng biệt(như timer) chúng ta có thể tổ chức một thread xử lý từng sự kiện cùng loại hay có cùng tính chất.

Khi sử dụng thread dạng chia sẻ, ta không gọi phương thức Start() mà gọi phương thức QueueUserWorkItem của lớp ThreadPool cùng với chuyển địa chỉ sẽ thực thi thread cho phương thức:

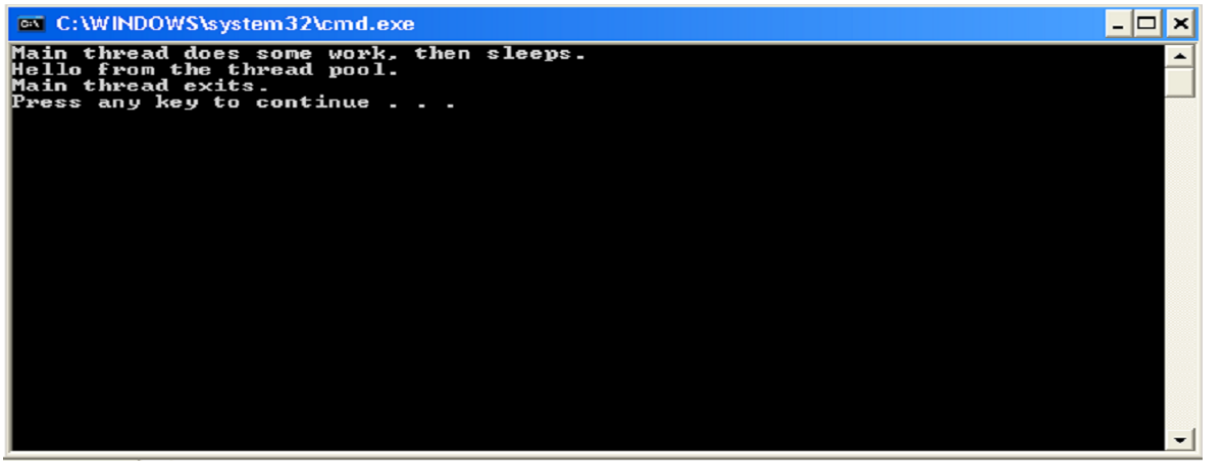
```
AutoResetEvent AisDone = new AutoResetEvent(false);
ThreadPool.QueueUserWorkItem(new WaitCallback(DisplayA), AisDone);
```

Ta xét ví dụ sau:

```
using System;
using System.Threading;
public class Example
{
   // This thread procedure performs the task.
   static void ThreadProc(Object stateInfo)
   {
      // No state object was passed to QueueUserWorkItem, so
      // stateInfo is null.
      Console.WriteLine("Hello from the thread pool.");
   }
   public static void Main()
   {
      // Queue the task.
      ThreadPool.QueueUserWorkItem(new WaitCallback(ThreadProc));
      Console.WriteLine("Main thread does some work, then sleeps.");
      Thread.Sleep(1000);
```

```
        Console.WriteLine("Main thread exits.");
    }
}
```

Kết quả thể hiện như sau:

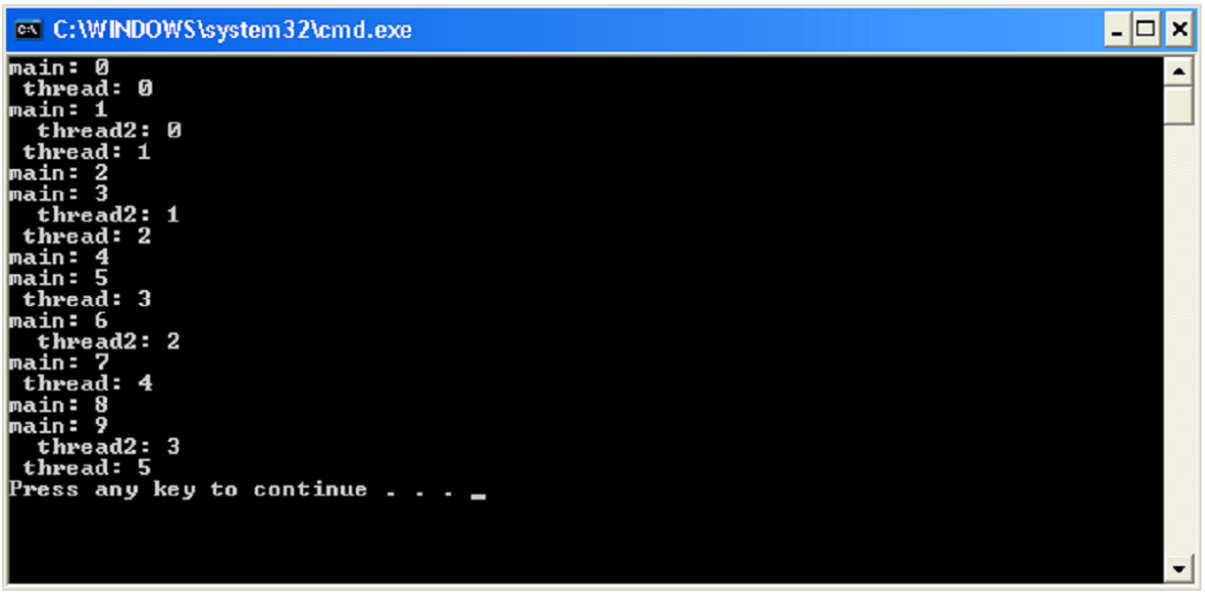


Ta xét một ví dụ tiếp sau đây:

```
using System;
using System.Threading;
class ThreadPoolSample
{
        public static void Main()
        {
                ThreadPoolSample tps = new ThreadPoolSample();
        }
        public ThreadPoolSample()
        {
                int i;
                ThreadPool.QueueUserWorkItem(new WaitCallback(Counter));
                ThreadPool.QueueUserWorkItem(new WaitCallback(Counter2));
                for(i = 0; i < 10; i++)
                {
                        Console.WriteLine("main: {0}", i);
                        Thread.Sleep(1000);
                }
        }
        void Counter(object state)
        {
                int i;
                for (i = 0; i < 10; i++)
                {
                        Console.WriteLine(" thread: {0}", i);
                        Thread.Sleep(2000);
                }
        }
```

```
        void Counter2(object state)
        {
                int i;
                for (i = 0; i < 10; i++)
                {
                        Console.WriteLine("  thread2: {0}", i);
                        Thread.Sleep(3000);
                }
        }
}
```

Sau khi biên dịch và thực thi chương trình ta có kết quả sau:



2.9. Sự tranh tài nguyên giữa các thread:

Khi ứng dụng sử dụng nhiều thread thì khả năng xảy ra đụng độ và tranh tài nguyên là rất cao. Mỗi thread chạy trong một thời gian quy định và dừng lại chờ thread khác thực thi xong, lúc này các quá trình sử dụng tài nguyên bị tạm dừng.

Khi một thread khác thực thi nó có thể sử dụng tiếp tài nguyên của thread dừng trước đó và đây là khả năng xung đột hay tranh chấp tài nguyên xảy ra.

Hay một ví dụ khác, giả sử bạn có một thread đặt tên là Producer xuất tài nguyên cho một thread thứ hai tên là Consumer sử dụng. Nếu Producer không đáp ứng kịp tài nguyên cho Consumer xử lý thì Consumer sẽ ra sao?

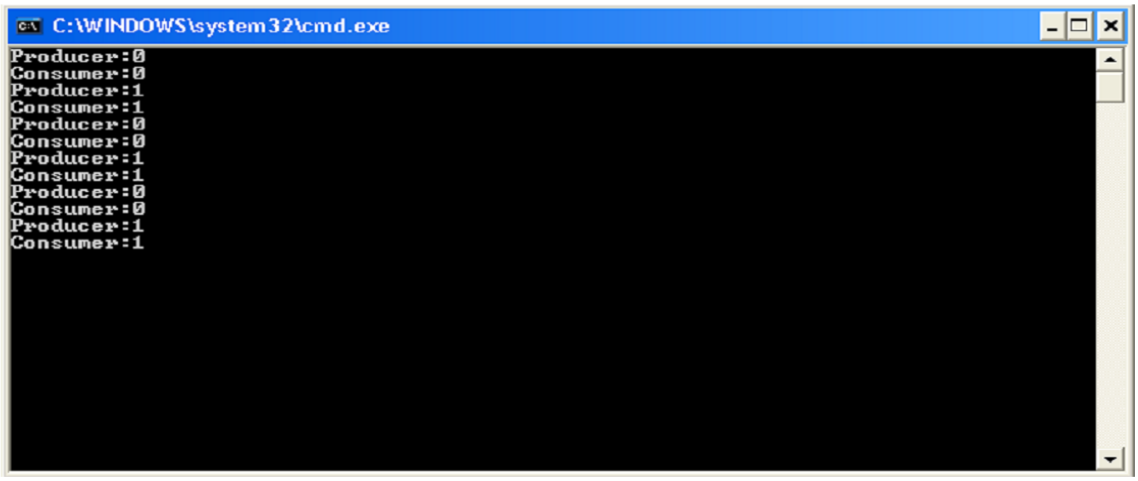
Ví dụ: thread Producer thực thi theo thủ tục sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo12
{
   class Program
```

```
{
    public static int Buffer;
    public static Boolean BufferEmpty= true;
    static void Producer()
    {
        int Value = 0;
        do
        {
            if(BufferEmpty)
            {
                BufferEmpty = false;
                Buffer = Value;
            }
            if (Value = = 0)
            {
                Value = 1;
            }
            else
            {
                Value = 0;
            }
            Console.WriteLine("Producer:"+Buffer);
            //Dung 1 giay
            Thread.Sleep(1000);
        } while (true);
    }
    static void Consumer()
    {
        int Value;
        do
        {
            if ( !BufferEmpty)
            {
                BufferEmpty = true;
                Value = Buffer;
                Console.WriteLine("Consumer:"+ Value);
            }

        } while (true);
    }
    static void Main(string[] args)
    {
        Thread ProducerThread;
        Thread ConsumerThread;
        ProducerThread = new Thread(Producer);
        ConsumerThread = new Thread(Consumer);
```

```
            ProducerThread.Start();
            ConsumerThread.Start();
            Console.ReadLine();
        }
    }
}
```

Kết quả của chương trình thực hiện như sau:



Chương trình thực hiện mặc dù kết xuất có vẻ đúng nhưng chương trình thật sự có nhiều tiềm năng bị lỗi rất nặng. Ví dụ như khi Producer sinh ra dữ liệu trước khi thread Consumer có khả năng tiêu thụ hay sử dụng dữ liệu thì sao? Hay Consumer tiêu thụ dữ liệu thì nhanh nhưng Producer không kịp sinh dữ liệu?

Khi phát biểu If ước lượng bằng true (có nghĩa rằng bộ đệm rỗng), Producer ngay lập tức đặt một cờ cho biết vùng đệm có dữ liệu.

Để phát hiện ra lỗi chúng ta thay đổi phương thức Consumer như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo13
{
   class Program
   {
      public static int Buffer;
      public static Boolean BufferEmpty= true;
      public static int OldValue;
      static void Producer()
      {
         int Value = 0;
         do
         {
            if(BufferEmpty)
```

```
        {
            BufferEmpty = false;
            Buffer = Value;
        }
        if (Value = = 0)
        {
            Value = 1;
        }
        else
        {
            Value = 0;
        }
        Console.WriteLine("Producer:"+Buffer);
        //Dung 1 giay
        Thread.Sleep(3000);
    } while (true);
}
static void Consumer()
{
    int Value = -1;
    do
    {
        if (!BufferEmpty)
        {
            BufferEmpty = true;
            OldValue = Value;
            Value = Buffer;
            if (Value = = OldValue)
            {
                Console.WriteLine("Loi Consumer!");
                break;
            }
        }
        Console.WriteLine("Consumer:"+Value);
    } while (true);
}
static void Main(string[] args)
{
    Thread ProducerThread;
    Thread ConsumerThread;
    ProducerThread = new Thread(Producer);
    ConsumerThread = new Thread(Consumer);
    ProducerThread.Start();
    ConsumerThread.Start();
    Console.ReadLine();
}
```

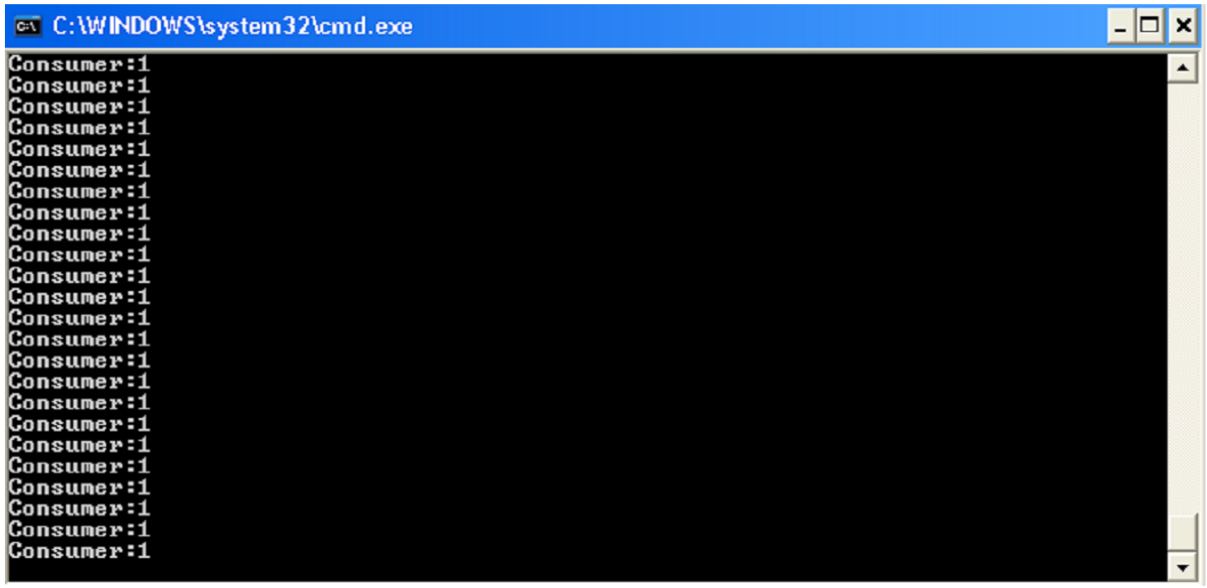
```
    }
}
```

Ta có thể có nhu cầu kéo dài thời gian hoạt động của thread trước khi thời gian hoán chuyển CPU diễn ra để tránh lỗi.

Ví dụ: Ta đưa thêm dòng Thread.Sleep(3000); vào thủ tục Consumer().

Ta thường gọi lỗi kiểu này là lỗi đồng bộ hóa xuất hiện trong ứng dụng có nhiều thread, các thread cùng đua tranh để kiểm soát được tài nguyên ứng dụng.

Ta có kết xuất sau:



2.10. Giết một Thread (Kill thread)

Thông thường thread sẽ chấm dứt khi hàm mà nó thực thi trở về. Tuy nhiên bạn có thể yêu cầu một thread "tự tử" bằng cách gọi hàm **Interrupt()** của nó. Điều này sẽ làm cho exception **ThreadInterruptedException** được ném ra. Thread bị yêu cầu "tự tử" có thể bắt exception này để tiến hành dọn dẹp tài nguyên.

```
catch (ThreadInterruptedException)
{
 Console.WriteLine("[{0}] Interrupted! Cleaning up...",
 Thread.CurrentThread.Name);
}
```

3. Đồng bộ hóa (Synchronization)

Khi ta cần bảo vệ một tài nguyên, trong một lúc chỉ cho phép một thread thay đổi hoặc sử dụng tài nguyên đó, ta cần **đồng bộ hóa**.

Đồng bộ hóa được cung cấp bởi một khóa trên đối tượng đó, khóa đó sẽ ngăn cản thread thứ 2 truy cập vào đối tượng nếu thread thứ nhất chưa trả quyền truy cập đối tượng.

Vậy khi ta cần phối hợp các hoạt động của nhiều tuyến trình để bảo đảm sử dụng hiệu quả các tài nguyên dùng chung, và ta không làm sai lạc dữ liệu dùng chung khi một phép chuyển ngữ cảnh tuyến trình (*thread context switch*) xảy ra trong quá trình thay đổi dữ liệu ta cần đồng bộ hóa.

Sau đây là ví dụ **cần** sự đồng bộ hóa. Giả sử 2 thread sẽ tiến hành tăng **tuần tự 1 đơn vị** một biến tên là counter.

int counter = 0;

Hàm làm thay đổi giá trị của Counter:

```
public static void Incrementer()
{
      int counter=0;
      try
      {
         while (counter < 1000)
         {
            int temp = counter;
            temp++; // increment
            // simulate some work in this method
            Thread.Sleep(10);
            counter = temp;
           Console.WriteLine("Thread {0}. Incrementer: {1}",
        Thread.CurrentThread.Name, counter);
         }
      }
        catch
        {
                  Console.Write("Thread B the end");
        }
}
```

Chương trình như sau:

```
using System;
using System.Threading;
public class Example
{
   // This thread procedure performs the task.
   public static void Incrementer()
   {
      int counter = 0;
      try
      {
         while (counter < 1000)
         {
            int temp = counter;
            temp++; // increment
            // simulate some work in this method
            Thread.Sleep(10);
            // assign the Incremented value
            // to the counter variable
            // and display the results
            counter = temp;
```

```
                Console.WriteLine("Thread {0}. Incrementer: {1}",
                Thread.CurrentThread.Name, counter);
            }
        }
        catch
        {
            Console.Write("Thread B the end");
        }
    }
    public static void Main()
    {
        Thread th1 = new Thread(new ThreadStart(Incrementer));
        Thread th2 = new Thread(new ThreadStart(Incrementer));
        th1.Start();
        th2.Start();
    }
}
```

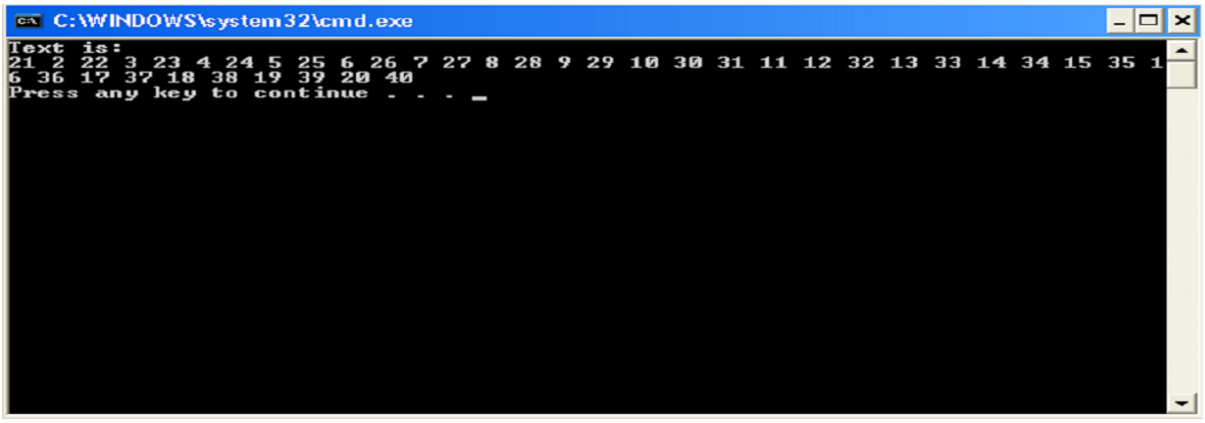
Vấn đề ở chỗ thread 1 đọc giá trị counter vào biến tạm rồi tăng giá trị biến tạm, trước khi thread 1 ghi giá trị mới từ biến tạm trở lại counter thì thread 2 lại đọc giá trị counter ra biến tạm của thread 2. Sau khi thread 1 ghi giá trị vừa tăng 1 đơn vị trở lại counter thì thread 2 lại ghi trở lại counter giá trị mới bằng với giá trị mà thread 1 vừa ghi. Như vậy sau 2 lần truy cập giá trị của biến counter chỉ tăng 1 đơn vị trong khi yêu cầu là phải tăng 2 đơn vị.

Ta xét ví dụ tiếp sau đây để thấy rõ hơn vấn đề về sự đồng bộ hóa:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo14
{
    class Program

    {
        static StringBuilder text = new StringBuilder();
        public static void Func1()
        {
            for (int i = 1; i <= 20; i++)
            {
                Thread.Sleep(10);
                text.Append(i.ToString() + " ");
            }
        }
        public static void Func2()
        {
            for (int i = 21; i <= 40; i++)
```

```
            {
                Thread.Sleep(10);
                text.Append(i.ToString() + " ");
            }
        }
        static void Main(string[] args)
        {
            Thread firstThread = new Thread(new ThreadStart(Func1));
            Thread secondThread = new Thread(new ThreadStart(Func2));
            firstThread.Start();
            secondThread.Start();
            firstThread.Join();
            secondThread.Join();
            Console.WriteLine("Text is:\r\n{0}",text.ToString());
        }
    }
}
```

Thực thi ta có kết quả như sau:



Chúng ta có thể thay đổi chương trình trên bằng cách dùng từ khóa Lock với cú pháp như sau:

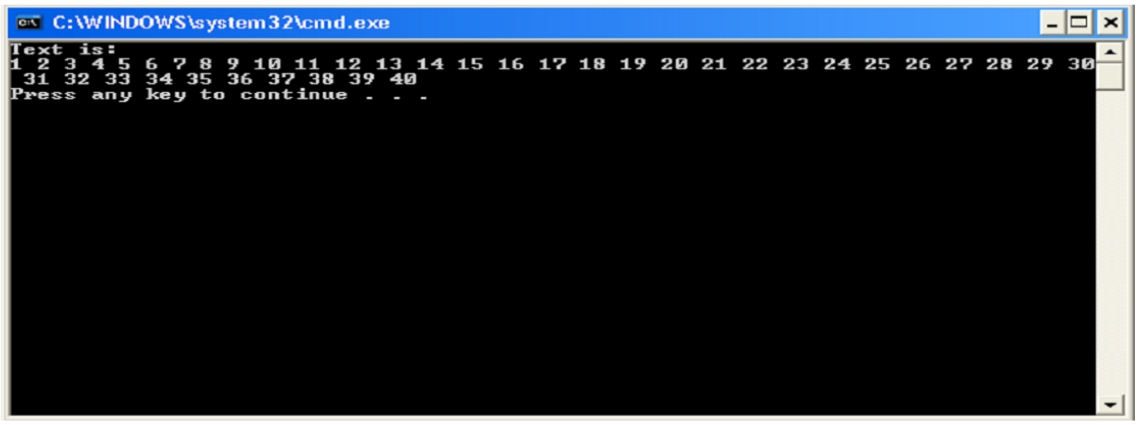
```
Lock(x)
{
        Dosomething();
}
```

Vậy chương trình trên được viết lại mà có sử dụng từ khóa lock như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace ThreadDemo15
{
   class Program
   {
```

```
        static StringBuilder text = new StringBuilder();
        public static void Func1()
        {
            lock (text)
            {
                for (int i = 1; i <= 20; i++)
                {
                    Thread.Sleep(10);
                    text.Append(i.ToString() + " ");
                }
            }
        }
        public static void Func2()
        {
            lock (text)
            {
                for (int i = 21; i <= 40; i++)
                {
                    Thread.Sleep(10);
                    text.Append(i.ToString() + " ");
                }
            }
        }
        static void Main(string[] args)
        {
            Thread firstThread = new Thread(new ThreadStart(Func1));
            Thread secondThread = new Thread(new ThreadStart(Func2));
            firstThread.Start();
            secondThread.Start();
            firstThread.Join();
            secondThread.Join();
            Console.WriteLine("Text is:\r\n{0}",text.ToString());
        }
    }
}
```

Biên dịch và thực thi thủ tục ta được kết quả như sau:



Câu lệnh lock sẽ bao 1 đối tượng gọi là mutual exclusion lock hay mutex. Trong khi mutex bao 1 biến, thì không 1 thread nào được quyền truy nhập vào biến đó.

Trong đoạn mã trên khi câu lệnh hợp thực thi và nếu time slice(chạy trong khoảng thời gian ngắn) của thread này kết thúc và thread kế tiếp thử truy xuất vào biến x, việc truy xuất đến biến sẽ bị từ chối. Thay vào đó window đơn giản đặt thread đó ngủ cho đến khi mutex được giải phóng. Mutex là 1 cơ chế đơn giản được dùng để điều khiển việc truy nhập vào các biến. Tất cả việc điều khiển này nằm trong lớp System.Threading.Monitor. Câu lệnh lock gồm 1 số phương thức gọi đến lớp này.

3.1.Các vấn đề đồng bộ:

Việc đồng bộ các thread là quan trọng trong các ứng dụng đa thread. Tuy nhiên có 1 số lỗi tinh vi và khó thăm dò có thể xuất hiện cụ thể là **deadlock** và **race condition.**

Deadlock

Deadlock là 1 lỗi mà có thể xuất hiện khi hai thread cần truy nhập vào các tài nguyên mà bị khoá lẫn nhau. Giả sử 1 thread đang chạy theo đoạn mã bên dưới, trong đó a, b là 2 đối tượng tham chiếu mà cả hai thread cần truy nhập:

```
lock (a)
{
        // do something

        lock (b)
        {
                // do something

        }
}
```

Vào cùng lúc đó 1 thread khác đang chạy:

```
lock (b)
{
        // do something

        lock (a)
        {
            // do something
```

```
        }
}
```

Có thể xảy ra biến cố sau: thread đầu tiên yêu cầu 1 lock trên a, trong khi vào cùng thời điểm đó thread thứ hai yêu cầu lock trên b. Một khoảng thời gian ngắn sau, thread a gặp câu lệnh lock(b), và ngay lập tức bước vào trạng thái ngủ, đợi cho lock trên b được giải phóng. Và tương tự sau đó, thread thứ hai gặp câu lệnh lock(a) và cũng rơi vào trạng thái ngủ chờ cho đến khi lock trên a được giải phóng.

Không may, lock trên a sẽ không bao giờ được giải phóng bởi vì thread đầu tiên mà đã lock trên a đang ngủ và không thức dậy. cho đến khi lock trên b được giải phóng điều này cũng không thể xảy ra cho đến khi nào thread thứ hai thức dậy.

Kết quả là deadlock. Cả hai thread đều không làm gì cả, đợi lẫn nhau để giải phóng lock. Loại lỗi này làm toàn ứng dụng bị treo.Ta phải dùng Task Manager để hủy nó.Deadlock có thể được tránh nếu cả hai luồn yêu cầu lock trên đối tượng theo cùng thứ tự. Trong ví dụ trên nếu thread thứ hai yêu cầu lock cùng thứ tự với thread đầu, a đầu tiên rồi tới b thì những thread mà lock trên a đầu sẽ hoàn thành nhiệm vụ của nó sau đó các thread khác sẽ bắt đầu.

Ta xét ví dụ sau đây:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
namespace ThreadDemo16
{
   class Test
   {
      static StringBuilder text = new StringBuilder();
      static StringBuilder doc = new StringBuilder();
      public static void Func1()
      {
         lock (text)
         {
            Thread.Sleep(10);
            for (int i = 1; i <= 20; i++)
            {
               text.Append(i.ToString() + " ");
            }
            lock (doc)
            {
               doc.Append(text.ToString());
               for (int i = 1; i <= 20; i++)
               {
                  doc.Append(i.ToString()+" ");
               }
            }
         }
```

```
    }
    public static void Func2()
    {
        lock (doc)
        {
            Thread.Sleep(10);
            for (int i = 21; i <= 40; i++)
            {
                text.Append(i.ToString() + " ");
            }
            lock (text)
            {
                text.Append(doc.ToString());
                for (int i = 21; i <= 40; i++)
                {
                    text.Append(i.ToString() + " ");
                }
            }
        }
    }
    static void Main(string[] args)
    {
        Thread firstThread = new Thread(new ThreadStart(Func1));
        Thread secondThread = new Thread(new ThreadStart(Func2));
        firstThread.Start();
        secondThread.Start();
    }
}

}
```

Race condition

Race condition là 1 cái gì đó tinh vi hơn deadlock. Nó hiếm khi nào dừng việc thực thi của tiến trình, nhưng nó có thể dẫn đến việc dữ liệu bị lỗi.

Nói chung nó xuất hiện khi vài thread cố gắng truy nhập vào cùng 1 dữ liệu và không quan tâm đến các thread khác làm gì để hiểu ta xem ví dụ sau : Giả sử ta có 1 mảng các đối tượng, mỗi phần tử cần được xử lí bằng 1 cách nào đó, và ta có 1 số thread giữa chúng làm tiến trình nay. Ta có thể có 1 đối tượng gọi là ArrayController chứa mảng đối tượng và 1 số int chỉ định số phần tử được xử lí.

Ta có phương thức:

```
int GetObject(int index)
{
```

```
            // trả về đối tượng với chỉ mục được cho
        }
```

Và thuộc tính read/write:

```
int ObjectsProcessed
{
    // chỉ định bao nhiêu đối tượng được xử lí
}
```

Bây giờ mỗi thread mà dùng để xử lí các đối tượng có thể thi hành đoạn mã sau:

```
lock(ArrayController)
{
        int nextIndex = ArrayController.ObjectsProcessed;
        Console.WriteLine("object to be processed next is " + nextIndex);
        ++ArrayController.ObjectsProcessed;
        object next = ArrayController.GetObject();
}
ProcessObject(next);
```

Nếu ta muốn tài nguyên không bị giữ quá lâu, ta có thể không giữ lock trên ArrayController trong khi ta đang trình bày thông điệp người dùng.

Do đó ta viết lại đoạn mã trên:

```
lock(ArrayController)
{
        int nextIndex = ArrayController.ObjectsProcessed;
}
        Console.WriteLine("object to be processed next is " + nextIndex);


lock(ArrayController)
{
        ++ArrayController.ObjectsProcessed;
        object next = ArrayController.GetObject();
}
ProcessObject(next);
```

Ta có thể gặp 1 vấn đề. Nếu 1 thread lấy 1 đối tượng(đối tượng thứ 11 trong mảng) và đi tới trình bày thông điệp nói về việc xử lí đối tượng này.

Trong khi đó thread thứ hai cũng bắt đầu thi hành cũng đoạn mã gọi ObjectProcessed, và quyết định đối tượng xử lí kế tiếp là đối tượng thứ 11, bởi vì thread đầu tiên vẫn chưa được cập nhật.

ArrayController.ObjectsProcessed trong khi thread thứ hai đang viết đến màn hình rằng bây giờ nó sẽ xử lí đối tượng thứ 11, thread đầu tiên yêu cầu 1 lock khác trên ArrayController và bên trong lock này tăng ObjectsProcessed. không may, nó quá

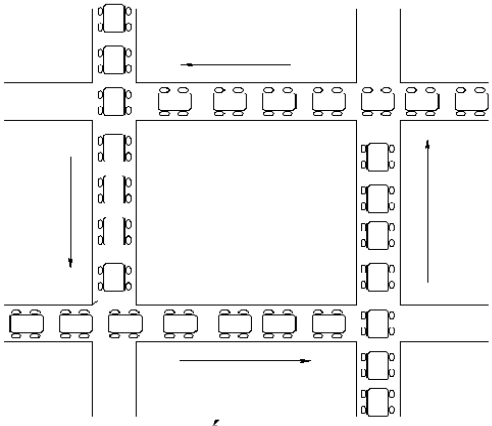
trễ. Cả hai thread đều đang xử lí cùng 1 đối tượng và loại tình huống này ta gọi là Race Condition.

3.2. Sau đây chúng ta sẽ xem một số ưu điểm và hạn chế của cơ chế đa tuyến.

Ưu điểm và hạn chế của cơ chế đa tuyến

a) Ưu điểm: Trong một ứng dụng có thể cho nhiều "công việc" (tuyến-thread) được thực thi đồng thời. Các công việc này hoạt động độc lập với nhau và đóng vai trò như những chương trình riêng biệt. Một thread có thể treo (Hang) mà không ảnh hưởng đến các thread khác. Đa tuyến đặc biệt hữu ích trong việc xây dựng các ứng dụng Server – cái mà đáp ứng nhiều yêu cầu từ các Client.

b) Hạn chế: Việc cài đặt chương trình theo kiểu đa tuyến phức tạp hơn. Đồng thời phải giải quyết vấn đề xung đột tài nguyên dùng chung. Việc cài đặt đa tuyến cũng sẽ tốn bộ nhớ hơn vì hệ thống phải lưu lại những thông tin về mỗi thread. Hơn nữa Processor phải mất nhiều thời gian hơn cho việc luân chuyển (switching) giữa các tuyến với nhau. Một vấn đề khác nữa là tình huống khóa chết – deadlock !



Khóa chết – Deadlock

Đa tuyến trong .NET

MS.NET Framework cung cấp cho nhà phát triển cơ chế đa tuyến theo mô hình Free Threading. Việc tạo và quản lý tuyến được thực hiện thông qua lớp **Thread**, thuộc namespace là **System.Threading**.

Sau đây ta sẽ thống kê một số phương thức và thuộc tính thường dùng.

a) Mô tả về lớp Thread:

Một số phương thức thường dùng

Public Method Name	Mô tả
Abort()	Kết thúc Thread
Join()	Buộc chương trình (tuyến hiện tại) phải chờ cho thread kết thúc (Block) thì mới thực hiện tiếp (các câu lệnh đứng sau Join).
Resume()	
Sleep()	Static method : Tạm dừng thread trong một khoảng thời gian.
Start()	Bắt đầu chạy (khởi động) một thread. Sau khi gọi phương thức này,

Public Method Name	Mô tả
	trạng thái của thread chuyển từ trạng thái hiện hành sang Running.
Suspend()	Dừng vô thời hạn một thread.

b)Một số thuộc tính public thường dùng:

Public Property Name	Mô tả
CurrentThread	This static property: Trả về thread hiện hành đang chạy.
IsAlive	Trả về giá trị cho biết trạng thái thực thi của thread hiện hành.
IsBackground	Sets or gets giá trị cho biết là thread là background hay foreground thread.
IsThreadPoolThread	Gets a value indicating whether a thread is part of a thread pool.
Priority	Sets or gets giá trị để chỉ định độ ưu tiên (dành nhiều hay ít CPU cho thread). Cao nhất là 4, thấp nhất là 0.
ThreadState	Lấy về trạng thái của thread (đang dừng, hay đang chạy…)

4. Bài tập(xem trong phần bài tập ở phía sau).

5. Xử lý tiến trình(Process):

5.1. Tìm hiểu tiến trình với lớp Process:

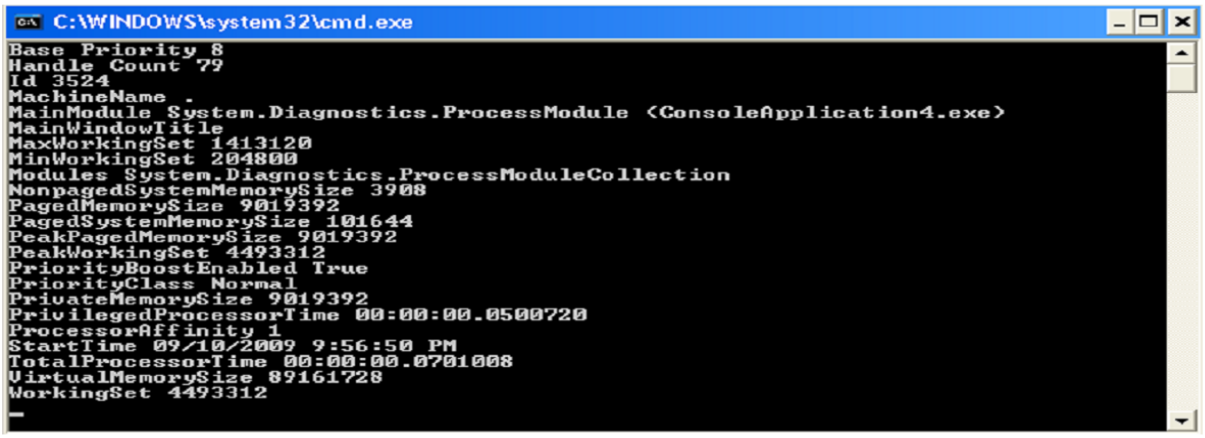
Trong môi trường Windows, chương trình thường chạy trong ngữ cảnh của một tiến trình(process). Lập trình viên thường sử dụng hoán chuyển giữa thuật ngữ. Sự khác nhau đó là tiến trình nằm trong ngữ cảnh được hệ điều hành cấp phát tài nguyên như CPU xử lý, bộ nhớ ...Visual cung cấp cho chúng ta lớp Process với rất nhiều thuộc tính và phương thức xử lý tiến trình đang chạy, gọi thực thi và dừng tiến trình, lấy thông tin về tài nguyên mà tiến trình đang sử dụng....

Chương trình sau đây sử dụng các thuộc tính của lớp đối tượng Process để hiển thị thông tin về chương trình đang chạy như số lượng ký ức tiến trình đang sử dụng, quyền ưu tiên tiến trình, mức độ sử dụng CPU của các thread trong tiến trình.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.Diagnostics;

namespace ThreadDemo16
{
   class Program
   {
      static void Main(string[] args)
      {
```

```
            Process aProcess;
            aProcess = Process.GetCurrentProcess();
            Console.WriteLine("Base Priority {0}",aProcess.BasePriority);
            Console.WriteLine("Handle Count {0}", aProcess.HandleCount);
            Console.WriteLine("Id {0}", aProcess.Id);
            Console.WriteLine("MachineName {0}", aProcess.MachineName);
            Console.WriteLine("MainModule {0}", aProcess.MainModule);
            Console.WriteLine("MainWindowTitle{0}", aProcess.MainWindowTitle);
            Console.WriteLine("MaxWorkingSet {0}", aProcess.MaxWorkingSet);
            Console.WriteLine("MinWorkingSet {0}", aProcess.MinWorkingSet);
            Console.WriteLine("Modules {0}", aProcess.Modules);
           Console.WriteLine("NonpagedSystemMemorySize {0}",
           aProcess.NonpagedSystemMemorySize);
            Console.WriteLine("PagedMemorySize {0}", aProcess.PagedMemorySize);
           Console.WriteLine("PagedSystemMemorySize {0}",
           aProcess.PagedSystemMemorySize);
            Console.WriteLine("PeakPagedMemorySize {0}",
           aProcess.PeakPagedMemorySize);
            Console.WriteLine("PeakWorkingSet {0}", aProcess.PeakWorkingSet);
           Console.WriteLine("PriorityBoostEnabled {0}",
           aProcess.PriorityBoostEnabled);
            Console.WriteLine("PriorityClass {0}", aProcess.PriorityClass);
            Console.WriteLine("PrivateMemorySize {0}", aProcess.PrivateMemorySize);
           Console.WriteLine("PrivilegedProcessorTime {0}",
           aProcess.PrivilegedProcessorTime);
            Console.WriteLine("ProcessorAffinity {0}", aProcess.ProcessorAffinity);
            Console.WriteLine("StartTime {0}", aProcess.StartTime);
            Console.WriteLine("TotalProcessorTime {0}", aProcess.TotalProcessorTime);
            Console.WriteLine("VirtualMemorySize {0}", aProcess.VirtualMemorySize);
            Console.WriteLine("WorkingSet {0}", aProcess.WorkingSet);
            Console.ReadLine();
        }
    }
}
```

Kết quả thể hiện như sau:



5.2. Tải tiến trình mới sử dụng lớp Process:

Ta có thể gọi phương thức Start của lớp Process để chạy một chương trình đang nằm trên máy.

Ví dụ:

```
Process.Start("C:\Program Files\UNIKEY\UNIKEY.EXE");
```

Ta có thể mở một file tài liệu dựa vào phần tên mở rộng của chúng.

Ví dụ:

```
Process.Start("D:\Vanban.doc");
```

Ta xét chương trình sau đây để minh họa cho những điều đã trình bày ở trên:

5.3. Mở một trang Web tự động:

Ta có thể mở một trang Web theo địa chỉ Website hay file .html cũng bằng Start().

Ví dụ:

```
Process.Start("IExplore.exe","www.vcse.edu.vn");
```

5.4. Chấm dứt một tiến trình:

Khi chúng ta muốn chấm dứt một tiến trình đang chạy chúng ta sử dụng phương thức Kill của lớp Process để chấm dứt nó.

```
Process.GetProcessById(186).Kill();
Process.GetCurrentProcess().Kill();
```

Phương thức	Mô tả
`GetCurrentProcess`	Trả về đối tượng `Process` mô tả tiến trình hiện đang tích cực.
`GetProcessById`	Trả về đối tượng `Process` mô tả tiến trình với *ID* được chỉ định.
`GetProcesses`	Trả về mảng các đối tượng `Process` mô tả tất cả các tiến trình hiện đang tích cực.
`GetProcessesByName`	Trả về mảng các đối tượng `Process` mô tả tất cả các tiến trình hiện đang tích cực với tên thân thiện được chỉ định. Tên thân thiện là tên của file thực thi không tính phần mở rộng và đường dẫn; ví dụ, *notepad* hay *calc*.

Ví dụ: Ta tải Internet Explorer hiển thị file trợ giúp html, sau đó khi chương trình chấm dứt ta cũng kết thúc đóng chương trình Internet Explorer lại.

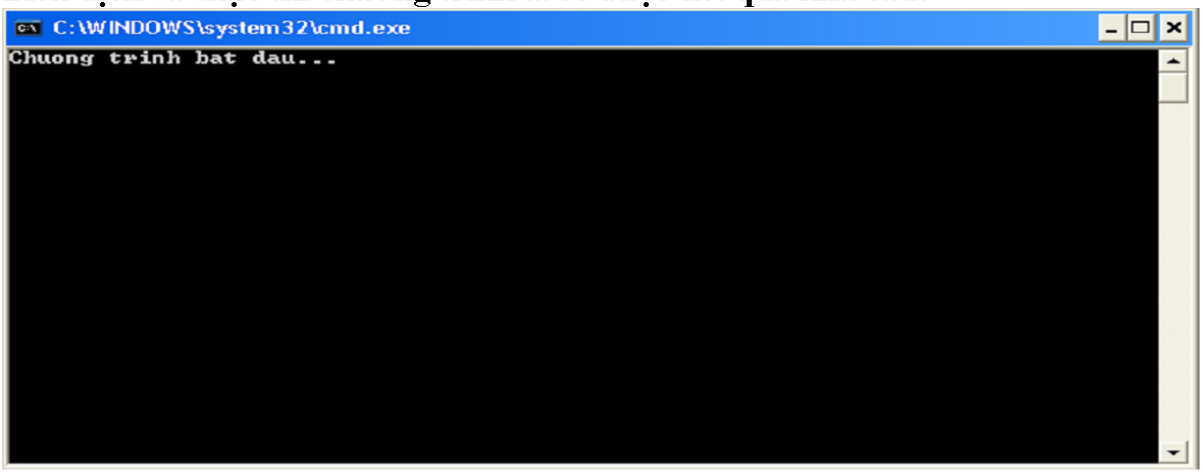
5.5. Ngăn hai bản sao của cùng một chương trình chạy đồng thời:

Tùy theo mục đích của chương trình ta có thể không muốn hai bản sao của cùng một chương trình chạy cùng một lúc. Trong những trường hợp như vậy, chương trình của ta có thể sử dụng phương thức GetProcessByName của lớp Process để tìm kiếm tiến trình cùng tên với tiến trình mà ta đang chạy, nếu trùng tên có nghĩa là chương trình đã chạy trước đó.

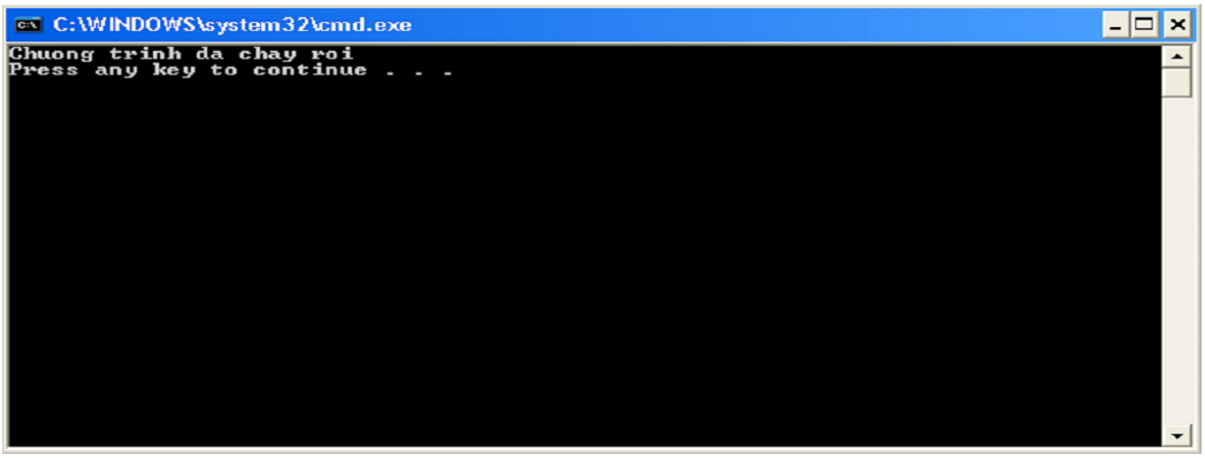
Ta sẽ thấy rõ điều này khi xem ví dụ dưới đây:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace ThreadDemo17
{
   class Program
   {
      static void Main(string[] args)
      {
         string TargetName;
         TargetName = Process.GetCurrentProcess().ProcessName;
         Process[] MatchingNames = Process.GetProcessesByName(TargetName);
         if (MatchingNames.Length == 1)
         {
            Console.WriteLine("Chuong trinh bat dau...");
            Console.ReadLine();
         }
         else
         {
            Console.WriteLine("Chuong trinh da chay roi");
         }
      }
   }
}
```

Biên dịch và thực thi chương trình ta sẽ được kết quả như sau:



Khi ta chạy thêm một lần nữa thì sẽ xuất hiện màn hình sau:



5.6. Xem thông tin về các tiến trình trong hệ thống:

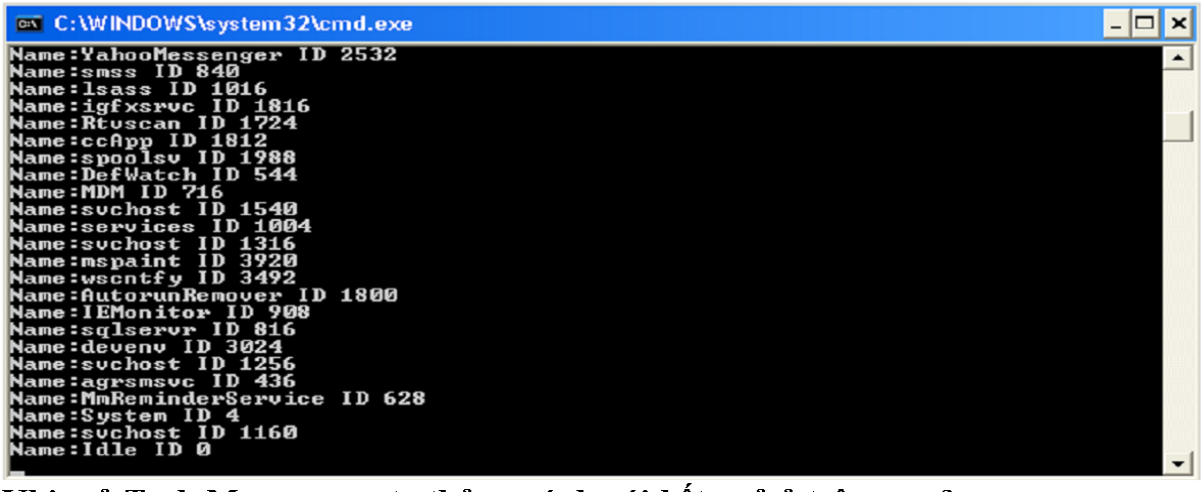
Nếu ta cần biết thông tin về tiến trình khác đang chạy trong hệ thống, ta có thể gọi phương thức GetProcess của lớp Process trả về một tập hợp những tiến trình hiện hành của hệ thống.

Sau khi ta biên dịch và thực thi chương trình thì danh sách các tiến trình lấy được hoàn toàn tương ứng với danh sách các tiến trình hiển thị trong danh sách Processes của cửa sổ Task Manager.

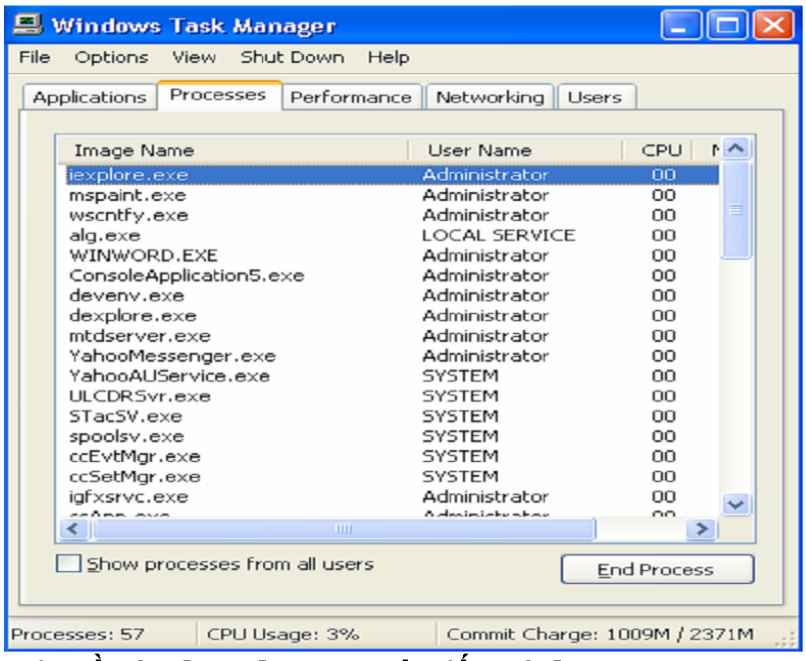
Ta xem ví dụ phía dưới đây để thấy rõ điều này:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace ThreadDemo18
{
   class Program
   {
      static void Main(string[] args)
      {
         Process[] ProcessList;
         ProcessList = Process.GetProcesses();
         foreach(Process Proc in ProcessList)
         {
            Console.WriteLine("Name:{0} ID {1}",Proc.ProcessName,Proc.Id);
         }
         Console.ReadLine();

      }
   }
}
```

Biên dịch và thực thi thủ tục ta sẽ được kết quả sau:



Khi mở Task Manager ra ta thử so sánh với kết quả ở trên xem?



5.7. Lấy thông tin về các thread trong một tiến trình:

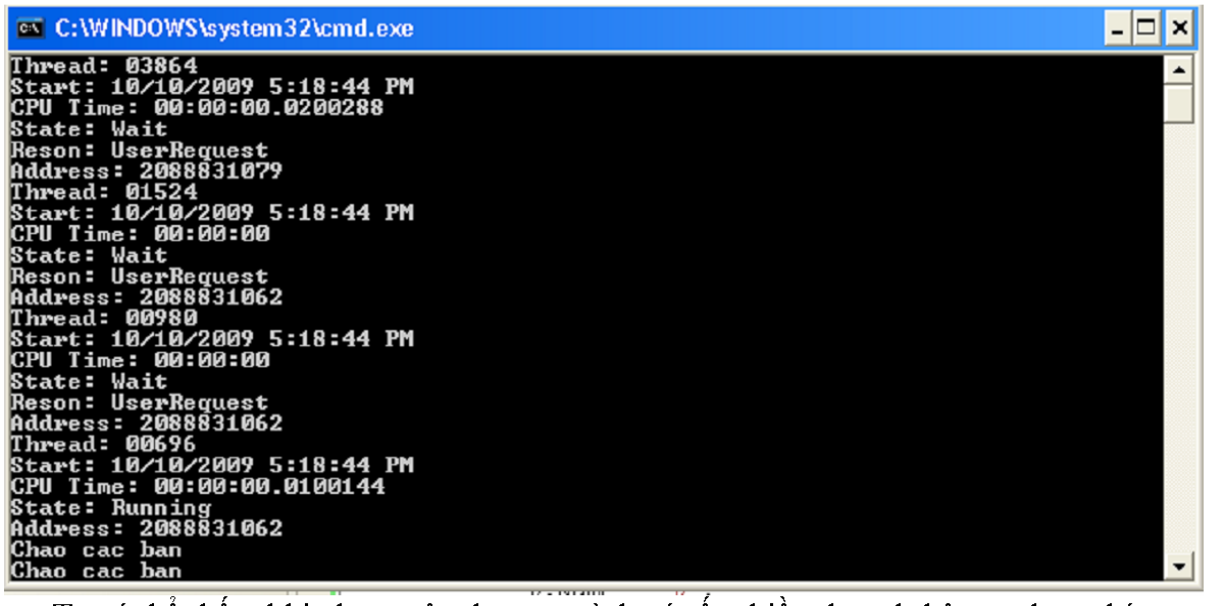
Một tiến trình có thể có nhiều thread thực thi, để lấy về danh sách các đối tượng thread bạn gọi phương thức Threads() của lớp.

Ví dụ:

```
 using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using System.Threading;

namespace ThreadDemo19
{
   class Program
   {
      static void Display()
```

```
        {
            Console.WriteLine("Chao cac ban");
        }
        static void ThreadInfo()
        {
            ProcessThreadCollection List;
            List = Process.GetCurrentProcess().Threads;
            foreach(ProcessThread objThread in List)
            {
                Console.WriteLine("Thread: {0:D5}",objThread.Id);
                Console.WriteLine("Start: {0}", objThread.StartTime);
                Console.WriteLine("CPU Time: {0}", objThread.TotalProcessorTime);
                Console.WriteLine("State: {0}", objThread.ThreadState);
                if (objThread.ThreadState == System.Diagnostics.ThreadState.Wait)
                {
                    Console.WriteLine("Reson: {0}",objThread.WaitReason);
                }
                Console.WriteLine("Address: {0}",objThread.StartAddress);
            }
        }
        static void Main(string[] args)
        {
            ThreadStart entryPoint = new ThreadStart(Display);
            ThreadStart entryPointInfo = new ThreadStart(ThreadInfo);
            Thread A = new Thread(entryPoint);
            Thread B = new Thread(entryPoint);
            Thread C = new Thread(entryPoint);
            Thread D = new Thread(entryPointInfo);
            A.Name = "A";
            B.Name ="B";
            C.Name = "C";
            D.Name = "D";
            D.Start();
            D.Join();
            A.Start();
            B.Start();
            C.Start();
        }
    }
}
```

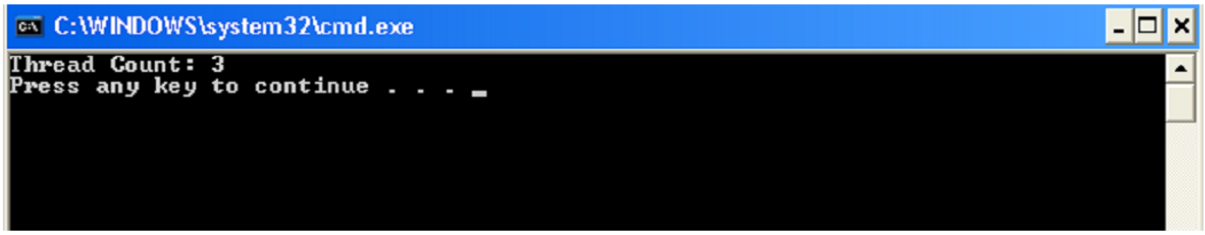
Biên dịch và thực thi chương trình trên ta được:



Ta có thể thấy, khi chạy một chương trình có rất nhiều thread thật sự chạy phía sau một tiến trình. Nếu chương trình của ta cho phép chế độ gỡ lỗi thì một số thread sẽ thuộc về trình gỡ rối(debugger). Nếu tiến trình của ta sử dụng nhiều thread, một số thread sẽ tương ứng với thread của phần mềm quản lý.

Ví dụ: Chương trình sau đây sẽ có một phát biểu đơn giản là trình bày số thread trong một tiến trình: ta xem hình minh họa phía dưới để thấy kết quả khi biên dịch và thực thi chương trình.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using System.Threading;
namespace ThreadDemo20
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Thread Count: {0}",
                                    Process.GetCurrentProcess().Threads.Count);
      }
   }
}
```



BÀI 2. LẬP TRÌNH VỚI SOCKET

Mục tiêu của bài:

Nhằm trang bị cho người học:

- Kiến thức về Socket.
- Kiến thức và kỹ năng lập trình với giao thức TCP/IP nhằm tạo các chương trình đơn giản qua mạng Lan và mạng Internet.
- Kiến thức và kỹ năng lập trình múc Socket nhằm tạo các chương trình cao cấp truyền dữ liệu qua Socket.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

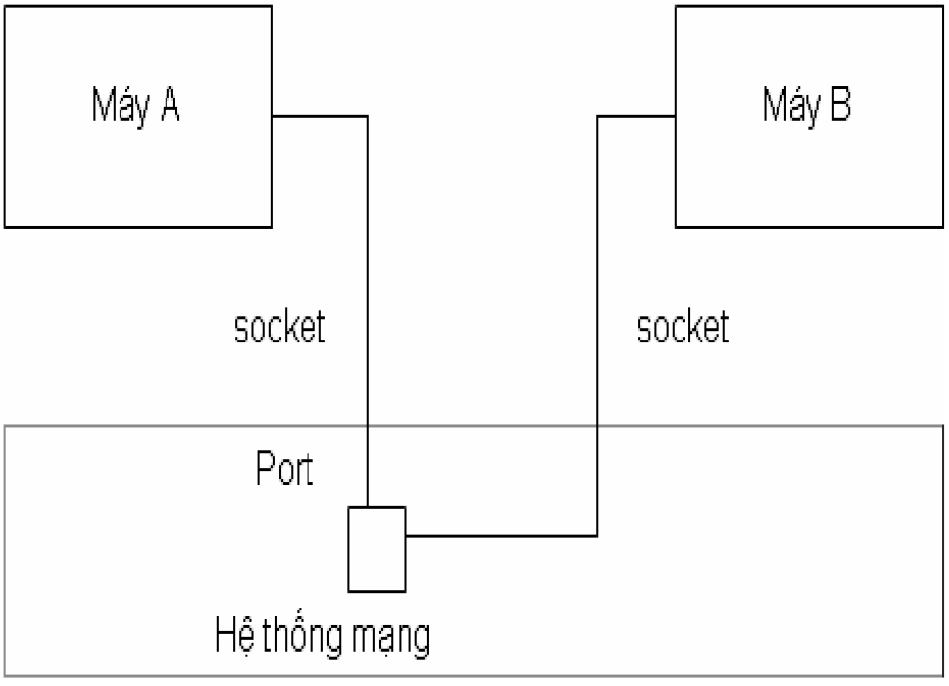
1.Giới thiệu về Socket:

Socket là một giao diện lập trình ứng dụng (API) mạng. Thông qua giao diện này chúng ta có thể lập trình điều khiển việc truyền thông giữa hai máy sử dụng các giao thức mức thấp là TCP, UDP…

Socket là sự trừu tượng hoá ở mức cao, có thể tưởng tượng nó như là thiết bị truyền thông hai chiều gửi - nhận dữ liệu giữa hai máy tính với nhau.

Socket là một thiết bị truyền thông 2 chiều tương tự như tập tin(có thể đọc, ghi) tuy nhiên mỗi socket là một thành phần trong một mối nối nào đó giữa các máy trên mạng và các thao tác đọc/ghi chính là sự trao đổi dữ liệu giữa các ứng dụng trên nhiều máy khác nhau.

Trong nghi thức truyền thông TCP mỗi mối nối giữa 2 máy tính được xác định bởi 1 port, mỗi port tương ứng với một số nguyên dương.



Các loại Socket:

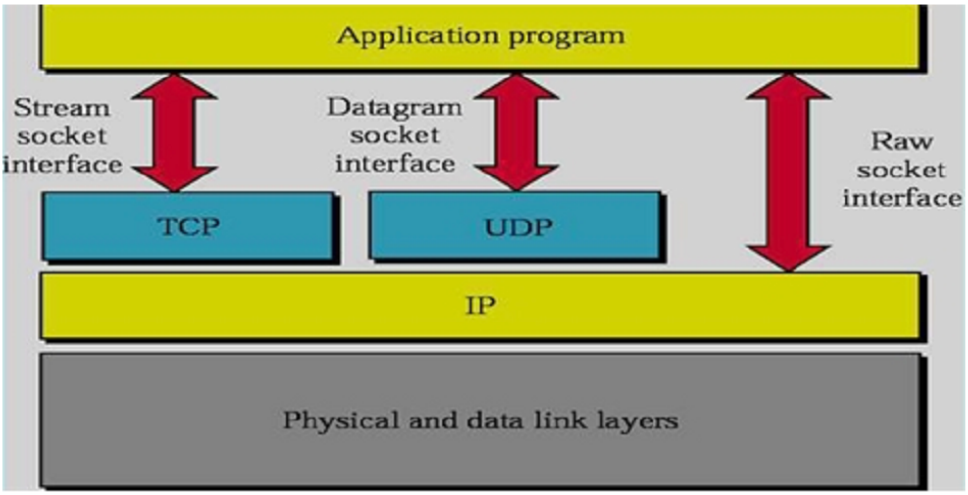
- Socket hướng kết nối (TCP Socket).
- Socket không hướng kết nối (UDP Socket).
- Raw Socket.
- **Đặc điểm của Socket hướng kết nối:**
- Có 1 đường kết nối ảo giữa 2 tiến trình.
- Một trong 2 tiến trình phải đợi tiến trình kia yêu cầu kết nối.
- Có thể sử dụng để liên lạc theo mô hình Client/Server.
- Trong mô hình Client/Server thì Server lắng nghe và chấp nhận một yêu cầu kết nối.
- Mỗi thông điệp gửi đều có xác nhận trở về.
- Các gói tin chuyển đi tuần tự.

Đặc điểm của Socket không hướng kết nối:

- Hai tiến trình liên lạc với nhau không kết nối trực tiếp.
- Thông điệp gửi đi phải kèm theo địa chỉ của người nhận.
- Thông điệp có thể gửi nhiều lần.
- Người gửi không chắc chắn thông điệp tới tay người nhận.
- Thông điệp gửi sau có thể đến đích trước thông điệp gửi trước đó.

Số hiệu cổng của Socket:

- Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng.
- Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống.
- Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các quá trình khác.
- Quá trình còn lại cũng yêu cầu tạo ra một socket.



Hình 2.2

Cách tạo Socket cụ thể sẽ được trình bày chi tiết ở phần tiếp theo.

1.1. Khái niệm địa chỉ và cổng(Address & Port):

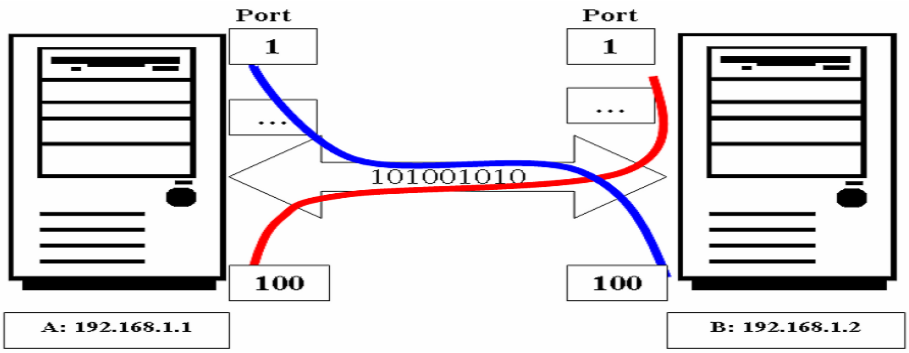
a) Nguyên lý:

Trong một máy có rất nhiều ứng dụng muốn trao đổi với các ứng dụng khác thông qua mạng. Như chúng ta thấy hình phía dưới có 2 ứng dụng trong máy A muốn trao đổi với 2 ứng dụng trên máy B.

Mỗi máy tính chỉ có duy nhất một đường truyền để gửi và nhận.

Vậy vấn đề được đặt ra là: rất có thể xảy ra nhầm lẫn khi dữ liệu từ máy A gửi đến máy B thì không biết là dữ liệu đó gửi cho ứng dụng nào trên máy B?.

Giải quyết vấn đề như sau: Mỗi ứng dụng trên máy B sẽ được gán một số hiệu(mà ta vẫn quen gọi là cổng(Port)). Số hiệu cổng này từ 1...65535. Khi ứng dụng trên máy A muốn gởi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu cổng(vào trường RemotePort) vào gói tin cần gửi. Trên máy B, các ứng dụng chỉ việc kiểm tra giá trị cổng trên mỗi gói tin xem có trùng với số hiệu cổng của mình(đã được gán, chính là giá trị LocalPort) hay không?. Nếu bằng thì xử lý còn trái lại thì không làm gì cả(vì không phải của mình).



Hình 2.3

Như vậy: Khi cần trao đổi dữ liệu cho nhau thì hai ứng dụng cần phải biết thông tin tối thiểu là **Địa chỉ** (Address) và **số hiệu cổng** (Port) của ứng dụng kia.

- Hai ứng dụng có thể cùng nằm trên một máy.
- Hai ứng dụng trên cùng một máy không được trùng số hiệu cổng.

- **LocalHost:** (Địa chỉ máy hiện đang chạy ứng dụng):
  Với **B**: LocalHost = 192.168.1.2, với **A** thì Localhost = 192.168.1.1;
- **RemoteHost** (Địa chỉ của máy chạy ứng dụng đang tham gia trao đổi thông tin với ứng dụng hiện tại).
  - RemoteHost của ứng dụng chạy trên máy A là: 192.168.1.2; RemoteHost của ứng dụng chạy trên máy B là: 192.168.1.1.
- **LocalPort:** LocalPort của ứng dụng chạy trên máy A (FTP) là 100, của ứng dụng chạy trên máy B (FTP) là 5.
- **RemotePort:** RemotePort của ứng dụng chạy trên máy A (FTP) là 5, của ứng dụng chạy trên máy B (FTP) là 100.
- Hai ứng dụng đặt trên hai máy khác nhau thì LocalPort có thể giống nhau(Nhưng nếu đặt trên một máy thì không được trùng nhau).

Một số cổng và các giao thức thông dụng:

- FTP: 21
- Telnet: 23
- SMTP: 25
- POP3: 110
- HTTP:80

1.2.Giới thiệu về NameSpace System.Net và System.Net.Sockets:

Cung cấp một giao diện lập trình đơn giản cho rất nhiều các giao thức mạng.

Có rất nhiều lớp để lập trình. Ta quan tâm lớp IPAdress, IPEndPoint, DNS, …

a) Lớp IPAdress:

Một số Field cần chú ý:

- **Any:** Cung cấp một địa chỉ IP để chỉ ra rằng Server phải lắng nghe trên tất cả các Card mạng.
- **Broadcast:** Cung cấp một địa chỉ IP quảng bá.
- **Loopback:** Trả về một địa chỉ IP lặp.
- **AdressFamily:** Trả về họ địa chỉ của IP hiện hành.

Một số phương thức cần chú ý:

Phương thức khởi tạo:

IPAddress(Byte[]).

IPAddress(Int64).

- **IsLoopback:** Cho biết địa chỉ có phải địa chỉ lặp không.
- **Parse:** Chuyển IP dạng xâu về IP chuẩn.
- **ToString:** Trả địa chỉ IP về dạng xâu.
- **TryParse:** Kiểm tra IP ở dạng xâu có hợp lệ không?

b) Lớp IPEndPoint:

- **Một số phương thức cần chú ý:**

  Phương thức khởi tạo:
  - IPEndPoint (Int64, Int32)

- IPEndPoint (IPAddress, Int32)

❖ **Create:** Tạo một EndPoint từ một địa chỉ Socket.

❖ **ToString:** Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng Địa Chỉ:Cổng

Ví dụ: 192.168.1.1:8080

c) Lớp IPHostEntry:

❖ **Một số phương thức cần chú ý:**

❖ **AddressList:** Nhận hoặc thiết lập một danh sách các địa chỉ IP kết hợp với một host.

❖ Aliases: **:** Nhận hoặc thiết lập một danh sách Aliases kết hợp với một host.

❖ **HostName:** Nhận hoặc thiết lập tên **DNS** kết hợp với một host.

d)Lớp DNS:

Một số thành phần của lớp:

❖ **HostName:**Cho biết tên của máy được phân giải.

❖ **GetHostAddress:** Trả về tất cả IP của một trạm.

❖ **GetHostEntry:** Giải đáp tên hoặc địa chỉ truyền vào và trả về đối tượng IPHostEntry.

❖ **GetHostName:** Lấy về tên của máy tính cục bộ.

*** Lưu ý:** Đây là các **phương thức tĩnh**, do vậy khi gọi thì gọi trực tiếp từ tên lớp mà không cần phải khai báo một đối tượng mới của lớp này.

Ví dụ ta gọi: DNS.Resolve, Dns.GetHostname, Dns.GetHostEntry v.v…

e)NameSpace System.Net.Sockets:

Một số lớp hay dùng: TcpClient, UdpClient, TcpListener, Socket, NetWorkStream,...

1.3. Lớp IPAddress

1.3.1. Giới thiệu

Trên Internet mỗi một trạm (có thể là máy tính, máy in, thiết bị…) đều có một định danh duy nhất, định danh đó thường được gọi là một địa chỉ (Address). Địa chỉ trên Internet là một tập hợp gồm 4 con số có giá trị từ 0-255 và cách nhau bởi dấu chấm.

Để thể hiện địa chỉ này, người ta có thể viết dưới các dạng sau:

- Tên : ví dụ May01, Server.
- Địa chỉ IP nhưng đặt trong một xâu: "192.168.1.1", "127.0.0.1"
- Đặt trong một mảng 4 byte, mỗi byte chứa một số từ 0-255.

  Ví dụ: Để biểu diễn địa chỉ 192.168.1.1 ta có thể viết:

```
Byte[] DiaChi = new Byte[3];
DiaChi[0] = 192;
DiaChi[1] = 168;
DiaChi[2] = 1;
DiaChi[3] = 1;
```

- Hoặc cũng có thể là một số (long), có độ dài 4 byte.

  Ví dụ: với địa chỉ 192.168.1.1 ở trên thì giá trị đó sẽ là: 16885952 (đây là số ở hệ thập phân khi xếp liền 4 byte ở trên lại với nhau.

00000001 00000001 **10101000** 11000000

1 (Byte 0) 1 168 192 (Byte 3)

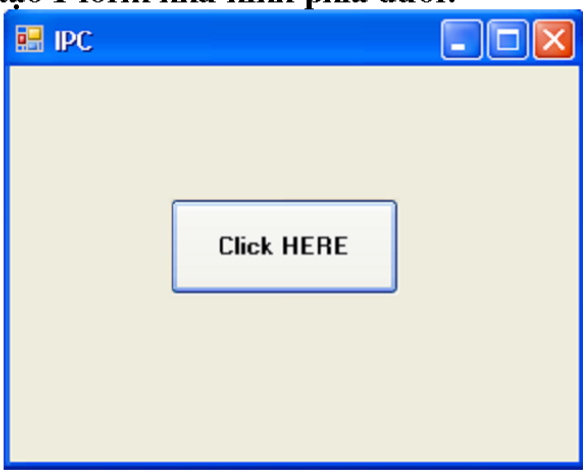
➔ Như vậy, để đổi một địa chỉ chuẩn ra dạng số ta chỉ việc tính toán cho từng thành phần.

Ví dụ: Đổi địa chỉ 192.168.1.2 ra số, ta tính như sau :

2 * 256 ^ 3 + 1* 256 ^ 2 + 168 * 256 ^ 1 + 192 * 256 ^ 0

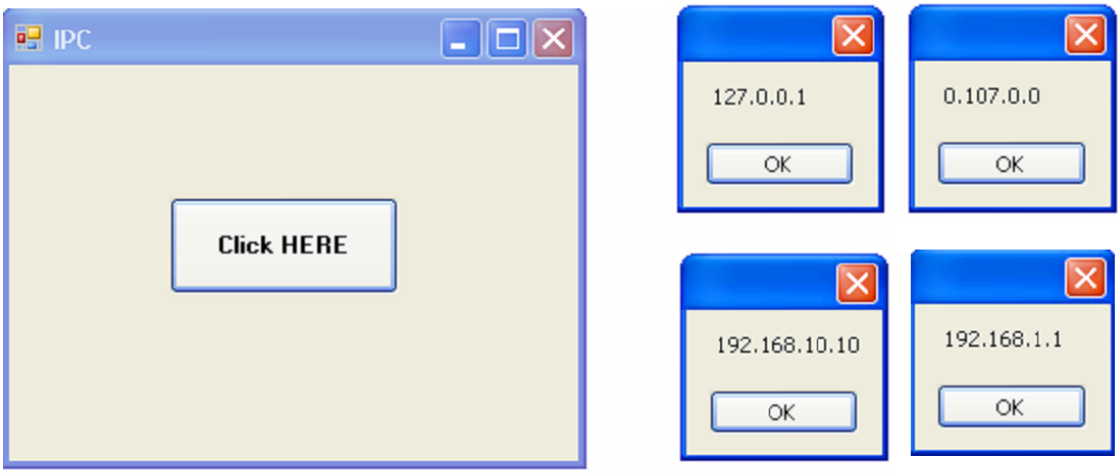
Trong MS.NET, IPAddress là một lớp dùng để mô tả địa chỉ này. Đây là lớp rất cơ bản được sử dụng khi chúng ta thao tác (truyền) vào các lớp như IPEndpoint, UDP, TCP, Socket …

❖ **Ví dụ: Yêu cầu tạo 1 form như hình phía dưới:**



Hình 2.4

Khi Click vào nút Click HERE thì sẽ xuất hiện kết quả như sau:



Hình 2.5

Kết quả thể hiện trên hộp thoại MessageBox lần lượt là các kết quả theo yêu cầu phía dưới đây:

a) Tạo một địa chỉ IP (Tạo một đối tượng IPAddress) có giá trị là 16885952.
b) Tạo một địa chỉ IP từ một mảng byte tương ứng với địa chỉ 192.168.10.10.
c) Tạo một địa chỉ IP từ một xâu.
d) Tạo một địa chỉ 192.168.1.2.

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;

namespace WindowsApp
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void CreateIP()
      {
         Byte[] b = new Byte[4];
         b[0] = 192;
         b[1] = 168;
         b[2] = 10;
         b[3] = 10;
         // Tạo địa chỉ từ các hàm khởi tạo
         IPAddress Ip1 = new IPAddress(b);
         IPAddress Ip2 = new IPAddress(16885952);
         IPAddress Ip3 = IPAddress.Parse("127.0.0.1");
         MessageBox.Show(Ip1.ToString());
         MessageBox.Show(Ip2.ToString());
        MessageBox.Show(Ip3.ToString());
         //Tạo địa chỉ thông qua tính toán
         long Number = 192 * 256 ^ 0 + 168 * 256 ^ 1 + 1 * 256 ^ 2 + 2 * 256 ^ 3;
         IPAddress Ip4 = new IPAddress(Number);
         MessageBox.Show(Ip4.ToString());
      }

      private void btnClick_Click(object sender, EventArgs e)
      {
         CreateIP();
      }
   }
}
```

❖ **Lưu ý:** Tham số thứ hai là một đối tượng bất kỳ thuộc kiểu IPAddress, do vậy bạn có thể viết New IPAddress(0), IPAddress(1),…

1.4. Lớp IPEndpoint

1.4.1. Giới thiệu

Trong mạng, để hai trạm có thể trao đổi thông tin được với nhau thì chúng cần phải biết được địa chỉ (**IP**) của nhau và số hiệu cổng mà hai bên dùng để trao đổi thông tin.

Lớp IPAddress mới chỉ cung cấp cho ta một vế là địa chỉ IP (**IPAddress**), như vậy vẫn còn thiếu vế thứ hai là số hiệu cổng (**Port number**). Như vậy, lớp **IPEndpoint** chính là lớp chứa đựng cả IPAddress và Port number.

Đối tượng IPEndpoint sẽ được dùng sau này để truyền trực tiếp cho các đối tượng UDP, TCP…

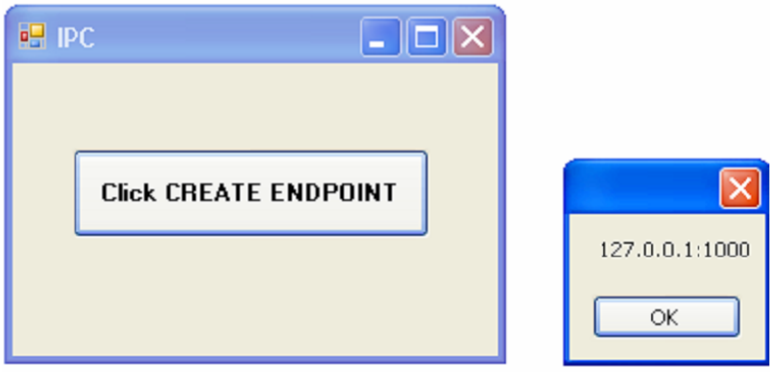
1.4.2. Ví dụ:

Tạo một Endpoint có địa chỉ là "127.0.0.1", cổng là 1000

Để tạo một IPEndpoint, ta có thể dùng 2 hàm thiết lập, trong đó có một hàm thiết lập đòi hỏi phải truyền một đối tượng IPAddress vào. Khi đó chúng ta cần phải tạo đối tượng IPAddress trước theo các cách như đã đề cập.

Tạo Form như hình bên dưới:

Khi bấm vào nút **Click CREATE ENDPOINT** thì kết quả sẽ xuất hiện trong **MessageBox** như hình bên dưới.



Hình 2.6

Bài giải:

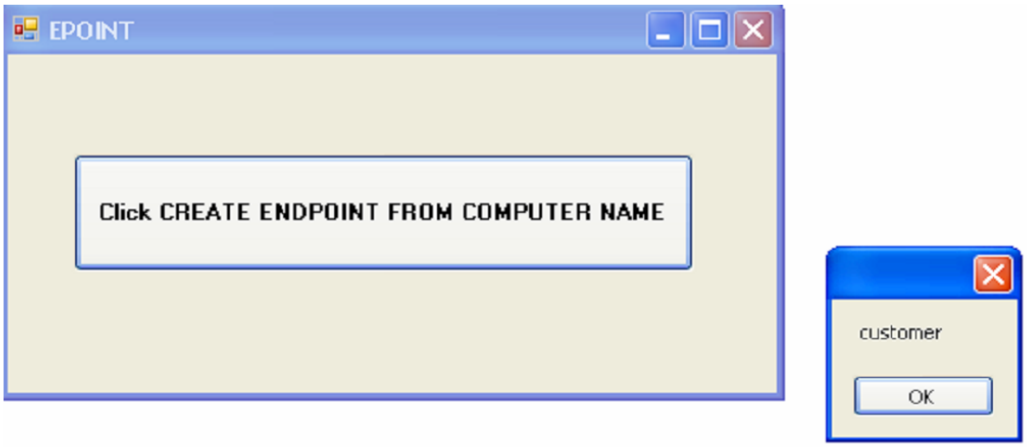
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;

namespace WindowsApplication5
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void CreateEndpoint()
```

```
        {
            // Tạo một đại chỉ IP
            IPAddress IpAdd = IPAddress.Parse("127.0.0.1");
            // Truyền vào cho hàm khởi tạo để tạo IPEndPoint
            IPEndPoint IPep = new IPEndPoint(IpAdd,1000);
            MessageBox.Show(IPep.ToString());
        }
        private void btnClick_Click(object sender, EventArgs e)
        {
            CreateEndpoint();
        }
    }
}
```

Tạo một EndPoint từ tên máy: Ta cũng có thể tạo đối tượng IPAddress từ tên của máy thông qua **phương thức tĩnh** DNS.GetHostAddresses của lớp DNS. Sau đó truyền đối tượng IP này vào cho phương thức khởi tạo của IPEndPoint để tạo đối tượng IPEndpoint mới.

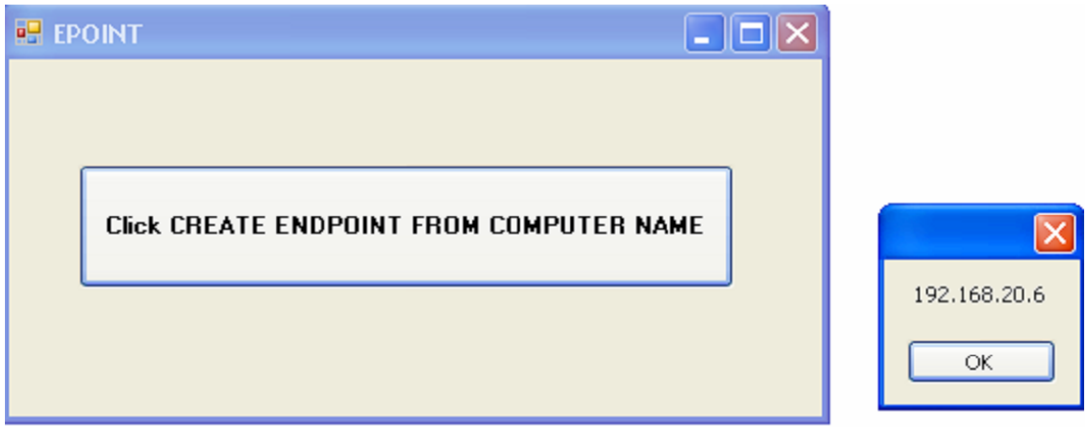
Tạo Form như hình bên dưới:

Khi bấm vào nút **Click CREATE ENDPOINT COMPUTER NAME** thì kết quả sẽ xuất hiện trong **MessageBox** như hình bên dưới.



Hình 2.7

Khi nhấn vào nút OK thì kết quả sẽ như sau:



Hình 2.8

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
namespace WindowsApp
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      public static void DoGetHostAddresses(string hostname)
      {
         IPAddress[] ips;
         ips = Dns.GetHostAddresses(hostname);
         MessageBox.Show(hostname);
         foreach (IPAddress ip in ips)
         {
           MessageBox.Show(ip.ToString());
         }
      }
      private void btnCreatENDPOINT_Click(object sender, EventArgs e)
      {
         DoGetHostAddresses("customer");
      }
      }
   }
```

*** Lưu ý:**

Vì một máy tính có thể có nhiều Card mạng (Interface) do vậy có thể có nhiều hơn 1 địa chỉ IP. Hàm GetHostAddresses sẽ trả về cho ta một **mảng chứa tất cả các địa chỉ** đó.

1.5. Lớp IPHostEntry

1.5.1. Giới thiệu:

IPHostEntry là lớp chứa (Container) về thông tin địa chỉ của các máy trạm trên Internet.

Lưu ý: Nó chỉ là nơi để "chứa", do vậy trước khi sử dụng cần phải "Nạp" thông tin vào cho nó. Lớp này rất hay được dùng với lớp **DNS.**

	NAME	DESCRIPTION
	AddressList	Lấy hoặc đặt địa chỉ IP của một host.
	Aliases	Lấy hoặc đặt aliases của một host.
	HostName	Lấy hoặc đặt tên DNS của host.

1.6. Lớp DNS:

1.6.1. Giới thiệu

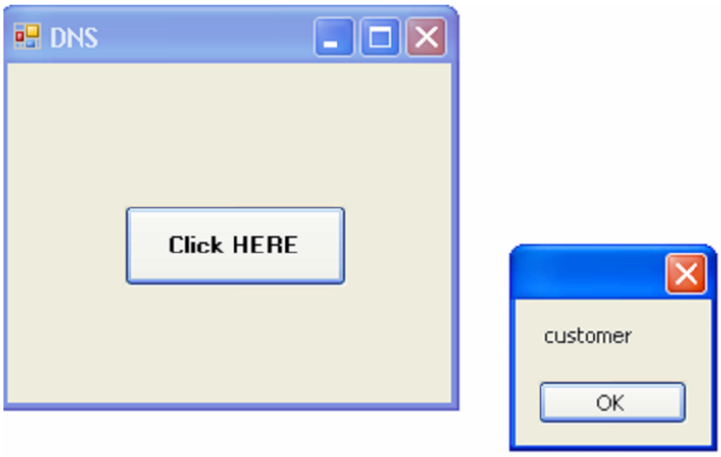
DNS (Domain Name Service) là một lớp giúp chúng ta trong việc phân giải tên miền (Domain Resolution) đơn giản. (Phân giải tên miền tức là: Đầu vào là Tên của máy trạm, ví dụ ServerCNTT thì đầu ra sẽ cho ta địa chỉ IP tương ứng của máy đó, ví dụ 192.168.3.8). Ngoài ra lớp DNS còn có rất nhiều phương thức cho ta thêm thông tin về máy cục bộ như tên, địa chỉ v.v…

1.6.2. Ví dụ:

a) Hiển thị tên của máy tính hiện hành:

```
MessageBox.Show(Dns.GetHostName());
```

Ta tạo Form như hình bên dưới:



Hình 2.9

Khi Click vào nút **Click HERE** thì sẽ hiện ra hộp thông báo chứa tên máy tính hiện hành.

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;

namespace WindowsAppDNS
```

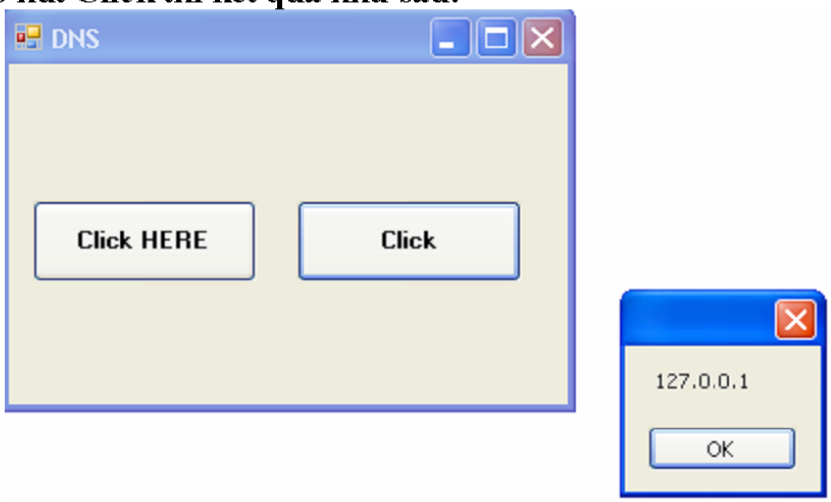
```
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void btnDNS_Click(object sender, EventArgs e)
      {
         MessageBox.Show(Dns.GetHostName());
      }
   }
}
```

b) Hiển thị tất cả địa chỉ IP của một máy nào đó.

Ta tạo Form như hình bên dưới:

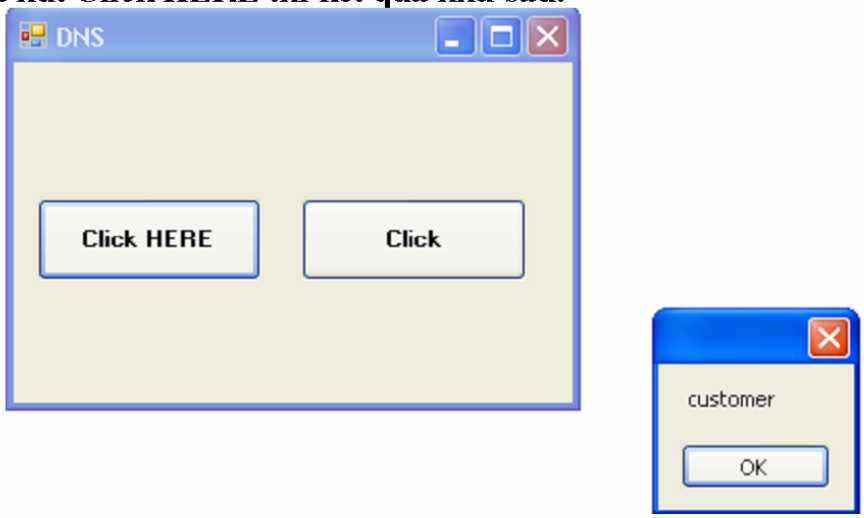
Hình 2.10

Khi bấm vào nút Click thì kết quả như sau:



Hình 2.11

Khi bấm và nút Click HERE thì kết quả như sau:



Hình 2.12

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;

namespace WindowsAppDNS
{
   public partial class Form1 : Form
   {
      public Form1()
```

```
        {
            InitializeComponent();
        }
        private void ShowIPs()
        {
            IPAddress[] add;
            //Lấy tất cả địa chỉ IP của máy customer
            add = Dns.GetHostAddresses("customer");
            //Duyệt sử dụng foreach
            foreach(IPAddress ip in add)
            {
                MessageBox.Show(ip.ToString());
            }
            //Hoặc duyệt theo kiểu mảng
            for (int i = 0; i < add.Length - 1; i++)
            {
                MessageBox.Show(add[i].ToString());
            }
        }
        private void btnDNS_Click(object sender, EventArgs e)
        {
            MessageBox.Show(Dns.GetHostName());
        }
        private void btnShowIPs_Click(object sender, EventArgs e)
        {
            ShowIPs();
        }
    }
}
```

c) Tạo một IPHostEntry từ máy có tên là " customer "

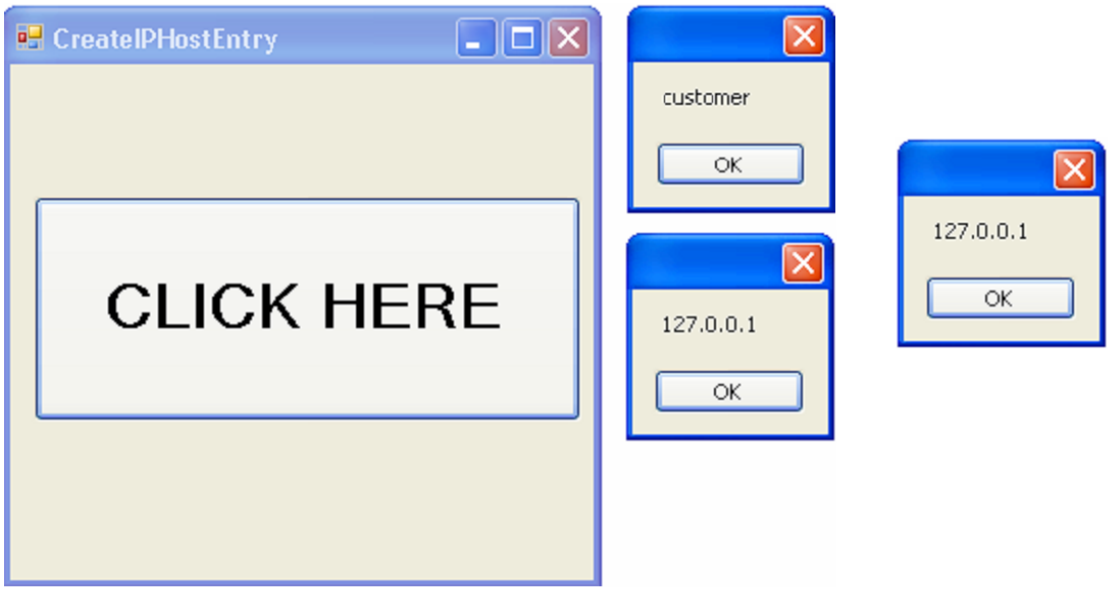
d) Tạo một IPHostEntry từ địa chỉ "127.0.0.1"

e) Tạo một IPHostEntry từ một đối tượng IPAddress, có địa chỉ IP là 127.0.0.1

Ta tạo Form như hình bên dưới:

Hình 2.13

Khi bấm vào nút CLICK HERE thì sẽ thể hiện kết quả như sau:



Hình 2.14

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;

namespace WindowsApp
{
   public partial class frmCreateIPHostEntry : Form
   {
      public frmCreateIPHostEntry()
      {
         InitializeComponent();
      }
      private void CreateIPHostEntry()
      {
         IPHostEntry iphe1, iphe2, iphe3;
         IPAddress ipadd = IPAddress.Parse("127.0.0.1");
         iphe1 = Dns.GetHostEntry("customer");
         iphe2 = Dns.GetHostEntry("127.0.0.1");
         iphe3 = Dns.GetHostEntry(ipadd);
         MessageBox.Show(iphe1.HostName);
         MessageBox.Show(iphe2.HostName);
         MessageBox.Show(iphe3.HostName);
      }
```

```
        private void btnCreateIPHostEntry_Click(object sender, EventArgs e)
        {
            CreateIPHostEntry();
        }
    }
}
```

*** Lưu ý**: Đối tượng **IPHostEntry** chúng ta tạo ở trên sẽ được dùng rất nhiều trong các phần sau của bài giảng này.

f)Lấy địa chỉ IP của máy tính hiện hành:

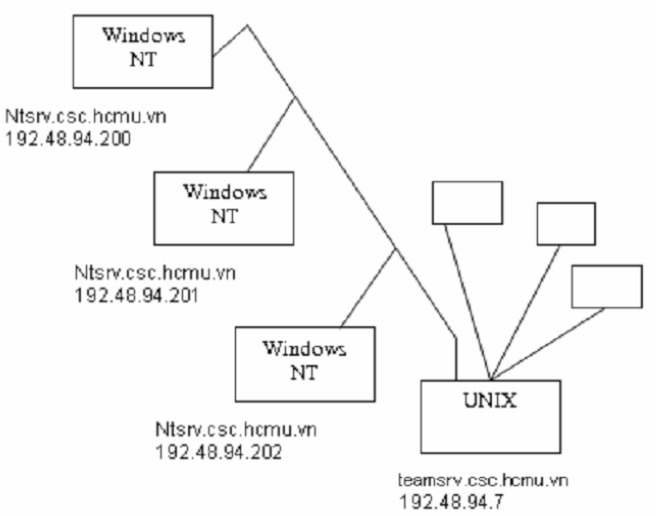
```
using System;
using System.Net;
public class GetIPAddress
{
private static void Main()
 {
        // Lấy tên host của máy tính hiện hành.
        string hostName = Dns.GetHostName();
        // Lấy địa chỉ IP trùng khớp đầu tiên.
        string ipAddress =
        Dns.GetHostByName(hostName).AddressList[0].ToString();
        Console.WriteLine("Host name: " + hostName);
        Console.WriteLine("IP address: " + ipAddress);
        Console.ReadLine();
}
}
```

2. Lập trình với giao thức TCP/IP:

2.1 Giao thức TCP/IP

- Giao thức TCP/IP được dùng trong mạng Unix và Internet. Vậy với kỹ thuật này các máy trên Windows NT có thể giao tiếp với các máy trên mạng khác(khai thác các dịch vụ mạng Unix và Internet)
- Các ứng dụng khai thác giao thức TCP/IP thông qua một thư viện tên là Windows socket.
- Địa chỉ Internet

  Trên hệ thống mạng Internet mỗi máy đều có một tên và một địa chỉ IP . địa chỉ IP đều xác định duy nhất một máy trong hệ thống mạng Internet. Khi lập trình có thể chuyển đổi từ tên sang địa chỉ IP và ngược lại.



Hình 2.18

- Giao thức truyền thông TCP/IP cung cấp cả 2 cơ chế bắt cặp(point to point) và broadcast.
- Nghi thức truyền thông TCP/IP cung cấp cho người lập trình nhiều nghi thức ứng dụng khác nhau. Các nghi thức ứng dụng này chia làm 2 loại: nghi thức chuẩn và không chuẩn. Các nghi thức ứng dụng chuẩn được dùng cho các phần mềm khác nhau dùng để khai thác hay cung cấp các dịch vụ thông dụng (như FTP, SMTP...)
- Các Giao thức không chuẩn do người lập trình tự đặt ra và chỉ có ý nghĩa cục bộ với ứng dụng tự xây dựng.
- Các giao thức ứng dụng chuẩn sử dụng các port được định nghĩa trước dưới dạng các hằng số từ 0 đến 999, trong khi các nghi thức ứng dụng không chuẩn được phép sử dụng các port tùy ý từ 1000 đến 64000.
- *TCP* là một giao thức đáng tin cậy dựa-trên-kết-nối, cho phép hai máy tính giao tiếp thông qua một network.
- Để tạo một kết nối *TCP*, một máy tính phải đóng vai trò là server và bắt đầu lắng nghe trên một endpoint cụ thể (endpoint được định nghĩa là một địa chỉ *IP*, cho biết máy tính và số port).
- Một máy tính khác phải đóng vai trò là client và gửi một yêu cầu kết nối đến endpoint mà máy tính thứ nhất đang lắng nghe trên đó.
- Một khi kết nối được thiết lập, hai máy tính có thể trao đổi các thông điệp với nhau.
- Cả hai máy tính chỉ đơn giản đọc/ghi từ một System.Net.Sockets.NetworkStream.
- Một máy tính (server) phải lắng nghe bằng lớp System.Net.Sockets.TcpListener.
- Mỗi khi một kết nối được thiết lập, cả hai máy tính đều có thể giao tiếp bằng lớp System.Net.Sockets.TcpListener.

2.2 Phân giải tên miền thành địa chỉ IP:

- Trên web, các địa chỉ *IP* có thể truy xuất công khai thường được ánh xạ đến tên miền để dễ nhớ hơn. Ví dụ, địa chỉ *207.171.185.16* được ánh xạ đến tên miền *www.amazon.com*.
- Để xác định địa chỉ *IP* khi có tên miền, máy tính cần liên lạc với một *DNS*-server.
- Quá trình phân giải tên miền được thực hiện một cách trong suốt khi sử dụng lớp System.Net.Dns.
- Lớp này cho phép lấy địa chỉ *IP* của một tên miền bằng phương thức GetHostByName.
- Dưới đây là đoạn mã trình bày cách lấy danh sách các địa chỉ *IP* được ánh xạ đến tên miền *www.microsoft.com*.

```
using System;
using System.Net;
public class ResolveIP{
private static void Main() {
     foreach (IPAddress ip in
    Dns.GetHostByName("www.microsoft.com").AddressList) {
            Console.Write(ip.AddressFamily.ToString() + ": ");
            Console.WriteLine(ip.ToString());
    }
            Console.ReadLine();
}
}
```

2.3 Cách tạo Socket:

a) Unix cung cấp một hàm C socket() để tạo ra một socket mới:

int socket(int *domain*, int *type*, int *protocol*)

Hàm này trả về một socket dùng để gửi và nhận dữ liệu từ mạng.

+ 3 tham số được dùng như sau:

Domain Value	Description
PF_UNIX	Unix IPC communication
PF_INET	IPv4 Internet protocol, which is the type covered in this book
PF_INET6	IPv6 Internet protocol
PF_IPX	Novell protocol
PF_NETLINK	Kernel user interface driver
PF_X25	ITU-T X.25 /ISO-8208 protocol
PF_AX25	Amateur radio AX.25 protocol
PF_ATMPVC	Access to raw ATM PVC's
PF_APPLETALK	AppleTalk protocol
PF_PACKET	Low-level packet interface

Bảng giá trị cho tham số Domain

Có 2 giá trị type phổ biến nhất được dùng cho truyền thông IP là SOCK_STREAM
(kiểu truyền thông hướng nối kết) và SOCK_DGRAM(kiểu truyền thông không hướng nối kết)

Type Value	Description
SOCK_STREAM	Uses connection-oriented communication packets
SOCK_DGRAM	Uses connectionless communication packets
SOCK_SEQPACKET	Uses connection-oriented packets with a fixed maximum length
SOCK_RAW	Uses raw IP packets
SOCK_RDM	Uses a reliable datagram layer that does not guarantee packet ordering

Bảng giá trị cho kiểu Socket:

Tham số thứ 3 là protocol: Giá trị của tham số này được dùng để tạo socket phụ thuộc vào giá trị của tham số type mà bạn chọn.

• Giá protocol mặc định = 0.

• **Ví dụ:**

int *newsocket;*

newsocket = socket(PF_INET, SOCK_STREAM, 0);

❖ Trong ví dụ này tạo ra một socket TCP chuẩn để truyền dữ liệu tới một host từ xa.

b) Tạo Socket trong C#:

System.Net.Sockets namespace là lớp Socket.

Nguyên mẫu hàm tạo ra Socket như sau:

Socket(AddressFamily *af*, SocketType *st*, ProtocolType *pt*)

– *AddressFamily:* định rõ loại mạng.

– *SocketType :* xác rõ loại dữ liệu nối kết.

– *ProtocolType*: định rõ Protocol mạng.

❖ Mỗi tham số được biểu diễn bởi danh sách liệt kê riêng trong System.Net.Sockets namespace.

❖ Thông thường giá trị AddressFamily.InterNetwork nên được dùng cho AddressFamily với giá trị này thì tham số *SocketType* phải khớp với tham số *ProtocolType.*

❖ ***Bảng mô tả như sau:***

SocketType	Protocoltype	Description
Dgram	Udp	Connectionless communication
Stream	Tcp	Connection-oriented -communication
Raw	Icmp	Internet Control Message Protocol
Raw	Raw	Plain IP packet communication

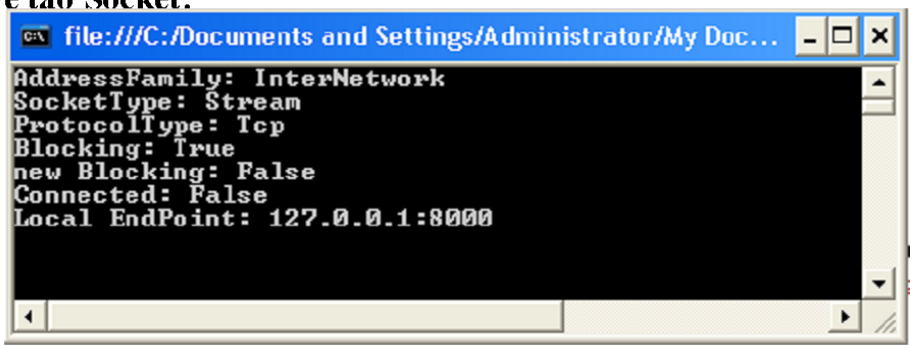
Ví dụ tạo Socket như sau:

Socket newsock =Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

❖ **Một vài thuộc tính của Socket:**

Property	Description
AddressFamily	Gets the address family of the Socket
Available	Gets the amount of data that is ready to be read
Blocking	Gets or sets whether the Socket is in blocking mode
Connected	Gets a value that indicates if the Socket is connected to a remote device
Handle	Gets the operating system handle for the Socket
LocalEndPoint	Gets the local EndPoint object for the Socket
ProtocolType	Gets the protocol type of the Socket
RemoteEndPoint	Gets the remote EndPoint information for the Socket
SocketType	Gets the type of the Socket

Ví dụ về tao Socket:




```
using System.Text;
using System.Threading;
using System.Net;
using System.Net.Sockets;
class SockProp{
   public static void Main(){
        IPAddress ia = IPAddress.Parse("127.0.0.1");
        IPEndPoint ie = new IPEndPoint(ia, 8000);
        Socket test = new Socket(AddressFamily.InterNetwork,
                                        SocketType.Stream, ProtocolType.Tcp);
        Console.WriteLine("AddressFamily: {0}", test.AddressFamily);
       Console.WriteLine("SocketType: {0}", test.SocketType);
       Console.WriteLine("ProtocolType: {0}", test.ProtocolType);
        Console.WriteLine("Blocking: {0}", test.Blocking);
        test.Blocking = false;
```

```
        Console.WriteLine("new Blocking: {0}", test.Blocking);
        Console.WriteLine("Connected: {0}", test.Connected); test.Bind(ie);
        IPEndPoint iep = (IPEndPoint)test.LocalEndPoint;
        Console.WriteLine("Local EndPoint: {0}", iep.ToString()); test.Close();
        Console.ReadKey();
    }
}
```

2.4 Lớp UDP

2.4.1 Giới thiệu

Giao thức **UDP** (User Datagram Protocol hay User Define Protocol) là một giao thức phi kết nối (Connectionless) có nghĩa là một bên có thể gửi dữ liệu cho bên kia mà không cần biết là bên đó đã sẵn sàng hay chưa? (Nói cách khác là không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin). Giao thức này không tin cậy bằng giao thức **TCP** nhưng tốc độ lại nhanh và dễ cài đặt. Ngoài ra, với giao thức **UDP** ta còn có thể gửi các gói tin quảng bá (Broadcast) cho đồng thời nhiều máy.

Trong .NET, lớp **UDPClient** (nằm trong System.Net.Sockets) đóng gói các chức năng của giao thức UDP.

2.4.2 Các thành phần của lớp UDPClient

a) Phương thức khởi tạo:

Constructor methods	Description(mô tả)
UdpClient ()	Tạo một đối tượng (thể hiện) mới của lớp UDPClient.
UdpClient (AddressFamily)	Tạo một đối tượng (thể hiện) mới của lớp UDPClient. Thuộc một dòng địa chỉ (AddressFamily) được chỉ định.
UdpClient (LocalPort: Int32)	Tạo một **UdpClient** và gắn (bind) một cổng cho nó.
UdpClient (IPEndPoint)	Tạo một **UdpClient** và gắn (bind) một IPEndpoint (gán địa chỉ IP và cổng) cho nó.
UdpClient (Int32, AddressFamily)	Tạo một **UdpClient** và gán số hiệu cổng,AddressFamily
UdpClient(Remotehost : String,Int32)	Tạo 1 **UdpClient** và thiết lập với một trạm từ xa mặc định.

b) Các phương thức:

PUBLIC METHOD

Name		Description
	BeginReceive	Nhận dữ liệu không đồng bộ từ máy ở xa.
	BeginSend	Gửi không đồng bộ dữ liệu tới máy ở xa
	Close	Đóng kết nối.
	Connect	Thiết lập một Default remote host.
	EndReceive	Kết thúc nhận dữ liệu không đồng bộ ở trên
	EndSend	Kết thúc việc gửi dữ liệu không đồng bộ ở trên
	Receive	**Nhận dữ liệu (đồng bộ) do máy ở xa gửi.** Đồng bộ có nghĩa là các lệnh ngay sau lệnh Receive chỉ được thực thi nếu Receive đã nhận được dữ liệu về. Còn nếu nó chưa nhận được(dù chỉ một chút) thì nó vẫn cứ chờ (blocking).
	Send	**Gửi dữ liệu (đồng bộ) cho máy ở xa.**

- Đồng bộ: Synchronous.
- Không đồng bộ: Asynchronous.

2.4.3 Ví dụ:

+ Chuyển đổi một chuỗi kí tự sang mảng byte:

```
Byte[] msg;
msg = Encoding.UTF8.GetBytes(" Hello World!!! ");
```

+ Chuyển đổi mảng byte sang chuỗi ký tự:

```
string S = Encoding.UTF8.GetString(msg);
```

a) Ví dụ 1:

Tạo Form như hình vẽ bên dưới: Khi bấm vào nút CONVERT thì xuất hiện hộp thông báo như hình.

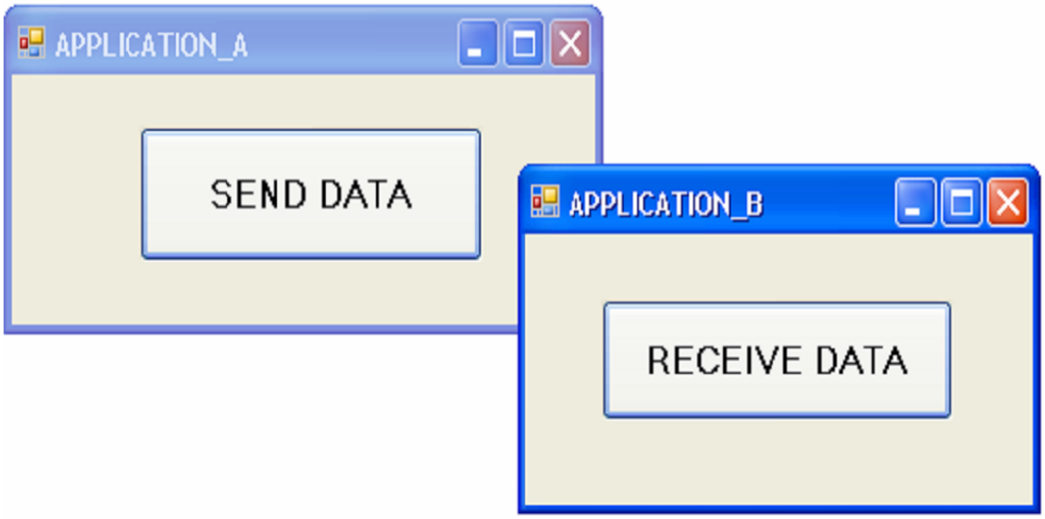
Hình 2.15

Bài giải:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

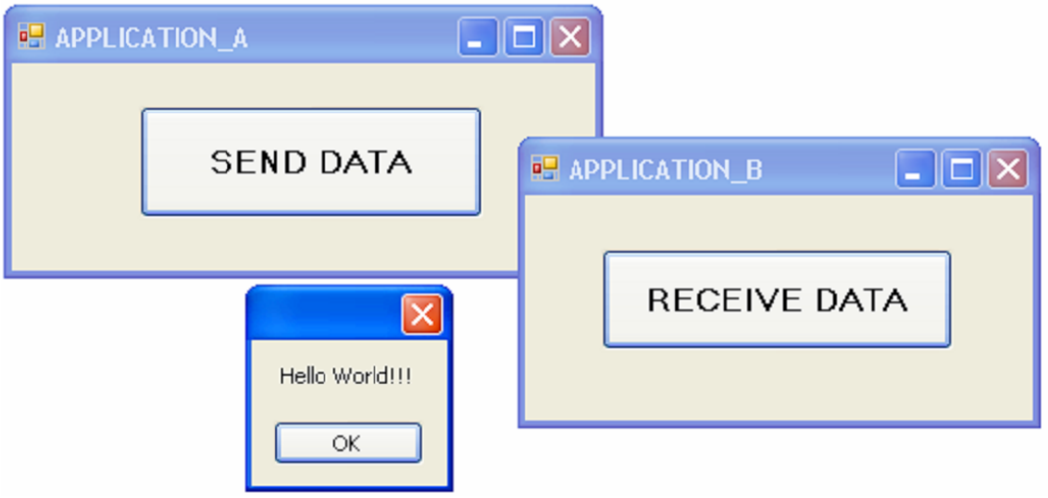
```
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
namespace Demo
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void btnConvert_Click(object sender, EventArgs e)
      {
         Byte[] msg;
         msg = Encoding.UTF8.GetBytes(" Hello World!!! ");
         string S = Encoding.UTF8.GetString(msg);
         MessageBox.Show(S);
      }
   }
}
```

b) Ví dụ 2: Tạo các Form như hình dưới đây:



Hình 2.16

Trong đó có một Form gửi và một Form nhận dữ liệu. Khi bấm vào nút SEND DATA sau đó bấm vào nút RECEIVE DATA thì kết quả sẽ như sau:



Hình 2.17

Tạo một UDPClient gắn vào cổng 10 và Gửi một gói tin "Hello" tới một ứng dụng UDP khác đang chạy trên máy có địa chỉ là "127.0.0.1" và cổng 1000.

Ứng dụng A:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace WindowsApp1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      const int Local_Port =10;
      const int Remote_Port =1000;
      // Tạo một UDP gắn(Bind)vào cổng 10
      UdpClient App1 = new UdpClient(Local_Port);
      private void SendData()
      {
         Byte[] Msg;
         //Chuyển chuỗi "Hello World!!!" thành mảng byte để gởi đi.
         Msg = Encoding.UTF8.GetBytes("Hello World!!!");
         // Gửi vào cổng 1000 của máy 127.0.0.1
         App1.Send(Msg,Msg.Length,"127.0.0.1",Remote_Port);
```

```
        }
        private void btn_ SendData _Click(object sender, EventArgs e)
        {
            SendData();
        }
    }
}
```

Tạo một UDPClient gắn vào cổng 1000 và nhận dữ liệu từ ứng dụng khác gửi đến.

Ứng dụng B:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace WindowsApp2
{
    public partial class Form1 : Form
    {
        public Form1(){
            InitializeComponent();
        }
        const int Local_Port = 1000;
        const int Remote_Port = 10;
        UdpClient App2 = new UdpClient(Local_Port);
        private void ReceiveData()
        {
            Byte[] Msg;
            // Vì phương thức Receive yêu cầu phải cho biết là nhận từ máy nào
            //(mà đại diện cho một máy là một IPEndPoint) nên trước tiên ta cần phải tạo
            // một IPEndPoint. Ở đây ta muốn lấy về từ máy 127.0.0.1 và RemotePort là 100
            IPEndPoint Ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"),1000);
            Msg = App2.Receive(ref Ipep);
            string S;
            S = Encoding.UTF8.GetString(Msg);
            MessageBox.Show(S);
        }
        private void btn_ ReceiveData _Click(object sender, EventArgs e){
            ReceiveData();
        }
    }
```

}

2.5 Lớp TCP(TCPClient):

2.5.1 Giới thiệu

Mục đích của lớp **UDPClient** ở trên là dùng cho lập trình với giao thức **UDP**, với giao thức này thì hai bên không cần phải thiết lập kết nối trước khi gửi do vậy mức độ tin cậy không cao.

Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta còn dùng một giao thức khác, gọi là giao thức có kết nối: **TCP**(Transport Control Protocol). Trên Internet chủ yếu là dùng loại giao thức này, ví dụ như **Telnet, HTTP, SMTP, POP3…** Để lập trình theo giao thức **TCP**, MS.NET cung cấp hai lớp có tên là **TCPClient** và **TCPListener.**

2.5.2 Các thành phần của lớp:

a) Phương thức khởi tạo:

CONSTRUCTOR METHOD	
NAME	**DESCRIPTION**
TcpClient ()	Tạo một đối tượng TcpClient. Chưa đặt thông số gì.
TcpClient(IPEndPoint)	Tạo một TcpClient và gắn cho nó một **EndPoint** cục bộ.(Gán địa chỉ máy cục bộ và số hiệu cổng để sử dụng trao đổi thông tin về sau).
TcpClient(RemoteHost: String, Int32)	Tạo một đối tượng **TcpClient** và kết nối đến một máy có địa chỉ và số hiệu cổng được truyền vào.. **RemoteHost** có thể là địa chỉ **IP** chuẩn hoặc tên máy.

b)Một số thuộc tính:

PROPERTIES	
NAME	**DESCRIPTION**
Available	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc.
Client	Trả về Socket ứng với **TCPClient** hiện hành.
Connected	Trạng thái cho biết đã kết nối được đến **Server** hay chưa?

c) Một số phương thức:

METHOD	
NAME	**DESCRIPTION**
Close	Giải phóng đối tượng **TcpClient** nhưng không đóng kết nối.
Connect(RemoteHost,Port)	Kết nối đến một máy TCP khác có Tên và số hiệu cổng.
GetStream	Trả về **NetworkStream** để từ đó giúp ta gửi hay nhận dữ liệu(Thường làm tham số khi tạo StreamReader và StreamWriter để gửi và nhận dữ liệu dưới dạng xâu ký tự). Khi đã gắn vào StreamReader và StreamWriter rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readline, writeline tương ứng của các lớp này.

Từ các thành viên của lớp **TCPClient** ở trên ta thấy rằng, việc kết nối và thực hiện gửi nhận **theo các trình tự sau:**

- **Bước 1:** Tạo một đối tượng TCPClient.
- **Bước 2:** Kết nối đến máy chủ (Server) dùng phương thức **Connect.**
- **Bước 3:** Tạo 2 đối tượng StreamReader (Receive)và StreamWriter (Send) và **"nối"** với GetStream của TCPClient.
- **Bước 4:**
  - Dùng đối tượng StreamWriter.Writeline/write vừa tạo ở trên để gửi dữ liệu đi.
  - Dùng đối tượng StreamReader.Readline/Read vừa tạo ở trên để đọc dữ liệu về.
- **Bước 5:** Đóng kết nối.

❖ Nếu muốn gửi/nhận dữ liệu ở mức byte (nhị phân) thì dùng NetworkStream (truyền GetStream cho NetworkStream).

2.5.3 Ví dụ: Ta sử dụng lớp TcpClient viết chương trình DateTimeClient như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
class DateTimeClient
{
static void Main(string[] args)
{
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9999);
        TcpClient client = new TcpClient();
        client.Connect(iep);
```

```
        StreamReader sr = new StreamReader(client.GetStream());
        StreamWriter sw = new StreamWriter(client.GetStream());
         while (true)
        {
        string input = Console.ReadLine();
        sw.WriteLine(input);
        sw.Flush();
        if (input.ToUpper().Equals("QUIT")) break;
        string kq = sr.ReadLine();
        Console.WriteLine(kq);
        }
        sr.Close();
        sw.Close();
        client.Close();
}
}
```

2.6 Lớp TCPListener

2.6.1 Giới thiệu

TCPListerner là một lớp cho phép người lập trình có thể xây dựng các ứng dụng Server (Ví dụ như SMTP Server, FTP Server, DNS Server, POP3 Server hay server tự định nghĩa …). Ứng dụng server khác với ứng dụng Client ở chỗ nó luôn luôn thực hiện lắng nghe và chấp nhận các kết nối đến từ Client.

2.6.2 Các thành phần của lớpTcpListener:

a)Phương thức khởi tạo:

CONSTRUCTOR METHOD	
NAME	**DESCRIPTION**
TcpListener(Port: Int32)	Tạo một **TcpListener** và lắng nghe tại cổng chỉ định.
TcpListener(IPEndPoint)	Tạo một **TcpListener** với giá trị Endpoint truyền vào.
TcpListener (IPAddress,Port: Int32)	Tạo một **TcpListener** và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định.

b) Một số phương thức:

	NAME	DESCRIPTION
	AcceptSocket	Chấp nhận một yêu cầu kết nối đang chờ.
	AcceptTcpClient	Chấp nhận một yêu cầu kết nối đang chờ. (Ứng dụng sẽ

		dừng tại lệnh này cho đến khi nào có một kết nối đến – "Blocking")
	Pending	Cho biết liệu có kết nối nào đang chờ đợi không ? (True = có).
	Start	Bắt đầu lắng nghe các yêu cầu kết nối.
	Stop	Dừng việc nghe.

2.6.3 Ví dụ:

Sử dụng lớp TcpListener ta viết chương trình DateTime Server như sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
class DateTimeServer
{
static void Main(string[] args)
{
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2009);
        TcpListener server = new TcpListener(iep);
        server.Start();
        TcpClient client = server.AcceptTcpClient();
        StreamReader sr = new StreamReader(client.GetStream());
        StreamWriter sw = new StreamWriter(client.GetStream());
        string kq="";
while (true)
{
        string s = sr.ReadLine();
        s=s.ToUpper();
        if (s.Equals("QUIT")) break;
        if (s.Equals("GETDATE"))
        kq=DateTime.Now.ToString("dd/MM/yyyy");
        else
        if (s.Equals("GETTIME"))
        kq=DateTime.Now.ToString("hh:mm:ss");
        else
        kq = "Khong hieu lenh";
        sw.WriteLine(kq);
        sw.Flush();
}
sr.Close();
```

```
sw.Close();
client.Close();
}
}
}
```

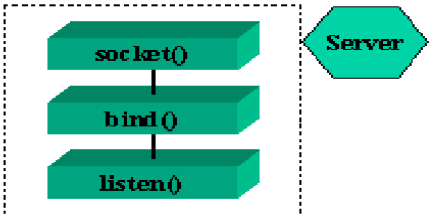
2.7 Xây dựng ứng dụng Client-Server với Socket

Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client- Server. Các ứng dụng trên mạng Internet như Web, Email, FTP là các ví dụ điển hình.

Phần này trình bày các bước cơ bản trong việc xây dựng các ứng dụng Client- Server sử dụng Socket làm phương tiện giao tiếp theo cả hai chế độ: Có nối kết và không nối kết.

2.7.1. Mô hình Client-Server sử dụng Socket ở chế độ có nối kết (TCP)

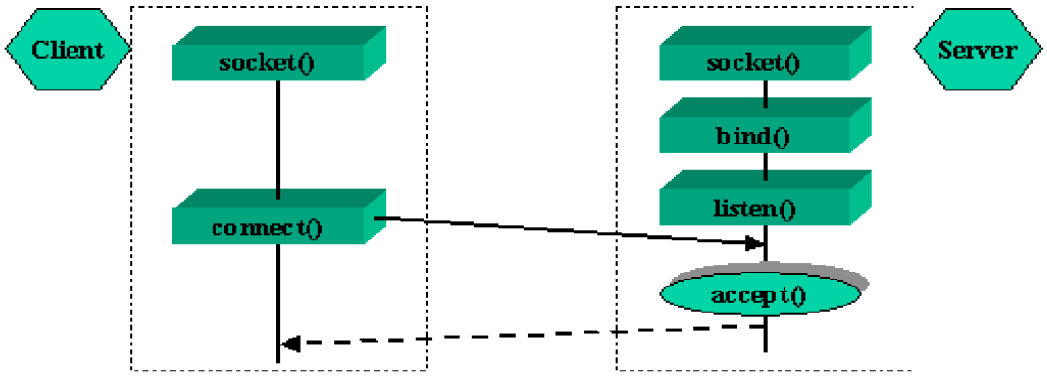
Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết.



- **socket():** Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- **bind():** Server yêu cầu gán số hiệu cổng (port) cho socket.
- **listen():** Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

Server sẵn sàng phục vụ Client.

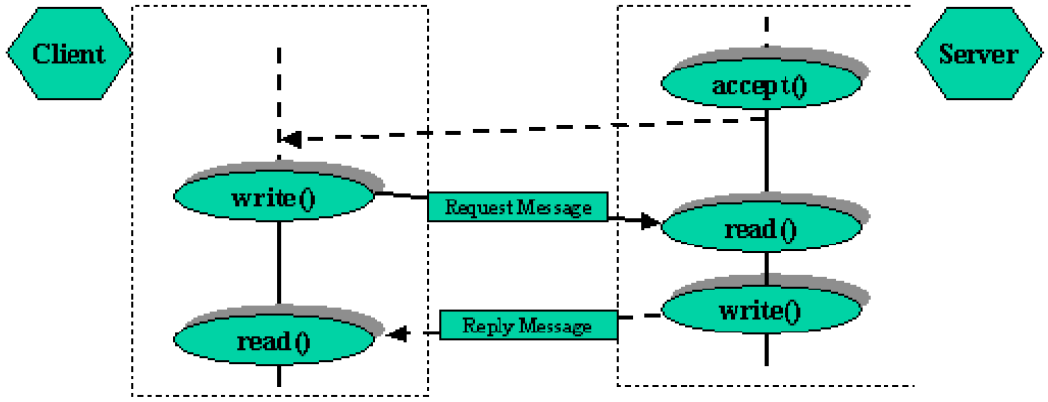
Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.



- **socket():** Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.

- **connect():** Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- **accept():** Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

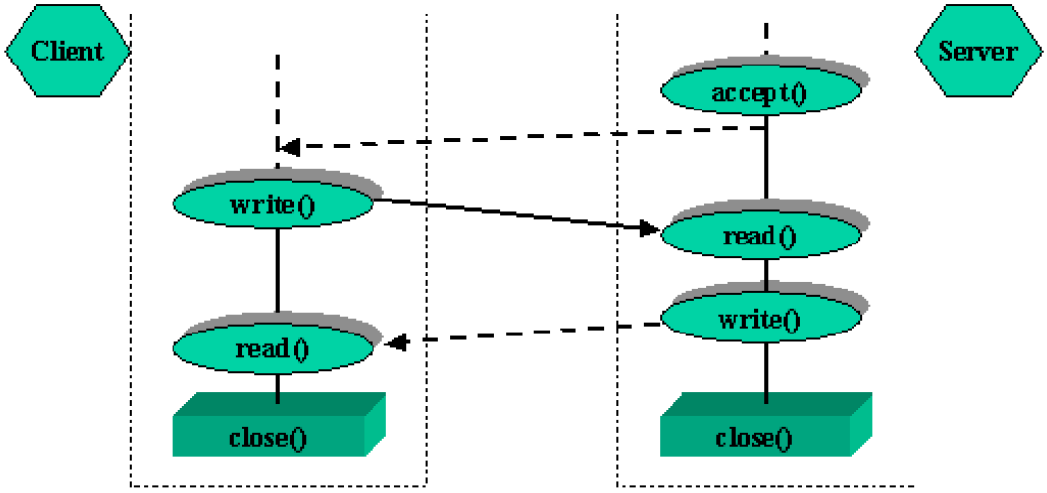


- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
- Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().

Trong giai đoạn này, việc trao đổi thông tin giữa Client và Server phải tuân thủ giao thức của ứng dụng (Dạng thức và ý nghĩa của các thông điệp, qui tắc bắt tay, đồng bộ hóa,...). Thông thường Client sẽ là người gởi yêu cầu đến Server trước.

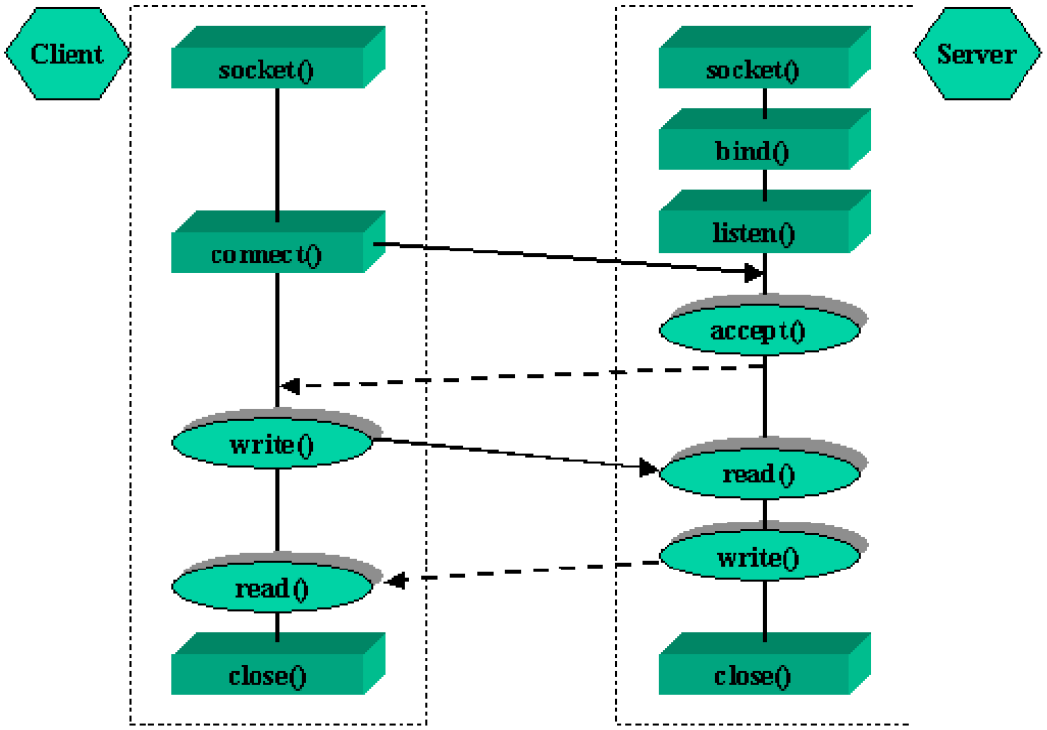
Nếu chúng ta phát triển ứng dụng theo các Protocol đã định nghĩa sẵn, chúng ta phải tham khảo và tuân thủ đúng những qui định của giao thức. Bạn có thể tìm đọc mô tả chi tiết của các Protocol đã được chuẩn hóa trong các tài liệu RFC (Request For Comments). Ngược lại, nếu chúng ta phát triển một ứng dụng Client-Server riêng của mình, thì công việc đầu tiên chúng ta phải thực hiện là đi xây dựng Protocol cho ứng dụng.

Giai đoạn 4: Kết thúc phiên làm việc.

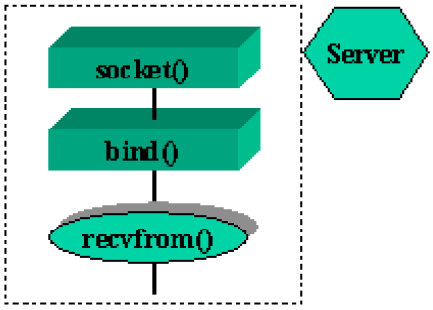


- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

Như vậy toàn bộ tiến trình diễn ra như sau:

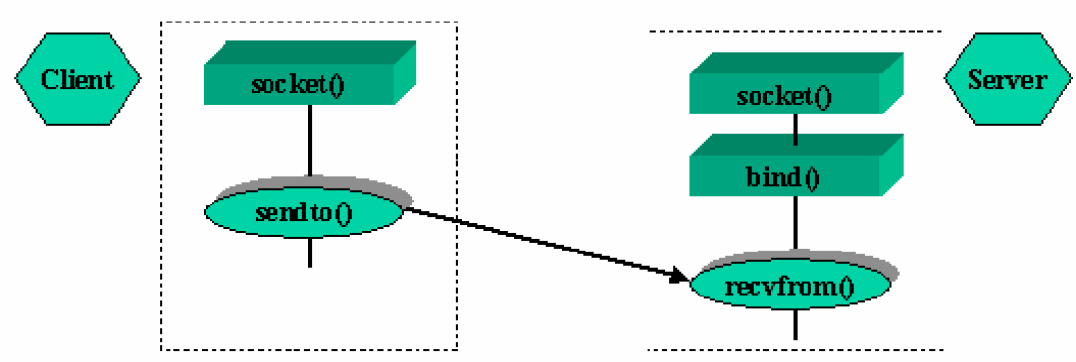


2.7.2. Mô hình Client-Server sử dụng Socket ở chế độ không nối kết (UDP)
Giai đoạn 1: Server tạo Socket - gán số hiệu cổng.

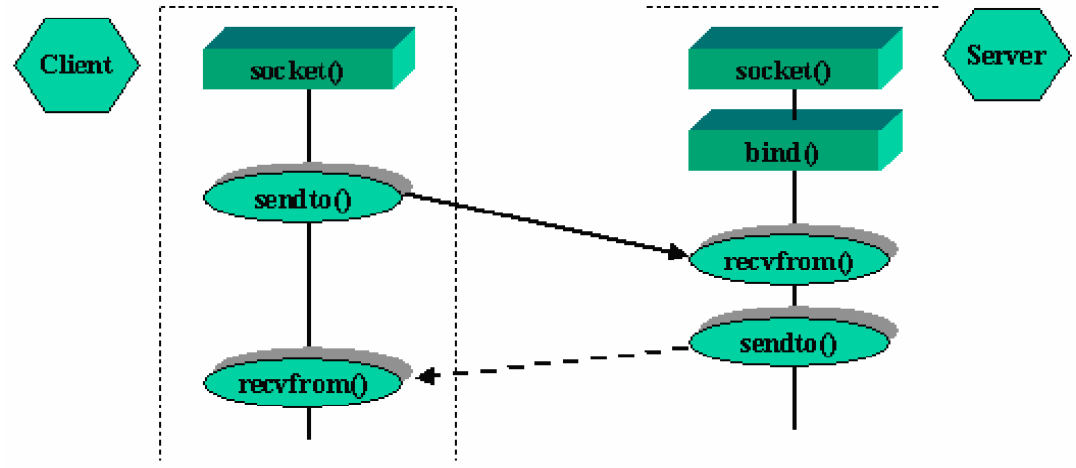


- **socket():** Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- **bind():** Server yêu cầu gán số hiệu cổng cho socket..

Giai đoạn 2: Client tạo Socket.



Giai đoạn 3: Trao đổi thông tin giữa Client và Server.



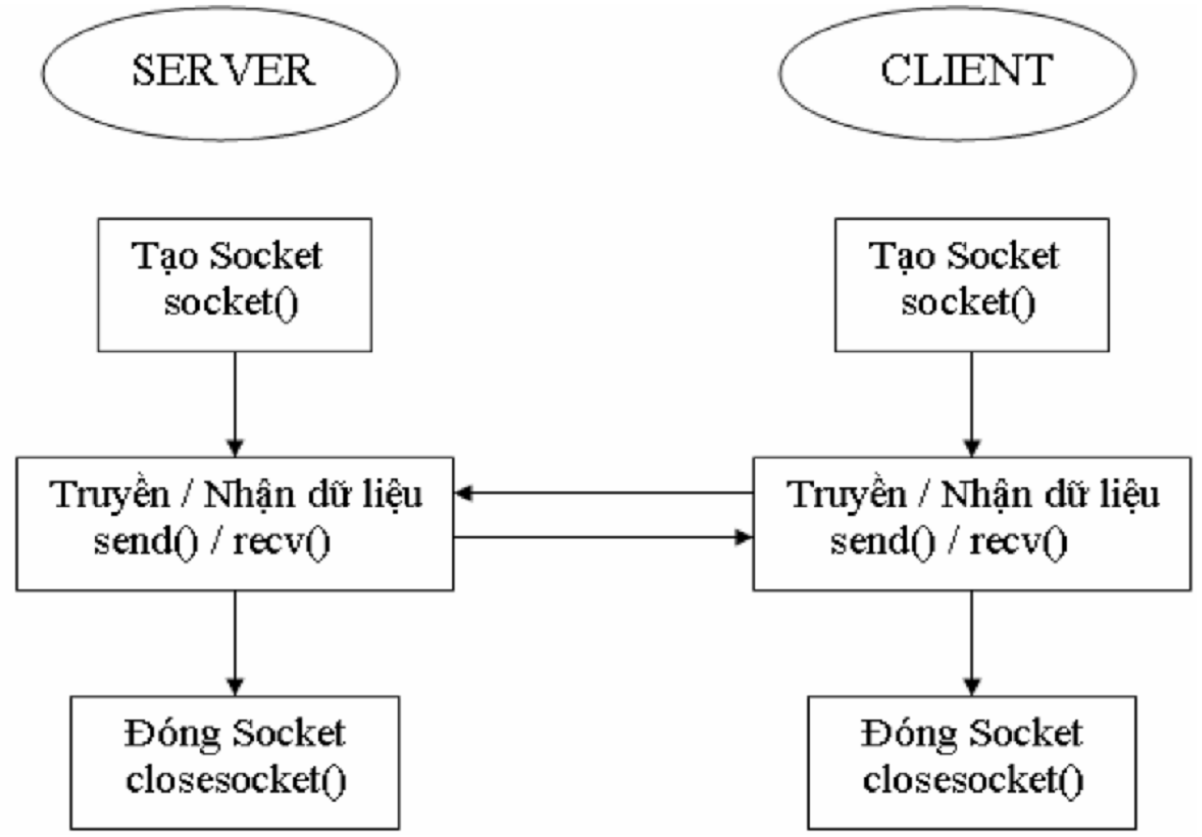
Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm sendto() và recvfrom(). Đơn vị dữ liệu trao đổi giữa Client và Server là các **Datagram Package**(Gói tin thư tín). Protocol của ứng dụng phải

định nghĩa khuôn dạng và ý nghĩa của các Datagram Package. Mỗi Datagram Package có chứa thông tin về địa chỉ người gởi và người nhận(IP, Port).

2.7.3 Ví dụ:

Ví dụ 1: Client gởi đến Server các chuỗi như hình bên dưới:

Ta xem thêm mô hình về sự liên lạc giữa Client và Server sử dụng giao thức UDP:



Các bước thực hiện như sau:

Viết chương trình cho phía máy chủ:

- Tạo một Socket
- Liên kết với một IPEndPoint cục bộ
- Lắng nghe kết nối
- Chấp nhận kết nối
- Gửi nhận dữ liệu theo giao thức đã thiết kế .
- Đóng kết nối sau khi đã hoàn thành và trở lại trạng thái lắng nghe chờ kết nối mới.

Đoạn mã chương trình như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace Chuongtrinhmaychu_Server
{
   class Program
   {
      static void Main(string[] args)
      {
         int recv;
         byte[] data = new byte[1024];
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any,9050);
        // Tạo Socket Server.
         Socket newSock = new Socket(AddressFamily.InterNetwork,
                                          SocketType.Stream, ProtocolType.Tcp);
        // Kết buộc Socket với đối tượng lớp IPEnpoint.
         newSock.Bind(ipep);
        // Bắt đầu lắng nghe các kết nối từ bên ngoài.
         newSock.Listen(10);
         Console.WriteLine("Waiting for a Client...");
        // Chấp nhận Client kết nối và kết quả trả về là một Socket để giao tiếp với
        // Client đã kết nối vào.
         Socket client = newSock.Accept();
         IPEndPoint clientep = (IPEndPoint)client.RemoteEndPoint;
         Console.WriteLine("Connect with {0} a
                                          port{1}",clientep.Address,clientep.Port);
         string welcome = "Welcome to my test Server";
         data = Encoding.UTF8.GetBytes(welcome);
        // Thực hiện gửi dữ liệu.
         client.Send(data,data.Length,SocketFlags.None);
```

```
            while (true)
            {
                data = new byte[1024];
            //  Nhận dữ liệu lưu vào data.
                recv = client.Receive(data);
                if (recv = = 0) break;
                Console.WriteLine(Encoding.ASCII.GetString(data,0,recv));
                client.Send(data,recv,SocketFlags.None);
            }
            Console.WriteLine("Disconnected from {0}",clientep.Address);
           // Đóng nối kết.
            client.Close();
            newSock.Close();
        }
    }
}
```

Viết chương trình cho phía máy khách:

- ❖ Xác định địa chỉ của Server.
- ❖ Tạo Socket.
- ❖ Kết nối đến Server.
- ❖ Gửi nhận dữ liệu theo giao thức đã thiết kế.
- ❖ Đóng Socket.

Đoạn mã chương trình như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace Chuongtrinhmaykhach_Client
{
   class Program
   {
      static void Main(string[] args)
      {
         byte[] data = new byte[1024];
         string input, stringdata;
         IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"),9050);
        // Tạo Socket.
        Socket server = new
        Socket(AddressFamily.InterNetwork,SocketType.Stream,ProtocolType.Tcp);
        // Bắt lỗi trong quá trình kết nối đến Server.
         try
         {
            server.Connect(ipep);
         }
```

```
        catch(SocketException ex)
        {
            Console.WriteLine("Unable to connect to Server.");
            Console.WriteLine(ex.ToString());
            return;
        }
        // Nhận dữ liệu lưu vào trong data.
        int recv = server.Receive(data);
        stringdata = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringdata);
        // Quá trình gửi và nhận dữ liệu từ Server.
        while (true)
        {
            input = Console.ReadLine();
            if (input = = "exit")
                break;
            server.Send(Encoding.ASCII.GetBytes(input));
            data = new byte[1024];
            recv = server.Receive(data);
            stringdata = Encoding.ASCII.GetString(data, 0, recv);
        }
            Console.WriteLine("Disconnecting from Server...");
        // Đóng kết nối.
            server.Shutdown(SocketShutdown.Both);
            server.Close();
    }
  }
}
```

Ví dụ 2: Lập trình dùng lớp hỗ trợ TCPClient và TCPListener:
Chương trình server

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpListenerSample
{
public static void Main()
{
        int recv;
        byte[] data = new byte[1024];
        TcpListener newsock = new TcpListener(9050);
        newsock.Start();
        Console.WriteLine("Waiting for a client...");
        TcpClient client = newsock.AcceptTcpClient();
        NetworkStream ns = client.GetStream();
```

```
        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        ns.Write(data, 0, data.Length);
while (true)
{
        data = new byte[1024];
        recv = ns.Read(data, 0, data.Length);
        if (recv = = 0)
        break;
        Console.WriteLine(
        Encoding.ASCII.GetString(data, 0, recv));
        ns.Write(data, 0, recv);
}
ns.Close();
client.Close();
newsock.Stop();
}
}
```

Chương trình client

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpClientSample
{
public static void Main()
{
        byte[] data = new byte[1024];
        string input, stringData;
        TcpClient server;
try
{
        server = new TcpClient("127.0.0.1", 9050);
}
catch (SocketException){
Console.WriteLine("Unable to connect to server");
return;
}
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
while (true){
        input = Console.ReadLine();
        if (input = = "exit")
        break;
```

```
            ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
            ns.Flush();
            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
}
Console.WriteLine("Disconnecting from server...");
ns.Close();
server.Close();
}
}
```

3. Bài tập(xem trong phần bài tập).

Bài 3. Tạo và sử dụng dịch vụ Windows

Mục tiêu bài:

Nhằm trang bị cho người học:

- Kiến thức về dịch vụ Windows.
- Kiến thức và kỹ năng tạo, quản lý và sử dụng các dịch vụ Windows.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

1.Giới thiệu dịch vụ Windows(Windows services).

Windows Service còn gọi là NT Service, trước đây được tạo bằng C++ sử dụng cho hệ điều hành Windows NT, Windows 2000 và XP.

Windows services là một ứng dụng chạy trên máy **server** hoặc **workstation** và cung cấp những chức năng mà sự diễn tiến của nó không cần sự tương tác trực tiếp của người dùng. Windows services thường được dùng để giám sát hoạt động hệ thống.

Windows services là hữu dụng đối với các ứng dụng phía server mà ta muốn chúng phải luôn ở trạng thái sẵn sàng phục vụ các yêu cầu từ client.

.NET Framework chứa một tập các class, cung cấp các chức năng cơ bản cho việc cài đặt và phát triển dễ dàng các ứng dụng Windows service trên nền **C# .NET.**

Một Windows service sẽ chạy trong tiến trình của riêng nó, không phụ thuộc người dùng hay các chương trình khác đang chạy trên cùng máy tính.

Windows services thường được cấu hình để tự động bắt đầu khi nào máy tính khởi động. Không giống như đa phần các ứng dụng, **Windows services** chạy dưới nhận dạng bảo mật của chính nó, thay vì dưới nhận dạng của đăng nhập người dùng hiện hành. Chúng có thể bắt đầu chạy, ngay cả nếu không có người dùng(**user**) đăng nhập máy tính.

Windows Services được xem như là một loại **ứng dụng Windows**, **không có giao diện**, **chạy thường trú**, và giao tiếp với các ứng dụng khác hoặc người sử dụng thông qua các lời gọi hàm nảy sinh do sự kiện.

Windows Services trao đổi thông tin với môi trường bên ngoài qua cơ chế thông điệp hay ghi chú sự kiện (event log).

Ví dụ:

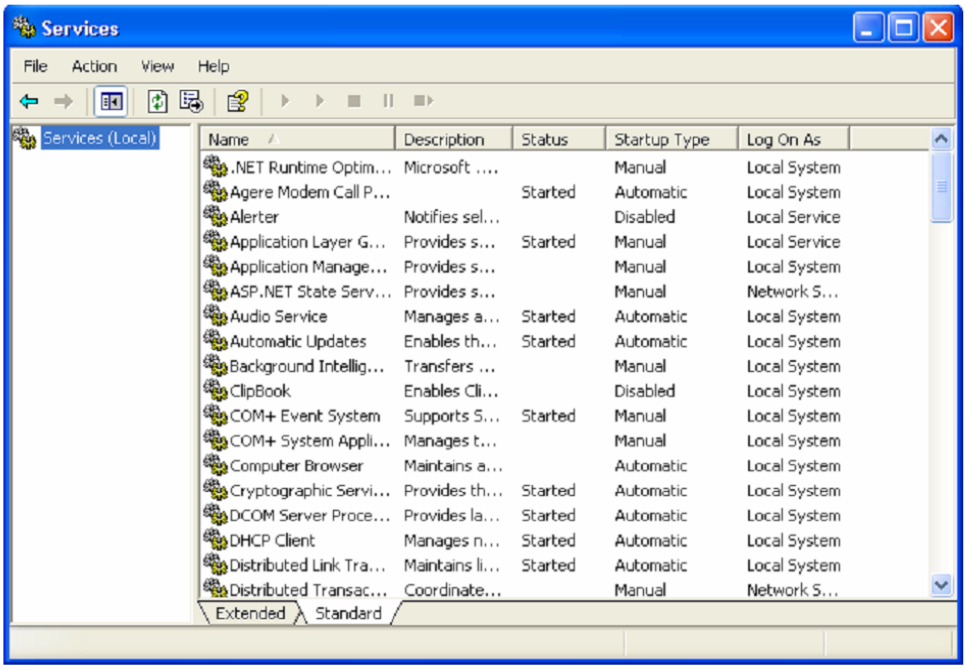
Ta cần xử lý thông tin trên các file được gởi về từ Internet thông qua FTP, thời điểm file được gởi về không được biết trước: cần một service thường trú để kiểm tra luồng file từ FTP, và có hành động xử lý thích hợp.

Người quản trị có thể quản lý tương tác với các ứng dụng **Windows services** bằng việc sử dụng công cụ **Service Control Manager**. Chúng ta có thể tìm thấy công cụ này dưới những thực đơn khác nhau tuỳ thuộc vào hệ điều hành mà chúng ta đang sử dụng:

Ví dụ trong Windows XP Professional chúng ta vào:

Start → Settings → Control Panel → Administrative Tools →Services.

Ta sẽ thấy Service Control Manager trình bày danh sách tất cả các **services** đã được cài đặt trên máy tính như hình bên dưới đây:



Hình 3.1

Đối với mỗi **service**, ta có thể thấy tên(name), mô tả (description), trạng thái hiện hành (current status nó bao gồm 3 trạng thái: Started, Paused, hoặc Stopped), loại khởi động (Automatic sẽ tự động bắt đầu khi khởi động, hoặc Manual) và nhận dạng mà service đó đăng nhập.

2. Tạo các dịch vụ Windows.

Giới thiệu:

Để tạo một dịch vụ *Windows* bằng tay, ta phải hiện thực một lớp dẫn xuất từ ServiceBase.Lớp ServiceBase cung cấp các chức năng cơ bản cho phép *Windows Service Control Manager* (*SCM*) cấu hình dịch vụ, thi hành dịch vụ dưới nền, và điều khiển thời gian sống của dịch vụ.

SCM cũng điều khiển việc các ứng dụng khác có thể điều khiển dịch vụ như thế nào. Lớp ServiceBase được định nghĩa trong System.Serviceprocess, do đó ta phải thêm một chiếu đến assembly này khi xây dựng lớp dịch vụ.

Để tham điều khiển dịch vụ, *SDM* sử dụng bảy phương thức protected thừa kế từ lớp ServiceBase. Ta cần chép đè các phương thức này để hiện thực các chức năng và cách thức hoạt động của dịch vụ.

Không phải tất cả dịch vụ đều hỗ trợ tất cả các thông điệp điều khiển. Các thuộc tính thừa kế từ lớp ServiceBase sẽ báo với *SCM* rằng dịch vụ của bạn hỗ trợ các thông điệp điều khiển nào; thuộc tính điều khiển mỗi kiểu thông điệp được ghi rõ trong bảng bên dưới đây.

Các phương thức dùng để điều khiển sự hoạt động của một dịch vụ

Phương thức	Mô tả
`OnStart`	Tất cả các dịch vụ đều phải hỗ trợ phương thức `OnStart`, *SCM* gọi phương thức này để khởi động dịch vụ. *SCM* truyền cho dịch vụ một mảng kiểu chuỗi chứa các đối số cần thiết. Nếu `OnStart` không trả về trong 30 giây thì *SCM* sẽ không chạy dịch vụ.
`OnStop`	Được *SCM* gọi để dừng một dịch vụ—*SCM* chỉ gọi `OnStop` nếu thuộc tính `CanStop` là `true`.
`OnPause`	Được *SCM* gọi để tạm dừng một dịch vụ—*SCM* chỉ gọi `OnPause` nếu thuộc tính `CanPauseAndContinue` là `true`.
`OnContinue`	Được *SCM* gọi để tiếp tục một dịch vụ bị tạm dừng—*SCM* chỉ gọi `OnContinue` nếu thuộc tính `CanPauseAndContinue` là `true`.
`OnShutdown`	Được *SCM* gọi khi hệ thống đang tắt—*SCM* chỉ gọi `OnShutDown` nếu thuộc tính `CanShutdown` là `true`.
`OnPowerEvent`	Được *SCM* gọi khi trạng thái nguồn mức-hệ-thống thay đổi, chẳng hạn một laptop chuyển sang chế độ suspend. *SCM* chỉ gọi `OnPowerEvent` nếu thuộc tính `CanHandlePowerEvent` là `true`.
`OnCustomCommand`	Cho phép mở rộng cơ chế điều khiển dịch vụ với các thông điệp điều khiển tùy biến; xem chi tiết trong tài liệu *.NET Framework SDK*.

Như được đề cập trong bảng trên, phương thức OnStart phải trả về trong vòng 30 giây, do đó bạn không nên sử dụng OnStart để thực hiện các thao tác khởi động tốn nhiều thời gian.

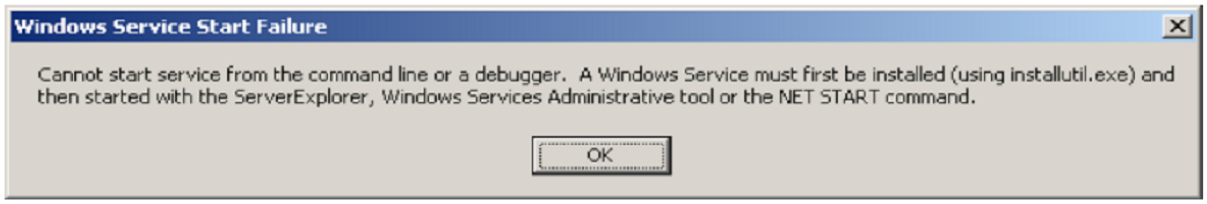
Một lớp dịch vụ nên hiện thực một phương thức khởi dựng để thực hiện các thao tác khởi động, bao gồm việc cấu hình các thuộc tính thừa kế từ lớp ServiceBase. Ngoài các thuộc tính khai báo các thông điệp điều khiển nào được dịch vụ hỗ trợ, lớp ServiceBase còn hiện thực ba thuộc tính quan trọng khác:

- **ServiceName:**Là tên được *SCM* sử dụng để nhận dạng dịch vụ, và phải được thiết lập trước khi dịch vụ chạy.
- **AutoLog:**Điều khiển việc dịch vụ có tự động ghi vào nhật ký sự kiện hay không khi nhận thông điệp điều khiển OnStart, OnStop, OnPause, và OnContinue.
- **EventLog:**Trả về một đối tượng EventLog được cấu hình trước với tên nguồn sự kiện(*event source*) trùng với thuộc tính ServiceName được đăng ký với nhật ký *Application*.

Bước cuối cùng trong việc tạo một dịch vụ là hiện thực phương thức tĩnh Main. Phương thức này phải tạo một thể hiện của lớp dịch vụ và truyền nó cho phương thức tĩnh ServiceBase.Run.

Nếu muốn chạy nhiều dịch vụ trong một tiến trình, bạn phải tạo một mảng các đối tượng ServiceBase và truyền nó cho phương thức ServiceBase.Run.

Mặc dù các lớp dịch vụ đều có phương thức Main nhưng bạn không thể thực thi mã lệnh dịch vụ một cách trực tiếp; bạn sẽ nhận được hộp thông báo như hình dưới(hình 3.2)nếu trực tiếp chạy một lớp dịch vụ. Chúng ta sẽ trình bày cách cài đặt dịch vụ trước khi thực thi ngay ở phần dưới đây.

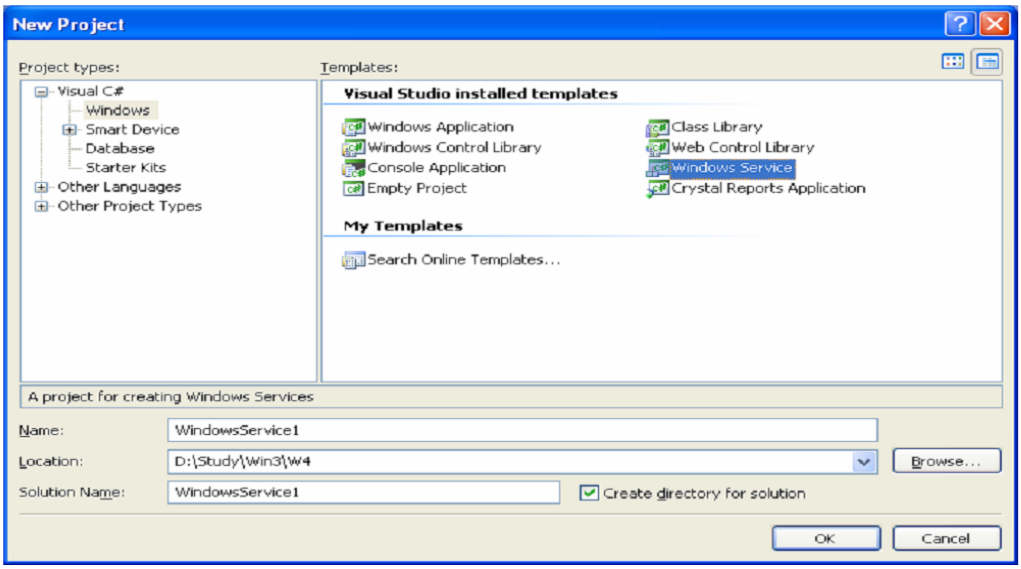


Hình 3.2

2.1 Các bước tạo:

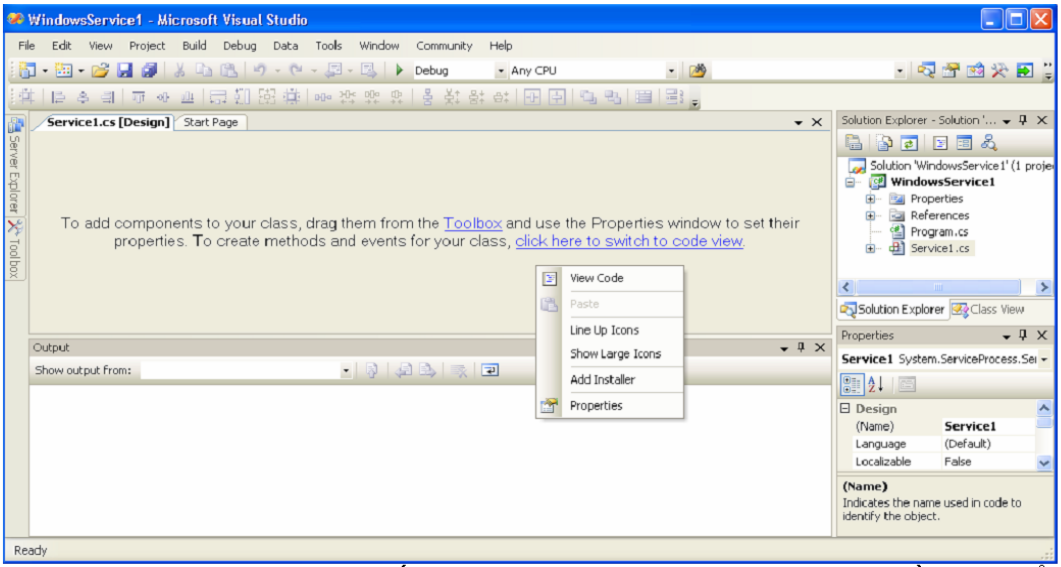
Bước 1: Khởi động **Visual Studio.Net** và tạo một **project** mới với việc sử dụng mẫu **Windows Service**. Sau đó ta đặt tên cho **Project** và lưu nó vào một thư mục. Sau đó ta nhấn OK để tạo.

Ta xem minh họa hình bên dưới:



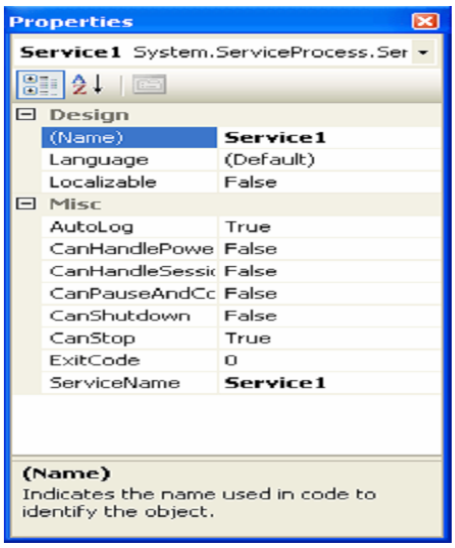
Hình 3.3

Bước 2: Bấm chuột phải vào vùng trống bất kì như hình bên dưới:



Bước 3: Click phải vào vùng trống Design sau đó chọn **Properties** rồi thay đổi những thuộc tính sau trong cửa sổ **Properties**:

- **Name:** Thay đổi tên service (Ví dụ: firstMyWS).
- **ServiceName:** Thay đổi serviceName (ví dụ: firstWSN).



Bước 4: Quay lại chế độ viết code. Viết code cho các overrides Method (OnStart,OnStop,….).

Bước 5: Click chuột phải lên vùng thiết kế chọn Add Installer.

- Click chuột phải lên **ServiceProcessInstaller1** → **Properties** rồi thay đổi thuộc tính: **Account Local system.**

Khi Windows cài đặt một dịch vụ, Windows thực hiện cài đặt theo ngữ cảnh mà đặc quyền có thể sử dụng. Nếu sử dụng **Service Process Installer** mà Visual Studio thêm vào dự án của ta, ta phải chỉ rõ một tài khoản trên hệ thống(tài khoản đặc biệt ta tạo ra cho dịch vụ) dùng cho phép truy cập vào ngữ cảnh của dịch vụ. Trước hết ta chọn kiểu tài khoản như mô tả dưới đây:

Kiểu tài khoản	Mục đích
Local Service	Định hướng Windows chạy dịch vụ trong ngữ cảnh cục bộ có một số đặc quyền ưu tiên.
Network Service	Định hướng Windows chạy dịch vụ trong ngữ cảnh không được ưu tiên sử dụng tài khoản cục bộ.
Local System	Hệ thống cục bộ, tương đương với một tài khoản vô danh đăng nhập sử dụng hệ thống.
User	Định hướng Windows mỗi khi chạy dịch vụ trong ngữ cảnh người dùng đều yêu cầu hỏi nhập trực tiếp username và password.

- Click chuột phải lên **ServiceInstaller1** → **Properties** rồi thay đổi thuộc tính: **StartType** có thể là **Automatic** hoặc **Manual**,… và **ServiceName**: firstWSN.

Bước 6:

Biên dịch chương trình: Build→Build Solution(khi đó trong thư mục bin\debug của Project tạo ra file *.exe).

Bước 7:

Vào Dos install service vào hệ thống

$Home:\>installutil /i YourServiceName.exe

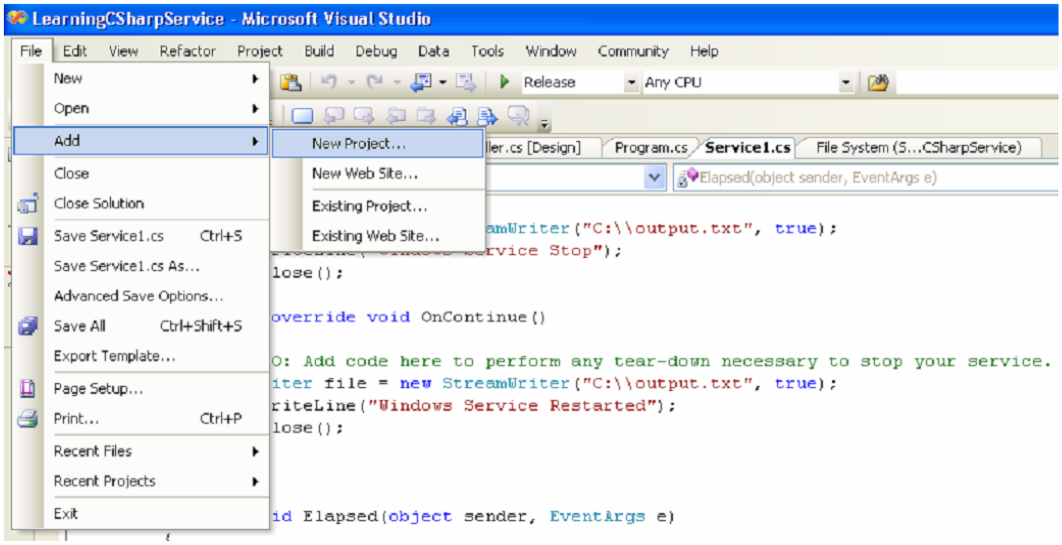
Tháo service ra khỏi

$Home:\>installutil /u YourServiceName.exe

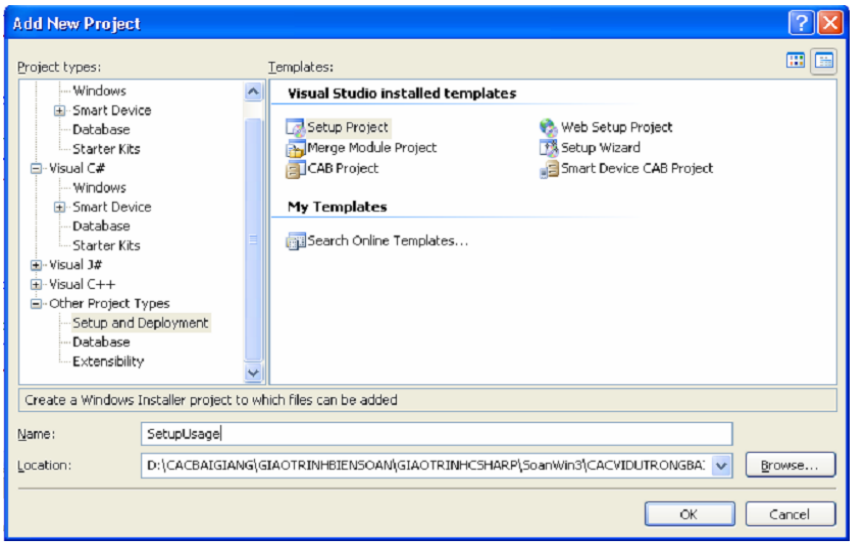
Hay sc delete "YourServiceName.exe"

Thông thường bước này ta ít khi sử dụng dòng lệnh như trên mà thường tạo một trình ứng dụng loại **setup** để cài đặt Service các bước tạo như sau:

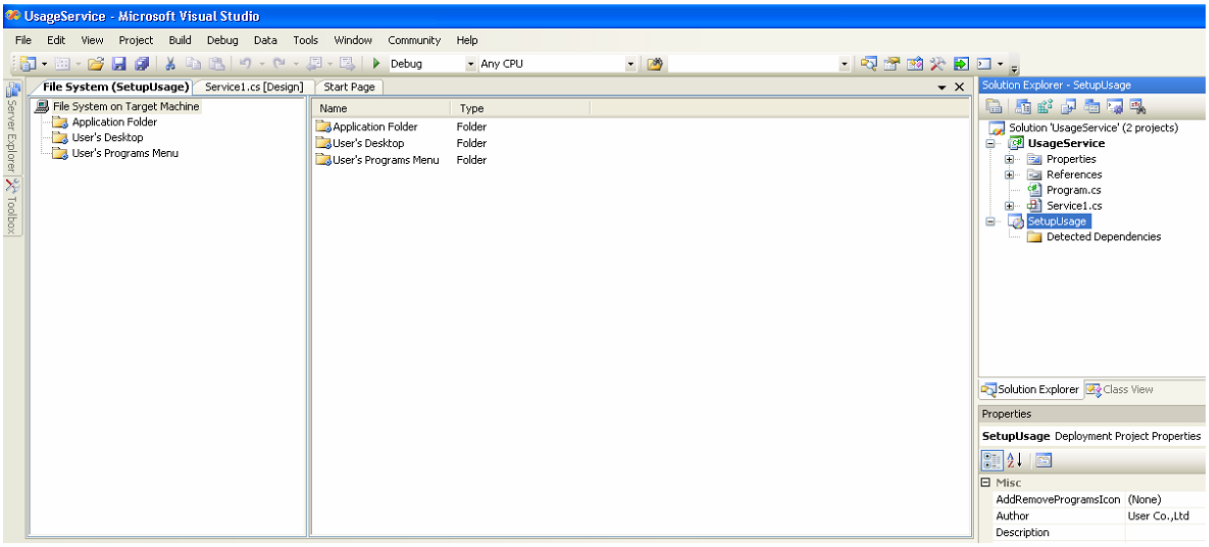
B1: Từ menu → File → Add →New Project.



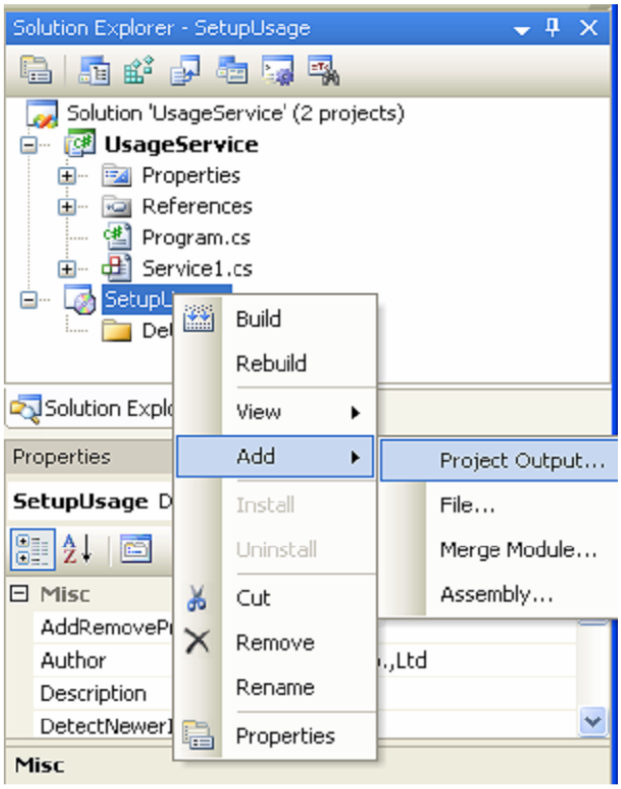
→ Đặt tên Projects(ví dụ là SetupUsage).



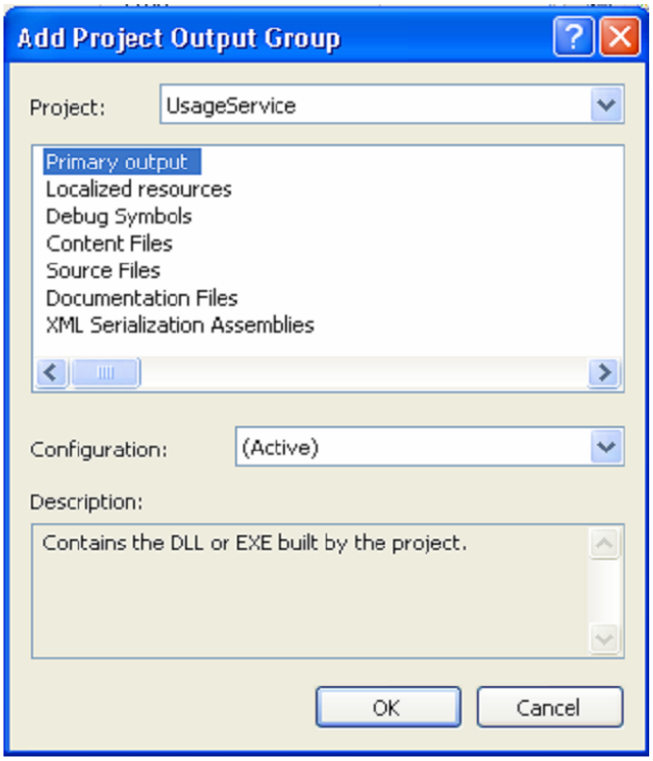
Sau đó ta bấm **OK**. Màn hình sẽ xuất hiện như sau:



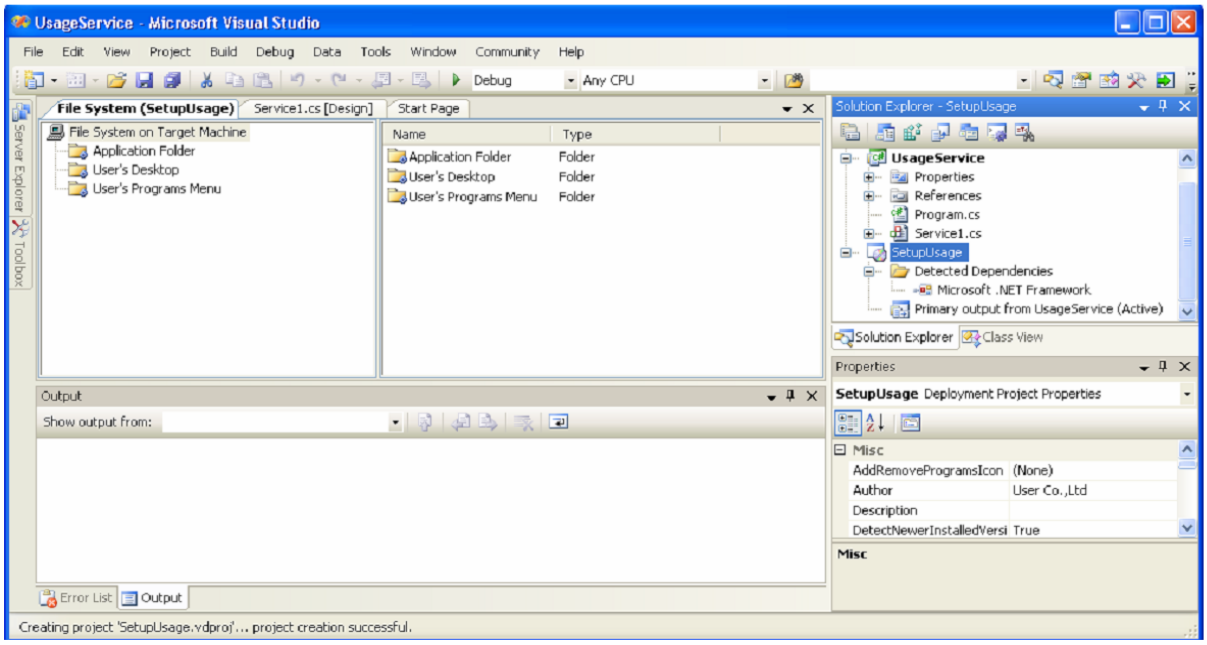
B2: Trong cửa sổ Solution Explorer, click phải chuột tại SetupUsage → Add → Project Output.



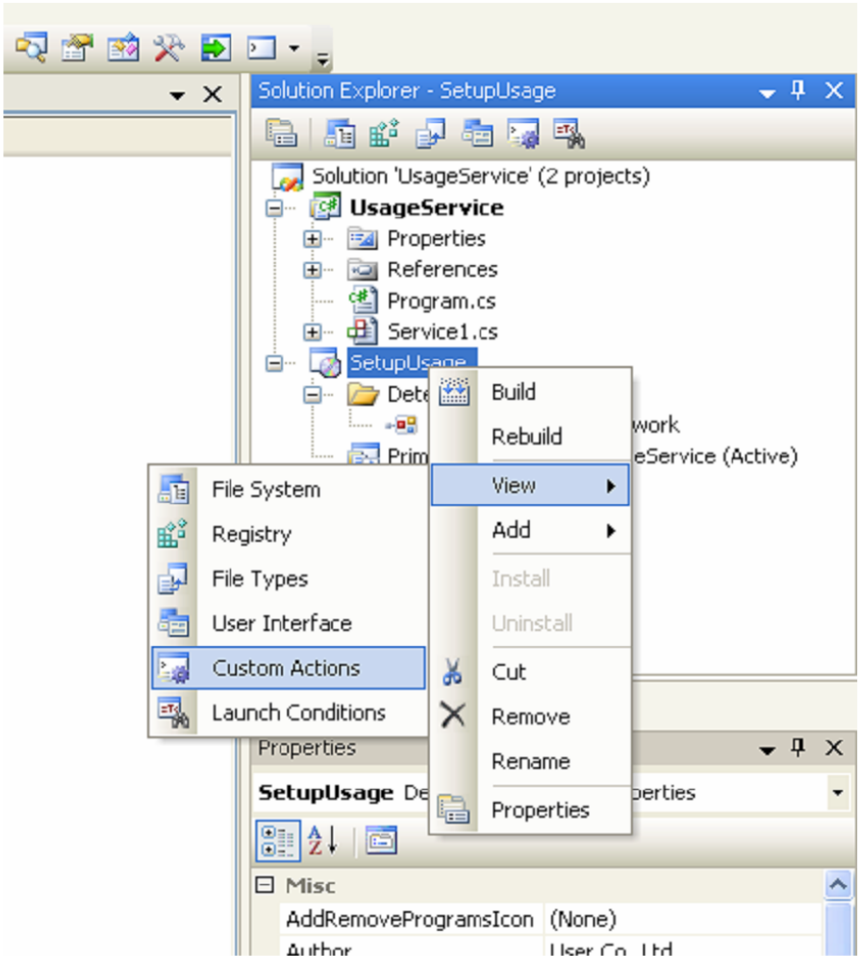
+ Từ cửa sổ Add Project Output Group → chọn Primary Output → click OK.



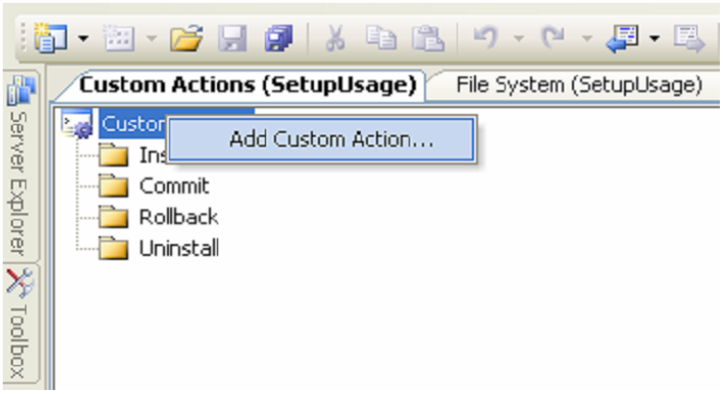
Màn hình xuất hiện như sau:



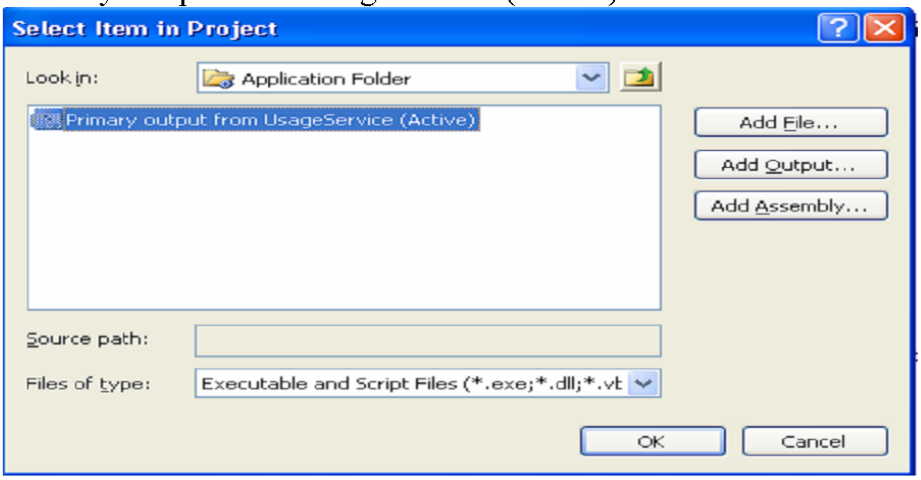
B3: Trong cửa sổ Solution Explorer, click phải chuột tại mục SetupUsage → View → Custom Actions.



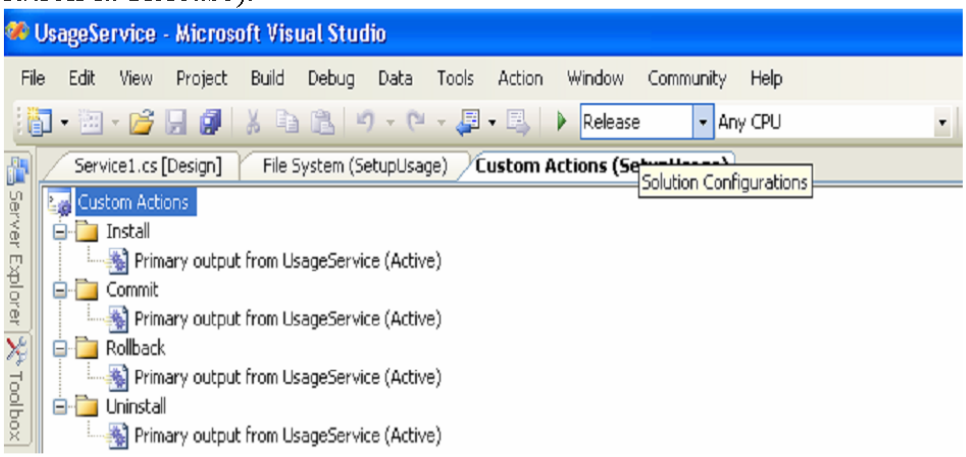
B4: Tại mục Custom Actions → click phải chuột → Add Custom Action.



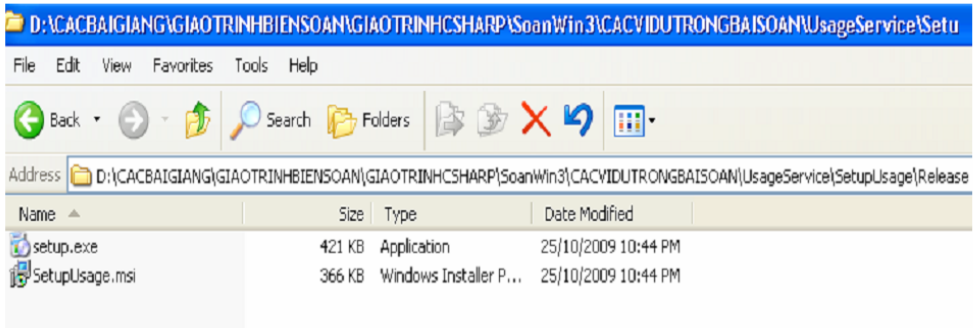
B5: Xuất hiện cửa sổ Select Item in Project → Double-click mục Application Folder, chọn mục Primary Output from UsageService (Active) → click OK.



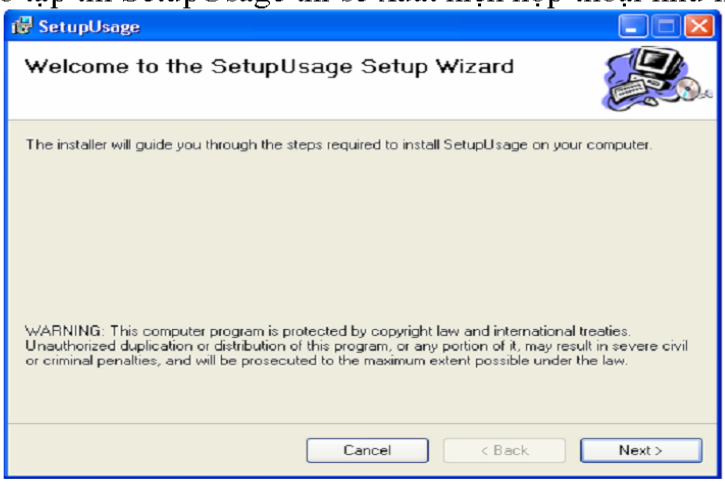
B6: Thực hiện việc build trình ứng dụng SetupUsage (chú ý nhớ chọn định cấu hình cho Solution là Release).



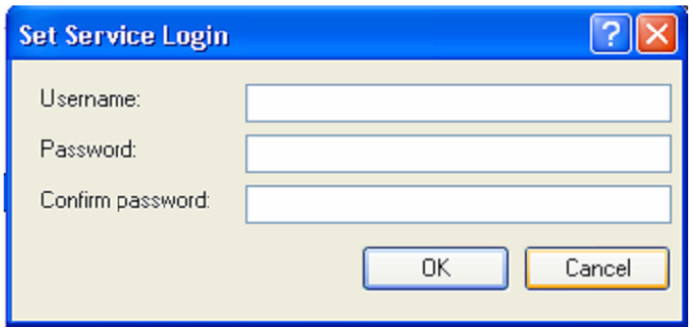
B7: Trong thư mục của project vừa tạo SetupUsage→ Ta vào thư mục con có tên Release và tìm tập tin có tên SetupUsage.msi. Double-click tập tin này để start việc cài đặt (installation). Thao tác này sẽ start trình Setup Wizard. Chấp nhận tất cả các giá trị ngầm định và hoàn tất việc cài đặt(installation).



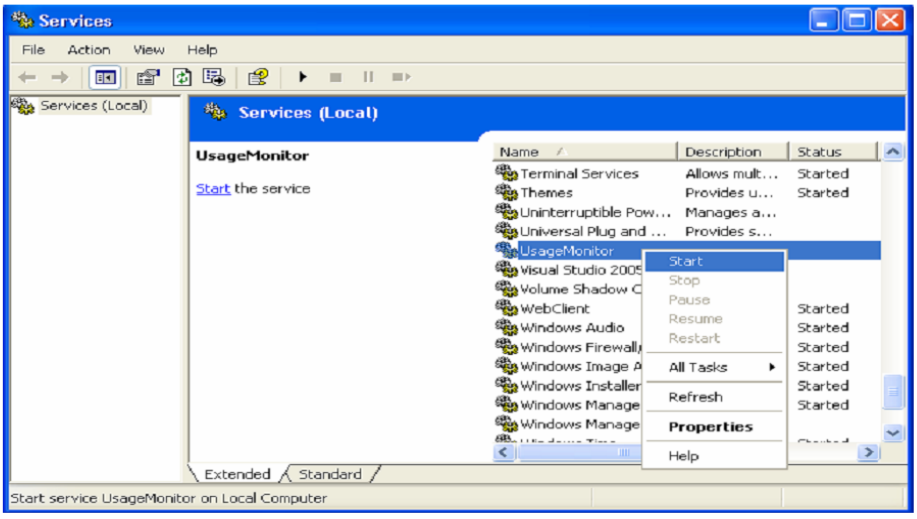
Khi Double-click vào tập tin SetupUsage thì sẽ xuất hiện hộp thoại như hình sau:



Chấp nhận tất cả các giá trị ngầm định và hoàn tất việc cài đặt(installation). Nhưng khi đến bước xuất hiện hộp thoại như dưới đây ta phải khai báo cho đúng nếu không sẽ bị lỗi.



B8: Để kiểm tra service vừa được cài đặt bạn cần sử dụng công cụ Service Control Manager với các bước thực hiện: click Start → Programs → Administrative Tools → Services. Chúng ta sẽ thấy UsageMonitor trong danh sách như hình bên dưới:



Bước 9:

Mở service manager để chạy ứng dụng: Control panel→Administrative Tools→Services.

Sau đó tìm service cần chạy click phải→Start.

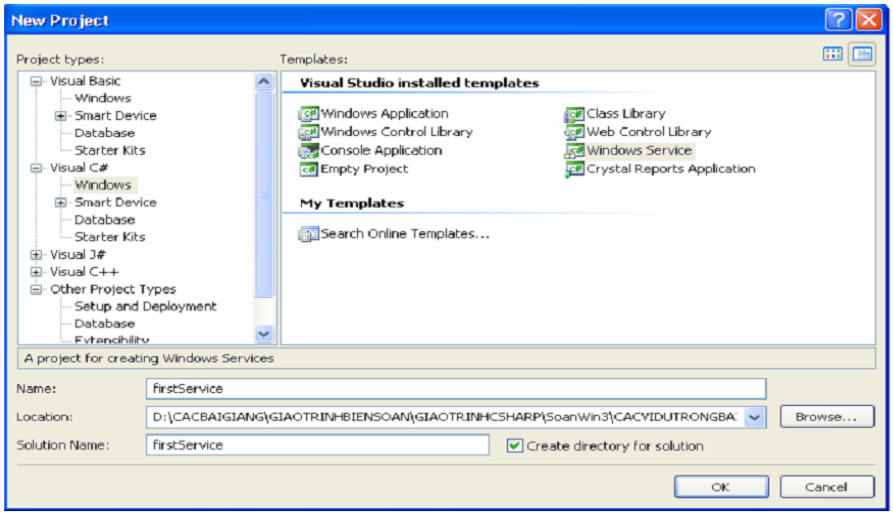
2.2 Ví dụ:

Ta sẽ thực hiện từng bước để tạo Windows Services thông qua các ví dụ đơn giản sau đây:

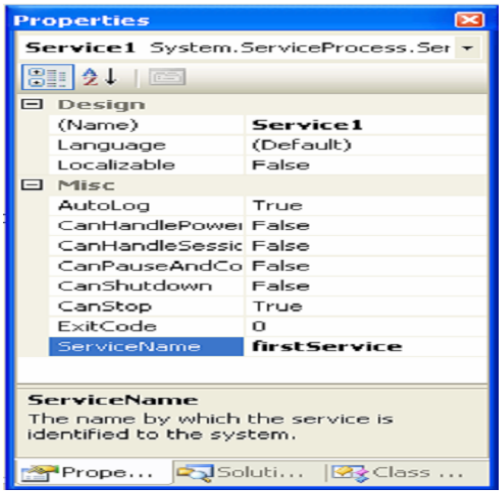
Ví dụ 1:

Sevice ta tạo trong ví dụ này có tên là **firstService**. Dịch vụ Service này sẽ không thực hiện bất cứ một xử lý nào mà dịch vụ sẽ ghi lại một số tác vụ truy xuất trong file sự kiện ở ổ đĩa gốc đó là file **DemoEvents.log**. Service sẽ ghi vào file log thông báo thời gian dịch vụ bắt đầu và chấm dứt.

Ta sẽ tạo từng bước như sau:



Trong màn hình **Design view** nhấn phải lên mục Project sẽ hiển thị một menu Popup. Chọn Properties từ menu. Sẽ hiển thị thuộc tính của dịch vụ. Ta tiến hành gán tên cho mục **ServiceName** là **firstService.**

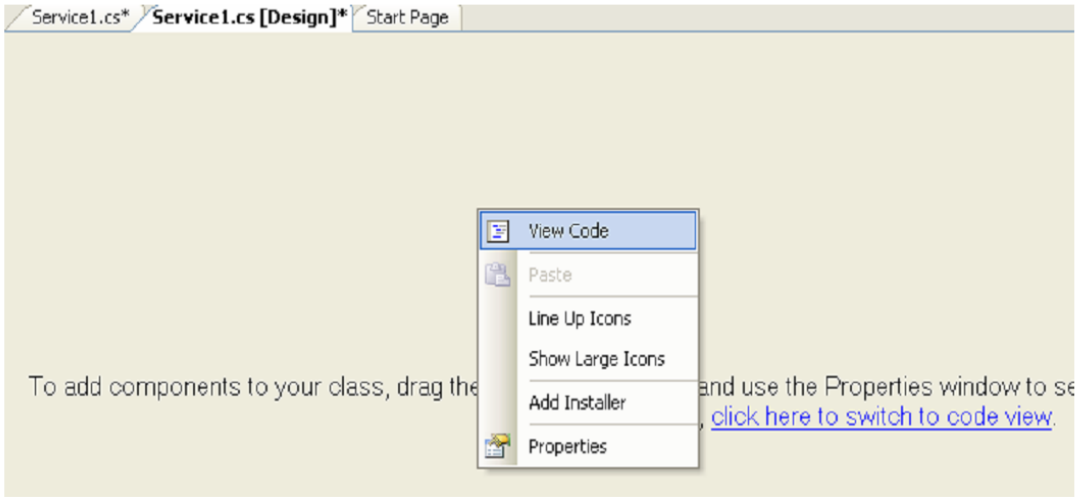


Vì mỗi khi **firstService** bắt đầu, dịch vụ sẽ nối vào file log một chuỗi thông báo. Để có thể sử dụng thao tác nhập xuất này ta cần khai báo thêm nameSpace System.IO như sau:

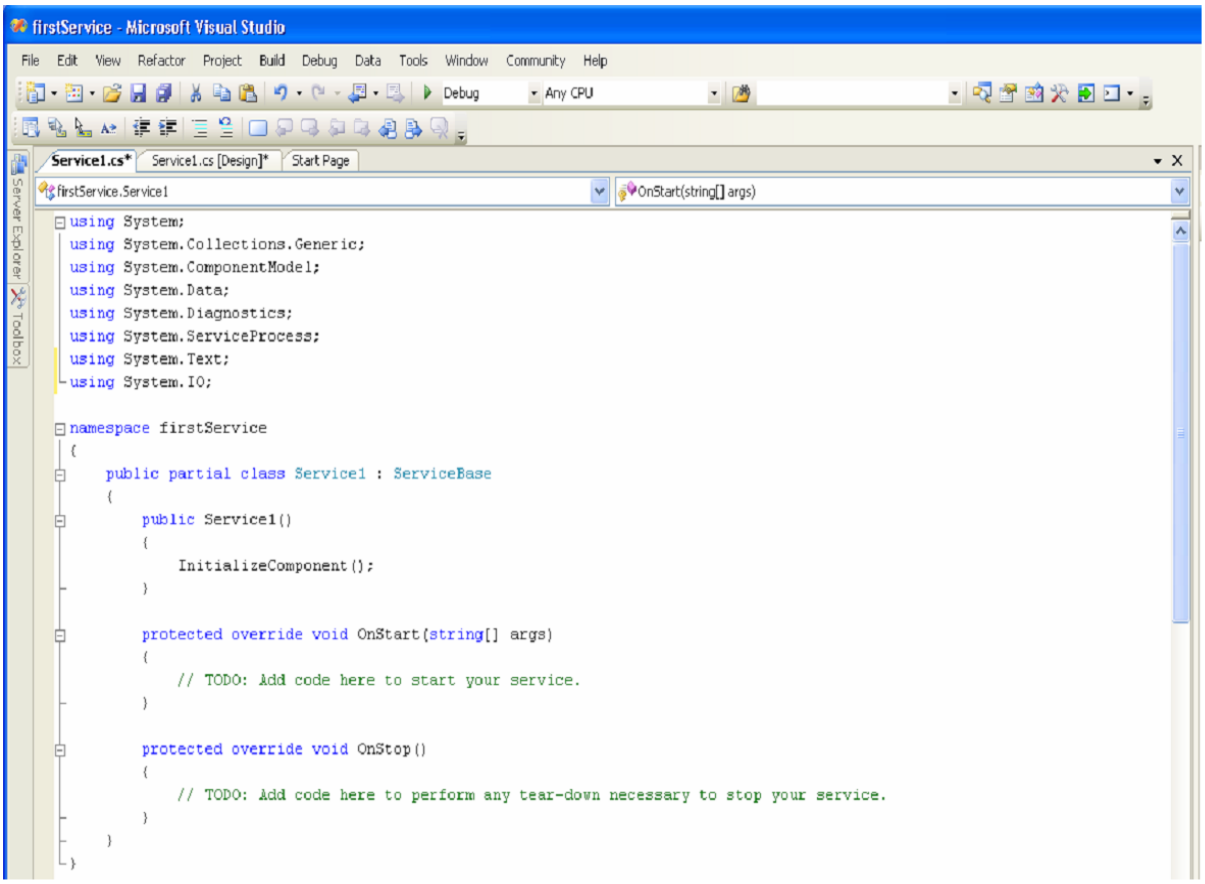
```
using System.IO;
```

Tiến hành viết code cho phương thức OnStart và OnStop như sau:

Đầu tiên bấm phải vào vùng trống Design sau đó chọn viewCode như sau:



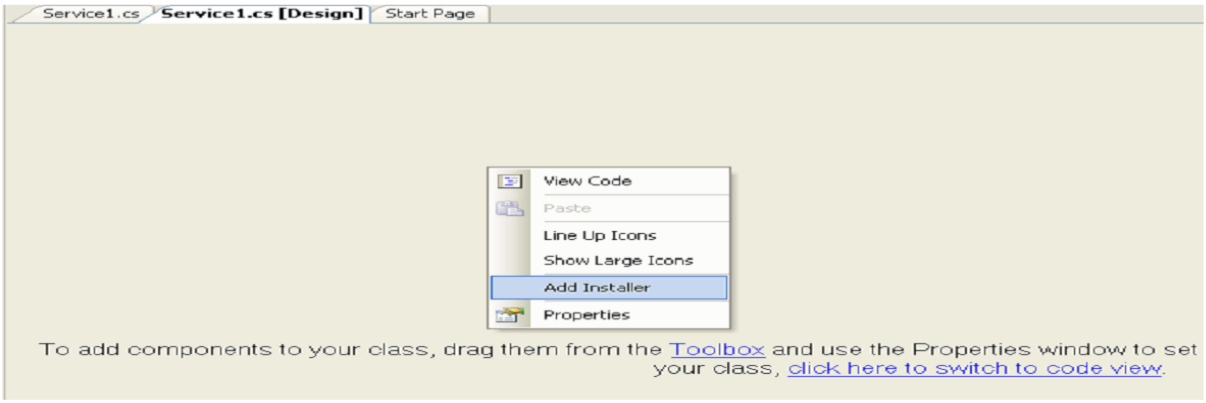
Màn hình viết code thể hiện như sau:



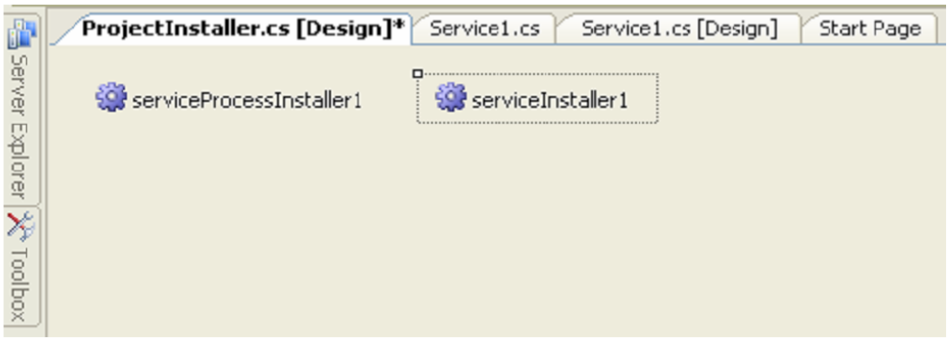
Tiến hành viết cho phương thức OnStart như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.ServiceProcess;
using System.Text;
using System.IO;

namespace firstService
{
   public partial class Service1 : ServiceBase
   {
      public Service1()
      {
         InitializeComponent();
      }
      protected override void OnStart(string[] args)
      {
         TextWriter file = new StreamWriter("C:\\DemoEvents.log", true);
         file.WriteLine("Service Start:"+DateTime.Now.ToString());
         file.Flush();
        file.Close();
      }
      protected override void OnStop()
      {
     // TODO: Add code here to perform any tear-down necessary to stop your service.
      }
   }
}
```

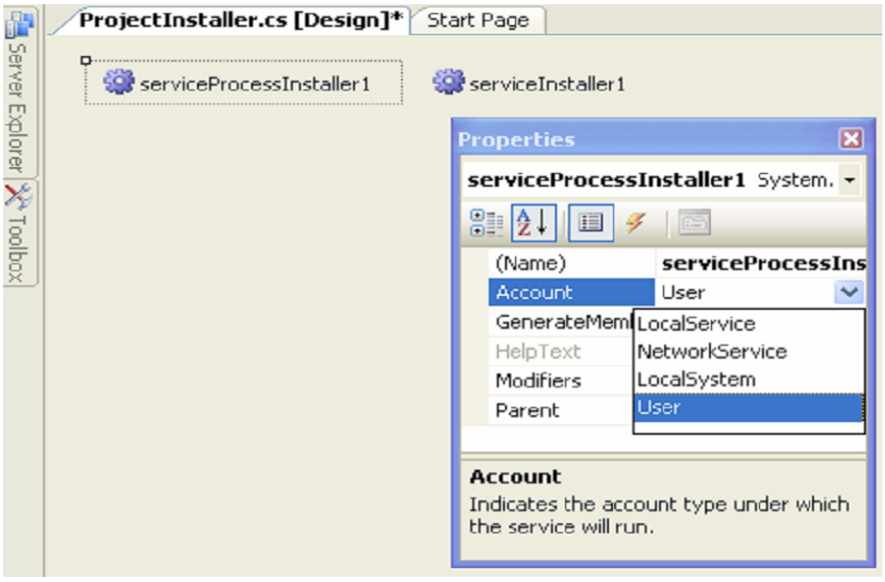
Trước khi cài đặt dịch vụ vào hệ thống ta chọn Design view sau đó Click chuột phải lên vùng thiết kế chọn **Add Installer**.



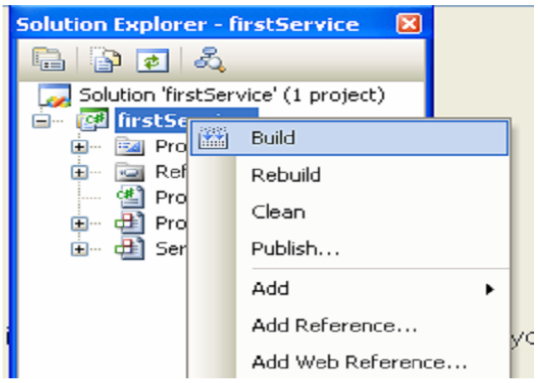
Xuất hiện màn hình chứa các biểu tượng sau đây:



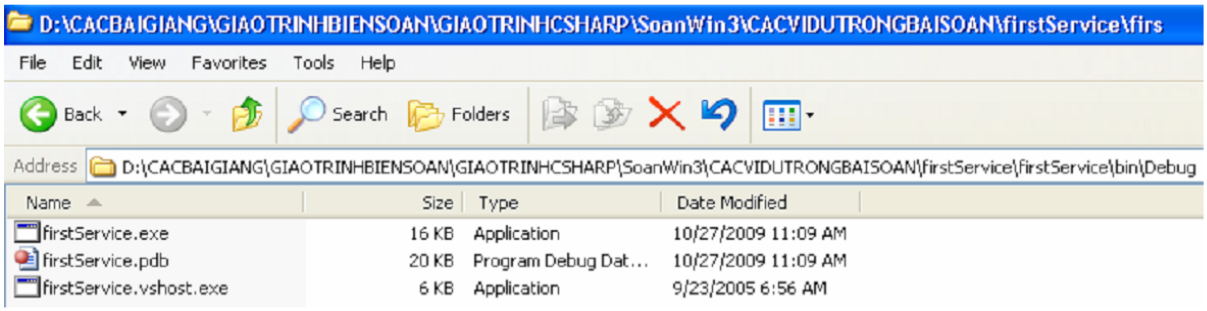
Ta tiến hành chỉ rõ thông tin cài đặt của dịch vụ, nhấn phím phải lên mục **ServiceProcessInstaller** trong cửa sổ **Properties** ta chỉ rõ kiểu tài khoản ở đây ta chon thử loại **User.** Sau đó ta biên dịch dịch vụ. Ở giai đoạn này, dịch vụ của chúng ta cần phải chứa các phát biểu xử lý, ở đây chúng ta chỉ ghi ra **log file** trong sự kiện **OnStart()** của dịch vụ. **Chọn Build | Build firstService** để biên dịch.



Sau đó bấm phải vào **firstService** như sau:

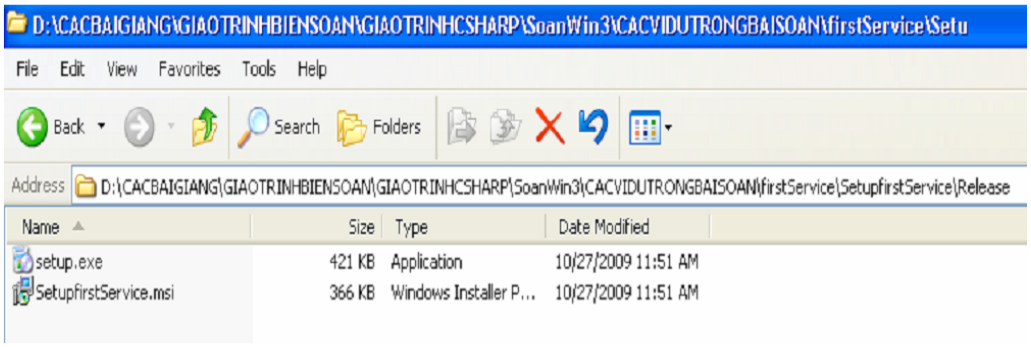


Vào thư mục lưu Project vào vào thư mục **bin\Debug** sẽ thấy file **.exe** như sau:

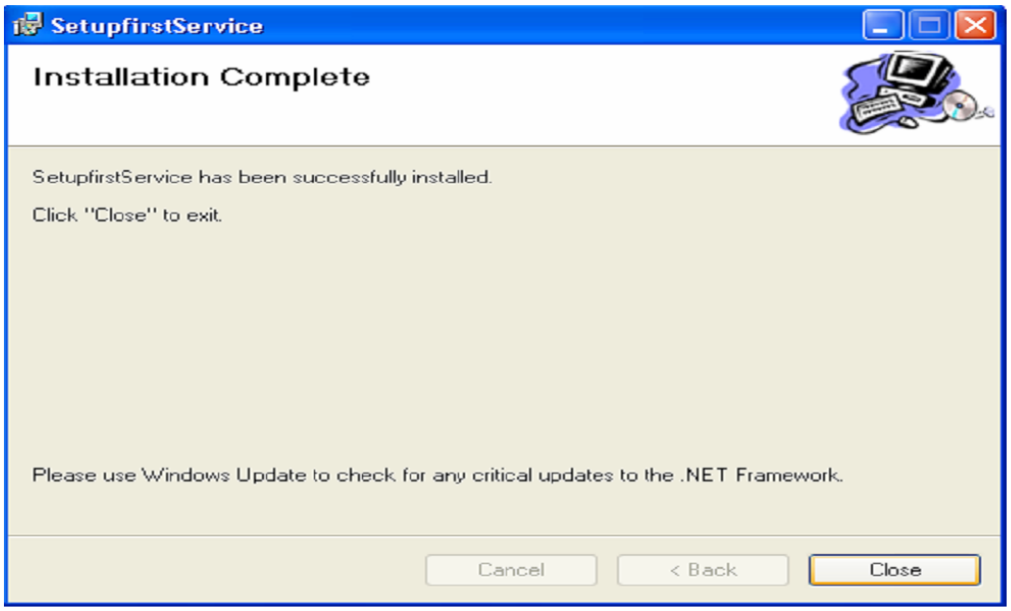


Sau đây chúng ta sẽ tiến hành tạo một trình ứng dụng loại **setup** để cài đặt Service các bước tạo ta **xem kĩ lại phần trên của bài giảng**.

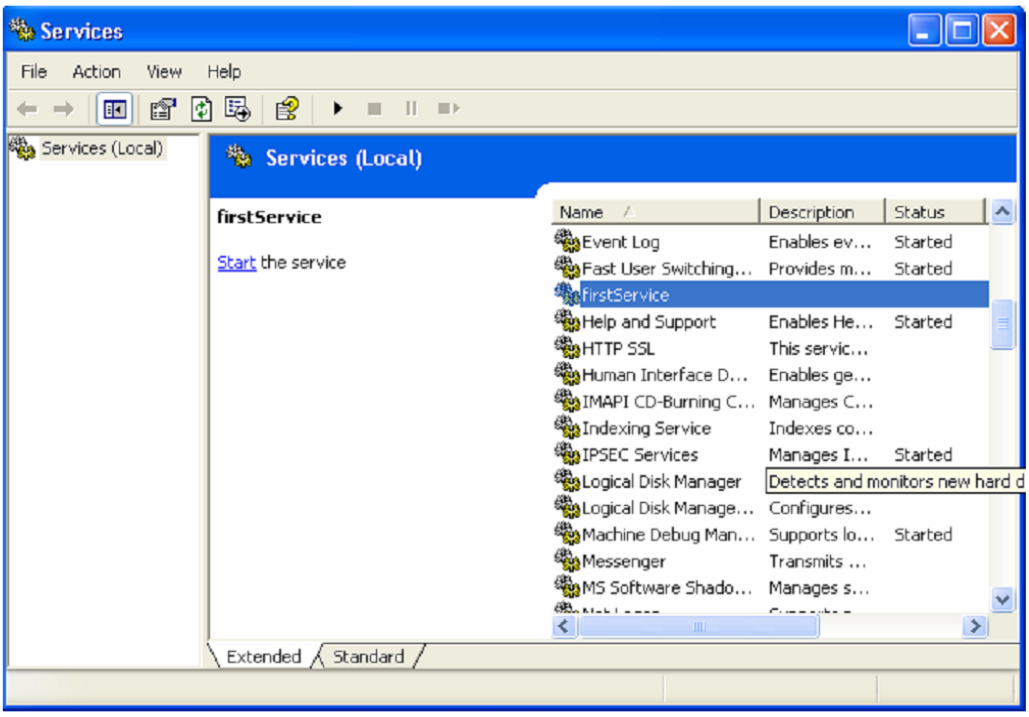
Sau khi thực hiên xong các bước như trên ta và thư mục của Project vừa tạo như hướng dẫn ở trên và thấy như sau:



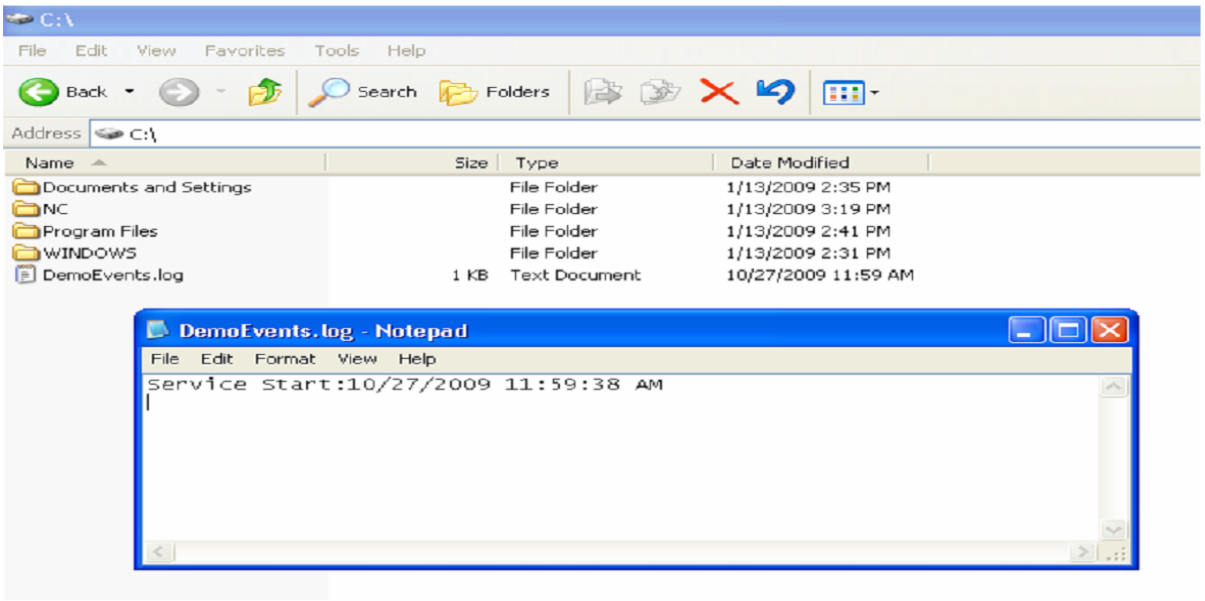
Tiến hành cài đặt và thực hiện đến bước sau là ta đã thành công việc tạo và cài đặt service:



Sau đó ta vào **Administrative Tools** để kiểm tra xem có Service chưa. Nếu có như hình bên dưới thì Service đã được cài đặt vào máy.



Tiến hành Start service và vào ổ C kiểm tra file log như hình phía dưới:



Ví dụ 2:

Tạo một **Service** theo dõi quá trình sử dụng **CPU**, kết quả sẽ lưu vào một file txt có tên là **Output.txt**, tên của Service này đặt là **ServiceUsageMonitor.**

+ Tạo một Project mới có tên là **ExampleService** như hình bên dưới.

Bài 4: Tạo và sử dụng các thành phần dịch vụ COM+

Mục tiêu của bài:

Nhằm trang bị cho người học:

- Kiến thức và kỹ năng lập trình sử dụng dịch vụ **COM+**.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

1. Giới thiệu sơ lượt về COM:

Các ngôn ngữ đều muốn chia sẻ các thành phần(component) dùng chung. Trong Windows thực hiện điều này bởi **COM(Component Object Model).** Mục đích duy nhất của COM là cho phép bất cứ ngôn ngữ nào hiểu được chuẩn nhị phân của COM sẽ sử dụng được đối tượng Component có trong hệ thống.

Với sự xuất hiện của .NET những vấn đề này không xuất hiện bởi vì tất cả ngôn ngữ .NET có thể thoải mái tương tác với các ngôn ngữ khác một cách dễ dàng. Sự phối hợp của CLS và CLR tạo ra sự tương tác như vậy.

COM (Component Object Model) là một cách thức để viết các thành phần phần mềm(software component) cung cấp các chức năng phục vụ cho các ứng dụng, các thành phần khác.

Viết một thành phần **COM** cũng như viết một hàm **API** động, hướng đối tượng. Các thành phần **COM** có thể được nối kết với các ứng dụng cũng như các thành phần **COM** khác.

a. COM là gì ?

COM chỉ là một sự đặt tả. Nó chỉ ra làm thế nào để xây dựng các thành phần có thể thay thế một cách **"động"**. COM cung cấp một **"chuẩn"** để các ứng dụng và các thành phần phải tuân theo để chúng có thể hoạt động được với nhau.

b. ***Thành phần COM là gì ?***

Thành phần COM là những đoạn mã hiện thực các hàm, các chức năng được chứa trong file .DLL hoặc .EXE. Thành phần COM cần phải thỏa mãn được những **"chuẩn"** của COM. Các thành phần COM có thể được liên kết động.

Các thành phần COM có thể được đóng bao vì nó thỏa mãn các ràng buộc:

- Thành phần COM hoàn toàn độc lập với ngôn ngữ hiện thực nó. Thành phần COM có thể được hiện thực bằng nhiều ngôn ngữ như Visual C++, Visual Basic, Java,C#…
- Thành phần COM được sử dụng dưới dạng mã nhị phân.
- Thành phần COM có thể được nâng cấp mà không làm hỏng chương trình đang chạy.
- Thành phần COM trong suốt về vị trí.

c. ***Giao diện (Interface)***

Giao diện là tất cả đối với COM. Các ứng dụng chỉ có thể nhìn thấy và truy xuất đến các thành phần COM thông qua giao diện của nó.

Giao diện làm tăng mức độ độc lập giữa ứng dụng và các thành phần COM. nhờ có giao diện mà một thành phần COM có thể được thay thế động mà không ảnh hưởng đến ứng dụng đang chạy.

Giao diện không bao giờ thay đổi. Nếu muốn nâng cấp một giao diện thì giao diện mới phải tồn tại song song với giao diện cũ.

d. ***COM trong Windows:***

Trong Windows, các thành phần COM cũng như giao diện của COM đều được gán một ID.

ID của thành phần COM được gọi là **CLSID**(class identifier); của giao diện COM là **IID**(interface identifier).

Các ID này được gọi chung là **GUID**(globally unique identifier) là một cấu trúc 16 byte và được xác định duy nhất trên toàn thế giới.

Tất cả các ID này đều được chứa trong registry của windows cùng với thông tin về các thành phần hoặc giao diện mà nó đại diện.

Khi ứng dụng muốn sử dụng COM, nó sẽ gọi một hàm API của windows là **CoCreateInstance** với thông số truyền vào là CLSID và IID tương ứng với thành phần và giao diện mà thành phần đó hỗ trợ để có được giao diện mong muốn.

e. COM+ là gì?

COM+ là sự phát triển của **Microsoft Component Object Model (COM)** và **Microsoft Transaction Server (MTS)**. **COM+** được xây dựng và mở rộng các ứng dụng viết bằng **COM, MTS** và những phần khác dựa trên công nghệ **COM.**

COM+ = COM + MTS(Microsoft Transaction Server)

COM+ xử lý nhiều nhiệm vụ quản lý tài nguyên mà trước đây ta phải xây dựng một chương trình để xử lý. Chẳng hạn như phân bổ thread và bảo mật.

COM+ cũng làm cho ứng dụng của chúng ta mở rộng thêm bằng cách cung cấp thêm thread pooling, Object pooling, và kích hoạt đối tượng just – in – time.

COM + cũng giúp bảo vệ sự toàn vẹn dữ liệu của chúng ta bằng cách cung cấp hỗ trợ giao dịch. Ngay cả khi kéo dài sự giao dịch dữ liệu qua mạng.

Trường hợp nào nên áp dụng COM+?

COM + có thể được dùng để phát triển doanh nghiệp rộng khắp, nhiệm vụ quan trọng, các ứng dụng được phân phối cho Windows.

Nếu chúng ta là một quản trị hệ thống, ta có thể cài đặt, triển khai, và cấu hình **COM +** và ứng dụng các thành phần của chúng.

Nếu ta là một lập trình viên, ta sẽ viết các thành phần và hợp nhất chúng thành ứng dụng. Nếu chúng ta là đại lý cung cấp các tools, ta sẽ phát triển và sửa đổi các tools để làm việc trong môi trường **COM+.**

Đối tượng phát triển:

COM + được thiết kế chủ yếu cho Microsoft Visual C++ và các nhà phát triển Microsoft Visual Basic hay Microsoft Visual C#.

Yêu cầu Run-Time:

COM + phiên bản **1.0** là bao gồm trong Windows 2000.**COM +** phiên bản **1.5** là bao gồm trong Windows bắt đầu với Windows XP và Windows Server 2003.

Có gì mới trong COM+ 1.5

Phiên bản **COM+ 1.5** bổ sung thêm các tính năng mới được thiết kế để làm tăng khả năng mở rộng tổng thể, sẵn có, và quản lý của **COM+** ứng dụng cho cả các nhà phát triển và cho quản trị viên hệ thống.

COM+ 1.5 bắt đầu với Windows XP và Windows Server 2003. Tính năng mới của **COM+ 1.5** không có trong Windows 2000.

Aplication- Level kiểm tra mức độ truy xuất mặc định:

Là một phần của tăng cường bảo mật hệ thống, kiểm tra truy cập được kích hoạt mặc định khi tạo một ứng dụng **COM +**.

Trong các phiên bản trước đó, kiểm tra truy cập được tắt theo mặc định ở cấp độ ứng dụng và kích hoạt mặc định ở cấp độ thành phần.

Bắt đầu với Windows Server 2003, kiểm tra truy cập được kích hoạt mặc định ở cấp độ ứng dụng và tắt theo mặc định ở cấp độ thành phần.

Application Pooling:

Với thuộc tính mới **ConcurrentApps** của đối tượng **COMAdminCatalogObject** trong các ứng dụng. **COM+ Application Pooling** thêm khả năng cho các tiến trình **Single-thread** và tích hợp với các dịch vụ tái chế ứng dụng **COM+**.

Application Recycling

Application Recycling làm tăng đáng kể sự ổn định chung của các ứng dụng. Bởi vì hiệu suất của hầu hết các ứng dụng có thể bị suy giảm theo thời gian do các yếu tố như rò rỉ bộ nhớ, dựa vào các mã **third-party**, và cách sử dụng tài nguyên nonscalable, **COM** + application recycling cung cấp một giải pháp đơn giản để gracefully tắt một tiến trình liên kết với một ứng dụng và khởi động lại nó

COM+ Partitions

Một tính năng cho phép nhiều phiên bản của COM + ứng dụng được cài đặt và cấu hình trên cùng một máy. Tính năng này cho phép chúng ta tiết kiệm chi phí và thời gian cho việc sử dụng nhiều máy chủ để quản lý các phiên bản khác nhau của một ứng dụng.

Trên một máy, mỗi phân vùng, trong hiệu ứng, giống như một máy chủ ảo. Sau khi cài đặt ứng dụng vào từng phân vùng, ta tạo lập bản đồ phân vùng cho người sử dụng đến các máy chủ hợp lý.

COM+ Services Without Components

Với **COM+ 1.5**, ta có thể sử dụng các dịch vụ cung cấp bởi **COM** + mà không cần phải xây dựng một thành phần có chứa các phương pháp gọi các dịch vụ.

Điều này giúp ích cho các nhà phát triển những người không thường sử dụng các thành phần, nhưng muốn sử dụng COM+ services như giao dịch hoặc COM+ Tracker. Bằng cách sử dụng COM+ services mà không có các thành phần, các nhà phát triển có thể tránh những nguyên tắc cần thiết của việc tạo ra một thành phần được sử dụng để truy cập chỉ COM+ services mà họ cần.

COM+ SOAP Service

Với **COM+ 1.5**, bây giờ ta có thể làm một **COM+** ứng dụng như một dịch vụ Web XML.

Điều này có nghĩa là ta có thể dễ dàng tạo mới dịch vụ Web XML trên hiện có **COM+** ứng dụng và dễ dàng kết hợp các dịch vụ Web XML vào **COM+** ứng dụng mới.

Configurable Isolation Levels

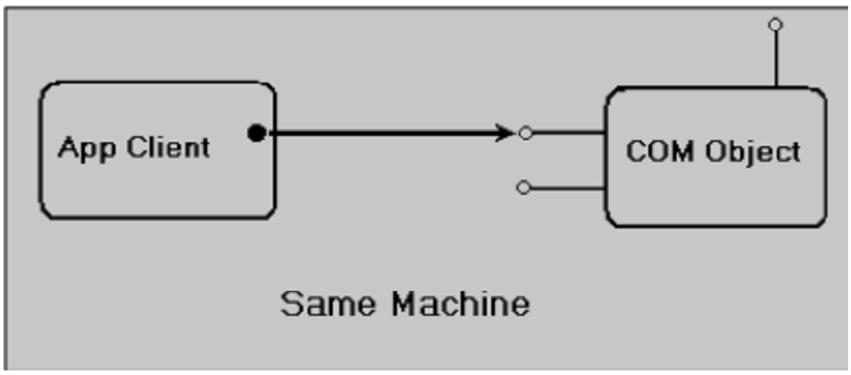
Nhà phát triển **COM+** có thể sử dụng mới **TxIsolationLevel** hoặc thành phần services để cấu hình một ứng dụng, giúp tăng đồng thời hiệu xuất và khả năng mở rộng.

Tạo một thành phần Private

Trong kịch bản, nơi ta có một số thành phần helper trong một ứng dụng(có nghĩa là chỉ được gọi từ các thành phần khác trong ứng dụng đó), điều này của **COM+** cho phép bạn sử dụng một thành phần mới, IsPrivateComponent, để đánh dấu những thành phần **Private.**

2. Các kiến trúc của COM/DCOM

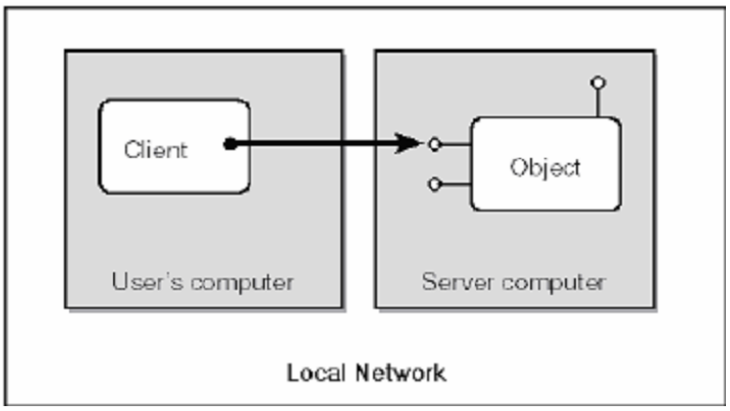
a. Ứng dụng Client và ứng dụng COM chạy trên cùng một máy.



Mục đích xây dựng thành phần đối với mô hình này là :

- Ứng dụng COM có thể được tái sử dụng cho nhiều ứng dụng Client khác nhau.
- Làm cho chương trình dễ viết, dễ kiểm tra lỗi, nâng cấp mà không ảnh hưởng đến những ứng dụng Client đang chạy nó.

b. Ứng dụng Client chạy một máy, Ứng dụng COM chạy một máy. Nhưng hai máy này cùng một mạng local.



Lúc này thành phần COM được gọi là DCOM. Có 3 cách để giao tiếp giữa ứng dụng Client và Ứng dụng COM:

- ActiveX (.exe)
- Dùng MTS (Microsoft Transaction Server) để triển khai ActiveX DLL từ xa
- RDS (Remote Data Access)

3. Tạo và đăng kí Serviced Components

Khai báo name space System.EnterpriseServices

Lớp System.EnterpriseServices.ServicedComponent

```
using System.EnterpriseServices;
public class Account : ServicedComponent
{
    static void Main()
        {
        }
}
```

Ví dụ:

```
using System;
using System.EnterpriseServices;
namespace SimpCom
{
        public interface Ca
        {
                int add(int a,int b);
                int sub(int a,int b);
        }
                public class dancom:ServicedComponent,Ca
                {
                        public dancom()
                        {
                        }
                        public int add(int a,int b)
                        {
                                return a+b;
                        }
                        public int sub(int a,int b)
                        {
                                return a-b;
                        }

                }
        }
```

Trong file AssemplyInfo ta khai báo như sau:

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.EnterpriseServices;
//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]

//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly
signing.
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//   (*) If no key is specified, the assembly is not signed.
//   (*) KeyName refers to a key that has been installed in the Crypto Service
//       Provider (CSP) on your machine. KeyFile refers to a file which contains
//       a key.
//   (*) If the KeyFile and the KeyName values are both specified, the
//       following processing occurs:
//       (1) If the KeyName can be found in the CSP, that key is used.
//       (2) If the KeyName does not exist and the KeyFile does exist, the key
//           in the KeyFile is installed into the CSP and used.
```

```
//   (*) In order to create a KeyFile, you can use the sn.exe (Strong Name) utility.
//       When specifying the KeyFile, the location of the KeyFile should be
//       relative to the project output directory which is
//       %Project Directory%\obj\<configuration>. For example, if your KeyFile is
//       located in the project directory, you would specify the AssemblyKeyFile
//       attribute as [assembly: AssemblyKeyFile("..\\..\\mykey.snk")]
//   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
//       documentation for more information on this.
//
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("../../bin/debug/stronname.snk")]
[assembly: AssemblyKeyName("")]
```

Tạo project có tên usecom như sau:

```
using System;
using SimpCom;
using System.EnterpriseServices;
namespace useCom
{
        /// <summary>
        /// Summary description for Class1.
        /// </summary>
        class Class1
        {
                /// <summary>
                /// The main entry point for the application.
                /// </summary>
                [STAThread]
                static void Main(string[] args)
                {
                        dancom d=new dancom();
                        dancom d1=new dancom();
                        Console.WriteLine("10+30={0}",d.add(10,30));
                        Console.WriteLine("30-10={0}",d1.sub(30,10));
                        Console.ReadLine();
                }
        }
}
```

Trong file AssemblyInfo khai báo như sau:

```
using System.Reflection;
using System.Runtime.CompilerServices;
//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly: AssemblyTitle("")]
```

```
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]

//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly
signing.
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//   (*) If no key is specified, the assembly is not signed.
//   (*) KeyName refers to a key that has been installed in the Crypto Service
//       Provider (CSP) on your machine. KeyFile refers to a file which contains
//       a key.
//   (*) If the KeyFile and the KeyName values are both specified, the
//       following processing occurs:
//       (1) If the KeyName can be found in the CSP, that key is used.
//       (2) If the KeyName does not exist and the KeyFile does exist, the key
//           in the KeyFile is installed into the CSP and used.
//   (*) In order to create a KeyFile, you can use the sn.exe (Strong Name) utility.
//       When specifying the KeyFile, the location of the KeyFile should be
//       relative to the project output directory which is
//       %Project Directory%\obj\<configuration>. For example, if your KeyFile is
//       located in the project directory, you would specify the AssemblyKeyFile
//       attribute as [assembly: AssemblyKeyFile("..\\..\\mykey.snk")]
//   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
//       documentation for more information on this.
//
```

```
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("")]
[assembly: AssemblyKeyName("")]
```

4. Dùng MTS để triển khai ActiveX DLL từ xa

Đối với Windows NT thì MTS được tách riêng ra thành một ứng dụng riêng biệt. Muốn sử dụng MTS thì phải cài đặt Option Pack 4 cho Windows NT. Còn đối với Windows 2000 thì MTS được tích hợp vào IIS 5.0, đưa COM và MTS tích hợp lại trong COM+ Applications.

Giới thiệu MTS (Microsoft Transaction Server)

MTS là một dịch vụ trên Windows giống như các dịch vụ khác như IIS, File hay Print …, có nghĩa là ta có thể chạy (start) hoặc ngưng chạy (stop) chúng khi cần và MTS chạy background để phục vụ cho các ứng dụng cần đến nó

Trong Windows NT 4.0, MTS được coi như là một phần phụ (add-on), được cài thông qua NT4 Option Pack. Còn trong Windows 2000 nó trở thành một phần mặc định của chính hệ điều hành.

Ngoài chức năng chính là quản lý giao dịch, MTS còn quản lý nhiều thứ khác "MTS quản lý cách mà ứng dụng sử dụng các thành phần" hay nói cách khác là MTS quản lý các thành phần giúp cho các ứng dụng sử dụng các thành phần đó được hiệu quả hơn.

Các bước thông thường để sử dụng một thành phần:

B1: Tạo một instance của thành phần.
B2: Khởi động giá trị của các thuộc tính của thành phần.
B3: Sử dụng thành phần.
B4: Giải phóng thành phần ra khỏi bộ nhớ.

Thông thường, việc khởi tạo và giải phóng thành phần nằm ở đầu và ở cuối một thủ tục hay chương trình. Nếu sử dụng tốt hơn thì khởi tạo thành phần ngay khi cần đến và giải phóng sau khi chắc chắn không còn cần dùng đến thành phần đó nữa.

Tuy nhiên trong khoảng thời gian sử dụng thành phần đó thường có rất nhiều thao tác không sử dụng thành phần hoặc không có thao tác nào sử dụng đến thành phần, nhưng vẫn phải giữ thành phần đó trong bộ nhớ chờ được sử dụng sau ở các thao tác khác mà không thể giải phóng thành phần vì khi giải phóng thì các trạng thái bên trong của thành phần cũng bị xóa đi.

Điều đó tạo nên một sự lãng phí rất lớn tài nguyên của hệ thống. MTS giải quyết vấn đề trên bằng cách: khi một ứng dụng cần một thành phần thì MTS sẽ cung cấp 1 instance của thành phần đó cho ứng dụng.

Nếu một ứng dụng thứ 2 muốn sử dụng thành phần đó, trong khi ứng dụng 1 không thực sự sử dụng nó (tuy nhiên ứng dụng thứ 1 vẫn chưa giải phóng thành phần) thì MTS sẽ đưa instance mà ứng dụng thứ 1 đang giữ cho ứng dụng thứ 2 sử dụng.

Nếu ứng dụng thứ 1 đột nhiên muốn sử dụng tiếp thành phần thì MTS sẽ tìm trong các instance của thành phần đó có instance nào không thực sự được sử dụng bởi ứng dụng khác không. Nếu có thì MTS sẽ đưa instance đó cho ứng dụng 1 sử dụng.

Nếu tất cả các instance của thành phần đó đều đang được sử dụng thì MTS sẽ tạo một instance mới và giao nó cho ứng dụng thứ 1 sử dụng.

Quản lý giao dịch trong MTS: MTS cung cấp môi trường giao dịch cho các thành phần. Các thành phần trong cùng một môi trường sẽ thực hiện một giao dịch, có nghĩa

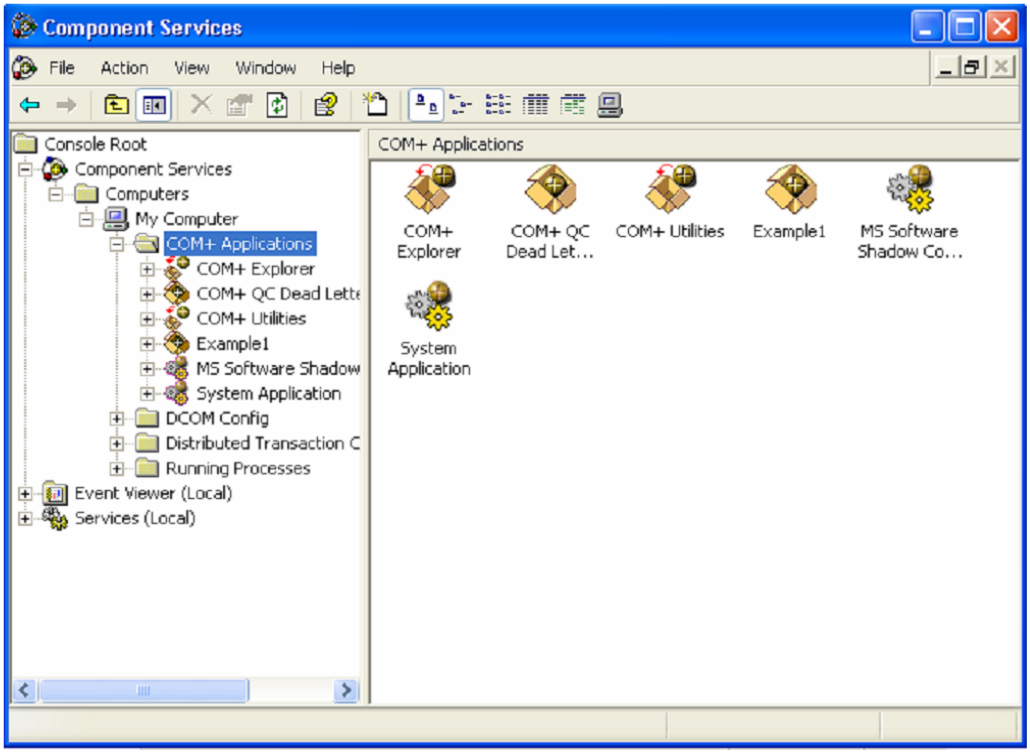
là nếu toàn bộ các thành phần trong môi trường cùng thực thi thành công nhiệm vụ của mình thì giao dịch coi như là thành công.

Nếu một trong các thành phần vì lý do nào đó không thể thực thi nhiệm vụ của mình thì toàn bộ các công việc của các thành phần khác sẽ bị hủy bỏ và hệ thống sẽ trở lại trạng thái ban đầu, trạng thái trước
khi giao dịch được thực hiện.

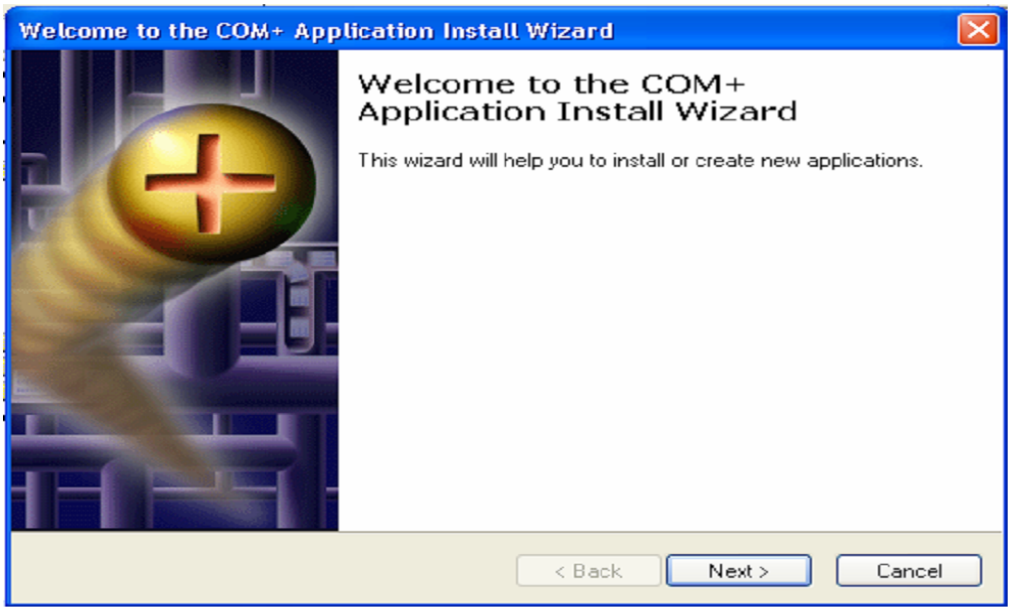
 Các bước để triển khai một thành phần từ xa trong Windows 2000 dùng MTS :

 Các bước thực hiện trên máy chạy thành phần COM DLL muốn triển khai từ xa.

1. Tạo một thành phần COM dạng ActiveX DLL
2. Nhấn Start \ Settings \ Control Panel
3. Trong cửa sổ Control Panel mở folder Administrative Tools
4. Mở Component Services
5. Trong console tree của Component Services, Nhấn đúp vào Computers
6. Nhấn đúp vào My Computers, nhấn đúp vào COM+ Applications
7. Nhấn phải chuột vào COM+ Applications, trỏ tới New rồi nhấn vào Application.

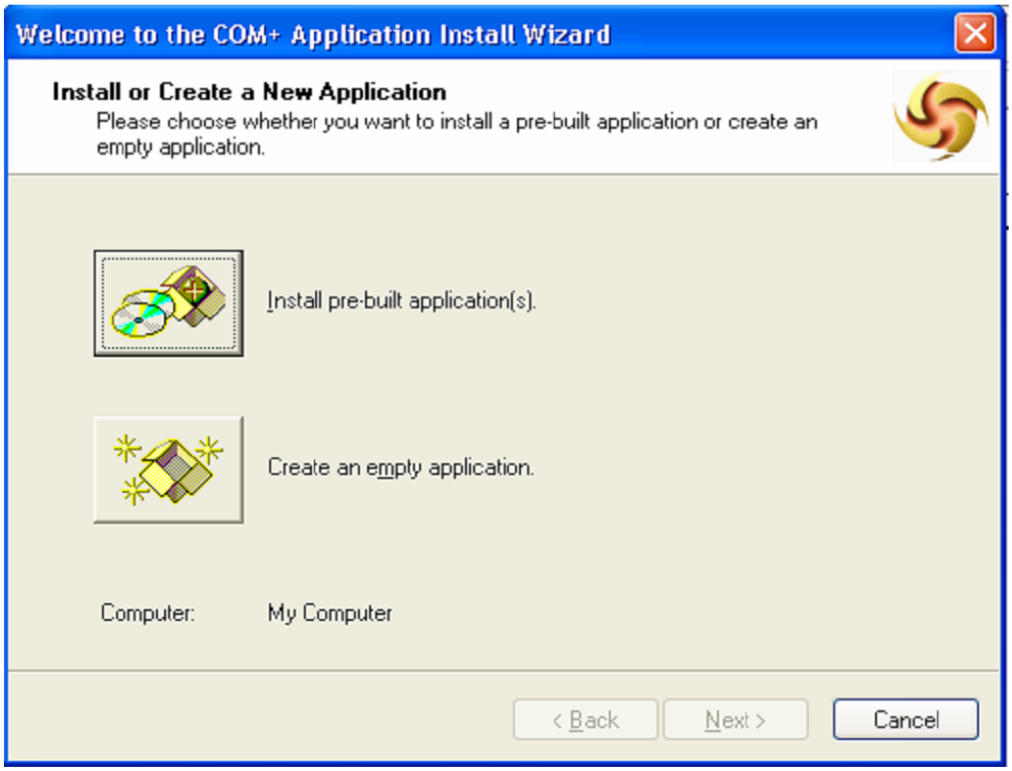


8. Xuất hiện hộp thoại COM Application Install Wizard, nhấn Next.

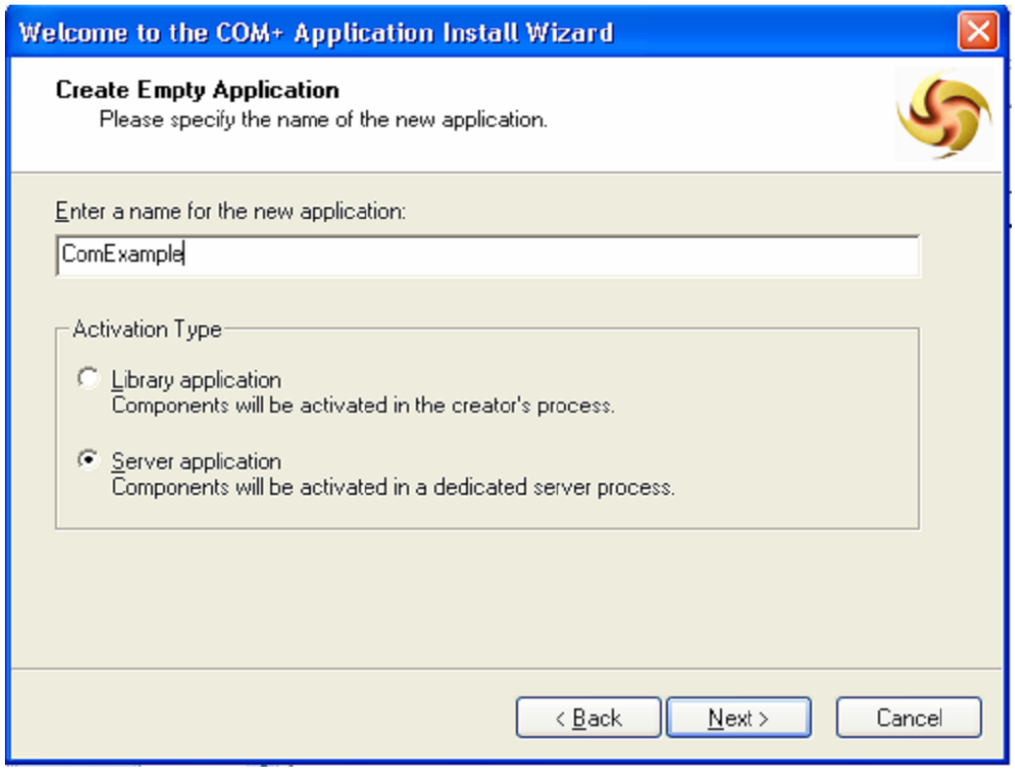


9. Tại đây sẽ có hai lựa chọn. Chọn *Create an emty Application*

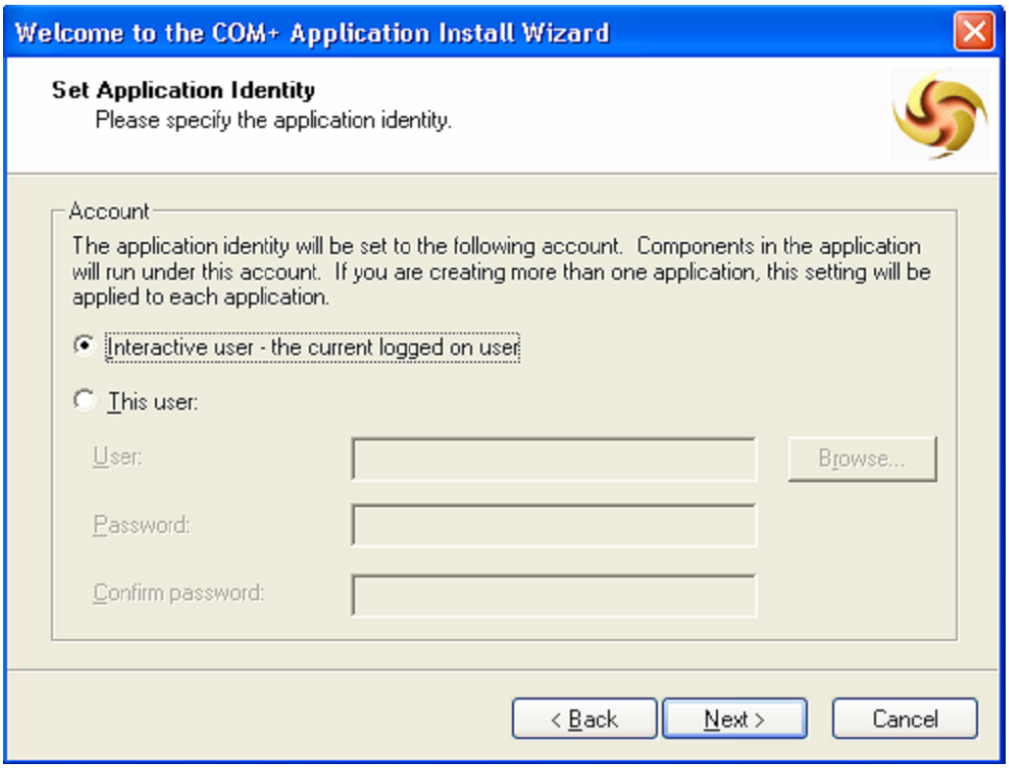
- *Install pre-built application(s)* Nếu muốn xây dựng lại ứng dụng mà ta đã tạo trước đó.
- *Create an emty Application* : Nếu tạo một ứng dụng trống rồi import thành phần COM (.dll) vào.



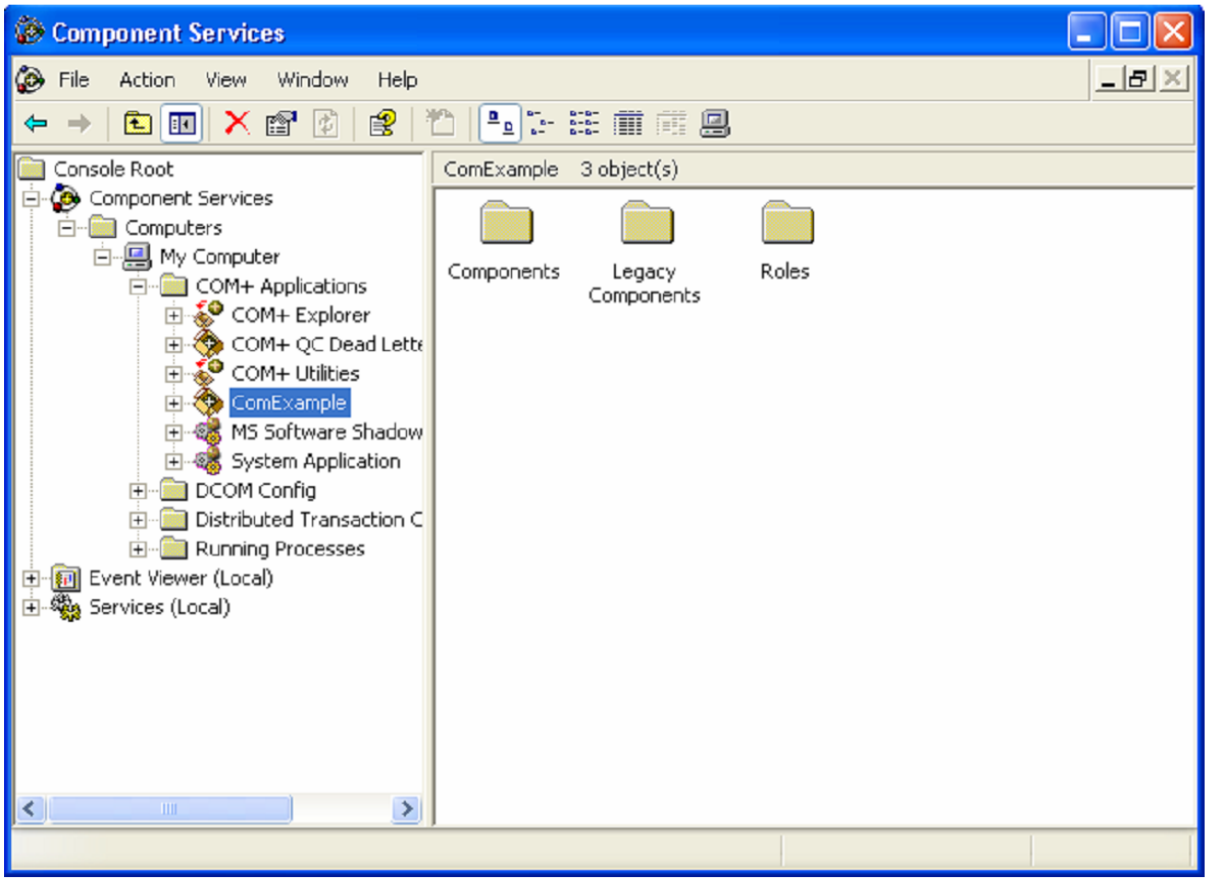
10. Gõ vào vào một cái tên nào đó cho ứng dụng vừa tạo, nhấn vào Next



11. Tiếp theo xuất hiện hộp thoại có hai lựa chọn, ta lựa chọn giá trị default của nó. Nhấn tiếp vào Next.

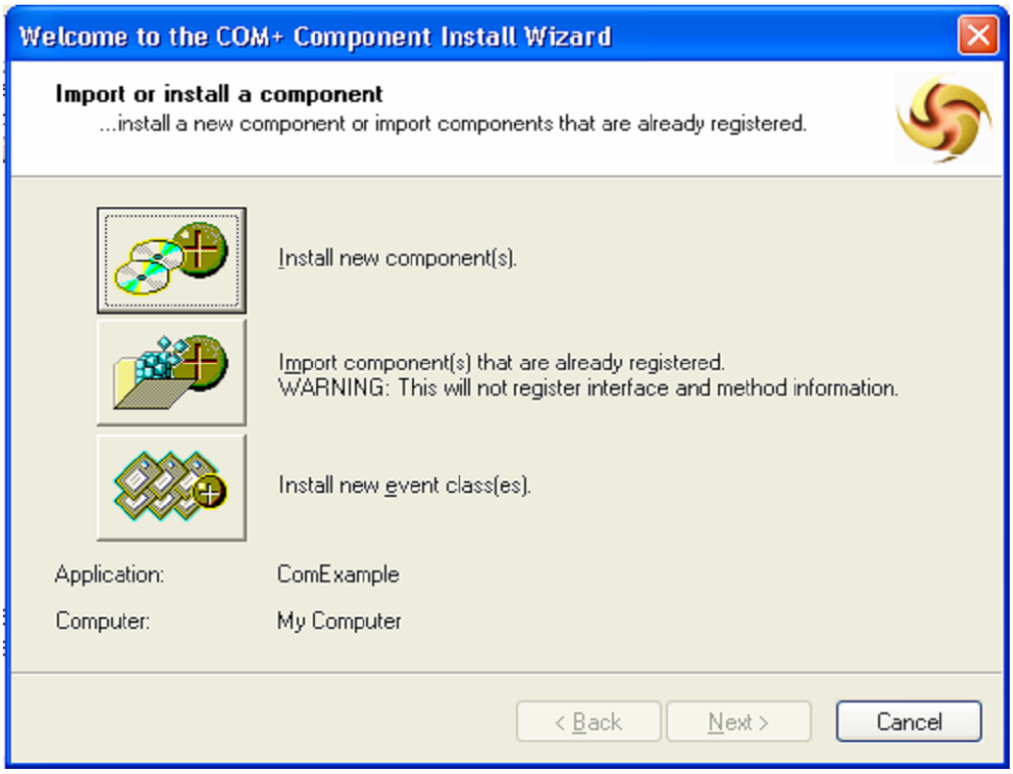


12. Nhấn Finish xuất hiện như sau:

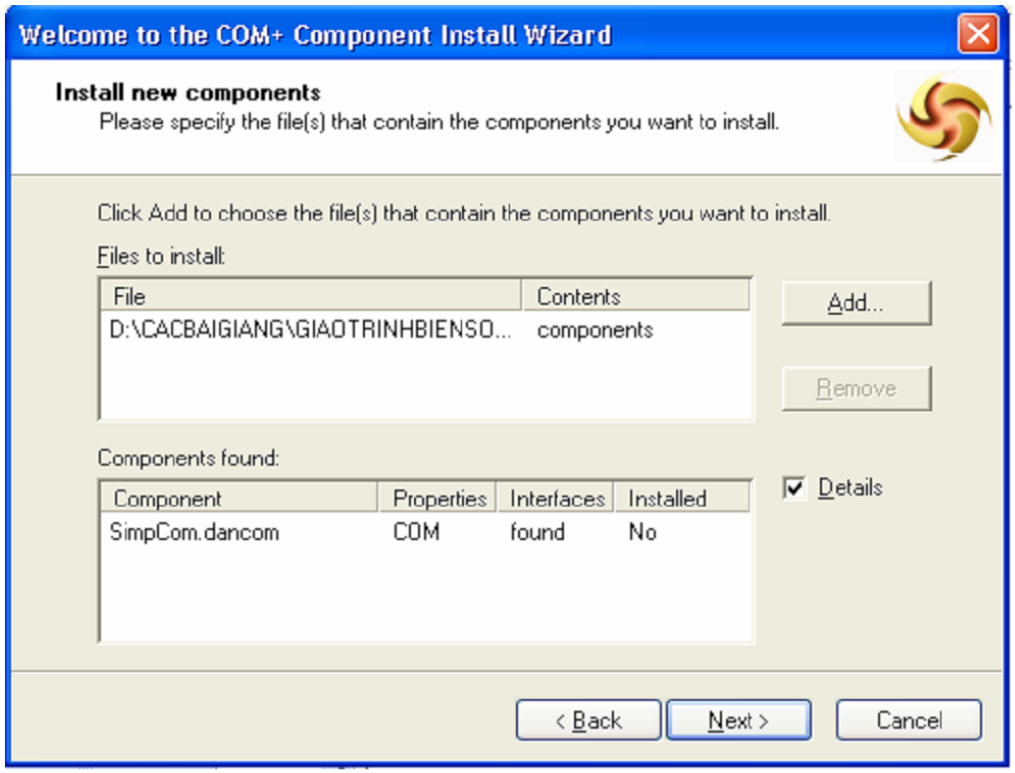


13. Nhấn đúp vào ứng dụng ComExample vừa tạo sau đó nhấn đúp vào Components.
14. Nhấn phải chuột vào Components, trỏ tới New, nhấn vào Component.

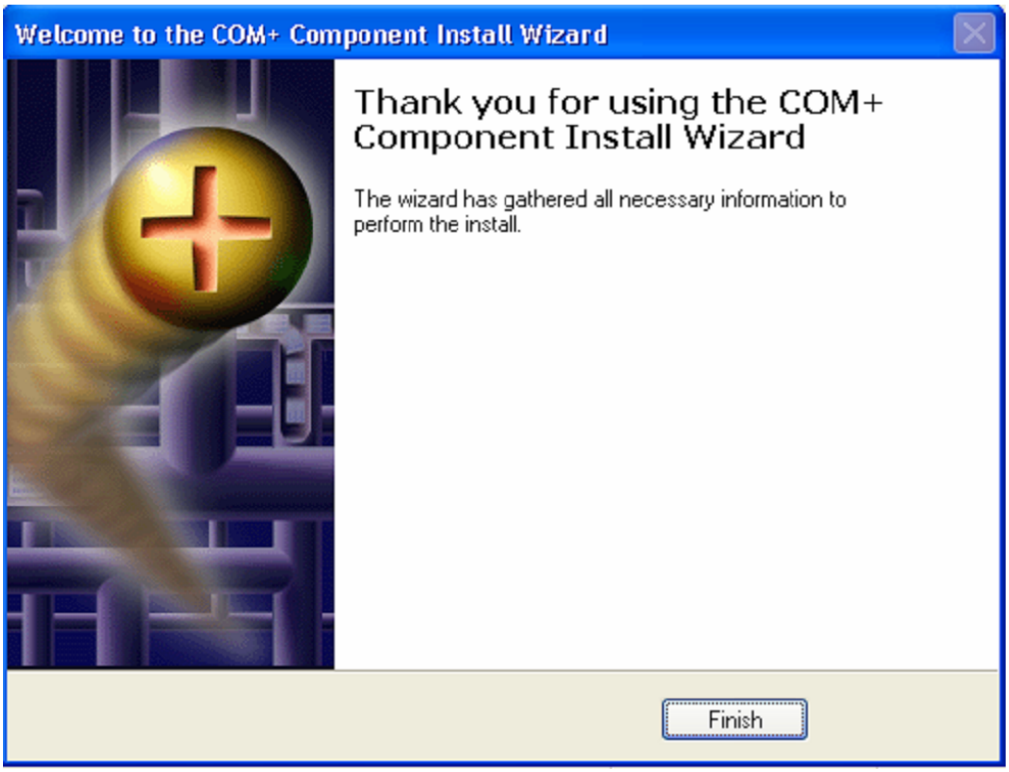
15. Nhấn vào Next, Hộp thoại Import or Install a Component xuất hiện, chọn Install new Component(s).



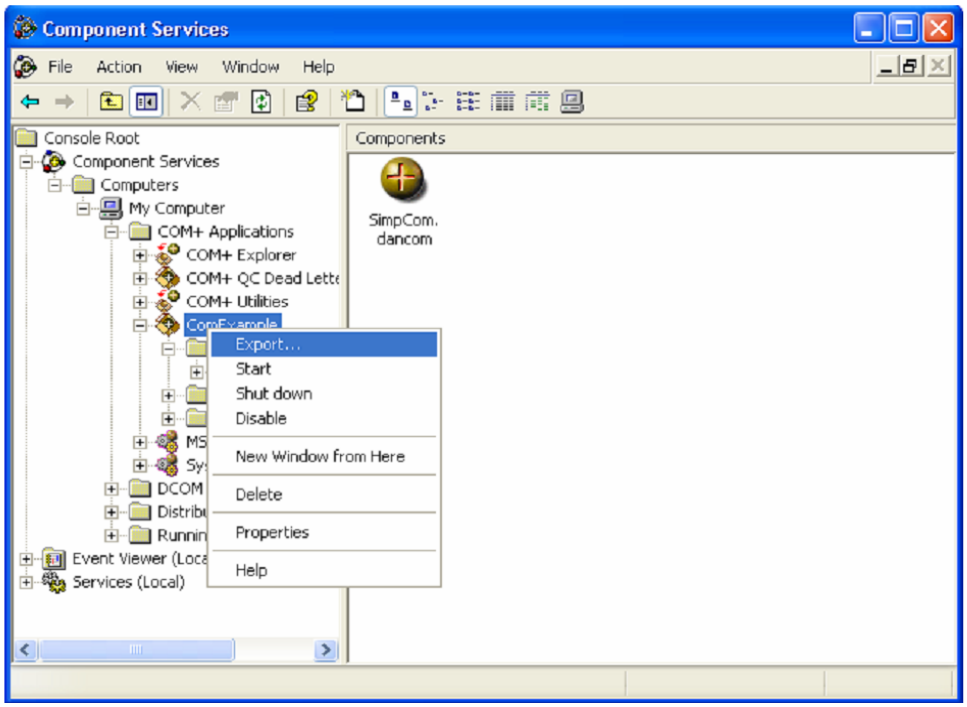
16. Dẫn đến chỗ lưu trữ file .dll, nhấn nút Open



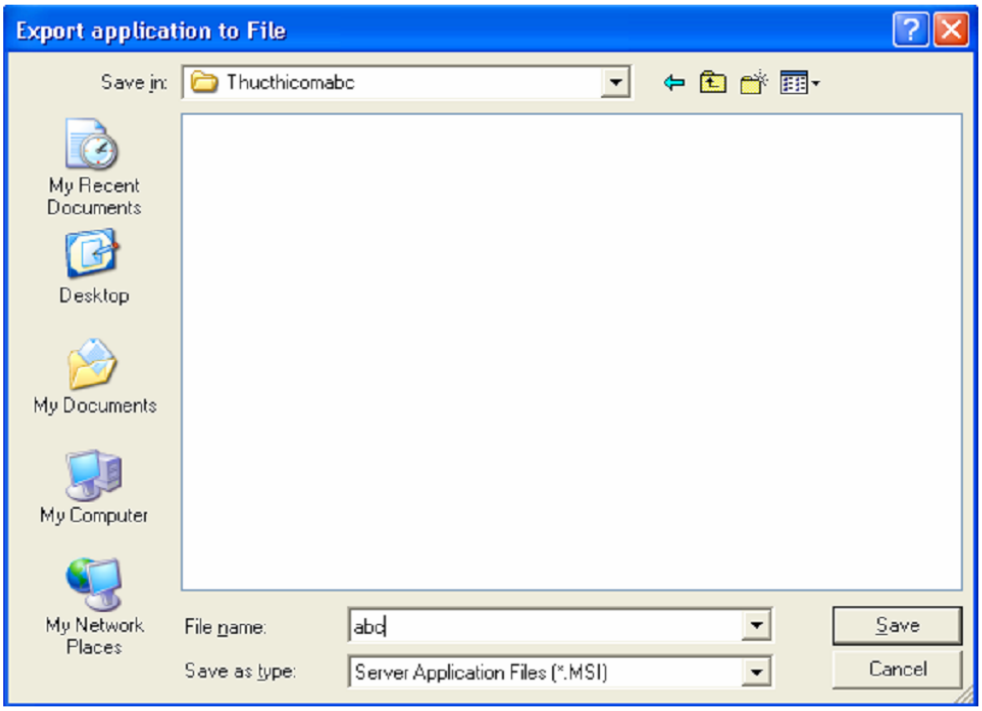
17. Nhấn vào nút Next. Rồi nhấn vào Finish để kết thúc.



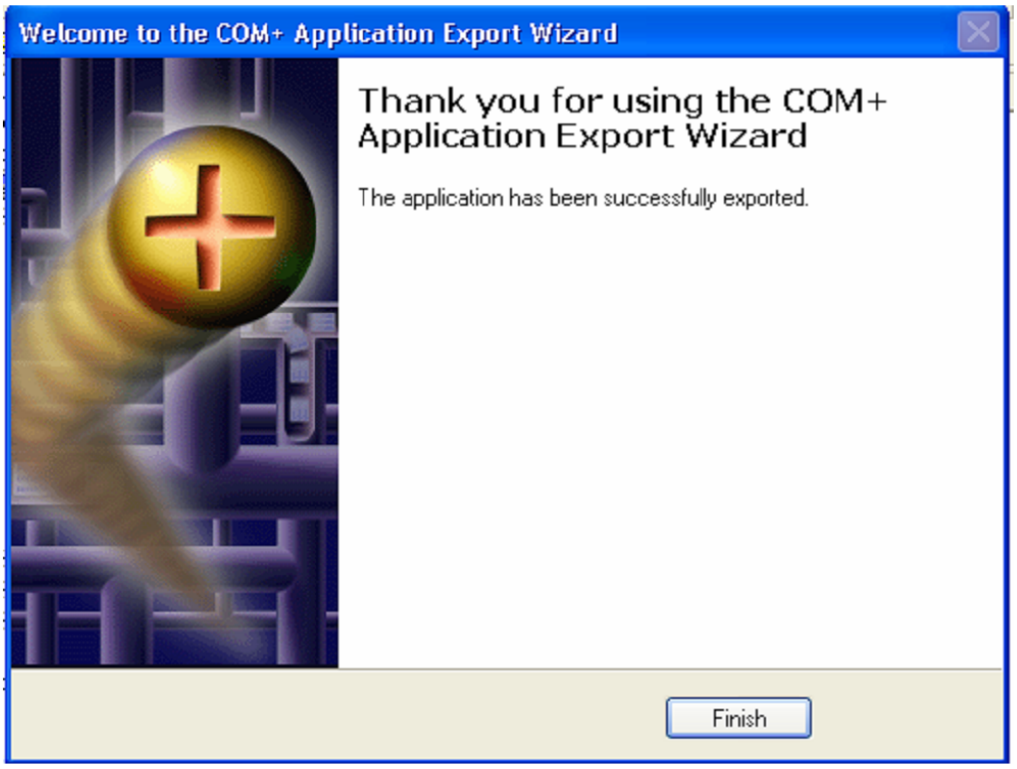
Để cho phép ứng dụng Client truy cập vào thành phần chạy dưới MTS, ta phải đăng ký vào Registry của máy Client để báo với nó rằng thành phần từ xa.
Để Client nhận ra thành phần MTS phân phát từ xa, ta làm như sau:
1. Nhấn phải chuột vào ComExample vừa tạo. Nhấn vào Export…



2. Dùng nút Browse để chọn một thư mục rỗng để export gói. Cho vào tên tập tin, nhấn Save.



3. Nhấn Next. Sau một lúc, hiển thị ra một hợp thoại thông báo sau:



Các bước thực hiện trên máy chạy ứng dụng Client.

1. Chép thư mục mà ta vừa Export được ở trên vào máy Client
2. Chạy tập tin .MSI trong thư mục này để đăng ký vào máy Client.

Muốn gỡ bỏ đăng ký này ra khỏi máy Client ta có thể vào Control Panel\Add/Remove Programs để gỡ bỏ ra.

c. Dùng RDS (Remote Data Access): Kỹ thuật này không nên sử dụng đối với mô hình này vì tốc độ chậm, kỹ thuật này được dùng cho trường hợp máy chạy ứng dụng và máy chạy ứng dụng COM không cùng mạng. Kỹ thuật này sẽ được giới thiệu sau.

Kết luận: Tóm lại đối với mô hình kiến trúc máy chạy ứng dụng Client và máy chạy ứng dụng COM cùng một mạng local thì ta chọn giải pháp "**Dùng MTS (Microsoft Transaction Server) để triển khai ActiveX DLL từ xa".**

Ưu điểm của giải pháp này sau:

- So với dạng ActiveX (.exe) thì dạng này quản lý các thể hiện thành phần và quản lý giao dịch tốt hơn nhiều nhờ sự hỗ trợ của MTS.
- Về vấn đề quản lý security thì dạng này cũng hỗ trợ tốt hơn so với ActiveX (.exe).
- Còn so với RDS thì kỹ thuật này sẽ làm cho ứng dụng chạy nhanh hơn. Vì kỹ thuật RDS được dùng hỗ trợ cho việc truy cập từ xa đối với những máy không cùng mạng. Ngoài ra kỹ thuật này còn hỗ trợ cho những ứng dụng WEB.

Khuyết điểm:

- Máy chạy ứng dụng COM của giải pháp này chỉ có thể chạy trên hệ điều hành Windows NT, hoặc Windows 2000.Còn giải pháp dùng ActiveX(.exe) và giải pháp dùng RDS, máy chạy ứng dụng COM có thể chạy trên nền Windows 98, Windows NT, Windows 2000.
- So với ứng dụng dùng giải pháp dùng ActiveX(.exe) thì ứng dụng dùng giải pháp này sẽ chạy chậm hơn vì có kết hợp với MTS.

Bài 5: Tạo và sử dụng các đối tượng .NET Remoting.

Mục tiêu của bài:

Nhằm trang bị cho người học:

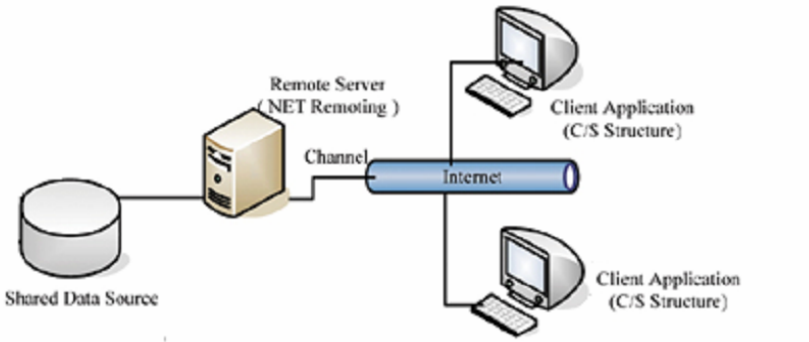
- Kiến thức về .NET Remoting.

- Kiến thức và kỹ năng tạo các đối tượng Server-Activated và Client-Activated.
- Kiến thức và kỹ năng vận chuyển thông điệp cho các ứng dụng.
- Kiến thức về các sự kiện và uỷ nhiệm trong .Net remoting.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

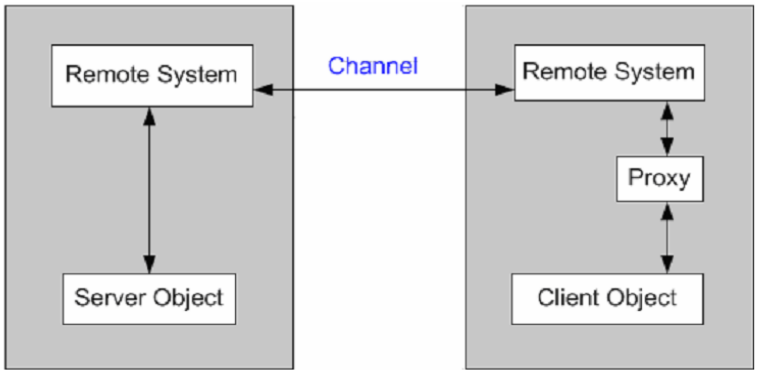
1. Giới thiệu.NET Remoting

.NET Remoting là gì?

Trước hết **.NET Remoting** là một kĩ thuật .NET được giới thiệu từ .NET framework 1.1. Cùng với .NET Webservice, **.NET remoting** là lựa chọn cho giải pháp xử lý tính toán từ xa. **.NET Remoting** là một kĩ thuật cho phép một đối tượng này truy xuất đến một đối tượng khác nằm ở các **Application Domain** khác nhau.



Hình 6.1



Hình 6.2: Giao tiếp giữa một client và một remote object

.NET Remoting và Distributed COM

Trước đây người ta thường thực hiện việc giao tiếp giữa các process bằng cách sử dụng **Distributed COM** hay còn gọi là **DCOM**. **DCOM** đã rất hữu ích cho những chương trình chạy trên các máy tính cùng loại và nằm trong cùng một mạng.

Tuy nhiên, **DCOM** trở nên lỗi thời vì nó không thể chạy trên Internet. **DCOM** dựa trên một tập giao thức mà không phải object nào cũng hỗ trợ và điều này

khiến **DCOM** không chạy được trên những platform khác nhau. Ngoài ra, **DCOM** sử dụng nhiều port trong khi các port ấy thường bị chặn bởi **firewall**.Tất nhiên mở những port đó để nó hoạt động được không khó nhưng đó là một trong những phiền phức.

.NET Remoting khắc phục những yếu kém của **DCOM** bằng cách hỗ trợ nhiều giao thức khác nhau.

.NET Remoting và Web Services

Về khía cạnh xử lý từ xa thì Web Services hoàn toàn tương tự như **.NET Remoting.** Thậm chí người ta có thể làm cho **.NET Remoting** trở thành 1 Web Services bằng cách host nó trong IIS.

.NET Remoting thì tùy giao thức và định dạng message mà nó có thể truyền đi thông tin như thế nào.

Ngoài ra theo như giới thiệu thì **.NET Remoting** có cho phép đối tượng được truyền vào theo cả kiểu tham chiếu(**reference**) và tham trị (**value**). **.NET Remoting** yêu cầu phía clients phải là **.NET application**.

Channels

Trong kĩ thuật **.NET Remoting** thì **Channel** được hiểu như là một kênh để giao tiếp giữa client và server.

Một object từ client sẽ thông qua **Channel** để giao tiếp với object phía server, **Channel** sẽ truyền tải những **message** từ hai phía. Có hai **channel** chính là **TcpChannel** và **HttpChannel** tương ứng với các giao thức **TCP** và **HTTP**. Ngoài ra, **TcpChannel** và **HttpChannel** đều có khả năng extend thành những **Custom Channel** của chúng ta.

Làm sao để tạo một Object có thể Remote được trong .NET Remoting ?

Một **Object remote** được chỉ là một object thông thường nhưng phải được **inherit** từ **MarshalByRefObject**.

2. Các đối tượng của .Net Remoting

a. SingleCall

Đối tượng này phục vụ một và chỉ một yêu cầu tới. Đối tượng này thật hữu ích trong viễn cảnh mà các đối tượng được yêu cầu làm việc ít nhất. SingleCall không được yêu cầu lưu trữ thông tin trạng thái.

b. Singleton

Đối tượng này phục vụ nhiều Client và chia sẻ dữ liệu cho các yêu cầu của client bằng cách lưu trữ thông tin trạng thái. Chúng thật hữu dụng trong mọi trường hợp mà dữ liệu cần phải chia sẻ giữa các Client.

c.Client-Activated

3.Thành phần của một ứng dụng đơn giản.

Bao gồm 3 project: Client, Server và Remote.

4.Các bước xây dựng chương trình:

Bước 1: Ta xây dựng lõi của chương trình đó chính là **Remoting Project**. Lõi này sau khi được xây dựng sẽ trở thành thư viện dll và được **Add Reference** vào các Project còn lại.

Bước 2: Xây dựng Server Project: Trên server chúng ta khởi tạo một kênh (channel)dùng để nghe. Sau đó chúng ta dùng **ChannelServices** để đăng ký kênh này.

Ví dụ:

```
TcpChannel channel = new TcpChannel(9999);
ChannelServices.RegisterChannel(channel);
RemotingConfiguration.RegisterWellKnownServiceType(typeof(myRemoteObject),
"HelloWorld", WellKnownObjectMode.Singleton);
```

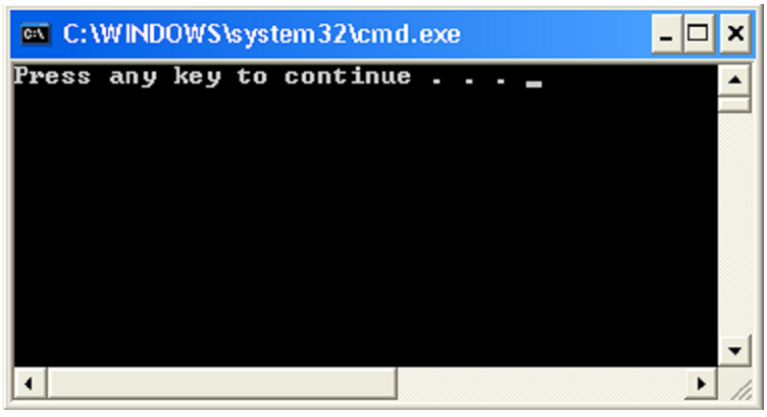
Bước 3: Xây dựng Client Project: Tại Client chúng ta khởi tạo một kênh(channel)và đăng ký kênh này.

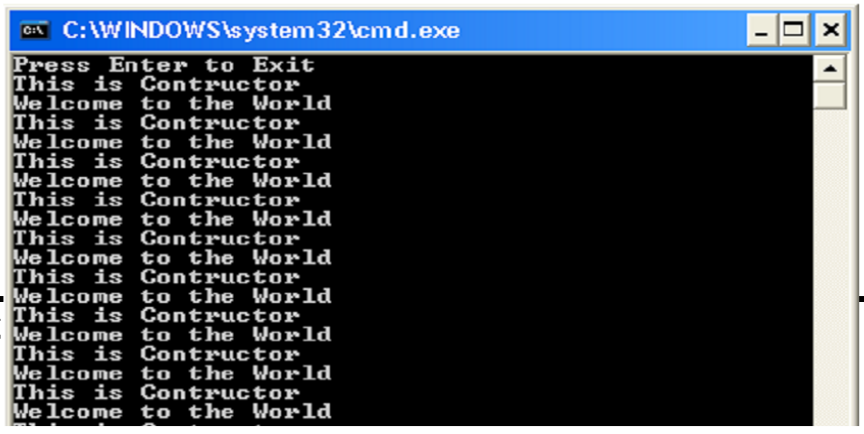
Ví dụ:

```
TcpChannel channel = new TcpChannel();
ChannelServices.RegisterChannel(channel);
remoteObj =
(Remote.myRemoteObject)Activator.GetObject(typeof(Remote.myRemoteObject),
"tcp://localhost:9999/HelloWorld");
```

Sau đây chúng ta sẽ xét một ví dụ đơn giản để minh họa cho các bước này:

Ví dụ 6.1 Tạo ứng dụng đơn giản sử dụng mô hình Remoting như sau:





Bước 1: Project Class Library có tên: Example có nội dung như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Example
{
   public class MyClass:MarshalByRefObject
   {
      public MyClass()
      {
         Console.WriteLine("This is Contructor");
      }
      public void Hello()
      {
         Console.WriteLine("Welcome to the World");

      }
   }
}
```

Bước 2: Tạo ứng dụng Console phía Server như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Channels.Http;
using Example;
namespace Server
{
   class Exp_Server
   {
      static void Main(string[] args)
      {
         TcpServerChannel channel = new TcpServerChannel(9999);
         ChannelServices.RegisterChannel(channel,true);
        RemotingConfiguration.RegisterWellKnownServiceType(typeof(MyClass),
        "HelloWorld",WellKnownObjectMode.SingleCall);
         Console.WriteLine("Press Enter to Exit");
         Console.ReadLine();
```

```
        }
    }
}
```

Bước 3: Tạo ứng dụng phía Client.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using Example;
namespace Client
{
    class Exp_Client
    {
        static void Main(string[] args)
        {
            TcpClientChannel c_channel = new TcpClientChannel();
            ChannelServices.RegisterChannel(c_channel,true);
            MyClass obj = (MyClass)Activator.GetObject(typeof(MyClass),
           "tcp://localhost:9999/HelloWorld");
            int cnt = 0;
            for (cnt = 0; cnt < 10; cnt++)
            {
                obj.Hello();
            }
        }
    }
}
```

Ta xét thêm một ví dụ đơn giản về **Remotable Object**. Lớp **SampleObject** có một số phương thức đơn giản. Giá trị trả về là kiểu số nguyên.

Nếu ta muốn trả về kiểu dữ liệu tự định nghĩa hoặc một thể hiện của lớp ta định nghĩa thì lớp đó của ta phải được khai báo với **attribute Serializable**.

Ví dụ 6.2:

```
using System;
public class SampleObject : MarshalByRefObject
{
    public int Add(int a, int b)
    {
        int c = a + b;
        return c;
    }
    public int Subtract(int a, int b)
    {
        int c = a - b;
```

```
        return c;
    }

    public int Multiply(int a, int b)
    {
        int c = a * b;
        return c;
    }
    public int Divide(int a, int b)
    {
        int c;
        if (b != 0)
            c = a / b;
        else
            c = 0;
        return c;
    }
}
```

Tạo chương trình Server để host Remotable Object

Ta tạo chương trình server để lắng nghe những request từ phía client. Trong ví dụ này chúng ta sẽ sử dụng TCP/IP channel. Đầu tiên chúng ta tạo một **instance channel** và đăng kí một port tương ứng cho nó. Khi có một Request từ phía client, server sẽ nhận request đó và **Remote Object** của chúng ta sẽ thực thi Request này.

Trong .NET Remoting, có hai cơ chế để tạo instance của Remote Object rồi từ đó thực thi request: **Singleton** và **Singlecall**. Tùy vào mục đích sử dụng, nhu cầu của chương trình mà server của ta có thể khai báo theo cơ chế **WellKnownObjectMode.SingleCall**, hay **WellKnownObjectMode.Singleton.** Khi khai báo **Singleton**, **Remote Object** sẽ được sinh ra, thực thi request, reply lại phía client và sau đó, object này vẫn được lưu lại chứ không bị hủy đi. Đến khi nào process chạy chương trình server kết thúc thì instance này mới bị trình hốt rác Garbage Collector hốt đi.

Và ngược lại, khi khai báo là **SingleCall**, Remote Object sẽ được khởi tạo và hủy đi đối với mỗi lần nhận request từ phía client, cơ chế này tương tự như mô hình .NET Web Service truyền thống.

Nếu bạn muốn sử dụng .NET Remoting trong IIS thì không cần tạo một chương trình server như thế này. Và tất nhiên, IIS chỉ hỗ trợ **HttpChannel**. Nếu host .NET Remoting bên trong IIS bạn sẽ mặc nhiên sử dụng được cơ chế **Authentication** của IIS, ngược lại nếu làm một chương trình server để host như trên thì ta phải cài đặt cơ chế **Authentication** của riêng mình. Để host một Remote Object bên trong IIS, trước tiên phải tạo 1 Virtual Directory cho application, sau đó đặt đoạn code đăng kí service bên trong event Application_Start (file global.asax).

Trong ví dụ này, chúng ta sẽ không sử dụng IIS mà sẽ tạo một console application.

Có nhiều lựa chọn khi không sử dụng IIS, ta có thể sử dụng console application, Win form application nhưng trong thực tế, người ta sẽ sử dụng một Windows Service để làm. Còn Console application hay Winform Application thường chỉ dùng để minh họa.

Trong ví dụ này, chúng ta sẽ sử dụng port **9999** cho may mắn. Có thể một chương trình nào đó trong máy của chúng ta đã sử dụng port này, nếu bị như vậy ta phải chọn port khác. Và sau cùng, để kiểm tra xem máy ta đang lắng nghe trên những port nào(port nào đã bị sử dụng) thì ta dùng lệnh "**netstat -a**" trong command prompt.

Còn bây giờ, hãy xem một console application project với một class tên là SampleServer. Trong project này chúng ta đã thêm reference tới System.Runtime.Remoting vào trong project để nó có thể chạy được.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
public class  Server
{
public static int Main()
{
        TcpChannel chan = new TcpChannel(9999);
        ChannelServices.RegisterChannel(chan, false);
RemotingConfiguration.RegisterWellKnownServiceType(typeof(SampleO
bject) , "SampleNetRemoting", WellKnownObjectMode.SingleCall);
        Console.WriteLine("Hit <enter> to exit...");
        Console.ReadLine();
}
}
```

Tạo chương trình client để sử dụng Remote Object.

Chương trình client trong ví dụ này cũng khá đơn giản, nó sẽ connect vào server, tạo một instance của Remote Object và excute method tính tổng, hiệu, tích, thương.

Chúng ta lưu ý rằng trong cả chương trình client và chương trình server đều phải reference tới class **SampleObject**. Client sẽ gọi method của instance **SampleObject**, nhưng server sẽ thực thi xử lý nó chứ không phải phía client.

```
using System;
```

```
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
public class Client
{
public static int Main (string[] argv)
{
        TcpChannel chan = new TcpChannel();
        ChannelServices.RegisterChannel(chan, false);
        SampleObject obj = (SampleObject)Activator.GetObject( typeof(SampleObject),
                                        "tcp://localhost:9999/SampleNetRemoting");
        if (obj = = null)
                System.Console.WriteLine("Could not locate server");
        else
        {
                int a = Convert.ToInt32(argv[0]);
                int b = Convert.ToInt32(argv[1]);
                int c = obj.Add(a, b);
                Console.WriteLine("a + b = {0}", c);
                c = obj.Subtract(a, b);
                Console.WriteLine("a - b = {0}", c);
                c = obj.Multiply(a, b);
                Console.WriteLine("a * b = {0}", c);
                c = obj.Divide(a, b);
                Console.WriteLine("a / b = {0}", c);
        }
Console.ReadKey();
}
}
```

Test thử chương trình

Trước tiên chạy chương trình server, Chúng ta sẽ thấy message "Press the enter key to exit" trong cửa sổ console. Như vậy server của ta đang lắng nghe trên port **9999**. Bây giờ ta chạy chương trình client và sẽ nhìn thấy kết quả trả về trên màn hình.

Chúng ta có thể chạy nhiều client để cùng request đến 1 server nhưng không thể chạy nhiều server.

Chúng ta có thể copy chương trình server sang một máy khác để chạy thử, còn máy đó sửa lại chương trình client, sửa "**localhost**" thành IP của máy kia và chạy thử để thấy kết quả.

Tóm tắt:

Ví dụ ở trên đã sử dụng code C# để khai báo các cấu hình cho server và client tuy nhiên .NET Remoting cho phép ta cấu hình trước trong file config (App.config). Chúng ta có thể tham khảo một số resource phía dưới để biết cách làm.

.NET Remoting là một trong những kĩ thuật tiện lợi cho những chương trình dạng **Distributed Computing**.

Cách sử dụng nó phức tạp hơn Web Service tuy nhiên nếu ta muốn tăng performance thì .NET Remoting với Singleton và TCP channel sẽ là lựa chọn rất tốt.

Với sự ra đời của .NET Framework 3.x, Microsoft đã giới thiệu nền tảng mới hơn cho các kĩ thuật RPC, đó là WCF mạnh hơn .NET Remoting rất nhiều.

Khai báo, cài đặt và đăng ký giao diện từ xa

Để cho chương trình có tính khả chuyển cao thay vì người ta xây dựng lớp Remote Object như ví dụ trên chúng ta khai báo một giao diện là lớp Remote Object và trong chương trình phía Server ta sẽ cài đặt giao diện này và đăng ký giao diện từ xa.

Như vậy để triển khai một hệ thống Remoting ta có 3 chương trình: Giao diện Remote Object, chương trình Server triển khai giao diện và đăng ký giao diện từ xa, chương trình Client triệu gọi phương thức từ xa.

- Khai báo giao diện từ xa.
- Cài đặt và đăng ký giao diện từ xa.

Triệu gọi phương thức từ xa

Chương trình phía Client chúng ta triệu gọi phương thức được cung cấp bởi giao diện từ xa đã được đăng ký và cung cấp bởi Server.

Bài 6: Assembly và Global Assembly Cache (Bài đọc thêm)

Mục tiêu của bài:

Nhằm trang bị cho người học:

- Kiến thức về Assembly.
- Kiến thức và kỹ năng tạo Assembly bằng Visual Studio.
- Tạo Assembly bằng tiện ích CSC.EXE
- Kiến thức và kỹ năng về Private Assembly, Share Assembly, Friend Assembly.
- Có thái độ làm việc cẩn thận, làm việc nhóm, bảo vệ máy tính khi làm việc.

1. Giới thiệu Assembly

Assembly được xem như một đơn vị chứa đựng khối một hay nhiều ứng dụng .Net FramWork, bao gồm các ứng dụng dạng EXE hay DLL.

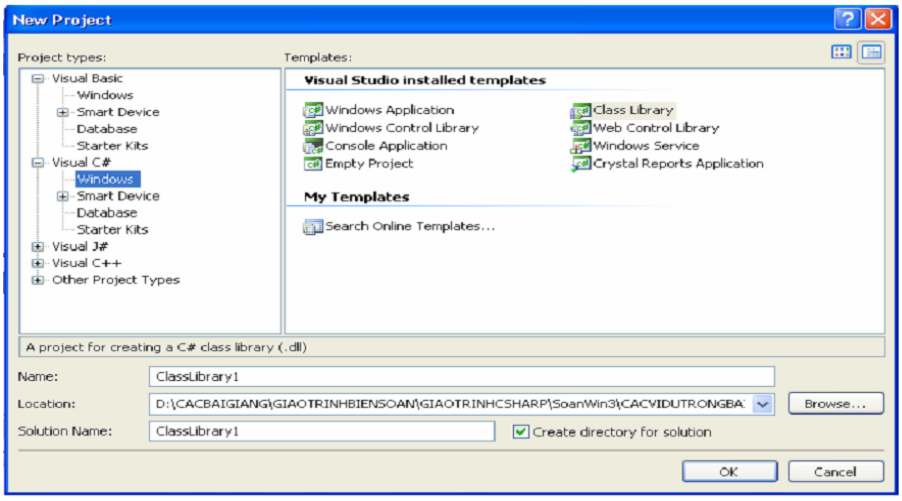
Assembly chứa đựng siêu dữ liệu(metadata) mà nó có thể thể hiện được chi tiết về loại đối tượng, khóa nhận dạng, tài nguyên như hình ảnh, thuộc tính liên quan đến **Asssembly** khác,....

Assembly chỉ được nạp khi có nhu cầu, điều này cho phép nó quản lý tốt tài nguyên trong dự án lớn.

Có hai cách để tạo **Assembly** cách thứ nhất là sử dụng Visual Studio, cách thứ hai là sử dụng tiện ích csc.exe(là một tiện ích miễn phí đi kèm theo .Net FrameWork).

1.1 Cách tạo Assembly dùng Visual Studio

Khi chọn Project ta chọn loại Project là Class Library như sau:



1.2 Cách tạo Assembly dùng tiện ích csc.exe

Cú pháp:

csc /t: library/out:libname.dll /r:reflibname.dll cscfilename.cs

Trong đó, libname.dll là tên của tập tin Assembly tạo ra, nếu không khai báo thì tên sẽ trùng với tên tập tin .cs.

Tùy chọn /r: là danh sách các tập tin Assembly khác đã được tham chiếu và khai báo sử dụng trong lớp thuộc Assembly muốn biên dịch.

Ví dụ trong lớp ta khai báo để tham chiếu đến hai Assembly là Example1.dll và Example2.dll như sau:

/r: Example1.dll Example2.dll

Tùy chọn /t:library chỉ định cho tiện ích csc.exe biên dịch ra tập tin .dll.

2. Private Assembly

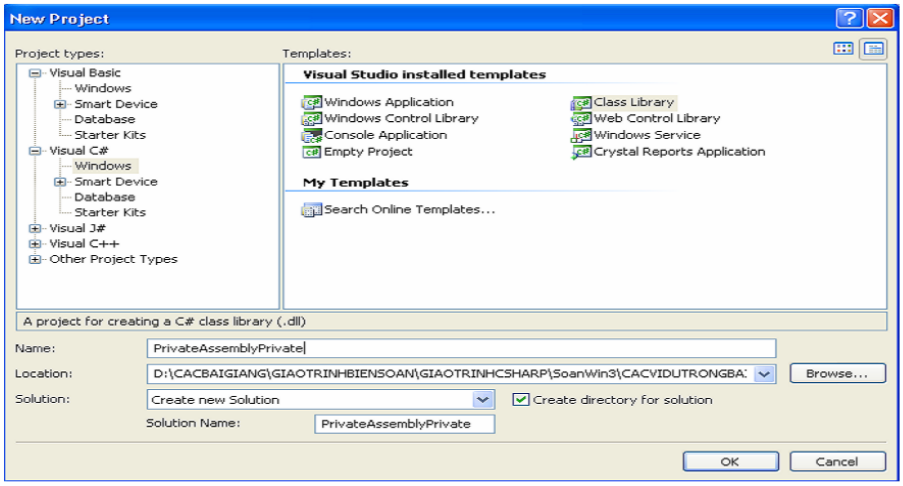
Private Asembly đặt cùng thư mục với tập tin thi hành và duy nhất trong mỗi ứng dụng. Bởi vậy nếu khi thay đổi cấu trúc thì những tham chiếu đến nó trước đó sẽ không có hiệu lực bởi vì khi tham chiếu trong visual thì bản copy được tạo ra và đặt cùng thư mục với tập tin thi hành. **Assembly** tạo ra thì nó mặc định là **Private.**

Vậy khi chúng ta cần xây dựng các thành phần dùng chung thì ta sẽ tạo **Assemply** rồi tham chiếu đến chúng mỗi khi cần.

2.1 Cách tạo

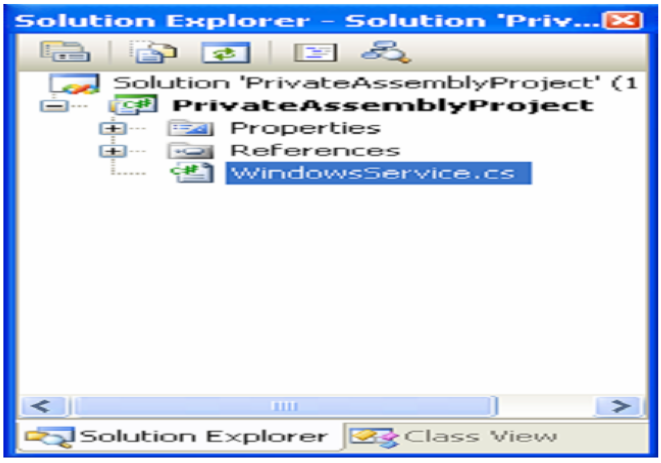
Tạo một **Private Assembly** cung cấp các phương thức để làm việc với dịch vụ của hệ điều hành.

Ta tạo như sau:



Sau khi tạo Project thì không gian tên mặc định có tên cùng với tên của Project là **PrivateAssemblyProject.**

Tiếp theo ta đổi tên lớp thành **WindowsService.cs** như sau:



Tiếp theo ta tham chiếu đến Assembly của .NET FrameWork có tên System.ServiceProcess bằng cách chọn vào thực đơn Project chọn Add Reference.....

Khai báo không gian tên:

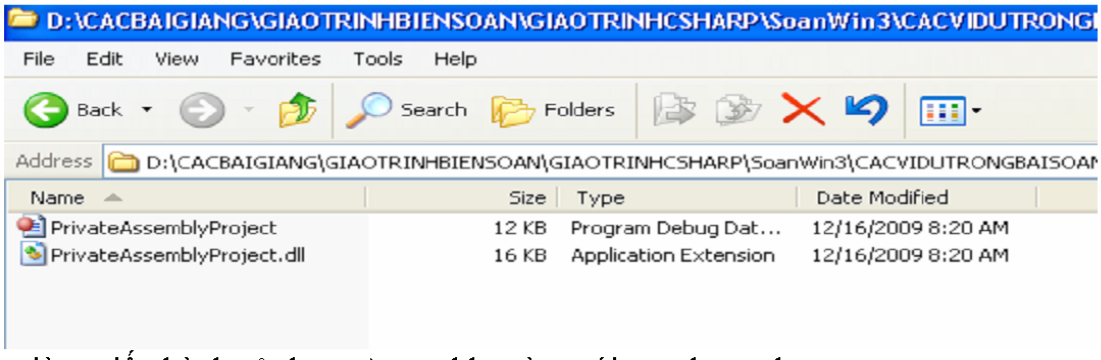
using System;

using System.ServiceProcess;

Khai báo phương thức thứ nhất có tên GetService trả về trạng thái của một dịch vụ và phương thức Overload thứ hai trả về mảng tên dịch vụ như sau:

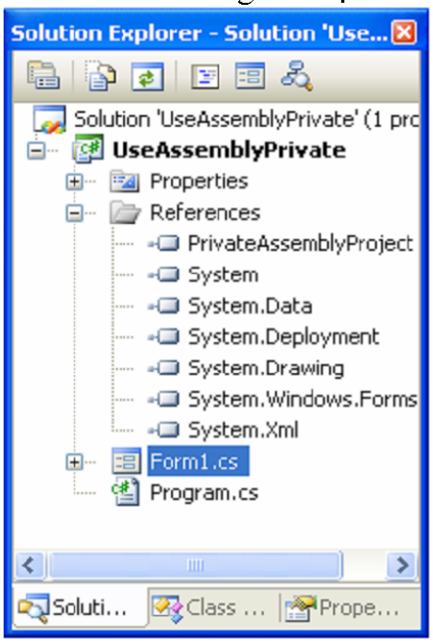
```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceProcess;
namespace PrivateAssemblyProject
{
   public class WindowsService
   {
      public string GetService(string serviceName)
      {
         ServiceController sc = new ServiceController();
         sc.ServiceName = serviceName;
         return sc.Status.ToString();
      }
      public string[] GetService()
      {
         ServiceController[] scs = ServiceController.GetServices();
         string[] _arrServiceName = new string[scs.Length];
         int i =0;
         foreach(ServiceController sc in scs)
         {
            _arrServiceName[i] = sc.ServiceName;
            i++;
         }
         return _arrServiceName;
      }
   }
}
```

Ta tiến hành biên dich Project ở chế độ Release sẽ xuất hiện file Dll như hình sau đây:



Bây giờ ta tiến hành sử dụng Assembly vừa mới tạo được như sau:

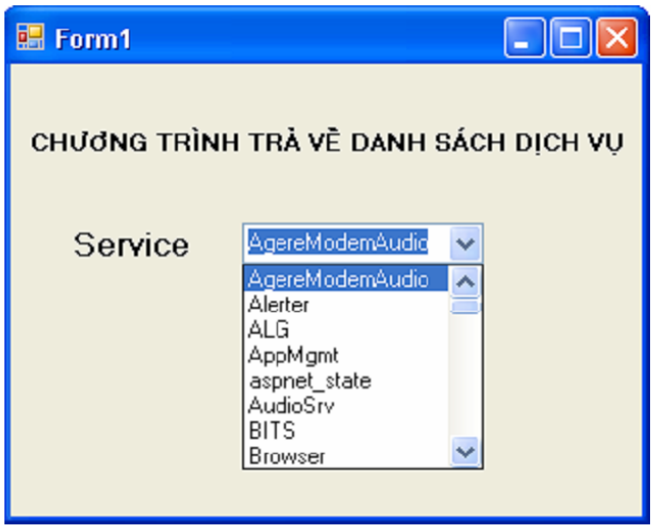
Tạo thêm một Project mới đặt tên là **UsePrivateAssembly** Sau đó chọn Add Rerence để tham chiếu đến tập tin **PrivateAssemblyProject.dll** trong thư mục PrivateAssemblyProject\Release và cuối cùng ta được như sau:



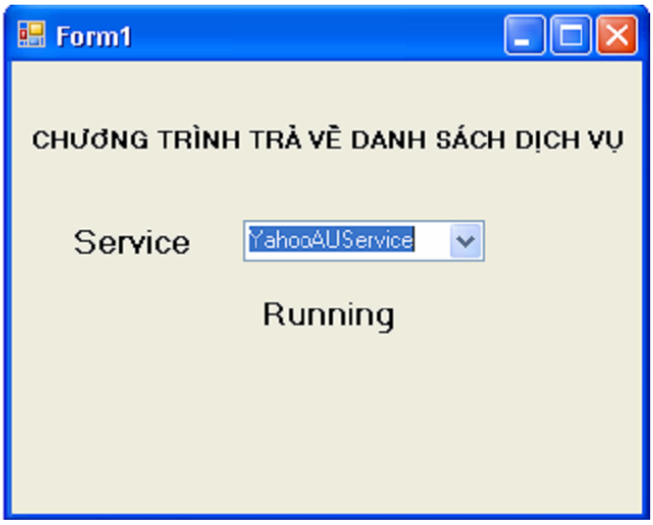
Trong phần Code ta khai báo thêm namespace như sau:

```
using PrivateAssemblyProject;
```

Ta thêm các điều khiển sau: ComboBox và Label vào Form rồi khai báo trong biến cố Form_Load và sử dụng phương thức GetService như sau:



Trong biến cố **SelectedIndexChanged** để trình bày trạng thái của dịch vụ đã chọn, ở đây ta gọi phương thức GetService thứ nhất như sau:



Code như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using PrivateAssemblyProject;
namespace UseAssemblyPrivate{
    public partial class Form1 : Form{
        public Form1(){
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e){
            WindowsService ws = new WindowsService();
            this.cboService.DataSource = ws.GetService();
```

```
        }
        private void cboService_SelectedIndexChanged(object sender, EventArgs e){
            if (cboService.SelectedValue != null){
                WindowsService ws = new WindowsService();
                this.lblStatus.Text = ws.GetService(cboService.SelectedValue.ToString());
            }
        }
    }
}
```

3. Share Assembly

Share Assembly có thể chia sẻ cho nhiều ứng dụng và chỉ tồn tại với một tên và phiên bản duy nhất trong mỗi máy. Do vậy hai Assembly cùng tên nhưng khác phiên bản sẽ cùng tồn tại trong **Global Assembly Cache(GAC).**

Để Assembly trở thành **Share Assembly** thì **Assembly** này phải được gán bởi một **Strong Name** và đăng kí vào GAC.

3.1 Cách tạo

Khi tạo Assembly mặc định sẽ là Private Assembly để trở thành Share Assembly thì phải gán Strong Name và đăng kí vào GAC.

Ta sẽ tiến hành tạo thông qua ví dụ cụ thể sau đây:

Tạo mới Project có tên ShareAssemblyProject sau đó đổi tên Class1.cs thành Access.cs và khai báo không gian tên như sau:

```
using System;
using System.Data.OleDb;
using System.Data;
```

Sau đó ta khai báo phương thức GetTable như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.OleDb;

namespace ShareAssemblyProject
{
    public class Access
    {
        private string strError = "";
        public string Errror
        {
            get
            {
                return Errror;
            }
        }
        public DataTable GetTable(string strPath, string strSQL)
        {
```

```
            OleDbConnection Con = new OleDbConnection();
            Con.ConnectionString = "Provider = " + " Microsoft.Jet.OLEDB.4.0;" + "Data
                                                        Source =" + strPath;
            OleDbDataAdapter da = new OleDbDataAdapter(strSQL, Con);
            DataTable dt = new DataTable();
            try
            {
               da.Fill(dt);
            }
            catch(Exception ex)
            {
               strError = "Error:" + ex.Message;
            }
            Con.Dispose();
            return dt;
        }
    }
}
```

3.2 Strong Name

Để có thể sử dụng shared assembly, bạn cần tuân thủ 3 đòi hỏi sau đây:

- Ta phải báo đúng assembly mà bạn muốn nạp vào. Do đó, bạn cần có một tên duy nhất mang tính toàn cục (global unique name) được gán cho shared asembly.
- Ta cần bảo đảm là assembly không bị giả mạo. Nghĩa là, bạn cần có một digital signature (chữ ký số) khi assembly được tạo dựng.
- Ta cần bảo đảm là assembly ta đang nạp vào đúng là assembly được phép của tác giả tạo ra assembly. Do đó, ta cần ghi nhận mã nhận diện người sáng tác (originator).

Do đó, khi ta muốn tạo ra một assembly có thể được chia sẻ sử dụng bởi nhiều ứng dụng khác nhau trên một máy tính nào đó, bước đầu tên là tạo ra một tên chia sẻ duy nhất đối với assembly, Tên này được gọi là strong name thường chứa những thông tin sau đây:

- Một tên kiểu string mang tính thân thiện và thông tin tuỳ chọn về culture (giống như với private assembly)
- Một mã nhận diện phiên bản.
- Một cặp mục khoá public-private
- Một chữ ký số (digital signature).

Việc hình thành một strong name đều được dựa trên một phương thức mật mã hoá chuẩn mục khoá công cộng (standard public key encryption).

Khi ta tạo một shared assembly ta phải kết sinh một cặp mục khoá public-private. Một đoạn mã băm (hash code) được lấy ra từ những tên và nội dung của các tập tin trong assembly.

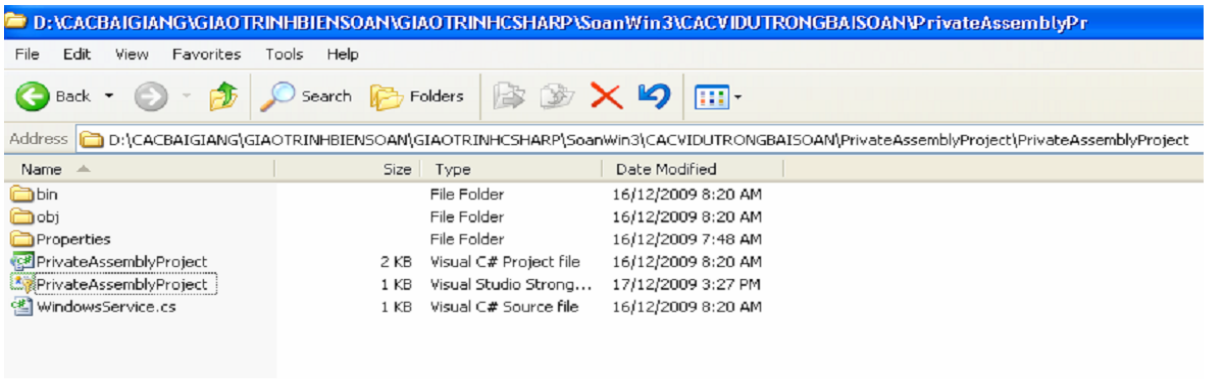
Mã băm này sẽ được mật mã hoá với một private key đối với assembly và được đưa vào manifest. Đây được biết là ký tên assembly (signing the assembly). Public key sẽ được sát nhập vào strong name của assembly.

Nói tóm lại, một cặp mục khoá sẽ được bao gồm trong chu kỳ build sử dụng một trình biên dịch .NET, trình này sẽ liệt kê một token của public key trên manifest của assembly (thông qua tag [.publickeytoken]). Private key sẽ không được liệt kê ra trên manifest nhưng lại được "ký tên" với public key. Dấu ấn kết quả sẽ được trữ tập tin định nghĩa assembly manifest).

Khi một ứng dụng cho nạp assembly vào, thì CLR sẽ dùng public key để giải mã đoạn mã băm đối với các tập tin thuộc assembly bảo đảm là chúng không bị giả mạo. Việc này cũng bảo vệ chống đụng độ về tên.

Vậy tóm lại: Strong Name chứa đựng nhận dạng của Assemply, nó bao gồm tên, phiên bản và các tập tin khác cộng với khóa công khai(Public key) và chữ ký số, mỗi Strong Name được tạo ra là duy nhất.

Có nhiều cách để tạo Strong Name, chúng ta sẽ tìm hiểu cách tạo bằng tiện ích SN.EXE. Để sử dụng tiện ích này chúng ta gõ dòng lệnh sn –k filename.snk hoặc lệnh sn –k filename.reg từ cửa sổ dòng lệnh **Visual Studio 2005 Command Prompt.** Khi tạo xong ta sẽ thấy tập tin này xuất hiện trong thư mục mà ta đã tạo.

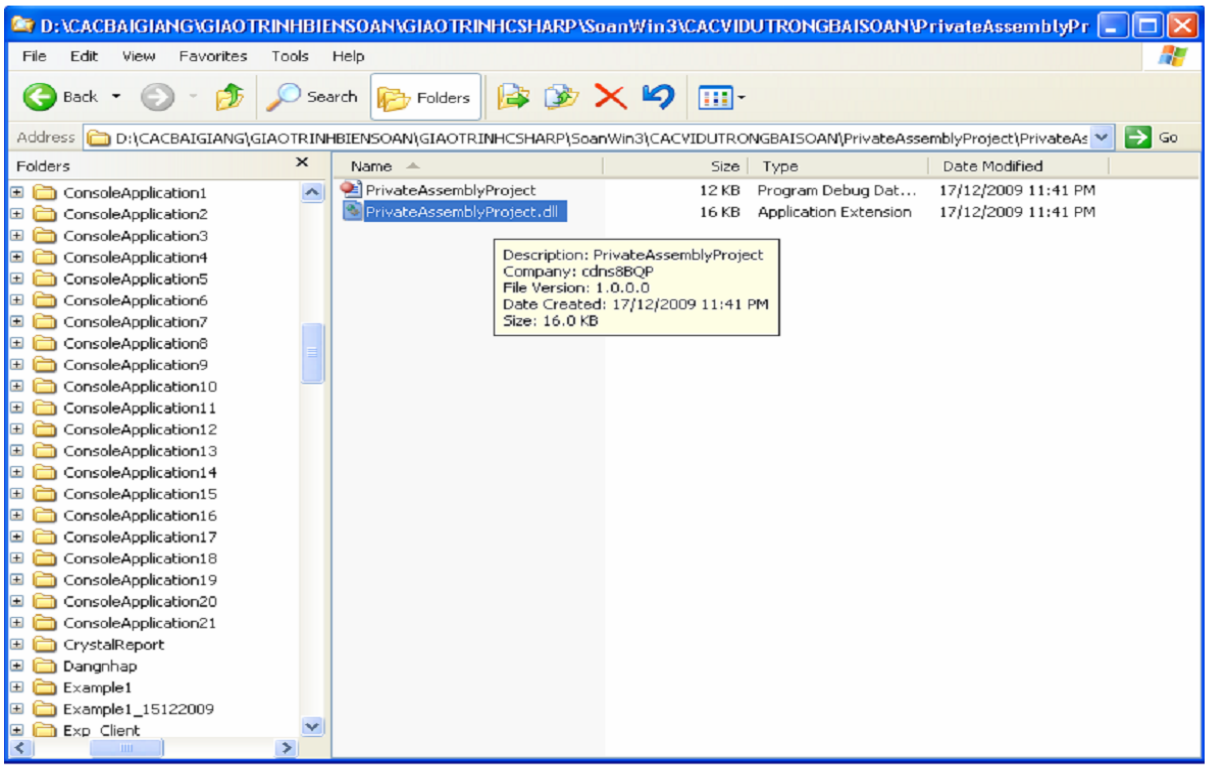


Sau đó chúng ta khai báo sử dụng tập tin này trong tập tin AssemblyInfo.cs như sau:

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("PrivateAssemblyProject")]
[assembly: AssemblyDescription("nghiemnvan@yahoo.com")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("cdns8BQP")]
[assembly: AssemblyProduct("PrivateAssemblyProject")]
[assembly: AssemblyCopyright("Copyright © nghiemnvan@yahoo.com 2009")]
[assembly: AssemblyTrademark("nghiemnvan@yahoo.com")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
```

```
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("a9560df8-9fcd-4d1f-a37d-6f447dd504f9")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: AssemblyKeyFileAttribute(@"..\\..\\PrivateAssemblyProject.snk")]
```

Khi mở thư mục chứa file dll ta sẽ thấy như trên hình:



3.3 Global Assembly Cache

Mỗi máy tính sau khi cài đặt thành công .NET FrameWork thì sẽ tạo ra vùng truy cập nhanh trong thư viện của chúng; vùng truy cập nhanh cho phép nạp danh sách các Assembly của .NET FrameWork lẫn các Assembly khác do chúng ta tạo ra.

Để tìm thấy các Assembly này, chúng ta có thể vào thư mục có tên Assembly trong thư mục Windows.

Vậy GAC trông giống như thư mục nhưng nó không phải là thư mục, nơi lưu trữ tất cả các Assembly để chia sẻ cho tất cả các ứng dụng.

Nếu chúng ta muốn chia sẻ Assembly cho nhiều ứng dụng thì ta phải cài đặt Assembly đó vào GAC.

Nói cách khác thì nếu có nhu cầu sử dụng chung thì cần cài đặt vào GAC, trong trường hợp ngược lại ta sẽ sử dụng Assembly dạng Private.

Có ba cách để cài đặt Assembly vào GAC:

+ **Cách 1:** Sử dụng ứng dụng Microsoft Windows Installer 2.0
+ **Cách 2:** Sử dụng tiện ích Global Assembly Cache(Gacuntil.exe).
+ **Cách 3:** Chọn và kéo thả vào thư mục Assembly.

Nếu sử dụng tiện ích GACUTIL.EXE thì ta dùng cú pháp sau đây:

Gacutil [option] [assemblyName.dll | assemblyPath | assemblyListFile]

Trong đó, tùy chọn option là:

+ /i: cài đặt Assembly vào GAC.
+ /u: gỡ bỏ Assembly đang tồn tại trong GAC.
+ /if: cài đặt Assembly vào GAC, trong trường hợp đã tồn tại thì ghi đè.
+ /f: cài đặt Assembly vào GAC với 2 tùy chọn là /i và /l.

Sử dụng tiện ích GACUTIL.EXE để cài đặt và gỡ bỏ vào GAC như sau:

Gautil –i filename

Gautil –u filename

Gỡ bỏ Assembly ta còn có thể bấm chuột phải vào tên file đó và chọn Uninstall.

3.4 Sử dụng Share Assembly.

Cũng tương tự như Private Assembly ta cần khai báo tham chiếu đến tập tin Assembly. Trong trường hợp này thì Visual không tạo bản copy trong thư mục của ứng dụng như là với Private Assembly.

Để thấy được Assembly xuất hiện trong .NET mỗi khi chọn Add Reference ta copy tập tin này vào thư mục C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727.

4. Friend Assembly

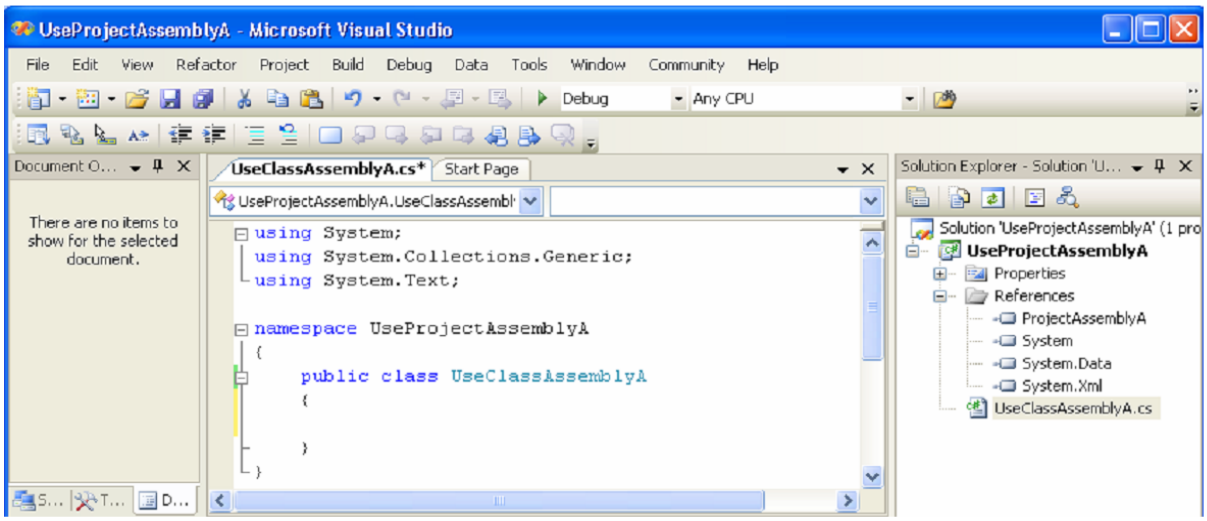
Để cho phép truy cập từ Assembly này đến Assembly khác thì tầm vực của lớp và thành viên của chúng được khai báo tầm vực là public.

Ví dụ: Ta tạo mới Project sau đây đặt tên là **ProjectAssemblyA** và lớp ClassAssembly với tầm vực là internal(mặc định) như sau đây:

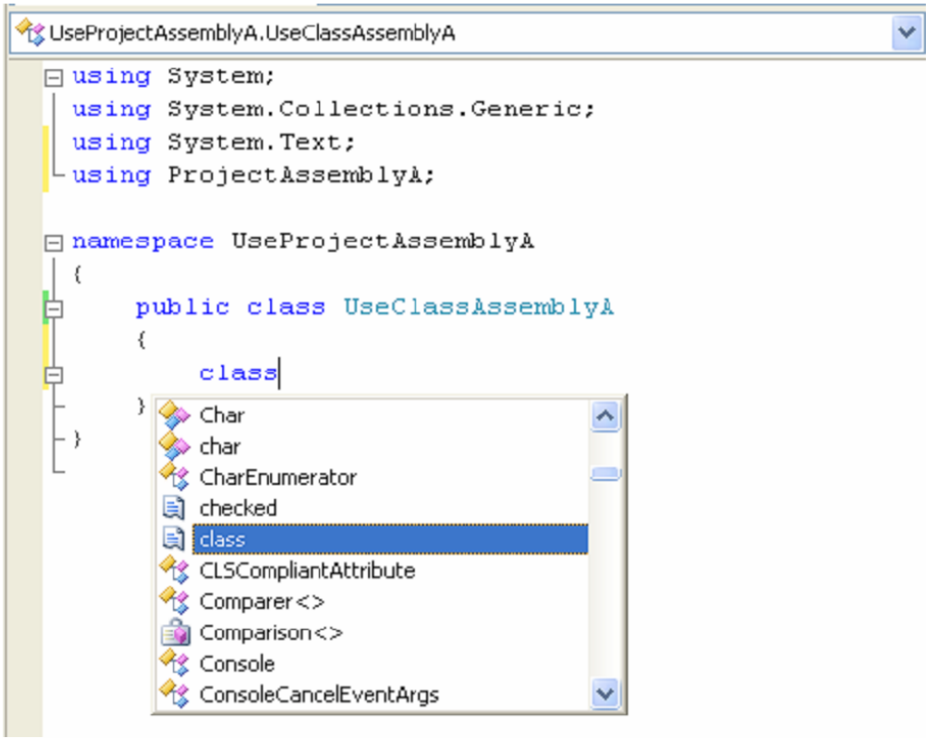
```
using System;
using System.Collections.Generic;
using System.Text;
namespace ProjectAssemblyA
{
  internal class ClassAssembly
  {
    public string StringA()
    {
      return "This is String";
    }
  }
```

}

Tiếp theo ta tạo Project mới với tên UseProjectAssemblyA và tham chiếu vào như hình vẽ sau:



Ta không thể truy cập đến lớp **ClassAssembly** và các thành viên của nó do tầm vực là internal.



Theo định nghĩa thì Friend Assembly bao gồm lớp và các thành viên của nó đều có tầm vực là internal nhưng vẫn có thể truy cập từ Assembly khác.

Để làm điều này ta khai báo thuộc tính InternalsVisibleTo ứng với Assembly khác được chỉ định cho phép thấy Assembly này khi truy cập như cú pháp sau đây:

[assembly : InternalsVisibleTo(”....”)]

Thuộc tính này thuộc không gian tên CompilerServices, chính vì vậy chúng ta phải khai báo sử dụng không gian tên này trong Project.

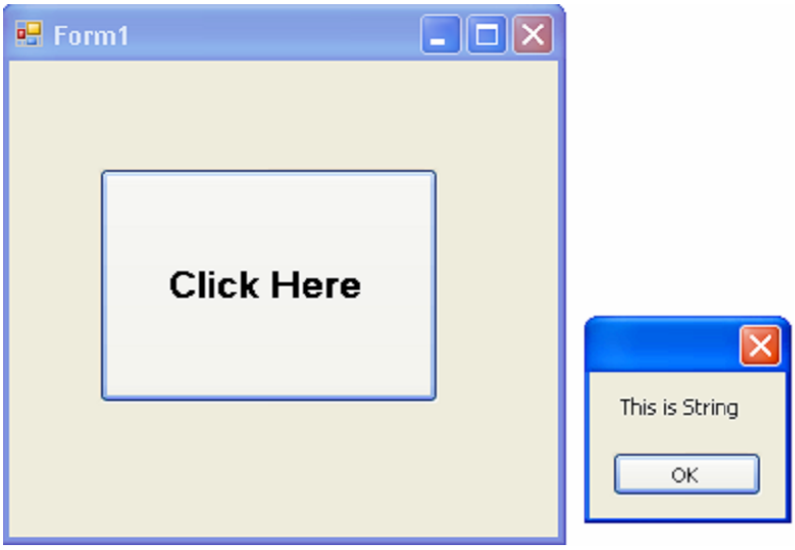
Vậy bây giờ ta sẽ làm ví dụ với vấn đề này như sau:

Tạo Project có tên ProjectAssemblyA và khai báo như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.CompilerServices;
[assembly: InternalsVisibleTo("UseProjectAssemblyA")]
namespace ProjectAssemblyA
{
   internal class ClassAssembly
   {
      public string StringA()
      {
         return "This is String";
      }
   }
}
```

Tiếp theo ta tạo một Project khác và đặt tên là UseProjectAssemblyA, tạo Form như hình sau:

Khi bấm vào nút **ClickHere** ta sẽ được câu thông báo như sau:



Ta tiến hành viết code như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ProjectAssemblyA;
namespace UseProjectAssemblyA
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void btnClickHere_Click(object sender, EventArgs e)
      {
         ClassAssembly a = new ClassAssembly();
         string strString;
         strString = a.StringA();
         MessageBox.Show(strString);
      }
   }
}
```

LẬP TRÌNH C TRÊN WIN

Lương Văn Vân - Khoa CNTT

Programming Language

Ngôn ngữ lập trình?

- Ngôn ngữ lập trình là hệ thống hữu hạn các ký hiệu, quy ước về ngữ pháp dùng để xây dựng các chương trình.
- Hướng dẫn máy tính giải một bài toán hay một yêu cầu đặt ra.

MỘT SỐ TIÊU ĐIỂM CỦA MÔN HỌC

- **Tổng số tiết: 90 tiết**
  - Lý thuyết: 25 tiết
  - Bài tập: 5 tiết
  - Thực hành: 60 tiết
- **Sinh viên cần phải được học trước các môn**
  - Kỹ thuật lập trình cơ bản.
  - Lập trình C.
  - Kỹ thuật lập trình hướng đối tượng.
- **Thi kết thúc môn học bằng hình thức thi thực hành.**

NỘI DUNG

C1. Môi trường lập trình trong Windows

C2. Thực đơn, thanh công cụ và thanh trạng thái

C3. Đồ họa và xử lý các thông điệp đầu vào

C4. Hộp hội thoại và các điều khiển

TÀI LIỆU THAM KHẢO

"Bài giảng lập trình C trên Windows"
Lương Văn Vân - Khoa CNTT

"Lập trình C trên Windows"
Đặng Văn Đức, NXB Khoa học kỹ thuật

"Lập trình Windows bằng Visual C++"
Đặng Văn Đức - Lê Quốc Hưng, Giáo Dục

"http://www.codeproject.com"
"http://www.codeguru.com"

MFC

Mọi thắc mắc xin liên hệ

0983.890088

luongvandlu@gmail

Văn phòng khoa

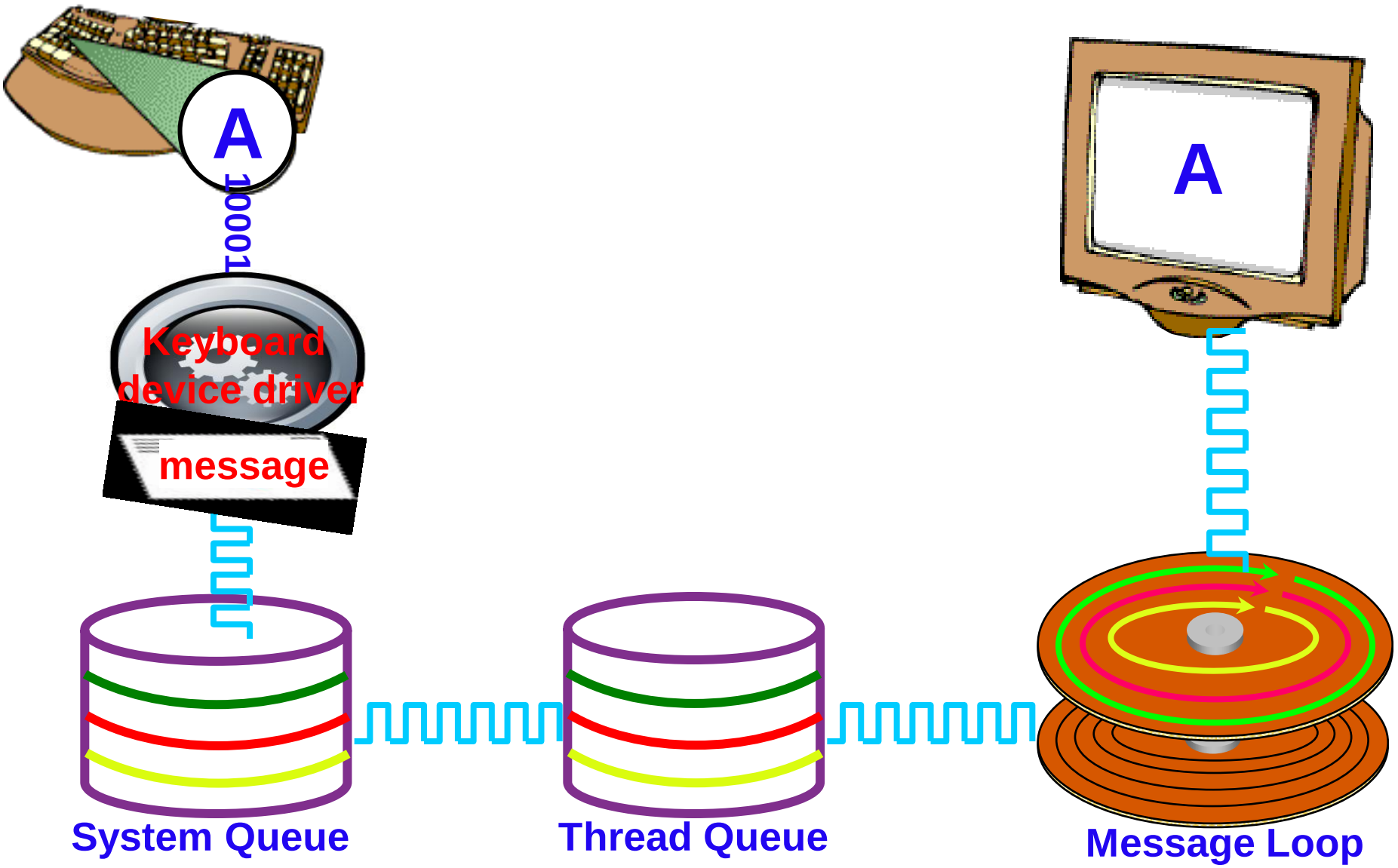
Môi trường lập trình trong Windows

- Đặc điểm của môi trường windows
- Vòng lặp thông điệp
- Giới thiệu MFC
- Tạo chương trình bằng AppWizard
- Phân tích các tập tin của một ứng dụng

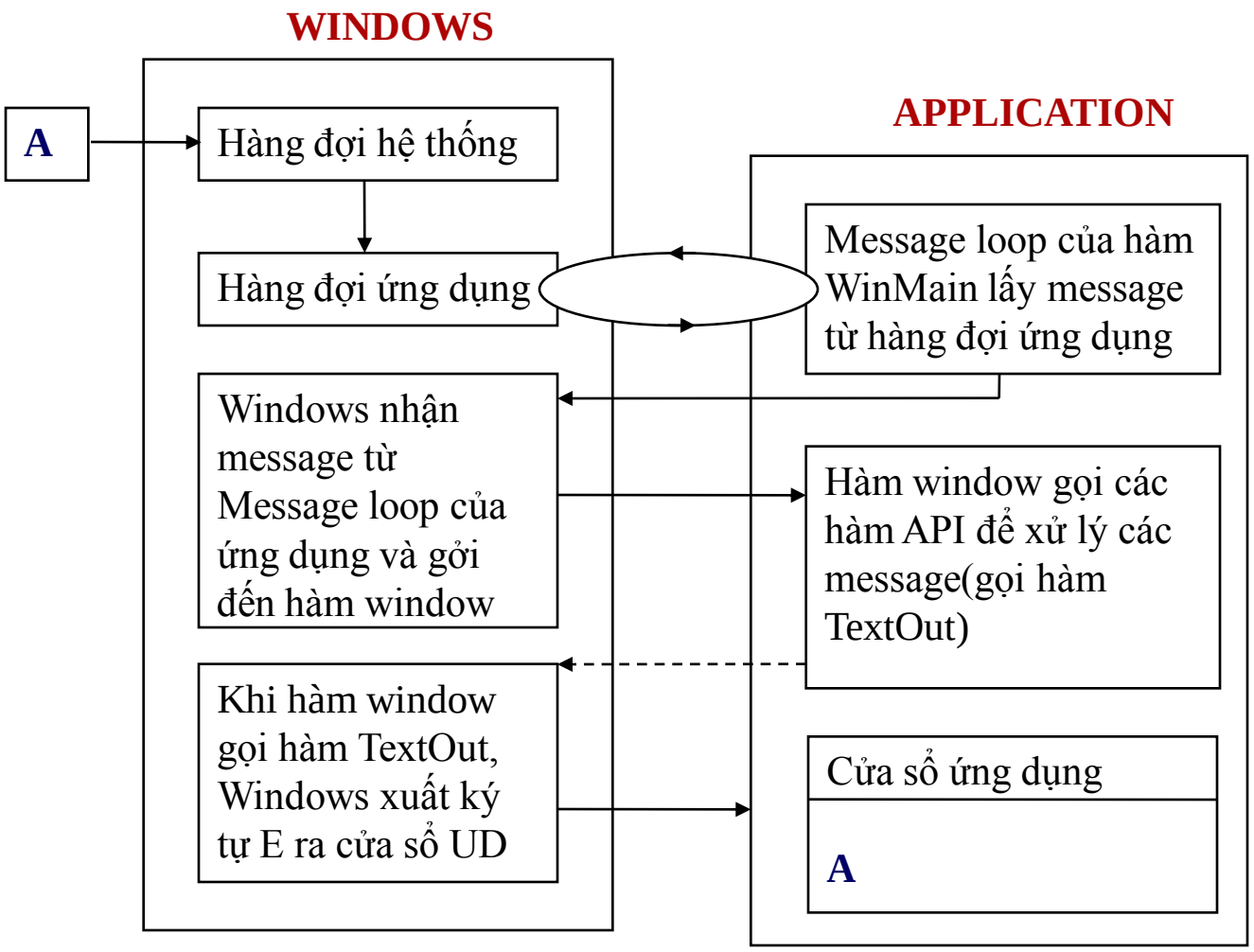
Đặc điểm của môi trường windows

1. Giao diện người dùng kiểu đồ họa (GUI)
2. Đa nhiệm
3. Quản lý bộ nhớ
4. Tư tưởng hướng đối tượng
5. Giao diện đồ họa độc lập với thiết bị
6. Kiến trúc hướng thông điệp
7. Thủ tục cửa sổ
8. Tài nguyện

Vòng lặp thông điệp



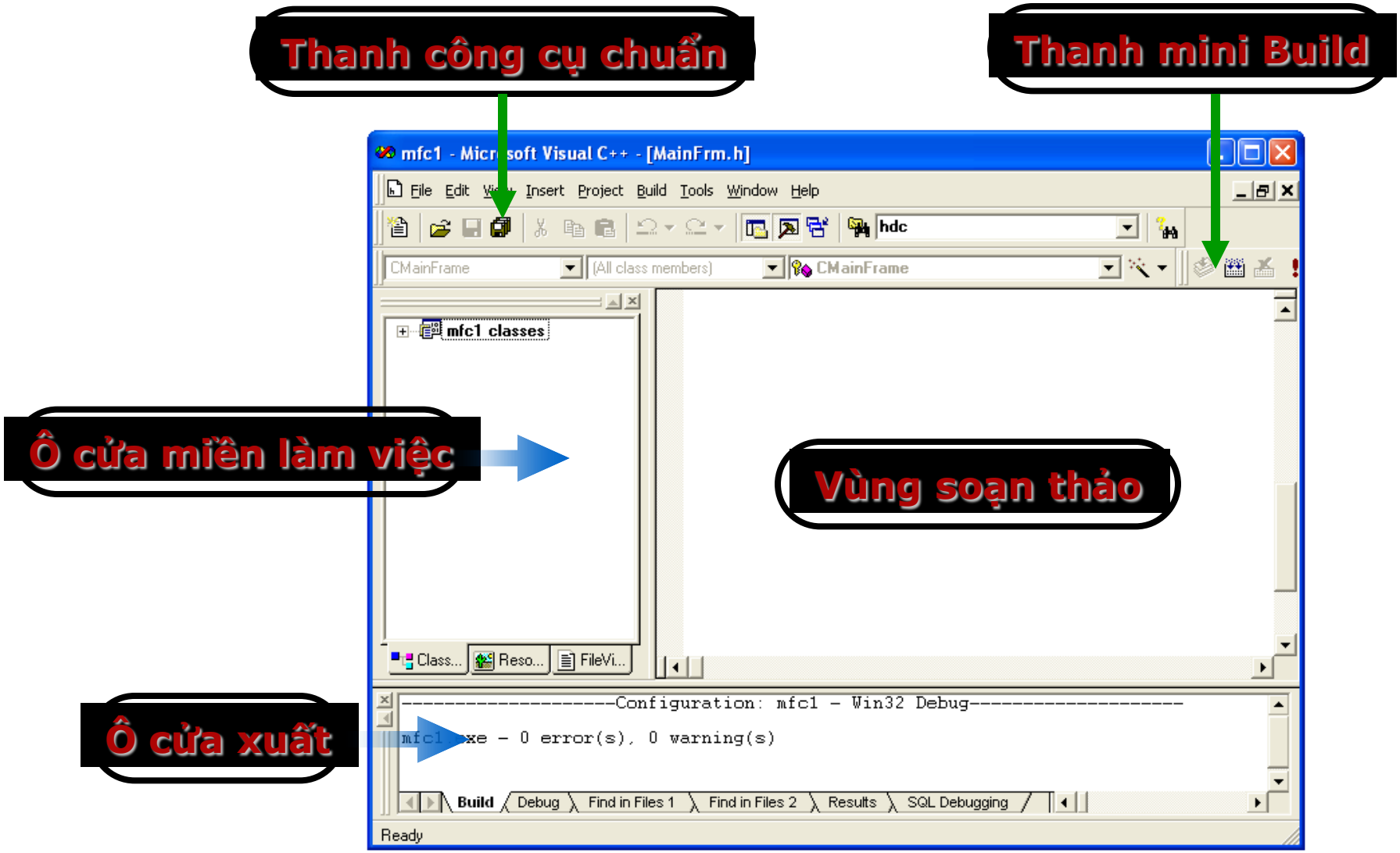
Cơ chế hoạt động của thông điệp



Giới thiệu MFC

- **Đặc điểm lập trình MFC**
  - **Microsoft Foundation Class**: Tập hợp các lớp định nghĩa sẵn.
  - Biểu diễn cách tiếp cận hướng đối tượng đến lập trình **Windows** và gói các **Windows API.**
  - Cho phép **Lập trình viên** ít phải lo lắng về giao diện **Windows.**
  - Làm đơn giản tiến trình phát triển mã trình cho các loại máy có hệ điều hành khác nhau.
  - Có hơn 130 lớp.
  - ……

Môi trường phát triển Visual C++

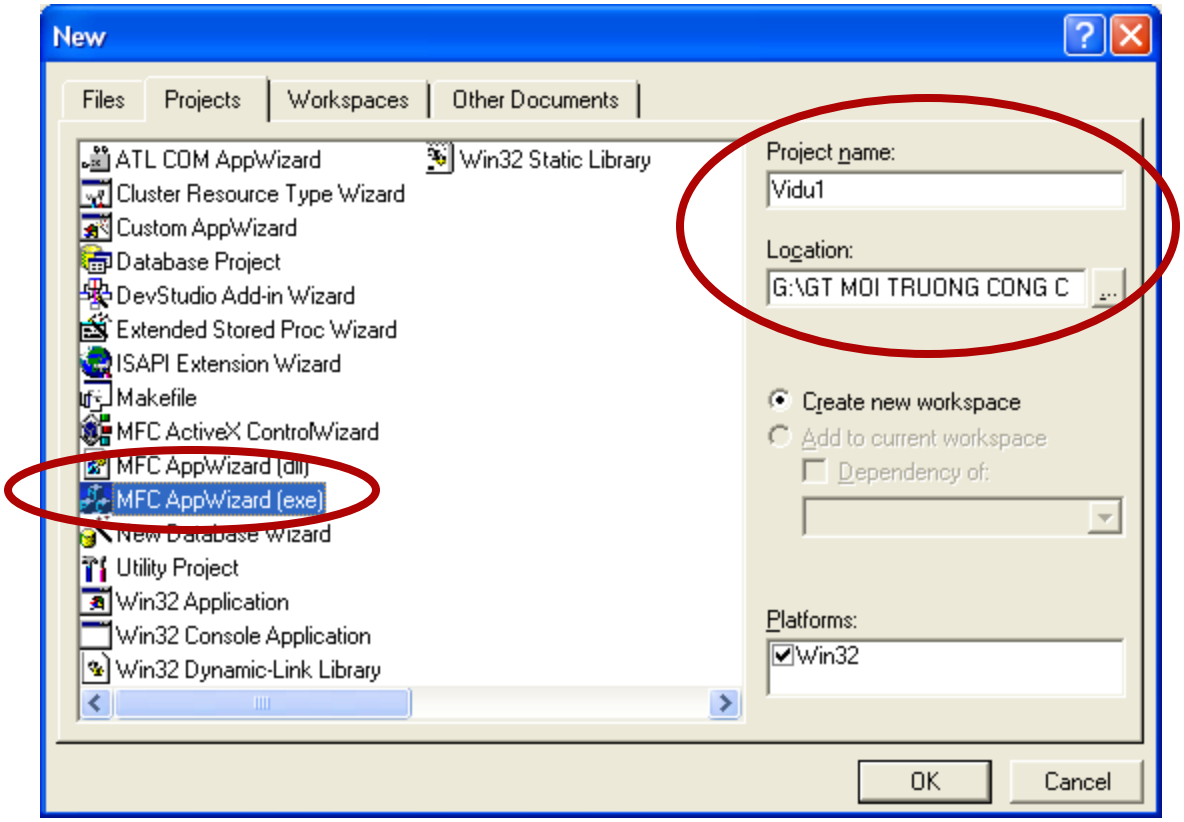


Môi trường phát triển Visual C++

- **Miền làm việc:** chứa các thành phần của ứng dụng:
  - Class View: điều hành và thao tác mã nguồn trên mức lớp.
  - Resource View: tìm và chọn lọc tài nguyên của ứng dụng.
  - File View: xem và điều hành tất cả các file.
- **Ô cửa sổ:** thông báo lỗi và lời cảnh báo của trình biên dịch.
- **Vùng soạn thảo:** nơi các cửa sổ soạn thảo mã hiển thị khi bạn soạn thảo mã nguồn.
- **Thanh mini Build:** lệnh xây dựng và chạy.

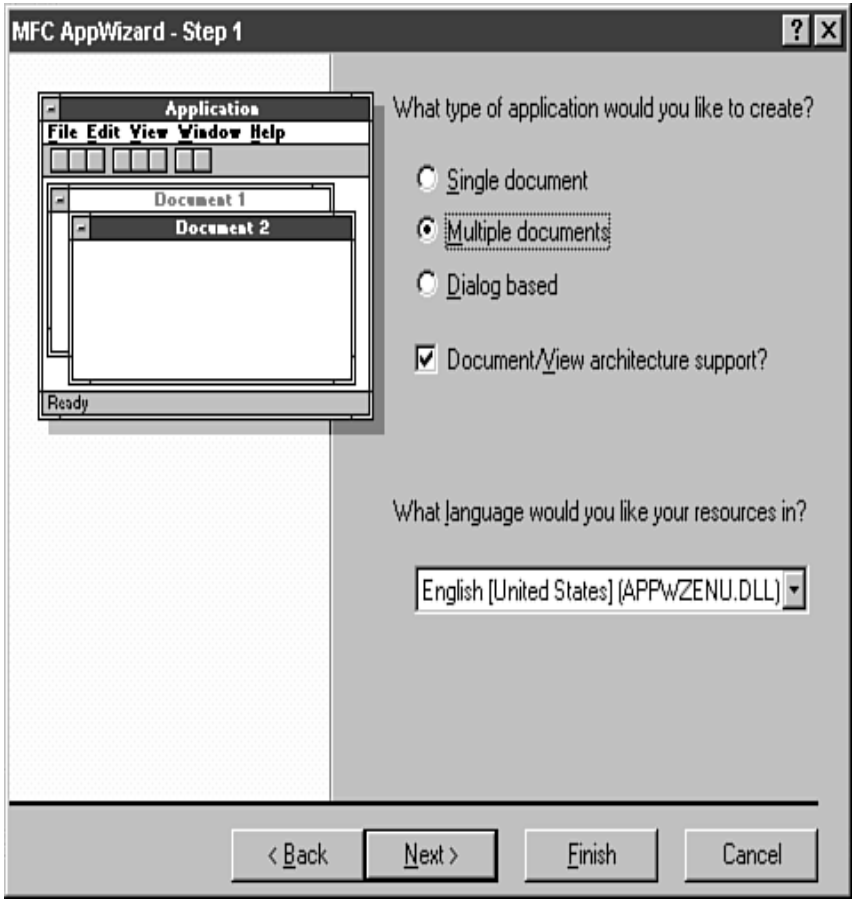
Tạo chương trình bằng AppWizard

- **Microsoft Visual C++ chọn menu File - New**
  - **MFC AppWizard (exe)**
  - **Project Name: điền tên dự án vào**
  - **Location: nơi chứa dự án.**



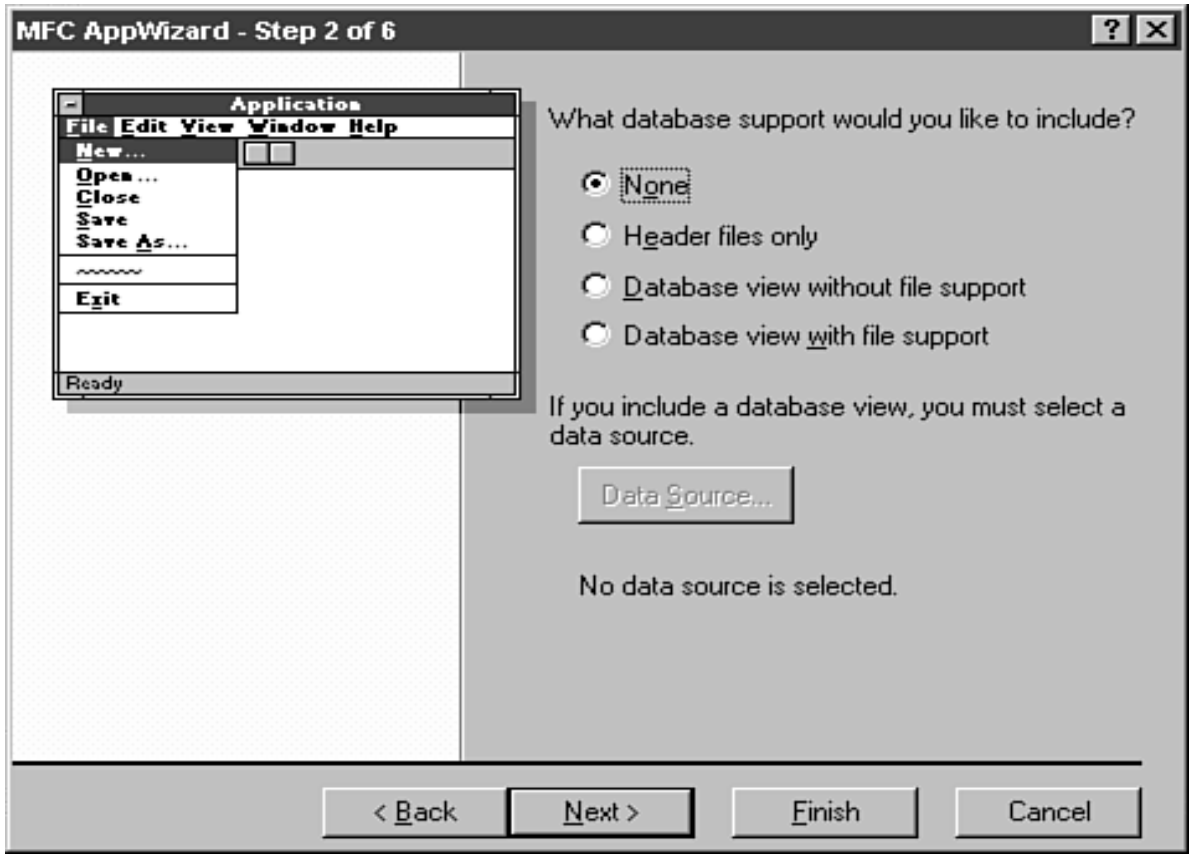
Tạo chương trình bằng AppWizard

- **Bước 1: Có 3 kiểu ứng dụng được chọn tại bước này:**
  - **Single document: một tài liệu trong một lúc.**
  - **Multiple document: nhiều tài liệu cùng một lúc để làm việc.**
  - **Dialog based: hộp hội thoại làm giao diện người sử dụng.**



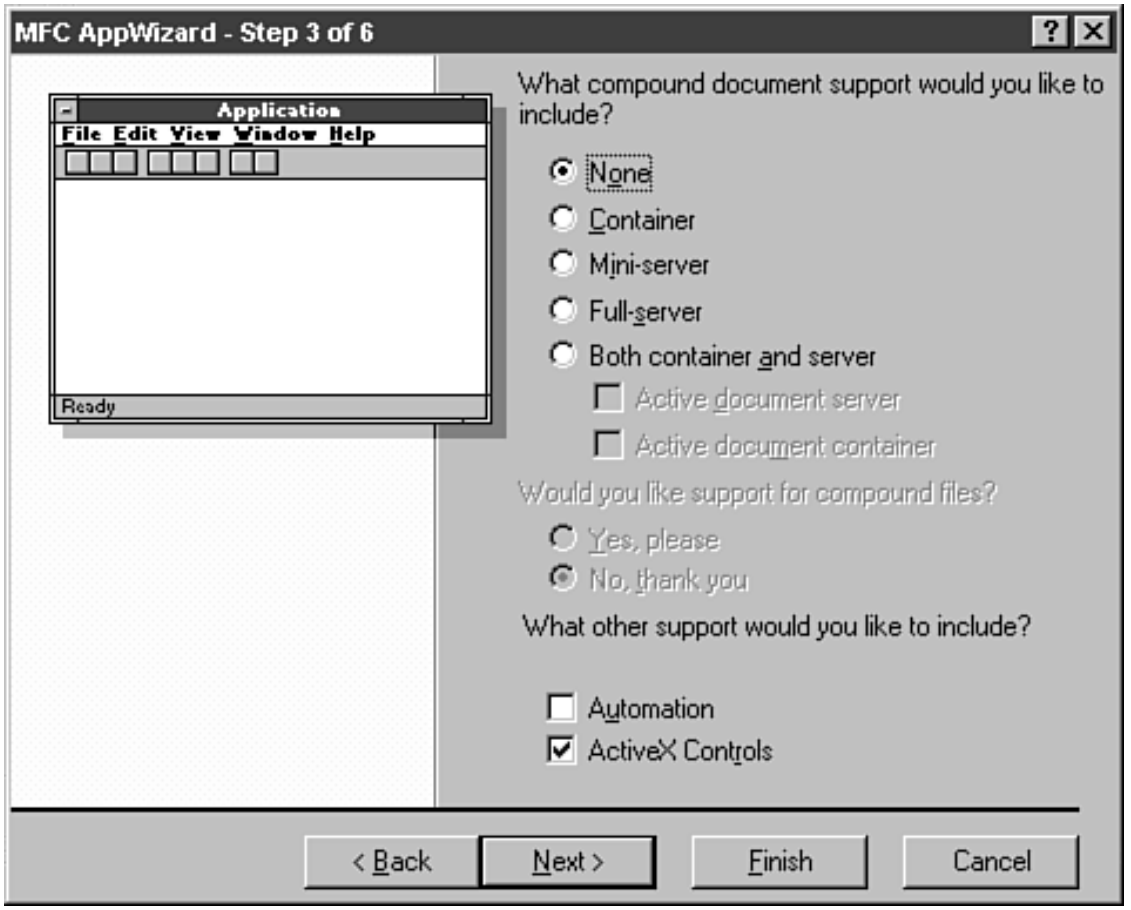
Tạo chương trình bằng AppWizard

- **Bước 2**: cho phép chọn mức hỗ trợ cơ sơ dữ liệu:



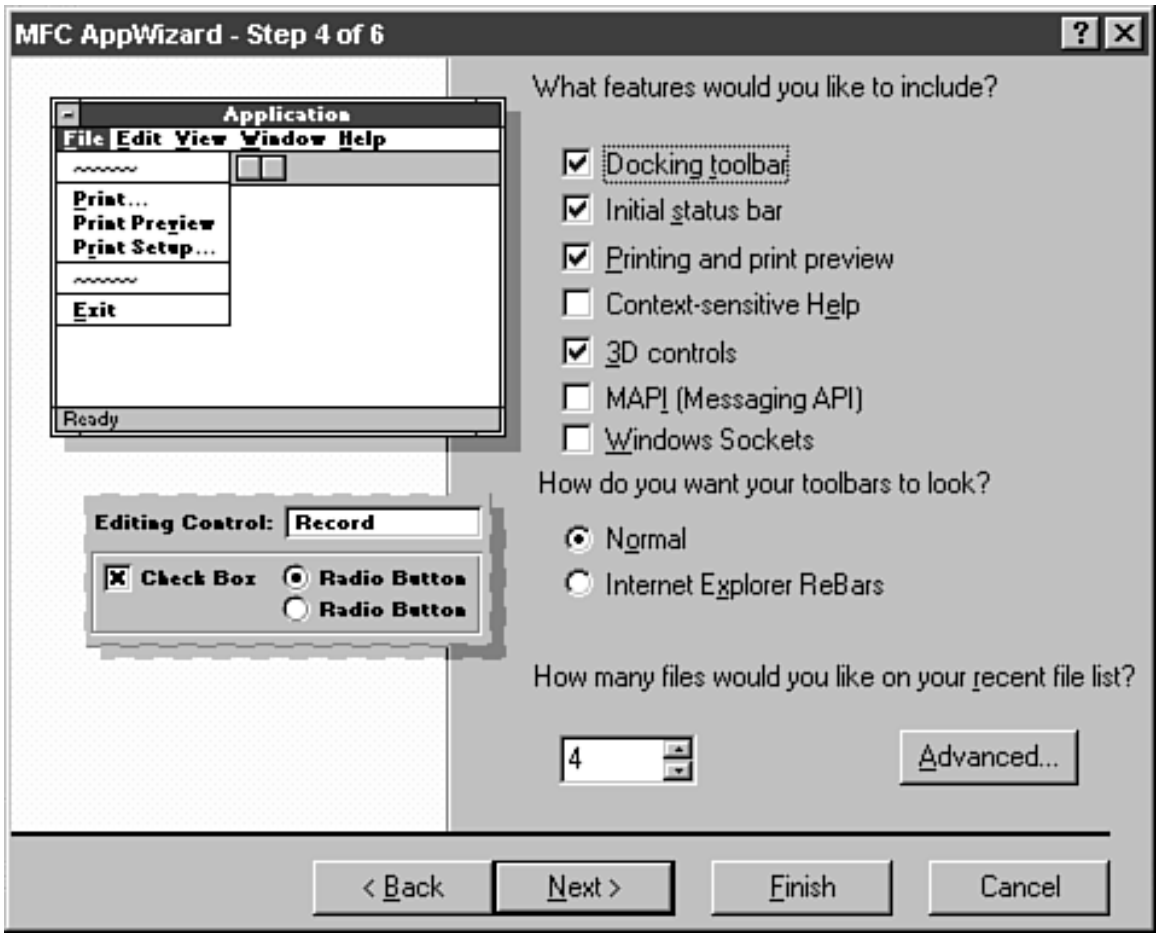
Tạo chương trình bằng AppWizard

- **Bước 3**: chọn tài liệu đa hợp hỗ trợ mà chương trình cần:



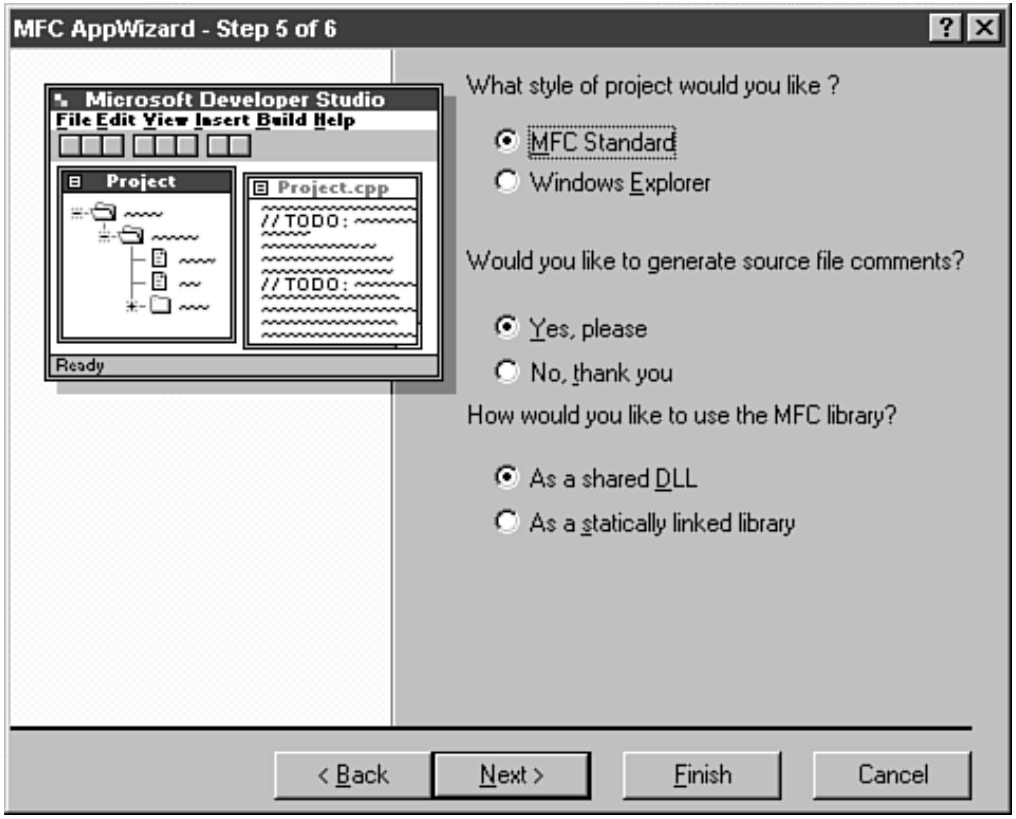
Tạo chương trình bằng AppWizard

❖ **Bước 4**: Chọn một số đặc tính cho giao diện.



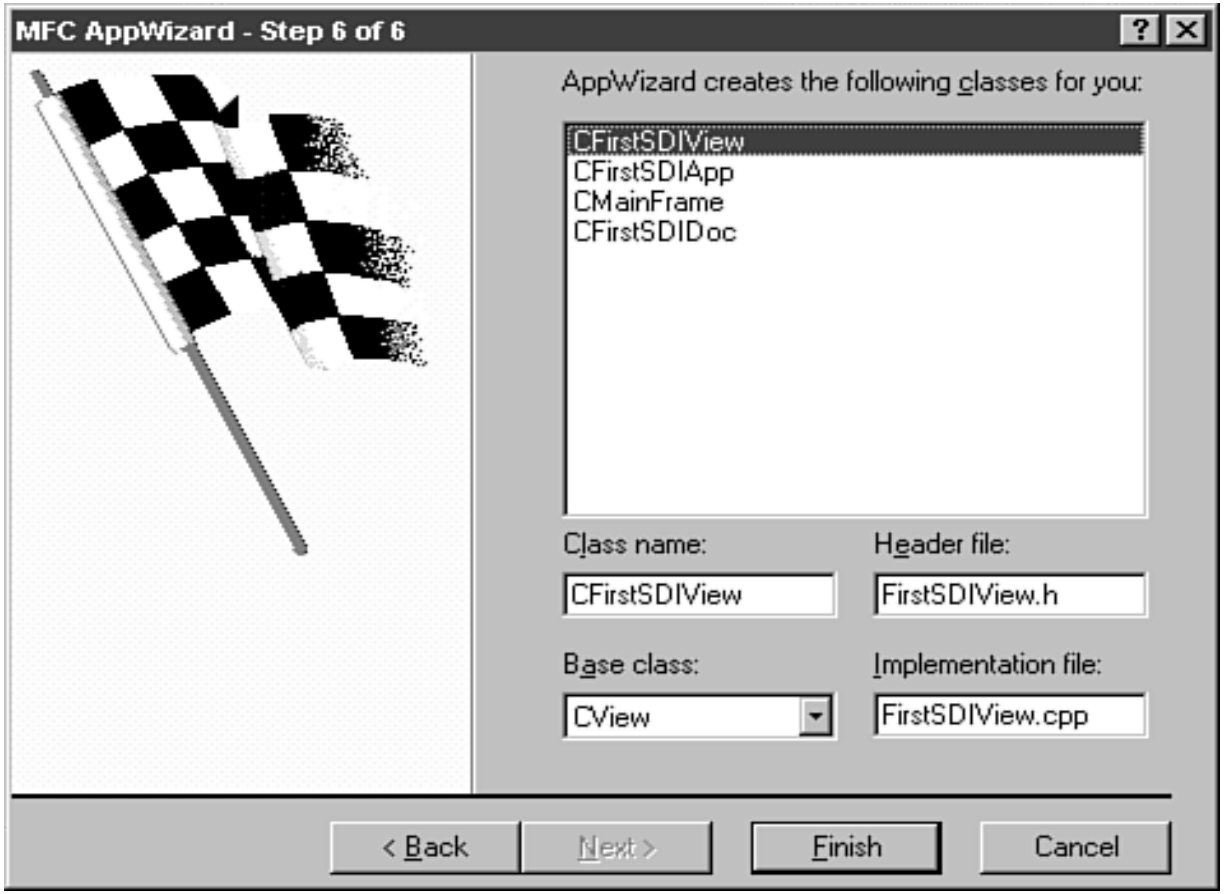
Tạo chương trình bằng AppWizard

- **Bước 5**: Chọn những ghi chú và những thư viện MFC trong chương trình.



Tạo chương trình bằng AppWizard

❖ **Bước 6: Chứng thực filenames và classname.**



Phân tích các tập tin của một ứng dụng

- **Ứng dụng Single Document**
  - Đối tượng dẫn xuất từ **4** lớp đối tượng cơ sở **Application classes**, **Document classes**, **ViewClasses** và **Frames classes** .
  - Ví dụ:

Lớp cơ sở	Lớp dẫn xuất	Tên tập tin
CWinApp	Cvd1App	Vd1.cpp
CDocument	Cvd1Doc	Vidu1Doc.cpp
CView	Cvd1View	Vidu1View.cpp
CFrameWnd	CmainFrame	MainFrm.cpp
CMDIChildWnd	CchildFrame	ChildFrm.cpp

Phân tích các tập tin của một ứng dụng

- **Tập tin RESOURCE.H:** chứa toàn bộ những lệnh **#define** dùng định nghĩa chỉ danh ID những tài nguyên như là:
  - **DD_ABOUTBOX**: Đối với tài nguyên là một hộp hội thoại about box
  - **IDR_MAINFRAME**: Đối với việc chia sẻ sử dụng bởi nhiều loại tài nguyên.
  - **IDR_VIDU1TYPE**: Đối với tài nguyên là biểu tượng tài liệu

Phân tích các tập tin của một ứng dụng

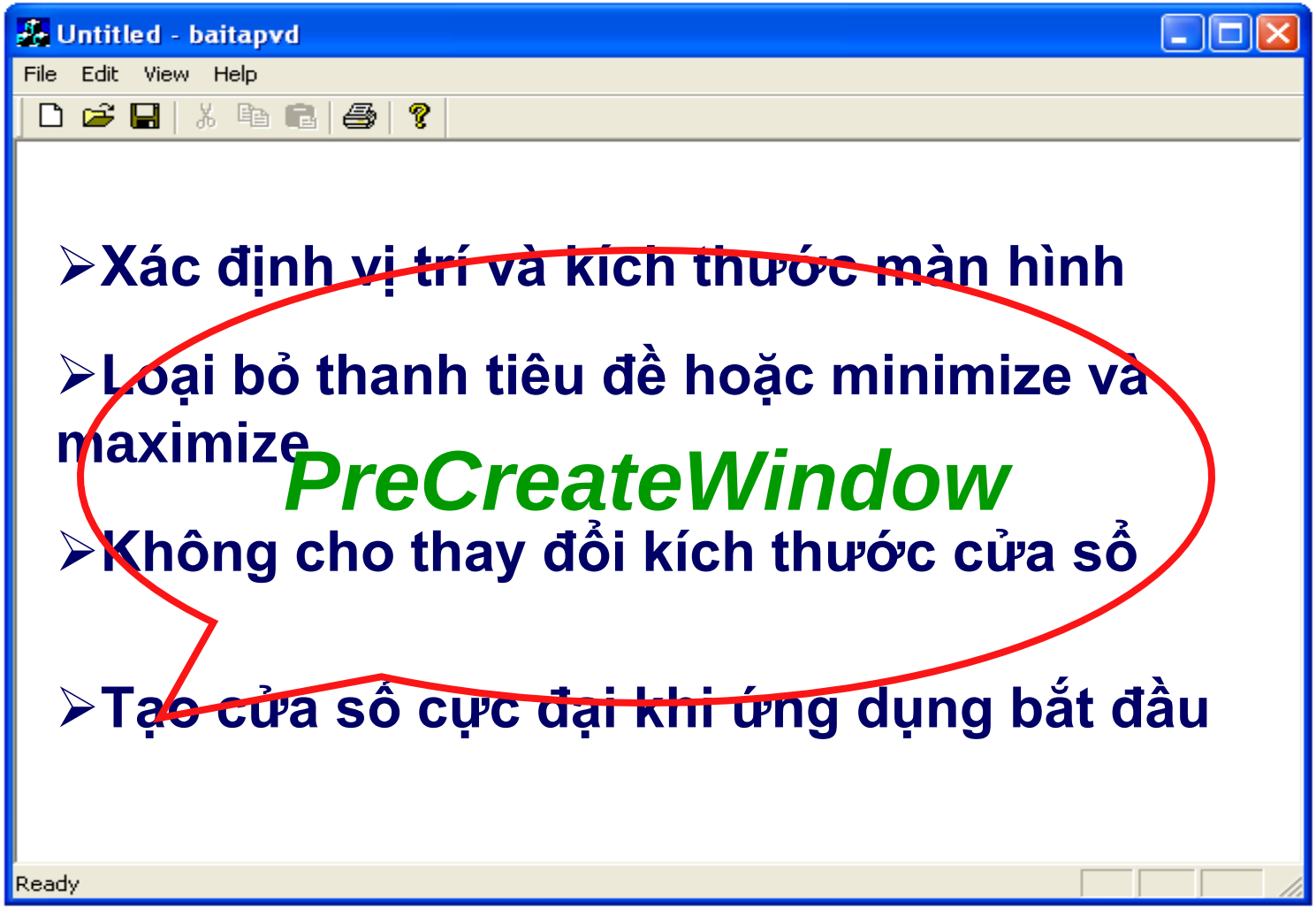
- **Tập tin Vidu1.H:** chứa các khai báo liên quan đến lớp Cvidu1App:
- **Tập tin MainFrm.H**: chứa các khai báo liên quan đến lớp khung cửa sổ chính (main frame) của ứng dụng.
- **Tập tin Vidu1Doc.H:** chứa các khai báo liên quan đến tài liệu.
- **Tập tin Vidu1.RC:** chứa các tài nguyên liên quan đến menu, phím nóng, hộp hội thoại,…
- **Tập tin Vidu1.CPP:** định nghĩa các hàm đã khai báo trong lớp CVidu1App, là lớp chương trình chính.

Thực đơn, thanh công cụ và thanh trạng thái

- Màn hình ứng dụng
- Thực đơn (Menu)
- Thanh công cụ (ToolBar)
- Thanh trạng thái (Status bar)

Thực đơn, thanh công cụ và thanh trạng thái

❖ **Màn hình ứng dụng:**



Xác định vị trí và kích thước màn hình

- **Để đặt ứng dụng giữa màn hình và chiếm 90% màn hình:**

```
int xSize= GetSystemMetrics(SM_CXSCREEN);
int ySize= GetSystemMetrics(SM_CYSCREEN);
cs.cx = xSize *9/10;
cs.cy = ySize *9/10;
cs.x   = (xSize – cs.cx )/2;
cs.y   = (ySize – cs.cy )/2;
```

- **Trong đó: cs là biến cấu trúc *CREATESTRUCT***

Màn hình ứng dụng

- **Loại bỏ minimize và maximize:**

  cs.style &= ~(WS_MAXIMIZEBOX | WS_MINIMIZEBOX);

- **Không cho thay đổi kích thước cửa sổ:**

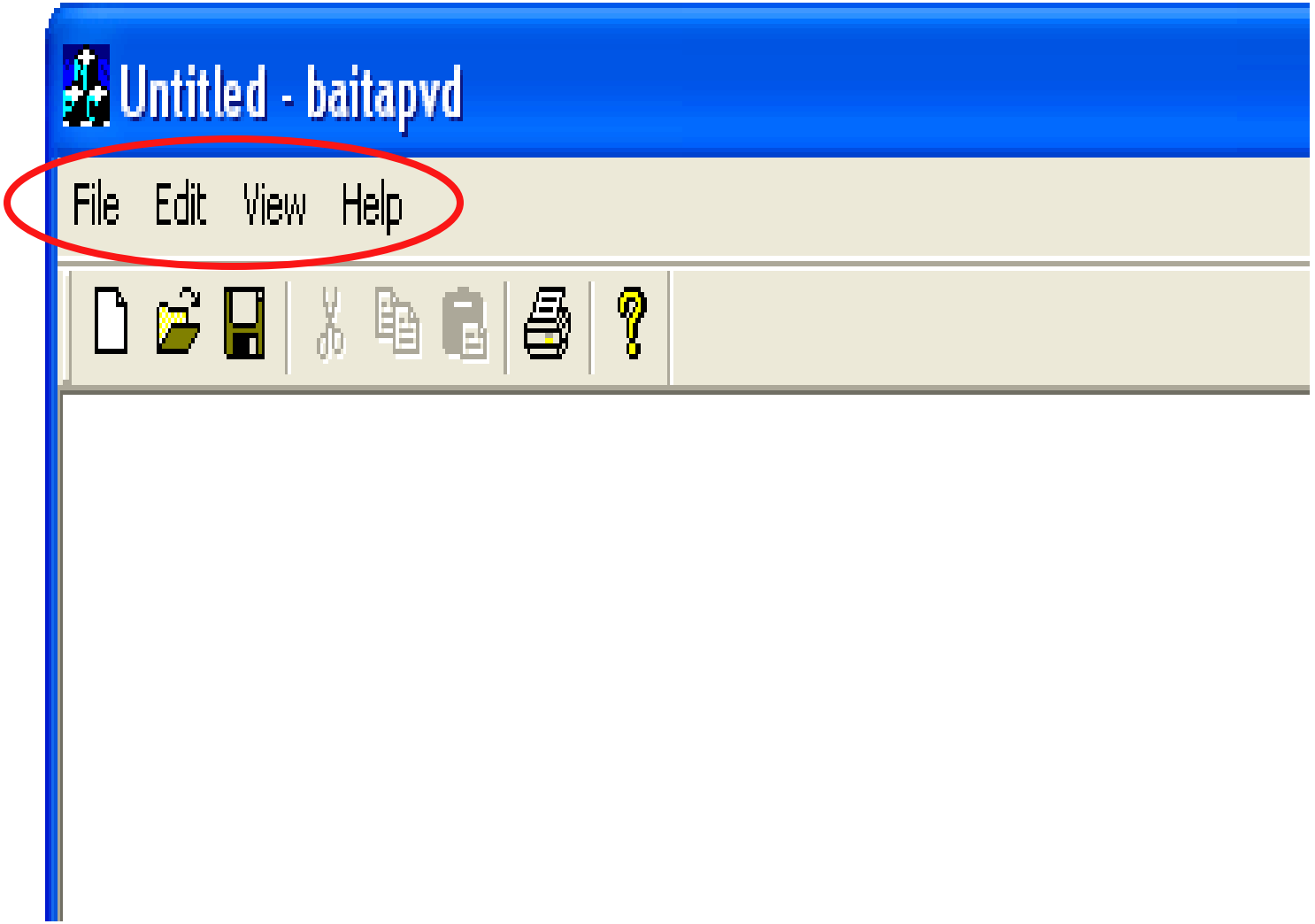
  cs.style &= ~WS_THICKFRAME;

- **Tạo cửa sổ cực đại khi ứng dụng bắt đầu:**

  Trong hàm **ShowWindow()** thay đổi cờ m_nCmdShow thành cờ SW_SHOWMAXIMIZED:

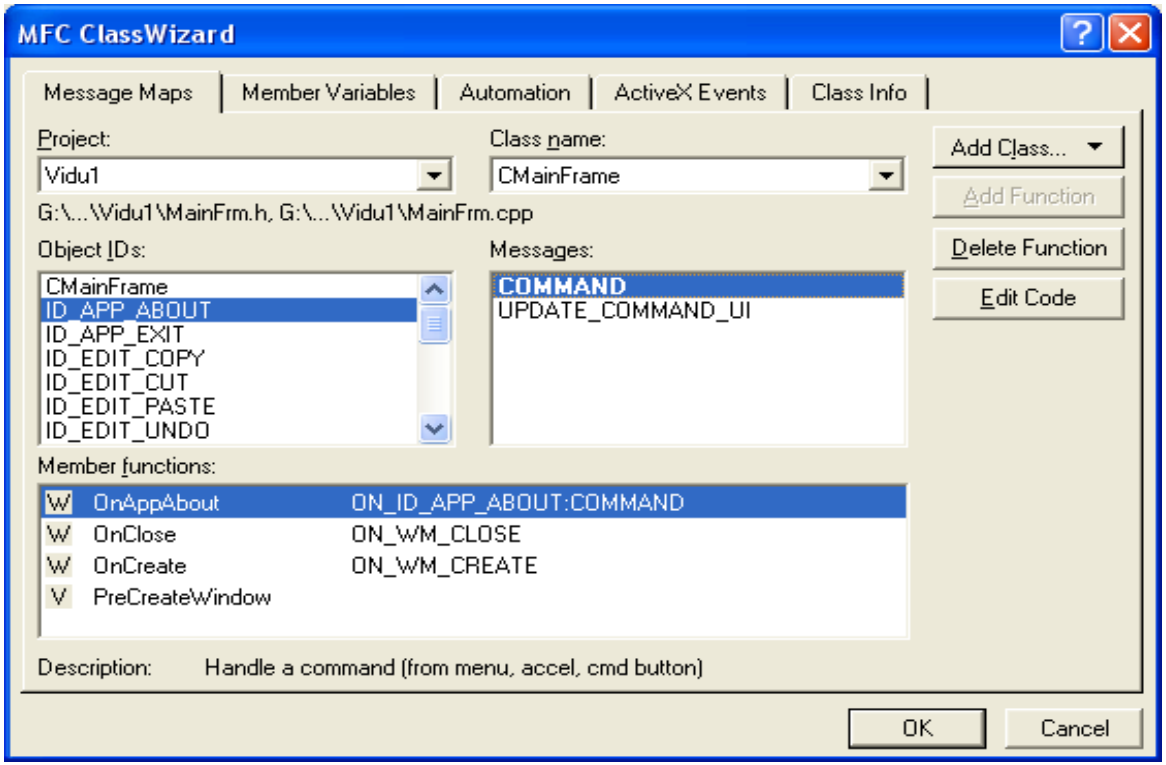
  m_pMainWnd-> ShowWindow(SW_SHOWMAXIMIZED);

Thực đơn (Menu)



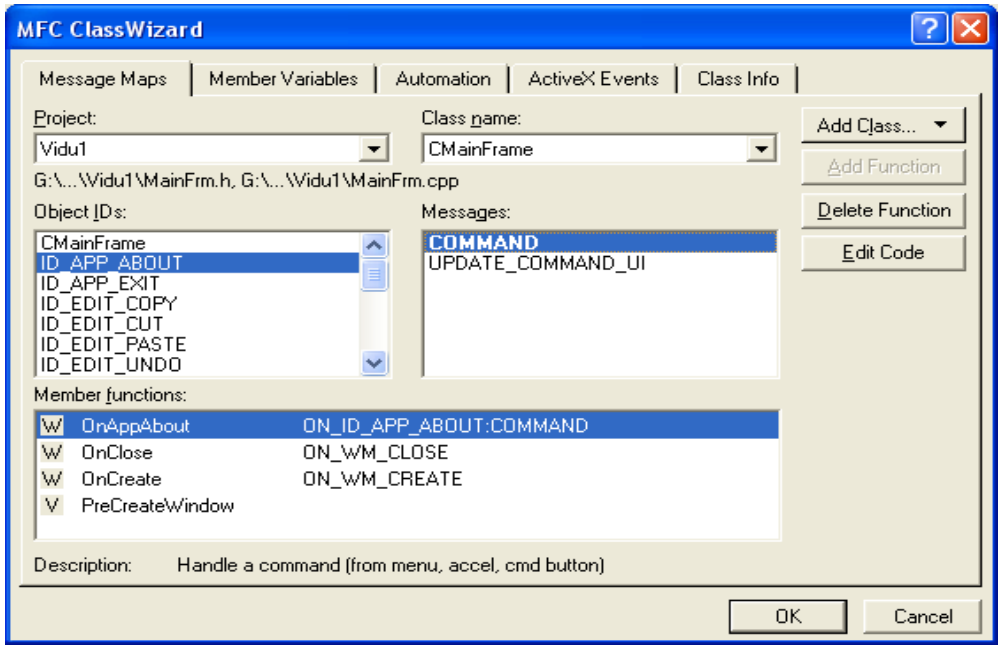
Thực đơn (Menu)

- **Sử dụng ClassWizard để tự động thêm một bộ phận điều khiển lệnh menu:** là một hàm thành viên của lớp. Hàm thành viên này được thi hành khi người sử dụng kích hoạt vào menu.



Các bước tạo

- **Chọn View/ClassWizard – Hộp hội thoại hiện ra và chọn lớp muốn xử lý lệnh menu:**



- **Chọn ID mục menu cần xử lý trong khung Object Ids và chọn COMMAND trong khung Message.**
- **ClassWizard tự tạo hàm xử lý trống trong khung Member function.**
- **Click vào nút Edit Code để chuyển đến hàm xử lý. Từ đây thêm vào đoạn mã xử lý cho lệnh menu.**

Thực đơn (Menu)

- **Ví dụ: Tạo một mục menu, với yêu cầu click vào mục menu đó thi sẽ xuất ra một thông báo.**

```
void CMainFrame::Tên_Hàm_Xử_Lý_Menu()
{
  // TODO: Add your command handler code here
    AfxMessageBox(“Thông báo", MB_YESNO);
}
```

Thực đơn (Menu)

❖ Các lệnh menu có hiệu lực và không có hiệu lực:

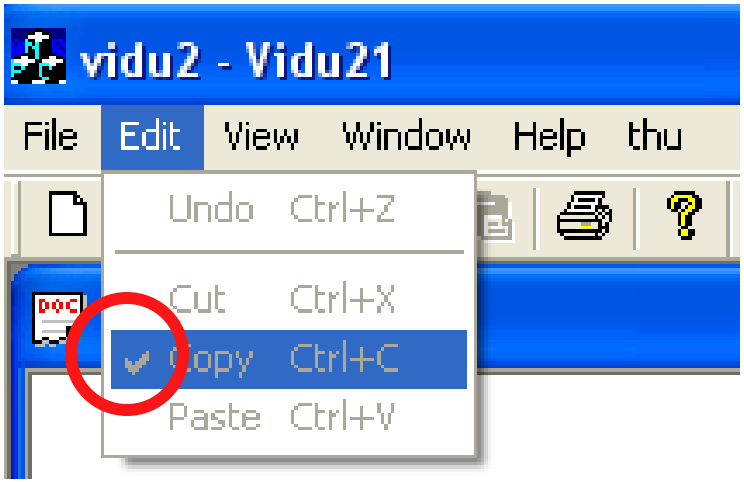
- **Khi sử dụng ClassWizard để thêm mới một mục menu thay vì chọn COMMAND trong khung Messages thì chọn UPDATE_COMMAND_UI và trong hàm xử thông điệp bổ sung câu lệnh:**

```
pCmdUI->Enable(m_bWzd);
```

- **Nếu m_bWzd =TRUE thì menu trở nên hoạt động,**
- **Nếu m_bWzd = FALSE có kết quả ngược lại.**

Thực đơn (Menu)

❖ Đánh dấu kiểm tra các lệnh menu:



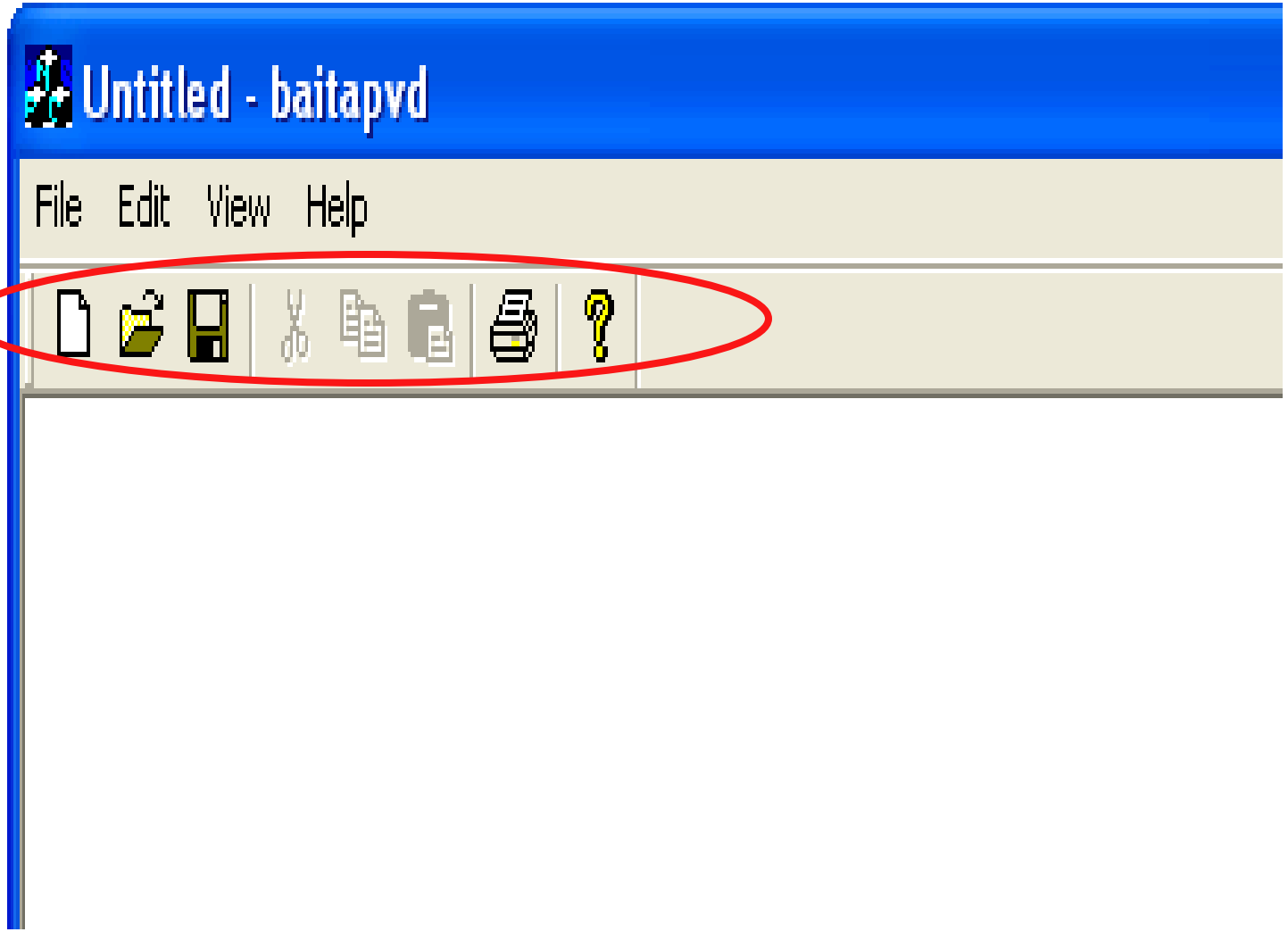
- **Thực hiện bằng ClassWizard thêm hàm xử lý chọn UPDATE_COMMAND_UI và viết lệnh xử lý:**

  pCmdUI->SetCheck(m_bWzd);

  m_bWzd = TRUE có dấu kiểm tra

  m_bWzd = FALSE bỏ dấu kiểm tra.

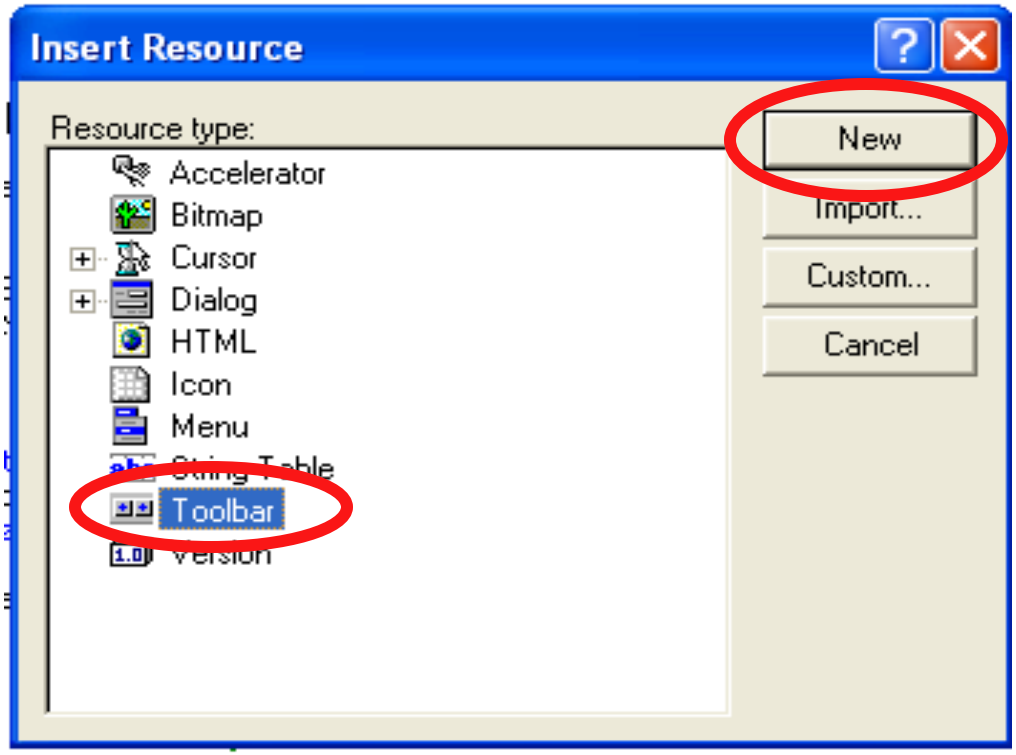
Thanh công cụ (ToolBar)



Thanh công cụ (ToolBar)

❖ Thêm thanh công cụ mới vào ứng dụng:

- **Bước 1: Soạn thảo (tạo) thanh công cụ**
  - **Click vào menu Insert/Resource, chọn toolbar từ danh sách và click vào new**



Thanh công cụ (ToolBar)

- **Thêm thanh công cụ mới vào ứng dụng:**
  - **Bước 2: khai báo đối tượng ToolBar thuộc lớp CToolBar trong thành phần protected của lớp CframeWnd.**

Cú pháp:

```
CToolBar  Tên_đối_tượng_ToolBar;
```

Thanh công cụ (ToolBar)

- **Thêm thanh công cụ mới vào ứng dụng:**
  - **Bước 3: Viết mã lệnh trong hàm Create() của lớp CMainFrame .**

```
if (!toolbar.Create(this) || !toolbar.LoadToolBar(ID_TOOLBAR))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }
    toolbar.SetBarStyle(oolbar.GetBarStyle() | CBRS_TOOLTIPS |
    CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
    oolbar.EnableDocking(CBRS_ALIGN_ANY);
```

Thanh công cụ (ToolBar)

- **Các thanh công cụ thêm vào sẽ được sắp xếp theo hàng:**

  DockControlBar(CControlBar *pBar, UINT nDockBarID= 0);

  pBar: con trỏ đối tượng toolbar

  nDockBarID: gtrị xác định vị trí thanh công cụ

  - **AFX_IDW_DOCKBAR_TOP : Toolbar xếp vào phía trên vùng client cửa sổ**
  - **AFX_IDW_DOCKBAR_BOTTOM : Toolbar xếp vào phía dưới vùng client cửa sổ**
  - **AFX_IDW_DOCKBAR_LEFT : Toolbar xếp vào phía bên trái vùng client cửa sổ**
  - **AFX_IDW_DOCKBAR_RIGHT : Toolbar xếp vào phía bên phải vùng client cửa sổ**

Thanh công cụ (ToolBar)

- **Làm có hiệu lực, mất hiệu lực các nút thanh công cụ :**
  - **Sử dụng ClassWizard , chọn ID của nút công cụ – UPDATE_COMMAND_UI**

```
void CVd2View::OnUpdateButton(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bWzd);
    // TODO: Add your command update UI handler code here
}
```

- Nếu m_bWzd = TRUE thì có hiệu lực

- Ngược lại m_bWzd = FALSE thì không có hiệu lực.

Thanh công cụ (ToolBar)

- **Thêm text vào nút :**
  - ***phương thức :***

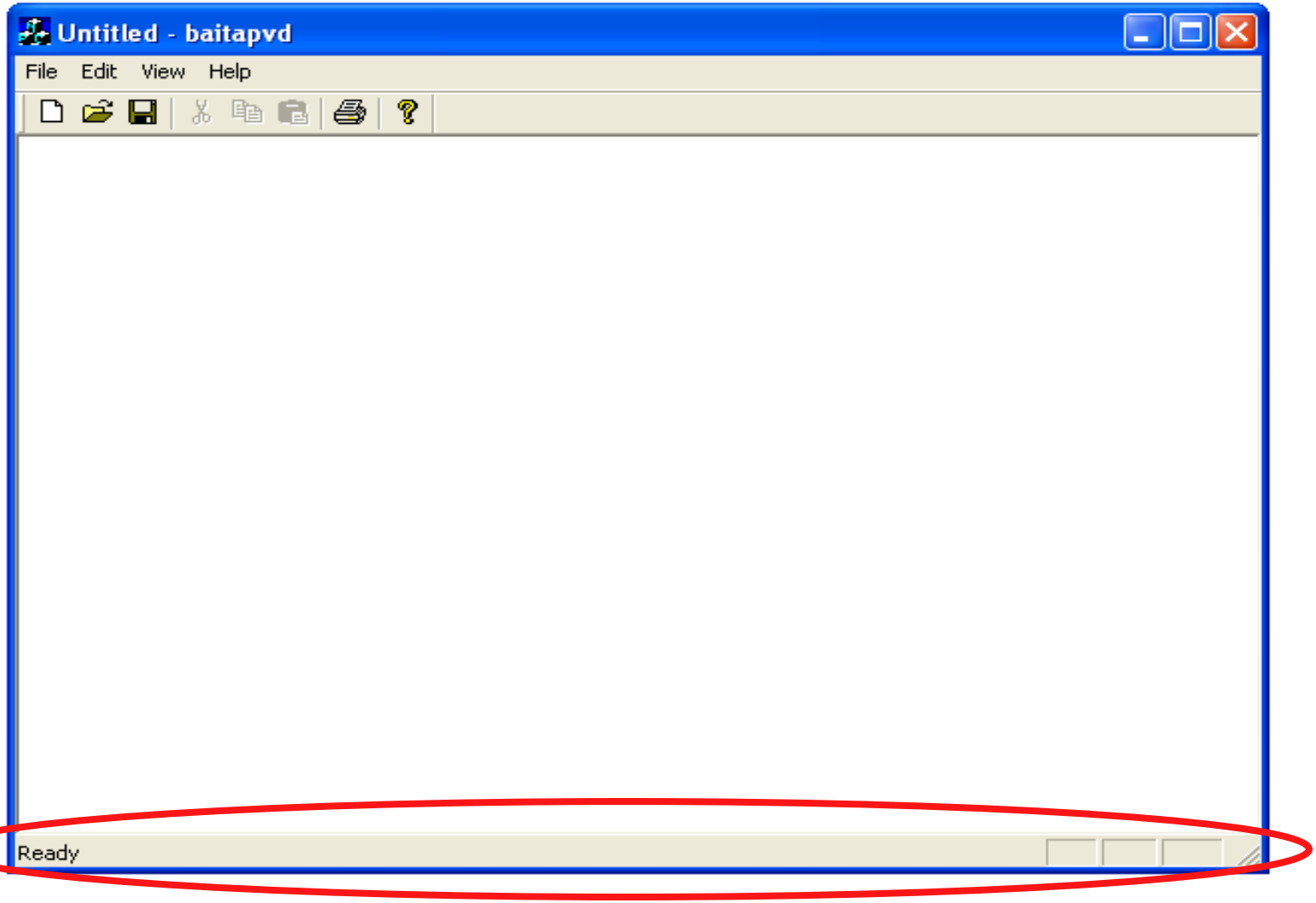
    SetButtonText(int nIndex, LPCTSTR lpszText);
    - ***nIndex: số thứ tự của nút thêm text***
    - ***lpszText: chuỗi văn bản thêm vào***
  - ***Xác định kích thước các nút trong toolbar và kíchước iMage***

    SetSizes(SIZE sizeButton, SIZE sizeImage);

  Ví dụ:

  ```
  ToolBar.SetButtonText (9, _T ("Print"));
  ToolBar.SetSizes (CSize (48, 42), CSize (40, 19));
  ```

Thanh trạng thái (Status bar)



Thanh trạng thái (Status bar)

- **Thêm thanh trạng thái mới vào ứng dụng:**
  - **khai báo đối tượng m_wndStatusBar thuộc lớp CStatusBar trong thành phần protected của lớp CframeWnd.**
  - **Viết mã lệnh trong hàm Create**

```
if (!m_wndStatusBar.Create(this)
    ||!m_wndStatusBar.SetIndicators(indicators,
    sizeof(indicators)/sizeof(UINT)))
  {
            TRACE0("Failed to create status bar\n");
            return -1;      // fail to create
  }
```

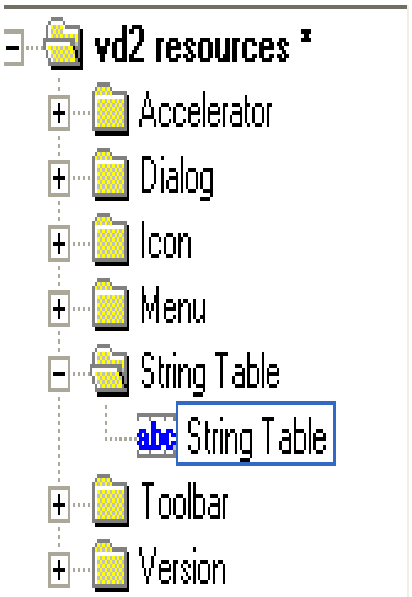
Thanh trạng thái (Status bar)

- **Hàm SetIndicator đặt các chỉ danh indicator vào thanh trạng thái, số indicator và indicator phải khai báo như sau trong đầu tập tin MainFrm.cpp**

```
static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_N1,
    ID_INDICATOR_TIME,
};
```

Thanh trạng thái (Status bar)

Mỗi ID_INDICATOR tương ứng với một chuỗi trạng thái cần hiển thị và được khai báo trong tập tin tài nguyên String table



ID_INDICATOR_EXT	59136	EXT
ID_INDICATOR_CAPS	59137	CAP
ID_INDICATOR_NUM	59138	NUM
ID_INDICATOR_SCRL	59139	SCRL
ID_INDICATOR_OVR	59140	OVR
ID_INDICATOR_REC	59141	REC
ID_INDICATOR_N1	59142	Thong tin
ID_INDICATOR_TIME	59143	
ID_VIEW_TOOLBAR	59392	Show or hide the toolbar\nToggle ToolBar
ID_VIEW_STATUS_BAR	59393	Show or hide the status bar\nToggle StatusBar
AFX_IDS_SCSIZE	61184	Change the window size

Thanh trạng thái (Status bar)

- **Để hiện thị các thông tin lên indicator trên thanh trạng thái cần bổ sung hàm:**

void CMainFrame:: OnUpdateIndicatorN1(CCmdUI* pCmdUI)

{

// TODO: Add your command update UI handler code here

pCmdUI->Enable(TRUE);

}

Và bổ sung khai báo trong tập tin MainFrm.h

```
// Generated message map functions
protected:
    //{{AFX_MSG(CVd2View)
    afx_msg void OnViewToolbar();
    afx_msg void OnUpdateButton32774(CCmdUI* pCmdUI);
    afx_msg void OnUpdateIndicatorN1(CCmdUI* pCmdUI);
    afx_msg void OnButton32774();

    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

Thanh trạng thái (Status bar)

- ***Khai báo bảng thông điệp trong tập tin MainFrm.cpp***

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    ON_WM_CLOSE()
    ON_COMMAND(ID_GT_CN, OnGtCn)
    ON_UPDATE_COMMAND_UI(ID_GT_CN, OnUpdateGtCn)
    ON_COMMAND(ID_GT_CQ, OnGtCq)
    ON_UPDATE_COMMAND_UI(ID_GT_CQ, OnUpdateGtCq)
    ON_COMMAND(ID_NEW, OnNew)
    ON_COMMAND(ID_BUTTON32774, OnButton32774)
    ON_COMMAND(ID_BUTTON32775, OnButton32775)
    ON_COMMAND(ID_VIEW_TOOLBAR, OnViewToolbar)
    ON_UPDATE_COMMAND_UI(ID_VIEW_TOOLBAR, OnUpdateViewToolbar)
    ON_WM_TIMER()
    ON_UPDATE_COMMAND_UI (ID_INDICATOR_TIME, OnUpdateTime)
    ON_UPDATE_COMMAND_UI(ID_INDICATOR_N1, OnUpdateIndicatorN1)
    //}}AFX_MSG_MAP


END_MESSAGE_MAP()
```

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- Các hàm đồ họa cơ sở
- Xử lý văn bản
- Xửlý sự kiện chuột (mouse)
- Xử lý sự kiện phím (keyboard)

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖Device context:

- **Device context** là một cấu trúc dữ liệu được duy trì để kết hợp với một thiết bị hiển thị như màn hình, máy in,…
- Khi vẽ đầu tiên phải khai báo **device context** và sau khi sử dụng thiết bị **device context** phải giải phóng **device context**.

  - ***CDC * pDC = GetDC();; // Khai báo thiết bị DC.***
  - ***ReleaseDC(hWnd, hDC); // Giải phóng DC***

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- Hàm **LineTo(): Vẽ đường thẳng**

  pDC -> LineTo(int X, int Y);

  Vẽ đường thẳng từ vị trí hiện hành đến tọa độ X, Y

- Hàm **MoveToEx(): Thiết đặt vị trí hiện hành**

  pDC -> MoveToEx(int X, int Y);

  Tọa độ hiện hành được xác định bởi X, Y

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖ *Vẽ hình chữ nhật:*

pDC -> Rectangle(int upX, int upY, int lowX, int lowY)

- **upX, upY:** **góc trên bên trái**
- **lowX, lowY:** **góc dưới bên phải**

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖ *Vẽ Elip:*

*pDC->**Ellipse**(int **upX**, int **upY**, int **lowX**, int **lowY**)*

- **upX, upY:** **góc trên bên trái**
- **lowX, lowY:** **góc dưới bên phải**

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖ *Vẽ Pie:*

*pDC->**pie**(int **upX**, int **upY**, int **lowX**, int **lowY**, int **Xs**, int **Ys**, int **Xe**, int **Ye**)*

- **upX, upY: góc trên bên trái**
- **lowX, lowY: góc dưới bên phải**
- ***Xs, Ys: Điểm bắt đầu cung***
- ***Xe, Ye: Điểm kết thúc cung***

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- ***Vẽ đường liền kề nhau:***

  pDC->Polyline(POINT pt[], int n);

  - **pt: mảng chứa các điểm tọa độ**
  - **n : số lượng tọa độ**

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- ***Tạo thuộc tính bút vẽ:***

CPen pen(int style, int width, COLOREF color);

- **Style: kiểu bút vẽ và có các giá trị:**
  - PS_DASH : Gạch ngang
  - PS_DASHDOT : Gạch ngang chấm
  - PS_DASHDOTDOT : Gach ngang chấm chấm
  - PS_DOT : Chấm
  - PS_SOLID : Đường thẳng liền nét.
- **Width: độ dày của bút vẽ**
- ***Color: RGB(R, G, B)***

CPen *pPen = pDC->SelectObject(&pen) //thiết lập

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- ***Tạo chổi quét:***
  - ***Cbrush Brush (COLOREF color)***
    - ***Color: RGB(R, G, B)***

- ***Thiết lập chổi quét***
  - ***CBrush *pBrush = pDC->SelectObject (&Brush)***

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖ Hàm TextOut:

pDC->TextOut(x, y, string, strLength);

Trong đó:

- **pDC** là device context của cửa sổ cần xuất.
- **x, y** điểm bắt đầu của chuỗi ký tự trong client
- **string** là một con trỏ trỏ đốn một chuỗi ký tự.
- **strLength** là số ký tự trong chỗi cần xuất.

Ví dụ:

- CString string(“This is text”);
- pDC->TextOut(x, y, str, str.GetLength());
- pDC-> SetTextAlign(TA_CENTER) // TA_RIGHT

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- **Thiết lập màu chữ**
  - **pDC-> SetTextColor(*COLOREF color*)**
- **Thiết lập màu nền**
  - **pDC-> SetBkColor(*COLOREF color*)**

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- **Font**: có các thuộc tính
  - **Typeface**: loại font (Times, Courier, Arial,…)
  - **Style**: kiểu dáng (normal, thin, bold,…)
  - **Size**: kích cỡ chữ, được xác định theo đơn vị point, 1 point = 1/72 inch = 0.013837 inch

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

❖ Thiết lập font chữ cho vùng client

- ***1. CFont Font; //khai báo đối tượng Font***
- **2**. Gọi **Font.CreateFont** để tạo ra font có thuộc tính xác định

```
Font.CreateFont(int Height, //kích thước điểm
                0, 0, 0,
                FW_NORMAL, // Độ dầy, có thể là FW_BOLD
                0,                  // nếu 1  nghiêng
                0,                  // Nếu 1 gạch dưới
                0,                  //nếu 1 gạch ngang
                0,0, 0, 0,0,“VNI-Times”); //kiểu chữ
```

- **3**. Gọi **SelectObject**() cài đặt Font vào vùng client.

CFont *pFont = (CFont *) pDC->SelectObject(&font);

SetFont(pFont); //thiết lập

ĐỒ HỌA VÀ XỬ LÝ CÁC THÔNG ĐIỆP ĐẦU VÀO

- **Thuật vẽ bitmap**
  - **Tạo nguồn bitmap trong tập tin tài nguyên** *:*
    - ***ID_BITMAP BITMAP "c:\filebitmap.bmp"***
  - **Nạp bitmap từ nguồn tài nguyên**
    - CBitmap bitmap; *//khao báo*
    - bitmap.LoadBitmap(IDB_BITMAP);
  - ***Đạt được DC của bộ nhớ sẽ chứa bitmap***
    - CDC DcComp;
    - DcComp.CreateCompatibleDC(pDC);
  - ***Chọn bitmap trong device context của bộ nhớ.***
    - DcComp.SelectObject(bitmap);

Thuật vẽ bitmap (TT)

- **Nhận thông tin của BitMap**
  - BITMAP bmpinfo; *//khao báo*
  - bitmap.GetObject(sizeof(bmpinfo), &bmpinfo));
- ***Sao chép bitmap từ DC của bộ nhớ sang DC của cửa sổ***
  - *PDC->BitBlt(x,y, bmpinfo.bmWidth, bmpinfo.bmHeight, &DcComp,0,0,SRCCOPY);*

CÁC THÔNG ĐIỆP SỰ KIỆN CHUỘT

- **Các thông điệp MOUSE trong vùng client**
  - ***Khi mouse di chuyển trên màn hình luôn phát ra thông điệp WM_MOUSEMOVE và khi các nút mouse được nhấn hay thả trong vùng client của một cửa sổ hàm window nhận các thông điệp:***

Nút	Nhấn	Nhả	Nhấn (lần 2)
Trái	WM_LBUTTONDOWN	WM_LBUTTONUP	WM_LBUTTONDBLCLK
Giữa	WM_MBUTTONDOWN	WM_MBUTTONUP	WM_MBUTTONDBLCLK
Phải	WM_RBUTTONDOWN	WM_RBUTTONUP	WM_RBUTTONDBLCLK

CÁC THÔNG ĐIỆP SỰ KIỆN CHUỘT

- **Hàm xử lý thông điệp MOUSE:**
  - **Lớp :: Thông_Điệp (UINT nFlags, CPoint point)**
  - **Trong đó:**
    - **nFlags: Cờ xác định nút chuột được nhấn hay nhả và nút đó là nút nào.**
      - MK_LBUTTON: Nút trái được nhấn
      - MK_MBUTTON: Nút giữa được nhấn
      - MK_RBUTTON: Nút phải được nhấn
    - **Point: Tọa độ xác định vị trí chuột**

CÁC THÔNG ĐIỆP SỰ KIỆN PHÍM

- Thông điệp bàn phím mà chương trình ứng dụng nhận được do **Windows** chuyển đến phân biệt tình huống phím gõ hay ký tự gõ.

- Các phím Shift, phím chức năng(F1, F2,….), các phím mũi tên di chuyển, các phím insert, Delete,… phát sinh phím gõ mà không phát sinh thông điệp ký tự gõ.

CÁC THÔNG ĐIỆP SỰ KIỆN PHÍM

Thông điệp phím gõ

- Khi một phím được nhấn Windows đặt thông điệp **WM_KEYDOWN** và đưa vào hàng đợi thông điệp của ứng dụng hay cửa sổ nhận được sự quan tâm
- Khi nhả phím Windows đặt thông điệp **WM_KEYUP** sẽ được hệ điều hành phân phát tới hàng đợi đó

CÁC THÔNG ĐIỆP SỰ KIỆN PHÍM

- **Mã phím ảo:** Cho biết giá trị của phím nhấn hay thả.

Giá trị thập phân	Giá trị Hex	Mã phím ảo	Phím
08	08	VK_BACK	Backspace
09	09	VK_TAB	Tab
13	0D	VK_RETURN	Enter
16	10	VK_SHIFT	Shift trái , phải
17	11	VK_CONTROL	Ctrl trái, phải
18	12	VK_MENU	Alt trái, phải
19	13	VK_PAUSE	Pause
20	14	VK_CAPITAL	Caps Lock
27	1B	VK_ESCAPE	Esc
32	20	VK_SPACE	Space bar
33	21	VK_PRIOR	Page up
34	22	VK_NEXT	Page down

CÁC THÔNG ĐIỆP SỰ KIỆN PHÍM

35	23	VK_END	End
36	24	VK_HOME	Home
37	25	VK_LEFT	Mũi tên trái
38	26	VK_UP	Mũi tên lên
39	27	VK_RIGHT	Mũi trón phải
40	28	VK_DOWN	Mũi tên xuống
45	2D	VK_INSERT	Insert
46	2E	VK_DELETE	Delete
112 –123	70- 7B	VK_F1,.... VK_F12	Phím chức năng
144	90	VK_NUMLOCK	Numlock
145	91	VK_SCROLL	Scroll Lock

CÁC THÔNG ĐIỆP SỰ KIỆN CHUỘT

- **Hàm xử lý thông điệp PHÍM:**
  - **Lớp :: Thông_Điệp (UINT nChar, UINT nRepCnt UINT nFlags)**
  - **Trong đó:**
    - **nChar: Phím được nhấn**
    - **nRepCnt: Số lần nhấn phím.**
      - **Nếu nhả thì giá trị = 1.**
    - **nFlags: Cờ xác định có kèm theo phím mở rộng được nhấn hay không.**

BỘ ĐỊNH THỜI TIMER

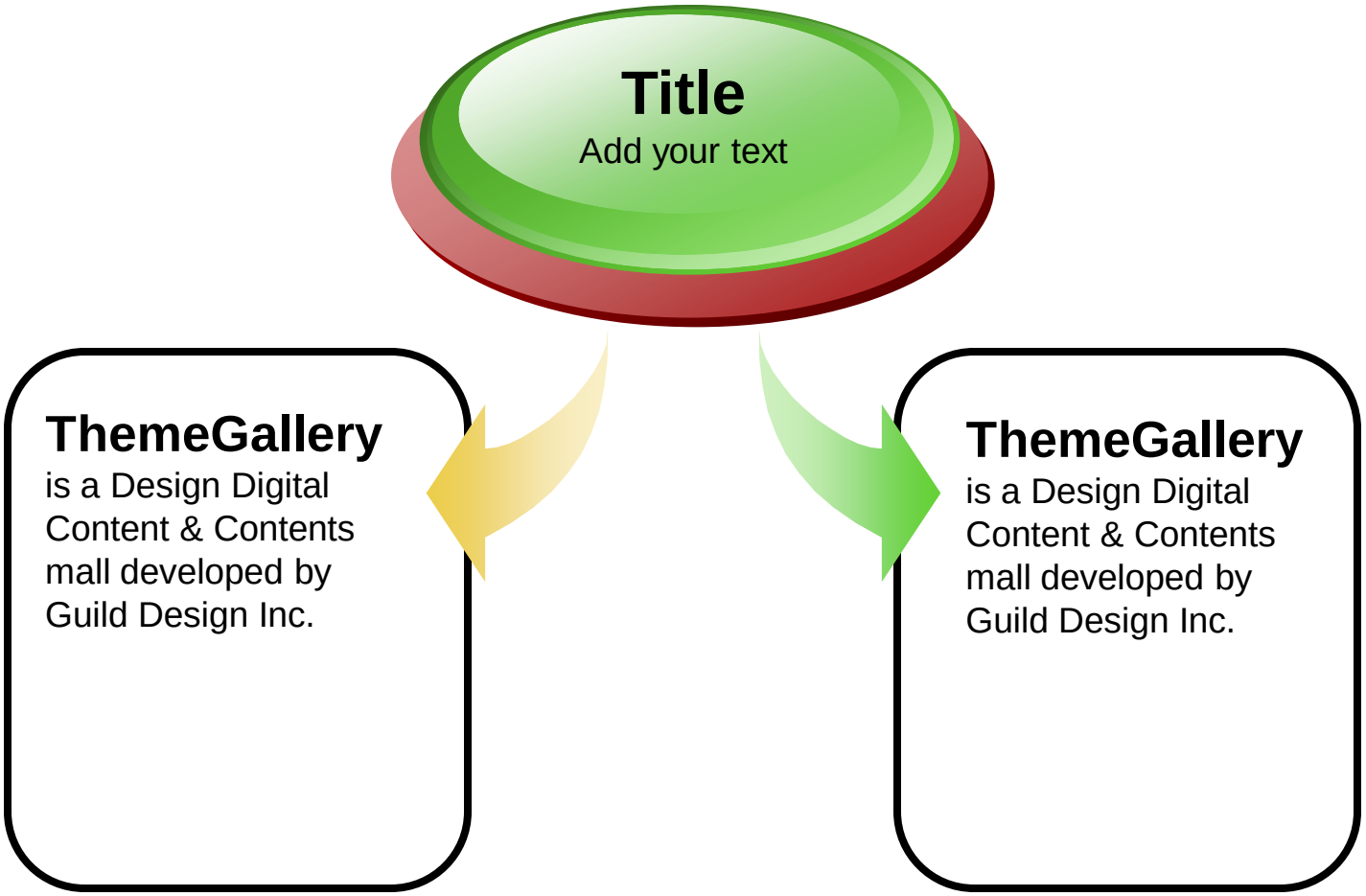
- Timer của hệ điều hành **Windows** là một dịch vụ định kỳ thông báo cho các ứng dụng biết rằng có một khoảng thời gian đã trôi qua.
- Thực hiện bằng cách sau một chu kỳ thời gian **Windows** gởi đến hàm window thông điệp **WM_TIMER**.

BỘ ĐỊNH THỜI TIMER

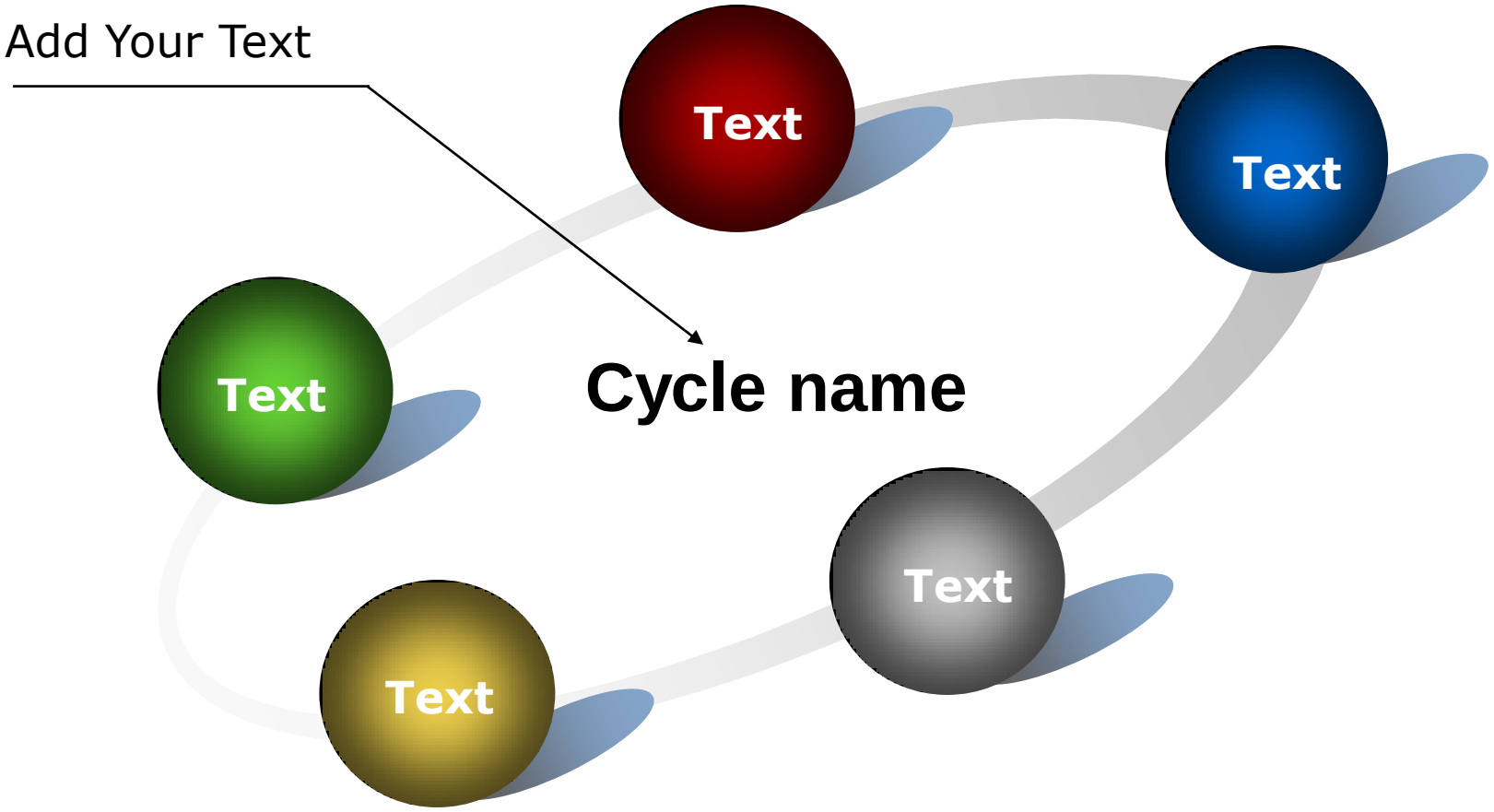
- Thiết lập bộ định thời: **sử dụng hàm** SetTimer**.**
  - ***UINT SetTimer(UINT nIDEvent, UINT uElapse, TIMERPROC lpTimerFunc);***

    - **nIDEvent : Chỉ danh của bộ Timer**
    - **uElapse: Chu kỳ thời gian mà Windows gởi thông điệp WM_TIMER tới hàm windows (tính bằng mili giây)**
    - **lpTimerFunc: Địa chỉ hàm timer – Có thể là NULL**

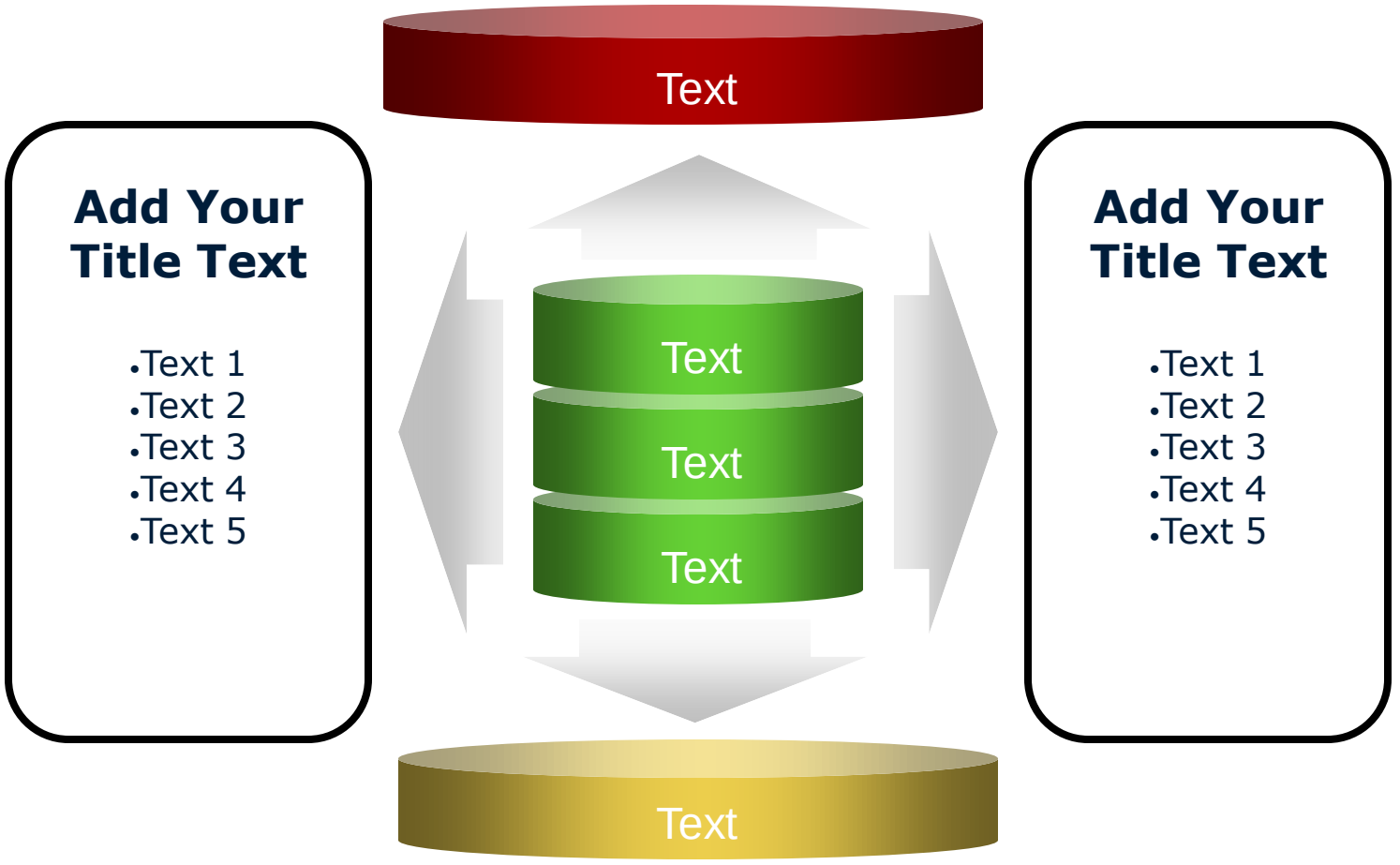
Diagram


Title
Add your text
ThemeGallery
is a Design Digital Content & Contents mall developed by Guild Design Inc.
ThemeGallery
is a Design Digital Content & Contents mall developed by Guild Design Inc.

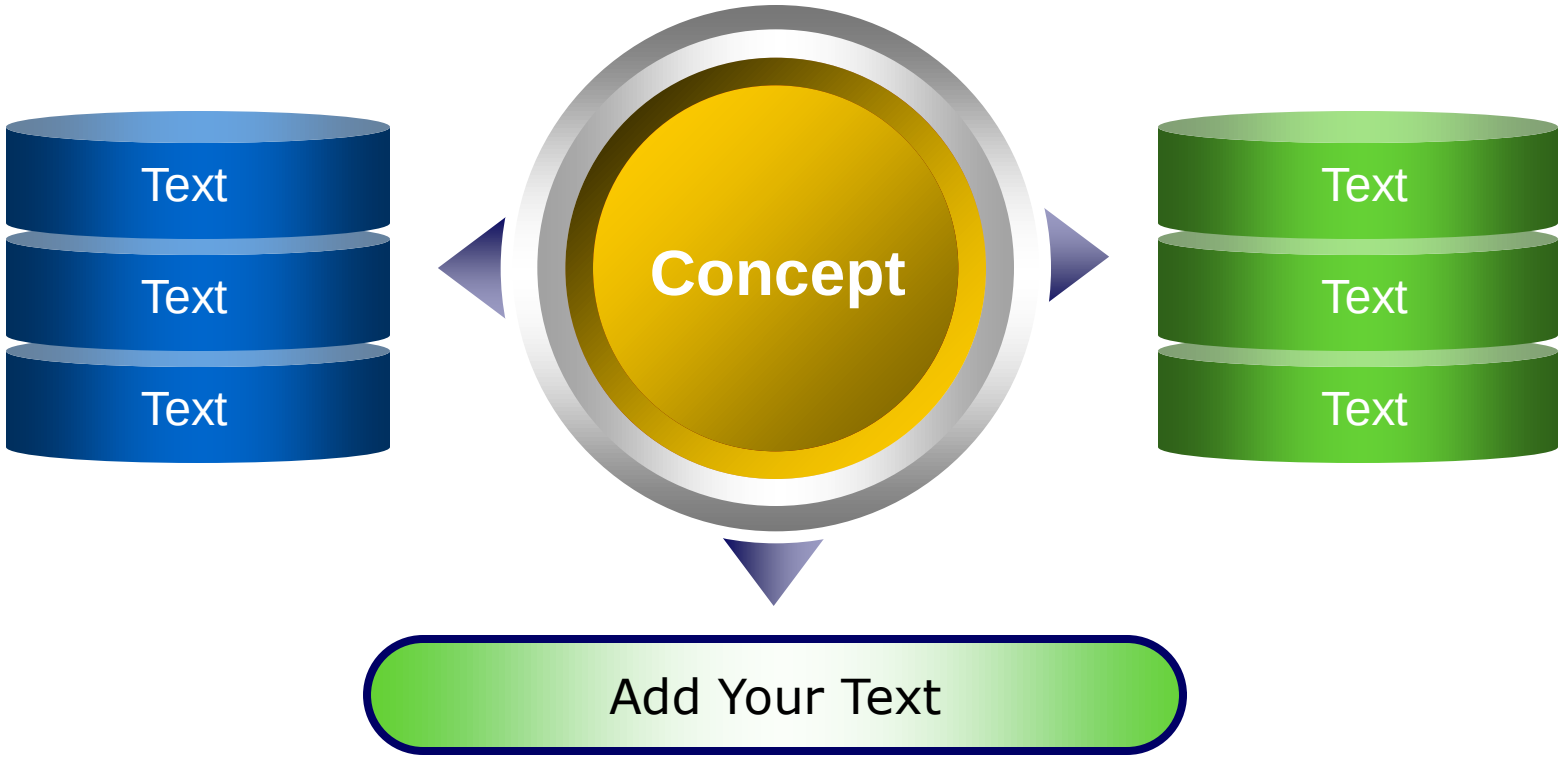
Cycle Diagram



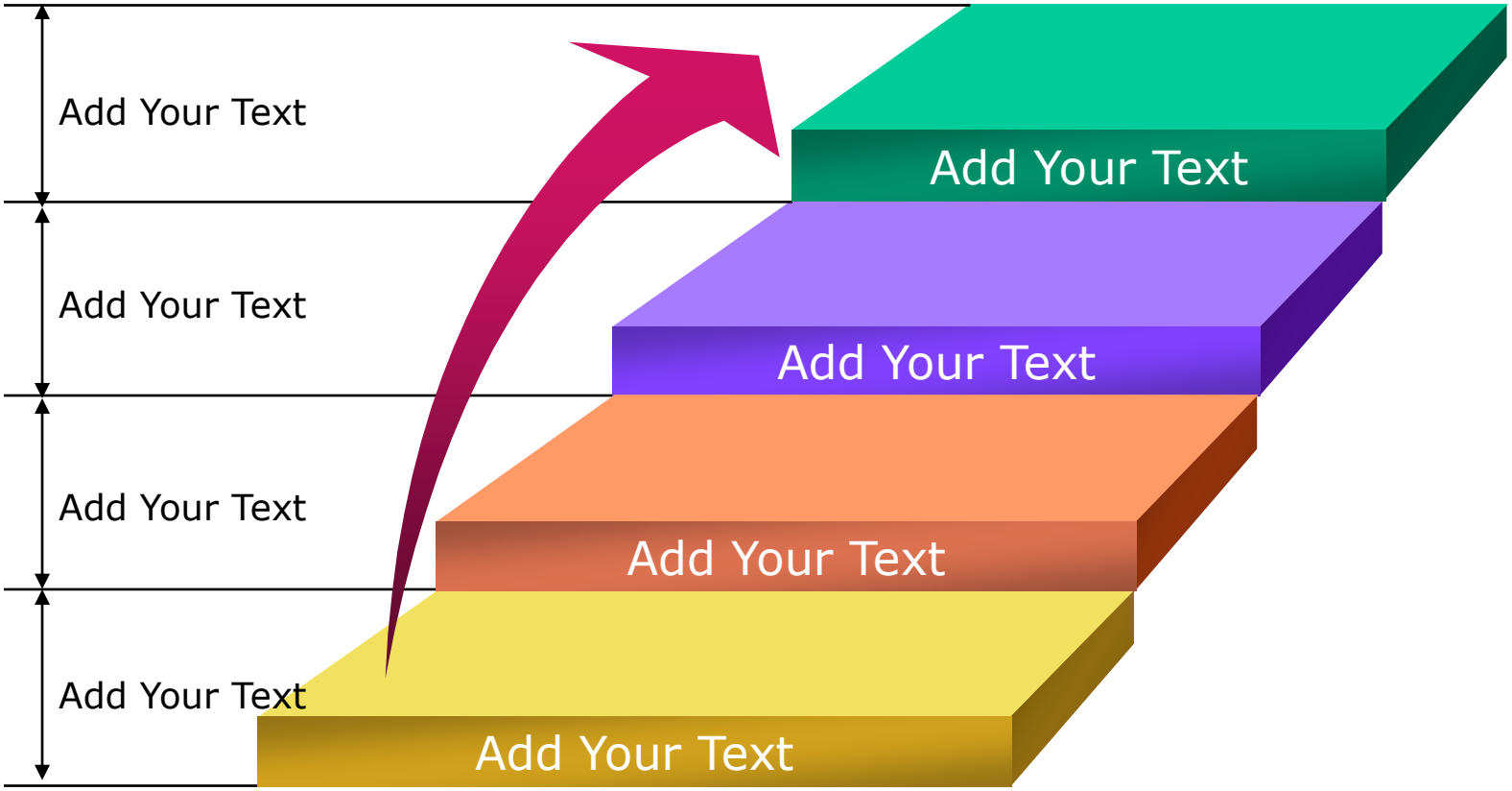
Diagram


Text
Add Your Title Text
•Text 1
•Text 2
•Text 3
•Text 4
•Text 5
Text
Text
Text
Add Your Title Text
•Text 1
•Text 2
•Text 3
•Text 4
•Text 5
Text

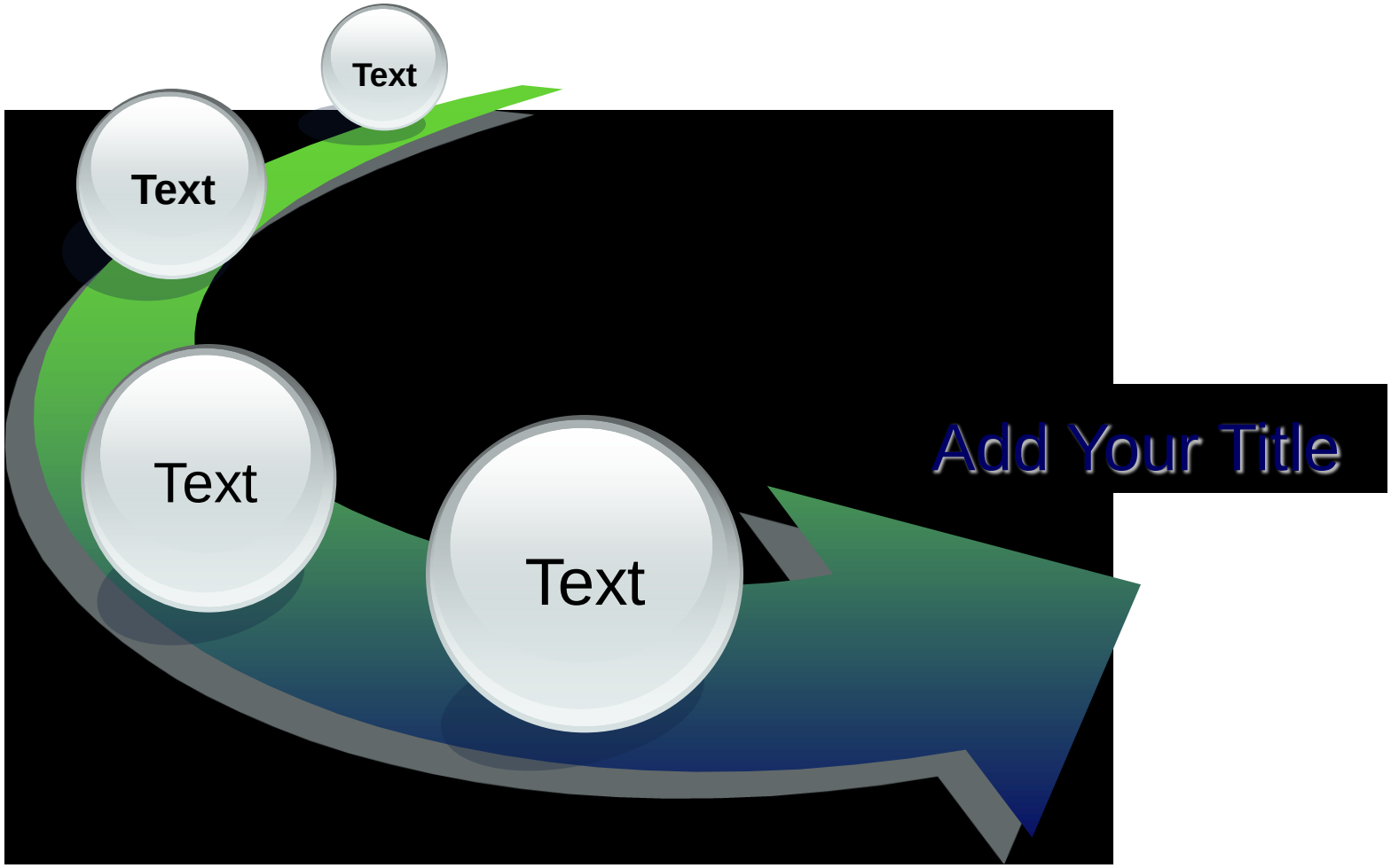
Diagram



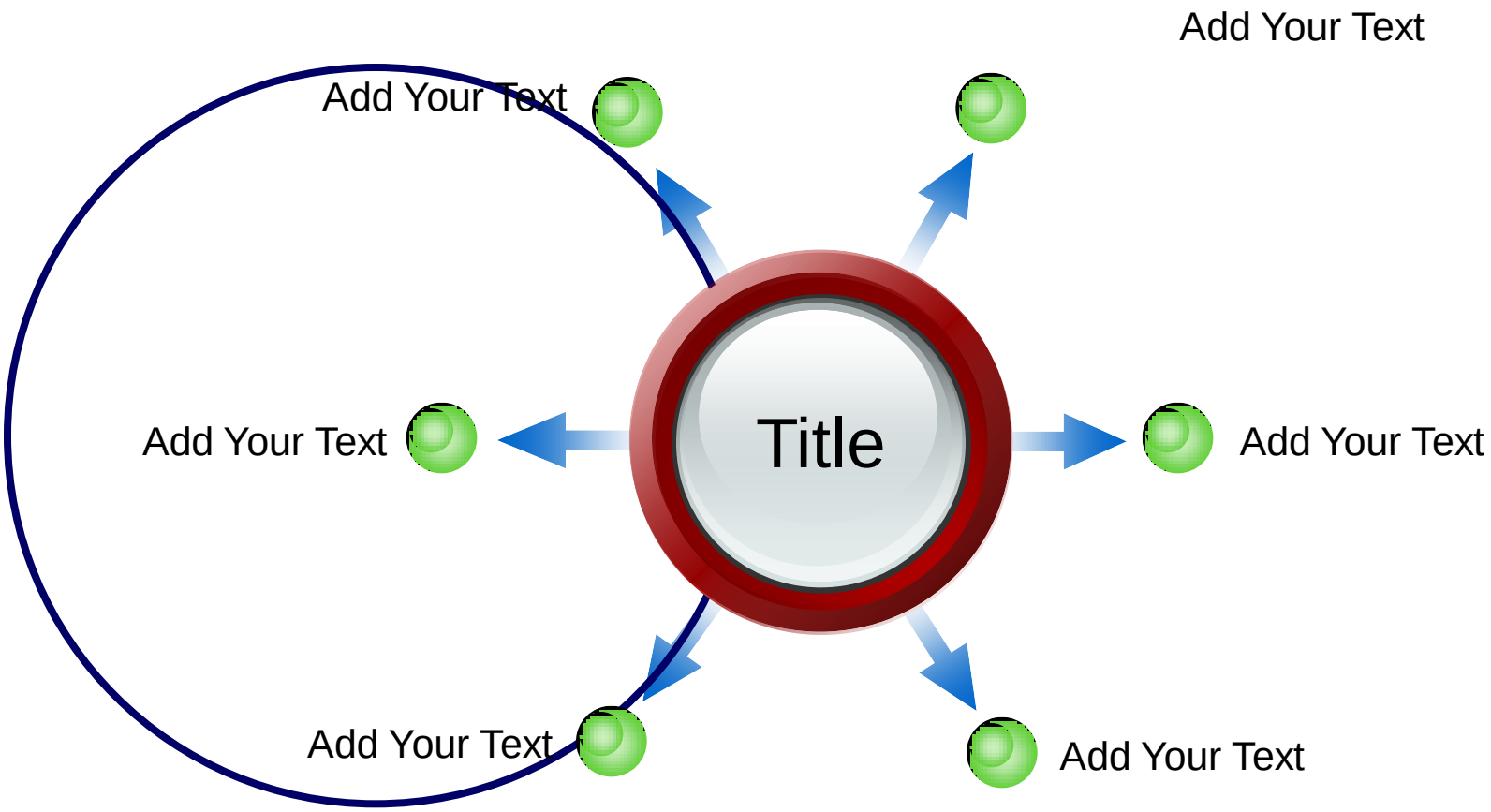
Diagram


Add Your Text
Add Your Text
Add Your Text
Add Your Text
Add Your Text
Add Your Text
Add Your Text
Add Your Text

Diagram



Diagram


Add Your Text
Add Your Text
Add Your Text
Title
Add Your Text
Add Your Text
Add Your Text

Diagram

1

ThemeGallery is a Design Digital Content & Contents mall developed by Guild Design Inc.

2

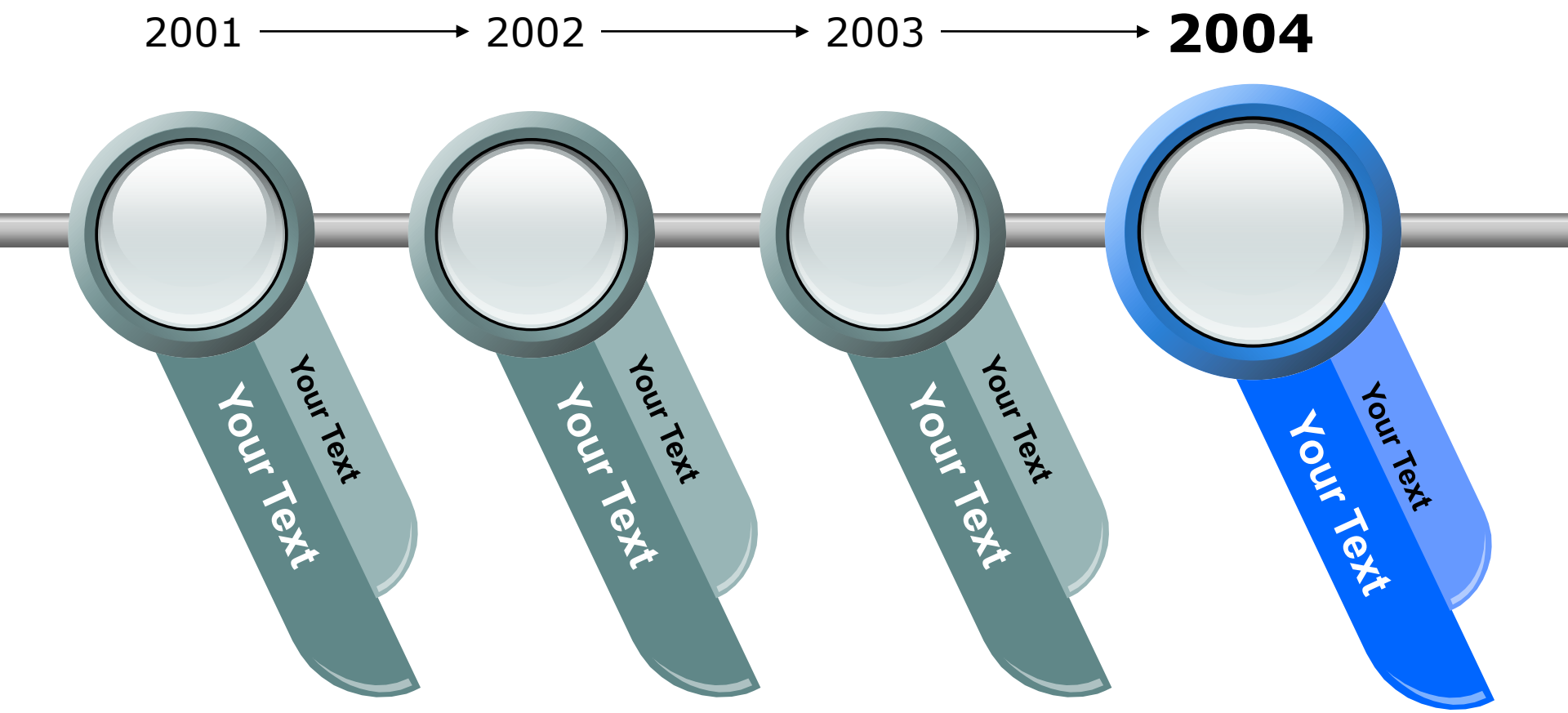
ThemeGallery is a Design Digital Content & Contents mall developed by Guild Design Inc.

3

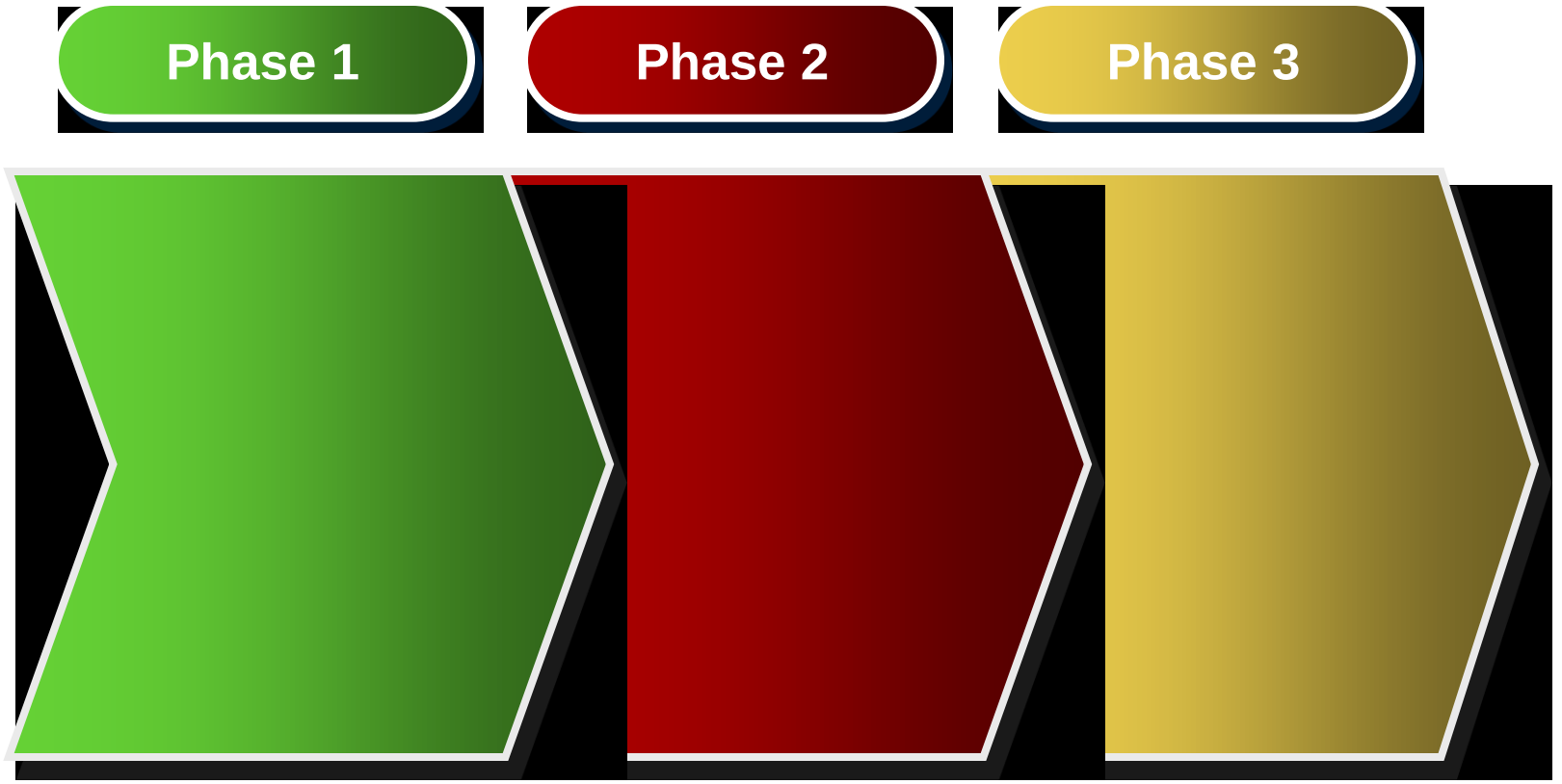
ThemeGallery is a Design Digital Content & Contents mall developed by Guild Design Inc.

Diagram

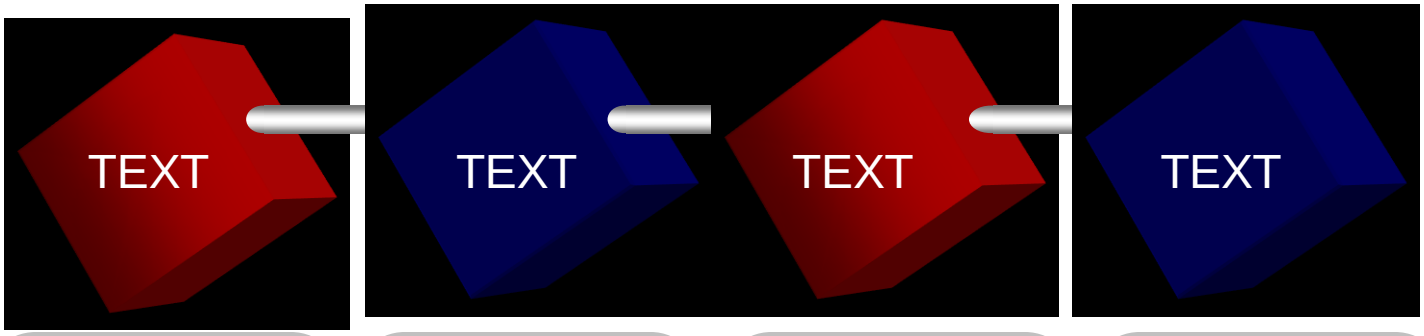
Diagram


2001
2002
2003
2004
Your Text
Your Text
Your Text
Your Text
Your Text
Your Text
Your Text
Your Text

Progress Diagram



Block Diagram

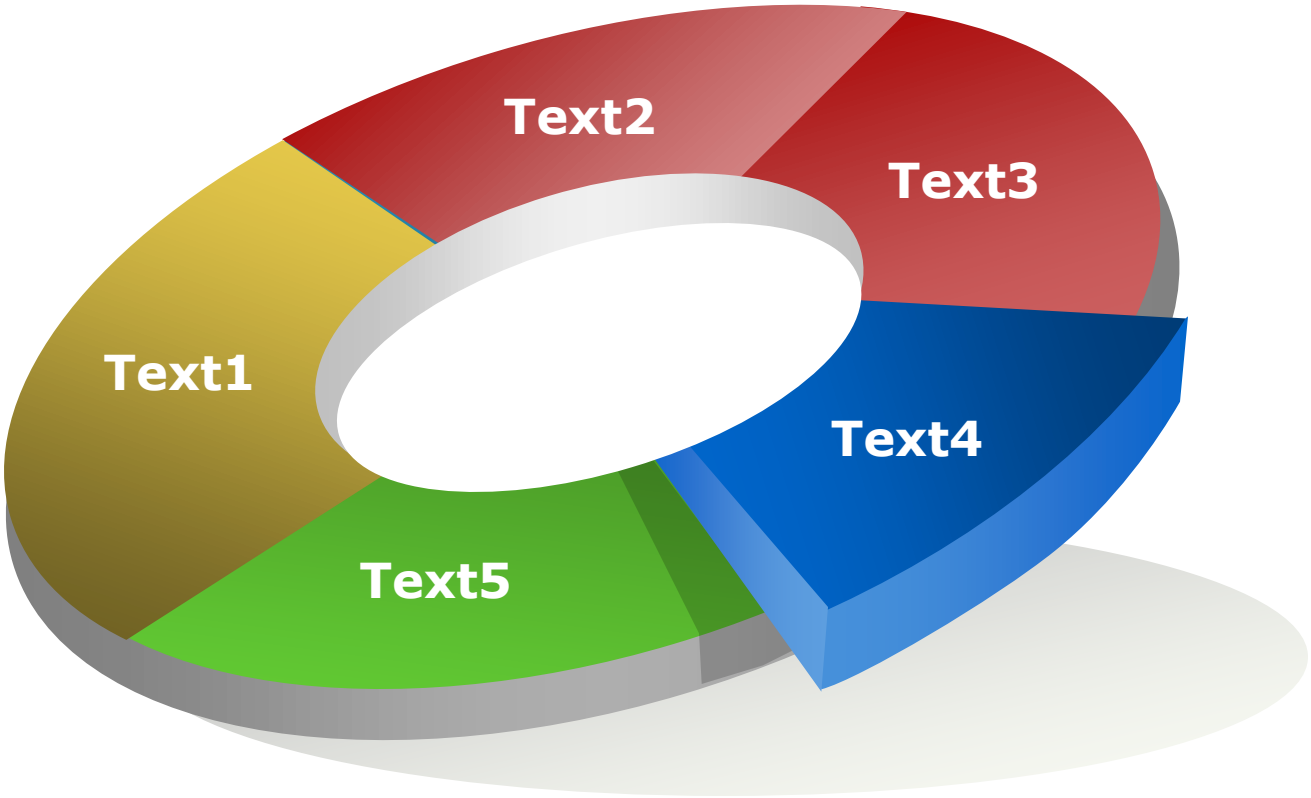


TEXT	TEXT	TEXT	TEXT

Table

	TEXT	TEXT	TEXT	TEXT	TEXT
Title A					
Title B					
Title C					
Title D					
Title E					
Title F					

3-D Pie Chart



Marketing Diagram

Add Your Text


Thank You !
www.themegallery.com

Lập trình Windows Form

Chương 1.

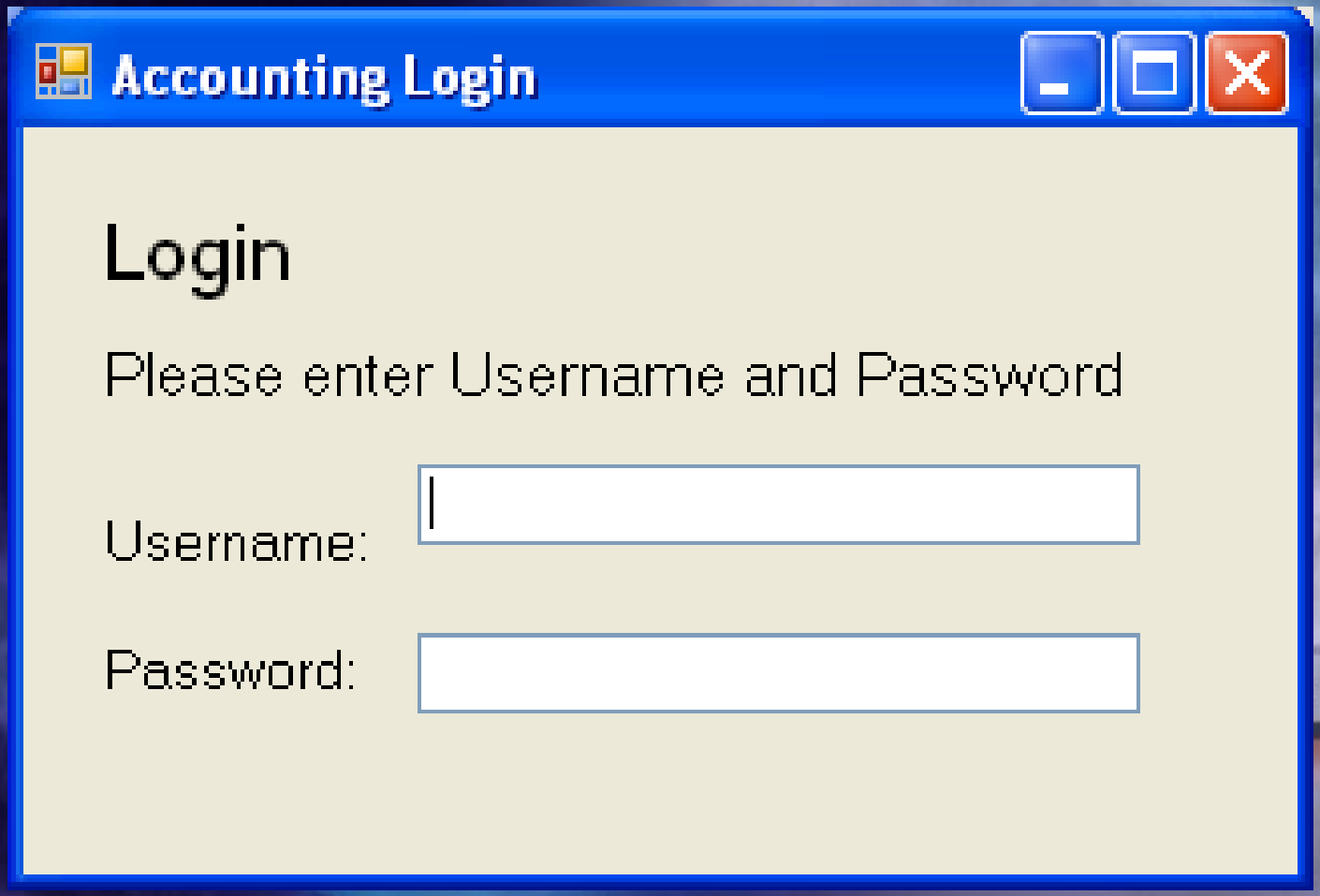
Giới thiệu Windows Form

1.1. Giới thiệu

- Tạo ra các ứng dụng chạy trên máy để bàn có cài đặt .NET Framework 2.0
- Sử dụng không gian tên System. Windows.Forms
- Thiết kế giao diện trực quan sử dụng Visual Studio 2005 IDE.

1.1. Giới thiệu

Ví dụ:

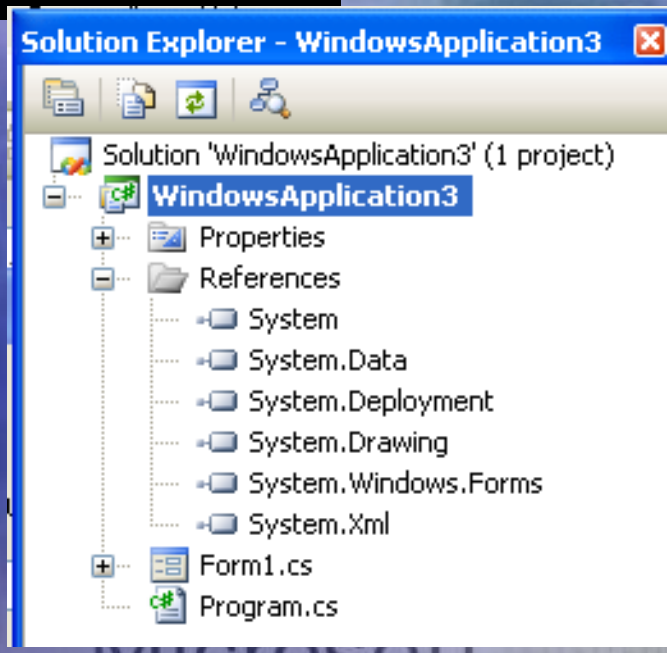


1.2. Ứng dụng Windows Forms

- Các chương trình quản lý tài chính, nhân sự, sản xuất, quản lý doanh nghiệp…

1.3. Không gian tên

- Khi tạo Project loại Windows Application có 6 không gian tên mặc định:
  - System,
  - System.Data,
  - System.Deployment
  - System.Drawing,
  - System.Windows.Forms
  - System.Xml.

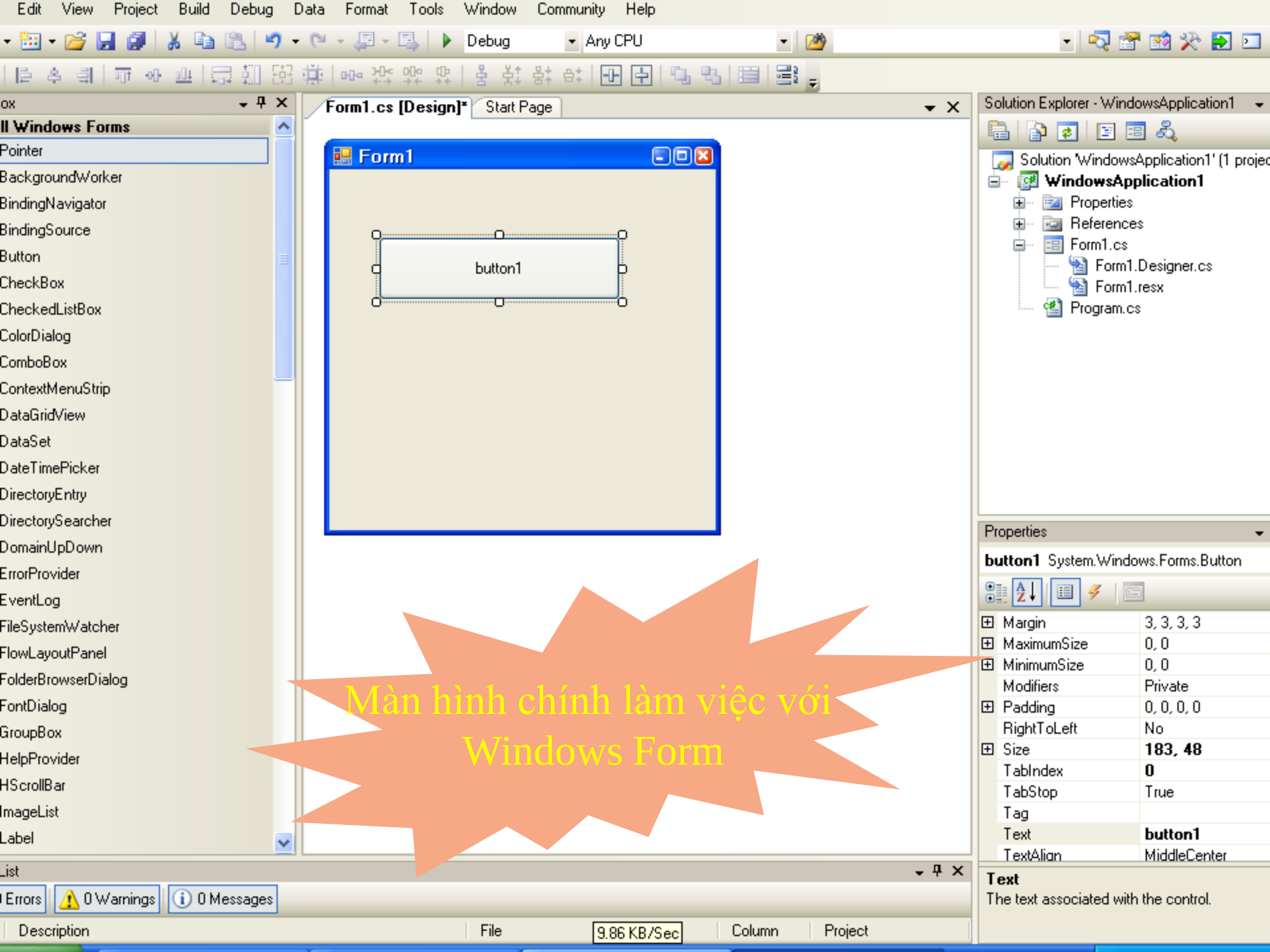


System.Windows.Forms

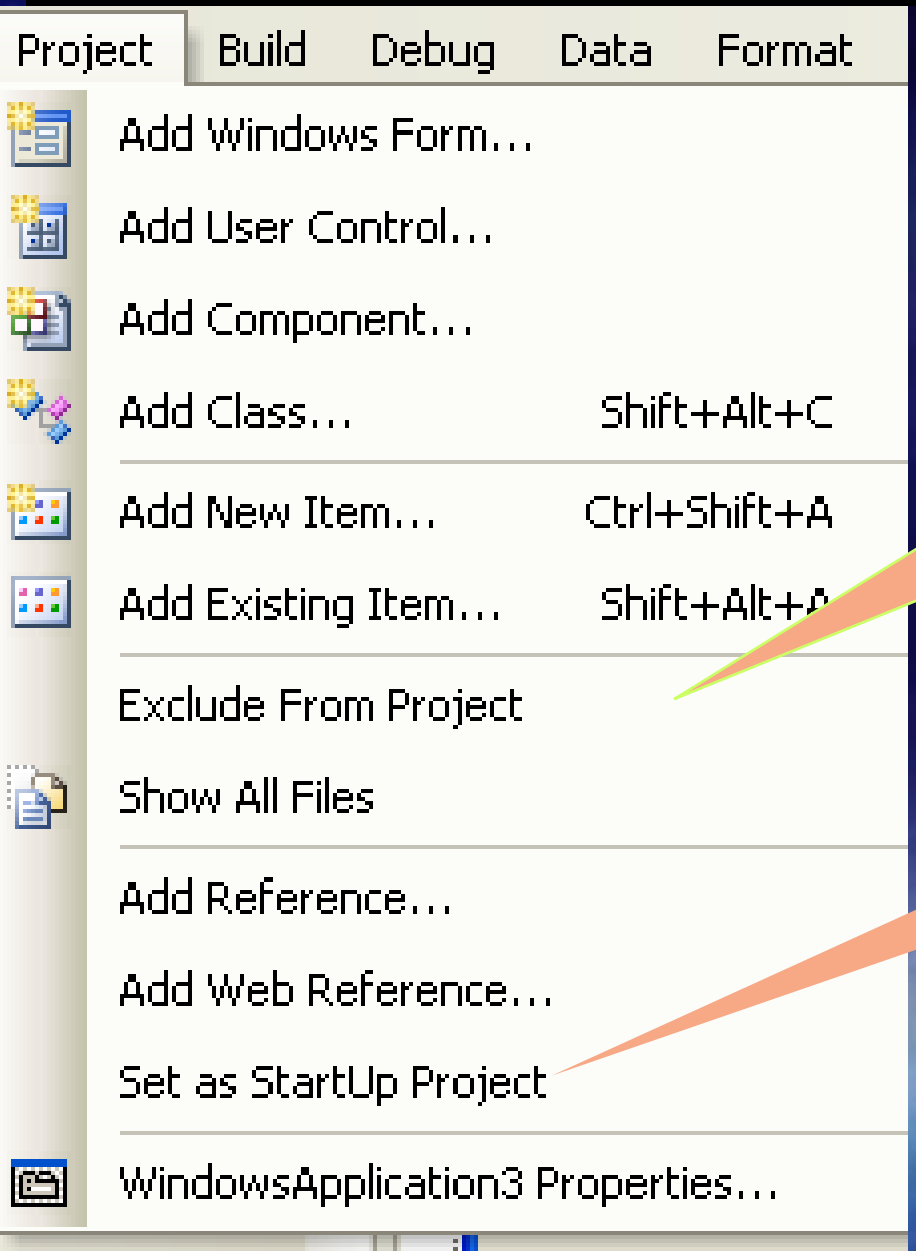
- Là không gian chính cung cấp các lớp dùng để xây dựng các ứng dụng Desktop.
- Các lớp của System.Windows.Forms chia thành các nhóm sau:
  - Control, User Control, Form
  - Menu, Toolbar: ToolStrip, MenuStrip, ContextMenuStrip, StatusStrip
  - Controls: Textbox, Combobox, Label, Listview, WebBrowser, HtmlDocument

System.Windows.Forms

- Layout: Giúp định dạng và tổ chức các điều khiển trình bày trên Form
- Data và Data Binding: định nghĩa kiến trúc đa dạng cho việc liên kết dữ liệu nguồn hay tệp tin XML vào các điều khiển.
  - VD: DataGridView
- Componets: ToolTop, ErrorProvider, HelpProvider
- Command Dialog Boxes: Làm việc với File, Font, Color, Print. VD: OpenFileDiaglog, SaveFileDiaglog. ColorFileDiaglog…


Edit View Project Build Debug Data Format Tools Window Community Help
Debug
Any CPU
Form1.cs [Design]*
Start Page
Form1
button1
Solution Explorer - WindowsApplication1
Solution 'WindowsApplication1' (1 projec
WindowsApplication1
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Pointer
BackgroundWorker
BindingNavigator
BindingSource
Button
CheckBox
CheckedListBox
ColorDialog
ComboBox
ContextMenuStrip
DataGridView
DataSet
DateTimePicker
DirectoryEntry
DirectorySearcher
DomainUpDown
ErrorProvider
EventLog
FileSystemWatcher
FlowLayoutPanel
FolderBrowserDialog
FontDialog
GroupBox
HelpProvider
HScrollBar
ImageList
Label
Properties
button1 System.Windows.Forms.Button
Margin 3, 3, 3, 3
MaximumSize 0, 0
MinimumSize 0, 0
Modifiers Private
Padding 0, 0, 0, 0
RightToLeft No
Size 183, 48
TabIndex 0
TabStop True
Tag
Text button1
TextAlign MiddleCenter
Màn hình chính làm việc với Windows Form
0 Warnings
0 Messages
Description
File
9.86 KB/Sec
Column
Project
Text
The text associated with the control.

1.4.Thực đơn Projector

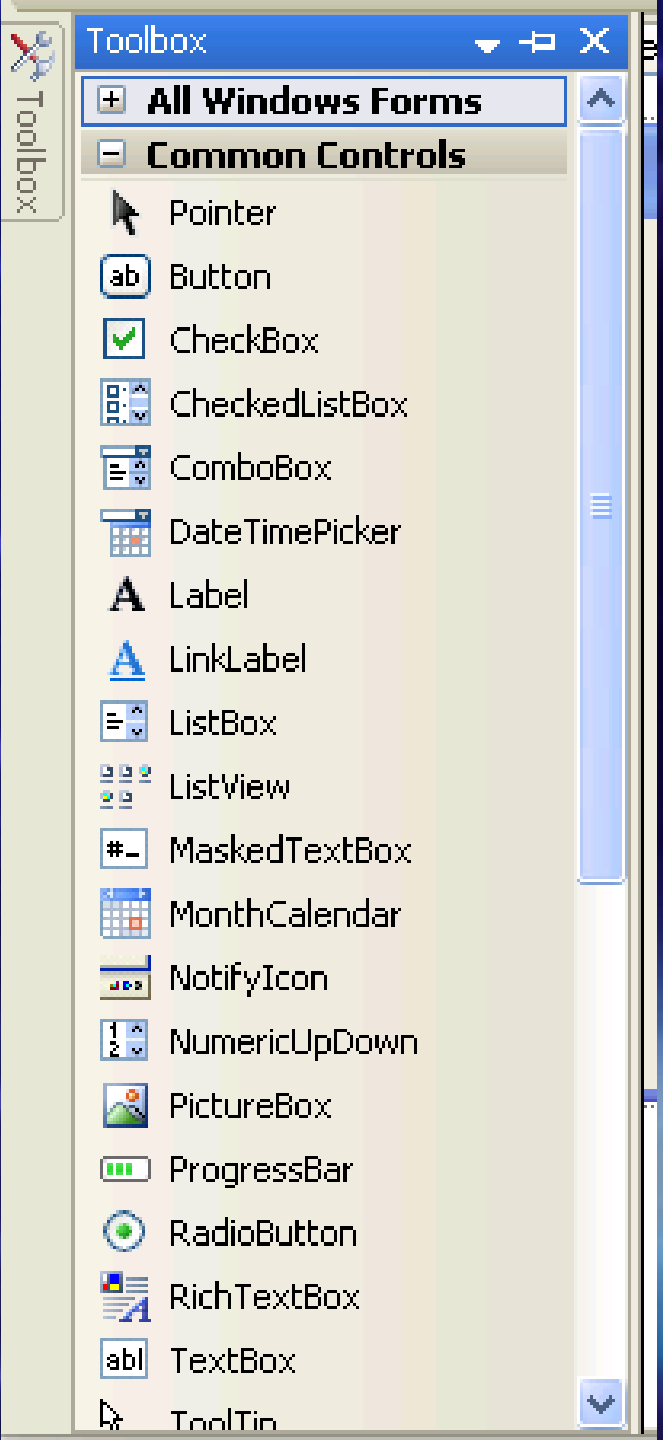


Loại bỏ đối tượng khỏi Project

Đặt Project khởi động

Microsoft .net

1.5. Hộp công cụ

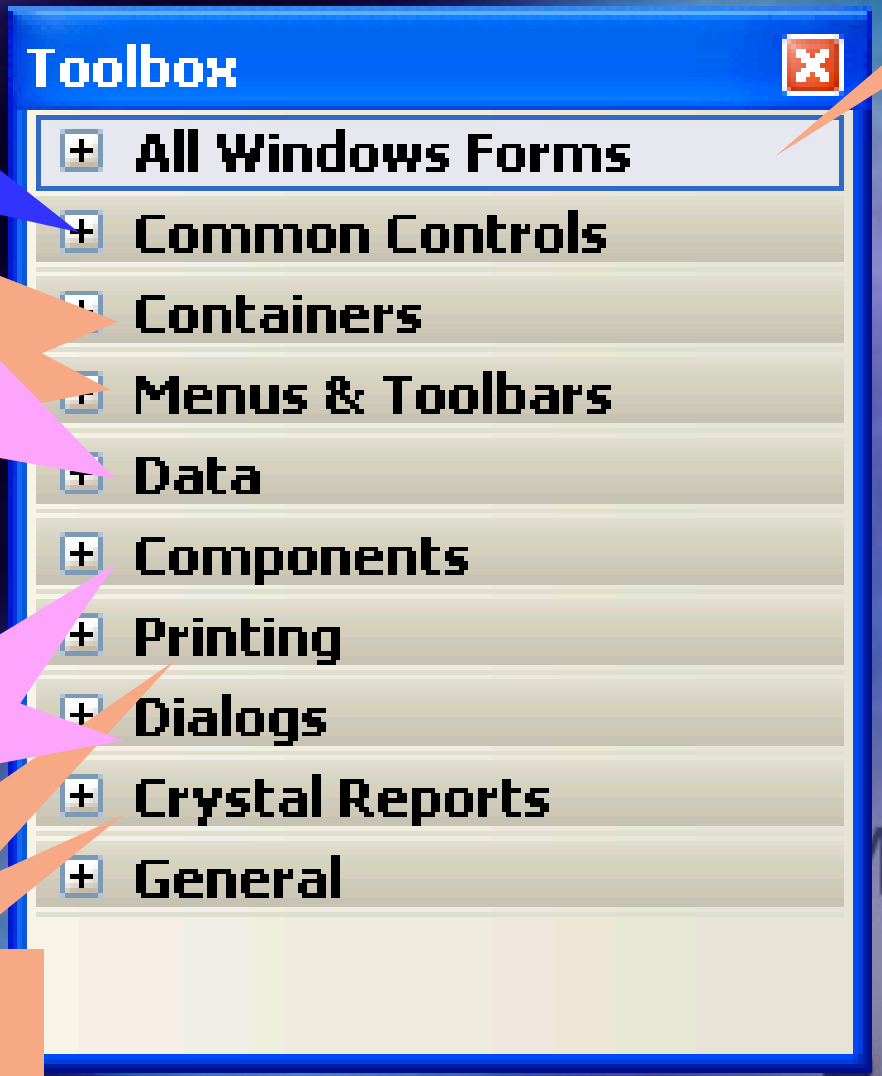


- Cung cấp danh sách các Component liệt kê theo nhóm, cho phép thiết kế giao tiếp với người dùng.
  - Windows Forms: Windows Control.
- Hiện ToolBox:
  - View \ Toolbox
  - nhấn chọn biểu tượng trên thanh công cụ
  - Ctrl+W và X

1.5. Hộp công cụ

Chứa các điều khiển thông thường: TextBox, Label, Button,

Chứa tất cả đối tượng

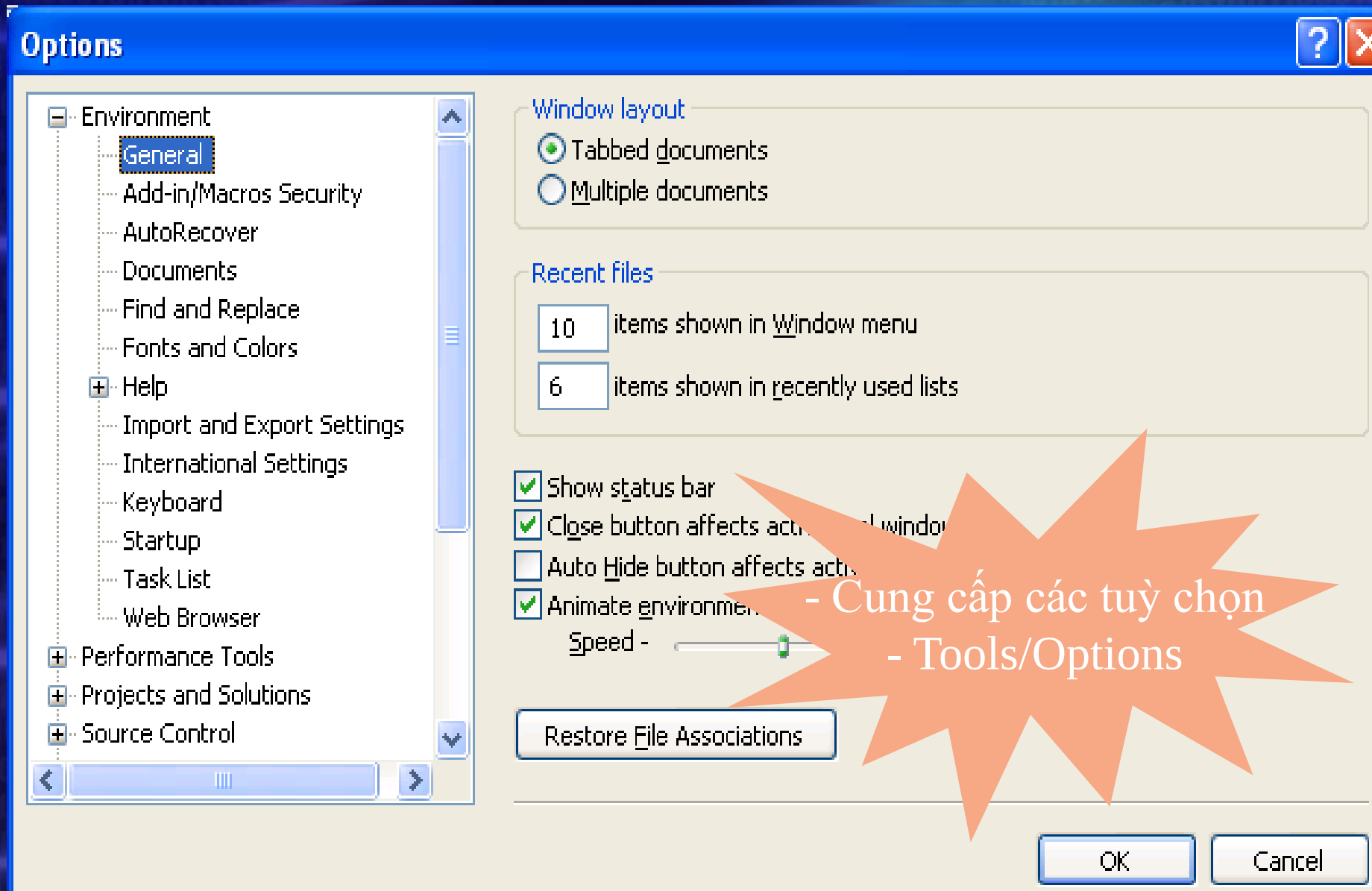


Chứa tất cả đối tượng làm việc với CSDL: DataSet, DataGridView,

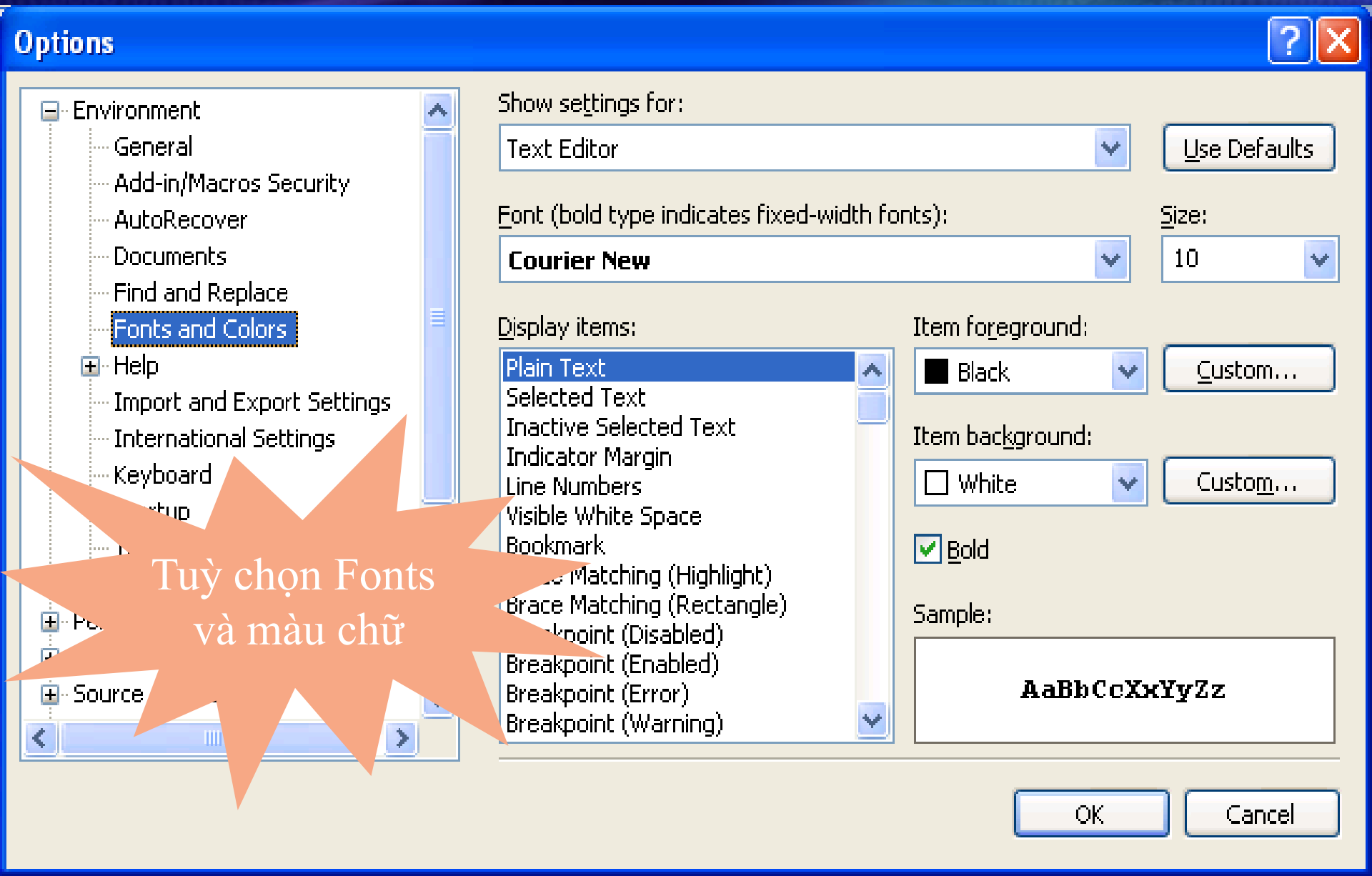
Cung cấp các điều khiển sử dụng để kiểm tra dữ liệu nhập như: ErrorProvider, HelpProvider,…

khiển làm việc với báo cáo

1.6. Cửa sổ Option

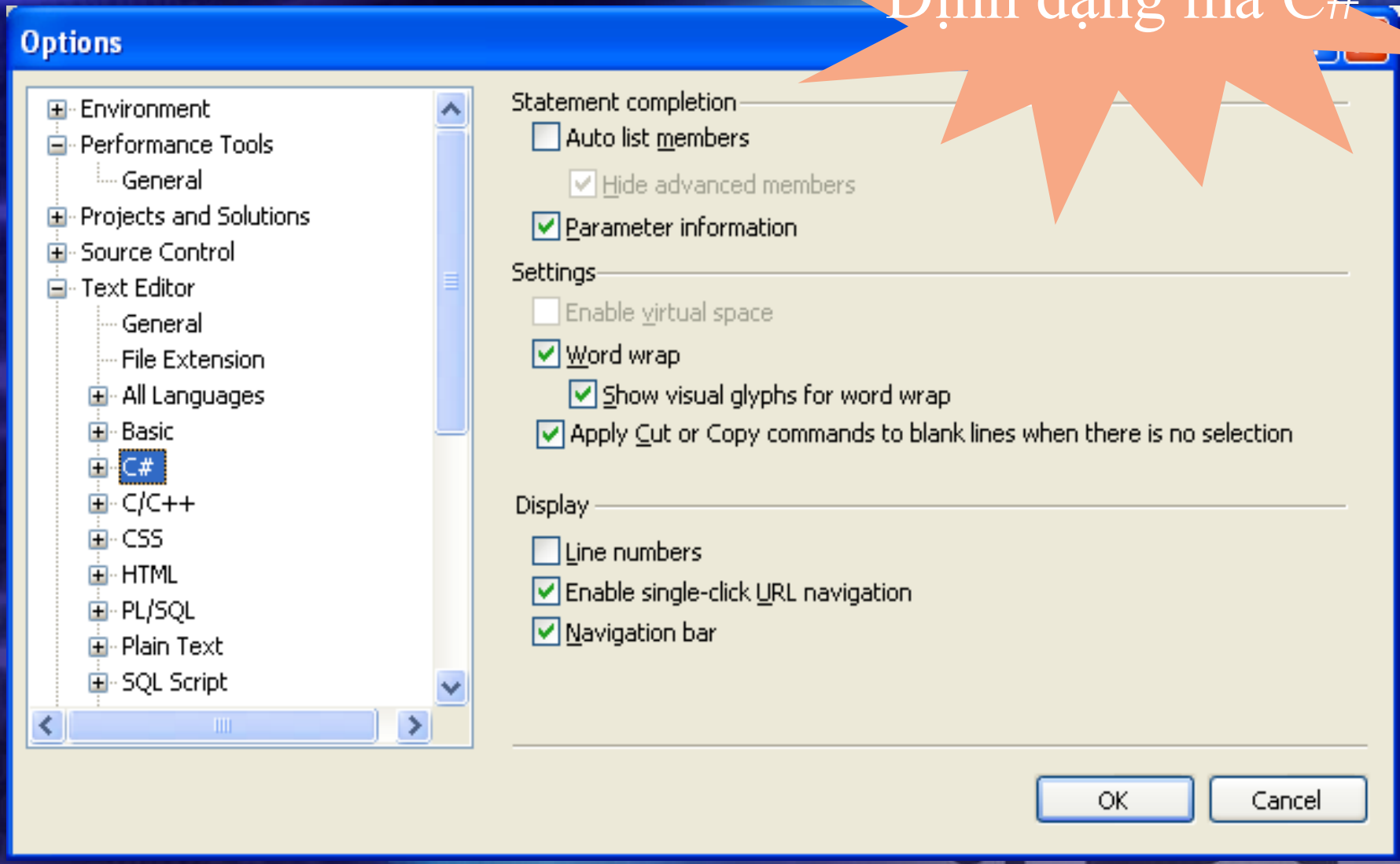


1.6. Cửa sổ Option



1.6. Cửa sổ Option

Định dạng mã C#



Thực hành

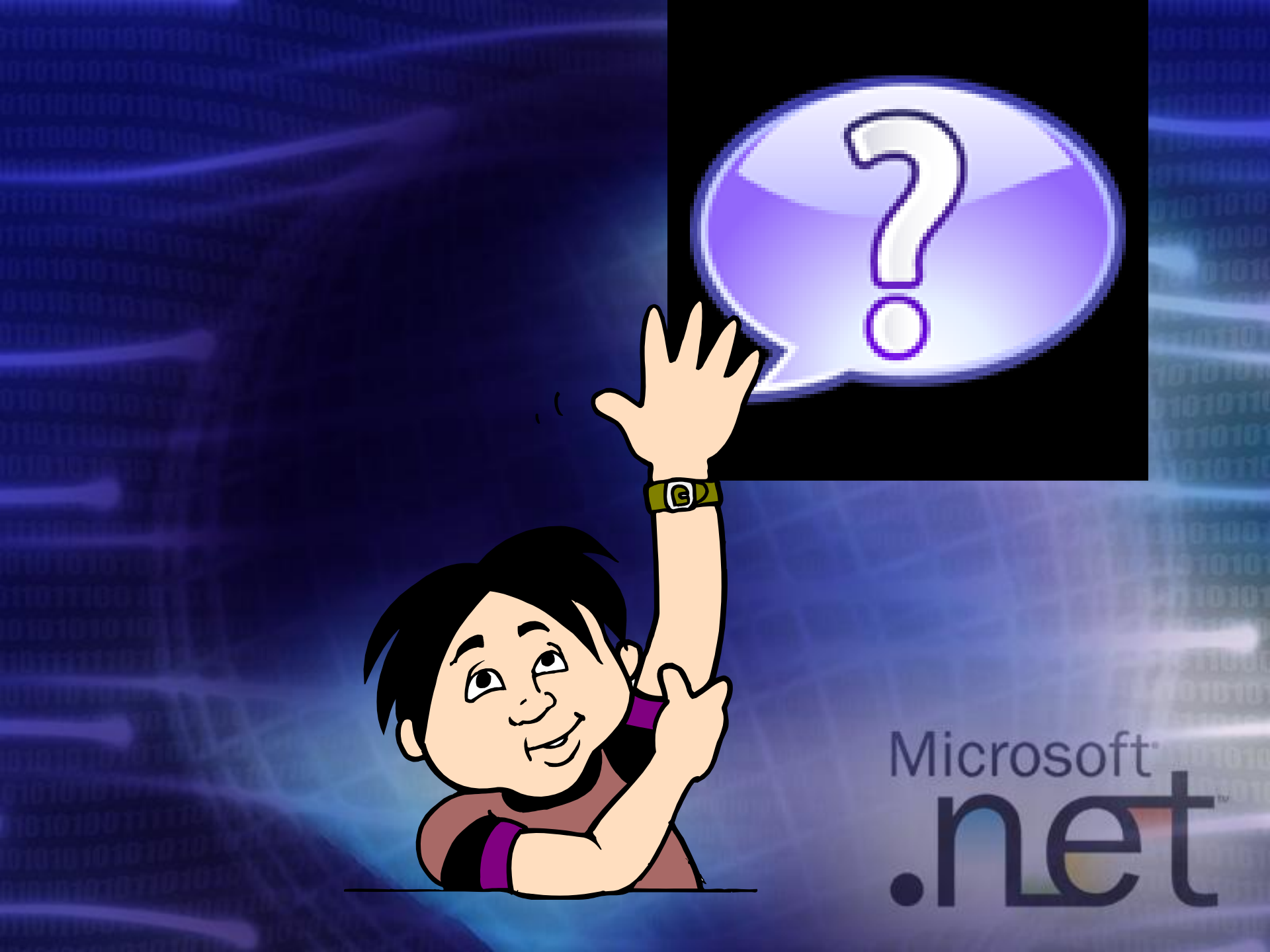
- Tìm hiểu nhanh về hệ thống thực đơn, thanh công cụ,Toolbox, cửa sổ Properties
- Đặt thuộc tính Font, cỡ chữ mặc định khi viết code
- Thay đổi màu nền, màu chữ
- Thêm số thứ tự đầu dòng ở mỗi dòng Code
- Đặt chế độ tự xuống dòng khi dòng code dài
- Đặt lại thư mục lưu Project

Microsoft .net

Thực hành – Bài tập

- Yêu cầu:
  - Nhập vào 3 số thực Double a, b, c; cần kiểm tra ngoại lệ nếu nhập a, b, c, không phải là số
  - Click vào nút tính nghiệm sẽ đưa ra kết quả ở Textbox4
  - Click vào Tiếp tục: Giải PT bạc hai khác
  - Click vào Thoát (hoặc Alt+T) sẽ thoát khỏi chương trình

Microsoft .net


Microsoft
.net

Chương 2.

Form và các định dạng Form

2.1. Các loại Form

- **MDI Form:**
  - Form chứa các form khác
  - Thuộc tính isMDIFormContainer=true
  - VD: Form frm=new Form2()

    Frm. isMDIFormContainer=true

    Frm.Show()

  ➔ Tạo Form2 và cho Form2 là MDI Form

2.1. Các loại Form

- **Child Form:**
  - Form nằm trong MDI Form
  - Phải khai báo thuộc tính MDIParent ứng với MDI Form
  - VD: Form Frm=new Form3()

    Frm. isMDIParent=this

    Frm.Show()

➔ This là từ khoá chỉ định Form gọi đến Form3 là MDI Form

2.1. Các loại Form

- **Normal Form:**
  - Không phải MDI Form hoặc ChildForm

Microsoft .net

Nạp Form

- VD: frm=new Form()
- Frm.Show(): Hiển thị Form
- Frm.ShowDialog(): Form mở ở dạng Modal. Form modal không cho phép người sử dụng dùng Form khác trtừ khi Form này được đóng lại

Microsoft .net

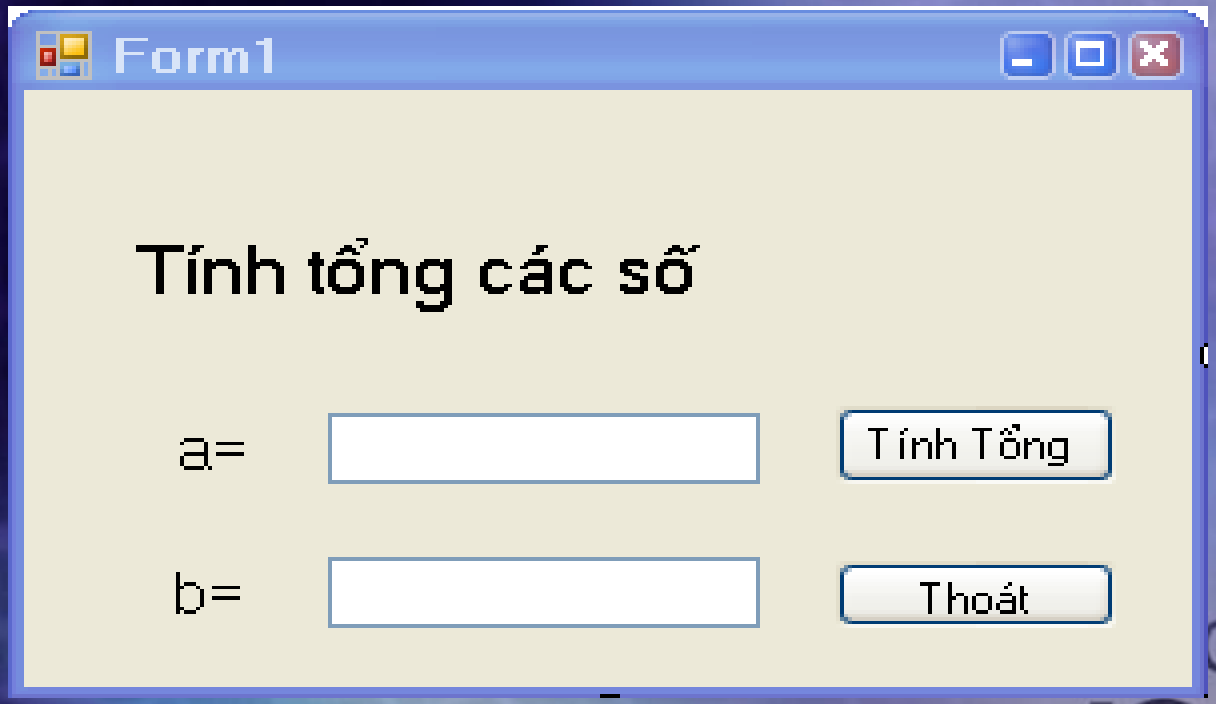
Tạo Form lúc thi hành

- Sử dụng từ khoá New để tạo Form, sau đó gán các thuộc tính cho Form
- VD:

```
Form Frm=new Form()
Frm.Text="New Form";
Frm.Show();
```

Form kế thừa

VD: Thiết kế Form1 như sau:



Form kế thừa

- Thêm Form2: Project\Add Windows Form
- D-Click vào Form2 xuất hiện

```
public partial class Form2 : Form1
{
    public Form2()
    {
        InitializeComponent();
    }
```

Thay class Form2: Form bởi class Form2: Form1

Form kế thừa

- Kết quả

Có thể thiết kế lại Form2

2.2. Các thuộc tính của Form

- **Nhóm thuộc tính nhận dạng**
  - **Name**: Tên duy nhất của đối tượng Form trong Project
  - **Text**: Chuỗi hiển thị trên thanh tiêu đề
  - **ShowIcon**=True: Cho hiện Icon góc trên bên trái; =False: Không hiện
  - **ShowInTaskBar**: =True: Khi chạy hiện biểu tượng trên TaskBar; False: Không hiện
  - **Icon**: Cho phép chỉ định tệp tin *.ico làm biểu tượng trên thanh tiêu đề của Form

2.2. Các thuộc tính của Form

- **Nhóm thuộc tính Định dạng**
  - **BackColor**: Màu nền của Form
    - VD: Form1.BackColor=Color.Azủe;
  - **ForeColor**: Màu của các chuỗi trên các Control của Form
  - **StartPossition**: Vị trí hiển thị Form
  - **WindowStates**: =Minimized (thu nhỏ), Maximized (phóng to), Nomal (trạng thái như thiết kế)
  - **isMDIContainer**: =True (Form được chọn là MDI Form); False: không
  - **ControlBox**

Thực hành

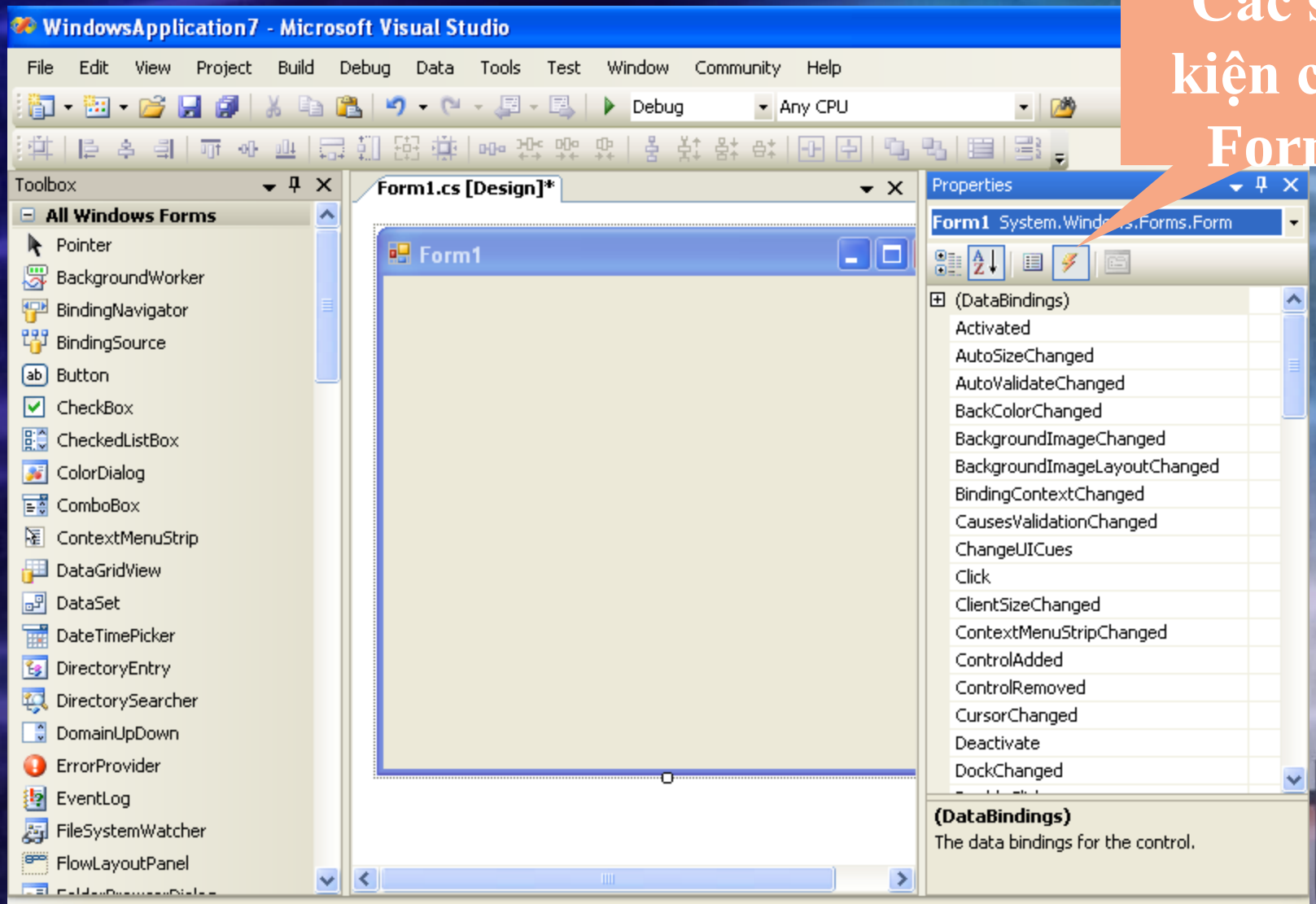
- Tạo Form và thử các thuộc tính của Form

2.3. Biến cố của Form

- **FormClosed:** Thực hiện khi Form đã đóng
- **FormClosing**: SỰ kiện khi đang đóng Form
- **Click**: Sự kiện khi Click vào Form
- **Activated**: Xảy ra khi Form được kích hoạt bằng mã hay do tác động của người sử dụng
- **Disactiave**: Xảy ra khi Form khác kích hoạt trên màn hình.
- **Load**: Xả ra khi nạp Form
- **KeyPress**: Xảy ra khi 1 phím được nhấn
- **Resize**: Xảy ra khi thay đổi kích thước Form

2.3. Biến cố của Form

Các sự kiện của Form



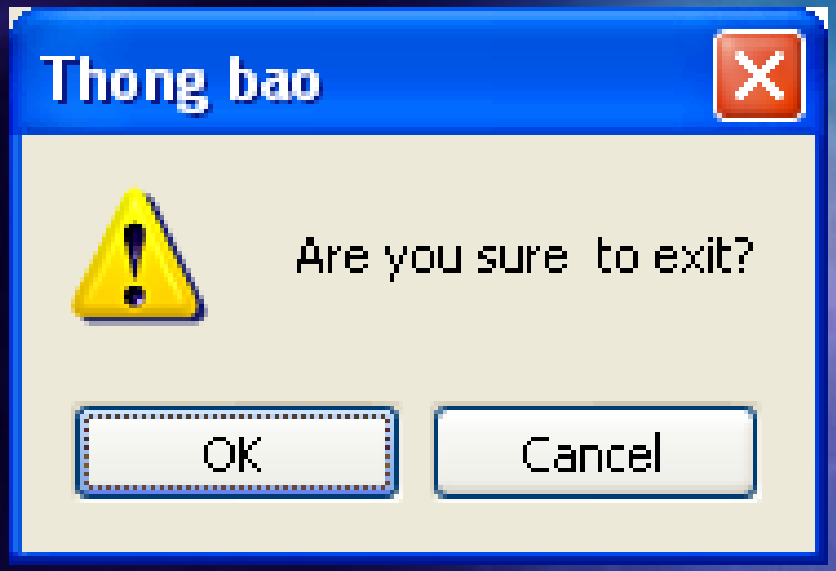
Ví dụ: Biến cố Load Form

```
private void Form1_Load(object sender,
    EventArgs e)
    {
    MessageBox.Show("Dang Load Form");
    //…
    }
```

Ví dụ: Biến cố Click form

```
private void Form1_Load(object sender,
    EventArgs e)
    {
    MessageBox.Show("Dang Load Form");
    //…
    }
```

Ví dụ: Biến cố Closing Form



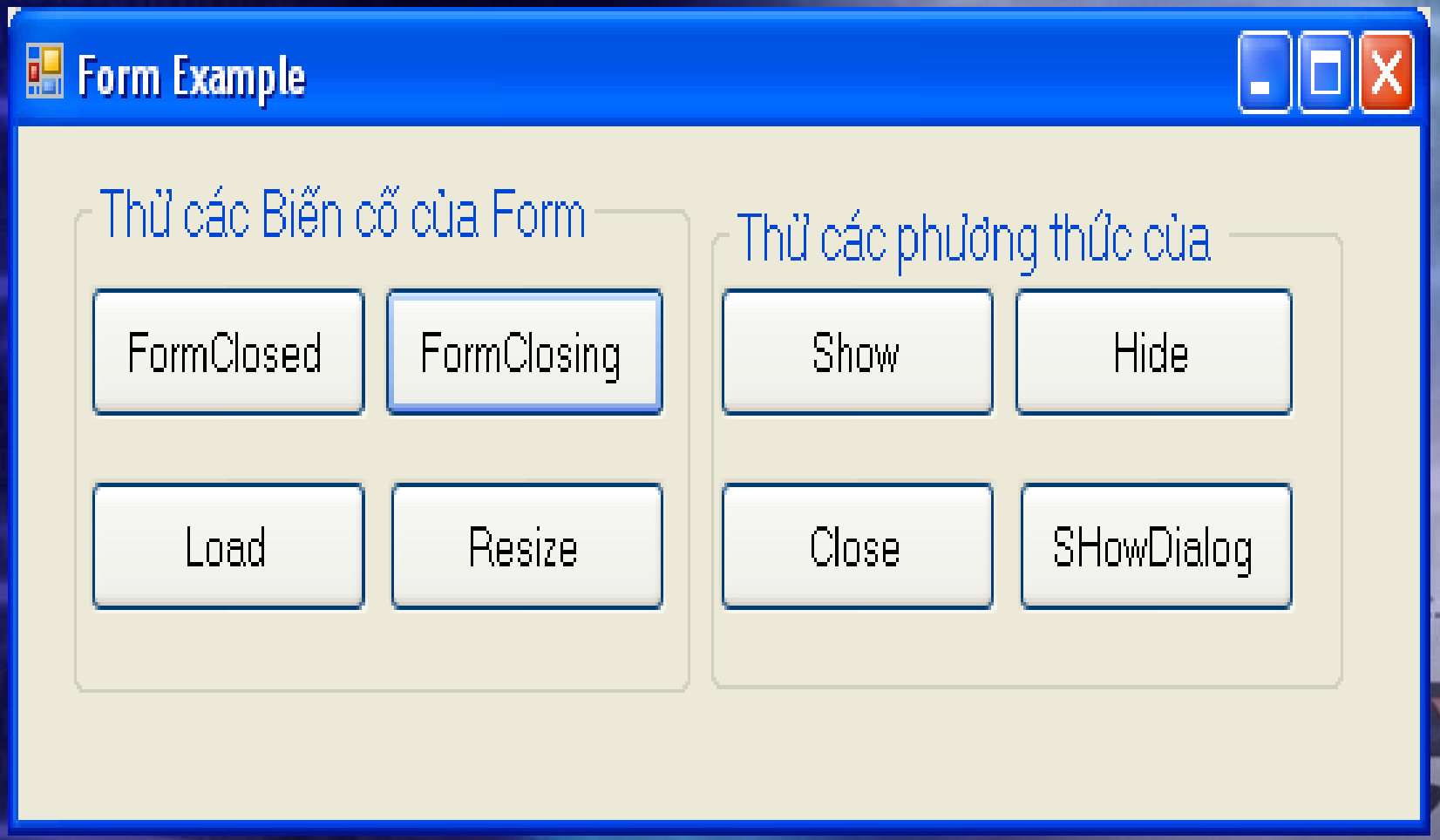

```
 private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
 MessageBox.Show("Are you sure  to exit?", "Thong bao",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);

        }
```

2.4. Phương thức của Form

- **Close():** Dùng để đóng Form
  - Vd: this.Close()
- **Hide():** Ẩn form
  - VD: this.hide
- **Show():** Nạp form
  - VD: Frm.Show()
- **ShowDialog():** Nạp Form dạng Modal
  - VD: frm.ShowDialog

Thực hành

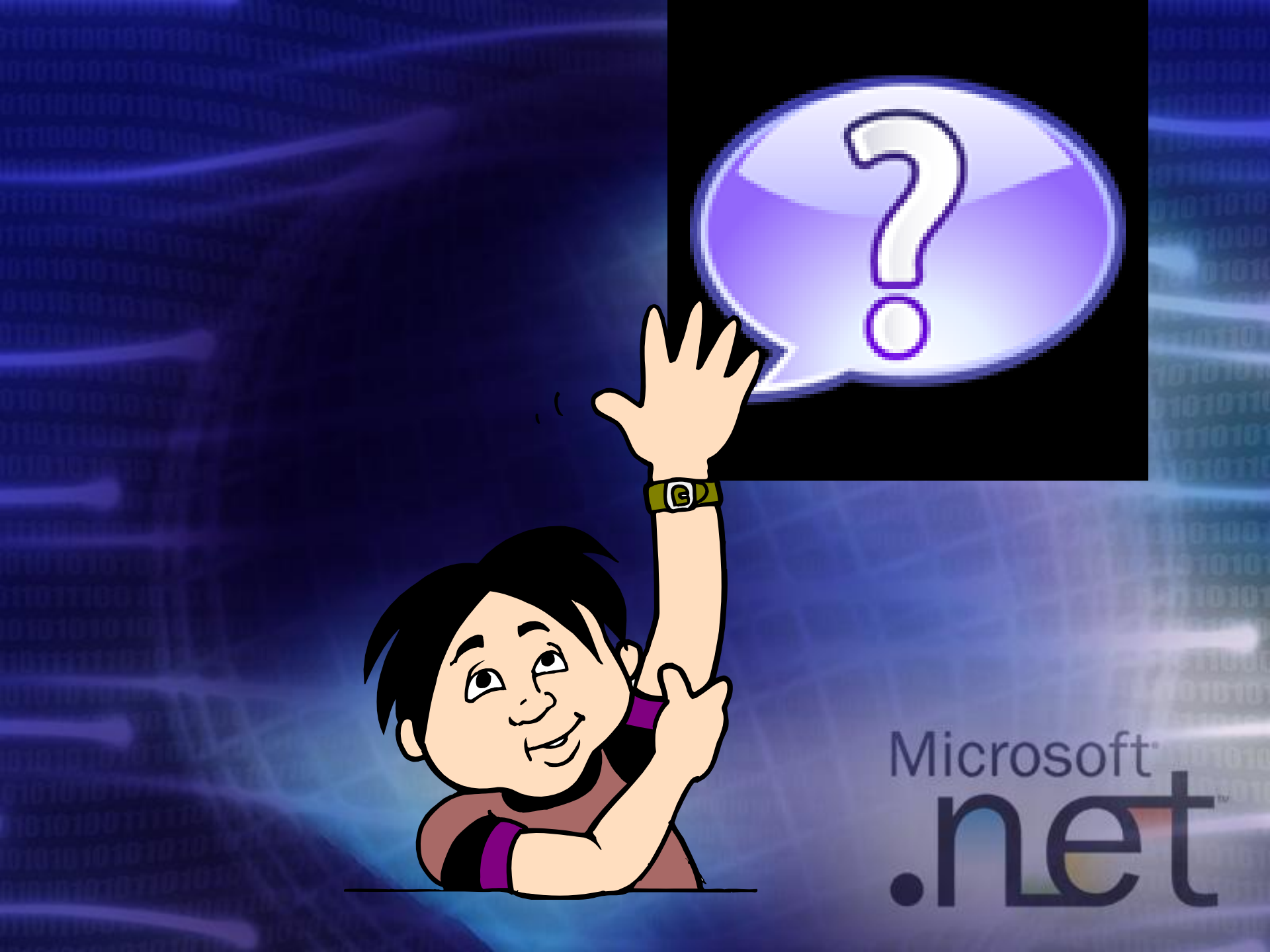
Thử các biến cố và phương thức của Form



Thực hành – Bài tập

- Viết chương trình giải PT bậc 2
- Yêu cầu: Thiết kế Form như sau:

Microsoft .net


Microsoft
.net

Chương 3.

Điều khiển thông thường

Thuộc tính chung của các điều khiển

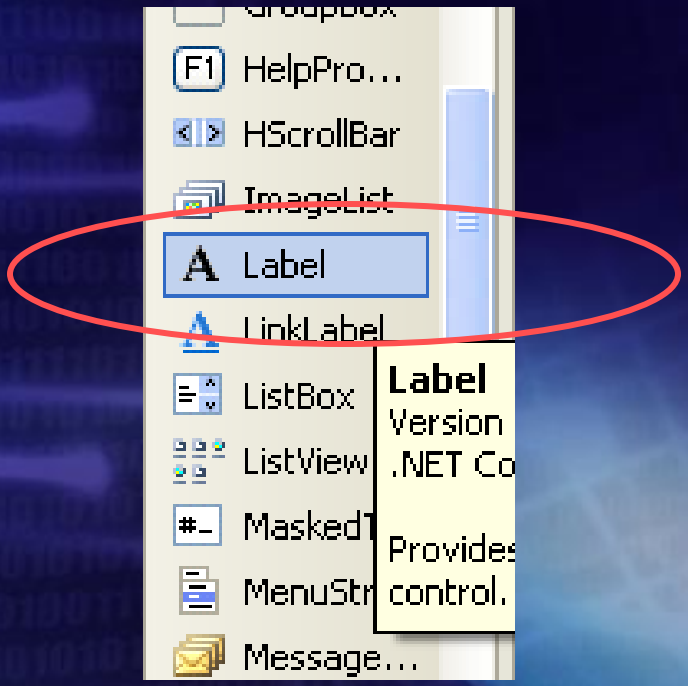
- **BackColor**: Màu nền của điều khiển
- **ForeColor**: Màu chữ của chuỗi trình bày trên điều khiển
- **Text**: Chuỗi trình bày trên điều khiển
- **Visible**: Thuộc tính che dấu hay hiển thị điều khiển
- **Name**: Tên của điều khiển
- **Looked**: Khoá không cho di chuyển trên Form

Sự kiện chung của các điều khiển

- **Click**: Xảy ra khi người dùng nhấn chuột phải
- **MouseMove**: Xảy ra khi người dùng di chuyển chuột qua vùng làm việc cảu điều khiển
- **MouseUp**: Nhấn chuột xuống vùng làm việc cảu điều khiển rồi thả ra
- **MouseDown**: Nhấn chuột xuống vùng làm việc cảu điều khiển
- **Move**: Xảy ra khi di chuyển điều khiển bằng mã hay bởi người sử dụng
- **REsize**: Xảy ra khi kích thước điều khiển được thay đổi bằng mã hay bởi người sử dụng

3.1. Điều khiển Label

- Trình bày thuộc tính dạng tiêu đề, chú giải cho các điều khiển khác (đã quen thuộc)



Microsoft .net

3.1. Điều khiển Label

- **BorderStyle**: Đường viền của điều khiển
- **Font**: Kích thước và Font chữ
- **TextAlign**: Căn chỉnh

Microsoft .net

3.1. Điều khiển Label

- Ví dụ

```
//Khai báo và khởi tạo đối tượng Label
Void CreatControls()
{
    Label lb=new Label();
    Lb.Text="This is Label Object";
    this.Controls.Add(lb);
}
```

3.2. Điều khiển TextBox

- Dùng để nhập dữ liệu
- Một số thuộc tính:
  - **BorderStyle**: Kiểu đường viền của điều khiển
  - **CharacterCasing**: Định dạng chuỗi nhập vào chuyển sang kiểu chữ hoa (Upper), chữ thường (Lower) hay mặc định (Normal)
  - **Enabled**: Vô hiệu hoá hay cho phép sử dụng
  - **Maxlength**: Số ký tự cho phép nhập
  - **MultiLine** : Giá trị True cho phép nhập nhiều dòng
  - **PasswordChar**: Giá trị nhập được thay thế bởi ký tự khai báo trong thuộc tính này (Multiline=False)

3.2. Điều khiển TextBox

- **Một số thuộc tính:**
  - **ReadOnly**: =True chỉ cho phép đọc giá trị
  - **ScrollBars**: Nếu thuộc tính MultiLine=true thì cho phép hiện thanh trượt hay không (Vertical - Cuộn dọc, Horizontal - Cuộn ngang, both - Cả 2 thanh cuộn, none – Không có thanh cuộn)
  - **WordWrap**: Tự động xuống dòng nếu chuỗi giá trị dài hơn kích thước của điều khiển

3.2. Điều khiển TextBox

- **Một số biến cố**
  - **MouseClick:** Xảy ra khi Click vào Textbox
  - **MouseDoubleClick:** Xảy ra khi Click đúp vào Textbox
  - **TextChanged:** Xảy ra khi chuỗi trên điều khiển thay đổi

3.3. Điều khiển Button

- Cho phép người dùng chuột để nhấn, phím Enter hay phím Spacebar nếu điều khiển này đang được kích hoạt
- Các thuộc tính, biến cố (giống VB6.0)
  - Lưu ý: thuộc tính Caption trong VB ←→ thuộc tính Text trong C#

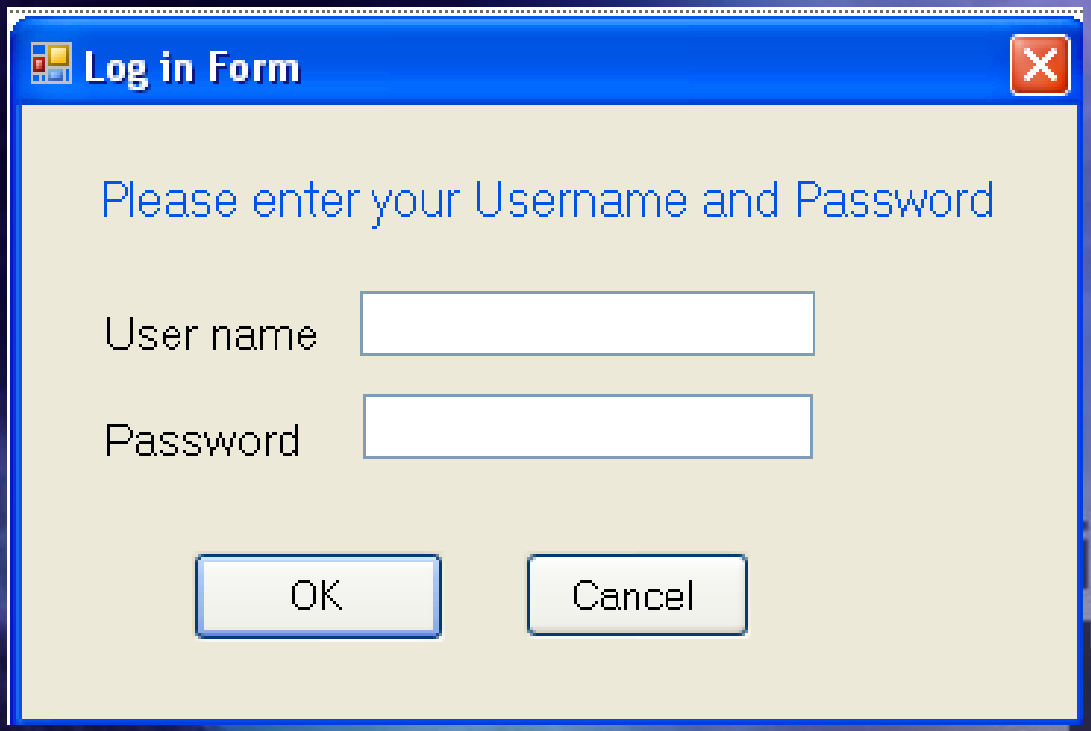
3.3. Điều khiển Button

- Khai báo và khởi tạo đối tượng Button sau đó thêm vào Form

```
Button btn=new Button();
btn.Name="btnSave";
btn.Text="&Save";
this.Controls.Add (btn);
```

Ví dụ 1

- Tạo Form đăng nhập hệ thống như sau:



Ví dụ 1

- Yêu cầu:
  - Nếu Username khác rỗng → Nút OK được kích hoạt
  - Không nhập Password mà nhấn OK→ có thông báo yêu cầu nhập Password
  - Nhập sai Uername, Password → Thông báo nhập sai, không cho đăng nhập hệ thống
  - Nhập Username=“admin” và Password = “123456” → có thông báo đăng nhập thành công và hiện Form chính của chương trình

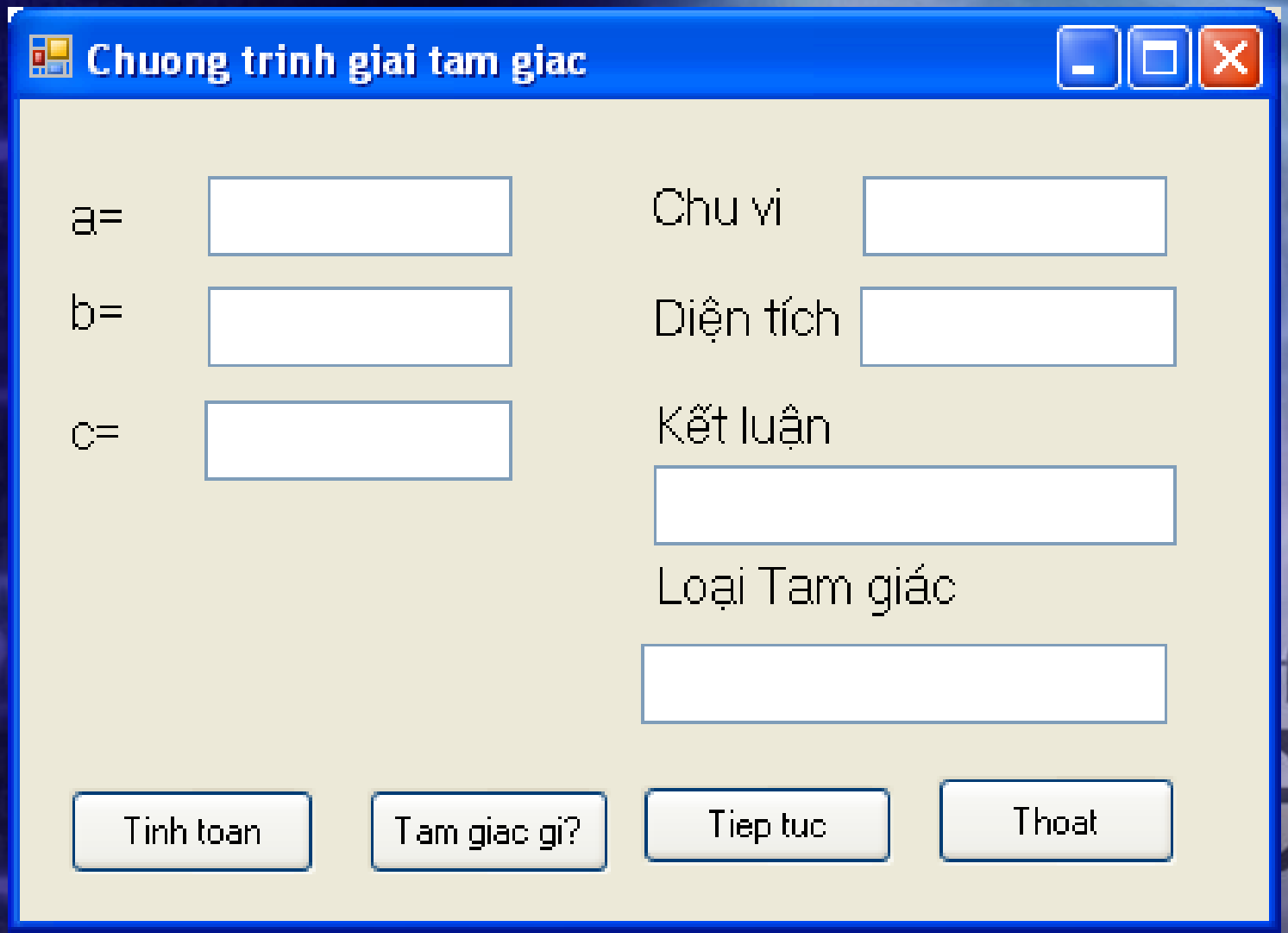
Ví dụ 2

- Viết chương trình nhập 3 số a, b, c vào 3 textbox và kiểm tra 3 số có là 3 cạnh tam giác hay không? Nếu là 3 cạnh tam giác thì tính diện tích, chu vi tam giác đó và kiểm tra xem đó là tam giác gì?

Microsoft .net

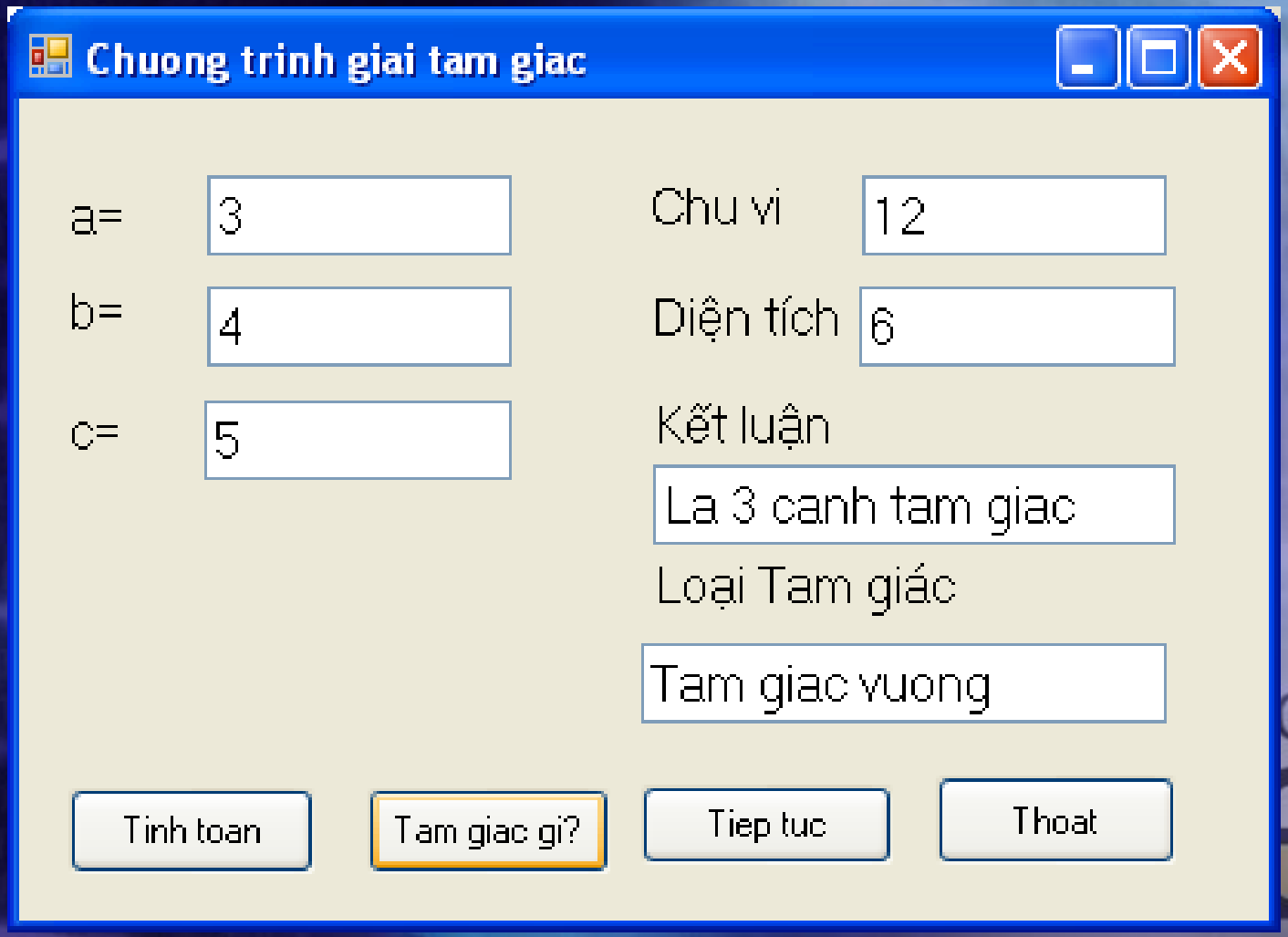
Ví dụ 2

Thiết kế Form như sau:



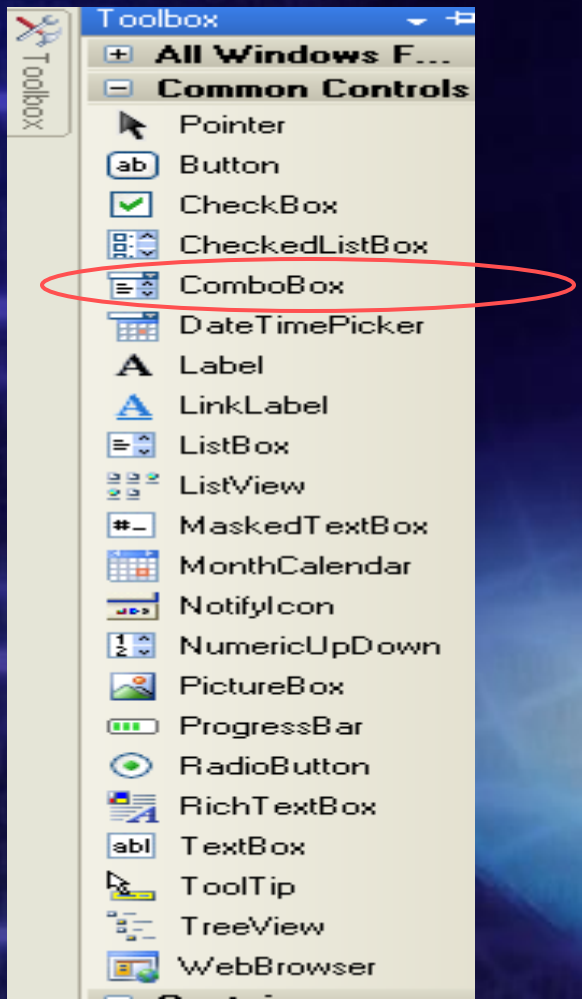
Ví dụ 2

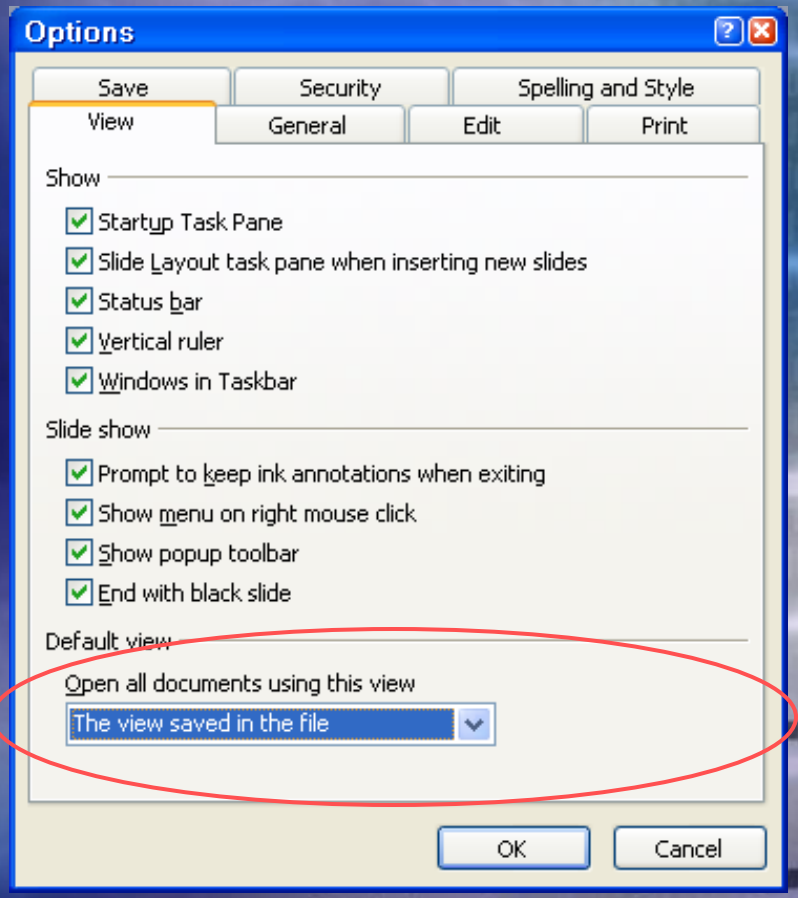
Kết quả thực hiện chương trình



3.3. Nhóm điều khiển ComboBox, ListBox

- **ComboBox:** Giống VB





3.3. Nhóm điều khiển ComboBox, ListBox

- **ComboBox – Một số thuộc tính**
  - **DataSource:** Tập dữ liệu điền vào điều khiển
  - **Items**: Tập các phần tử có trong điều khiển, có thể sử dụng phương thức Add và AddRange để thêm phần tử vào ComboBox

Microsoft .net

3.3. Nhóm điều khiển ComboBox, ListBox

- **ComboBox:** Ví dụ Thêm các mục vào ComboBox1 bằng phương thức Add

```
private void button1_Click(object sender, EventArgs e)
    {
        for (int i = 1; i < 10; i++ )
            comboBox1.Items.Add("Phan tu " + i.ToString());
    }
```

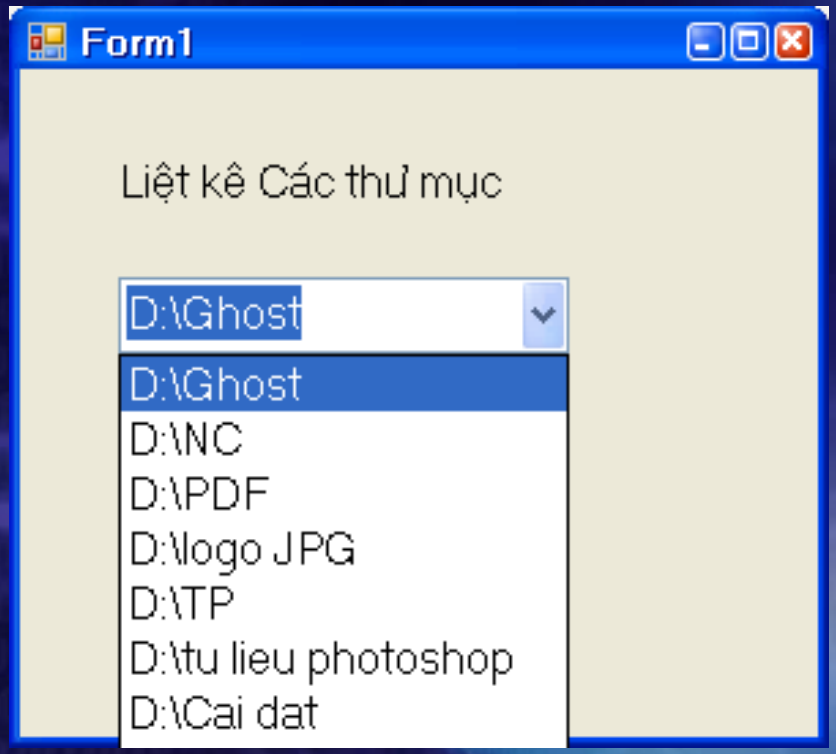
3.3. Nhóm điều khiển ComboBox, ListBox

- **ComboBox: Ví dụ Thêm các mục vào ComboBox1 bằng phương thức AddRange**

```
private void button2_Click(object sender, EventArgs e)
    {
        string[] week = new string[7]
        { “Sun", “Mon", “Tue", "Wed", "Thu", "Fri", "Sat" };
        comboBox2.Items.AddRange(week);
    }
```

3.3. Nhóm điều khiển ComboBox, ListBox

- **ComboBox:** Ví dụ liệt kê các thư mục. Sử dụng phương thức DataSource



Microsoft .net

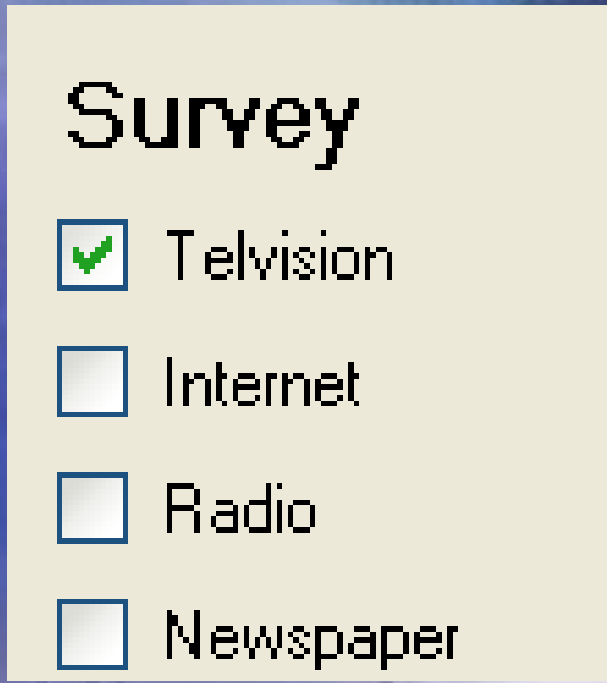
3.3. Nhóm điều khiển ComboBox, ListBox

- **ListBox:** Giống VB
  - Các thuộc tính và phương thức: Tương tự COmboBOx

➔ SV tự tìm hiểu

3.4. Nhóm điều khiển CheckBox, RadioButton

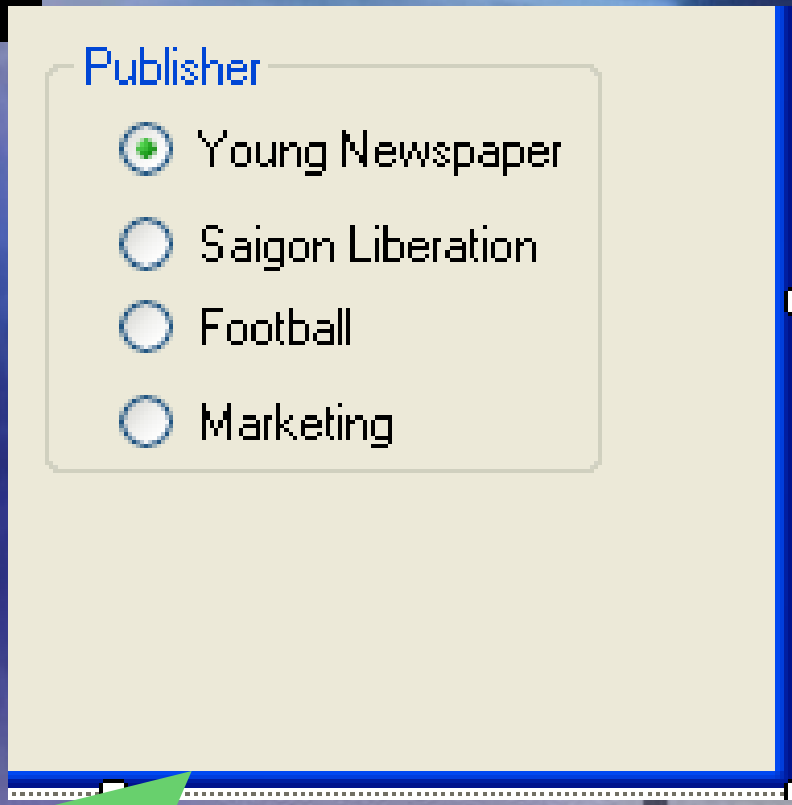
- **CheckBox:** Giống VB
- Một số thuộc tính đáng chú ý:
  - **Checked**: Trạng thái chọn (true), không chọn (False)
  - **CheckState**: Trạng thái của điều khiển **CHeckBox** đang chọn, có 3 trạng thái: Checked, Unchecked, Indeterminate.



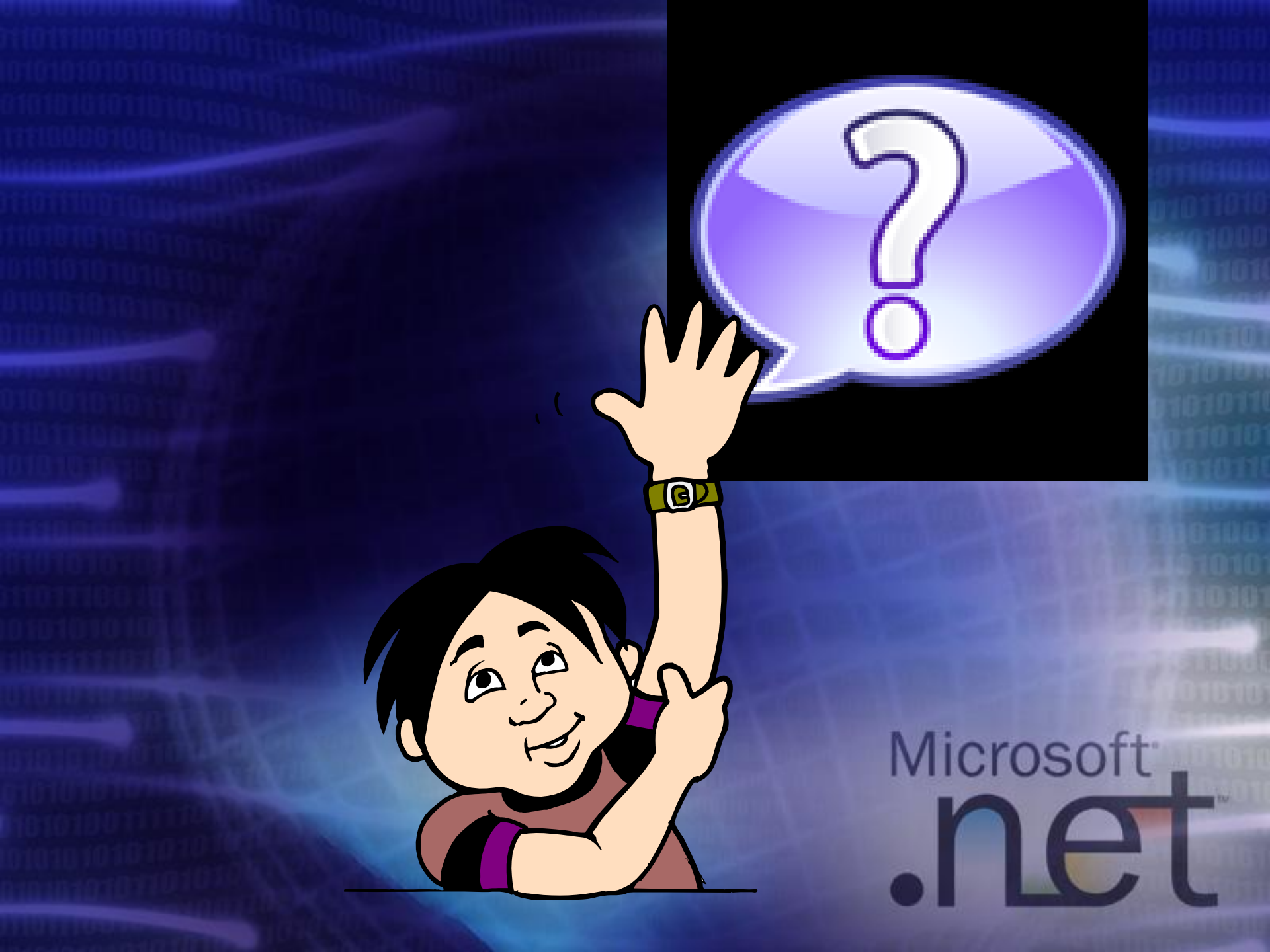
SV tự tìm hiểu

3.4. Nhóm điều khiển CheckBox, RadioButton

- **RadioButton:** Giống VB



SV tự tìm hiểu


Microsoft
.net

Chương 4.

ĐIỀU KHIỂN ĐẶC BIỆT

Microsoft .net

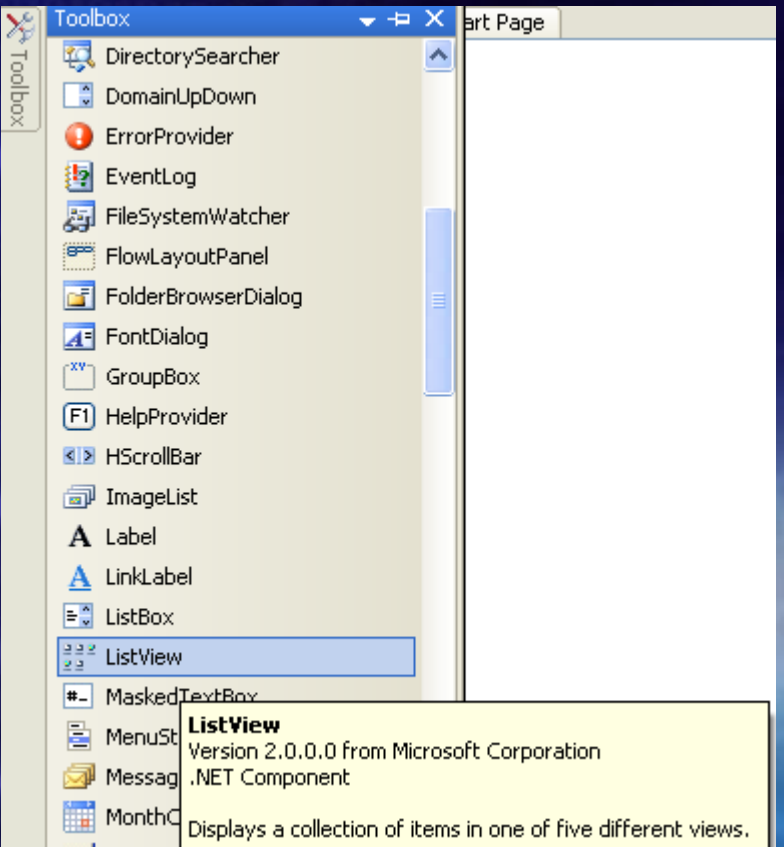
4.1. Điều khiển ImageList

- Chứa mảng các Picture, thường sử dụng với Listview, Treeview
- Giống VB 6.0
- Ví dụ:

Microsoft .net

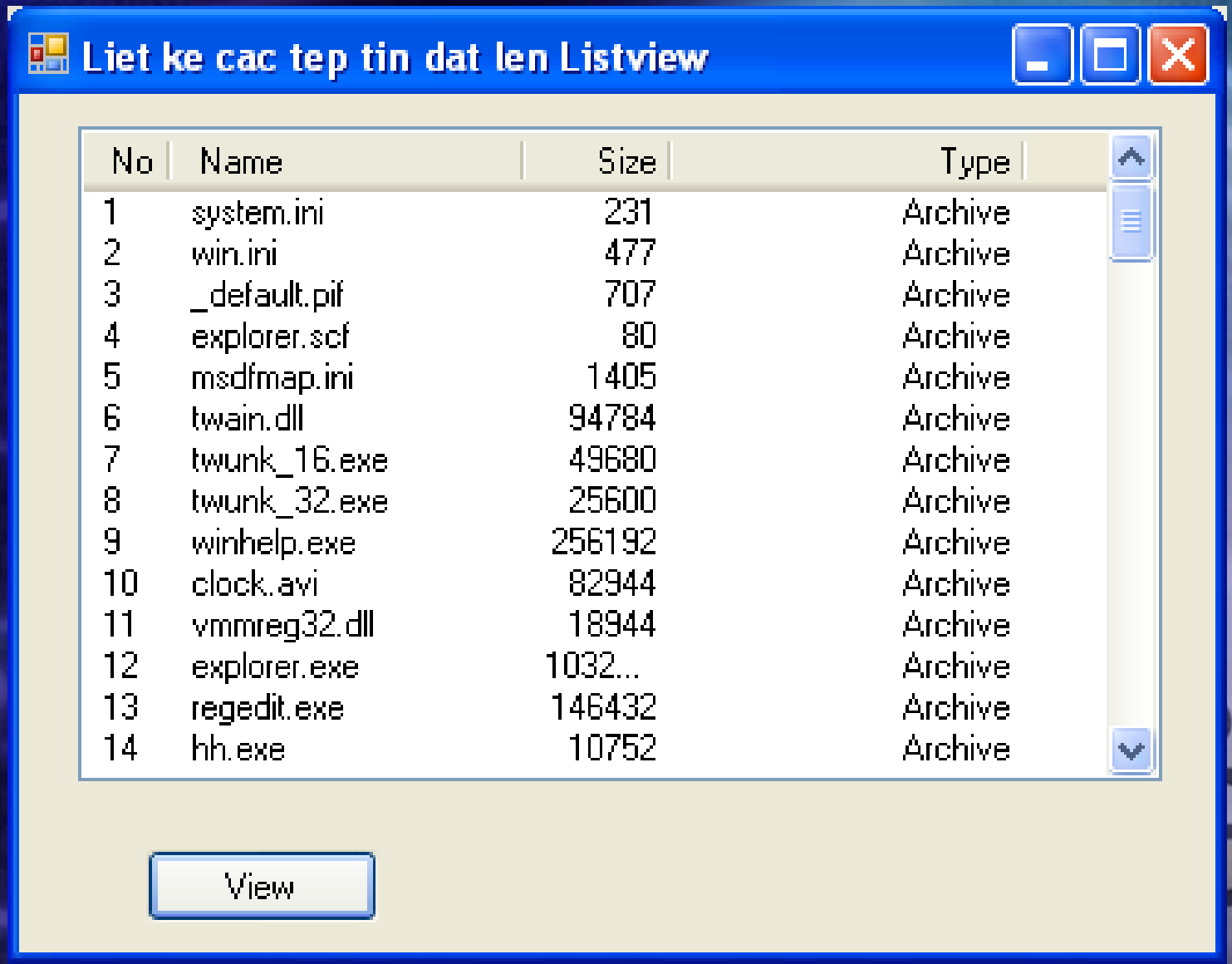
4.2 Điều khiển ListView

- Trình bày các phần tử dạng danh sách với nhiều hình dạng khác nhau.

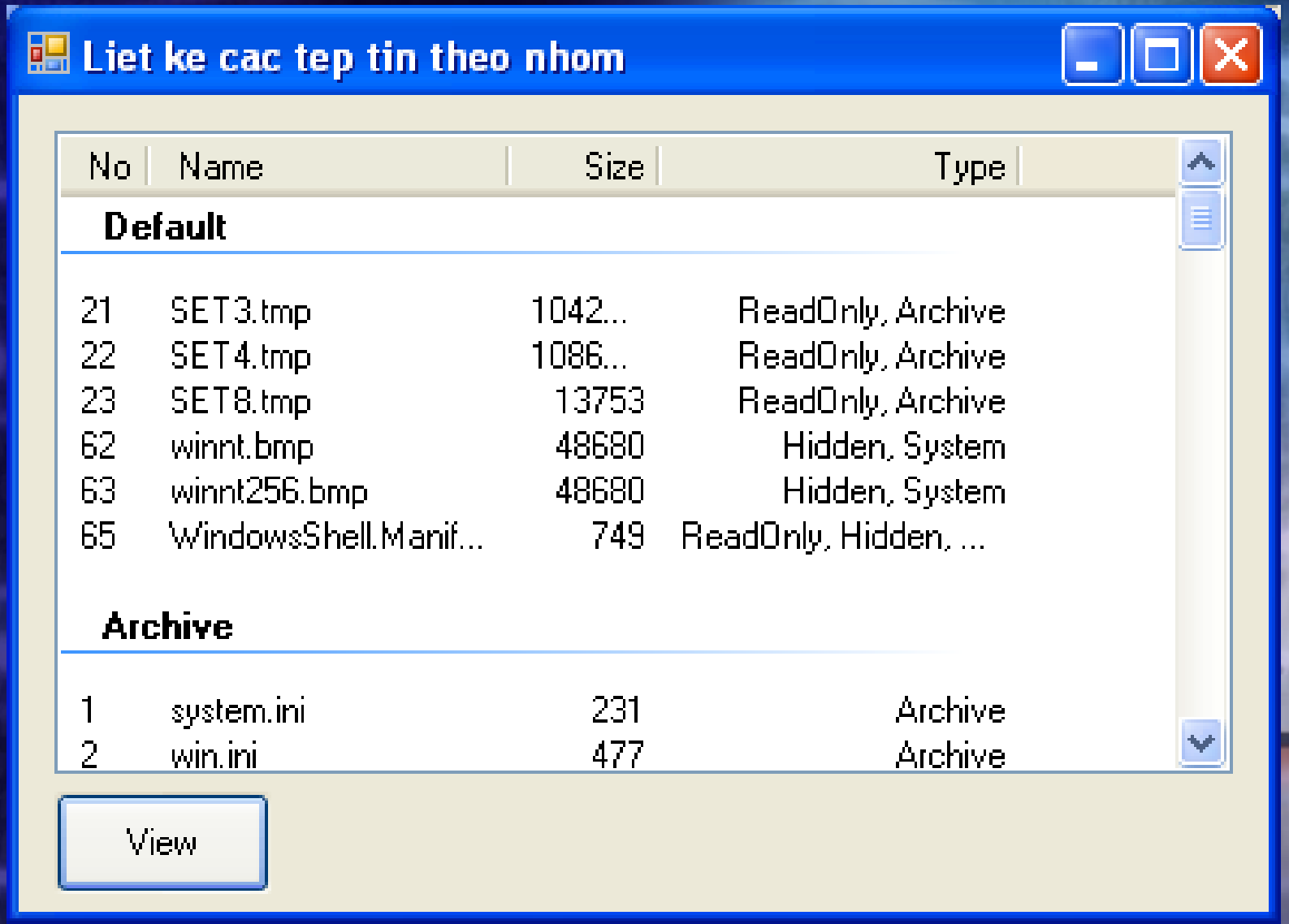


Microsoft .net

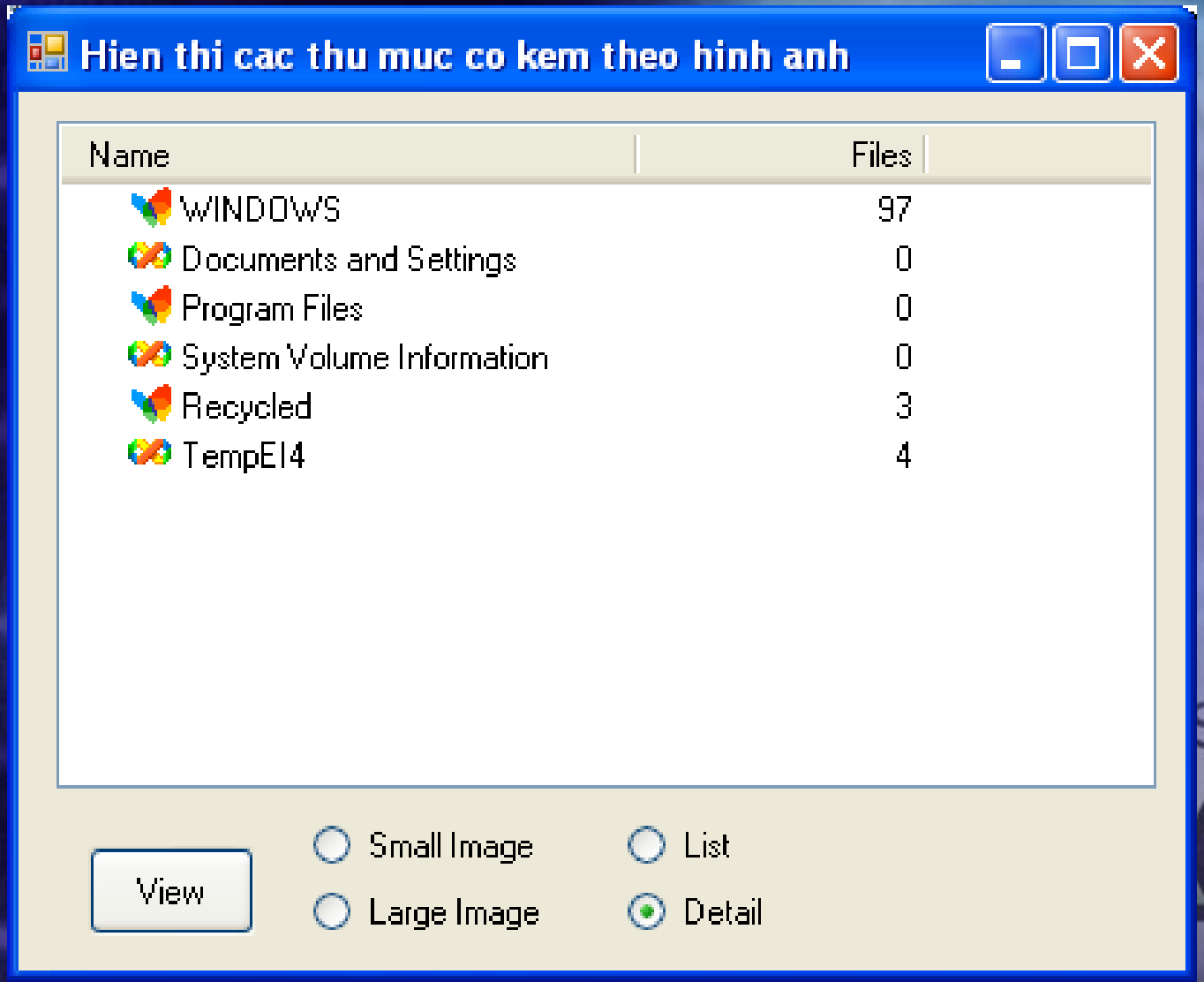
4.2. Điều khiển ListView



4.2 Điều khiển ListView



4.2. Điều khiển ListView

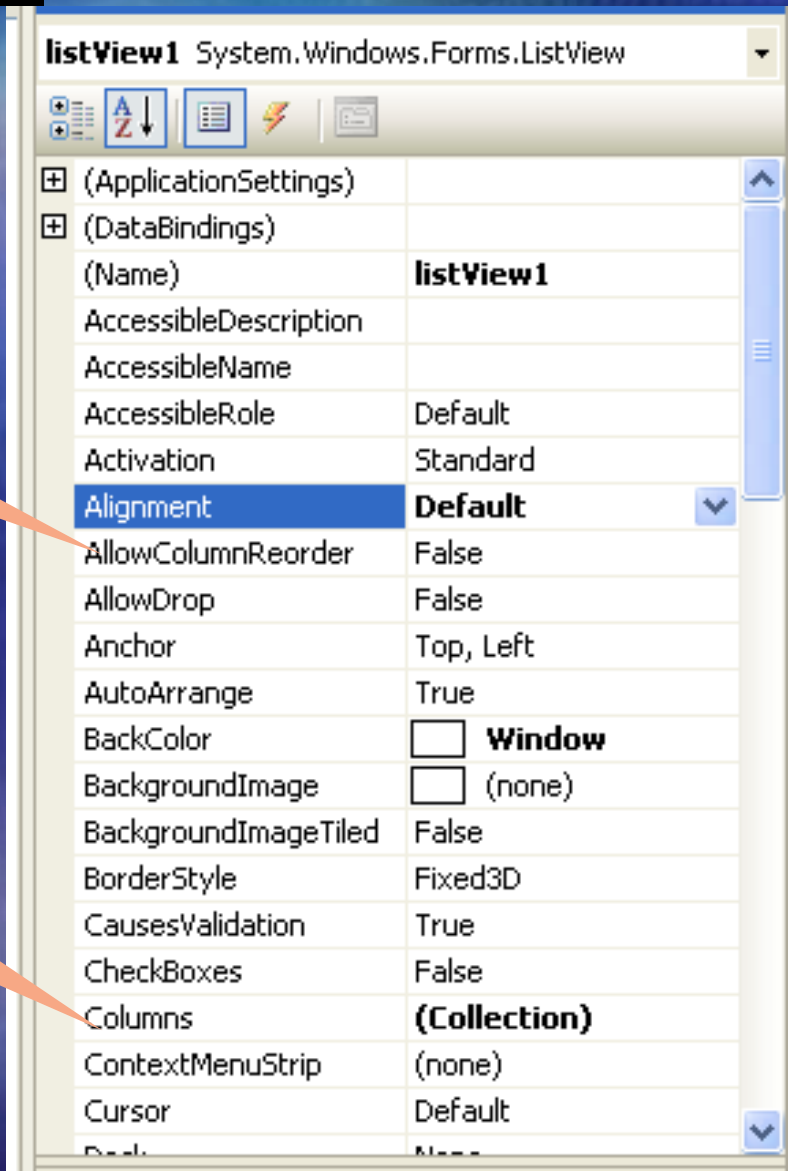


4.2. Điều khiển ListView

Một số thuộc tính cơ bản

Cho phép sắp xếp cột trên điều khiển Listview ở chế độ thi hành

Khai báo số cột (có Header) của điều khiển Listview



4.2. Điều khiển ListView

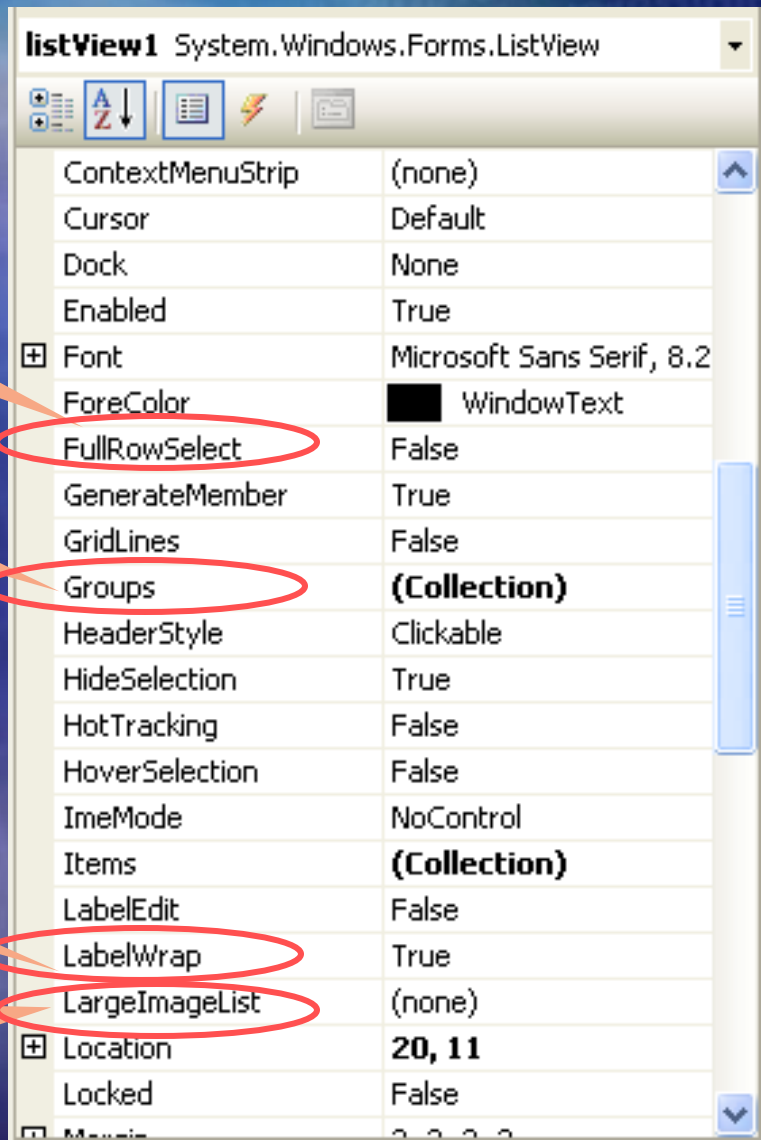
Một số thuộc tính cơ bản

=True: Cho phép tô màu ứng với hàng của phần tử được chọn

Khai báo nhóm để phân loại các phần tử sau khi trình bày trên điều khiển Listview

=True: Chuỗi sẽ tự động xuống dòng khi chiều dài không đủ để trình bày

Đối tượng ImageList chứa danh sách các Image theo số chỉ mục từ 0 đến n-1 được sử dụng cho trường hợp thuộc tính View là LargeIcon

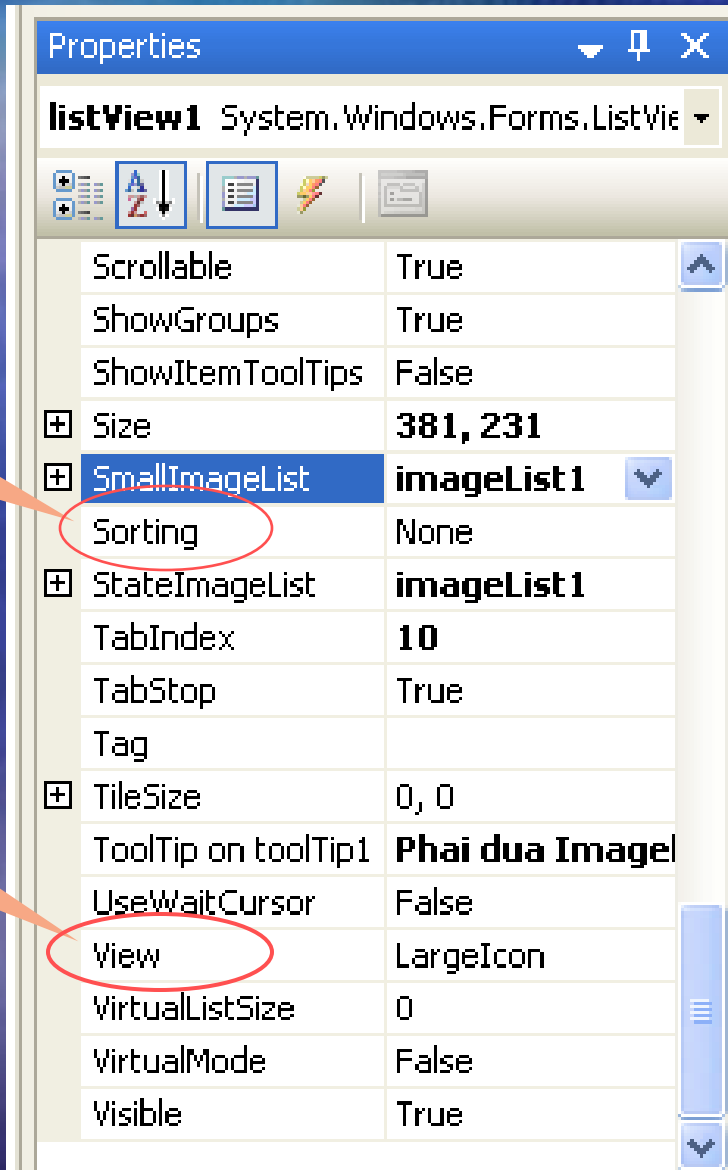


4.2. Điều khiển ListView

Một số thuộc tính cơ bản

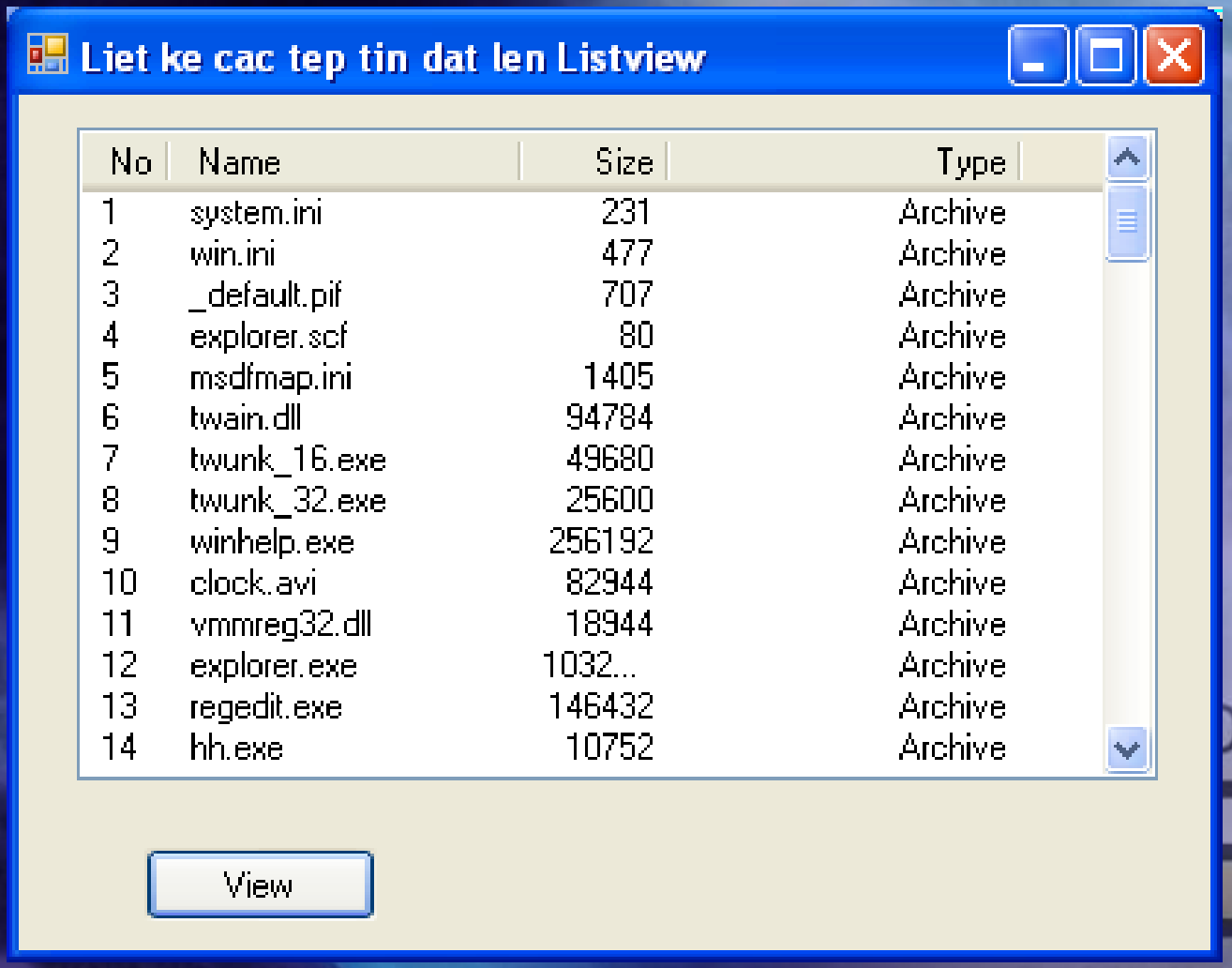
Các phần tử trên List view sẽ được sắp xếp tăng dần (Asccending), giảm dần (Descending) hoặc không sắp (None)

Chế độ trình bày tương ứng trên điều khiển như: List, Details, LargeIcon, SmallIcon, Title.



4.2. Điều khiển ListView

Ví dụ: Liệt kê danh sách các tệp tin



4.2. Điều khiển ListView
Ví dụ: Liệt kê danh sách các tệp tin

- **Chú ý khi viết Code**
  - Khai báo: **using System.IO;**
  - Khai báo sử dụng đối tượng DirectoryInfo để lấy thông tin của thư mục:

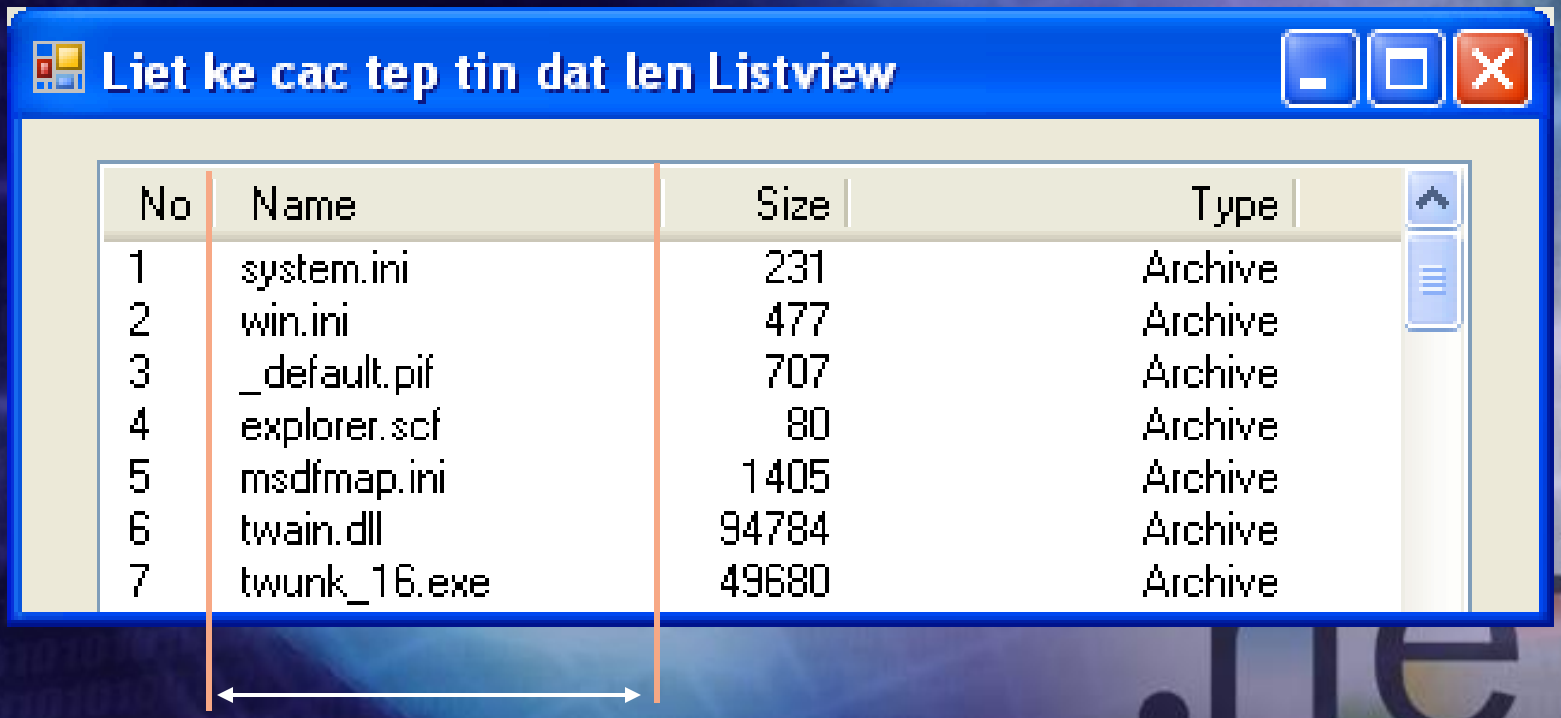
    DirectoryInfo dir = new DirectoryInfo("C:\\Windows\");
  - **dir.GetFiles("*.*")**: Lấy ra danh sách các File trong thư mục "dir"
  - **FileInfo f**: Khai báo đối tượng f chứa thông tin về các tệp tin
    - **f.Name**: Tên tệp tin
    - **f.Length**: Dung lượng tệp tin (byte)
    - **f.Attributes**: Thuộc tính của tệp tin
    - **f.CreationTime**: Ngày giờ tạp ra tệp tin

4.2. Điều khiển ListView

Ví dụ: Liệt kê danh sách các tệp tin

- **Chú ý khi viết Code**
  - **Khai báo cột trên Listview**

```
this.listView1.Columns.Add("Name",200,  HorizontalAlignment.Left);
```



4.2. Điều khiển ListView

Ví du: Liêt kê danh sách các tệp tin

- **Chú ý khi viết Code**
  - Chế độ hiển thị

```
listView1.View = View.Details;
```

  - Thêm các tệp tin vào List view1

```
ListViewItem item1;  // Khai báo Item1 thuộc đối tượng ListViewItem
    foreach (FileInfo f in dir.GetFiles("*.*"))  // Lấy thông tin của tệp tin
        {                                        // đưa vào Listview1
            i++;
            item1 = new ListViewItem(i.ToString());
            item1.SubItems.Add(f.Name);
            item1.SubItems.Add(f.Length.ToString());
            item1.SubItems.Add( f.Attributes.ToString());
            listView1.Items.Add(item1);
        }
```

4.2. Điều khiển ListView

Bài tập

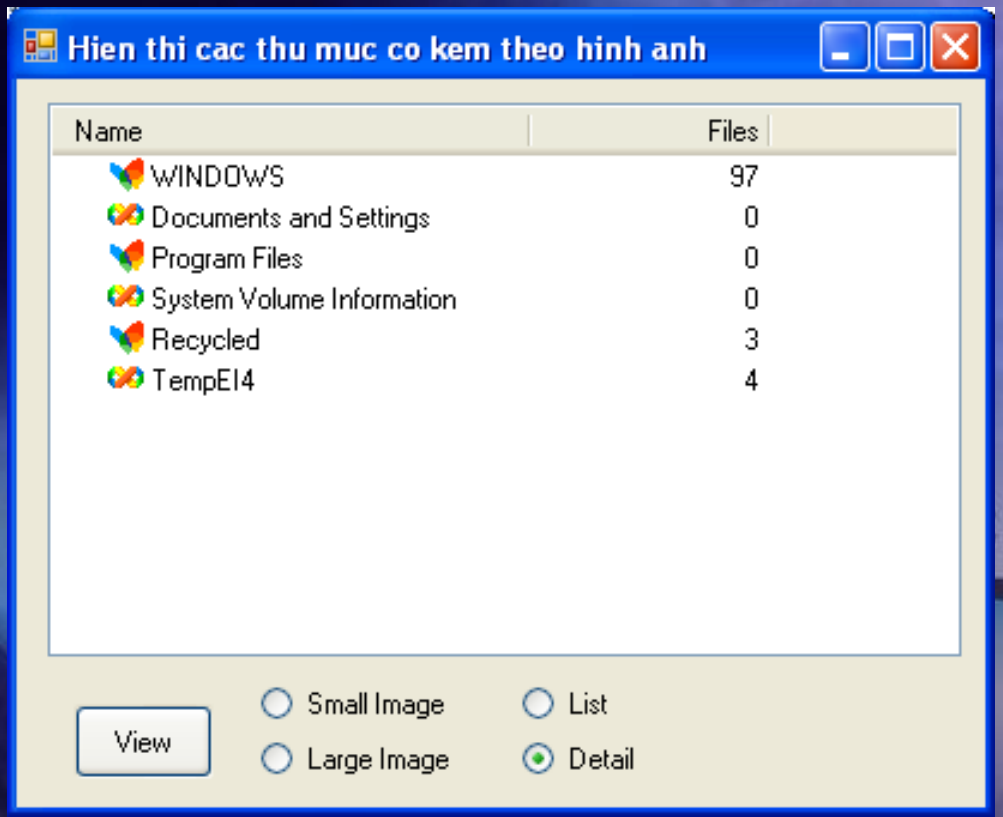
SV tạo Listview để chứa danh sách các tệp tin lấy từ ổ đĩa D, tương tự như ví dụ trên

Microsoft .net

4.2. Điều khiển ListView

Ví dụ 2

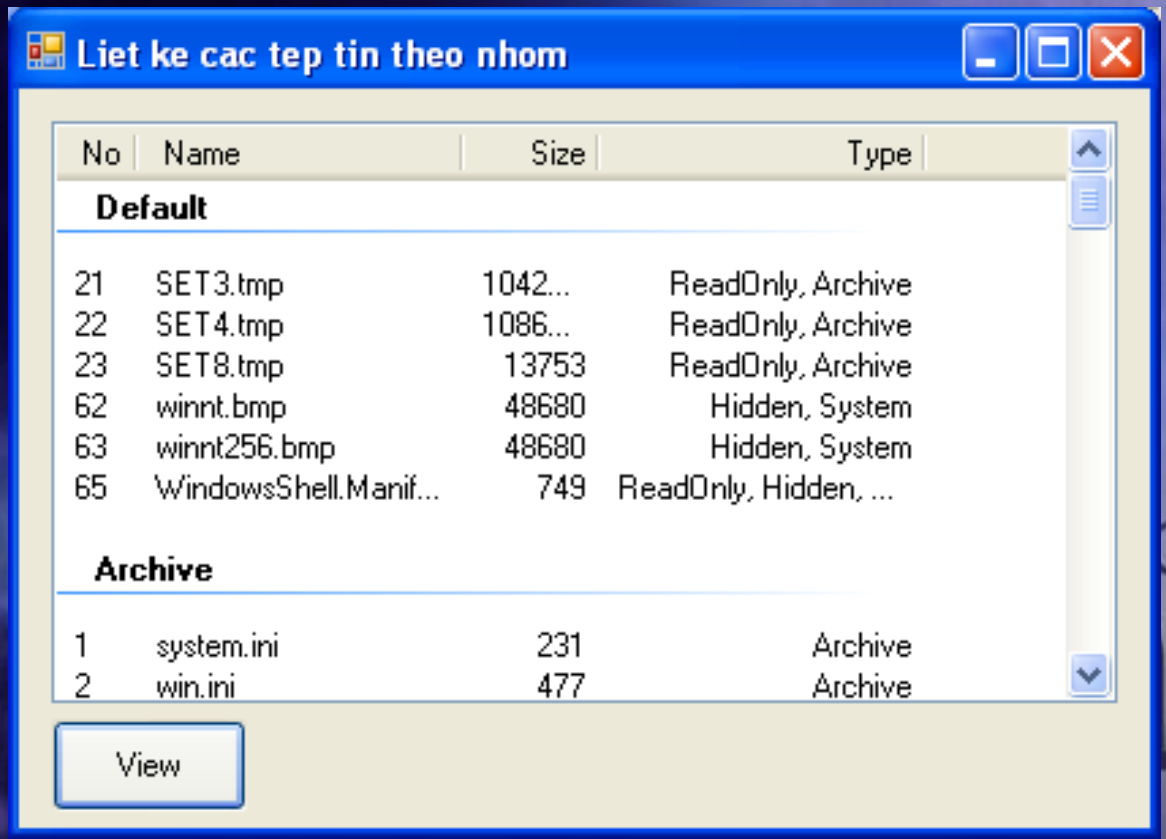
Tạo List view liệt kê các thư mục con, có chứa hình ảnh như sau:



4.2. Điều khiển ListView

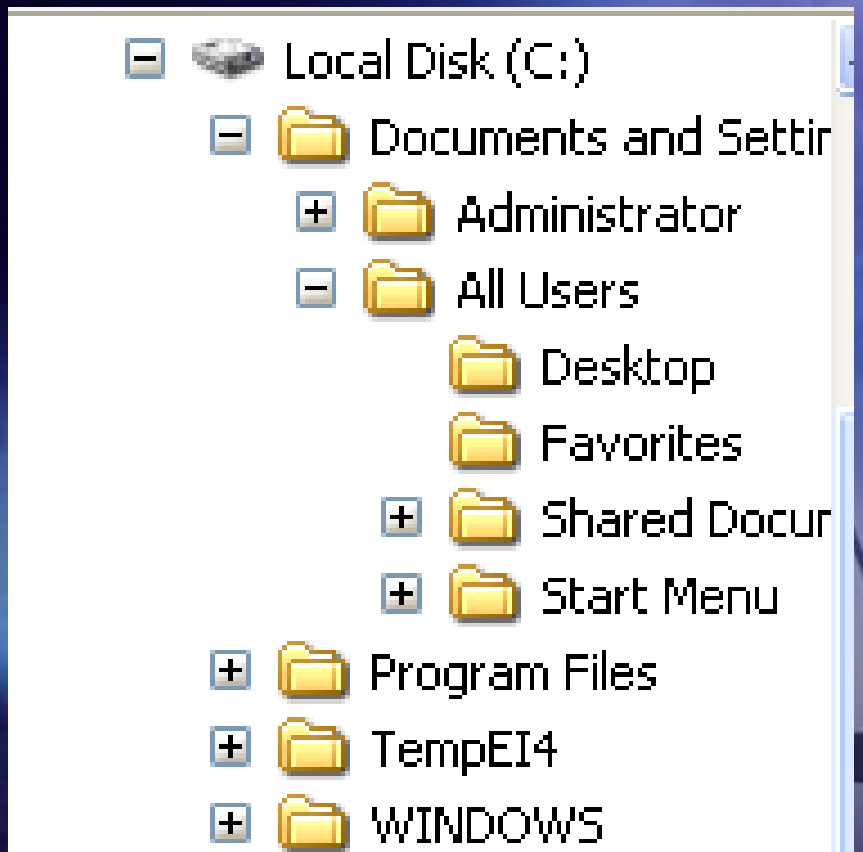
Ví dụ 3

Tạo List view liệt kê các thư mục con theo 4 nhóm (Archieve, System, Normal, Default) như sau:



4.3. Điều khiển TreeView

- Trình bày danh sách phần tử phân cấp theo từng nút (Giống Windows Explorer của Windows)



4.3. Điều khiển TreeView

Một số thuộc tính

- **CheckBoxes**: Xuất hiện Checkbox bên cạnh từng nút của Treeview
- **Nodes**: Khai báo số Node (có header) của Listview
- **FullRowSelect**: là true – cho phép tô màu ứng với hàng của phần tử được chọn, giá trị mặc định là False
- **ShowLine**: Cho phép có đường viền ứng với từng nút, mặc định là True
- **LabelEdit**: là true nếu cho phép thay đổi chuỗi của mỗi nút

4.3. Điều khiển TreeView

Một số thuộc tính

- **ShowPlusMinus**: là true thì có biểu tượng dấu + và - xuất hiện trên mỗi nút
- **ShowRootLine**: Chọn giá trị true nếu cho trình bày nút gốc
- **ImageList:** Chỉ ra đối tượng ImageList được đưa vào làm ảnh trên các nút của Treeview theo thứ tự chỉ mục từ 0 đến n-1 (giả sử ImageList có n ảnh)

Microsoft .net

4.3. Điều khiển TreeView

Một số Phương thức

- **CollapseAll**: Trình bày tất cả các nút trên Treeview
- **ExpandAll**: Thu gọn tất cả các nút trên Treeview

Microsoft .net

4.3. Điều khiển TreeView

Thêm nút vào Treeview

this.Treeview1.Nodes.Add(……..)

7 hàm nạp chồng

```
int TreeNodeCollection.Add(TreeNode node)  (+ 6 overload(s))
Adds a previously created tree node to the end of the tree node collection.

Exceptions:
   System.ArgumentException
```

Microsoft .net

4.3. Điều khiển TreeView

1 of 7 TreeNode TreeNodeCollection.Add (**string text**)

text: The label text displayed by the System.Windows.Forms.TreeNode.

```
this.Treeview1.Nodes.Add("My Computer")
```

2 of 7 int TreeNodeCollection.Add (**TreeNode node**)

node: The System.Windows.Forms.TreeNode to add to the collection.

```
this.Treeview1.Nodes[level1].Nodes.Add("Computer")
```

3 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text)

key: The name of the tree node.

```
this.Treeview1.Nodes.Add("Root","My Computer")
```

4.3. Điều khiển TreeView

4 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text, int imageIndex)
key: The name of the tree node.

```
this.Treeview1.Nodes.Add("Root","My Computer",1)
```

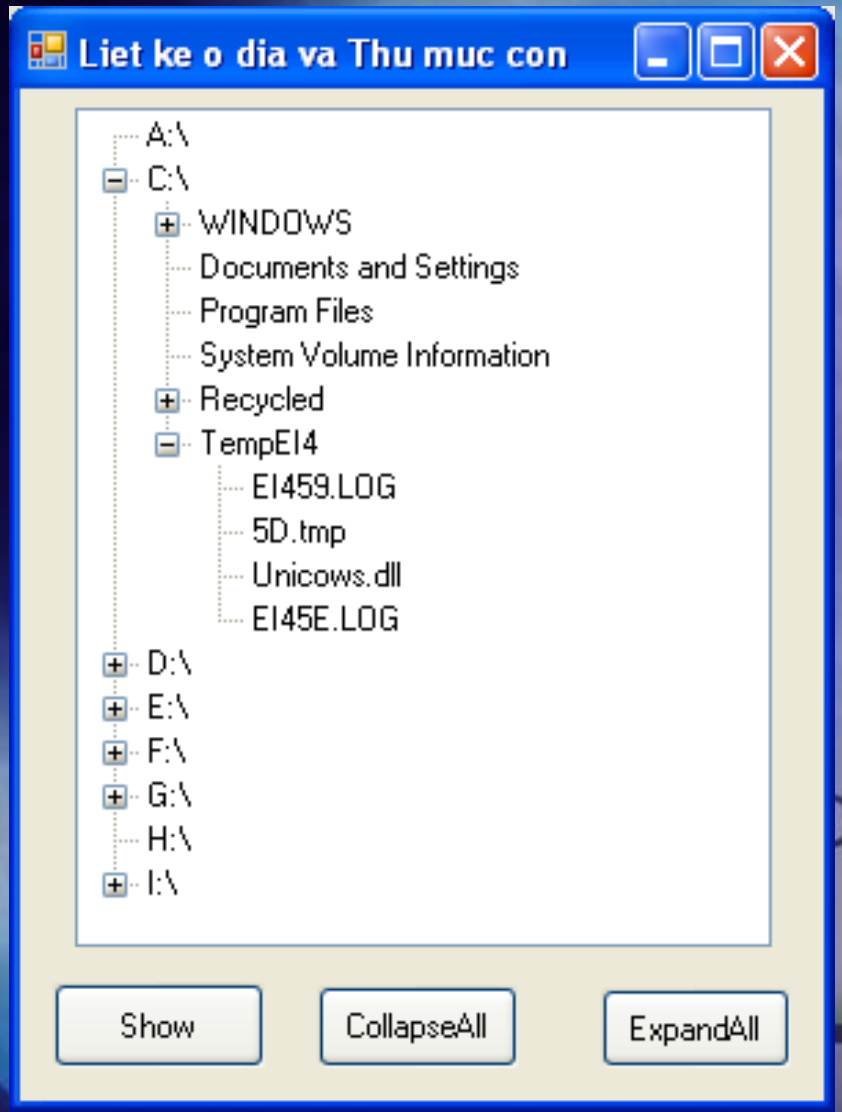
5 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text, string imageKey)
key: The name of the tree node.

```
this.Treeview1.Nodes.Add("Root","My Computer",
    "C:\\Picture\\computer1.ico")
```

Microsoft .net

4.3. Điều khiển TreeView

Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa



4.3. Điều khiển TreeView

Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- **Chú ý khi viết Code**
  - Khai báo: **using System.IO;**
  - Khai báo sử dụng đối tượng **Directory**
    - **Directory.GetLogicalDrives():** Lấy ds cách ổ đĩa logic
    - **Directory.GetDirectories(F):** Lấy danh sách các thư mục con của thư mục F
    - **Directory.GetFile(F):** Lấy danh sách các tệp tin của thư mục F

4.3. Điều khiển TreeView

Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- **Chú ý khi viết Code**
  - **Thêm nút vào TreeView như sau:**

    this.Treeview1.Nodes.Add(TreeNode node)

VD:

this.Treeview1.Nodes.Add(“Root,”My Computer”,1)

4.3. Điều khiển TreeView

Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- Liệt kê các ổ Logic đặt lên Treevie
- Nút Show gọi hàm GetDisk()

```
void GetDisk()
    {
        foreach (string d in Directory.GetLogicalDrives())
        {
            this.treeView1.Nodes.Add(d);
        }
```

4.3. Điều khiển TreeView

- Liệt kê các Thư mục đặt lên Treeview

```
void GetFolder(string name, int level)
    {
        try
        {
          foreach (string d in Directory.GetDirectories(name))
           {
              this.treeView1.Nodes[level].Nodes.Add(d.Substring(3));
          }   //Cắt đi 3 ký tự đầu tiên VD: C:\TP\Bin thì còn TP\Bin
        }
        catch (Exception ex)
        {
           MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning);
        }
    }
```

4.3. Điều khiển TreeView

- Để liệt kê tất cả các thư mục trên các ổ đĩa, ta sửa lại hàm GetDisk như sau:

```
void GetDisk()
      {
       int i = 0;
          foreach (string d in Directory.GetLogicalDrives())
          {
             this.treeView1.Nodes.Add(d);
           GetFolder(d, i);
            i++;
          }
      }
```

4.3. Điều khiển TreeView

- Liệt kê các File có trong 1 thư mục đặt lên Treeview

```
void GetFile(string name, int level, int level1)
    {
        try
        {
            foreach (string d in Directory.GetFiles(name))
            {
                    this.treeView1.Nodes[level].Nodes[level1].
                    Nodes.Add(d.Substring(name.Length + 1));
            }
        }
        catch (Exception ex)
        {
MessageBox.Show(ex.Message,"Error",MessageBoxButtons.AbortRetryIgnore,
MessageBoxIcon.Warning);
        }
    }
```

4.3. Điều khiển TreeView

- Để Liệt kê các File, các thư mục con của các ổ Logic đặt lên Treeview ta viết lại GetFolder như sau:

```
void GetFolder(string name, int level)
    {
        try
        {   int level1 = 0;
            foreach (string d in Directory.GetDirectories(name))
            {
                this.treeView1.Nodes[level].Nodes.Add(d.Substring(3));
                GetFile(d, level, level1);                level1++;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning);
        }
    }
```

4.3. Điều khiển TreeView

- Viết Code cho Nút CollapseAl và ExpandAll

```
private void button2_Click(object sender, EventArgs e)
    {
        treeView1.CollapseAll();
    }

  private void button3_Click(object sender, EventArgs e)
    {
        treeView1.ExpandAll();
    }
```

4.3. Điều khiển TreeView

- Bài tập

SV làm lại ví dụ trên

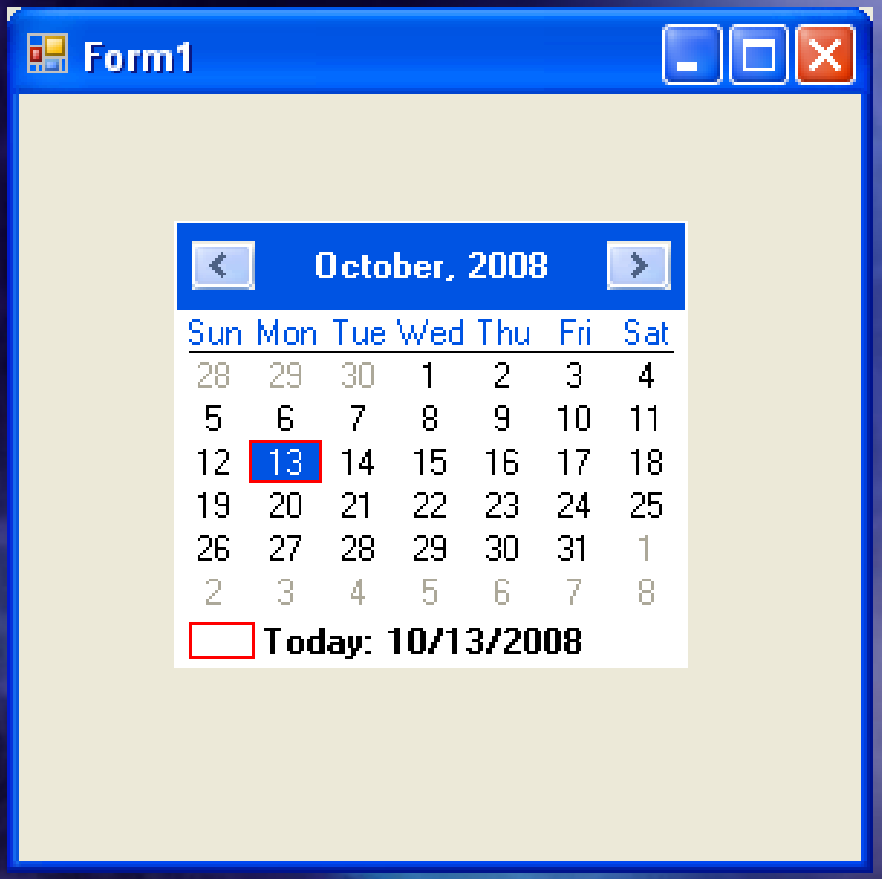
4.4. Điều khiển DateTimePicker

- Giống VB 6.0

Microsoft .net

4.5. Điều khiển MonthCalendar

- Giống VB 6.0

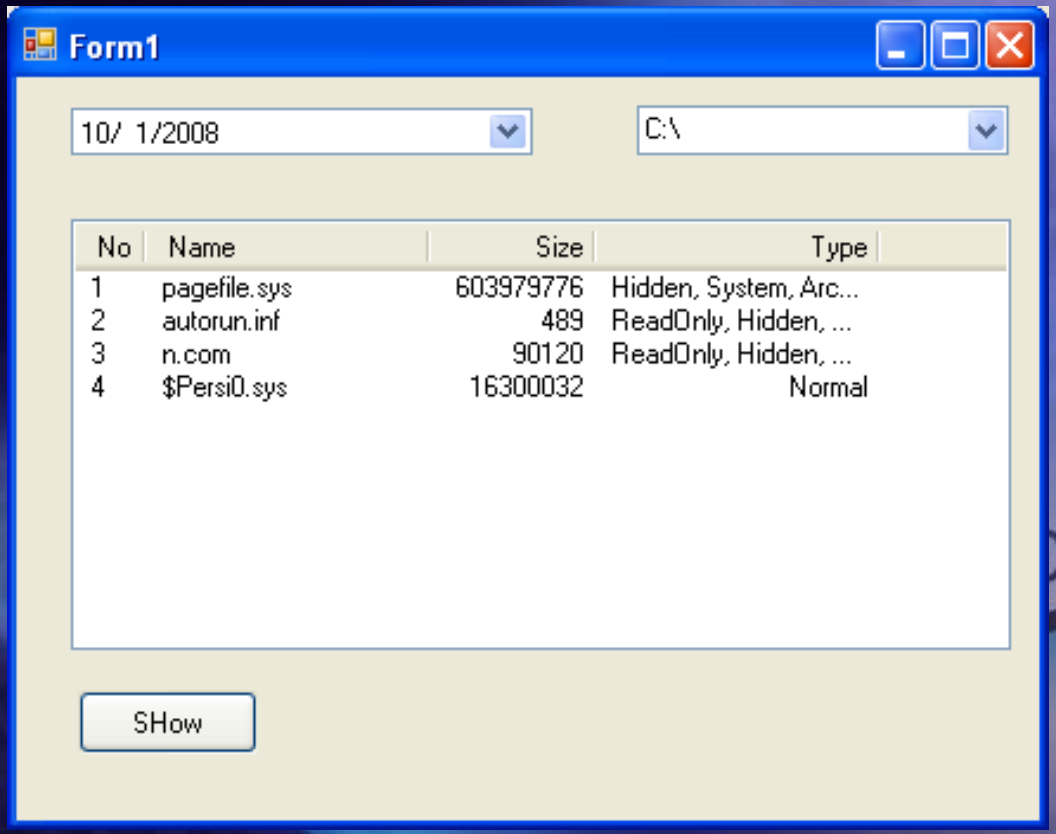


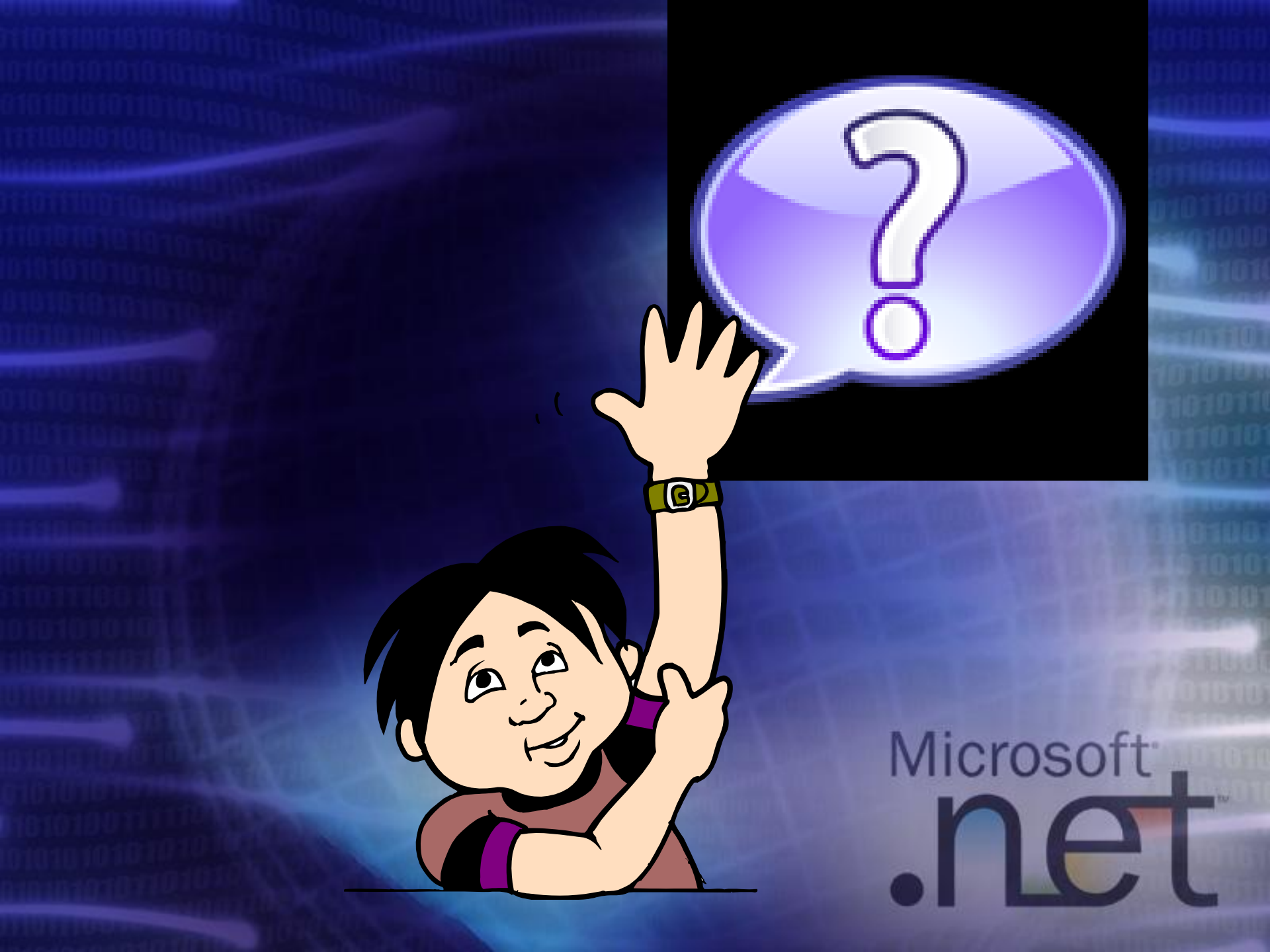
Microsoft .net

4.5. Điều khiển MonthCalendar

Bài tập

- Liệt kê các tệp tin được tạo ra trước ngày chỉ ra trong Combobox1 trong ổ đĩa (Chỉ ra trong ComboBox2)




Microsoft
.net

Chương 5.
ĐIỀU KHIỂN DÙNG ĐỂ XÂY DỰNG MENU

Microsoft .net

5.1. Điều khiển ImageList

Microsoft .net

5.2. Điều khiển MenuStrip

Microsoft .net

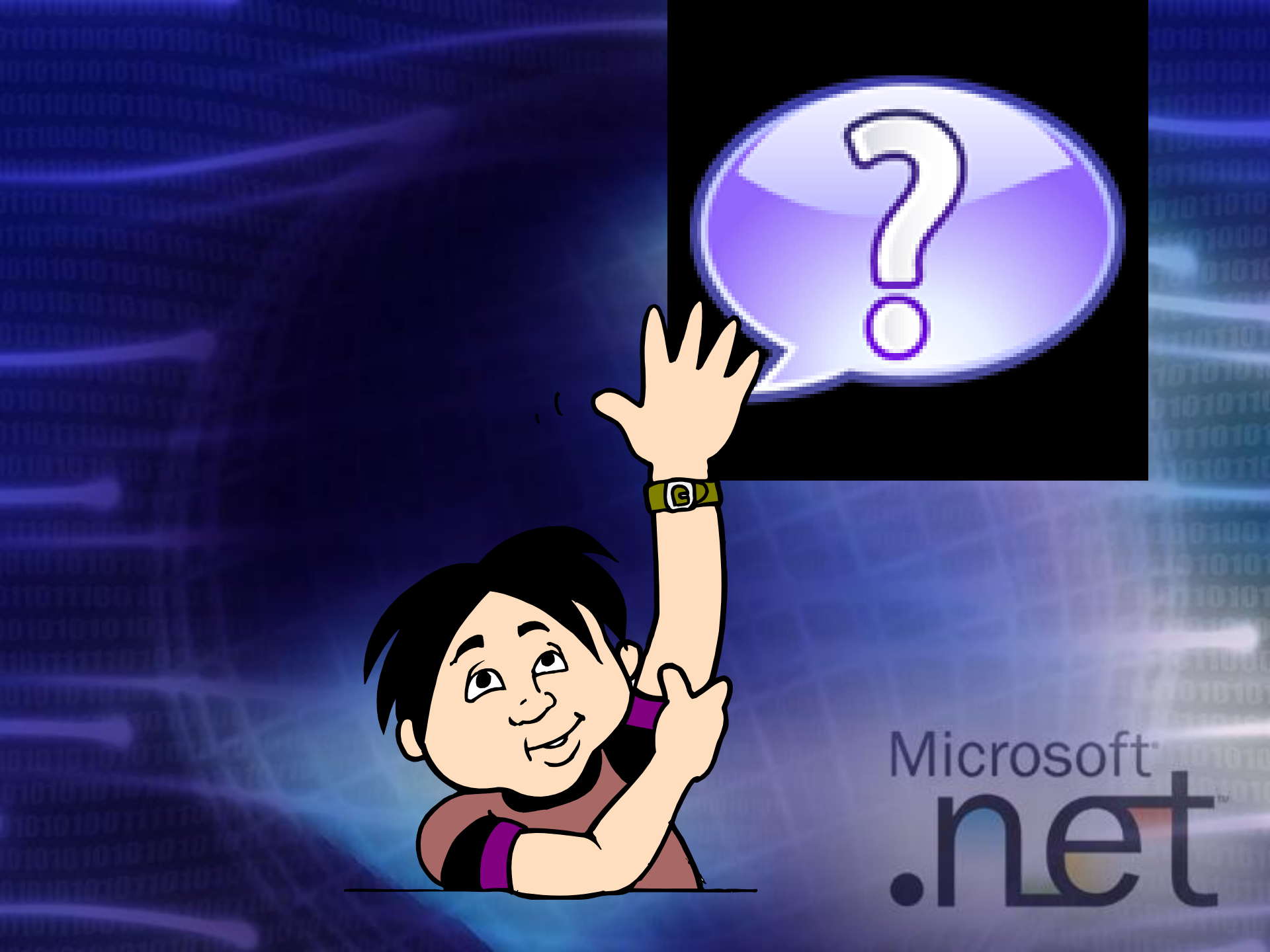
5.3. Điều khiển ConTextMenuStrip

5.4. Điều khiển NotifyIcon

5.5. điều khiển StatusStrip

5.6. Điều khiển ToolStrip

Microsoft .net


Microsoft
.net

Chương 6.
ĐIỀU KHIỂN CHỨA ĐỰNG CÁC ĐIỀU KHIỂN KHÁC

Microsoft .net

6.1. Điều khiển GroupBox

Microsoft .net

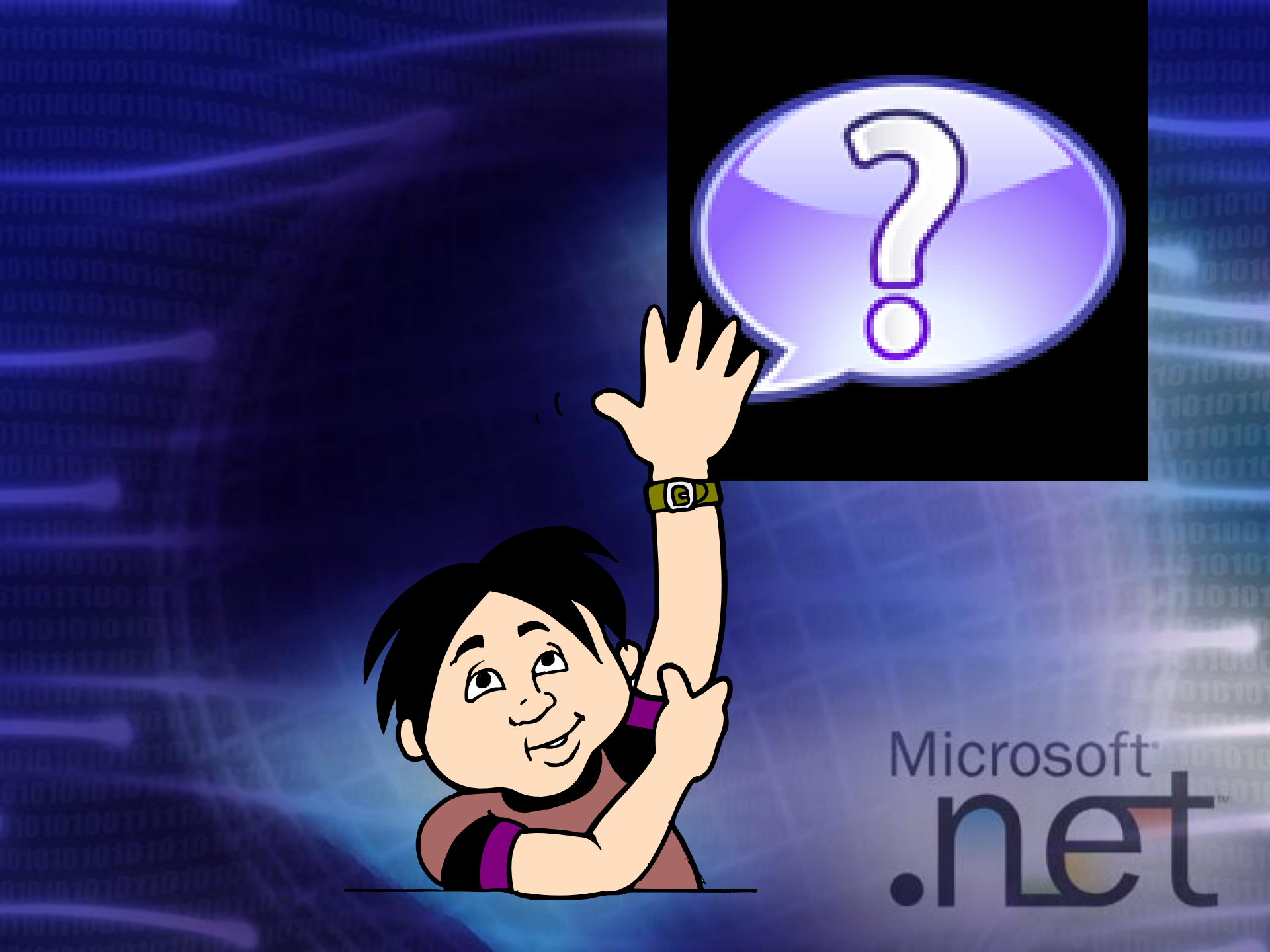
6.2. Điều khiển TabControl

6.3. Điều khiển Panel

Microsoft .net

6.4. Điều khiển FlowLayoutPanel

6.5. Điều khiển TabLayoutPanel


Microsoft
.net

Chương 7.

ĐIỀU KHIỂN DIALOG VÀ PHƯƠNG THỨC MESSAGEBOX

Microsoft .net

7.1. Lớp MessageBox

Microsoft .net

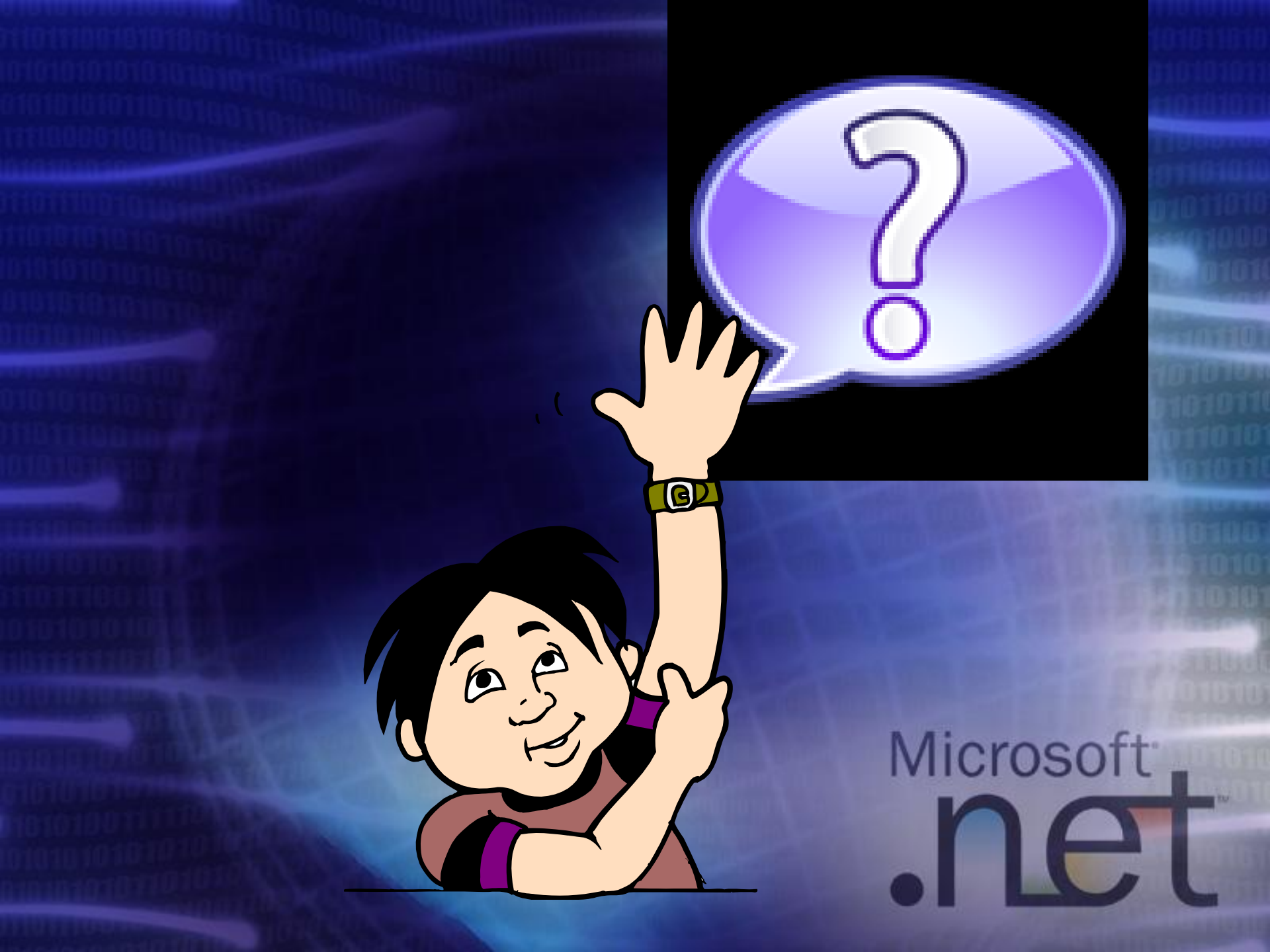
7.2. Điều khiển ColorDiaglog

7.3. Điều khiển FontDialog

7.4. Điều khiển OpenFileDialog

7.5. Điều khiển SaveFileDialog

7.6. Điều khiển FolderBrowserDialog.


Microsoft
.net

Chương 8.

LÀM VIỆC VỚI ĐIỀU KHIỂN IN ẤN

Microsoft .net

8.1. Điều khiển PageSetupDialog

8.2. Điều khiển PrintPreviewDialog

Microsoft .net

8.3. Điều khiển PrintPreviewControl

Microsoft .net

8.4. Điều khiển PrintDialog

8.5. Điều khiển PrintDocument.

Microsoft .net

Trường Đại học Kỹ Thuật Công nghệ
Khoa Công nghệ Thông tin

GIÁO TRÌNH MÔN HỌC

LẬP TRÌNH WINDOWS

VỚI VC/MFC

Biên soạn: Nguyễn Chánh Thành

Tháng 03 năm 2006

TÀI LIỆU THAM KHẢO

- Sách:
  - Các sách tiếng Việt về Visual C++ /lập trình Windows (của SAMIS, của nhóm tác giả ELICOM, hay của các tác giả khác)
  - Sách tiếng Anh:
    - Beginning Visual C++ 6
    - Professional Visual C++ 6 (của nhà xuất bản WROX)
  - Các eBook tiếng Anh về Visual C++ hay lập trình Windows như:
    - Programming Microsoft C++, 5th Edition eBook (của Microsoft Press)
    - Programming Windows with MFC, 2nd Edition eBook (của Microsoft Press)
- Chương trình tham khảo:
  - MSDN (bộ đĩa CD tài liệu tham khảo của Mircosoft)
  - Source code mẫu ở website:
    - http://www.wrox.com
  - Các ví dụ đặc biệt ở website:
    - http://www.codeguru.com
    - http://www.codeproject.com

CHƯƠNG 0. ÔN TẬP LÝ THUYẾT C/C++

0.1 Ôn tập C

0.1.1 Kiểu dữ liệu, biến và chuyển đổi kiểu

0.2 Hàm và lời gọi hàm

0.2.1 Phát biểu điều khiển

0.2.2 Array

0.2.3 Pointer

0.2.4 File

0.2.5 Debug – bẫy lỗi

0.3 Ôn tập C++

0.3.1 Class

0.3.2 Cấu trúc thừa kế

0.3.3 Tầm vực truy xuất

0.3.4 Object

CHƯƠNG 1. CÁC VẤN ĐỀ CƠ BẢN CỦA ỨNG DỤNG WINDOWS VÀ MFC

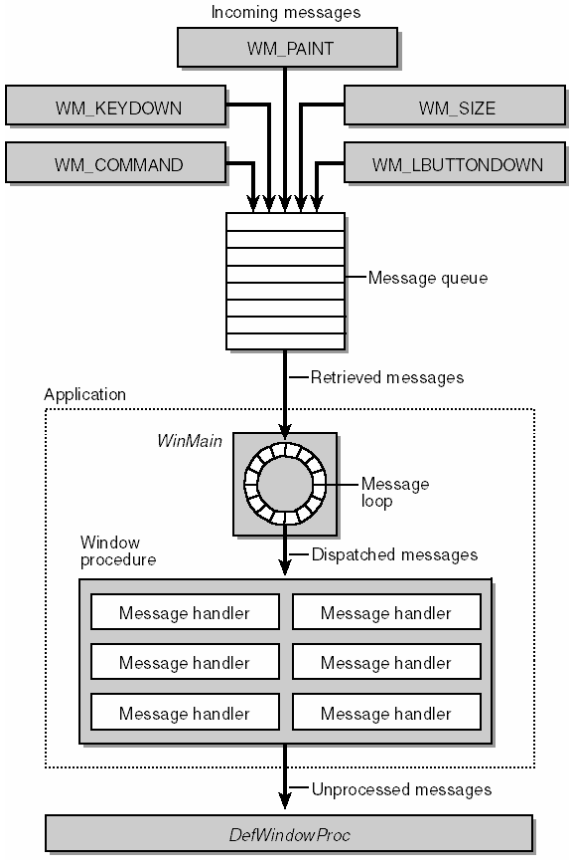
1.1 GIỚI THIỆU KHUNG ỨNG DỤNG WINDOWS (WINDOWS APPLICATION) VÀ XÂY DỰNG CHƯƠNG TRÌNH MẪU VỚI MFC APP FRAMEWORK

1.1.1 Lập trình Windows

Lập trình Windows là kỹ thuật lập trình sử dụng các hàm Windows API để xây dựng các trình ứng dụng trong Windows (*Window App*) và các dạng ứng dụng khác như DLL, ActiveX, …Tuy là kỹ thuật lập trình mạnh mẽ nhưng đòi hỏi tính chuyên nghiệp cao của lập trình viên, giải quyết kế thừa kém, khó phát triển nhanh một ứng dụng.

1.1.2 Mô hình lập trình Windows

Kỹ thuật lập trình sử dụng các hàm Windows API còn gọi là lập trình Windows SDK. Một ứng dụng xây dựng theo kỹ thuật này chứa đựng hàm WinMain (xử lý các thông báo (*message*) nhận được/gửi đi nhằm đáp ứng yêu cầu tương tác của người dùng và của hệ thống cũng như của ứng dụng khác) và hàm DefWinProc (điều phối hoạt động tương ứng với các thông báo nhận được). Tổ chức hệ thống của ứng dụng Windows dạng SDK như sau:



Ví dụ:

```
#include <windows.h>

LONG WINAPI WndProc(HWND, UINT, WPARAM, LPARAM);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpszCmdLine, int nCmdShow)
{
    WNDCLASS wc;
    HWND hwnd;
    MSG msg;
```

```
    wc.style = 0;                           // Class style
    wc.lpfnWndProc = (WNDPROC) WndProc;     // Window procedure address
    wc.cbClsExtra = 0;                      // Class extra bytes
    wc.cbWndExtra = 0;                      // Window extra bytes
    wc.hInstance = hInstance;               // Instance handle
    wc.hIcon = LoadIcon(NULL, IDI_WINLOGO);         // Icon handle
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Cursor handle
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1); // Background color
    wc.lpszMenuName = NULL;                          // Menu name
    wc.lpszClassName = "MyWndClass";                 // WNDCLASS name
    RegisterClass(&wc);

    hwnd = CreateWindow(
        "MyWndClass",               // WNDCLASS name
        "SDK Application",          // Window title
        WS_OVERLAPPEDWINDOW,        // Window style
        CW_USEDEFAULT,              // Horizontal position
        CW_USEDEFAULT,              // Vertical position
        CW_USEDEFAULT,              // Initial width
        CW_USEDEFAULT,              // Initial height
        HWND_DESKTOP,               // Handle of parent window
        NULL,                       // Menu handle
        hInstance,                  // Application's instance handle
        NULL                        // Window-creation data
    );

    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    while(GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam,
    LPARAM lParam)
{
    PAINTSTRUCT ps;
    HDC hdc;
    switch(message) {
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        Ellipse(hdc, 0, 0, 200, 100);
        EndPaint(hwnd, &ps);
        return 0;

    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    }
    return DefWindowProc(hwnd, message, wParam, lParam);
}
```

1.1.3 Lập trình Windows với MFC

Lập trình Windows với MFC là kỹ thuật lập trình sử dụng bộ thư viện MFC của Microsoft để xây dựng các trình ứng dụng trong Windows (*Window App*) và các dạng ứng dụng khác như DLL, COM, ActiveX …

MFC (*Microsoft Foundation Classes*) là thư viện cơ sở chứa các lớp (*class*) C++ do Microsoft cung cấp nhằm đặt một trình bao bọc cho Windows API tạo sự thuận lợi cao cho người dùng trong việc phát triển ứng dụng. Ngoài ra, MFC còn cung cấp kiến trúc View/Document giúp định nghĩa cấu trúc chương trình và cấu trúc tài liệu cho trình ứng dụng đơn giản, uyển chuyển và dễ phát triển hơn. Do đó MFC còn được xem là một khung ứng dụng (*application framework*)

Việc hỗ trợ lớp thừa kế và các hàm AFX cũng như các lớp tiện ích của MFC giúp người dùng thuận tiện hơn việc phát triển ứng dụng tạo nhanh các điều khiển (*control*) trong Windows và truy xuất chúng nhanh chóng và dễ dàng.

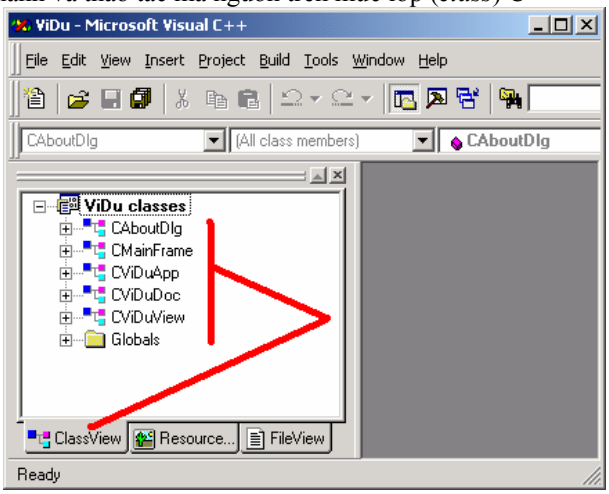
1.1.4 Môi trường lập trình MS Visual C++

Môi trường lập trình Visual C++ bao gồm:

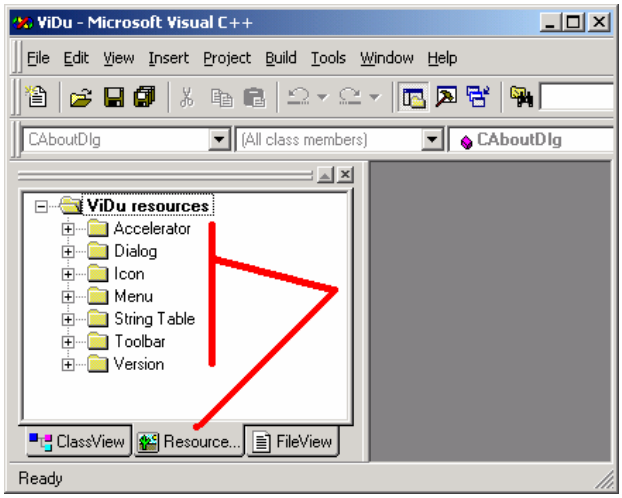
1.1.4.1 Miền làm việc

Khi khởi động lần đầu tiên, vùng bên trái Developer Studion được gọi là miền làm việc, đây chính là vùng để điều hành các phần khác nhau của các dự án pháp triển (*project*). Miền làm việc này cho phép xem các phần của ứng dụng theo ba các khác nhau (như các hình dưới đây):

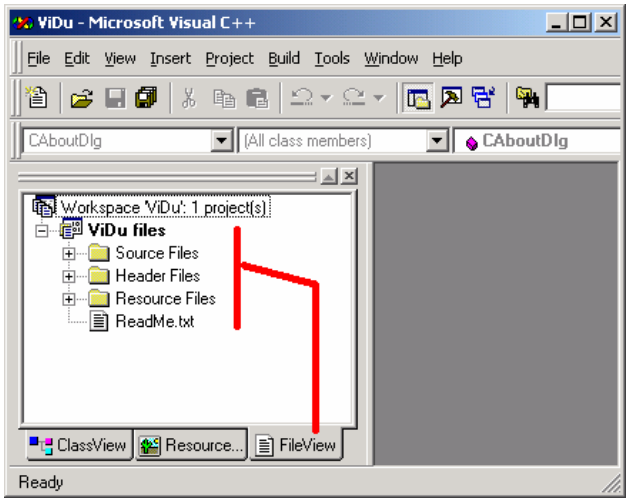
Class View: cho phép điều hành và thao tác mã nguồn trên mức lớp (*class*) C++



Resource View: cho phép tìm và chọn lọc các tài nguyên khác nhau trong ứng dụng như thiết kế cửa sổ hội thoại, biểu tượng, menu, toolbar...

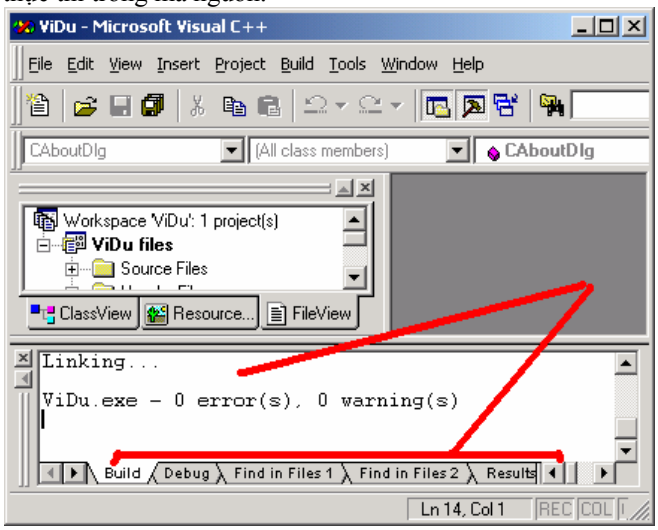


File View: cho phép xem và điều hành tất cả các file trong ứng dụng.



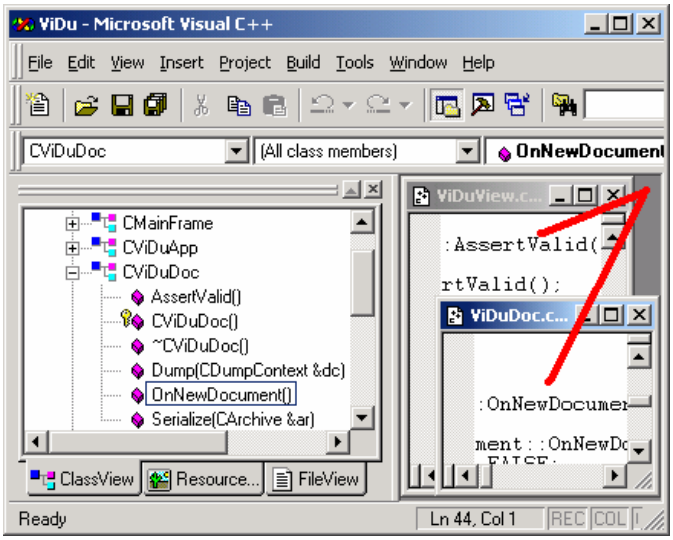
1.1.4.2 Cửa sổ xuất (output pane)

Cửa sổ này nằm ở phần dưới cùng trong cửa sổ ứng Visual C++, thường có thể không hiện trên màn hình khi khởi động ứng dụng Visual C++ lần đầu tiên mà sẽ xuất hiện sau khi thực hiện biên dịch ứng dụng lần đầu tiên. Phần cửa sổ này là nơi cung cấp tất cả thông tin cần thiết cho người dùng như: các câu lệnh, lời cảnh báo và thông báo lỗi của trình biên dịch, đồng thời là nơi chương trình gỡ rối hiển thị tất cả các iến với những giá trị hiện hành trong thời gian thực thi trong mã nguồn.



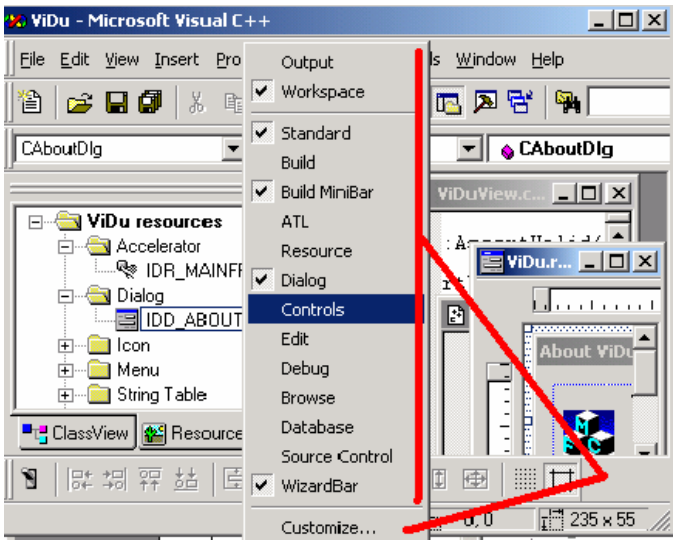
1.1.4.3 Vùng soạn thảo

Đây là vùng bên phải của môi trường để người dùng thực hiện tất cả thao tác soạn thảo chương trình khi sử dụng Visual C++, nơi các cửa sổ soạn thảo chương trình hiển thị, đồng thời là nơi cửa sổ vẽ hiển thị khi người dùng thiết kế hộp thoại.

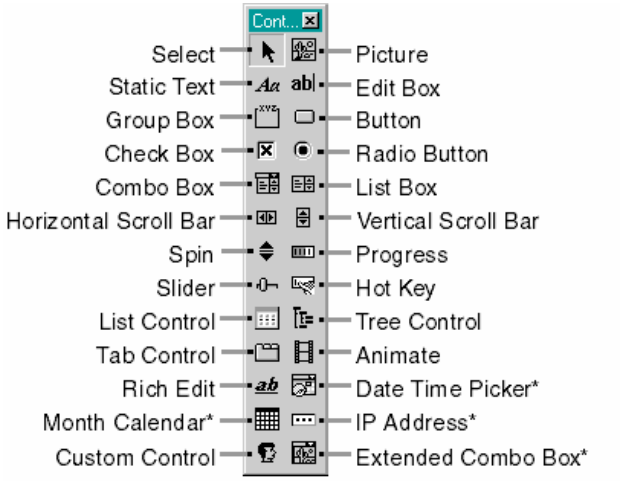


1.1.4.4 Thanh thực đơn (menu)

Lần đầu tiên chạy Visual C++, có ba thanh công cụ hiển thị ngay dưới thanh menu (*menu bar*). Trong Visual C++ có sẵn nhiều thanh công cụ khác nhau, người dùng có thể tùy biến tạo các thanh công cụ phù hợp nhất cho riêng mình.



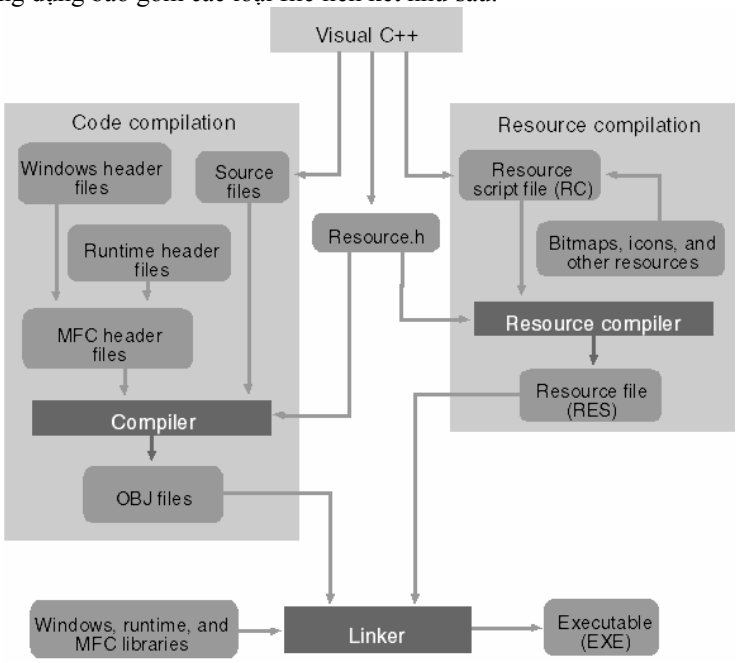
1.1.4.5 Thanh công cụ



*Indicates a new Internet Explorer 4 common control introduced in Visual C++ 6.0. These are covered in Chapter 9.

1.1.5 Các thành phần của ứng dụng phát triển với MS Visual C++

Các thành phần của ứng dụng bao gồm các loại file liên kết như sau:



Các loại file liên quan đến ứng dụng VC++:

Phần mở rộng	Diễn giải
APS	Supports ResourceView
BSC	Browser information file
CLW	Supports ClassWizard
DEP	Dependency file
DSP	Project file*
DSW	Workspace file*
MAK	External makefile
NCB	Supports ClassView
OPT	Holds workspace configuration
PLG	Builds log file

Ví dụ tổng hợp:

Hello.h

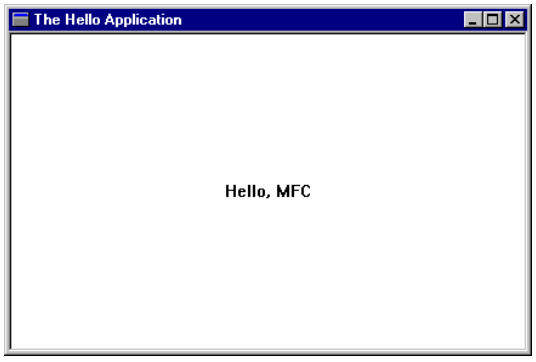
```
class CMyApp : public CWinApp
{
public:
    virtual BOOL InitInstance();
};
class CMainWindow : public CFrameWnd
{
public:
    CMainWindow();

protected:
    afx_msg void OnPaint();
    DECLARE_MESSAGE_MAP()
};
```

Hello.cpp

```
#include <afxwin.h>
#include "Hello.h"

CMyApp myApp;
```

```
//////////////////////////////////////////////////////////////////////////
// CMyApp member functions
BOOL CMyApp::InitInstance()
{
    m_pMainWnd = new CMainWindow;

    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}
//////////////////////////////////////////////////////////////////////////
// CMainWindow message map and member functions
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
    ON_WM_PAINT()
END_MESSAGE_MAP()
CMainWindow::CMainWindow()
{
    Create(NULL, _T("The Hello Application"));
}
void CMainWindow::OnPaint()
{
    CPaintDC dc(this);

    CRect rect;
    GetClientRect(&rect);

    dc.DrawText(_T("Hello, MFC"), -1, &rect,
        DT_SINGLELINE | DT_CENTER | DT_VCENTER);
}
```

Màn hình kết quả như sau:



Một ứng dụng phát triển dựa trên tập thư viện cơ sở MFC bao gồm một số đối tượng và quá trình xử lý như sau::

1.1.5.1 Đối tượng ứng dụng (Application)

Trung tâm của một ứng dụng MFC là đối tượng (*application*) dựa trên lớp CWinApp. CWinApp cung cấp một vòng lặp thông báo để nhận các thông báo và phân phối chúng cho cửa sổ ứng dụng. Nó cũng bao gồm các hàm ảo chính yếu (nó mà có thể bị khai báo và điều chỉnh lại các hành vi của ứng dụng). CWinApp và các lớp MFC khác được đưa vào trong ứng dụng khi chúng ta gắn kết (*include*) file tiêu đề Afxwin.h. Ứng dụng MFC có thể có duy nhất một đối tượng ứng dụng và đối tượng ứng dụng này cần được khai báo với phạm vi toàn cục được khởi tạo trong bộ nhớ từ lúc khởi điểm của chương trình.

CMyApp (trong ví dụ trên) khai báo không có biến thành viên và khai báo lại (*overrides*) một hàm kế thừa từ lớp cha CWinApp. Hàm InitInstance được gọi từ rất sớm trong thời gian sống của ứng dụng, ngay sau khi ứng dụng bắt đầu thực thi nhưng trước khi cửa sổ ứng dụng được tạo ra. Thực tế, ngoại trừ việc hàm InitInstance tạo một cửa sổ cho ứng dụng thì ứng dụng không hề có một cửa sổ. Đây chính là lý do thậm chí một ứng dụng MFC ở mức tối thiểu củng cần kế thừa một lớp từ CWinApp và khai báo lại hàm CWinApp::InitInstance.

1.1.5.2 Đối tượng Khung Cửa sổ (Frame Window)

Lớp CWnd và các phát sinh của nó cung cấp một giao diện hướng đối tượng cho một hay nhiều cửa sổ do ứng dụng tạo ra. Lớp cửa sổ chính của ứng dụng, CMainWindow, được kế thừa từ lớp CFrameWnd (cùng được kế thừa từ CWnd). Lớp CFrameWnd mô hình hoá các hành vi của khung cửa sổ, đồng thời nó là cửa sổ mức cao nhất phục vụ như một giao diện chủ yếu của ứng dụng với thế giới bên ngoài. Trong ngữ cảnh lý tưởng của kiến trúc document/view, cửa sổ khung đóng vai trò như là một lớp chứa thông minh cho các views, toolbars, status bars, và các đối tượng giao diện người sử dụng (*user-interface, UI*) khác.

Một ứng dụng MFC tạo một cửa sổ thông qua việc tạo một đối tượng cửa sổ và gọi hàm Create hay CreateEx của nó có dạng như sau:

```
BOOL Create (LPCTSTR lpszClassName,
   LPCTSTR lpszWindowName,
   DWORD dwStyle = WS_OVERLAPPEDWINDOW,
   const RECT& rect = rectDefault,
   CWnd* pParentWnd = NULL,
   LPCTSTR lpszMenuName = NULL,
   DWORD dwExStyle = 0,
   CCreateContext* pContext = NULL)
```

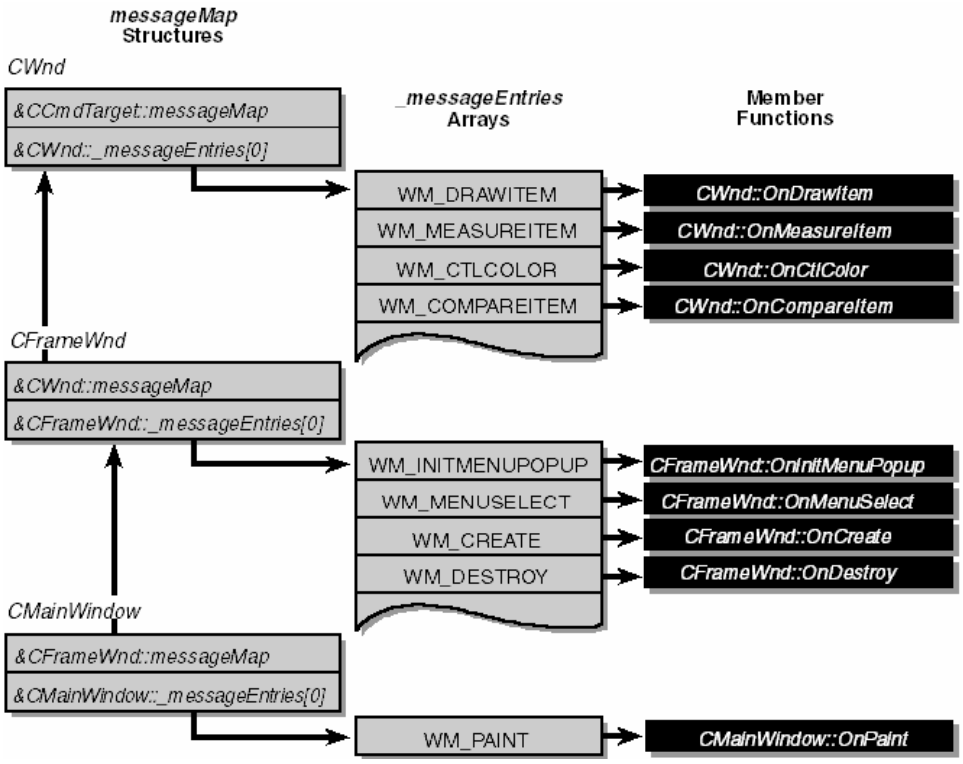
1.1.5.3 Quá trình làm việc của các ánh xạ thông báo (Message Map)

Quá trình làm việc này khảo sát các macro DECLARE_MESSAGE_MAP, BEGIN_MESSAGE_MAP, và END_MESSAGE_MAP trong Afxwin.h và mã lệnh cho hàm CWnd::WindowProc trong Wincore.cpp

```
// In the class declaration
DECLARE_MESSAGE_MAP()


// In the class implementation
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
   ON_WM_PAINT()
END_MESSAGE_MAP()
```

Để phân phối thông báo, khung ứng dụng gọi hàm ảo WindowProc (mà lớp CMainWindow kế thừa từ CWnd). Hàm WindowProc gọi OnWndMsg mà trong đó gọi tiếp hàm GetMessageMap để lấy con trỏ chỉ đến CMainWindow::messageMap và tìm kiếm CMainWindow::_messageEntries cho những thông báo có ID trùng với ID của các thông báo đang chờ xử lý. Nếu kết quả tìm thấy, lớp CMainWindow tương ứng (của những địa chỉ được lưu trong dãy _messageEntries với ID của thông báo) được gọi. Ngược lại, OnWndMsg tham khảo CMainWindow::messageMap cho một con trỏ chỉ tới CFrameWnd::messageMap và lặp lại quá trình cho lớp cơ sở. Nếu lớp cơ sở không có một điều khiển (*handler*) cho thông báo, khung ứng dụng the framework phát triển lên mức khác và tham khảo đến lớp cơ sở cha.

Các ánh xạ thông báo CMainWindow thể hiện dạng sơ đồ như hình dưới đây và thể hiện các nhánh truy xuất/tìm kiếm cho một điều khiển trùng với ID của thông báo đã cho, bắt đầu với các ánh xạ thông báo cho CMainWindow.



Hình. Quá trình xử lý thông báo

1.1.5.4 Windows, Character Sets, và _T Macro

Windows 98 và Windows NT sử dụng hai tập ký tự khác nhau từ các dạng ký tự và chuỗi. Windows 98 và các phiên bản trước đó sử dụng tập ký tự ANSI 8 bit tương tự với tập ký tự ASCII thân thiện với các lập trình viên. Windows NT và Windows 2000 sử dụng tập ký tự Unicode 16 bit bao trùm cả tập ký tự ANSI nhằm phục vụ cho các ứng dụng quốc tế (có thể không sử dụng bảng mẫu tự tiếng Anh).

Các chương trình được biên dịch với ký tự ANSI sẽ hoạt động được trên Windows NT and Windows 2000, nhưng các chương trình dùng Unicode sẽ có thể thực thi nhanh hơn vì Windows NT và Windows 2000 không hỗ trợ việc chuyển đổi từ ANSI sang Unicode cho tất cả ký tự. Ngược lại, ứng dụng dùng Unicode không thực thi trên Windows 98 ngoại trừ khi thực hiện việc chuyển đổi mọi chuỗi ký tự từ Unicode sang dạng ANSI.

Nếu một chuỗi như: "Hello" thì trình biên dịch sẽ thể hiện dạng chuỗi ký tự ANSI.

Nếu khai báo chuỗi trên theo dạng L"Hello" thì trình biên dịch sẽ thể hiện dạng chuỗi ký tự Unicode.

Nhưng nếu dùng macro _T (của MFC) cho chuỗi trên theo dạng _T ("Hello") thì kết quả sẽ được thể hiện dạng Unicode nếu ký hiệu tiền xử lý _UNICODE được định nghĩa, và mặc định là dạng ANSI.

1.1.5.5 Hàm UpdateData

Hàm có dạng **UpdateData**(*tham_số*) với:

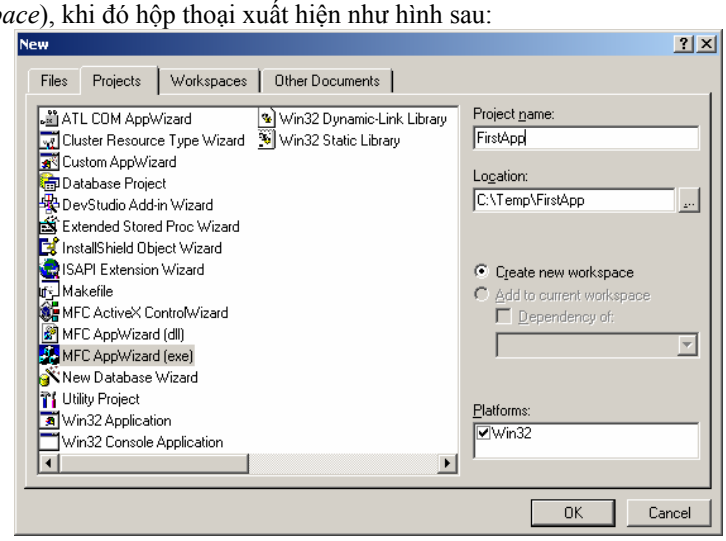
- *tham_số* là TRUE: hàm sẽ thực hiện việc cập nhật dữ liệu trong các điều khiển vào các biến liên kết tương ứng.
- *tham_số* là FALSE: hàm sẽ thực hiện việc cập nhật dữ liệu từ các biến liên kết vào trong các điều khiển tương ứng và hiển thị trên giao diện chương trình.

Một số lưu ý:

- Nên khai báo ký tự kiểu TCHAR thay cho kiểu char. Nếu ký hiệu _UNICODE được định nghĩa, TCHAR xác định kiểu wchar_t (ký tự Unicode 16 bit). Nếu _UNICODE không được định nghĩa thì TCHAR trở thành ký hiệu thông thường.
- Không nên dùng char* hay wchar_t* để khai báo con trỏ kiểu chuỗi TCHAR. Nên dùng TCHAR* hay LPTSTR (con trỏ chỉ tới chuỗi TCHAR) và LPCTSTR (con trỏ chỉ tới chuỗi hằng TCHAR).
- Không nên giả định là ký tự chỉ có độ rộng 8 bit. Để chuyển một độ dài của bộ đệm nhanh ở dạng byte sang dạng ký tự, nên dùng sizeof(TCHAR).
- Thay các việc gọi hàm chuỗi trong thư việc C-trong thời gian thực thi (ví dụ strcpy) với các macros tương ứng trong file tiêu đề Tchar.h (ví dụ, _tcscpy).

1.1.6 Tạo ứng dụng với MS Visual C++

Từ menu File, người dùng chọn lệnh New... để tạo mới một dự án (project), một tập tin (*file*) hay một không gian làm việc (*workspace*), khi đó hộp thoại xuất hiện như hình sau:



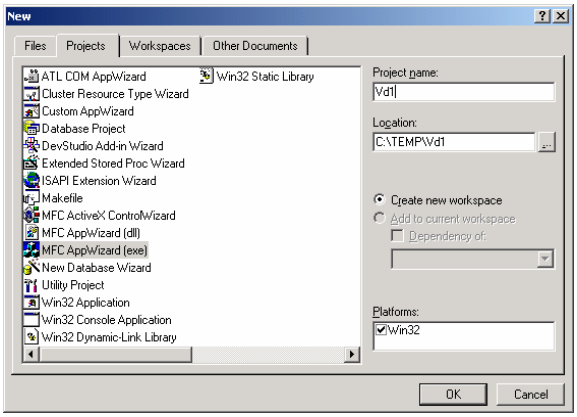
Trong hộp thoại này, người dùng có nhiều loại trình ứng dụng có thể tạo với MS Visual C++:

- Tạo ứng dụng thực thi trong Windows (dạng EXE file) với MFC có hỗ trợ tư vấn với **MFC AppWizard (exe)**
- Tạo thư viện trong Windows (dạng DLL file) với MFC có hỗ trợ tư vấn với **MFC AppWizard (dll)**
- Tạo ứng dụng thực thi trong Windows (dạng EXE file) dạng thông thường sử dụng API với **Win32 Application**
- Tạo ứng dụng thực thi trong DOS (dạng EXE file) dạng thông thường với **Win32 Console Application...**

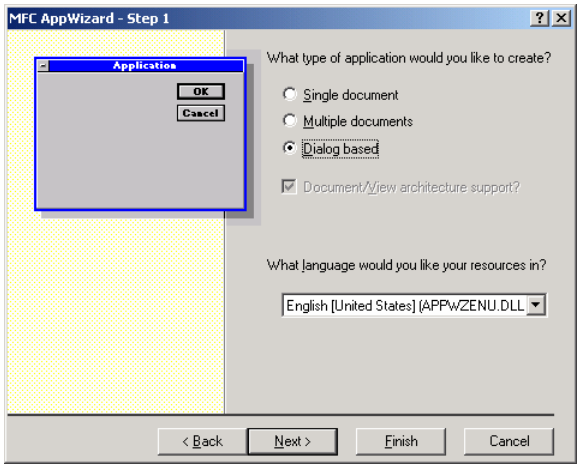
Đặc biệt, khi người dùng chọn cách tạo ứng dụng dạng cửa sổ với **MFC AppWizard (exe)**, người dùng có thể tạo trình ứng dụng dạng hộp thoại (*dialog*), ứng dụng đơn tài liệu (*Single Document Interface - SDI*), ứng dụng đa tài liệu (*Multi Document Interface - MDI*)

Nếu tạo ứng dụng dạng hộp thoại (dialog), người dùng cần làm như sau:

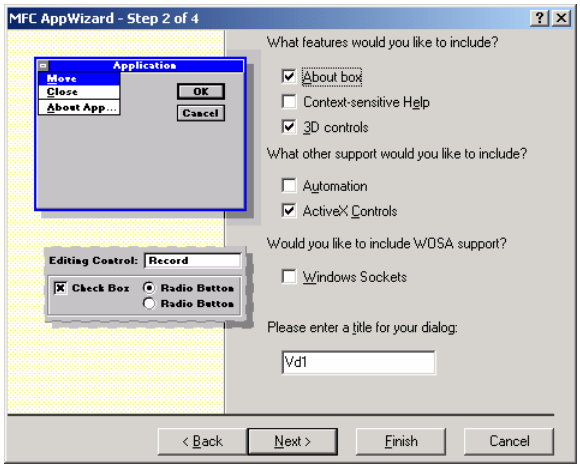
- Bước khởi đầu:



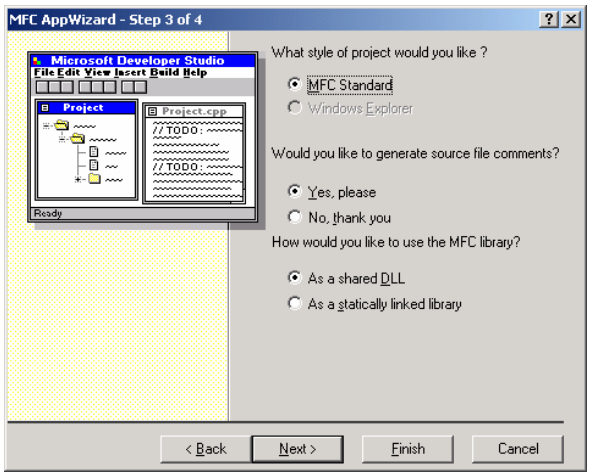
- Bước 1:



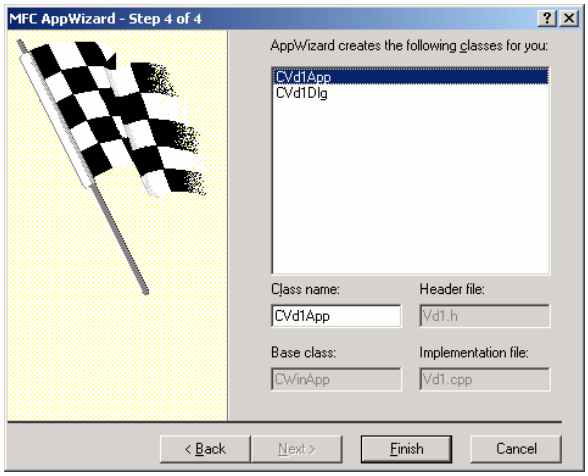
- Bước 2:



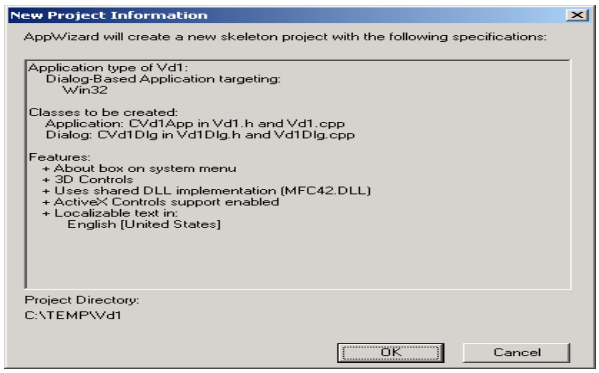
- Bước 3:



- Bước 4:

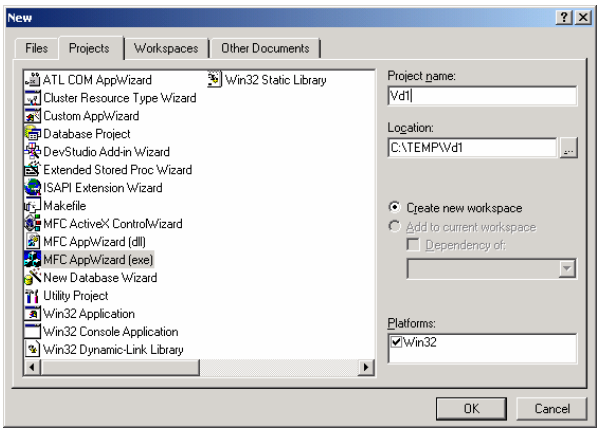


- Bước kết thúc:

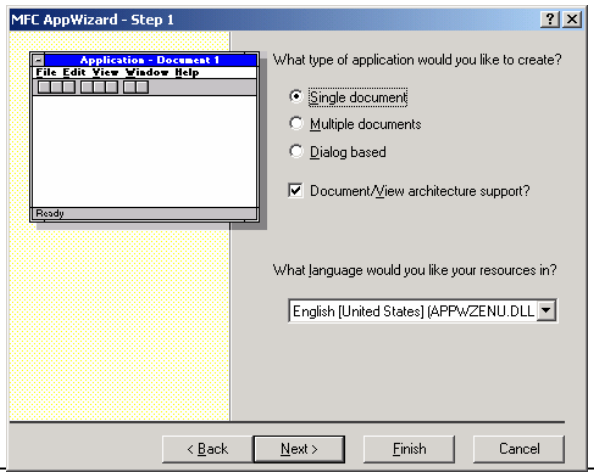


Nếu tạo ứng dụng dạng đơn tài liệu hay đa tài liệu, người dùng cần làm như sau:

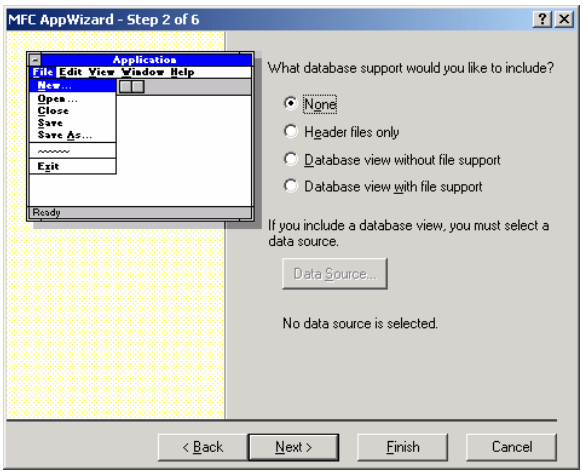
- Bước khởi đầu:



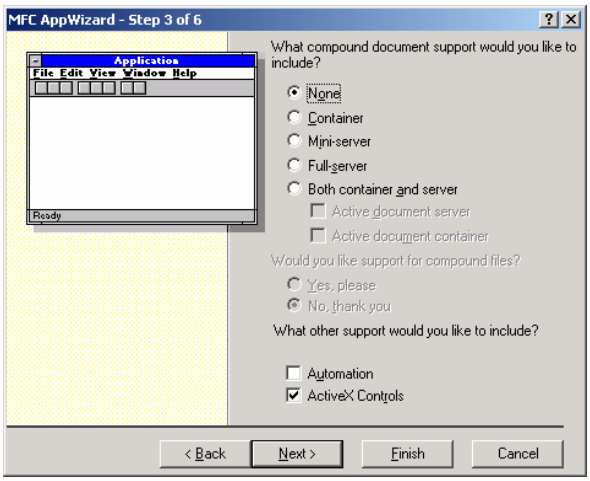
- Bước 1:



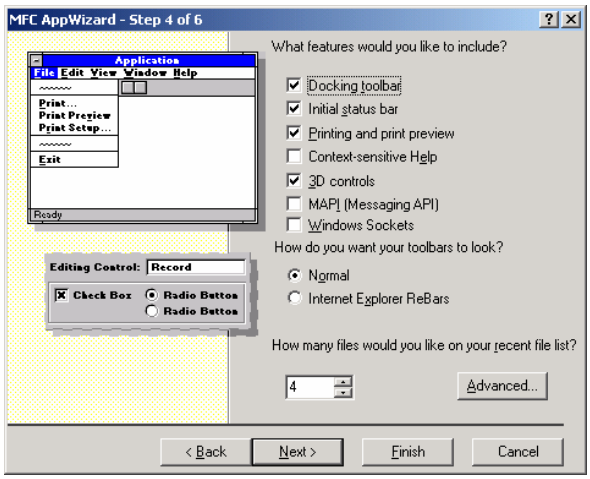
- Bước 2:



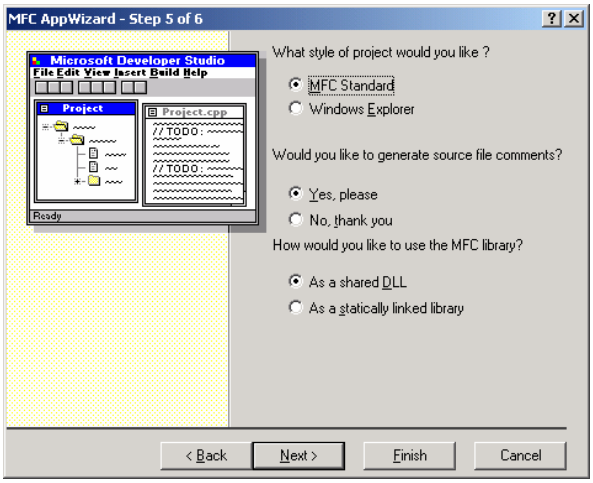
- Bước 3:



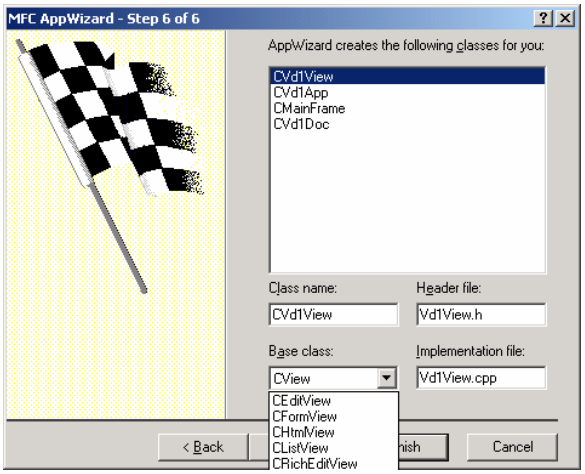
- Bước 4:



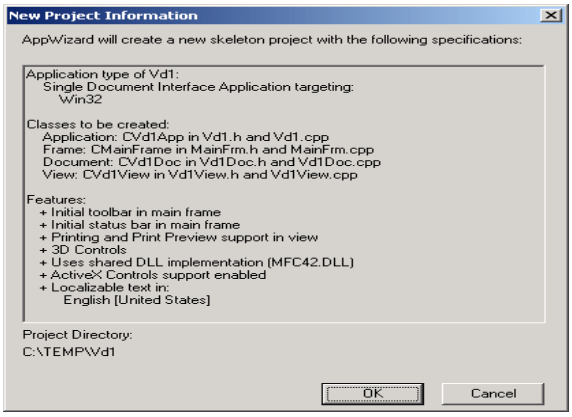
- Bước 5:



- Bước 6:



- Bước kết thúc:



1.2 XỬ LÝ VẼ HÌNH TRONG ỨNG DỤNG WINDOWS

1.2.1 Vấn đề quan tâm

- Tìm hiểu về ngữ cảnh thiết bị và giao diện thiết bị đồ hoạ.
- Sự hỗ trợ của MFC về các lớp công cụ vẽ (CPen, CBrush…)

1.2.2 Giới thiệu

Hệ điều hành Windows cung cấp thiết bị đồ hoạ ảo (*Graphics Device Interface - GDI*) để giúp người dùng thực hiện các thao tác vẽ đồ hoạ dễ dàng hơn vì thiết bị này không phụ thuộc vào phần cứng đồ hoạ của hệ thống.

Một chương trình ứng dụng (*WindowsApp*) không vẽ trực tiếp ra màn hình, máy in… mà chỉ vẽ trên "bề mặt luận lý" thể hiện bởi ngữ cảnh thiết bị (*Device Context – DC*). Ngữ cảnh thiết bị chứa các thông tin về hệ thống, ứng dụng và cửa sổ của WindowsApp cũng như các đối tượng đồ hoạ đang được vẽ trong WindowApp đó. Thực chất ngữ cảnh thiết bị dùng hiển thị đồ hoạ là ngữ cảnh ảo của cửa sổ.

1.2.3 Truy xuất ngữ cảnh thiết bị

MFC cung cấp một số lớp ngữ cảnh thiết bị (kế thừa từ lớp CDC) như:

Class	Mô tả
CPaintDC	Sử dụng cho việc vẽ trong vùng ứng dụng của cửa sổ (chỉ thao tác với sự kiện OnPaint)
CClientDC	Sử dụng cho việc vẽ trong vùng ứng dụng của cửa sổ (vẽ bất cứ nơi nào nhưng chỉ thao tác với sự kiện OnPaint)
CWindowDC	Sử dụng cho việc vẽ trong cửa sổ, bao gồm cả vùng không là vùng ứng dụng của cửa sổ
CMetaFileDC	Sử dụng cho việc vẽ một GDI metafile

Để lấy ngữ cảnh thiết bị, dùng:

```
CDC dc(this);
```

hay

```
CDC* pDC = GetDC();
```

Để giải phóng ngữ cảnh thiết bị, dùng:

```
ReleaseDC(pDC);
delete pDC;
```

Ngữ cảnh thiết bị dùng "bút vẽ" (*pen*) để vẽ các đường thẳng/hình dáng, và "cọ vẽ" (*brush*) để tô đầy các vùng của hình vẽ trên màn hình.

Ví dụ:

```
CRect rect;
GetClientRect(&rect);
CClientDC dc(this);
dc.MoveTo(rect.left, rect.top);
dc.LineTo(rect.right, rect.bottom);
dc.Ellipse(0, 0, 100, 100);
```

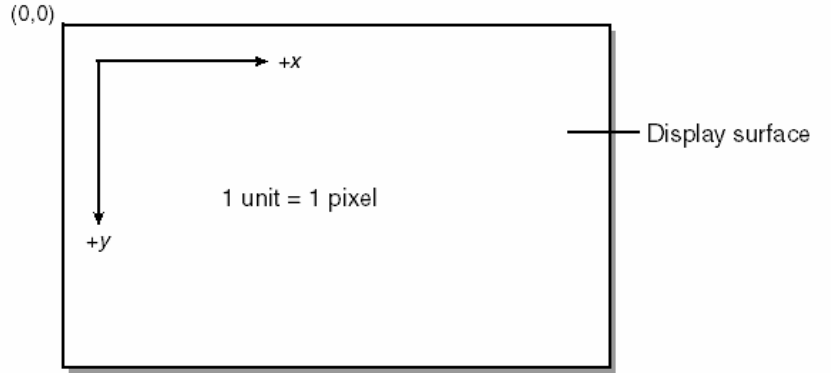
hay:

```
CRect rect;
GetClientRect(&rect);
CDC* pDC = GetDC();
pDC->MoveTo(rect.left, rect.top);
pDC->LineTo(rect.right, rect.bottom);
pDC->Ellipse(0, 0, 100, 100);
```

1.2.3.1 Xác định chế độ đo lường

Chế độ liên kết	Khoảng cách tương ứng với một đơn vị luận lý	Hướng của trục x và y
MM_TEXT	1 pixel	
MM_LOMETRIC	0.1 mm	
MM_HIMETRIC	0.01 mm	

MM_LOENGLISH	0.01 in.	
MM_HIENGLISH	0.001 in.	
MM_TWIPS	1/1440 in. (0.0007 in.)	
MM_ISOTROPIC	Người dùng định nghĩa (x và y có tỉ lệ xác định)	Người dùng định nghĩa
MM_ANISOTROPIC	Người dùng định nghĩa (x và y có tỉ lệ xác định)	Người dùng định nghĩa



1.2.4 Thao tác vẽ với bút vẽ

Trong MFC, lớp bút vẽ là CPen, được dùng để vẽ kiểu đường bất kỳ với màu/độ rộng xác định, ví dụ như sau tạo bút vẽ mới và chọn làm bút vẽ hiện thời, dùng:

```
CDC *pDC = GetDC();
CPen cp(PS_SOLID, 1, RGB(0,0,0));
pDC->SelectObject(&cp);
...
```

tạo bút vẽ mới đồng thời lưu lại bút vẽ đã sử dụng trước đó:

```
CDC *pDC = GetDC();
CPen pen (PS_SOLID, 10, RGB (255, 0, 0));
CPen* pOldPen = pDC->SelectObject (&pen);
...
```

trong đó hàm ***RGB(r, g, b)*** tạo màu chỉ định dựa trên 3 màu cơ bản là R, G, B với các tham số r, g và b $\in$ [0, 255] và các kiểu nét như PS_SOLID, PS_DASH, PS_DOT ... như sau:

PS_SOLID

PS_DASH

PS_DOT

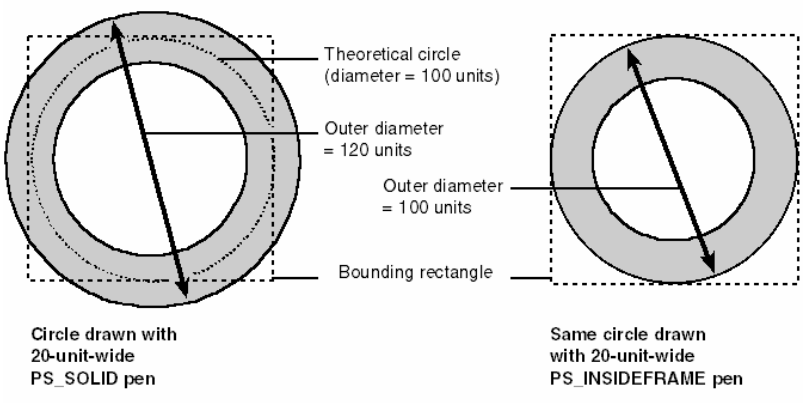
PS_DASHDOT

PS_DASHDOTDOT

PS_NULL

PS_INSIDEFRAME

trong đó



Một số phương thức vẽ đường:

Phương thức	Mô tả
MoveTo	Di chuyển bút vẽ đến 1 điểm xác định
LineTo	Vẽ 1 đoạn thẳng từ điểm hiện hành của bút vẽ đến 1 điểm xác định và di chuyển vị trí hiện hành đến điểm mới này
Polyline	Vẽ đường gấp khúc (tập hợp các đoạn gấp khúc)
PolylineTo	Vẽ đường gấp khúc và di chuyển vị trí hiện hành đến đỉnh cuối cùng của đường này.
Arc	Vẽ cung
ArcTo	Vẽ cung và di chuyển vị trí hiện hành đến đỉnh cuối cùng của cung này.
PolyBezier	Vẽ đường Bezier
PolyBezierTo	Vẽ đường Bezier và di chuyển vị trí hiện hành đến đỉnh cuối cùng của đường này.

Một số phương thức vẽ hình khối:

Phương thức	Mô tả
Chord	Vẽ hình dạng bán cầu
Ellipse	Vẽ hình dạng ellipse
Pie	Vẽ hình dạng bánh
Polygon	Nối một tập các điểm của một đa giác
Rectangle	Vẽ hình dạng chữ nhật
RoundRect	Vẽ hình dạng chữ nhật tròn góc

Một số mã màu thông dụng:

Tên màu	R	G	B	Tên màu	R	G	B
Black	0	0	0	Light gray	192	192	192
Blue	0	0	192	Bright blue	0	0	255
Green	0	192	0	Bright green	0	255	0
Cyan	0	192	192	Bright cyan	0	255	255
Red	192	0	0	Bright red	255	0	0
Magenta	192	0	192	Bright magenta	255	0	255
Yellow	192	192	0	Bright yellow	255	255	0
Dark gray	128	128	128	White	255	255	255

Chú ý:

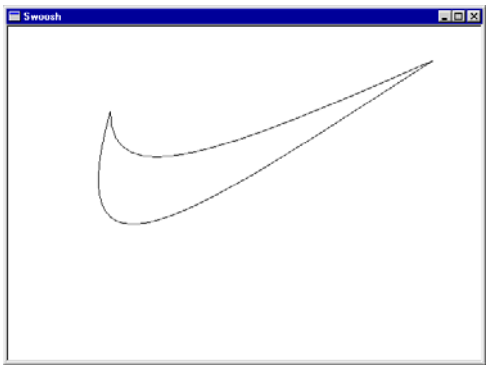
Việc xử lý vẽ có thể đặt trong sự kiện OnPaint, OnDraw hay các sự kiện liên quan đến thao tác chuột và bàn phím.

Ví dụ:

Với đoạn chương trình sau đây:

```
POINT aPoint1[4] = { 120, 100, 120, 200, 250, 150, 500, 40 };
POINT aPoint2[4] = { 120, 100,  50, 350, 250, 200, 500, 40 };
dc.PolyBezier (aPoint1, 4);
dc.PolyBezier (aPoint2, 4);
```

Màn hình kết quả là :

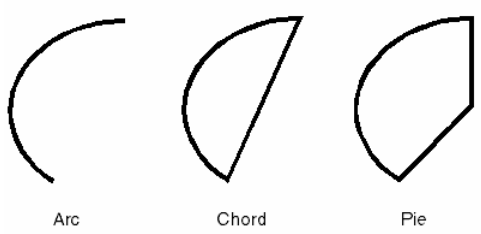


Ví dụ:

Với đoạn chương trình sau đây:

```
#include <math.h>
#define PI 3.1415926

void CMainWindow::OnPaint()
{
    CPaintDC dc (this);
    int nRevenues[4] = { 125, 376, 252, 184 };

    CRect rect;
    GetClientRect(&rect);
    dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

    int nTotal = 0;
    for (int i=0; i<4; i++)
        nTotal += nRevenues[i];
    int x1 = 0;
    int y1 = -1000;
    int nSum = 0;
    for (i=0; i<4; i++) {
        nSum += nRevenues[i];
        double rad = ((double)(nSum * 2 * PI)/(double) nTotal) + PI;
        int x2 = (int) (sin (rad) * 1000);
        int y2 = (int) (cos (rad) * 1000 * 3) / 4;
        dc.Pie (-200, -150, 200, 150, x1, y1, x2, y2);
        x1 = x2;
        y1 = y2;
    }
}
```

Màn hình kết quả là



1.2.5 Thao tác tô màu với cọ vẽ

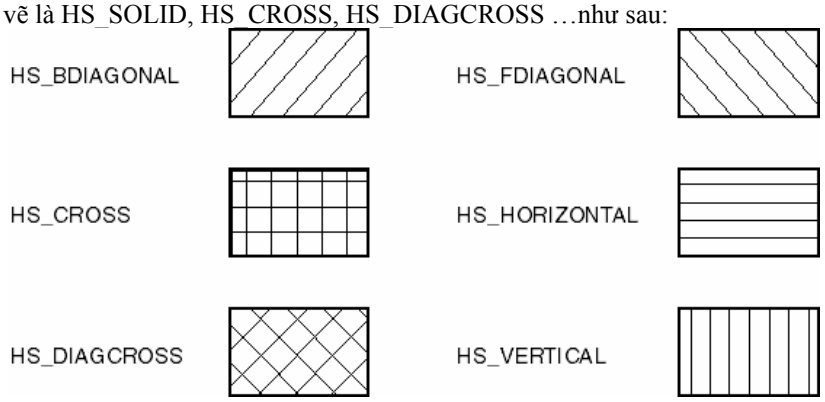
Lớp cọ vẽ là CBrush, được dùng để tạo cọ vẽ với màu/mẫu vẽ xác định:

```
CDC *pDC=GetDC();
CBrush brush (HS_DIAGCROSS, RGB (255, 255, 255));
pDC->SelectObject(&brush);
...
```

tạo cọ vẽ mới đồng thời lưu lại cọ vẽ đã sử dụng trước đó:

```
CDC *pDC=GetDC();
CBrush brush (HS_DIAGCROSS, RGB (255, 255, 255));
CBrush* pOldBrush = pDC->SelectObject(&brush);
...
```

trong đó các mẫu vẽ là HS_SOLID, HS_CROSS, HS_DIAGCROSS …như sau:



Ví dụ:

```
CBrush brush (HS_DIAGCROSS, RGB (0, 0, 0));
pDC->SelectObject (&brush);
pDC->SetBkMode (TRANSPARENT);
pDC->Rectangle (0, 0, 100, 100);
```

hay:

```
CBrush brush (HS_DIAGCROSS, RGB (255, 255, 255));
pDC->SelectObject (&brush);
pDC->SetBkColor (RGB (192, 192, 192));
pDC->Rectangle (0, 0, 100, 100);
```

1.2.6 Hiển thị văn bản trong môi trường đồ hoạ

Ngữ cảnh thiết bị cung cấp một số hàm thực hiện việc hỗ trợ hiển thị văn bản trong môi trường đồ hoạ như sau:

Hàm	Mô tả
DrawText	Vẽ một văn bản trong một khung chữ nhật định dạng trước
TextOut	Xuất một hành văn bản tại vị trí hiện hành hay tại điểm xác định
TabbedTextOut	Xuất một hành văn bản chứa đựng các ký hiệu tab
ExtTextOut	Xuất một hàng văn bản trong một khung chữ nhật có màu tùy chọn hay các ký tự xen giữa khác nhau
GetTextExtent	Lấy độ rộng chuỗi với font sử dụng hiện hành
GetTabbedTextExtent	Lấy độ rộng chuỗi (có chứa đựng ký hiệu tab) với font sử dụng hiện hành
GetTextMetrics	Lấy font metrics (chiều cao ký tự, độ rộng trung bình) của font hiện hành

SetTextAlign	Canh lề của văn bản cho hàm TextOut và các hàm xuất nội dung văn bản khác
SetTextJustification	Canh đều văn bản
SetTextColor	Đặt màu cho văn bản xuất ra
SetBkColor	Đặt màu nền

Ví dụ:

```
CString string = _T ("Now is the time");
CSize size = dc.GetTextExtent (string);
dc.SetTextJustification (nWidth - size.cx, 3);
dc.TextOut (0, y, string);
```

hay:

```
dc.DrawText (_T ("Hello, MFC"), -1, &rect,
    DT_SINGLELINE ¦ DT_CENTER ¦ DT_VCENTER);
```

1.2.7 GDI Fonts và lớp CFont

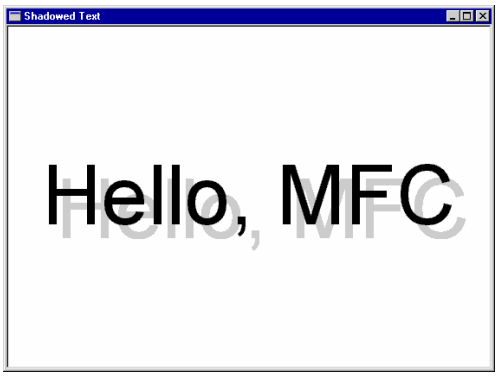
Ngoài việc sử dụng font chữ mặc định, người dùng còn có thể tạo font chữ trong chế độ đồ hoạ tùy chọn.

Ví dụ:

Với đoạn chương trình sau đây:

```
void CMainWindow::OnPaint ()
{
    CRect rect;
    GetClientRect (&rect);

    CFont font;
    font.CreatePointFont (720, _T ("Arial"));

    CPaintDC dc (this);
    dc.SelectObject (&font);
    dc.SetBkMode (TRANSPARENT);

    CString string = _T ("Hello, MFC");

    rect.OffsetRect (16, 16);
    dc.SetTextColor (RGB (192, 192, 192));
    dc.DrawText (string, &rect, DT_SINGLELINE ¦
        DT_CENTER ¦ DT_VCENTER);

    rect.OffsetRect (-16, -16);
    dc.SetTextColor (RGB (0, 0, 0));
    dc.DrawText (string, &rect, DT_SINGLELINE ¦
        DT_CENTER ¦ DT_VCENTER);
}
```

Màn hình kết quả như sau:

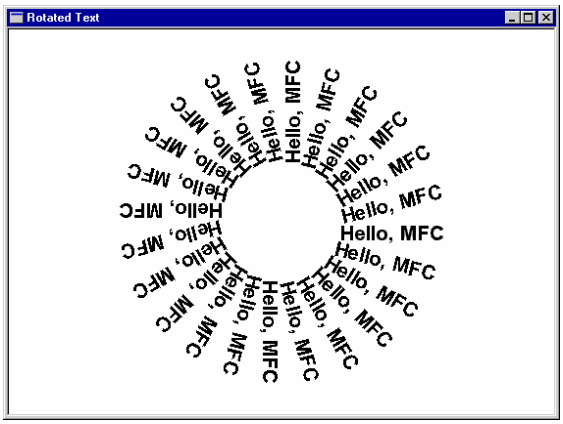


Ví dụ:

Với đoạn chương trình sau đây:

```
void CMainWindow::OnPaint()
{
    CRect rect;
    GetClientRect(&rect);

    CPaintDC dc(this);
    dc.SetViewportOrg(rect.Width () / 2, rect.Height () / 2);
    dc.SetBkMode(TRANSPARENT);

    for (int i=0; i<3600; i+=150) {
        LOGFONT lf;
        ::ZeroMemory(&lf, sizeof (lf));
        lf.lfHeight = 160;
        lf.lfWeight = FW_BOLD;
        lf.lfEscapement = i;
        lf.lfOrientation = i;
        ::lstrcpy(lf.lfFaceName, _T ("Arial"));

        CFont font;
        font.CreatePointFontIndirect(&lf);

        CFont* pOldFont = dc.SelectObject(&font);
        dc.TextOut(0, 0, CString (_T ("          Hello, MFC")));
        dc.SelectObject(pOldFont);
    }
}
```

Màn hình kết quả như sau:



1.2.8 Ví dụ tổng hợp

1.2.8.1 Chương trình 1

Accel.h

```
#define LINESIZE 8

class CMyApp : public CWinApp
{
public:
    virtual BOOL InitInstance();
};
class CMainWindow : public CFrameWnd
{
protected:
    int m_nCellWidth;   // Cell width in pixels
    int m_nCellHeight;  // Cell height in pixels
```

```
    int m_nRibbonWidth; // Ribbon width in pixels
    int m_nViewWidth;   // Workspace width in pixels
    int m_nViewHeight;  // Workspace height in pixels
    int m_nHScrollPos;  // Horizontal scroll position
    int m_nVScrollPos;  // Vertical scroll position
    int m_nHPageSize;   // Horizontal page size
    int m_nVPageSize;   // Vertical page size

public:
    CMainWindow();
protected:
    afx_msg void OnPaint();
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnHScroll(UINT nCode, UINT nPos,
        CScrollBar* pScrollBar);
    afx_msg void OnVScroll(UINT nCode, UINT nPos,
        CScrollBar* pScrollBar);

    DECLARE_MESSAGE_MAP()
};
```

Accel.cpp

```
#include <afxwin.h>
#include "Accel.h"

CMyApp myApp;

//////////////////////////////////////////////////////////////////////
// CMyApp member functions
BOOL CMyApp::InitInstance()
{
    m_pMainWnd = new CMainWindow;
    m_pMainWnd->ShowWindow (m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}
//////////////////////////////////////////////////////////////////////
// CMainWindow message map and member functions
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
    ON_WM_CREATE()
    ON_WM_SIZE()
    ON_WM_PAINT()
    ON_WM_HSCROLL()
    ON_WM_VSCROLL()
END_MESSAGE_MAP()
CMainWindow::CMainWindow()
{
    Create(NULL, _T ("Accel"),
        WS_OVERLAPPEDWINDOW ¦ WS_HSCROLL ¦ WS_VSCROLL);
}
int CMainWindow::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
```

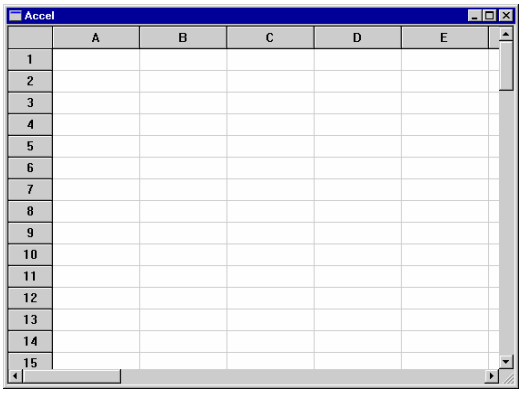
```
    CClientDC dc(this);
    m_nCellWidth = dc.GetDeviceCaps(LOGPIXELSX);
    m_nCellHeight = dc.GetDeviceCaps(LOGPIXELSY) / 4;
    m_nRibbonWidth = m_nCellWidth / 2;
    m_nViewWidth = (26 * m_nCellWidth) + m_nRibbonWidth;
    m_nViewHeight = m_nCellHeight * 100;
    return 0;
}
void CMainWindow::OnSize(UINT nType, int cx, int cy)
{
    CFrameWnd::OnSize(nType, cx, cy);
    //
    // Set the horizontal scrolling parameters.
    //
    int nHScrollMax = 0;
    m_nHScrollPos = m_nHPageSize = 0;

    if (cx < m_nViewWidth) {
        nHScrollMax = m_nViewWidth - 1;
        m_nHPageSize = cx;
        m_nHScrollPos = min(m_nHScrollPos, m_nViewWidth -
            m_nHPageSize - 1);
    }

    SCROLLINFO si;
    si.fMask = SIF_PAGE | SIF_RANGE | SIF_POS;
    si.nMin = 0;
    si.nMax = nHScrollMax;
    si.nPos = m_nHScrollPos;
    si.nPage = m_nHPageSize;

    SetScrollInfo(SB_HORZ, &si, TRUE);
    //
    // Set the vertical scrolling parameters.
    //
    int nVScrollMax = 0;
    m_nVScrollPos = m_nVPageSize = 0;

    if (cy < m_nViewHeight) {
        nVScrollMax = m_nViewHeight - 1;
        m_nVPageSize = cy;
        m_nVScrollPos = min(m_nVScrollPos, m_nViewHeight -
            m_nVPageSize - 1);
    }

    si.fMask = SIF_PAGE | SIF_RANGE | SIF_POS;
    si.nMin = 0;
    si.nMax = nVScrollMax;
    si.nPos = m_nVScrollPos;
    si.nPage = m_nVPageSize;

    SetScrollInfo(SB_VERT, &si, TRUE);
}
void CMainWindow::OnPaint ()
{
```

```
CPaintDC dc (this);
//
// Set the window origin to reflect the current scroll positions.
//
dc.SetWindowOrg(m_nHScrollPos, m_nVScrollPos);
//
// Draw the grid lines.
//
CPen pen(PS_SOLID, 0, RGB (192, 192, 192));
CPen* pOldPen = dc.SelectObject (&pen);

for (int i=0; i<99; i++) {
    int y = (i * m_nCellHeight) + m_nCellHeight;
    dc.MoveTo(0, y);
    dc.LineTo(m_nViewWidth, y);
}

for (int j=0; j<26; j++) {
    int x = (j * m_nCellWidth) + m_nRibbonWidth;
    dc.MoveTo(x, 0);
    dc.LineTo(x, m_nViewHeight);
}
dc.SelectObject (pOldPen);
//
// Draw the bodies of the rows and the column headers.
//
CBrush brush;
brush.CreateStockObject(LTGRAY_BRUSH);

CRect rcTop(0, 0, m_nViewWidth, m_nCellHeight);
dc.FillRect(rcTop, &brush);
CRect rcLeft(0, 0, m_nRibbonWidth, m_nViewHeight);
dc.FillRect(rcLeft, &brush);

dc.MoveTo(0, m_nCellHeight);
dc.LineTo(m_nViewWidth, m_nCellHeight);
dc.MoveTo(m_nRibbonWidth, 0);
dc.LineTo(m_nRibbonWidth, m_nViewHeight);

dc.SetBkMode(TRANSPARENT);
//
// Add numbers and button outlines to the row headers.
//
for (i=0; i<99; i++) {
    int y = (i * m_nCellHeight) + m_nCellHeight;
    dc.MoveTo(0, y);
    dc.LineTo(m_nRibbonWidth, y);

    CString string;
    string.Format(_T ("%d"), i + 1);

    CRect rect(0, y, m_nRibbonWidth, y + m_nCellHeight);
    dc.DrawText(string, &rect, DT_SINGLELINE ¦
        DT_CENTER ¦ DT_VCENTER);
```

```
        rect.top++;
        dc.Draw3dRect(rect, RGB (255, 255, 255),
            RGB (128, 128, 128));
    }
    //
    // Add letters and button outlines to the column headers.
    //
    for (j=0; j<26; j++) {
        int x = (j * m_nCellWidth) + m_nRibbonWidth;
        dc.MoveTo(x, 0);
        dc.LineTo(x, m_nCellHeight);

        CString string;
        string.Format(_T ("%c"), j + `A');

        CRect rect(x, 0, x + m_nCellWidth, m_nCellHeight);
        dc.DrawText(string, &rect, DT_SINGLELINE ¦
            DT_CENTER ¦ DT_VCENTER);

        rect.left++;
        dc.Draw3dRect(rect, RGB (255, 255, 255),
            RGB (128, 128, 128));
    }
}
void CMainWindow::OnHScroll(UINT nCode, UINT nPos, CScrollBar* pScrollBar)
{
    int nDelta;

    switch (nCode) {
    case SB_LINELEFT:
        nDelta = -LINESIZE;
        break;

    case SB_PAGELEFT:
        nDelta = -m_nHPageSize;
        break;
    case SB_THUMBTRACK:
        nDelta = (int) nPos - m_nHScrollPos;
        break;
    case SB_PAGERIGHT:
        nDelta = m_nHPageSize;
        break;
    case SB_LINERIGHT:
        nDelta = LINESIZE;
        break;
    default: // Ignore other scroll bar messages
        return;
    }

    int nScrollPos = m_nHScrollPos + nDelta;
    int nMaxPos = m_nViewWidth - m_nHPageSize;

    if (nScrollPos < 0)
        nDelta = -m_nHScrollPos;
    else if (nScrollPos > nMaxPos)
```

```
        nDelta = nMaxPos - m_nHScrollPos;

    if (nDelta != 0) {
        m_nHScrollPos += nDelta;
        SetScrollPos (SB_HORZ, m_nHScrollPos, TRUE);
        ScrollWindow (-nDelta, 0);
    }
}
void CMainWindow::OnVScroll(UINT nCode, UINT nPos, CScrollBar* pScrollBar)
{
    int nDelta;

    switch (nCode) {
    case SB_LINEUP:
        nDelta = -LINESIZE;
        break;
    case SB_PAGEUP:
        nDelta = -m_nVPageSize;
        break;
    case SB_THUMBTRACK:
        nDelta = (int) nPos - m_nVScrollPos;
        break;
    case SB_PAGEDOWN:
        nDelta = m_nVPageSize;
        break;
    case SB_LINEDOWN:
        nDelta = LINESIZE;
        break;
    default: // Ignore other scroll bar messages
        return;
    }

    int nScrollPos = m_nVScrollPos + nDelta;
    int nMaxPos = m_nViewHeight - m_nVPageSize;

    if (nScrollPos < 0)
        nDelta = -m_nVScrollPos;
    else if (nScrollPos > nMaxPos)
        nDelta = nMaxPos - m_nVScrollPos;

    if (nDelta != 0) {
        m_nVScrollPos += nDelta;
        SetScrollPos(SB_VERT, m_nVScrollPos, TRUE);
        ScrollWindow(0, -nDelta);
    }
}
```

Màn hình kết quả như sau



1.2.8.2 Chương trình 2

Ruler.h

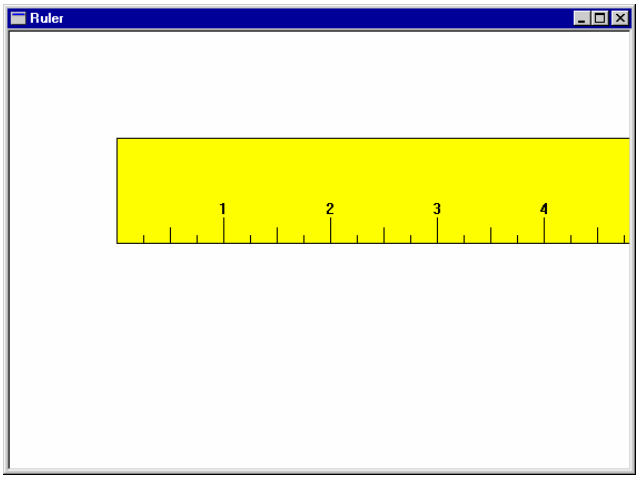
```
class CMyApp : public CWinApp
{
public:
    virtual BOOL InitInstance();
};

class CMainWindow : public CFrameWnd
{
public:
    CMainWindow();

protected:
    afx_msg void OnPaint();
    DECLARE_MESSAGE_MAP()
};
```

Ruler.cpp

```
#include <afxwin.h>
#include "Ruler.h"

CMyApp myApp;
//////////////////////////////////////////////////////////////////////////
// CMyApp member functions
BOOL CMyApp::InitInstance()
{
    m_pMainWnd = new CMainWindow;
    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}
//////////////////////////////////////////////////////////////////////////
// CMainWindow message map and member functions
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
    ON_WM_PAINT()
END_MESSAGE_MAP()
CMainWindow::CMainWindow()
{
    Create(NULL, _T ("Ruler"));
}
void CMainWindow::OnPaint()
{
    CPaintDC dc(this);
```

```
    //
    // Initialize the device context.
    //
    dc.SetMapMode(MM_LOENGLISH);
    dc.SetTextAlign(TA_CENTER | TA_BOTTOM);
    dc.SetBkMode(TRANSPARENT);
    //
    // Draw the body of the ruler.
    //
    CBrush brush (RGB (255, 255, 0));
    CBrush* pOldBrush = dc.SelectObject(&brush);
    dc.Rectangle(100, -100, 1300, -200);
    dc.SelectObject(pOldBrush);
    //
    // Draw the tick marks and labels.
    //
    for (int i=125; i<1300; i+=25) {
        dc.MoveTo(i, -192);
        dc.LineTo(i, -200);
    }

    for (i=150; i<1300; i+=50) {
        dc.MoveTo(i, -184);
        dc.LineTo(i, -200);
    }

    for (i=200; i<1300; i+=100) {
        dc.MoveTo(i, -175);
        dc.LineTo(i, -200);

        CString string;
        string.Format(_T ("%d"), (i / 100) - 1);
        dc.TextOut(i, -175, string);
    }
}
```

Màn hình kết quả như sau:



1.3 XỬ LÝ BÀN PHÍM/CHUỘT TRONG ỨNG DỤNG WINDOWS

1.3.1 Vấn đề quan tâm

- Các thông điệp (*message*) và sự kiện của chuột, bàn phím

- Nhận biết và xử lý các thao tác liên quan đến các sự kiện của chuột, bàn phím.

1.3.2 Các sự kiện của chuột

MFC hỗ trợ thao tác với chuột bằng việc liên hệ các thông điệp và các sự kiện như:

Thông điệp	Macro kết hợp	Sự kiện	Hàm điều khiển
WM_LBUTTONDOWN	*ON_WM_LBUTTONDOWN*	Nút trái chuột được nhấn xuống.	OnLButtonDown
WM_LBUTTONUP	*ON_WM_LBUTTONUP*	Nút trái chuột được nhả ra.	OnLButtonUp
WM_LBUTTONDBLCLK	*ON_WM_LBUTTONDBLCLK*	Nút trái chuột được nhấn kép(double)	OnLButtonDblClk
WM_MBUTTONDOWN	*ON_WM_MBUTTONDOWN*	Nút giữa chuột được nhấn xuống.	OnMButtonDown
WM_MBUTTONUP	*ON_WM_MBUTTONUP*	Nút giữa chuột được nhả ra.	OnMButtonUp
WM_MBUTTONDBLCLK	*ON_WM_MBUTTONDBLCLK*	Nút giữa chuột được nhấn kép(double)	OnMButtonDblClk
WM_RBUTTONDOWN	*ON_WM_RBUTTONDOWN*	Nút phải chuột được nhấn xuống.	OnRButtonDown
WM_RBUTTONUP	*ON_WM_RBUTTONUP*	Nút phải chuột được nhả ra.	OnRButtonUp
WM_RBUTTONDBLCLK	*ON_WM_RBUTTONDBLCLK*	Nút phải chuột được nhấn kép(double)	OnRButtonDblClk
WM_MOUSEMOVE	*ON_WM_MOUSEMOVE*	Con trỏ chuột di chuyển trong vùng hiển thị của cửa sổ.	OnMouseMove

Các hàm điều khiển có dạng:

```
afx_msg void OnMsgName(UINT nFlags, CPoint point)
```

Trong đó nFlags(dùng để nhận biết nút bấm chuột và các phím Ctrl/Shift bởi phép toán &) có giá trị:

Mặt nạ (Mask)	Nhận biết khi
MK_LBUTTON	Nút trái chuột được nhấn
MK_MBUTTON	Nút giữa chuột được nhấn
MK_RBUTTON	Nút phải chuột được nhấn
MK_CONTROL	Phím Ctrl được nhấn
MK_SHIFT	Phím Shift được nhấn

Ví dụ:

Để nhận biết phím Ctrl có được nhấn kèm với chuột dùng:

```
if ((nFlags & MK_CONTROL) == MK_CONTROL) {…}
```

Ví dụ:

Kết hợp xử lý chuột và vẽ hình:

```
void CVd3aView::OnMouseMove(UINT nFlags, CPoint point)
{
```

```
	// TODO: Add your message handler code here and/or call default
	CDC *pDC = GetDC();
	CString szTextOut;
	szTextOut.Format("Current point(%d, %d)", point.x, point.y);
	pDC->TextOut(0, 0, szTextOut);
	if((nFlags & MK_CONTROL)==MK_CONTROL)
		MyDrawFunction(point);
	CView::OnMouseMove(nFlags, point);
}
void CVd3aView::OnLButtonDown(UINT nFlags, CPoint point)
{
	// TODO: Add your message handler code here and/or call default
	MyDrawFunction(point);
	CView::OnLButtonDown(nFlags, point);
}
void CVd3aView::MyDrawFunction(CPoint point)
{
	CDC *pDC = GetDC();
	if((m_PrevPoint.x == -1)&&(m_PrevPoint.y == -1))
		pDC->MoveTo(point);
	else
	{
		pDC->MoveTo(m_PrevPoint);
		pDC->LineTo(point);
	}
	m_PrevPoint = point;
}
```

1.3.3 Các sự kiện của bàn phím

MFC hỗ trợ thao tác với bàn phím bằng việc liên hệ các thông điệp và các sự kiện như sau:

Thông điệp	Macro kết hợp	Sự kiện	Hàm điều khiển
WM_KEYDOWN	*ON_WM_KEYDOWN*	Phím được nhấn xuống.	OnKeyDown
WM_KEYUP	*ON_WM_KEYUP*	Phím được nhả ra.	OnKeyUp
WM_SYSKEYDOWN	*ON_WM_SYSKEYDOWN*	Nút chức năng được nhấn xuống.	OnSysKeyDown
WM_SYSKEYUP	*ON_WM_SYSKEYUP*	Phím chức năng được nhả ra.	OnSysKeyUp

Các hàm điều khiển có dạng:

```
afx_msg void OnMsgName(UINT nChar, UINT nRepCnt, UINT nFlags)
```

Trong đó, ngoài các ký tự thông thường, nChar(là Virtual Key Code) còn có giá trị như:

Mã phím ảo	Phím tương ứng
VK_F1_VK_F12	Function keys F1_F12
VK_NUMPAD0_VK_NUMPAD9	Numeric keypad 0_9 with Num Lock on
VK_CANCEL	Ctrl-Break
VK_RETURN	Enter
VK_BACK	Backspace
VK_TAB	Tab
VK_CLEAR	Numeric keypad 5 with Num Lock off
VK_SHIFT	Shift
VK_CONTROL	Ctrl
VK_MENU	Alt

VK_PAUSE	Pause
VK_ESCAPE	Esc
VK_SPACE	Spacebar
VK_PRIOR	Page Up and PgUp
VK_NEXT	Page Down and PgDn
VK_END	End
VK_HOME	Home
VK_LEFT	Left arrow
VK_UP	Up arrow
VK_RIGHT	Right arrow
VK_DOWN	Down arrow
VK_SNAPSHOT	Print Screen
VK_INSERT	Insert and Ins
VK_DELETE	Delete and Del
VK_MULTIPLY	Numeric keypad *
VK_ADD	Numeric keypad +
VK_SUBTRACT	Numeric keypad -
VK_DECIMAL	Numeric keypad .
VK_DIVIDE	Numeric keypad /
VK_CAPITAL	Caps Lock
VK_NUMLOCK	Num Lock
VK_SCROLL	Scroll Lock
VK_LWIN	Left Windows key
VK_RWIN	Right Windows key
VK_APPS	Menu key

Và nFlags được dùng để nhận biết phím Alt có được nhấn kèm hay không.

Ngoài ra, có thể dùng hàm:

void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)

Ví dụ:

```
void CVd3bView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	// TODO: Add your message handler code here and/or call default
	if(((nChar >= _T('A')) &&(nChar <= _T('Z'))) ||
		((nChar >= _T('a')) &&(nChar <= _T('z')))) {
		// Display the character
		m_szInput += _T(nChar);
		MyDrawFunction(m_szInput);
	}
	else if((nChar >= _T('0')) &&(nChar <= _T('9'))) {
		// Display the character
		m_szInput += _T(nChar);
		MyDrawFunction(m_szInput);
	}
	else if(nChar == VK_SPACE) {
		// Display the character
		m_szInput += _T(' ');
		MyDrawFunction(m_szInput);
	}
	else if(nChar == VK_RETURN) {
		// Process the Enter key
		m_nLine++;
	}
	else if(nChar == VK_BACK) {
```

```
            // Process the Backspace key
            m_szInput = m_szInput.Left(m_szInput.GetLength()-1);
            MyDrawFunction(m_szInput);
      }
      CView::OnChar(nChar, nRepCnt, nFlags);
}
void CVd3bView::MyDrawFunction(CString szText)
{
      CDC *pDC = GetDC();
      CBrush *pCurrentBrush = pDC->GetCurrentBrush();
      CBrush *pNewBrush = new CBrush();
      pNewBrush->CreateSolidBrush(RGB(255, 255, 255));
      CRect rect;
      GetClientRect(&rect);
      pDC->SelectObject(pNewBrush);
      pDC->FillRect(rect, pNewBrush);
      pDC->TextOut(0, 20*m_nLine, szText);
}
```

1.4 CÁC LỚP MFC COLLECTION: ARRAY, LIST, MAP*, TYPE POINTER MAP*

1.4.1 Vấn đề quan tâm

- Tìm hiểu cách sử dụng các MFC collection classes

1.4.2 Array collection

MFC cung cấp một số lớp hỗ trợ việc sử dụng array thuận tiện hơn như:

Tên lớp	Kiểu dữ liệu
CArray	Lớp mảng tổng quát nhất
CByteArray	Lớp mảng kiểu số nguyên 8-bit -bytes(BYTEs)
CWordArray	Lớp mảng kiểu số nguyên 16-bit -words(WORDs)
CDWordArray	Lớp mảng kiểu số nguyên 32-bit -double words(DWORDs)
CUIntArray	Lớp mảng kiểu số nguyên không âm -unsigned integers(UINTs)
CStringArray	Lớp mảng kiểu chuỗi CCtrings
CPtrArray	Lớp mảng kiểu con trỏ không kiểu -void pointers
CObArray	Lớp mảng kiểu con trỏ đối tượng -CObject pointers

Chú ý:

Người dùng phải khai báo **#include "Afxtempl.h"** khi sử dụng các lớp này.

Ví dụ 1:

```
// Add 10 items.
CUIntArray array;
array.SetSize(10);
for(int i=0; i<10; i++)
array[i] = i + 1;
// Remove the item at index 0.
array.RemoveAt(0);
TRACE(_T("Count = %d\n"), array.GetSize()); // 9 left.
// Remove items 0, 1, and 2.
array.RemoveAt(0, 3);
TRACE(_T("Count = %d\n"), array.GetSize()); // 6 left.
// Empty the array.
array.RemoveAll();
TRACE(_T("Count = %d\n"), array.GetSize()); // 0 left.
```

Ví dụ 2:

```
CArray<CPoint, CPoint&> array;
```

```
// Populate the array, growing it as needed.
for(int i=0; i<10; i++)
  array.SetAtGrow(i, CPoint(i*10, 0));
// Enumerate the items in the array.
int nCount = array.GetSize();
for(i=0; i<nCount; i++) {
  CPoint point = array[i];
  TRACE(_T("x=%d, y=%d\n"), point.x, point.y);
}
```

Ví dụ 3:

```
class CProduct
{
public :
CString m_szProductName;
CString m_szCategoryName;
}
```

....

```
CArrar <CProduct, CProduct&> arrayProductList ;
CProduct oProduct ;
oProduct.m_szProductName = 'product 1';
oProduct.m_ szCategoryName = 'category 1';
arrayProductList.Add(oProduct) ;
oProduct.m_szProductName = 'product 2';
oProduct.m_ szCategoryName = 'category 2';
arrayProductList.Add(oProduct) ;
```

...

```
oProduct = arrayProductList.GetAt(0);
AfxMessageBox(oProduct.m_szProductName);
```

Ngoài ra, có thể khai báo một kiểu dữ liệu mới từ CArray như sau:

Ví dụ 4:

```
typedef CArray<UINT, UINT> CUIntArray;
```

Chú ý:

Định nghĩa mảng kiểu đối tượng cần phải có ký hiệu & trong thành phần thứ hai trong định nghĩa.

```
CArray <kiểu_đối_tượng, kiểu_đối_tượng&> array;
```

Định nghĩa mảng kiểu cơ sở không có ký hiệu & trong thành phần thứ hai trong định nghĩa.

```
CArray <kiểu_cơ_sở, kiểu_cơ_sở> array;
```

1.4.3 List collection

MFC cung cấp các lớp hỗ trợ truy xuất danh sách liên kết như sau:

Tên lớp	Kiểu dữ liệu
CList	Lớp danh sách liên kết tổng quát nhất
CObList	Lớp danh sách liên kết kiểu con trỏ đối tượng -CObject pointers
CPtrList	Lớp danh sách liên kết kiểu con trỏ không kiểu-void pointers
CStringList	Lớp danh sách liên kết kiểu CString

Chú ý:

Người dùng phải khai báo **#include "Afxtempl.h"** khi sử dụng các lớp này.

Ví dụ 1:

```
// Schools of the Southeastern Conference
TCHAR szSchools[10][20];
szSchools[0] = _T("Alabama"); szSchools[1] = _T("Arkansas");
szSchools[2] = _T("Florida"); szSchools[3] = _T("Georgia");
```

```
szSchools[4] = _T("Kentucky"); szSchools[5] = _T("Mississippi");
szSchools[6] = _T("Mississippi State"); szSchools[7] = _T("South Carolina");
szSchools[8] = _T("Tennessee"); szSchools[0] = _T("Vanderbilt") ;
CStringList list;
for(int i=0; i<10; i++)
  list.AddTail(szSchools[i]);
POSITION pos = list.GetHeadPosition();
while(pos != NULL) {
  CString string = list.GetNext(pos);
  TRACE(_T("%s\n"), string);
}
```

Ví dụ 2:

```
CList <CPoint, CPoint&> list;
// Populate the list.
for(int i=0; i<10; i++)
  list.AddTail(CPoint(i*10, 0));
// Enumerate the items in the list.
POSITION pos = list.GetHeadPosition();
While (pos != NULL) {
  CPoint point = list.GetNext(pos);
  TRACE(_T("x=%d, y=%d\n"), point.x, point.y);
}
```

1.4.4 Map

Một map, được xem như một dictionary, là một bảng gồm nhiều phần tử được xác định bởi khoá. MFC cung cấp tập các lớp hỗ trợ sử dụng map như:

Tên lớp	Mô tả
CMap	Lớp từ điển tổng quát nhất.
CMapWordToPtr	Lớp từ điển kiểu ánh xạ WORD với con trỏ
CMapPtrToWord	Lớp từ điển kiểu ánh xạ con trỏ với WORD
CMapPtrToPtr	Lớp từ điển kiểu ánh xạ con trỏ với con trỏ khác
CMapWordToOb	Lớp từ điển kiểu ánh xạ WORD với con trỏ đối tượng
CMapStringToOb	Lớp từ điển kiểu ánh xạ CString với con trỏ đối tượng
CMapStringToPtr	Lớp từ điển kiểu ánh xạ CString với con trỏ
CMapStringToString	Lớp từ điển kiểu ánh xạ CString với String

Chú ý:

Người dùng phải khai báo **#include "Afxtempl.h"** khi sử dụng các lớp này.

Ví dụ:

```
CMapStringToString map;
map[_T("Sunday")]   = _T("Dimanche");
map[_T("Monday")]   = _T("Lundi");
map[_T("Tuesday")]  = _T("Mardi");
map[_T("Wednesday")] = _T("Mercredi");
map[_T("Thursday")] = _T("Jeudi");
map[_T("Friday")]   = _T("Vendredi");
map[_T("Saturday")] = _T("Samedi");
POSITION pos = map.GetStartPosition();
while(pos != NULL) {
  CString strKey, strItem;
  map.GetNextAssoc(pos, strKey, strItem);
  TRACE(_T("Key=%s, Item=%s\n"), strKey, strItem);
}
```

1.4.5 Type pointer map

Tên lớp	Mô tả
CTypedPtrArray	Lớp quản lý mảng con trỏ

CTypedPtrList	Lớp quản lý danh sách liên kết con trỏ
CTypedPtrMap	Lớp quản lý từ điển con trỏ

Chú ý:

Người dùng phải khai báo **#include "Afxtempl.h"** khi sử dụng các lớp này.

Ví dụ:

```
CTypedPtrList <CObList, CLine*> list;
// Populate the list.
for(int i=0; i<10; i++) {
  int x = i * 10;
  CLine* pLine = new CLine(x, 0, x, 100);
  list.AddTail(pLine);
}
// Enumerate the items in the list.
POSITION pos = list.GetHeadPosition();
while(pos != NULL)
  CLine* pLine = list.GetNext(pos); // No casting!
```

1.5 TRUY XUẤT FILE (I/O) VÀ SERIALIZATION

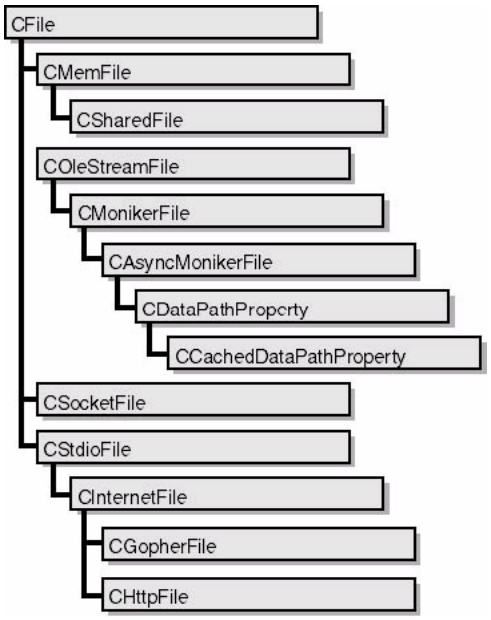
1.5.1 Vấn đề quan tâm

- Lớp CFile và thao tác truy xuất file
- Lớp CArchive các thao tác chuỗi hoá đối tượng lưu trữ.

1.5.2 Lớp CFile

MFC cung cấp lớp CFile hỗ trợ các thao tác truy xuất file như: tạo mới file, đọc/ghi file…

Tổ chức thừa kế lớp CFile và các lớp con như sau:



Việc truy xuất file liên quan đến chọn lựa "chia sẻ" quyền truy cập như:

Kiểu chia sẻ	Mô tả
CFile::shareDenyNone	Mở một file không ngăn cấm việc loại trừ
CFile::shareDenyRead	Cấm chương trình khác truy xuất đọc
CFile::shareDenyWrite	Cấm chương trình khác truy xuất ghi
CFile::shareExclusive	Cấm chương trình khác truy xuất đọc lẫn ghi

Và chọn lựa kiểu truy xuất như:

Kiểu truy xuất	Mô tả
CFile::modeReadWrite	Yêu cầu truy xuất đọc và ghi
CFile::modeRead	Yêu cầu truy xuất chỉ đọc
CFile::modeWrite	Yêu cầu truy xuất chỉ ghi

Ví dụ 1:

```
BYTE buffer[0x1000];
try
{
  CFile file(_T("C:\\MyDocument.txt"), CFile::modeReadWrite);
  DWORD dwBytesRemaining = file.GetLength();
  while(dwBytesRemaining)
  {
    DWORD dwPosition = file.GetPosition();
    UINT nBytesRead = file.Read(buffer, sizeof(buffer));
    ::CharLowerBuff((LPTSTR)buffer, nBytesRead);// chuyển sang chữ thường
    file.Seek(dwPosition, CFile::begin);
    file.Write(buffer, nBytesRead);
    dwBytesRemaining -= nBytesRead;
  }
}
catch(CFileException* e)
{
  e->ReportError();
  e->Delete();
}
file.Close();
```

Ví dụ 2:

Liệt kê tất cả các file trong thư mục hiện hành:

```
WIN32_FIND_DATA fd;
HANDLE hFind = ::FindFirstFile (_T ("*.*"), &fd);

if (hFind != INVALID_HANDLE_VALUE) {
    do {
        if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
            TRACE (_T ("%s\n"), fd.cFileName);
    } while (::FindNextFile (hFind, &fd));
    ::FindClose (hFind);
}
```

và liệt kê tất cả thư mục con trong thư mục hiện hành:

```
WIN32_FIND_DATA fd;
HANDLE hFind = ::FindFirstFile (_T ("*.*"), &fd);

if (hFind != INVALID_HANDLE_VALUE) {
    do {
        if (fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
            TRACE (_T ("%s\n"), fd.cFileName);
    } while (::FindNextFile (hFind, &fd));
    ::FindClose (hFind);
}
```

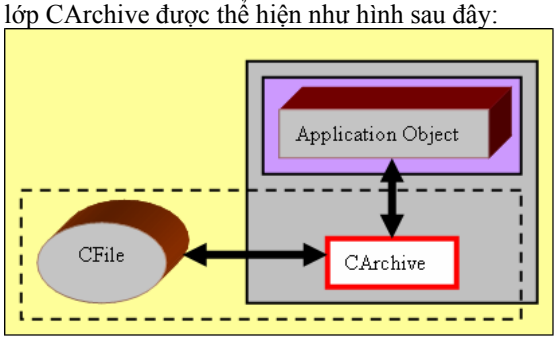
1.5.3 Chuỗi hoá và CArchive

Dù lớp CFile khá hữu ích trong quá trình xây dựng WinApp, nhưng MFC cung cấp lớp CArchive nhằm đơn giản hoá việc truy xuất với việc dùng toán tử >> và <<

Chuỗi hoá là một phần khá quan trọng trong lập trình MFC vì nó giúp chuyển đổi các đối tượng sang dạng dòng (*stream*) dữ liệu thuận lợi hơn trong quá trình lưu trữ/truy cập. Lớp CArchive được sử dụng trong hàm Serialize trên các đối tượng dữ liệu và tài liệu trong ứng dụng.

Trong hàm Serialize, để xác định đối tượng cần lưu trữ hiện tại đang được lưu xuống đĩa hay đang được đọc lên từ đĩa, cần sử dụng hàm thành phần **IsStoring()** và **IsLoading()**.

Sơ đồ tương tác và vai trò của lớp CArchive được thể hiện như hình sau đây:



Ví dụ 3:

```
void CVd3bDoc::Serialize(CArchive& ar)
{
	if(ar.IsStoring())
	{
		// TODO: add storing code here
		ar << m_MyVar;
	}
	else
	{
		// TODO: add loading code here
		ar >> m_MyVar;
	}
}
```

Hàm Serialize có thể được đặt trong bất cứ class nào nhằm xử lý thao tác lưu/đọc cho đối tượng đó. Để xử lý cho trường hợp này cho class (giả sử có tên CMyClass), cần thực thi theo trình tự sau:

- Cho lớp CMyClass này *kế thừa lớp CObject*.
- *Thêm macro DECLARE_SERIAL trong phần khai báo* của lớp CMyClass theo dạng sau: DECLARE_SERIAL(tên class thừa kế này). Ví dụ: DECLARE_SERIAL(CMyClass)
- *Thêm hàm Serialize* vào class CMyClass và khai báo lại hàm Serialize này để xử lý chuỗi hoá dữ liệu của lớp CMyClass
- *Thêm hàm sinh (constructor)* cho lớp CMyClass nếu nó chưa có.
- *Thêm macro IMPLEMENT_SERIAL trong phần thân* của lớp CMyClass theo dạng sau: IMPLEMENT_SERIAL(tên class thừa kế, tên class cha, số hiệu phiên bản). Ví dụ: IMPLEMENT_SERIAL(CMyClass, CObject, 1)

Ví dụ 4:

Bước 0:

```
class CLine
{
protected:
  CPoint m_ptFrom;
  CPoint m_ptTo;

public:
  CLine(CPoint from, CPoint to) { m_ptFrom = from; m_ptTo = to; }
};
```

Bước 1-2:

```
class CLine : public CObject
{
DECLARE_SERIAL(CLine)

protected:
  CPoint m_ptFrom;
  CPoint m_ptTo;
```

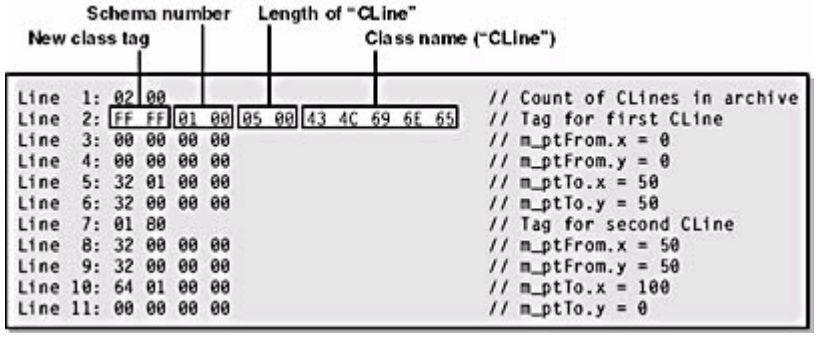
```
public:
  CLine() {} // Required!
  CLine(CPoint from, CPoint to) { m_ptFrom = from; m_ptTo = to; }
  void Serialize(CArchive& ar);
};
```

Bước 3:

```
void CLine::Serialize(CArchive& ar)
{
  CObject::Serialize(ar);
  if(ar.IsStoring())
    ar << m_ptFrom << m_ptTo;
  else // Loading, not storing
    ar >> m_ptFrom >> m_ptTo;
}
```

Bước 5a:

```
_AFX_INLINE CArchive& AFXAPI operator<<(CArchive& ar,
  const CObject* pOb)
  { ar.WriteObject(pOb); return ar; }
Bước 5b:
_AFX_INLINE CArchive& AFXAPI operator>>(CArchive& ar, CObject*& pOb)
  { pOb = ar.ReadObject(NULL); return ar; }

_AFX_INLINE CArchive& AFXAPI operator>>(CArchive& ar,
  const CObject*& pOb)
  { pOb = ar.ReadObject(NULL); return ar; }
```



Và có thể sử dụng để lưu trữ lớp CLine như sau:

Lưu xuống:

```
// Create two CLines and initialize an array of pointers.
CLine line1(CPoint(0, 0), CPoint(50, 50));
CLine line2(CPoint(50, 50), CPoint(100, 0));
CLine* pLines[2] = { &line1, &line2 };
int nCount = 2;

// Serialize the CLines and the CLine count.
for(int i=0; i<nCount; i++)
  ar << pLines[i];
// Lưu thêm số lượng đối tượng
ar << nCount;
```

Đọc lên:

```
int nCount;
ar >> nCount;
CLine* pLines = new CLine[nCount];
for(int i=0; i<nCount; i++)
```

```
  ar >> pLines[i];
// đọc lên số phần tử thực có
ar >> nCount;
```

Và cách thực thi khác:

```
// Serialize.
CLine line(CPoint(0, 0), CPoint(100, 50));
line.Serialize(ar);

// Deserialize.
CLine line;
line.Serialize(ar);
```

Ví dụ 5:

- Ghi vào file thông qua bộ xử lý chuỗi hoá:

```
try
{
	CFile myFile("C:\\myfile.dat", CFile::modeCreate  | CFile::modeWrite);
	// Create a storing archive.
	CArchive arStore(&myFile, CArchive::store);
	////////////////////////////////////
	CString szSize;
	szSize.Format("%d", (m_arrDSSV.GetSize()+1));
	arStore.Write(szSize, 10);
	////////////////////////////////////
	int nCount;
	////////////////////////////////////
	for (nCount=0; nCount < m_arrDSSV.GetSize(); nCount++)
	{
		// Write the object to the archive
		arStore.WriteObject( m_arrDSSV[nCount] );
	}
	// Close the storing archive
	arStore.Close();
	return TRUE;
}
catch(CException *e)
{
	return FALSE;
}
```

- đọc từ file lên thông qua bộ xử lý phi chuỗi hoá

```
try
{
	////////////////////////////////////////////////
	XoaNoiDung();
	////////////////////////////////////////////////
	CFile myFile("C:\\myfile.dat", CFile::modeRead);
	// Create a loading archive.
	myFile.SeekToBegin();
	CArchive arLoad(&myFile, CArchive::load);
	CSinhVien* pSV;
	////////////////////////////////
	char szSize[10];
	arLoad.Read(szSize, 10);
	int nSize = (int) atoi(szSize);
	int nCount;
```

```
		////////////////////////////////
		for (nCount=0; nCount < nSize; nCount++)
		{
			// Verify the object is in the archive.
			pSV = (CSinhVien*)arLoad.ReadObject(RUNTIME_CLASS(CSinhVien));
			if (pSV)
				m_arrDSSV.Add(pSV);
		}
		////////////////////////////////
		arLoad.Close();
		return TRUE;
	}
	catch(CException *e)
	{
		return FALSE;
	}
```

1.6 CÁC LỚP MFC CỦA CÁC ĐIỀU KHIỂN WINDOWS.

1.6.1 Vấn đề quan tâm

- Tính chất/hoạt động và xử lý/thao tác với các điều khiển Windows (*Windows control*)

1.6.2 Các loại điều khiển

MFC cung cấp các lớp đại diện cho các Windows control như:

Kiểu điều khiển	WNDCLASS	MFC Class
Buttons	"BUTTON"	CButton
List boxes	"LISTBOX"	CListBox
Edit controls	"EDIT"	CEdit
Combo boxes	"COMBOBOX"	CComboBox
Scroll bars	"SCROLLBAR"	CScrollBar
Static controls	"STATIC"	CStatic

Đồng thời với việc tạo các điều khiển này thông qua thanh công cụ trong chế độ Resource View, người dùng cũng có thể tạo các điều khiển này bằng hàm thành viên **Create**.

Ví dụ 1:

```
CButton myButton1, myButton2, myButton3, myButton4;

// Create a push button.
myButton1.Create(_T("My button"), WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
  CRect(10,10,100,30), pParentWnd, 1);

// Create a radio button.
myButton2.Create(_T("My button"), WS_CHILD|WS_VISIBLE|BS_RADIOBUTTON,
  CRect(10,40,100,70), pParentWnd, 2);

// Create an auto 3-state button.
myButton3.Create(_T("My button"), WS_CHILD|WS_VISIBLE|BS_AUTO3STATE,
  CRect(10,70,100,100), pParentWnd, 3);

// Create an auto check box.
myButton4.Create(_T("My button"), WS_CHILD|WS_VISIBLE|BS_AUTOCHECKBOX,
  CRect(10,100,100,130), pParentWnd, 4);
```

Hay:

```
extern CWnd* pParentWnd;
// The pointer to my list box.
extern CListBox* pmyListBox;
```

```
pmyListBox->Create(WS_CHILD|WS_VISIBLE|LBS_STANDARD|WS_HSCROLL,
  CRect(10,10,200,200), pParentWnd, 1);
Và:
CEdit* pEdit = new CEdit;
  pEdit->Create(ES_MULTILINE | WS_CHILD | WS_VISIBLE | WS_TABSTOP | WS_BORDER,
   CRect(10, 10, 100, 100), this, 1);
```

Chú ý:

- Nên sử dụng các biến con trỏ để đại diện cho các điều hiển để tạo các điều khiển trong thời gian thực thi.
- Có thể xử lý việc tạo các điều khiển này trong thời gian thực thi bằng cách đặt chúng vào trong các hàm thành viên ::OnInitialUpdate() và hàm khởi tạo ::PreCreateWindow()

1.6.3 Loại CButton

Kiểu	Mô tả
BS_PUSHBUTTON	Tạo một điều khiển nút nhấn
BS_DEFPUSHBUTTON	Tạo một điều khiển nút nhấn và sử dụng trong hộp thoại với trạng thái mặc định khi nhấn phím Enter
BS_CHECKBOX	Tạo một điều khiển check box
BS_AUTOCHECKBOX	Tạo một điều khiển check box tự động chọn và bỏ chọn khi được bấm vào
BS_3STATE	Tạo một điều khiển check box 3 trạng thái.
BS_AUTO3STATE	Tạo một điều khiển check box 3 trạng thái tự động chọn và bỏ chọn khi được bấm vào
BS_RADIOBUTTON	Tạo một điều khiển radio button
BS_AUTORADIOBUTTON	Tạo một điều khiển radion button tự động chọn và bỏ chọn khi được bấm vào
BS_GROUPBOX	Tạo một nhóm các điều khiển

Và các kiểu riêng phần:

Kiểu	Mô tả
BS_LEFTTEXT	Đặt văn bản của các điều khiển radio button hay check box nằm bên trái của điều khiển
BS_RIGHTBUTTON	Tương tự như BS_LEFTTEXT nhưng về phía phải
BS_LEFT	Canh lề trái văn bản trong điều khiển
BS_CENTER	Canh lề giữa văn bản trong điều khiển
BS_RIGHT	Canh lề phải văn bản trong điều khiển
BS_TOP	Canh lề phía đỉnh trên văn bản trong điều khiển
BS_VCENTER	Canh lề giữa văn bản trong điều khiển theo chiều dọc
BS_BOTTOM	Canh lề phía dưới văn bản trong điều khiển
BS_MULTILINE	Xác định chế độ nhiều dòng

1.7 DIALOG BOX, COMMON DIALOG VÀ PROPERTY SHEET.

1.7.1 Vấn đề quan tâm

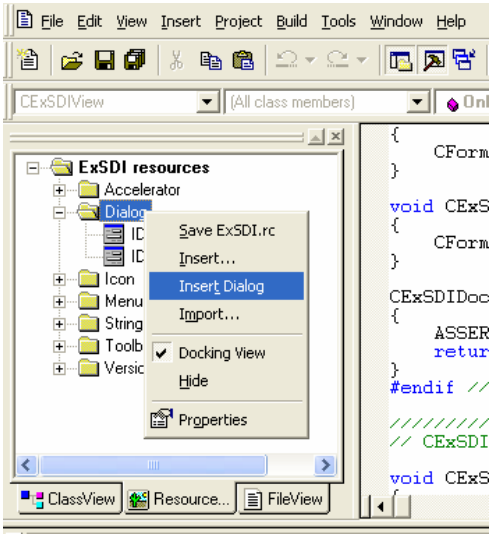
- Tính chất/hoạt động và xử lý/thao tác với các Dialog
- Cách tạo PropertySheet/PropertyPage

1.7.2 Hộp thoại (dialog)

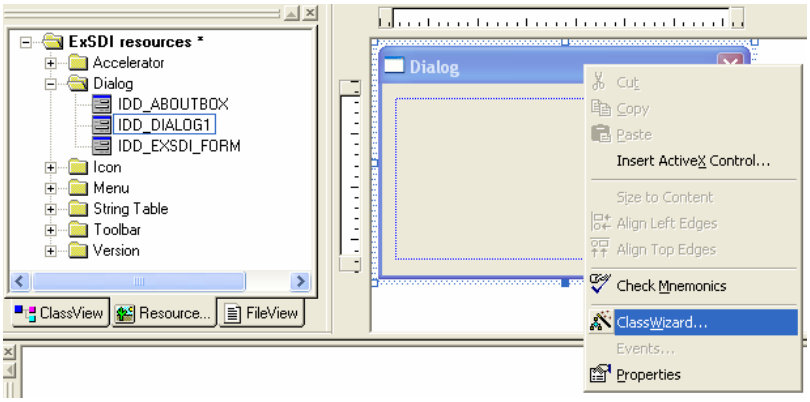
Có thể tạo các hộp thoại tùy ý bằng cách kết hợp giữa việc tạo Dialog form (trong ResourceView) và 1 class liên kết (tạo bởi ClassWizard)

Việc tạo này theo trình tự sau đây:

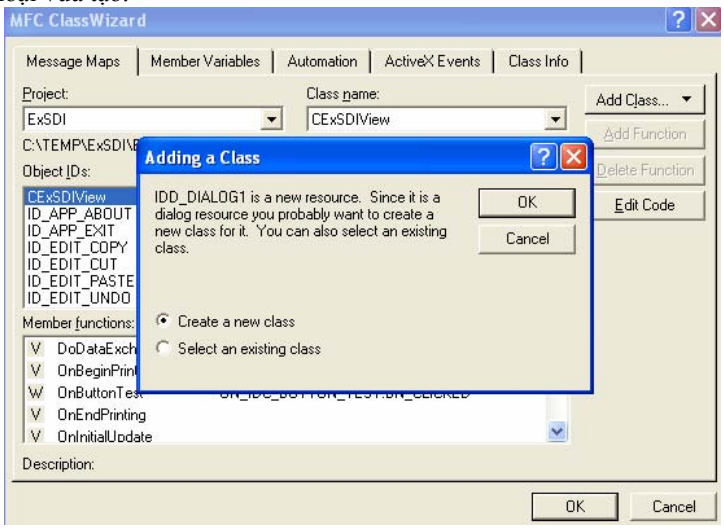
- Bước 1:



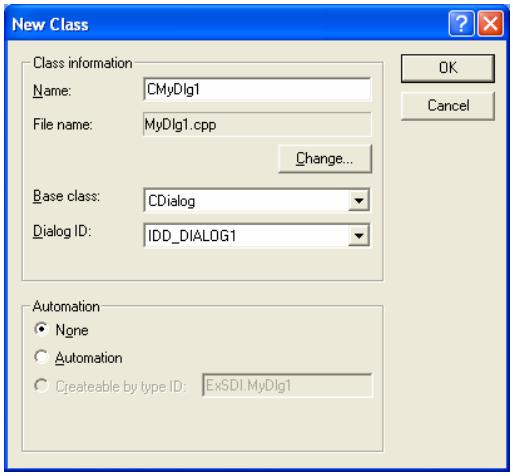
- Bước 2:



- Bước 3: Có thể cập nhật ID của hộp thoại mới thêm này, và chọn Create New Class để tạo một lớp liên kết với hộp thoại vừa tạo.



nhập thông tin cần thiết:



- Bước 4:

Tại file cần sử dụng hộp thoại này, thêm 1 hàng

```
#include "tên_file.h"
```

- Bước 5:

Tạo biến đại diện cho hộp thoại này và gọi hàm **DoModal()** của hộp thoại để kích hoạt hộp thoại hiển thị lên.

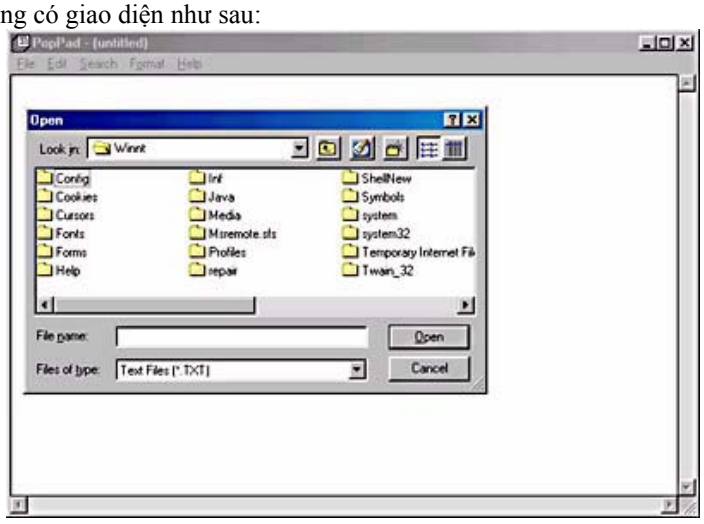
1.7.3 Các hộp thoại thông dụng (Common Dialog Classes)

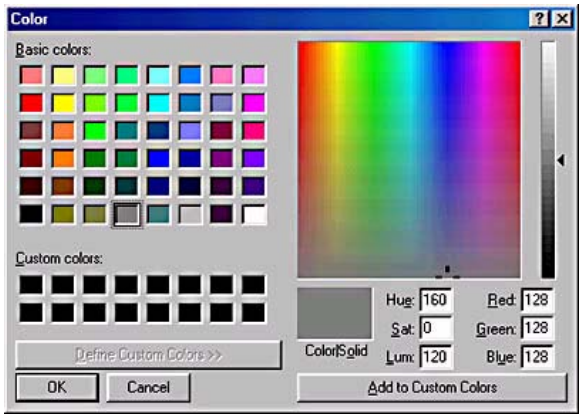
Các hộp thoại thông dụng như FontDialog, ColorDialog, FileDialog… được MFC cung cấp các class đại diện giúp việc sử dụng dễ dàng hơn.

MFC hỗ trợ các lớp đại diện cho các hộp thoại này như sau:

Tên lớp	Kiểu hộp thoại
CFileDialog	Hộp thoại mở và lưu trữ (Open and Save As dialog boxes)
CPrintDialog	Hộp thoại in ấn và cài đặt in ấn (Print and Print Setup dialog boxes)
CPageSetupDialog	Hộp thoại cài đặt trang (Page Setup dialog boxes)
CFindReplaceDialog	Hộp thoại tìm kiếm và thay thế (Find and Replace dialog boxes)
CColorDialog	Hộp thoại chọn màu (Color dialog boxes)
CFontDialog	Hộp thoại chọn phông chữ (Font dialog boxes)

Các hộp thoại thông dụng có giao diện như sau:





Ví dụ 1:

```
TCHAR szFilters[] = _T("Text files(*.txt)¦*.txt¦All files *.*)¦*.*¦¦");

CFileDialog
    dlg(TRUE,_T("txt"),_T("*.txt"),OFN_FILEMUSTEXIST¦OFN_HIDEREADONLY,zFilters);

if(dlg.DoModal() == IDOK) {
  filename = dlg.GetPathName();
  // Open the file and read it.
}
```

Ví dụ 2:

```
void CChildView::OnFileSaveAs()
{
CFileDialog
    dlg(FALSE,_T("phn"),m_strPathName,
          OFN_OVERWRITEPROMPT¦OFN_PATHMUSTEXIST¦OFN_HIDEREADONLY,
          m_szFilters);

if(dlg.DoModal() == IDOK)
{
       if(SaveFile(dlg.GetPathName()))
           m_strPathName = dlg.GetPathName();
}
```

Ví dụ 3:

```
CColorDialog dlg(RGB(255, 0, 0), CC_FULLOPEN);
if(dlg.DoModal() == IDOK)
{
       ////////////////////////////////////
       AfxMessageBox("Chon mau", 0, 0);
       m_oColor = dlg.GetColor();
}
```

1.7.4 Property Sheet/Property Page

Property Sheet còn gọi là tab, Property Page còn gọi là page.

PropertySheet được sử dụng nhằm cung cấp nhiều sự chọn lựa cho người dùng bằng cách tích hợp nhiều chọn lựa – mỗi chọn lựa là một hộp thoại (*property page*).

Để tạo Property Sheet/Property Page, cần làm theo các bước sau:

- Xác định số page (property page) cần có trong một property sheet
- Tạo (chọn) các class dự định dùng làm các page hiển thị trong property page – **kế thừa từ class CPropertyPage**.
- Thêm các control và các biến liên kết vào trong từng page và class liên kết ở bước trên.
- Tạo class mới cho property sheet bởi việc **kế thừa từ CPropertySheet class**

- Kích hoạt hàmDoModal()

Ví dụ 4:

- Tạo:

```
class CFirstPage : public CPropertyPage
{
public:
  CFirstPage() : CPropertyPage(IDD_FIRSTPAGE) {};
  // Declare CFirstPage's data members here.

protected:
  virtual void DoDataExchange(CDataExchange*);
};
class CSecondPage : public CPropertyPage
{
public:
  CSecondPage() : CPropertyPage(IDD_SECONDPAGE) {};
  // Declare CSecondPage's data members here.

protected:
  virtual void DoDataExchange(CDataExchange*);
};
class CMyPropertySheet : public CPropertySheet
{
public:
  CFirstPage m_firstPage;     // First page
  CSecondPage m_secondPage;    // Second page

  // Constructor adds the pages automatically.
  CMyPropertySheet(LPCTSTR pszCaption,
    CWnd* pParentWnd = NULL) :
    CPropertySheet(pszCaption, pParentWnd, 0)
  {
    AddPage(&m_firstPage);
    AddPage(&m_secondPage);
  }
};
```

- Sử dụng:

```
CMyPropertySheet ps(_T("Properties"));
ps.DoModal();
```

- Nếu muốn nhúng property sheet vào của sổ chương trình hiện hành thì cập nhập bởi các lệnh

```
CMyPropertySheet *ps = new CMyPropertySheet(_T("Properties"));
ps->Create(this, WS_CHILD);
ps->ShowWindow(SW_SHOW);
ps->MoveWindow(0,0,800,600);
```

Ví dụ tổng hợp:

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_)
```

```
#define AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "ChildView.h"

class CMainFrame : public CFrameWnd
{
public:
    CMainFrame();
protected:
    DECLARE_DYNAMIC(CMainFrame)
// Attributes
public:
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
        AFX_CMDHANDLERINFO* pHandlerInfo);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
    CChildView    m_wndView;
// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg void OnSetFocus(CWnd *pOldWnd);
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    afx_msg LRESULT OnApply (WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "PropDemo.h"
#include "MainFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_SETFOCUS()
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
    ON_MESSAGE (WM_USER_APPLY, OnApply)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    cs.lpszClass = AfxRegisterWndClass(0);
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
void CMainFrame::OnSetFocus(CWnd* pOldWnd)
{
    // forward focus to the view window
    m_wndView.SetFocus();
}
BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra,
    AFX_CMDHANDLERINFO* pHandlerInfo)
{
    // let the view have first crack at the command
    if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
        return TRUE;
    // otherwise, do default handling
```

```
    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
        CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
        return -1;
    return 0;
}
LRESULT CMainFrame::OnApply (WPARAM wParam, LPARAM lParam)
{
    m_wndView.SendMessage (WM_USER_APPLY, wParam, lParam);
    return 0;
}
```

ChildView.h

```
// ChildView.h : interface of the CChildView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
/////////////////////////////////////////////////////////////////////////////
// CChildView window
class CChildView : public CWnd
{
// Construction
public:
    CChildView();
// Attributes
public:
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildView)
    protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CChildView();
    // Generated message map functions
protected:
    int m_nUnits;
    int m_nHeight;
    int m_nWidth;
    int m_nColor;
    //{{AFX_MSG(CChildView)
```

```
    afx_msg void OnPaint();
    afx_msg void OnFileProperties();
    //}}AFX_MSG
    afx_msg LRESULT OnApply (WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
//!defined(AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__INCLUDED_)
```

ChildView.cpp

```
// ChildView.cpp : implementation of the CChildView class
//
#include "stdafx.h"
#include "PropDemo.h"
#include "ChildView.h"
#include "MyPropertySheet.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CChildView
CChildView::CChildView()
{
    m_nWidth = 4;
    m_nHeight = 2;
    m_nUnits = 0;
    m_nColor = 0;
}
CChildView::~CChildView()
{
}
BEGIN_MESSAGE_MAP(CChildView,CWnd )
    //{{AFX_MSG_MAP(CChildView)
    ON_WM_PAINT()
    ON_COMMAND(ID_FILE_PROPERTIES, OnFileProperties)
    //}}AFX_MSG_MAP
    ON_MESSAGE (WM_USER_APPLY, OnApply)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CChildView message handlers
BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CWnd::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
    cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
        ::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1), NULL);
```

```
    return TRUE;
}
void CChildView::OnPaint()
{
    CPaintDC dc(this); // Device context for painting.

    CBrush brush (CColorPage::m_clrColors[m_nColor]);
    CBrush* pOldBrush = dc.SelectObject (&brush);

    switch (m_nUnits) {
    case 0: // Inches.
        dc.SetMapMode (MM_LOENGLISH);
        dc.Ellipse (0, 0, m_nWidth * 100, -m_nHeight * 100);
        break;
    case 1: // Centimeters.
        dc.SetMapMode (MM_LOMETRIC);
        dc.Ellipse (0, 0, m_nWidth * 100, -m_nHeight * 100);
        break;
    case 2: // Pixels.
        dc.SetMapMode (MM_TEXT);
        dc.Ellipse (0, 0, m_nWidth, m_nHeight);
        break;
    }
    dc.SelectObject (pOldBrush);
}
void CChildView::OnFileProperties()
{
    CMyPropertySheet ps (_T ("Properties"));
    ps.m_sizePage.m_nWidth = m_nWidth;
    ps.m_sizePage.m_nHeight = m_nHeight;
    ps.m_sizePage.m_nUnits = m_nUnits;
    ps.m_colorPage.m_nColor = m_nColor;
    if (ps.DoModal () == IDOK) {
        m_nWidth = ps.m_sizePage.m_nWidth;
        m_nHeight = ps.m_sizePage.m_nHeight;
        m_nUnits = ps.m_sizePage.m_nUnits;
        m_nColor = ps.m_colorPage.m_nColor;
        Invalidate ();
    }
}
LRESULT CChildView::OnApply (WPARAM wParam, LPARAM lParam)
{
    ELLPROP* pep = (ELLPROP*) lParam;
    m_nWidth = pep->nWidth;
    m_nHeight = pep->nHeight;
    m_nUnits = pep->nUnits;
    m_nColor = pep->nColor;
    Invalidate ();
    return 0;
}
```

MyPropertySheet.h

```
#if
!defined(AFX_MYPROPERTYSHEET_H__418271A3_90D4_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MYPROPERTYSHEET_H__418271A3_90D4_11D2_8E53_006008A82731__INCLUDED_
```

```
#include "SizePage.h"    // Added by ClassView
#include "ColorPage.h"    // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// MyPropertySheet.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CmyPropertySheet
class CMyPropertySheet : public CPropertySheet
{
    DECLARE_DYNAMIC(CMyPropertySheet)

// Construction
public:
    CMyPropertySheet(UINT nIDCaption, CWnd* pParentWnd = NULL,
        UINT iSelectPage = 0);
    CMyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd = NULL,
        UINT iSelectPage = 0);
// Attributes
public:
    CColorPage m_colorPage;
    CSizePage m_sizePage;
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyPropertySheet)
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CMyPropertySheet();
    // Generated message map functions
protected:
    //{{AFX_MSG(CMyPropertySheet)
        // NOTE - the ClassWizard will add and remove
        // member functions here.
    //}}AFX_MSG
    afx_msg void OnApply ();
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//  AFX_MYPROPERTYSHEET_H__418271A3_90D4_11D2_8E53_006008A82731__INCLUDED_)
```

MyPropertySheet.cpp

```
// MyPropertySheet.cpp : implementation file
//
#include "stdafx.h"
#include "PropDemo.h"
#include "MyPropertySheet.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMyPropertySheet
IMPLEMENT_DYNAMIC(CMyPropertySheet, CPropertySheet)

CMyPropertySheet::CMyPropertySheet(UINT nIDCaption, CWnd* pParentWnd,
    UINT iSelectPage) : CPropertySheet(nIDCaption, pParentWnd, iSelectPage)
{
    AddPage (&m_sizePage);
    AddPage (&m_colorPage);
}
CMyPropertySheet::CMyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd,
    UINT iSelectPage) : CPropertySheet(pszCaption, pParentWnd, iSelectPage)
{
    AddPage (&m_sizePage);
    AddPage (&m_colorPage);
}
CMyPropertySheet::~CMyPropertySheet()
{
}
BEGIN_MESSAGE_MAP(CMyPropertySheet, CPropertySheet)
    //{{AFX_MSG_MAP(CMyPropertySheet)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
    ON_BN_CLICKED (ID_APPLY_NOW, OnApply)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMyPropertySheet message handlers
void CMyPropertySheet::OnApply ()
{
    GetActivePage ()->UpdateData (TRUE);

    ELLPROP ep;
    ep.nWidth = m_sizePage.m_nWidth;
    ep.nHeight = m_sizePage.m_nHeight;
    ep.nUnits = m_sizePage.m_nUnits;
    ep.nColor = m_colorPage.m_nColor;

    GetParent ()->SendMessage (WM_USER_APPLY, 0, (LPARAM) &ep);

    m_sizePage.SetModified (FALSE);
    m_colorPage.SetModified (FALSE);
}
```

SizePage.h

```
#if !defined(AFX_SIZEPAGE_H__418271A1_90D4_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SIZEPAGE_H__418271A1_90D4_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
// SizePage.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CSizePage dialog
class CSizePage : public CPropertyPage
{
    DECLARE_DYNCREATE(CSizePage)

// Construction
public:
    CSizePage();
    ~CSizePage();

// Dialog Data
    //{{AFX_DATA(CSizePage)
    enum { IDD = IDD_SIZE_PAGE };
    int        m_nWidth;
    int        m_nHeight;
    int        m_nUnits;
    //}}AFX_DATA
// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CSizePage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CSizePage)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    afx_msg void OnChange ();
   DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
//!defined(AFX_SIZEPAGE_H__418271A1_90D4_11D2_8E53_006008A82731__INCLUDED_)
```

SizePage.cpp

```
// SizePage.cpp : implementation file
//
#include "stdafx.h"
#include "PropDemo.h"
#include "SizePage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSizePage property page
```

```
IMPLEMENT_DYNCREATE(CSizePage, CPropertyPage)

CSizePage::CSizePage() : CPropertyPage(CSizePage::IDD)
{
    //{{AFX_DATA_INIT(CSizePage)
    m_nWidth = 0;
    m_nHeight = 0;
    m_nUnits = -1;
    //}}AFX_DATA_INIT
}
CSizePage::~CSizePage()
{
}
void CSizePage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSizePage)
    DDX_Text(pDX, IDC_WIDTH, m_nWidth);
    DDV_MinMaxInt(pDX, m_nWidth, 1, 128);
    DDX_Text(pDX, IDC_HEIGHT, m_nHeight);
    DDV_MinMaxInt(pDX, m_nHeight, 1, 128);
    DDX_Radio(pDX, IDC_INCHES, m_nUnits);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSizePage, CPropertyPage)
    //{{AFX_MSG_MAP(CSizePage)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
    ON_EN_CHANGE (IDC_WIDTH, OnChange)
    ON_EN_CHANGE (IDC_HEIGHT, OnChange)
    ON_BN_CLICKED (IDC_INCHES, OnChange)
    ON_BN_CLICKED (IDC_CENTIMETERS, OnChange)
    ON_BN_CLICKED (IDC_PIXELS, OnChange)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSizePage message handlers
void CSizePage::OnChange ()
{
    SetModified (TRUE);
}
```

ColorPage.h

```
#if !defined(AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ColorPage.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CColorPage dialog
class CColorPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CColorPage)
```

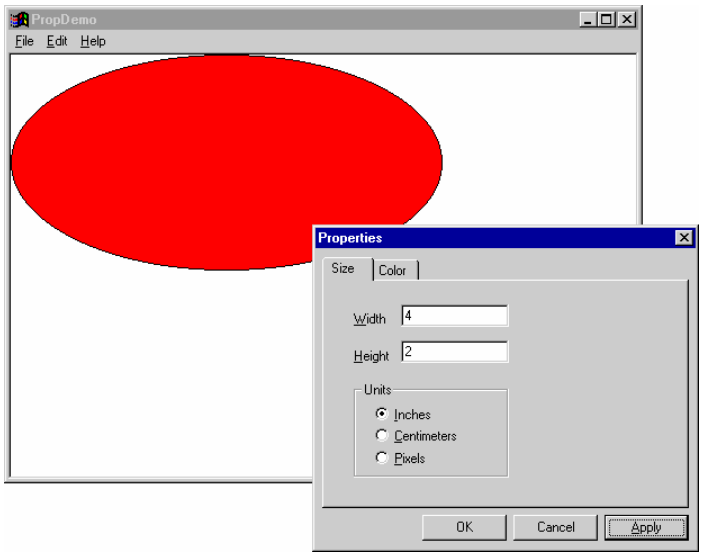
```
// Construction
public:
    CColorPage();
    ~CColorPage();
    static const COLORREF m_clrColors[3];
// Dialog Data
    //{{AFX_DATA(CColorPage)
    enum { IDD = IDD_COLOR_PAGE };
    int        m_nColor;
    //}}AFX_DATA
// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CColorPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CColorPage)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    afx_msg void OnChange ();
    DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
//defined(AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_)
```

ColorPage.cpp

```
// ColorPage.cpp : implementation file
//
#include "stdafx.h"#include "PropDemo.h"
#include "ColorPage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CColorPage property page
IMPLEMENT_DYNCREATE(CColorPage, CPropertyPage)

const COLORREF CColorPage::m_clrColors[3] = {
    RGB (255,   0,   0),    // Red
    RGB (  0, 255,   0),    // Green
    RGB (  0,   0, 255)     // Blue
};
CColorPage::CColorPage() : CPropertyPage(CColorPage::IDD)
{
    //{{AFX_DATA_INIT(CColorPage)
    m_nColor = -1;
```

```
    //}}AFX_DATA_INIT
}
CColorPage::~CColorPage()
{
}
void CColorPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CColorPage)
    DDX_Radio(pDX, IDC_RED, m_nColor);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CColorPage, CPropertyPage)
    //{{AFX_MSG_MAP(CColorPage)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
    ON_BN_CLICKED (IDC_RED, OnChange)
    ON_BN_CLICKED (IDC_GREEN, OnChange)
    ON_BN_CLICKED (IDC_BLUE, OnChange)
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////////
// CColorPage message handlers
void CColorPage::OnChange ()
{
    SetModified (TRUE);
}
```

Màn hình kết quả như sau:



1.8 Một số điều khiển trong Windows 9.x*

1.8.1 Các loại điều khiển

Các điều khiển được phát sinh thêm trong hệ điều hành Windows 95 (hay các hệ điều hành Windows mới hơn) có các lớp tương ứng trong MFC như sau:

Control Type	WNDCLASS	WNDCLASS Alias	MFC Class
Animation	"SysAnimate32"	ANIMATE_CLASS	*CAnimateCtrl*
ComboBoxEx*	"ComboBoxEx32"	WC_COMBOBOXEX	*CComboBoxEx*
Date-Time*	"SysDateTimePick32"	DATETIMEPICK_CLASS	*CDateTimeCtrl*

Header	"SysHeader32"	WC_HEADER	*CHeaderCtrl*
Hotkey	"msctls_hotkey32"	HOTKEY_CLASS	*CHotKeyCtrl*
Image list	N/A	N/A	*CImageList*
IP address**	"SysIPAddress32"	WC_IPADDRESS	*CIPAddressCtrl*
List view	"SysListView32"	WC_LISTVIEW	*CListCtrl*
Month calendar*	"SysMonthCal32"	MONTHCAL_CLASS	*CMonthCalCtrl*
Progress	"msctls_progress32"	PROGRESS_CLASS	*CProgressCtrl*
Property sheet	N/A	N/A	*CPropertySheet*
Rebar*	"ReBarWindow32"	REBARCLASSNAME	*CReBarCtrl*
Rich edit	"RichEdit20A" (ANSI) or "RichEdit20W" (Unicode)	RICHEDIT_CLASS	*CRichEditCtrl*
Slider	"msctls_trackbar32"	TRACKBAR_CLASS	*CSliderCtrl*
Spin button	"msctls_updown32"	UPDOWN_CLASS	*CSpinButtonCtrl*
Status bar	"msctls_statusbar32"	STATUSCLASSNAME	*CStatusBarCtrl*
Tab	"SysTabControl32"	WC_TABCONTROL	*CTabCtrl*
Toolbar	"ToolbarWindow32"	TOOLBARCLASSNAME	*CToolBarCtrl*
ToolTip	"tooltips_class32"	TOOLTIPS_CLASS	*CToolTipCtrl*
Tree view	"SysTreeView32"	WC_TREEVIEW	*CTreeCtrl*

1.8.2 Ví dụ tổng hợp:

1.8.2.1 Chương trình 1:

ChildView.h

```
// ChildView.h : interface of the CChildView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_CHILDVIEW_H__A4559BAA_ABE5_11D2_8E53_006008A82731__INCLUDED_)

#define AFX_CHILDVIEW_H__A4559BAA_ABE5_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CChildView window
class CChildView : public CWnd
{
// Construction
public:
    CChildView();

// Attributes
public:

// Operations
```

```
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildView)
    protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildView();

    // Generated message map functions
protected:
    int m_nWeight;
    int m_cy;
    int m_cx;
    //{{AFX_MSG(CChildView)
    afx_msg void OnPaint();
    afx_msg void OnOptionsGridSettings();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_CHILDVIEW_H__A4559BAA_ABE5_11D2_8E53_006008A82731__INCLUDED_)
```

ChildView.cpp

```
// ChildView.cpp : implementation of the CChildView class
//
#include "stdafx.h"
#include "GridDemo.h"
#include "ChildView.h"
#include "SettingsDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CChildView
CChildView::CChildView()
{
    m_cx = 8;
    m_cy = 8;
    m_nWeight = 4;
}
CChildView::~CChildView()
{
}
```

```
BEGIN_MESSAGE_MAP(CChildView,CWnd )
    //{{AFX_MSG_MAP(CChildView)
    ON_WM_PAINT()
    ON_COMMAND(ID_OPTIONS_GRID_SETTINGS, OnOptionsGridSettings)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CChildView message handlers
BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CWnd::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
    cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
        ::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1), NULL);

    return TRUE;
}
void CChildView::OnPaint()
{
    CRect rect;
    GetClientRect (&rect);

    int nShade = m_nWeight * 32;
    if (nShade != 0)
        nShade- -;

    CPaintDC dc (this);
    CPen pen (PS_SOLID, 1, RGB (nShade, nShade, nShade));
    CPen* pOldPen = dc.SelectObject (&pen);

    int x;
    for (int i=1; i<m_cx; i++) {
        x = (rect.Width () * i) / m_cx;
        dc.MoveTo (x, 0);
        dc.LineTo (x, rect.Height ());
    }

    int y;
    for (i=1; i<m_cy; i++) {
        y = (rect.Height () * i) / m_cy;
        dc.MoveTo (0, y);
        dc.LineTo (rect.Width (), y);
    }

    dc.SelectObject (pOldPen);
}
void CChildView::OnOptionsGridSettings()
{
    CSettingsDialog dlg;

    dlg.m_cx = m_cx;
    dlg.m_cy = m_cy;
```

```
    dlg.m_nWeight = m_nWeight;

    if (dlg.DoModal () == IDOK) {
        m_cx = dlg.m_cx;
        m_cy = dlg.m_cy;
        m_nWeight = dlg.m_nWeight;
        Invalidate ();
    }
}
```

SettingsDialog.h

```
#if !defined(
    AFX_SETTINGSDIALOG_H__A4559BB0_ABE5_11D2_8E53_006008A82731__INCLUDED_)
#define
    AFX_SETTINGSDIALOG_H__A4559BB0_ABE5_11D2_8E53_006008A82731__INCLUDED_

#include "MyToolTipCtrl.h"    // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SettingsDialog.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CSettingsDialog dialog
class CSettingsDialog : public CDialog
{
// Construction
public:
    int m_nWeight;
    CSettingsDialog(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CSettingsDialog)
    enum { IDD = IDD_SETTINGDLG };
    CSpinButtonCtrl    m_wndSpinVert;
    CSpinButtonCtrl    m_wndSpinHorz;
    CSliderCtrl    m_wndSlider;
    int        m_cx;
    int        m_cy;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSettingsDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    CMyToolTipCtrl m_ctlTT;
    // Generated message map functions
    //{{AFX_MSG(CSettingsDialog)
    virtual BOOL OnInitDialog();
    virtual void OnOK();
    //}}AFX_MSG
```

```
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//   AFX_SETTINGSDIALOG_H__A4559BB0_ABE5_11D2_8E53_006008A82731__INCLUDED_)
```

SettingsDialog.cpp

```
// SettingsDialog.cpp : implementation file
//
#include "stdafx.h"
#include "GridDemo.h"
#include "MyToolTipCtrl.h"
#include "SettingsDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSettingsDialog dialog
CSettingsDialog::CSettingsDialog(CWnd* pParent /*=NULL*/)
    : CDialog(CSettingsDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CSettingsDialog)
    m_cx = 0;
    m_cy = 0;
    //}}AFX_DATA_INIT
}
void CSettingsDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSettingsDialog)
    DDX_Control(pDX, IDC_SPINVERT, m_wndSpinVert);
    DDX_Control(pDX, IDC_SPINHORZ, m_wndSpinHorz);
    DDX_Control(pDX, IDC_SLIDER, m_wndSlider);
    DDX_Text(pDX, IDC_EDITHORZ, m_cx);
    DDX_Text(pDX, IDC_EDITVERT, m_cy);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSettingsDialog, CDialog)
    //{{AFX_MSG_MAP(CSettingsDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSettingsDialog message handlers
BOOL CSettingsDialog::OnInitDialog()
{
    CDialog::OnInitDialog();
    //
    // Initialize the slider control.
    //
```

```
    m_wndSlider.SetRange (0, 8);
    m_wndSlider.SetPos (m_nWeight);
    //
    // Initialize the spin button controls.
    //
    m_wndSpinHorz.SetRange (2, 64);
    m_wndSpinVert.SetRange (2, 64);
    //
    // Create and initialize a tooltip control.
    //
    m_ctlTT.Create (this);
    m_ctlTT.AddWindowTool (GetDlgItem (IDC_SLIDER),
        MAKEINTRESOURCE (IDS_SLIDER));
    m_ctlTT.AddWindowTool (GetDlgItem (IDC_EDITHORZ),
        MAKEINTRESOURCE (IDS_EDITHORZ));
    m_ctlTT.AddWindowTool (GetDlgItem (IDC_EDITVERT),
        MAKEINTRESOURCE (IDS_EDITVERT));
    return TRUE;
}
void CSettingsDialog::OnOK()
{
    //
    // Read the slider control's thumb position
    // before dismissing the dialog.
    //
    m_nWeight = m_wndSlider.GetPos ();
    CDialog::OnOK();
}
```

MyToolTipCtrl.h

```
#if !defined(
    AFX_MYTOOLTIPCTRL_H__A4559BB1_ABE5_11D2_8E53_006008A82731__INCLUDED_)
#define
    AFX_MYTOOLTIPCTRL_H__A4559BB1_ABE5_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// MyToolTipCtrl.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CMyToolTipCtrl window
class CMyToolTipCtrl : public CToolTipCtrl
{
// Construction
public:
    CMyToolTipCtrl();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
```

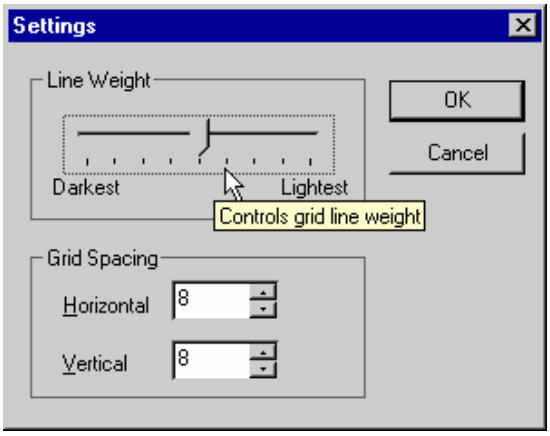
```
    //{{AFX_VIRTUAL(CMyToolTipCtrl)
    //}}AFX_VIRTUAL

// Implementation
public:
    BOOL AddRectTool (CWnd* pWnd, LPCTSTR pszText, LPCRECT pRect,
        UINT nIDTool);
    BOOL AddWindowTool (CWnd* pWnd, LPCTSTR pszText);
    virtual ~CMyToolTipCtrl();

    // Generated message map functions
protected:
    //{{AFX_MSG(CMyToolTipCtrl)
       // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//    AFX_MYTOOLTIPCTRL_H__A4559BB1_ABE5_11D2_8E53_006008A82731__INCLUDED_)
```

MyToolTipCtrl.cpp

```
// MyToolTipCtrl.cpp : implementation file
//
#include "stdafx.h"
#include "GridDemo.h"
#include "MyToolTipCtrl.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMyToolTipCtrl
CMyToolTipCtrl::CMyToolTipCtrl()
{
}
CMyToolTipCtrl::~CMyToolTipCtrl()
{
}
BEGIN_MESSAGE_MAP(CMyToolTipCtrl, CToolTipCtrl)
    //{{AFX_MSG_MAP(CMyToolTipCtrl)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMyToolTipCtrl message handlers
BOOL CMyToolTipCtrl::AddWindowTool(CWnd *pWnd, LPCTSTR pszText)
{
    TOOLINFO ti;
```

```
    ti.cbSize = sizeof (TOOLINFO);
    ti.uFlags = TTF_IDISHWND | TTF_SUBCLASS;
    ti.hwnd = pWnd->GetParent ()->GetSafeHwnd ();
    ti.uId = (UINT) pWnd->GetSafeHwnd ();
    ti.hinst = AfxGetInstanceHandle ();
    ti.lpszText = (LPTSTR) pszText;
    return (BOOL) SendMessage (TTM_ADDTOOL, 0, (LPARAM) &ti);
}
BOOL CMyToolTipCtrl::AddRectTool(CWnd *pWnd, LPCTSTR pszText,
    LPCRECT pRect, UINT nIDTool)
{
    TOOLINFO ti;
    ti.cbSize = sizeof (TOOLINFO);
    ti.uFlags = TTF_SUBCLASS;
    ti.hwnd = pWnd->GetSafeHwnd ();
    ti.uId = nIDTool;
    ti.hinst = AfxGetInstanceHandle ();
    ti.lpszText = (LPTSTR) pszText;
    ::CopyRect (&ti.rect, pRect);

    return (BOOL) SendMessage (TTM_ADDTOOL, 0, (LPARAM) &ti);
}
```

Màn hình kết quả như sau:



1.8.2.2 Chương trình 2:

PathListDlg.h

```
// PathListDlg.h : header file
//
#if !defined(
    AFX_PATHLISTDLG_H__710413E6_AC66_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_PATHLISTDLG_H__710413E6_AC66_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
/////////////////////////////////////////////////////////////////////////////
// CPathListDlg dialog
class CPathListDlg : public CDialog
{
// Construction
public:
    CPathListDlg(CWnd* pParent = NULL);    // standard constructor
// Dialog Data
```

```
    //{{AFX_DATA(CPathListDlg)
    enum { IDD = IDD_PATHLIST_DIALOG };
    CPathComboBox    m_wndCBEx;
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPathListDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    HICON m_hIcon;

    // Generated message map functions
    //{{AFX_MSG(CPathListDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnSelEndOK();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_PATHLISTDLG_H__710413E6_AC66_11D2_8E53_006008A82731__INCLUDED_)
```

PathListDlg.cpp

```
// PathListDlg.cpp : implementation file
//
#include "stdafx.h"
#include "PathList.h"
#include "PathComboBox.h"
#include "PathListDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CPathListDlg dialog
CPathListDlg::CPathListDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CPathListDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CPathListDlg)
    //}}AFX_DATA_INIT
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CPathListDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CPathListDlg)
```

```
    DDX_Control(pDX, IDC_CBEX, m_wndCBEx);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CPathListDlg, CDialog)
    //{{AFX_MSG_MAP(CPathListDlg)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_CBN_SELENDOK(IDC_CBEX, OnSelEndOK)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CPathListDlg message handlers
BOOL CPathListDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    SetIcon(m_hIcon, TRUE);
    SetIcon(m_hIcon, FALSE);
    //
    // Initialize the ComboBoxEx control.
    //
    TCHAR szPath[MAX_PATH];
    ::GetCurrentDirectory (sizeof (szPath) / sizeof (TCHAR), szPath);
    m_wndCBEx.SetPath (szPath);
    return TRUE;
}
void CPathListDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}
HCURSOR CPathListDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}
void CPathListDlg::OnSelEndOK()
```

```
{
    //
    // Display the path just selected from the ComboBoxEx control.
    //
    MessageBox (m_wndCBEx.GetPath ());
}
```

PathComboBox.h

```
#if !defined(
    AFX_PATHCOMBOBOX_H__710413F1_AC66_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_PATHCOMBOBOX_H__710413F1_AC66_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// PathComboBox.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CPathComboBox window
class CPathComboBox : public CComboBoxEx
{
// Construction
public:
    CPathComboBox();

// Attributes
public:
// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPathComboBox)
    //}}AFX_VIRTUAL

// Implementation
public:
    CString GetPath();
    BOOL SetPath (LPCTSTR pszPath);
    virtual ~CPathComboBox();

    // Generated message map functions
protected:
    void GetSubstring (int& nStart, CString& string, CString& result);
    int m_nIndexEnd;
    int m_nIndexStart;
    BOOL m_bFirstCall;
    CImageList m_il;
    //{{AFX_MSG(CPathComboBox)
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
```

```
// immediately before the previous line.
#endif
// !defined(
//     AFX_PATHCOMBOBOX_H__710413F1_AC66_11D2_8E53_006008A82731__INCLUDED_)
```

PathComboBox.cpp

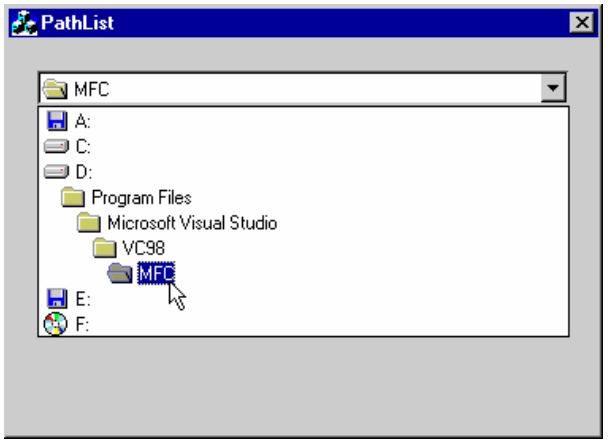
```
// PathComboBox.cpp : implementation file
//
#include "stdafx.h"
#include "PathList.h"
#include "PathComboBox.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CPathComboBox
CPathComboBox::CPathComboBox()
{
    m_bFirstCall = TRUE;
    m_nIndexStart = -1;
    m_nIndexEnd = -1;
}
CPathComboBox::~CPathComboBox()
{
}
BEGIN_MESSAGE_MAP(CPathComboBox, CComboBoxEx)
    //{{AFX_MSG_MAP(CPathComboBox)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CPathComboBox message handlers
BOOL CPathComboBox::SetPath(LPCTSTR pszPath)
{
    if (m_bFirstCall) {
        m_bFirstCall = FALSE;
        //
        // Add an image list containing drive and folder images.
        //
        m_il.Create (IDB_IMAGES, 16, 1, RGB (255, 0, 255));
        SetImageList (&m_il);
        //
        // Add icons representing the drives on the host system.
        //
        int nPos = 0;
        int nCount = 0;
        CString string = _T ("?:\\");

        DWORD dwDriveList = ::GetLogicalDrives ();

        while (dwDriveList) {
            if (dwDriveList & 1) {
                string.SetAt (0, _T (`A') + nPos);
                CString strDrive = string.Left (2);
```

```
                UINT nType = ::GetDriveType (string);

                int nImage = 0;
                switch (nType) {
                case DRIVE_FIXED:
                    nImage = 0;
                    break;
                case DRIVE_REMOVABLE:
                    nImage = 1;
                    break;
                case DRIVE_CDROM:
                    nImage = 2;
                    break;
                case DRIVE_REMOTE:
                    nImage = 3;
                    break;
                }

                COMBOBOXEXITEM cbei;
                cbei.mask = CBEIF_TEXT | CBEIF_IMAGE | CBEIF_SELECTEDIMAGE;
                cbei.iItem = nCount++;
                cbei.pszText = (LPTSTR) (LPCTSTR) strDrive;
                cbei.iImage = nImage;
                cbei.iSelectedImage = nImage;
                InsertItem (&cbei);
            }
            dwDriveList >>= 1;
            nPos++;
        }
    }

    //
    // Find the item that corresponds to the drive specifier in pszPath.
    //
    CString strPath = pszPath;
    CString strDrive = strPath.Left (2);

    int nDriveIndex = FindStringExact (-1, strDrive);
    if (nDriveIndex == CB_ERR)
        return FALSE;
    //
    // Delete previously added folder items (if any).
    //
    if (m_nIndexStart != -1 && m_nIndexEnd != -1) {
        ASSERT (m_nIndexEnd >= m_nIndexStart);
        int nCount = m_nIndexEnd - m_nIndexStart + 1;
        for (int i=0; i<nCount; i++)
            DeleteItem (m_nIndexStart);
        if (m_nIndexStart < nDriveIndex)
            nDriveIndex -= nCount;
        m_nIndexStart = -1;
        m_nIndexEnd = -1;
    }
    //
    // Add items representing the directories in pszPath.
```

```
    //
    int nCount = 0;
    int nStringIndex = strPath.Find (_T (`\\'), 0);

    if (nStringIndex++ != -1) {
        CString strItem;
        GetSubstring (nStringIndex, strPath, strItem);

        while (!strItem.IsEmpty ()) {
            COMBOBOXEXITEM cbei;
            cbei.mask = CBEIF_TEXT | CBEIF_IMAGE | CBEIF_SELECTEDIMAGE |
                CBEIF_INDENT;
            cbei.iItem = nDriveIndex + ++nCount;
            cbei.pszText = (LPTSTR) (LPCTSTR) strItem;
            cbei.iImage = 4;
            cbei.iSelectedImage = 5;
            cbei.iIndent = nCount;
            InsertItem (&cbei);

            GetSubstring (nStringIndex, strPath, strItem);
        }
    }
    //
    // Record the indexes of the items that were added, too.
    //
    if (nCount) {
        m_nIndexStart = nDriveIndex + 1;
        m_nIndexEnd = nDriveIndex + nCount;
    }
    //
    // Finish up by selecting the final item.
    //
    int nResult = SetCurSel (nDriveIndex + nCount);
    return TRUE;
}
void CPathComboBox::GetSubstring(int& nStart, CString &string,
    CString &result)
{
    result = _T ("");
    int nLen = string.GetLength ();
    if (nStart >= nLen)
        return;

    int nEnd = string.Find (_T (`\\'), nStart);
    if (nEnd == -1) {
        result = string.Right (nLen - nStart);
        nStart = nLen;
    }
    else {
        result = string.Mid (nStart, nEnd - nStart);
        nStart = nEnd + 1;
    }
}
CString CPathComboBox::GetPath()
{
```

```
    //
    // Get the index of the selected item.
    //
    CString strResult;
    int nEnd = GetCurSel ();
    int nStart = nEnd + 1;
    //
    // Find the index of the "root" item.
    //
    COMBOBOXEXITEM cbei;
    do {
        cbei.mask = CBEIF_INDENT;
        cbei.iItem = –nStart;
        GetItem (&cbei);
    } while (cbei.iIndent != 0);
    //
    // Build a path name by combining all the items from the root item to
    // the selected item.
    //
    for (int i=nStart; i<=nEnd; i++) {
        TCHAR szItem[MAX_PATH];
        COMBOBOXEXITEM cbei;
        cbei.mask = CBEIF_TEXT;
        cbei.iItem = i;
        cbei.pszText = szItem;
        cbei.cchTextMax = sizeof (szItem) / sizeof (TCHAR);
        GetItem (&cbei);
        strResult += szItem;
        strResult += _T ("\\");
    }
    //
    // Strip the trailing backslash.
    //
    int nLen = strResult.GetLength ();
    strResult = strResult.Left (nLen - 1);
    return strResult;
}
```

Màn hình kết quả như sau:



CHƯƠNG 2. CẤU TRÚC DOCUMENT-VIEW CỦA MFC WINDOWS APP

2.1 GIỚI THIỆU DOCUMENT-VIEW VÀ SDI (SINGLE DOCUMENT INTERFACE)

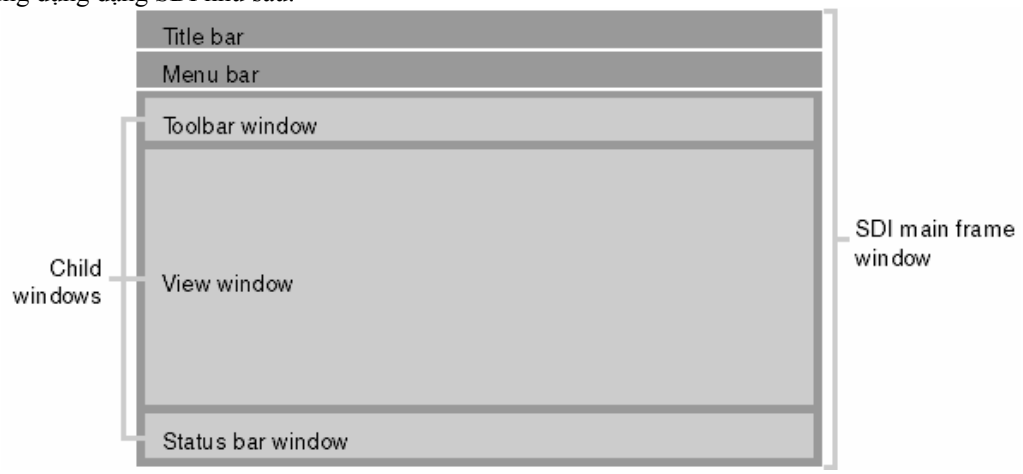
2.1.1 Vấn đề quan tâm

- Hiểu về các class thành phần trong 1 project.
- Biết và sử dụng đúng chức năng của các class về View, Document.

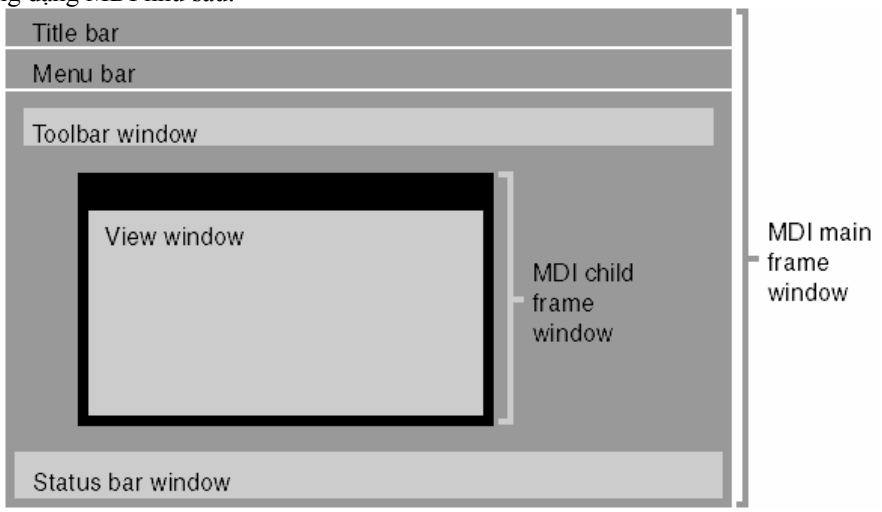
2.1.2 Giới thiệu

Kết cấu của ứng dụng dạng SDI và MDI được hợp thành bởi thành phần CView và CDocument, chính là cấu trúc Document/View.

Khung ứng dụng dạng SDI như sau:



Khung ứng dụng dạng MDI như sau:



Trong quá trình tạo ứng dụng dạng SDI và MDI, các class được tạo ra hầu hết được kế thừa (dẫn xuất) từ các class như:

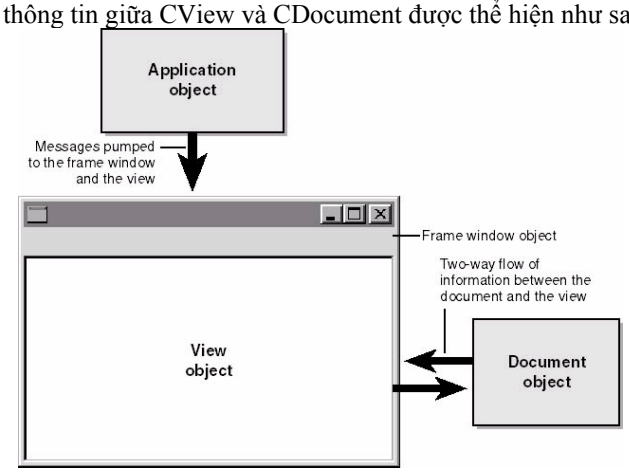
- Với dạng SDI, các class này là CWinApp, CFrameWnd, CDocument, CView
- Với dạng MDI, các class này là: CWinApp, CMDIFrameWnd, CMDIChildWnd, CDocument, CView

Vai trò của mỗi class được mô tả như sau:

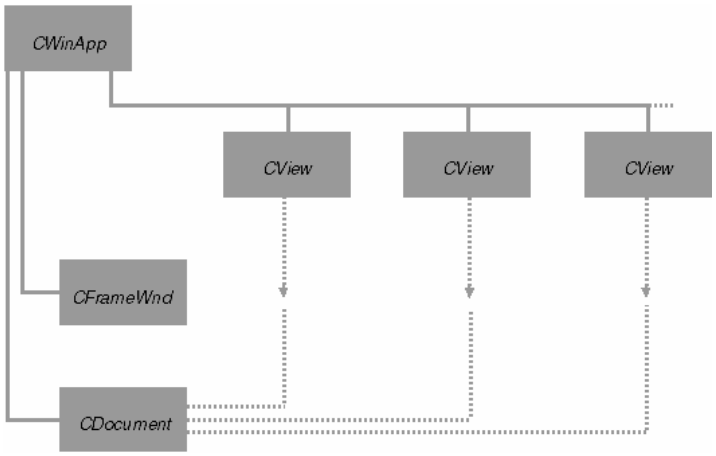
Lớp	Vai trò
CWinApp	Tạo tất cả thành phần khác trong ứng dụng. Đây là lớp nhận thông báo sự kiện và truyền tới lớp CView và CDocument
CFrameWnd	- Tạo và quản lý khung cửa sổ, chứa menu, toolbar, scrollbar và các đối tượng nhìn thấy được gắn vào cửa sổ.

	- Xác định số tài liệu nhìn thấy được trong một thời điểm bất kỳ
CMDIFrameWnd	- (thể hiện bởi CMainFrame trong project) là khung chính của ứng dụng - Cung cấp không gian bao bọc trên màn hình nền diễn ra toàn bộ tương tác ứng dụng. - Cửa sổ này là khung được gắn menu và thanh công cụ
CMDIChildFrameWnd	- (thể hiện bởi CChildFrame trong project) là khung chứa lớp CView - Là lớp truyền thông điệp (message) và sự kiện (event) tới các lớp hiển thị để xử lý hay hiển thị.
CDocument	- Chứa tài liệu của ứng dụng. Đây là nơi sẽ lưu trữ cấu trúc dữ liệu cần thiết để chứa và thao tác dữ liệu tạo nên tài liệu. - Lớp này nhận dữ liệu từ lớp CView và truyền thông tin sang lớp CView. - Lưu giữ và lấy dữ liệu về tài liệu từ các file
CView	- Quản lý phần hiển thị phần trình bày nhìn thấy của tài liệu cho người dùng xem. - Truyền thông tin vào lớp CDocument và nhận thông tin từ lớp CDocument.

Việc tương tác và trao đổi thông tin giữa CView và CDocument được thể hiện như sau:



hay:

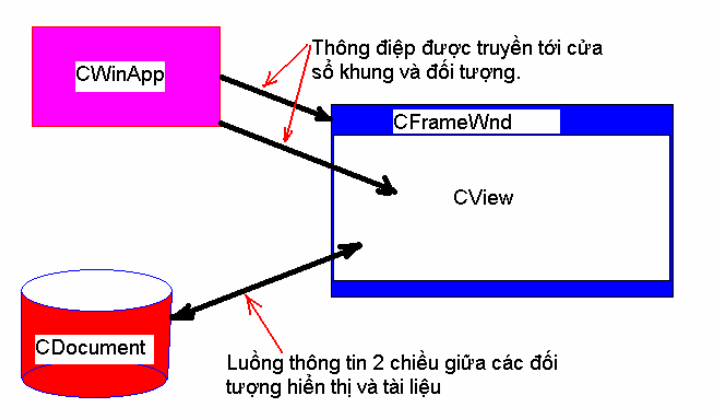


Đặc biệt, lớp CView cung cấp một số dạng giao diện đồ hoạ thông qua các lớp con như:

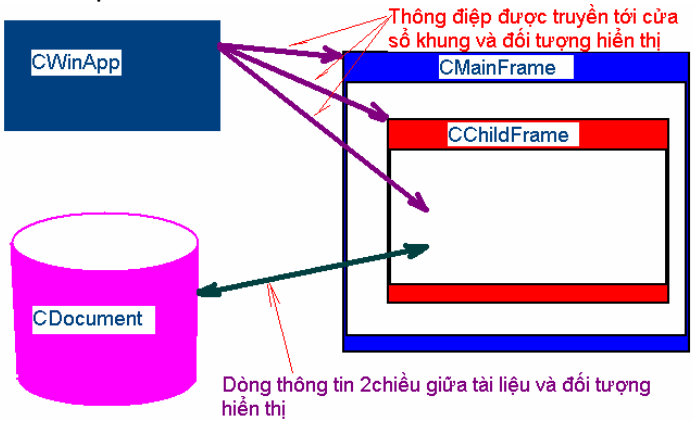
- ***CEditView***: cung cấp giao diện dạng editor soạn thảo
- ***CFormView***: cung cấp giao diện dạng form để gắn thêm các điều khiển
- ***CHtmlView***: cung cấp giao diện dạng trình duyệt web
- ***CListView***: cung cấp giao diện dạng danh sách
- ***CRichEditView***: cung cấp giao diện dạng editor soạn thảo có nhiều hỗ trợ
- ***CScrollView***: cung cấp giao diện có thanh cuộn
- ***CTreeView***: cung cấp giao diện dạng cây

2.1.3 Cấu trúc Document/View

Với dạng SDI, sự liên kết thể hiện như sau:



Với dạng MDI, sự liên kết thể hiện như sau:



2.1.4 Sự tương tác giữa phần Document và phần View

Với cả 2 dạng SDI và MDI, sự liên kết thể hiện thông qua 1 biến pointer kiểu CDocument (*biến này thường có tên là pDoc*). Thông qua biến này, chúng ta có thể điều khiển thao tác chuyển thông tin (*từ lớp CView sang CDocument*) hay lấy thông tin về (*từ lớp CDocument sang lớp CView*).

Thông thường, với cả ứng dụng dạng SDI hay MDI thì trong lớp CView luôn có hàm **GetDocument** nhằm hỗ trợ việc lấy biến con trỏ này.

Ví dụ 1:

```
CVd5bDoc* CVd5bView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CVd5bDoc)));
	return(CVd5bDoc*)m_pDocument;
}
```

và việc sử dụng có thể gọi hàm như sau:

```
void CVd5bView::OnButtonThem()
{
	CVd5bDoc* pDoc = GetDocument();

	// TODO: Add your control notification handler code here
	UpdateData(TRUE);
	pDoc->Them(m_nSTT, m_szMSSV, m_szHoTen, m_bPhai,
m_cboChuyenNganh.GetCurSel(), m_chkTheThao.GetCheck(), m_chkVanNghe.GetCheck());
	OnButtonReset();
	HienThiSoRecord();
}
```

2.2 CÁC KIỂU VIEW

2.2.1 Vấn đề quan tâm

- Hiểu về vai trò của các lớp con của CView.

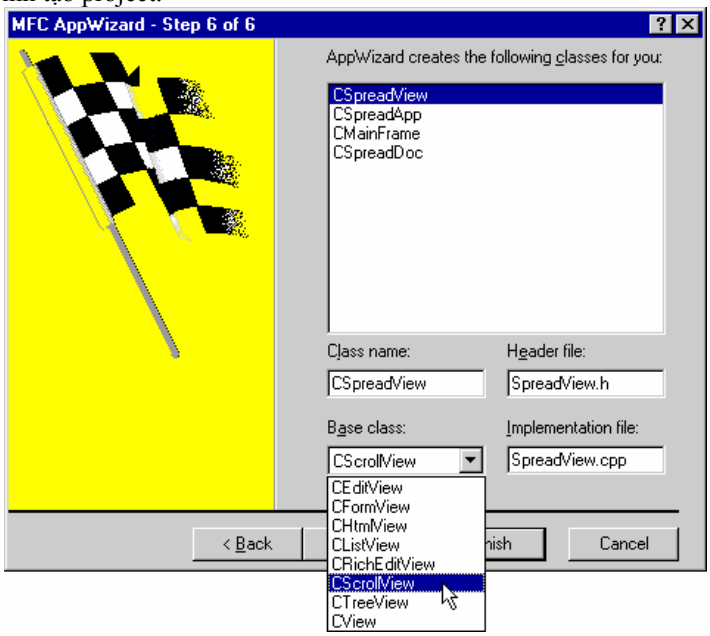
- Biết và sử dụng đúng chức năng của các class về View.

2.2.2 Giới thiệu

MFC cung cấp lớp CView và các lớp con để hỗ trợ vấn đề xây dựng giao diện như:

Tên lớp	Mô tả
CView	Lớp cơ sở
CCtrlView	Dùng làm lớp cơ sở cho các dạng view khác
CEditView	Cung cấp giao diện dạng editor
CRichEditView	Cung cấp giao diện dạng editor có nhiều hỗ trợ
CListView	Cung cấp giao diện dạng danh sách
CTreeView	Cung cấp giao diện dạng cây
CHtmlView	Cung cấp giao diện dạng trình duyệt web
CScrollView	Cung cấp giao diện có thanh cuộn
CFormView	Cung cấp giao diện dạng form để gắn các điều khiển
CRecordView	Cung cấp giao diện có tương tác điều khiển dữ liệu
CDaoRecordView	Cung cấp giao diện có tương tác điều khiển dữ liệu (dạng DAO)
COleDBRecordView	Cung cấp giao diện có tương tác điều khiển dữ liệu (dạng OLE)

Để có thể sử dụng các lớp này trong việc thể hiện giao diện của ứng dụng, chúng ta sẽ chọn chúng là lớp cơ sở trong bước 6 của quá trình tạo project.



2.2.3 Lớp CScrollView và ứng dụng

CScrollView được sử dụng trong trường hợp cần có giao diện hỗ trợ scrollbar.

Để xác định kích thước của vùng kích thước "luận lý" của cửa sổ, sử dụng hàm ảo (*virtual function*) **OnInitialUpdate** và hàm **SetScrollSizes**.

Ví dụ 1:

```
void CMyView::OnInitialUpdate()
{
  CScrollView::OnInitialUpdate();
  SetScrollSizes(MM_TEXT, CSize(1280, 1024));
}
```

hay

```
void CMyView::OnInitialUpdate()
{
  CScrollView::OnInitialUpdate();
  SetScrollSizes(MM_LOENGLISH, CSize(850, 1100));
}
```

Ngoài ra, có thể nhận biết được vị trí của scrollbar hiện hành hay ấn định vị trí của scrollbar bởi hàm **GetScrollPosition()** và **ScrollToPosition()**

Ví dụ 2:

```
CPoint pos = GetScrollPosition(); // lấy vị trí scrollbar hiện tại

ScrollToPosition(CPoint(100, 100));     // xác định vị trí scrollbar

CSize size = GetTotalSize();      // lấy kích thước cửa sổ hiện hành
int nWidth = size.cx;
int nHeight = size.cy;

SetScaleToFitSize(GetTotalSize());      // thay đổi tỉ lệ & kích thước
SetScrollSizes();          // khôi phục kích thước cửa sổ
```

Để chuyển đổi từ dạng CView sang dạng CScrollView, cần thực thi theo 2 bước sau:

- Tìm trong file .h và .cpp và thay thế CView bởi CScrollView
- Trong hàm OnInitialUpdate gọi SetScrollSize để xác định kích thước cửa sổ.

Ví dụ tổng hợp:

ScrollDemoView.h

```
// ScrollDemoView.h : interface of the CScrollDemoView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CScrollDemoView : public CScrollView
{
protected: // create from serialization only
    CScrollDemoView();
    DECLARE_DYNCREATE(CScrollDemoView)
// Attributes
public:
    CScrollDemoDoc* GetDocument();
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CScrollDemoView)
public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CScrollDemoView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
```

```
    BOOL m_bSmooth;
    void GetCellRect (int row, int col, LPRECT pRect);
    void DrawAddress (CDC* pDC, int row, int col);
    void DrawPointer (CDC* pDC, int row, int col, BOOL bHighlight);
    CFont m_font;
    int m_nCurrentCol;
    int m_nCurrentRow;
    int m_nRibbonWidth;
    int m_nCellHeight;
    int m_nCellWidth;
    //{{AFX_MSG(CScrollDemoView)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in ScrollDemoView.cpp
inline CScrollDemoDoc* CScrollDemoView::GetDocument()
   { return (CScrollDemoDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731__INCLUDED_)
```

ScrollDemoView.cpp

```
// ScrollDemoView.cpp : implementation of the CScrollDemoView class
//
#include "stdafx.h"
#include "ScrollDemo.h"
#include "ScrollDemoDoc.h"
#include "ScrollDemoView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView
IMPLEMENT_DYNCREATE(CScrollDemoView, CScrollView)

BEGIN_MESSAGE_MAP(CScrollDemoView, CScrollView)
    //{{AFX_MSG_MAP(CScrollDemoView)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView construction/destruction
CScrollDemoView::CScrollDemoView()
{
    m_font.CreatePointFont (80, _T ("MS Sans Serif"));
}
CScrollDemoView::~CScrollDemoView()
```

```
{
}
BOOL CScrollDemoView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView drawing
void CScrollDemoView::OnDraw(CDC* pDC)
{
    CScrollDemoDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    //
    // Draw the grid lines.
    //
    CSize size = GetTotalSize ();

    CPen pen (PS_SOLID, 0, RGB (192, 192, 192));
    CPen* pOldPen = pDC->SelectObject (&pen);
    for (int i=0; i<99; i++) {
        int y = (i * m_nCellHeight) + m_nCellHeight;
        pDC->MoveTo (0, y);
        pDC->LineTo (size.cx, y);
    }
    for (int j=0; j<26; j++) {
        int x = (j * m_nCellWidth) + m_nRibbonWidth;
        pDC->MoveTo (x, 0);
        pDC->LineTo (x, size.cy);
    }
    pDC->SelectObject (pOldPen);
    //
    // Draw the bodies of the rows and column headers.
    //
    CBrush brush;
    brush.CreateStockObject (LTGRAY_BRUSH);

    CRect rcTop (0, 0, size.cx, m_nCellHeight);
    pDC->FillRect (rcTop, &brush);
    CRect rcLeft (0, 0, m_nRibbonWidth, size.cy);
    pDC->FillRect (rcLeft, &brush);

    pDC->MoveTo (0, m_nCellHeight);
    pDC->LineTo (size.cx, m_nCellHeight);
    pDC->MoveTo (m_nRibbonWidth, 0);
    pDC->LineTo (m_nRibbonWidth, size.cy);

    pDC->SetBkMode (TRANSPARENT);
    //
    // Add numbers and button outlines to the row headers.
    //
    for (i=0; i<99; i++) {
        int y = (i * m_nCellHeight) + m_nCellHeight;
        pDC->MoveTo (0, y);
        pDC->LineTo (m_nRibbonWidth, y);
```

```
        CString string;
        string.Format (_T ("%d"), i + 1);

        CRect rect (0, y, m_nRibbonWidth, y + m_nCellHeight);
        pDC->DrawText (string, &rect, DT_SINGLELINE ¦
            DT_CENTER ¦ DT_VCENTER);

        rect.top++;
        pDC->Draw3dRect (rect, RGB (255, 255, 255),
            RGB (128, 128, 128));
    }
    //
    // Add letters and button outlines to the column headers.
    //
    for (j=0; j<26; j++) {
        int x = (j * m_nCellWidth) + m_nRibbonWidth;
        pDC->MoveTo (x, 0);
        pDC->LineTo (x, m_nCellHeight);

        CString string;
        string.Format (_T ("%c"), j + `A');

        CRect rect (x, 0, x + m_nCellWidth, m_nCellHeight);
        pDC->DrawText (string, &rect, DT_SINGLELINE ¦
            DT_CENTER ¦ DT_VCENTER);

        rect.left++;
        pDC->Draw3dRect (rect, RGB (255, 255, 255),
            RGB (128, 128, 128));
    }
    //
    // Draw address labels into the individual cells.
    //
    CRect rect;
    pDC->GetClipBox (&rect);
    int nStartRow = max (0, (rect.top - m_nCellHeight) / m_nCellHeight);
    int nEndRow = min (98, (rect.bottom - 1) / m_nCellHeight);
    int nStartCol = max (0, (rect.left - m_nRibbonWidth) / m_nCellWidth);
    int nEndCol = min (25, ((rect.right + m_nCellWidth - 1) -
        m_nRibbonWidth) / m_nCellWidth);

    for (i=nStartRow; i<=nEndRow; i++)
        for (j=nStartCol; j<=nEndCol; j++)
            DrawAddress (pDC, i, j);
    //
    // Draw the cell pointer.
    //
    DrawPointer (pDC, m_nCurrentRow, m_nCurrentCol, TRUE);
}
void CScrollDemoView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();

    m_nCurrentRow = 0;
    m_nCurrentCol = 0;
```

```
    m_bSmooth = FALSE;

    CClientDC dc (this);
    m_nCellWidth = dc.GetDeviceCaps (LOGPIXELSX);
    m_nCellHeight = dc.GetDeviceCaps (LOGPIXELSY) / 4;
    m_nRibbonWidth = m_nCellWidth / 2;

    int nWidth = (26 * m_nCellWidth) + m_nRibbonWidth;
    int nHeight = m_nCellHeight * 100;
    SetScrollSizes (MM_TEXT, CSize (nWidth, nHeight));
}
/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView diagnostics
#ifdef _DEBUG
void CScrollDemoView::AssertValid() const
{
    CScrollView::AssertValid();
}
void CScrollDemoView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}
CScrollDemoDoc* CScrollDemoView::GetDocument() // non-debug version is
                                               inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScrollDemoDoc)));
    return (CScrollDemoDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView message handlers
void CScrollDemoView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CScrollView::OnLButtonDown(nFlags, point);
    //
    // Convert the click point to logical coordinates.
    //
    CPoint pos = point;
    CClientDC dc (this);
    OnPrepareDC (&dc);
    dc.DPtoLP (&pos);
    //
    // If a cell was clicked, move the cell pointer.
    //
    CSize size = GetTotalSize ();
    if (pos.x > m_nRibbonWidth && pos.x < size.cx &&
        pos.y > m_nCellHeight && pos.y < size.cy) {

        int row = (pos.y - m_nCellHeight) / m_nCellHeight;
        int col = (pos.x - m_nRibbonWidth) / m_nCellWidth;
        ASSERT (row >= 0 && row <= 98 && col >= 0 && col <= 25);

        DrawPointer (&dc, m_nCurrentRow, m_nCurrentCol, FALSE);
        m_nCurrentRow = row;
        m_nCurrentCol = col;
```

```
        DrawPointer (&dc, m_nCurrentRow, m_nCurrentCol, TRUE);
    }
}
void CScrollDemoView::DrawPointer(CDC *pDC, int row, int col,
    BOOL bHighlight)
{
    CRect rect;
    GetCellRect (row, col, &rect);
    CBrush brush (bHighlight ? RGB (0, 255, 255) :
        ::GetSysColor (COLOR_WINDOW));
    pDC->FillRect (rect, &brush);
    DrawAddress (pDC, row, col);
}
void CScrollDemoView::DrawAddress(CDC *pDC, int row, int col)
{
    CRect rect;
    GetCellRect (row, col, &rect);

    CString string;
    string.Format (_T ("%c%d"), col + _T (`A'), row + 1);

    pDC->SetBkMode (TRANSPARENT);
    CFont* pOldFont = pDC->SelectObject (&m_font);
    pDC->DrawText (string, rect, DT_SINGLELINE ¦ DT_CENTER ¦ DT_VCENTER);
    pDC->SelectObject (pOldFont);
}
void CScrollDemoView::GetCellRect(int row, int col, LPRECT pRect)
{
    pRect->left = m_nRibbonWidth + (col * m_nCellWidth) + 1;
    pRect->top = m_nCellHeight + (row * m_nCellHeight) + 1;
    pRect->right = pRect->left + m_nCellWidth - 1;
    pRect->bottom = pRect->top + m_nCellHeight - 1;
}
```

2.2.4 Lớp CHtmlView và ứng dụng

CHtmlView được sử dụng trong trường hợp cần có giao diện hỗ trợ hiển thị trang web.

CHtmlView được xây dựng từ một số hàm tương tác với WebBrower control

Hàm	Mô tả
GetBusy	Trả về trạng thái chương trình có download nào đang hoạt động không
GetLocationName	Trả về tựa đề nếu là trang web; trả về tên file/thư mục nếu là đường dẫn vật lý
GetLocationURL	Trả về địa chỉ URL dạng http://.... hay file://...
GoBack	Quay về trang trước
GoForward	Đi tới trang kế
Navigate	Hiển thị nội dung của địa chỉ URL xác định
Refresh	Tải và hiển thị lại nội dung của địa chỉ URL hiện hành
Stop	Ngừng việc tải dữ liệu

Ví dụ 1:

```
// In CMyView's message map
ON_COMMAND(ID_BACK, OnBack)
ON_COMMAND(ID_FORWARD, OnForward)
ON_COMMAND(ID_REFRESH, OnRefresh)
ON_COMMAND(ID_STOP, OnStop)

void CMyView::OnBack()
{
```

```
  GoBack();
}
void CMyView::OnForward()
{
  GoForward();
}
void CMyView::OnRefresh()
{
  Refresh();
}
void CMyView::OnStop()
{
  Stop();
}
```

và:

```
Navigate(_T("http://www.microsoft.com"));
Navigate(_T("file://c:/my documents/budget.xls"));
```

Hàm	Mô tả
OnNavigateComplete2	Được gọi sau khi truy xuất một file
OnBeforeNavigate2	Được gọi sau khi truy xuất một file
OnProgressChange	Được gọi khi cung cấp trạng thái của quá trình tải dữ liệu
OnDownloadBegin	Được gọi khi quá trình tải dữ liệu bắt đầu
OnDownloadComplete	Được gọi khi quá trình tải dữ liệu hoàn tất
OnTitleChange	Được gọi khi tựa đề thay đổi
OnDocumentComplete	Được gọi khi tài liệu được tải hoàn tất

Ví dụ tổng hợp:

HtmlClockView.h

```
// HtmlClockView.h : interface of the CHtmlClockView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
//     AFX_HTMLCLOCKVIEW_H__D39825ED_99C0_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_HTMLCLOCKVIEW_H__D39825ED_99C0_11D2_8E53_006008A82731__INCLUDED_
#if _MSC_VER > 1000

#pragma once
#endif // _MSC_VER > 1000
class CHtmlClockView : public CHtmlView
{
protected: // create from serialization only
    CHtmlClockView();
    DECLARE_DYNCREATE(CHtmlClockView)

// Attributes
public:
    CHtmlClockDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CHtmlClockView)
    public:
```

```
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnTitleChange(LPCTSTR lpszText);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CHtmlClockView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CHtmlClockView)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in HtmlClockView.cpp
inline CHtmlClockDoc* CHtmlClockView::GetDocument()
   { return (CHtmlClockDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_HTMLCLOCKVIEW_H__D39825ED_99C0_11D2_8E53_006008A82731__INCLUDED_)
```

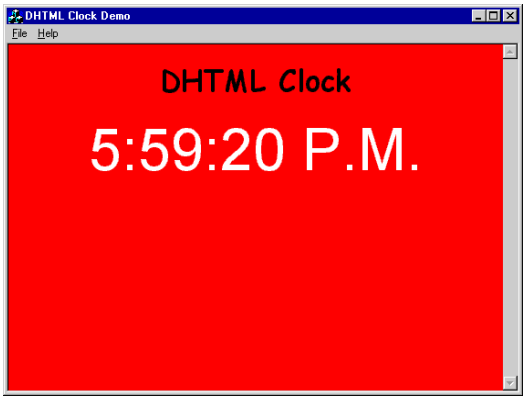
HtmlClockView.cpp

```
// HtmlClockView.cpp : implementation of the CHtmlClockView class
//
#include "stdafx.h"
#include "HtmlClock.h"
#include "HtmlClockDoc.h"
#include "HtmlClockView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CHtmlClockView
IMPLEMENT_DYNCREATE(CHtmlClockView, CHtmlView)

BEGIN_MESSAGE_MAP(CHtmlClockView, CHtmlView)
    //{{AFX_MSG_MAP(CHtmlClockView)
```

```
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CHtmlClockView construction/destruction
CHtmlClockView::CHtmlClockView()
{
}
CHtmlClockView::~CHtmlClockView()
{
}
BOOL CHtmlClockView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CHtmlView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CHtmlClockView drawing
void CHtmlClockView::OnDraw(CDC* pDC)
{
    CHtmlClockDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
}
void CHtmlClockView::OnInitialUpdate()
{
    CHtmlView::OnInitialUpdate();

    TCHAR szPath[MAX_PATH];
    ::GetModuleFileName (NULL, szPath, sizeof (szPath) / sizeof (TCHAR));

    CString string = szPath;
    int nIndex = string.ReverseFind (_T (`\\'));
    ASSERT (nIndex != -1);
    string = string.Left (nIndex + 1) + _T ("Clock.htm");
    Navigate (string);
}
/////////////////////////////////////////////////////////////////////////////
// CHtmlClockView diagnostics
#ifdef _DEBUG
void CHtmlClockView::AssertValid() const
{
    CHtmlView::AssertValid();
}
void CHtmlClockView::Dump(CDumpContext& dc) const
{
    CHtmlView::Dump(dc);
}
CHtmlClockDoc* CHtmlClockView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CHtmlClockDoc)));
    return (CHtmlClockDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CHtmlClockView message handlers
```

```
void CHtmlClockView::OnTitleChange(LPCTSTR lpszText)
{
    CHtmlView::OnTitleChange(lpszText);
    AfxGetMainWnd ()->SetWindowText (lpszText);
}
```

Màn hình kết quả như sau:



2.2.5 Lớp CTreeView và ứng dụng

CTreeView được sử dụng trong trường hợp cần có giao diện hỗ trợ hiển thị cây.

CTreeView được xây dựng liên quan đến TreeView control

Để điều khiển được cây trong giao diện chương trình, chúng ta sử dụng hàm **GetTreeCtrl** để lấy quyền điều khiển/truy xuất cây.

Ví dụ 1:

```
UINT nCount = GetTreeCtrl().GetCount(); // lấy số nút trên cây
```

Để tạo cây, có thể dùng các kiểu sau:

Kiểu	Mô tả
TVS_HASLINES	Adds lines connecting subitems to their parents.
TVS_LINESATROOT	Adds lines connecting items at the top level, or root, of the hierarchy. This style is valid only if TVS_HASLINES is also specified.
TVS_HASBUTTONS	Adds buttons containing plus or minus signs to items that have subitems. Clicking a button expands or collapses the associated subtree.
TVS_EDITLABELS	Enables in-place label editing notifications.
TVS_DISABLEDRAGDROP	Disables drag-and-drop notifications.
TVS_SHOWSELALWAYS	Specifies that the item that's currently selected should always be highlighted. By default, the highlight is removed when the control loses the input focus.

Và sử dụng các khai báo này trong hàm **PreCreateWindow**

Ví dụ 1:

```
BOOL CVd8cView::PreCreateWindow(CREATESTRUCT& cs) {
      // TODO: Modify the Window class or styles here by modifying       // the
CREATESTRUCT cs
cs.style |= TVS_HASLINES | TVS_LINESATROOT | TVS_HASBUTTONS |    TVS_SHOWSELALWAYS;
      return CTreeView::PreCreateWindow(cs);
}
```

Hàm **InsertItem** được dùng để thêm phần tử

Ví dụ 2:

```
// Root items first, with automatic sorting.
HTREEITEM hEagles = GetTreeCtrl().InsertItem(_T("Eagles"),
  TVI_ROOT, TVI_SORT);
HTREEITEM hDoobies = GetTreeCtrl().InsertItem(_T("Doobie Brothers"),
  TVI_ROOT, TVI_SORT);
```

```
// Eagles subitems second(no sorting).
GetTreeCtrl().InsertItem(_T("Eagles"), hEagles);
GetTreeCtrl().InsertItem(_T("On the Border"), hEagles);
GetTreeCtrl().InsertItem(_T("Hotel California"), hEagles);
GetTreeCtrl().InsertItem(_T("The Long Run"), hEagles);

// Doobie subitems third(no sorting).
GetTreeCtrl().InsertItem(_T("Toulouse Street"), hDoobies);
GetTreeCtrl().InsertItem(_T("The Captain and Me"), hDoobies);
GetTreeCtrl().InsertItem(_T("Stampede"), hDoobies);
```

Ví dụ tổng hợp:

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_MAINFRM_H__090B3829_959D_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__090B3829_959D_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_MAINFRM_H__090B3829_959D_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "DriveTree.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.style &= ~FWS_ADDTOTITLE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
```

```
// CMainFrame message handlers
```

DriveView.h

```
// DriveTreeView.h : interface of the CDriveView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CDriveView : public CTreeView
{
protected: // create from serialization only
    CDriveView();
    DECLARE_DYNCREATE(CDriveView)

// Attributes
public:
    CDriveTreeDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CDriveView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CDriveView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:

// Generated message map functions
protected:
    BOOL AddDriveItem (LPCTSTR pszDrive);
    int AddDirectories (HTREEITEM hItem, LPCTSTR pszPath);
    void DeleteAllChildren (HTREEITEM hItem);
    void DeleteFirstChild (HTREEITEM hItem);
    CString GetPathFromItem (HTREEITEM hItem);
    BOOL SetButtonState (HTREEITEM hItem, LPCTSTR pszPath);
    int AddDrives ();
    CImageList m_ilDrives;
    //{{AFX_MSG(CDriveView)
    afx_msg void OnItemExpanding(NMHDR* pNMHDR, LRESULT* pResult);
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in DriveTreeView.cpp
inline CDriveTreeDoc* CDriveView::GetDocument()
   { return (CDriveTreeDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_)
```

DriveView.cpp

```
// DriveTreeView.cpp : implementation of the CDriveView class
//
#include "stdafx.h"
#include "DriveTree.h"

#include "DriveTreeDoc.h"
#include "DriveView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Image indexes
#define ILI_HARD_DISK       0
#define ILI_FLOPPY          1
#define ILI_CD_ROM          2
#define ILI_NET_DRIVE       3
#define ILI_CLOSED_FOLDER   4
#define ILI_OPEN_FOLDER     5
/////////////////////////////////////////////////////////////////////////////
// CDriveView
IMPLEMENT_DYNCREATE(CDriveView, CTreeView)

BEGIN_MESSAGE_MAP(CDriveView, CTreeView)
    //{{AFX_MSG_MAP(CDriveView)
    ON_NOTIFY_REFLECT(TVN_ITEMEXPANDING, OnItemExpanding)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CDriveView construction/destruction
CDriveView::CDriveView()
{
}
CDriveView::~CDriveView()
{
}
BOOL CDriveView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CTreeView::PreCreateWindow (cs))
```

```
        return FALSE;

    cs.style |= TVS_HASLINES | TVS_LINESATROOT | TVS_HASBUTTONS |
        TVS_SHOWSELALWAYS;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////
// CDriveView drawing
void CDriveView::OnDraw(CDC* pDC)
{
    CDriveTreeDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
}
void CDriveView::OnInitialUpdate()
{
    CTreeView::OnInitialUpdate();
    //
    // Initialize the image list.
    //
    m_ilDrives.Create (IDB_DRIVEIMAGES, 16, 1, RGB (255, 0, 255));
    GetTreeCtrl ().SetImageList (&m_ilDrives, TVSIL_NORMAL);
    //
    // Populate the tree view with drive items.
    //
    AddDrives ();
    //
    // Show the folders on the current drive.
    //
    TCHAR szPath[MAX_PATH];
    ::GetCurrentDirectory (sizeof (szPath) / sizeof (TCHAR), szPath);
    CString strPath = szPath;
    strPath = strPath.Left (3);

    HTREEITEM hItem = GetTreeCtrl ().GetNextItem (NULL, TVGN_ROOT);
    while (hItem != NULL) {
        if (GetTreeCtrl ().GetItemText (hItem) == strPath)
            break;
        hItem = GetTreeCtrl ().GetNextSiblingItem (hItem);
    }
    if (hItem != NULL) {
        GetTreeCtrl ().Expand (hItem, TVE_EXPAND);
        GetTreeCtrl ().Select (hItem, TVGN_CARET);
    }
}
/////////////////////////////////////////////////////////////////////////
// CDriveView diagnostics
#ifdef _DEBUG
void CDriveView::AssertValid() const
{
    CTreeView::AssertValid();
}
void CDriveView::Dump(CDumpContext& dc) const
{
    CTreeView::Dump(dc);
```

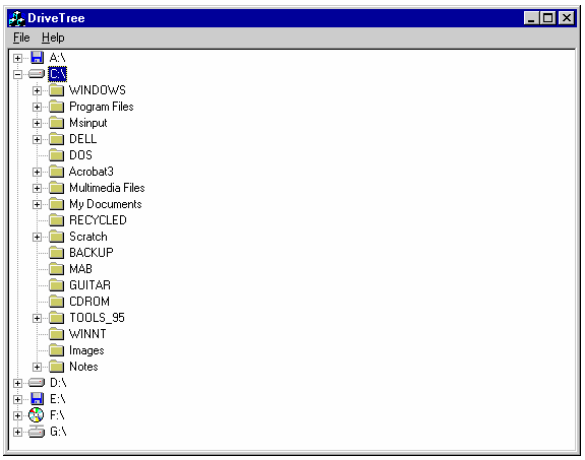
```
}
CDriveTreeDoc* CDriveView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CDriveTreeDoc)));
    return (CDriveTreeDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////
// CDriveView message handlers
int CDriveView::AddDrives()
{
    int nPos = 0;
    int nDrivesAdded = 0;
    CString string = _T ("?:\\");

    DWORD dwDriveList = ::GetLogicalDrives ();

    while (dwDriveList) {
        if (dwDriveList & 1) {
            string.SetAt (0, _T (`A') + nPos);
            if (AddDriveItem (string))
                nDrivesAdded++;
        }
        dwDriveList >>= 1;
        nPos++;
    }
    return nDrivesAdded;
}
BOOL CDriveView::AddDriveItem(LPCTSTR pszDrive)
{
    CString string;
    HTREEITEM hItem;

    UINT nType = ::GetDriveType (pszDrive);

    switch (nType) {
    case DRIVE_REMOVABLE:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_FLOPPY,
            ILI_FLOPPY);
        GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
            ILI_CLOSED_FOLDER, hItem);
        break;

    case DRIVE_FIXED:
    case DRIVE_RAMDISK:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_HARD_DISK,
            ILI_HARD_DISK);
        SetButtonState (hItem, pszDrive);
        break;

    case DRIVE_REMOTE:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_NET_DRIVE,
            ILI_NET_DRIVE);
        SetButtonState (hItem, pszDrive);
        break;
```

```
    case DRIVE_CDROM:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_CD_ROM,
            ILI_CD_ROM);
        GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
            ILI_CLOSED_FOLDER, hItem);
        break;

    default:
        return FALSE;
    }
    return TRUE;
}
BOOL CDriveView::SetButtonState(HTREEITEM hItem, LPCTSTR pszPath)
{
    HANDLE hFind;
    WIN32_FIND_DATA fd;
    BOOL bResult = FALSE;

    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    if ((hFind = ::FindFirstFile (strPath, &fd)) == INVALID_HANDLE_VALUE)
        return bResult;

    do {
        if (fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) {
            CString strComp = (LPCTSTR) &fd.cFileName;
            if ((strComp != _T (".")) && (strComp != _T (".."))) {
                GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                    ILI_CLOSED_FOLDER, hItem);
                bResult = TRUE;
                break;
            }
        }
    } while (::FindNextFile (hFind, &fd));

    ::FindClose (hFind);
    return bResult;
}
void CDriveView::OnItemExpanding(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    HTREEITEM hItem = pNMTreeView->itemNew.hItem;
    CString string = GetPathFromItem (hItem);

    *pResult = FALSE;

    if (pNMTreeView->action == TVE_EXPAND) {
        DeleteFirstChild (hItem);
        if (AddDirectories (hItem, string) == 0)
            *pResult = TRUE;
    }
```

```
    else { // pNMTreeView->action == TVE_COLLAPSE
        DeleteAllChildren (hItem);
        if (GetTreeCtrl ().GetParentItem (hItem) == NULL)
            GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                ILI_CLOSED_FOLDER, hItem);
        else
            SetButtonState (hItem, string);
    }
}
CString CDriveView::GetPathFromItem(HTREEITEM hItem)
{
    CString strResult = GetTreeCtrl ().GetItemText (hItem);

    HTREEITEM hParent;
    while ((hParent = GetTreeCtrl ().GetParentItem (hItem)) != NULL) {
        CString string = GetTreeCtrl ().GetItemText (hParent);
        if (string.Right (1) != _T ("\\"))
            string += _T ("\\");
        strResult = string + strResult;
        hItem = hParent;
    }
    return strResult;
}
void CDriveView::DeleteFirstChild(HTREEITEM hItem)
{
    HTREEITEM hChildItem;
    if ((hChildItem = GetTreeCtrl ().GetChildItem (hItem)) != NULL)
        GetTreeCtrl ().DeleteItem (hChildItem);
}
void CDriveView::DeleteAllChildren(HTREEITEM hItem)
{
    HTREEITEM hChildItem;
    if ((hChildItem = GetTreeCtrl ().GetChildItem (hItem)) == NULL)
        return;
    do {
        HTREEITEM hNextItem =
            GetTreeCtrl ().GetNextSiblingItem (hChildItem);
        GetTreeCtrl ().DeleteItem (hChildItem);
        hChildItem = hNextItem;
    } while (hChildItem != NULL);
}
int CDriveView::AddDirectories(HTREEITEM hItem, LPCTSTR pszPath)
{
    HANDLE hFind;
    WIN32_FIND_DATA fd;
    HTREEITEM hNewItem;

    int nCount = 0;
    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    if ((hFind = ::FindFirstFile (strPath, &fd)) == INVALID_HANDLE_VALUE) {
        if (GetTreeCtrl ().GetParentItem (hItem) == NULL)
```

```
            GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                ILI_CLOSED_FOLDER, hItem);
        return 0;
    }
   do {
        if (fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) {
            CString strComp = (LPCTSTR) &fd.cFileName;
            if ((strComp != _T (".")) && (strComp != _T ("..."))) {
                hNewItem =
                    GetTreeCtrl ().InsertItem ((LPCTSTR) &fd.cFileName,
                    ILI_CLOSED_FOLDER, ILI_OPEN_FOLDER, hItem);

                CString strNewPath = pszPath;
                if (strNewPath.Right (1) != _T ("\\"))
                    strNewPath += _T ("\\");

                strNewPath += (LPCTSTR) &fd.cFileName;
                SetButtonState (hNewItem, strNewPath);
                nCount++;
            }
        }
    } while (::FindNextFile (hFind, &fd));

    ::FindClose (hFind);
    return nCount;
}
```

Màn hình kết quả như sau:



2.2.6 Lớp CListView và ứng dụng

Tương tự các dạng view nêu trên, CListView giúp tạo giao diện dạng list

Việc điều khiển, tương tác với danh sách (*list*) được thực hiện thông qua việc gọi hàm **GetListCtrl** nhằm lấy quyền điều khiển

Các dạng list được chọn lựa theo các tùy chọn sau:

Kiểu	Mô tả
LVS_ICON	Selects large icon mode.
LVS_SMALLICON	Selects small icon mode.
LVS_LIST	Selects list mode.
LVS_REPORT	Selects report mode.
LVS_NOCOLUMNHEADER	Removes the header control that's normally displayed in report mode.
LVS_NOSORTHEADER	Disables the LVN_COLUMNCLICK notifications that are sent by default when a column header is clicked in report mode.
LVS_ALIGNLEFT	Aligns items along the left border in large and small icon mode.

LVS_ALIGNTOP	Aligns items along the top border in large and small icon mode.
LVS_AUTOARRANGE	Automatically arranges items in rows and columns in large and small icon mode.
LVS_EDITLABELS	Enables in-place label editing notifications.
LVS_NOLABELWRAP	Restricts labels to single lines in large icon mode.
LVS_NOSCROLL	Disables scrolling. Scrolling is enabled by default.
LVS_OWNERDRAWFIXED	Specifies that the control's owner will draw the items in response to WM_DRAWITEM messages.
LVS_SHAREIMAGELISTS	Prevents a list view from automatically deleting the image lists associated with it when the view itself is deleted.
LVS_SINGLESEL	Disables multiple-selection support.
LVS_SHOWSELALWAYS	Specifies that the selected items should always be highlighted. By default, the highlight is removed when the view loses the input focus.
LVS_SORTASCENDING	Specifies that items should be sorted in ascending order(for example, A through Z).
LVS_SORTDESCENDING	Specifies that items should be sorted in descending order(for example, Z through A).

Việc khởi tạo cho 1 list gồm 5 bước sau đây:

- Tạo 1 cặp hình bitmap bao gồm 1 hình cho chế độ xem lớn, và 1 hình cho các chế độ xem còn lại.
- Sử dụng SetImageList để liên kết danh sách hình với list view control
- Dùng **InsertColumn()** để thêm các cột cho list (tương ứng với chế độ xem chi tiết)
- Thêm các phần tử với lệnh **InsertItem()**
- Gán tựa cho nút con với **SetItemText()**

Ví dụ tổng hợp:

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_MAINFRM_H__18BD7B7C_95C6_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__18BD7B7C_95C6_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
```

```
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(AFX_MAINFRM_H__18BD7B7C_95C6_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "WinDir.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;

    cs.style &= ~FWS_ADDTOTITLE;
    return TRUE;
```

```
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

FileView.h

```
// FileView.h : interface of the CFileView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

typedef struct tagITEMINFO {
    CString     strFileName;
    DWORD       nFileSizeLow;
    FILETIME    ftLastWriteTime;
} ITEMINFO;

class CFileView : public CListView
{
protected: // create from serialization only
    CFileView();
    DECLARE_DYNCREATE(CFileView)
// Attributes
public:
    CWinDirDoc* GetDocument();

// Operations
public:
    static int CALLBACK CompareFunc (LPARAM lParam1, LPARAM lParam2,
        LPARAM lParamSort);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CFileView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
```

```
    //}}AFX_VIRTUAL

// Implementation
public:
    int Refresh (LPCTSTR pszPath);
    virtual ~CFileView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    CString m_strPath;
    void FreeItemMemory ();
    BOOL AddItem (int nIndex, WIN32_FIND_DATA* pfd);
    CImageList m_ilSmall;
    CImageList m_ilLarge;
    //{{AFX_MSG(CFileView)
    afx_msg void OnDestroy();
    afx_msg void OnGetDispInfo(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnColumnClick(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnViewLargeIcons();
    afx_msg void OnViewSmallIcons();
    afx_msg void OnViewList();
    afx_msg void OnViewDetails();
    afx_msg void OnUpdateViewLargeIcons(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewSmallIcons(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewList(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewDetails(CCmdUI* pCmdUI);
    afx_msg void OnFileNewDirectory();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in FileView.cpp
inline CWinDirDoc* CFileView::GetDocument()
    { return (CWinDirDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_)
```

FileView.cpp

```
// FileView.cpp : implementation of the CFileView class
//
#include "stdafx.h"
#include "WinDir.h"
#include "PathDialog.h"
#include "WinDirDoc.h"
#include "FileView.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CFileView
IMPLEMENT_DYNCREATE(CFileView, CListView)

BEGIN_MESSAGE_MAP(CFileView, CListView)
    //{{AFX_MSG_MAP(CFileView)
    ON_WM_DESTROY()
    ON_NOTIFY_REFLECT(LVN_GETDISPINFO, OnGetDispInfo)
    ON_NOTIFY_REFLECT(LVN_COLUMNCLICK, OnColumnClick)
    ON_COMMAND(ID_VIEW_LARGE_ICONS, OnViewLargeIcons)
    ON_COMMAND(ID_VIEW_SMALL_ICONS, OnViewSmallIcons)
    ON_COMMAND(ID_VIEW_LIST, OnViewList)
    ON_COMMAND(ID_VIEW_DETAILS, OnViewDetails)
    ON_UPDATE_COMMAND_UI(ID_VIEW_LARGE_ICONS, OnUpdateViewLargeIcons)
    ON_UPDATE_COMMAND_UI(ID_VIEW_SMALL_ICONS, OnUpdateViewSmallIcons)
    ON_UPDATE_COMMAND_UI(ID_VIEW_LIST, OnUpdateViewList)
    ON_UPDATE_COMMAND_UI(ID_VIEW_DETAILS, OnUpdateViewDetails)
    ON_COMMAND(ID_FILE_NEW_DIR, OnFileNewDirectory)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CFileView construction/destruction
CFileView::CFileView()
{
}
CFileView::~CFileView()
{
}
BOOL CFileView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CListView::PreCreateWindow (cs))
        return FALSE;

    cs.style &= ~LVS_TYPEMASK;
    cs.style |= LVS_REPORT;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CFileView drawing
void CFileView::OnDraw(CDC* pDC)
{
    CWinDirDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
}
void CFileView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    //
```

```
    // Initialize the image list.
    //
    m_ilLarge.Create (IDB_LARGEDOC, 32, 1, RGB (255, 0, 255));

    m_ilSmall.Create (IDB_SMALLDOC, 16, 1, RGB (255, 0, 255));

    GetListCtrl ().SetImageList (&m_ilLarge, LVSIL_NORMAL);
    GetListCtrl ().SetImageList (&m_ilSmall, LVSIL_SMALL);
    //
    // Add columns to the list view.
    //
    GetListCtrl ().InsertColumn (0, _T ("File Name"), LVCFMT_LEFT, 192);
    GetListCtrl ().InsertColumn (1, _T ("Size"), LVCFMT_RIGHT, 96);
    GetListCtrl ().InsertColumn (2, _T ("Last Modified"), LVCFMT_CENTER, 128);
    //
    // Populate the list view with items.
    //
    TCHAR szPath[MAX_PATH];
    ::GetCurrentDirectory (sizeof (szPath) / sizeof (TCHAR), szPath);
    Refresh (szPath);
}
////////////////////////////////////////////////////////////////////////////
// CFileView diagnostics
#ifdef _DEBUG
void CFileView::AssertValid() const
{
    CListView::AssertValid();
}
void CFileView::Dump(CDumpContext& dc) const
{
    CListView::Dump(dc);
}
CWinDirDoc* CFileView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CWinDirDoc)));
    return (CWinDirDoc*)m_pDocument;
}
#endif //_DEBUG
////////////////////////////////////////////////////////////////////////////
// CFileView message handlers
int CFileView::Refresh(LPCTSTR pszPath)
{
    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    HANDLE hFind;
    WIN32_FIND_DATA fd;
    int nCount = 0;

    if ((hFind = ::FindFirstFile (strPath, &fd)) != INVALID_HANDLE_VALUE) {
        //
        // Delete existing items (if any).
        //
```

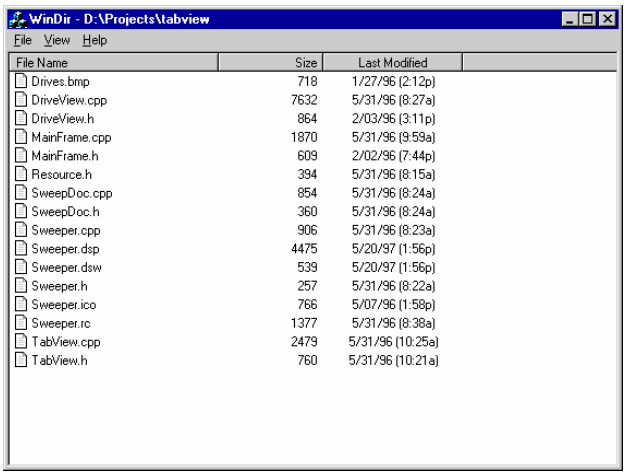
```
        GetListCtrl ().DeleteAllItems ();
        //
        // Show the path name in the frame window's title bar.
        //
        TCHAR szFullPath[MAX_PATH];
        ::GetFullPathName (pszPath, sizeof (szFullPath) / sizeof (TCHAR),
            szFullPath, NULL);
        m_strPath = szFullPath;

        CString strTitle = _T ("WinDir - ");
        strTitle += szFullPath;
        AfxGetMainWnd ()->SetWindowText (strTitle);
        //
        // Add items representing files to the list view.
        //
        if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
            AddItem (nCount++, &fd);

        while (::FindNextFile (hFind, &fd)) {
            if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
                if (!AddItem (nCount++, &fd))
                    break;
        }
        ::FindClose (hFind);
    }
    return nCount;
}
BOOL CFileView::AddItem(int nIndex, WIN32_FIND_DATA *pfd)
{
    //
    // Allocate a new ITEMINFO structure and initialize it with information
    // about the item.
    //
    ITEMINFO* pItem;
    try {
        pItem = new ITEMINFO;
    }
    catch (CMemoryException* e) {
        e->Delete ();
        return FALSE;
    }

    pItem->strFileName = pfd->cFileName;
    pItem->nFileSizeLow = pfd->nFileSizeLow;
    pItem->ftLastWriteTime = pfd->ftLastWriteTime;
    //
    // Add the item to the list view.
    //
    LV_ITEM lvi;
    lvi.mask = LVIF_TEXT ¦ LVIF_IMAGE ¦ LVIF_PARAM;
    lvi.iItem = nIndex;
    lvi.iSubItem = 0;
    lvi.iImage = 0;
    lvi.pszText = LPSTR_TEXTCALLBACK;
    lvi.lParam = (LPARAM) pItem;
```

```
    if (GetListCtrl ().InsertItem (&lvi) == -1)
        return FALSE;

    return TRUE;
}
void CFileView::FreeItemMemory()
{
    int nCount = GetListCtrl ().GetItemCount ();
    if (nCount) {
        for (int i=0; i<nCount; i++)
            delete (ITEMINFO*) GetListCtrl ().GetItemData (i);
    }
}
void CFileView::OnDestroy()
{
    FreeItemMemory ();
    CListView::OnDestroy ();
}
void CFileView::OnGetDispInfo(NMHDR* pNMHDR, LRESULT* pResult)
{
    CString string;
    LV_DISPINFO* pDispInfo = (LV_DISPINFO*) pNMHDR;

    if (pDispInfo->item.mask & LVIF_TEXT) {
        ITEMINFO* pItem = (ITEMINFO*) pDispInfo->item.lParam;

        switch (pDispInfo->item.iSubItem) {
        case 0: // File name.
            ::lstrcpy (pDispInfo->item.pszText, pItem->strFileName);
            break;
        case 1: // File size.
            string.Format (_T ("%u"), pItem->nFileSizeLow);
            ::lstrcpy (pDispInfo->item.pszText, string);
            break;
        case 2: // Date and time.
            CTime time (pItem->ftLastWriteTime);
            BOOL pm = FALSE;
            int nHour = time.GetHour ();
            if (nHour == 0)
                nHour = 12;
            else if (nHour == 12)
                pm = TRUE;
            else if (nHour > 12) {
                nHour -= 12;
                pm = TRUE;
            }
            string.Format (_T ("%d/%0.2d/%0.2d (%d:%0.2d%c)"),
                time.GetMonth (), time.GetDay (), time.GetYear () % 100,
                nHour, time.GetMinute (), pm ? _T (`p') : _T (`a'));
            ::lstrcpy (pDispInfo->item.pszText, string);
            break;
        }
    }
    *pResult = 0;
```

```
}
void CFileView::OnColumnClick(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_LISTVIEW* pNMListView = (NM_LISTVIEW*) pNMHDR;
    GetListCtrl ().SortItems (CompareFunc, pNMListView->iSubItem);
    *pResult = 0;
}
int CALLBACK CFileView::CompareFunc (LPARAM lParam1, LPARAM lParam2,
    LPARAM lParamSort)
{
    ITEMINFO* pItem1 = (ITEMINFO*) lParam1;
    ITEMINFO* pItem2 = (ITEMINFO*) lParam2;
    int nResult;

    switch (lParamSort) {
    case 0: // File name.
        nResult = pItem1->strFileName.CompareNoCase (pItem2->strFileName);
        break;
    case 1: // File size.
        nResult = pItem1->nFileSizeLow - pItem2->nFileSizeLow;
        break;
    case 2: // Date and time.
        nResult = ::CompareFileTime (&pItem1->ftLastWriteTime,
            &pItem2->ftLastWriteTime);
        break;
    }
    return nResult;
}
void CFileView::OnViewLargeIcons()
{
    ModifyStyle (LVS_TYPEMASK, LVS_ICON);
}
void CFileView::OnViewSmallIcons()
{
    ModifyStyle (LVS_TYPEMASK, LVS_SMALLICON);
}
void CFileView::OnViewList()
{
    ModifyStyle (LVS_TYPEMASK, LVS_LIST);
}
void CFileView::OnViewDetails()
{
    ModifyStyle (LVS_TYPEMASK, LVS_REPORT);
}
void CFileView::OnUpdateViewLargeIcons(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_ICON);
}
void CFileView::OnUpdateViewSmallIcons(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_SMALLICON);
}
void CFileView::OnUpdateViewList(CCmdUI* pCmdUI)
```

```
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_LIST);
}
void CFileView::OnUpdateViewDetails(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_REPORT);
}
void CFileView::OnFileNewDirectory()
{
    CPathDialog dlg;
    dlg.m_strPath = m_strPath;
    if (dlg.DoModal () == IDOK)
        Refresh (dlg.m_strPath);
}
```

Màn hình kết quả như sau:



2.3 MULTI-DOCUMENT, MULTI-VIEW VÀ MDI (MULTIPLE DOCUMENT INTERFACE)

2.3.1 Vấn đề quan tâm

- Hiểu về cấu trúc của dạng MDI
- Hiểu về vai trò của các lớp trong cấu trúc MDI.

2.3.2 Các hoạt động trong dạng MDI

Trong ứng dụng MDI, 1 document có thể được thể hiện trong nhiều view (**windows**), để có thể đồng bộ các view này cần sử dụng phương thức UpdateAllViews (trong lớp CDocument)

Ví dụ 1:

```
void UpdateAllViews(CView* pSender,
LPARAM lHint = 0L, CObject* pHint = NULL)
```

Hay

```
UpdateAllViews(NULL);
```

Khi gọi hàm này, chương trình sẽ gọi hàm **OnUpdate** của tất cả các view hiện hành và thực hiện thao tác cập nhật tương ứng.

Ví dụ 2:

```
// Thực hiện trong lớp CDocument
UpdateAllViews(NULL, 1, pLine);

// In the view class
void CMyView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
```

```
  if(lHint == 1) {
    CLine* pLine =(CLine*) pHint;
    CClientDC dc(this);
    pLine->Draw(&dc);
    return;
  }
  CView::OnUpdate(pSender, lHint, pHint);
}
```

Ví dụ tổng hợp:

MdiSquares.h

```
// MdiSquares.h : main header file for the MDISQUARES application
//
#if !defined(AFX_MDISQUARES_H__36D513DB_9CA0_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MDISQUARES_H__36D513DB_9CA0_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include `stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
/////////////////////////////////////////////////////////////////////////////
/ CMdiSquaresApp:
// See MdiSquares.cpp for the implementation of this class
//
class CMdiSquaresApp : public CWinApp
{
public:
    CMdiSquaresApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMdiSquaresApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL
       // Implementation
    //{{AFX_MSG(CMdiSquaresApp)
    afx_msg void OnAppAbout();
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_MDISQUARES_H__36D513DB_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

MdiSquares.cpp

```
// MdiSquares.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "MdiSquares.h"
#include "MainFrm.h"
#include "ChildFrm.h"
#include "SquaresDoc.h"
#include "SquaresView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMdiSquaresApp
BEGIN_MESSAGE_MAP(CMdiSquaresApp, CWinApp)
    //{{AFX_MSG_MAP(CMdiSquaresApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMdiSquaresApp construction
CMdiSquaresApp::CMdiSquaresApp()
{
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CMdiSquaresApp object
CMdiSquaresApp theApp;
/////////////////////////////////////////////////////////////////////////////
// CMdiSquaresApp initialization
BOOL CMdiSquaresApp::InitInstance()
{
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings();  // Load standard INI file
                               // options (including MRU)

    CMultiDocTemplate* pDocTemplate;
    pDocTemplate = new CMultiDocTemplate(
        IDR_MDISQUTYPE,
        RUNTIME_CLASS(CSquaresDoc),
        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
        RUNTIME_CLASS(CSquaresView));
    AddDocTemplate(pDocTemplate);
    // create main MDI Frame window
    CMainFrame* pMainFrame = new CMainFrame;
    if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
        return FALSE;
```

```
    m_pMainWnd = pMainFrame;
    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();
   // Enable DDE Execute open
    EnableShellOpen();
    RegisterShellFileTypes(TRUE);
    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    // The main window has been initialized, so show and update it.
    pMainFrame->ShowWindow(m_nCmdShow);
    pMainFrame->UpdateWindow();

    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
```

```
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CMdiSquaresApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
/////////////////////////////////////////////////////////////////////////////
// CMdiSquaresApp message handlers
```

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_MAINFRM_H__36D513DF_9CA0_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__36D513DF_9CA0_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CMDIFrameWnd

    DECLARE_DYNAMIC(CMainFrame)
public:
    CMainFrame();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

```
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
// AFX_MAINFRM_H__36D513DF_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "MdiSquares.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CMDIFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CMDIFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CMDIFrameWnd::Dump(dc);
}
```

```
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

ChildFrm.h

```
// ChildFrm.h : interface of the CChildFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_CHILDFRM_H__36D513E1_9CA0_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_CHILDFRM_H__36D513E1_9CA0_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CChildFrame : public CMDIChildWnd
{
    DECLARE_DYNCREATE(CChildFrame)
public:
    CChildFrame();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
// Generated message map functions
protected:
    //{{AFX_MSG(CChildFrame)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
// AFX_CHILDFRM_H__36D513E1_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

ChildFrm.cpp

```
// ChildFrm.cpp : implementation of the CChildFrame class
//
#include "stdafx.h"
#include "MdiSquares.h"
#include "ChildFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CChildFrame
IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
    //{{AFX_MSG_MAP(CChildFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CChildFrame construction/destruction
CChildFrame::CChildFrame()
{
}
CChildFrame::~CChildFrame()
{
}
BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CMDIChildWnd::PreCreateWindow(cs) )
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CChildFrame diagnostics
#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
    CMDIChildWnd::AssertValid();
}
void CChildFrame::Dump(CDumpContext& dc) const
{
    CMDIChildWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CChildFrame message handlers
```

SquaresDoc.h

```
// SquaresDoc.h : interface of the CSquaresDoc class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_SQUARESDOC_H__36D513E3_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

```
#define AFX_SQUARESDOC_H__36D513E3_9CA0_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CSquaresDoc : public CDocument
{
protected: // create from serialization only
    CSquaresDoc();
    DECLARE_DYNCREATE(CSquaresDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSquaresDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    void SetSquare (int i, int j, COLORREF color);
    COLORREF GetSquare (int i, int j);
    COLORREF GetCurrentColor();
    virtual ~CSquaresDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    COLORREF m_clrCurrentColor;
    COLORREF m_clrGrid[4][4];
    //{{AFX_MSG(CSquaresDoc)
    afx_msg void OnColorRed();
    afx_msg void OnColorYellow();
    afx_msg void OnColorGreen();
    afx_msg void OnColorCyan();
    afx_msg void OnColorBlue();
    afx_msg void OnColorWhite();
    afx_msg void OnUpdateColorRed(CCmdUI* pCmdUI);
    afx_msg void OnUpdateColorYellow(CCmdUI* pCmdUI);
    afx_msg void OnUpdateColorGreen(CCmdUI* pCmdUI);
    afx_msg void OnUpdateColorCyan(CCmdUI* pCmdUI);
    afx_msg void OnUpdateColorBlue(CCmdUI* pCmdUI);
    afx_msg void OnUpdateColorWhite(CCmdUI* pCmdUI);
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_SQUARESDOC_H__36D513E3_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

SquaresDoc.cpp

```
// SquaresDoc.cpp : implementation of the CSquaresDoc class
//
#include "stdafx.h"
#include "MdiSquares.h"
#include "SquaresDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc

IMPLEMENT_DYNCREATE(CSquaresDoc, CDocument)

BEGIN_MESSAGE_MAP(CSquaresDoc, CDocument)
    //{{AFX_MSG_MAP(CSquaresDoc)
    ON_COMMAND(ID_COLOR_RED, OnColorRed)
    ON_COMMAND(ID_COLOR_YELLOW, OnColorYellow)
    ON_COMMAND(ID_COLOR_GREEN, OnColorGreen)
    ON_COMMAND(ID_COLOR_CYAN, OnColorCyan)
    ON_COMMAND(ID_COLOR_BLUE, OnColorBlue)
    ON_COMMAND(ID_COLOR_WHITE, OnColorWhite)
    ON_UPDATE_COMMAND_UI(ID_COLOR_RED, OnUpdateColorRed)
    ON_UPDATE_COMMAND_UI(ID_COLOR_YELLOW, OnUpdateColorYellow)
    ON_UPDATE_COMMAND_UI(ID_COLOR_GREEN, OnUpdateColorGreen)
    ON_UPDATE_COMMAND_UI(ID_COLOR_CYAN, OnUpdateColorCyan)
    ON_UPDATE_COMMAND_UI(ID_COLOR_BLUE, OnUpdateColorBlue)
    ON_UPDATE_COMMAND_UI(ID_COLOR_WHITE, OnUpdateColorWhite)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc construction/destruction
CSquaresDoc::CSquaresDoc()
{
}
CSquaresDoc::~CSquaresDoc()
{
}
BOOL CSquaresDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;
```

```
    for (int i=0; i<4; i++)
        for (int j=0; j<4; j++)
            m_clrGrid[i][j] = RGB (255, 255, 255);

    m_clrCurrentColor = RGB (255, 0, 0);
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc serialization
void CSquaresDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        for (int i=0; i<4; i++)
            for (int j=0; j<4; j++)
                ar << m_clrGrid[i][j];
        ar << m_clrCurrentColor;
    }
    else
    {
        for (int i=0; i<4; i++)
            for (int j=0; j<4; j++)
                ar >> m_clrGrid[i][j];
        ar >> m_clrCurrentColor;
    }
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc diagnostics
#ifdef _DEBUG
void CSquaresDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CSquaresDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc commands
COLORREF CSquaresDoc::GetCurrentColor()
{
    return m_clrCurrentColor;
}
COLORREF CSquaresDoc::GetSquare(int i, int j)
{
    ASSERT (i >= 0 && i <= 3 && j >= 0 && j <= 3);
    return m_clrGrid[i][j];
}
void CSquaresDoc::SetSquare(int i, int j, COLORREF color)
{
    ASSERT (i >= 0 && i <= 3 && j >= 0 && j <= 3);
    m_clrGrid[i][j] = color;
```

```
    SetModifiedFlag (TRUE);
    UpdateAllViews (NULL);
}
void CSquaresDoc::OnColorRed()
{
    m_clrCurrentColor = RGB (255, 0, 0);
}

void CSquaresDoc::OnColorYellow()
{
    m_clrCurrentColor = RGB (255, 255, 0);
}
void CSquaresDoc::OnColorGreen()
{
    m_clrCurrentColor = RGB (0, 255, 0);
}
void CSquaresDoc::OnColorCyan()
{
    m_clrCurrentColor = RGB (0, 255, 255);
}
void CSquaresDoc::OnColorBlue()
{
    m_clrCurrentColor = RGB (0, 0, 255);
}
void CSquaresDoc::OnColorWhite()
{
    m_clrCurrentColor = RGB (255, 255, 255);
}
void CSquaresDoc::OnUpdateColorRed(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (255, 0, 0));
}
void CSquaresDoc::OnUpdateColorYellow(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (255, 255, 0));
}
void CSquaresDoc::OnUpdateColorGreen(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (0, 255, 0));
}
void CSquaresDoc::OnUpdateColorCyan(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (0, 255, 255));
}
void CSquaresDoc::OnUpdateColorBlue(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (0, 0, 255));
}
void CSquaresDoc::OnUpdateColorWhite(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (255, 255, 255));
}
```

SquaresView.h

```
// SquaresView.h : interface of the CSquaresView class
//
```

```
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_SQUARESVIEW_H__36D513E5_9CA0_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SQUARESVIEW_H__36D513E5_9CA0_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CSquaresView : public CView
{
protected: // create from serialization only
    CSquaresView();
    DECLARE_DYNCREATE(CSquaresView)

// Attributes
public:
    CSquaresDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSquaresView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CSquaresView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CSquaresView)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in SquaresView.cpp
inline CSquaresDoc* CSquaresView::GetDocument()
   { return (CSquaresDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
```

```
// !defined(
//     AFX_SQUARESVIEW_H__36D513E5_9CA0_11D2_8E53_006008A82731__INCLUDED_)
```

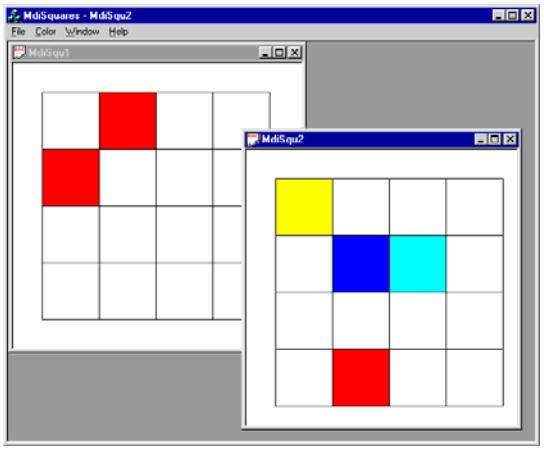
SquaresView.cpp

```
// SquaresView.cpp : implementation of the CSquaresView class
//
#include "stdafx.h"
#include "MdiSquares.h"
#include "SquaresDoc.h"
#include "SquaresView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSquaresView
IMPLEMENT_DYNCREATE(CSquaresView, CView)

BEGIN_MESSAGE_MAP(CSquaresView, CView)
    //{{AFX_MSG_MAP(CSquaresView)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSquaresView construction/destruction
CSquaresView::CSquaresView()
{
}
CSquaresView::~CSquaresView()
{
}
BOOL CSquaresView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CSquaresView drawing
void CSquaresView::OnDraw(CDC* pDC)
{
    CSquaresDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //
    // Set the mapping mode to MM_LOENGLISH.
    //
    pDC->SetMapMode (MM_LOENGLISH);
    //
    // Draw the 16 squares.
    //
    for (int i=0; i<4; i++) {
        for (int j=0; j<4; j++) {
            COLORREF color = pDoc->GetSquare (i, j);
            CBrush brush (color);
            int x1 = (j * 70) + 35;
            int y1 = (i * -70) - 35;
```

```
            int x2 = x1 + 70;
            int y2 = y1 - 70;
            CRect rect (x1, y1, x2, y2);
            pDC->FillRect (rect, &brush);
        }
    }
    //
    // Then draw the grid lines surrounding them.
    //
    for (int x=35; x<=315; x+=70) {
        pDC->MoveTo (x, -35);
        pDC->LineTo (x, -315);
    }
    for (int y=-35; y>=-315; y-=70) {
        pDC->MoveTo (35, y);
        pDC->LineTo (315, y);
    }
}
////////////////////////////////////////////////////////////////////////////
// CSquaresView diagnostics
#ifdef _DEBUG
void CSquaresView::AssertValid() const
{
    CView::AssertValid();
}
void CSquaresView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}
CSquaresDoc* CSquaresView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSquaresDoc)));
    return (CSquaresDoc*)m_pDocument;
}
#endif //_DEBUG
////////////////////////////////////////////////////////////////////////////
// CSquaresView message handlers
void CSquaresView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CView::OnLButtonDown(nFlags, point);
    //
    // Convert click coordinates to MM_LOENGLISH units.
    //
    CClientDC dc (this);
    dc.SetMapMode (MM_LOENGLISH);
    CPoint pos = point;
    dc.DPtoLP (&pos);
    //
    // If a square was clicked, set its color to the current color.
    //
    if (pos.x >= 35 && pos.x <= 315 && pos.y <= -35 && pos.y >= -315) {
        int i = (-pos.y - 35) / 70;
        int j = (pos.x - 35) / 70;
        CSquaresDoc* pDoc = GetDocument ();
        COLORREF clrCurrentColor = pDoc->GetCurrentColor ();
```

```
        pDoc->SetSquare (i, j, clrCurrentColor);
    }
}
```

Màn hình kết quả như sau:



2.3.3 Dạng splitter

Để phân chia cửa sổ thành các vùng riêng biệt, cần thực hiện việc phân chia cửa sổ, thao tác như sau:

2.3.3.1 Tạo vùng phân chia động:

- Thêm biến thành viên (thuộc class CSplitterWnd) vào class CMainFrame cửa sổ khung chương trình.
- Khai báo lại hàm **OnCreateClient** và gọi hàm CSplitterWnd::**Create** để tạo khung cửa sổ phân chia được trong cửa sổ khung.

Ví dụ:

```
class CMainFrame : public CFrameWnd
{
…
protected:
  CSplitterWnd m_wndSplitter;
…
};
```

và

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
  return m_wndSplitter.Create(this, 2, 1, CSize(8, 8), pContext);
}
```

2.3.3.2 Tạo vùng phân chia tĩnh:

- Thêm biến thành viên (thuộc class CSplitterWnd) vào class CMainFrame cửa sổ khung chương trình.
- Khai báo lại hàm **OnCreateClient** và gọi hàm CSplitterWnd::**CreateStatic** để tạo khung cửa sổ phân chia được trong cửa sổ khung
- Dùng CSplitterWnd::**CreateView** để tạo các view cho mỗi thành phần.

Ví dụ:

```
class CMainFrame : public CFrameWnd
{
…
protected:
  CSplitterWnd m_wndSplitter;
…
};
```

và

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
  if(!m_wndSplitter.CreateStatic(this, 1, 2) ¦¦
```

```
    !m_wndSplitter.CreateView(0, 0, RUNTIME_CLASS(CTextView),
      CSize(128, 0), pContext) ||
    !m_wndSplitter.CreateView(0, 1, RUNTIME_CLASS(CPictureView),
      CSize(0, 0), pContext))
    return FALSE;

  return TRUE;
}
```

2.3.4 Ví dụ tổng hợp

2.3.4.1 Chương trình 1:

Sketch.h

```
// Sketch.h : main header file for the SKETCH application
//
#if !defined(AFX_SKETCH_H__1260AFC5_9CAC_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SKETCH_H__1260AFC5_9CAC_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include `stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
/////////////////////////////////////////////////////////////////////////////
// CSketchApp:
// See Sketch.cpp for the implementation of this class
//
class CSketchApp : public CWinApp
{
public:
    CSketchApp();
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSketchApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CSketchApp)
    afx_msg void OnAppAbout();

// NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
```

```
// !defined(AFX_SKETCH_H__1260AFC5_9CAC_11D2_8E53_006008A82731__INCLUDED_)
```

Sketch.cpp

```
// Sketch.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "Line.h"
#include "Sketch.h"
#include "MainFrm.h"
#include "SketchDoc.h"
#include "SketchView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSketchApp
BEGIN_MESSAGE_MAP(CSketchApp, CWinApp)
    //{{AFX_MSG_MAP(CSketchApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSketchApp construction
CSketchApp::CSketchApp()
{
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CSketchApp object
CSketchApp theApp;
/////////////////////////////////////////////////////////////////////////////
// CSketchApp initialization
BOOL CSketchApp::InitInstance()
{
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings();  // Load standard INI file
                               // options (including MRU)

    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CSketchDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CSketchView));
    AddDocTemplate(pDocTemplate);

    // Enable DDE Execute open
    EnableShellOpen();
```

```
    RegisterShellFileTypes(TRUE);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

   // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();

    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
```

```
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CSketchApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
/////////////////////////////////////////////////////////////////////////////
// CSketchApp message handlers
```

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
//
#if !defined(AFX_MAINFRM_H__1260AFC9_9CAC_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__1260AFC9_9CAC_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs,
        CCreateContext* pContext);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
```

```
protected:
    CSplitterWnd m_wndSplitter;
    //{{AFX_MSG(CMainFrame)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(AFX_MAINFRM_H__1260AFC9_9CAC_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "Sketch.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
```

```
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
    return m_wndSplitter.Create (this, 2, 1, CSize (8, 8), pContext);
}
```

SketchDoc.h

```
// SketchDoc.h : interface of the CSketchDoc class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_SKETCHDOC_H__1260AFCB_9CAC_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SKETCHDOC_H__1260AFCB_9CAC_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

typedef CTypedPtrArray<CObArray, CLine*> CLineArray;
class CSketchDoc : public CDocument
{
protected: // create from serialization only
    CSketchDoc();
    DECLARE_DYNCREATE(CSketchDoc)
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSketchDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    virtual void DeleteContents();
    //}}AFX_VIRTUAL

// Implementation
public:
    CLine* GetLine (int nIndex);
    int GetLineCount ();
    CLine* AddLine (POINT from, POINT to);
    BOOL IsGridVisible ();
    virtual ~CSketchDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
```

```
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// Generated message map functions
protected:
    CLineArray m_arrLines;
    BOOL m_bShowGrid;
    //{{AFX_MSG(CSketchDoc)
    afx_msg void OnViewGrid();
    afx_msg void OnUpdateViewGrid(CCmdUI* pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//     AFX_SKETCHDOC_H__1260AFCB_9CAC_11D2_8E53_006008A82731__INCLUDED_)
```

SketchDoc.cpp

```
// SketchDoc.cpp : implementation of the CSketchDoc class
//
#include "stdafx.h"
#include "Line.h"
#include "Sketch.h"
#include "SketchDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSketchDoc

IMPLEMENT_DYNCREATE(CSketchDoc, CDocument)

BEGIN_MESSAGE_MAP(CSketchDoc, CDocument)
    //{{AFX_MSG_MAP(CSketchDoc)
    ON_COMMAND(ID_VIEW_GRID, OnViewGrid)
    ON_UPDATE_COMMAND_UI(ID_VIEW_GRID, OnUpdateViewGrid)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSketchDoc construction/destruction
CSketchDoc::CSketchDoc()
{
}
CSketchDoc::~CSketchDoc()
{
}
BOOL CSketchDoc::OnNewDocument()
```

```
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    m_bShowGrid = TRUE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CSketchDoc serialization
void CSketchDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        ar << m_bShowGrid;
    }
    else
    {
        ar >> m_bShowGrid;
    }
    m_arrLines.Serialize (ar);
}
/////////////////////////////////////////////////////////////////////////////
// CSketchDoc diagnostics
#ifdef _DEBUG
void CSketchDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CSketchDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CSketchDoc commands
BOOL CSketchDoc::IsGridVisible()
{
    return m_bShowGrid;
}
void CSketchDoc::OnViewGrid()
{
    if (m_bShowGrid)
        m_bShowGrid = FALSE;
    else
        m_bShowGrid = TRUE;

    SetModifiedFlag (TRUE);
    UpdateAllViews (NULL);
}
void CSketchDoc::OnUpdateViewGrid(CCmdUI* pCmdUI)
{
    pCmdUI->SetCheck (m_bShowGrid);
}
CLine* CSketchDoc::AddLine(POINT from, POINT to)
{
```

```
    CLine* pLine = NULL;

    try {
        pLine = new CLine (from, to);
        m_arrLines.Add (pLine);
        SetModifiedFlag (TRUE);
        UpdateAllViews (NULL, 0x7C, pLine);
    }
    catch (CMemoryException* e) {
        AfxMessageBox (_T ("Out of memory"));
        if (pLine != NULL) {
            delete pLine;
            pLine = NULL;
        }
        e->Delete ();
    }
    return pLine;
}
int CSketchDoc::GetLineCount()
{
    return m_arrLines.GetSize ();
}
CLine* CSketchDoc::GetLine(int nIndex)
{
    ASSERT (nIndex < GetLineCount ());
    return m_arrLines[nIndex];
}
void CSketchDoc::DeleteContents()
{
    int nCount = GetLineCount ();

    if (nCount) {
        for (int i=0; i<nCount; i++)
            delete m_arrLines[i];
        m_arrLines.RemoveAll ();
    }
    CDocument::DeleteContents();
}
```

SketchView.h

```
// SketchView.h : interface of the CSketchView class
//
/////////////////////////////////////////////////////////////////////////////
//
#if !defined(AFX_SKETCHVIEW_H__1260AFCD_9CAC_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SKETCHVIEW_H__1260AFCD_9CAC_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CSketchView : public CScrollView
{
protected: // create from serialization only
    CSketchView();
    DECLARE_DYNCREATE(CSketchView)
```

```
// Attributes
public:
    CSketchDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSketchView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CSketchView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    void InvertLine (CDC* pDC, POINT from, POINT to);
    CPoint m_ptFrom;
    CPoint m_ptTo;
    HCURSOR m_hCursor;
    //{{AFX_MSG(CSketchView)
    afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in SketchView.cpp
inline CSketchDoc* CSketchView::GetDocument()
    { return (CSketchDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//     AFX_SKETCHVIEW_H__1260AFCD_9CAC_11D2_8E53_006008A82731__INCLUDED_)
```

SketchView.cpp

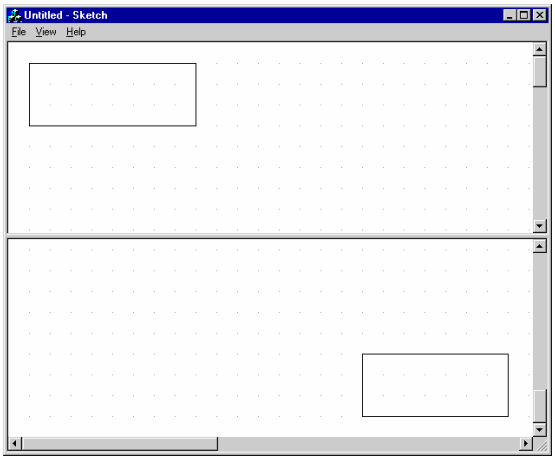
```
// SketchView.cpp : implementation of the CSketchView class
//
#include "stdafx.h"
#include "Line.h"
#include "Sketch.h"
#include "SketchDoc.h"
#include "SketchView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSketchView
IMPLEMENT_DYNCREATE(CSketchView, CScrollView)

BEGIN_MESSAGE_MAP(CSketchView, CScrollView)
    //{{AFX_MSG_MAP(CSketchView)
    ON_WM_SETCURSOR()
    ON_WM_LBUTTONDOWN()

    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONUP()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSketchView construction/destruction
CSketchView::CSketchView()
{
    m_hCursor = AfxGetApp ()->LoadStandardCursor (IDC_CROSS);
}
CSketchView::~CSketchView()
{
}
BOOL CSketchView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CSketchView drawing
void CSketchView::OnDraw(CDC* pDC)
{
    CSketchDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    //
    // Draw the snap grid.
    //
    if (pDoc->IsGridVisible ()) {
        for (int x=25; x<1600; x+=25)
            for (int y=-25; y>-1200; y-=25)
                pDC->SetPixel (x, y, RGB (128, 128, 128));
    }
    //
```

```
    // Draw the lines.
    //
    int nCount = pDoc->GetLineCount ();
    if (nCount) {
        for (int i=0; i<nCount; i++)
            pDoc->GetLine (i)->Draw (pDC);
    }
}
void CSketchView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
    SetScrollSizes(MM_LOENGLISH, CSize (1600, 1200));
}
////////////////////////////////////////////////////////////////////////////
// CSketchView diagnostics
#ifdef _DEBUG
void CSketchView::AssertValid() const
{
    CScrollView::AssertValid();
}
void CSketchView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CSketchDoc* CSketchView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSketchDoc)));
    return (CSketchDoc*)m_pDocument;
}
#endif //_DEBUG
////////////////////////////////////////////////////////////////////////////
// CSketchView message handlers
BOOL CSketchView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
    ::SetCursor (m_hCursor);
    return TRUE;
}
void CSketchView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CScrollView::OnLButtonDown(nFlags, point);

    CPoint pos = point;

   CClientDC dc (this);
    OnPrepareDC (&dc);
    dc.DPtoLP (&pos);

    if (GetDocument ()->IsGridVisible ()) {
        pos.x = ((pos.x + 12) / 25) * 25;
        pos.y = ((pos.y - 12) / 25) * 25;
    }

    m_ptFrom = pos;
    m_ptTo = pos;
```

```
    SetCapture ();
}
void CSketchView::OnMouseMove(UINT nFlags, CPoint point)
{
    CScrollView::OnMouseMove(nFlags, point);

    if (GetCapture () == this) {
        CPoint pos = point;
        CClientDC dc (this);
        OnPrepareDC (&dc);
        dc.DPtoLP (&pos);

        if (GetDocument ()->IsGridVisible ()) {
            pos.x = ((pos.x + 12) / 25) * 25;
            pos.y = ((pos.y - 12) / 25) * 25;
        }

        if (m_ptTo != pos) {
            InvertLine (&dc, m_ptFrom, m_ptTo);
            InvertLine (&dc, m_ptFrom, pos);
            m_ptTo = pos;
        }
    }
}
void CSketchView::OnLButtonUp(UINT nFlags, CPoint point)
{
    CScrollView::OnLButtonUp(nFlags, point);

    if (GetCapture () == this) {
        ::ReleaseCapture ();

        CPoint pos = point;
        CClientDC dc (this);
        OnPrepareDC (&dc);
        dc.DPtoLP (&pos);

        if (GetDocument ()->IsGridVisible ()) {
            pos.x = ((pos.x + 12) / 25) * 25;
            pos.y = ((pos.y - 12) / 25) * 25;
        }

        InvertLine (&dc, m_ptFrom, m_ptTo);

        CSketchDoc* pDoc = GetDocument ();
        CLine* pLine = pDoc->AddLine (m_ptFrom, m_ptTo);
    }
}
void CSketchView::InvertLine(CDC *pDC, POINT from, POINT to)
{
    int nOldMode = pDC->SetROP2 (R2_NOT);
    pDC->MoveTo (from);
    pDC->LineTo (to);
    pDC->SetROP2 (nOldMode);
}
void CSketchView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
```

```
{
    if (lHint == 0x7C) {
        CLine* pLine = (CLine*) pHint;
        ASSERT (pLine->IsKindOf (RUNTIME_CLASS (CLine)));
        CClientDC dc (this);
        OnPrepareDC (&dc);
        pLine->Draw (&dc);
        return;
    }
    CScrollView::OnUpdate (pSender, lHint, pHint);
}
```

Màn hình kết quả như sau:



2.3.4.2 Chương trình 2:

Wanderer.h

```
// Wanderer.h : main header file for the WANDERER application
//
#if !defined(AFX_WANDERER_H__AE0A6FFA_9B0F_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_WANDERER_H__AE0A6FFA_9B0F_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include `stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
/////////////////////////////////////////////////////////////////////////////
// CWandererApp:
// See Wanderer.cpp for the implementation of this class
//
class CWandererApp : public CWinApp
{
public:
    CWandererApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWandererApp)
    public:
```

```
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CWandererApp)
    afx_msg void OnAppAbout();
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_WANDERER_H__AE0A6FFA_9B0F_11D2_8E53_006008A82731__INCLUDED_)
```

Wanderer.cpp

```
// Wanderer.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "Wanderer.h"
#include "MainFrm.h"
#include "WandererDoc.h"
#include "DriveView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CWandererApp
BEGIN_MESSAGE_MAP(CWandererApp, CWinApp)
    //{{AFX_MSG_MAP(CWandererApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CWandererApp construction
CWandererApp::CWandererApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CWandererApp object
CWandererApp theApp;
/////////////////////////////////////////////////////////////////////////////
```

```
// CWandererApp initialization
BOOL CWandererApp::InitInstance()
{
    // Standard initialization

    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

    // Change the registry key under which our settings are stored.
    // TODO: You should modify this string to be something appropriate
    // such as the name of your company or organization.
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings();  // Load standard INI file
                               // options (including MRU)
    // Register the application's document templates.  Document templates
    //  serve as the connection between documents, frame windows and views.

    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CWandererDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CDriveView));
    AddDocTemplate(pDocTemplate);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
```

```
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CWandererApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
/////////////////////////////////////////////////////////////////////////////
// CWandererApp message handlers
```

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
//
#if !defined(AFX_MAINFRM_H__AE0A6FFE_9B0F_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__AE0A6FFE_9B0F_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
```

```
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
        AFX_CMDHANDLERINFO* pHandlerInfo);
    protected:
    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs,
        CCreateContext* pContext);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    CSplitterWnd m_wndSplitter;
    //{{AFX_MSG(CMainFrame)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(AFX_MAINFRM_H__AE0A6FFE_9B0F_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "Wanderer.h"
#include "WandererDoc.h"
#include "DriveView.h"
#include "FileView.h"
#include "MainFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
```

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)

        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;

    cs.style &= ~FWS_ADDTOTITLE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs,
    CCreateContext* pContext)
{
    //
    // Note: Create the CFileView first so the CDriveView's OnInitialUpdate
    // function can call OnUpdate on the CFileView.
    //
    if (!m_wndSplitter.CreateStatic (this, 1, 2) ||
        !m_wndSplitter.CreateView (0, 1, RUNTIME_CLASS
            (CFileView), CSize (0, 0), pContext) ||
        !m_wndSplitter.CreateView (0, 0, RUNTIME_CLASS (CDriveView),
            CSize (192, 0), pContext))
        return FALSE;

    return TRUE;
}
BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra,
```

```
    AFX_CMDHANDLERINFO* pHandlerInfo)
{
    //
    // Route to standard command targets first.
    //
    if (CFrameWnd::OnCmdMsg (nID, nCode, pExtra, pHandlerInfo))
        return TRUE;

    //
    // Route to inactive views second.
    //
    CWandererDoc* pDoc = (CWandererDoc*) GetActiveDocument ();
    if (pDoc != NULL) { // Important!
        return pDoc->RouteCmdToAllViews (GetActiveView (),
            nID, nCode, pExtra, pHandlerInfo);
    }
    return FALSE;
}
```

WandererDoc.h

```
// WandererDoc.h : interface of the CWandererDoc class
//
/////////////////////////////////////////////////////////////////////////////
//
#if !defined(AFX_WANDERERDOC_H__AE0A7000_9B0F_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_WANDERERDOC_H__AE0A7000_9B0F_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CWandererDoc : public CDocument
{
protected: // create from serialization only
    CWandererDoc();
    DECLARE_DYNCREATE(CWandererDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWandererDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    BOOL RouteCmdToAllViews (CView* pView, UINT nID, int nCode,
        void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo);
    virtual ~CWandererDoc();
#ifdef _DEBUG
```

```
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CWandererDoc)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_WANDERERDOC_H__AE0A7000_9B0F_11D2_8E53_006008A82731__INCLUDED_)
```

WandererDoc.cpp

```
// WandererDoc.cpp : implementation of the CWandererDoc class
//
#include "stdafx.h"
#include "Wanderer.h"

#include "WandererDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CWandererDoc
IMPLEMENT_DYNCREATE(CWandererDoc, CDocument)

BEGIN_MESSAGE_MAP(CWandererDoc, CDocument)
    //{{AFX_MSG_MAP(CWandererDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CWandererDoc construction/destruction
CWandererDoc::CWandererDoc()
{
}
CWandererDoc::~CWandererDoc()
{
}
BOOL CWandererDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
```

```
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CWandererDoc serialization
void CWandererDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}
/////////////////////////////////////////////////////////////////////////////
// CWandererDoc diagnostics
#ifdef _DEBUG
void CWandererDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CWandererDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CWandererDoc commands
BOOL CWandererDoc::RouteCmdToAllViews(CView *pView, UINT nID, int nCode,
    void *pExtra, AFX_CMDHANDLERINFO *pHandlerInfo)
{
    POSITION pos = GetFirstViewPosition ();

    while (pos != NULL) {
        CView* pNextView = GetNextView (pos);
        if (pNextView != pView) {
            if (pNextView->OnCmdMsg (nID, nCode, pExtra, pHandlerInfo))
                return TRUE;
        }
    }
    return FALSE;
}
```

DriveView.h

```
// DriveTreeView.h : interface of the CDriveView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CDriveView : public CTreeView
```

```
{
protected: // create from serialization only
    CDriveView();
    DECLARE_DYNCREATE(CDriveView)

// Attributes
public:
    CWandererDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CDriveView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CDriveView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    BOOL AddDriveItem (LPCTSTR pszDrive);
    int AddDirectories (HTREEITEM hItem, LPCTSTR pszPath);
    void DeleteAllChildren (HTREEITEM hItem);
    void DeleteFirstChild (HTREEITEM hItem);
    CString GetPathFromItem (HTREEITEM hItem);
    BOOL SetButtonState (HTREEITEM hItem, LPCTSTR pszPath);
    int AddDrives ();
    CImageList m_ilDrives;
    //{{AFX_MSG(CDriveView)
    afx_msg void OnItemExpanding(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnSelectionChanged(NMHDR* pNMHDR, LRESULT* pResult);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in DriveTreeView.cpp
inline CWandererDoc* CDriveView::GetDocument()
    { return (CWandererDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(
//     AFX_DRIVETREEVIEW_H__090B382D_959D_11D2_8E53_006008A82731__INCLUDED_)
```

DriveView.cpp

```
// DriveTreeView.cpp : implementation of the CDriveView class
//
#include "stdafx.h"
#include "Wanderer.h"

#include "WandererDoc.h"
#include "DriveView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// Image indexes
#define ILI_HARD_DISK       0
#define ILI_FLOPPY          1
#define ILI_CD_ROM          2
#define ILI_NET_DRIVE       3
#define ILI_CLOSED_FOLDER   4
#define ILI_OPEN_FOLDER     5
/////////////////////////////////////////////////////////////////////////////
// CDriveView
IMPLEMENT_DYNCREATE(CDriveView, CTreeView)

BEGIN_MESSAGE_MAP(CDriveView, CTreeView)
    //{{AFX_MSG_MAP(CDriveView)
    ON_NOTIFY_REFLECT(TVN_ITEMEXPANDING, OnItemExpanding)
    ON_NOTIFY_REFLECT(TVN_SELCHANGED, OnSelectionChanged)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CDriveView construction/destruction
CDriveView::CDriveView()
{
}
CDriveView::~CDriveView()
{
}
BOOL CDriveView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CTreeView::PreCreateWindow (cs))
        return FALSE;

    cs.style |= TVS_HASLINES | TVS_LINESATROOT | TVS_HASBUTTONS |
        TVS_SHOWSELALWAYS;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CDriveView drawing
```

```
void CDriveView::OnDraw(CDC* pDC)
{
    CWandererDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
}
void CDriveView::OnInitialUpdate()
{
    CTreeView::OnInitialUpdate();
    //
    // Initialize the image list.
    //
    m_ilDrives.Create (IDB_DRIVEIMAGES, 16, 1, RGB (255, 0, 255));
    GetTreeCtrl ().SetImageList (&m_ilDrives, TVSIL_NORMAL);
    //
    // Populate the tree view with drive items.
    //
    AddDrives ();
    //
    // Show the folders on the current drive.
    //
    TCHAR szPath[MAX_PATH];
    ::GetCurrentDirectory (sizeof (szPath) / sizeof (TCHAR), szPath);
    CString strPath = szPath;
    strPath = strPath.Left (3);

    HTREEITEM hItem = GetTreeCtrl ().GetNextItem (NULL, TVGN_ROOT);
    while (hItem != NULL) {
        if (GetTreeCtrl ().GetItemText (hItem) == strPath)
            break;
        hItem = GetTreeCtrl ().GetNextSiblingItem (hItem);
    }
    if (hItem != NULL) {
        GetTreeCtrl ().Expand (hItem, TVE_EXPAND);
        GetTreeCtrl ().Select (hItem, TVGN_CARET);
    }
    //
    // Initialize the list view.
    //
    strPath = GetPathFromItem (GetTreeCtrl ().GetSelectedItem ());
    GetDocument ()->UpdateAllViews (this, 0x5A,
        (CObject*) (LPCTSTR) strPath);
}
/////////////////////////////////////////////////////////////////////////////
// CDriveView diagnostics
#ifdef _DEBUG
void CDriveView::AssertValid() const
{
    CTreeView::AssertValid();
}
void CDriveView::Dump(CDumpContext& dc) const
{
    CTreeView::Dump(dc);
}
CWandererDoc* CDriveView::GetDocument() // non-debug version is inline
{
```

```
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CWandererDoc)));
    return (CWandererDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CDriveView message handlers
int CDriveView::AddDrives()
{
    int nPos = 0;
    int nDrivesAdded = 0;
    CString string = _T ("?:\\");

    DWORD dwDriveList = ::GetLogicalDrives ();

    while (dwDriveList) {
        if (dwDriveList & 1) {
            string.SetAt (0, _T (`A') + nPos);
            if (AddDriveItem (string))
                nDrivesAdded++;
        }
        dwDriveList >>= 1;
        nPos++;
    }
    return nDrivesAdded;
}
BOOL CDriveView::AddDriveItem(LPCTSTR pszDrive)
{
    CString string;
    HTREEITEM hItem;

    UINT nType = ::GetDriveType (pszDrive);

    switch (nType) {
    case DRIVE_REMOVABLE:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_FLOPPY,
            ILI_FLOPPY);
        GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
            ILI_CLOSED_FOLDER, hItem);
        break;
 case DRIVE_FIXED:
    case DRIVE_RAMDISK:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_HARD_DISK,
            ILI_HARD_DISK);
        SetButtonState (hItem, pszDrive);
        break;
    case DRIVE_REMOTE:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_NET_DRIVE,
            ILI_NET_DRIVE);
        SetButtonState (hItem, pszDrive);
        break;
    case DRIVE_CDROM:
        hItem = GetTreeCtrl ().InsertItem (pszDrive, ILI_CD_ROM,
            ILI_CD_ROM);
        GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
            ILI_CLOSED_FOLDER, hItem);
```

```
        break;
    default:
        return FALSE;
    }
    return TRUE;
}
BOOL CDriveView::SetButtonState(HTREEITEM hItem, LPCTSTR pszPath)
{
    HANDLE hFind;
    WIN32_FIND_DATA fd;
    BOOL bResult = FALSE;

    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    if ((hFind = ::FindFirstFile (strPath, &fd)) == INVALID_HANDLE_VALUE)
        return bResult;

    do {
        if (fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) {
            CString strComp = (LPCTSTR) &fd.cFileName;
            if ((strComp != _T (".")) && (strComp != _T (".."))) {
                GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                    ILI_CLOSED_FOLDER, hItem);
                bResult = TRUE;
                break;
            }
        }
    } while (::FindNextFile (hFind, &fd));

    ::FindClose (hFind);
    return bResult;
}
void CDriveView::OnItemExpanding(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    HTREEITEM hItem = pNMTreeView->itemNew.hItem;
    CString string = GetPathFromItem (hItem);

    *pResult = FALSE;

    if (pNMTreeView->action == TVE_EXPAND) {
        DeleteFirstChild (hItem);
        if (AddDirectories (hItem, string) == 0)
            *pResult = TRUE;
    }
    else { // pNMTreeView->action == TVE_COLLAPSE
        DeleteAllChildren (hItem);
        if (GetTreeCtrl ().GetParentItem (hItem) == NULL)
            GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                ILI_CLOSED_FOLDER, hItem);
        else
            SetButtonState (hItem, string);
```

```
    }
}
CString CDriveView::GetPathFromItem(HTREEITEM hItem)
{
    CString strResult = GetTreeCtrl ().GetItemText (hItem);

    HTREEITEM hParent;
    while ((hParent = GetTreeCtrl ().GetParentItem (hItem)) != NULL) {
        CString string = GetTreeCtrl ().GetItemText (hParent);

   if (string.Right (1) != _T ("\\"))
            string += _T ("\\");
        strResult = string + strResult;
        hItem = hParent;
    }
    return strResult;
}
void CDriveView::DeleteFirstChild(HTREEITEM hItem)
{
    HTREEITEM hChildItem;
    if ((hChildItem = GetTreeCtrl ().GetChildItem (hItem)) != NULL)
        GetTreeCtrl ().DeleteItem (hChildItem);
}
void CDriveView::DeleteAllChildren(HTREEITEM hItem)
{
    HTREEITEM hChildItem;
    if ((hChildItem = GetTreeCtrl ().GetChildItem (hItem)) == NULL)
        return;

    do {
        HTREEITEM hNextItem =
            GetTreeCtrl ().GetNextSiblingItem (hChildItem);
        GetTreeCtrl ().DeleteItem (hChildItem);
        hChildItem = hNextItem;
    } while (hChildItem != NULL);
}
int CDriveView::AddDirectories(HTREEITEM hItem, LPCTSTR pszPath)
{
    HANDLE hFind;
    WIN32_FIND_DATA fd;
    HTREEITEM hNewItem;

    int nCount = 0;

    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    if ((hFind = ::FindFirstFile (strPath, &fd)) == INVALID_HANDLE_VALUE) {
        if (GetTreeCtrl ().GetParentItem (hItem) == NULL)
            GetTreeCtrl ().InsertItem (_T (""), ILI_CLOSED_FOLDER,
                ILI_CLOSED_FOLDER, hItem);
        return 0;
    }
```

```
    do {
        if (fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) {
            CString strComp = (LPCTSTR) &fd.cFileName;
            if ((strComp != _T (".")) && (strComp != _T ("..."))) {
                hNewItem =
                    GetTreeCtrl ().InsertItem ((LPCTSTR) &fd.cFileName,
                    ILI_CLOSED_FOLDER, ILI_OPEN_FOLDER, hItem);

                CString strNewPath = pszPath;
                if (strNewPath.Right (1) != _T ("\\"))
                    strNewPath += _T ("\\");

                strNewPath += (LPCTSTR) &fd.cFileName;
                SetButtonState (hNewItem, strNewPath);
                nCount++;
            }
        }
    } while (::FindNextFile (hFind, &fd));

    ::FindClose (hFind);
    return nCount;
}
void CDriveView::OnSelectionChanged(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*) pNMHDR;
    CString strPath = GetPathFromItem (pNMTreeView->itemNew.hItem);
    GetDocument ()->UpdateAllViews (this, 0x5A,
        (CObject*) (LPCTSTR) strPath);
    *pResult = 0;
}
```

FileView.h

```
// FileView.h : interface of the CFileView class
//
/////////////////////////////////////////////////////////////////////////////
//
#if !defined(AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

typedef struct tagITEMINFO {
    CString     strFileName;
    DWORD       nFileSizeLow;
    FILETIME    ftLastWriteTime;
} ITEMINFO;

class CFileView : public CListView
{
protected: // create from serialization only
    CFileView();
    DECLARE_DYNCREATE(CFileView)
```

```
// Attributes
public:
    CWandererDoc* GetDocument();

// Operations
public:
    static int CALLBACK CompareFunc (LPARAM lParam1, LPARAM lParam2,
        LPARAM lParamSort);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CFileView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

// Implementation
public:
    int Refresh (LPCTSTR pszPath);
    virtual ~CFileView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    CString m_strPath;
    void FreeItemMemory ();
    BOOL AddItem (int nIndex, WIN32_FIND_DATA* pfd);
    CImageList m_ilSmall;
    CImageList m_ilLarge;
    //{{AFX_MSG(CFileView)
    afx_msg void OnDestroy();
    afx_msg void OnGetDispInfo(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnColumnClick(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnViewLargeIcons();
    afx_msg void OnViewSmallIcons();
    afx_msg void OnViewList();
    afx_msg void OnViewDetails();
    afx_msg void OnUpdateViewLargeIcons(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewSmallIcons(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewList(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewDetails(CCmdUI* pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in FileView.cpp
inline CWandererDoc* CFileView::GetDocument()
```

```
    { return (CWandererDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
#endif
// !defined(AFX_FILEVIEW_H__18BD7B80_95C6_11D2_8E53_006008A82731__INCLUDED_)
```

FileView.cpp

```
// FileView.cpp : implementation of the CFileView class
//
#include "stdafx.h"
#include "Wanderer.h"
#include "WandererDoc.h"
#include "FileView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CFileView
IMPLEMENT_DYNCREATE(CFileView, CListView)

BEGIN_MESSAGE_MAP(CFileView, CListView)
    //{{AFX_MSG_MAP(CFileView)
    ON_WM_DESTROY()
    ON_NOTIFY_REFLECT(LVN_GETDISPINFO, OnGetDispInfo)
    ON_NOTIFY_REFLECT(LVN_COLUMNCLICK, OnColumnClick)
    ON_COMMAND(ID_VIEW_LARGE_ICONS, OnViewLargeIcons)
    ON_COMMAND(ID_VIEW_SMALL_ICONS, OnViewSmallIcons)
    ON_COMMAND(ID_VIEW_LIST, OnViewList)
    ON_COMMAND(ID_VIEW_DETAILS, OnViewDetails)
    ON_UPDATE_COMMAND_UI(ID_VIEW_LARGE_ICONS, OnUpdateViewLargeIcons)
    ON_UPDATE_COMMAND_UI(ID_VIEW_SMALL_ICONS, OnUpdateViewSmallIcons)
    ON_UPDATE_COMMAND_UI(ID_VIEW_LIST, OnUpdateViewList)
    ON_UPDATE_COMMAND_UI(ID_VIEW_DETAILS, OnUpdateViewDetails)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CFileView construction/destruction
CFileView::CFileView()
{
}
CFileView::~CFileView()
{
}
BOOL CFileView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CListView::PreCreateWindow (cs))
        return FALSE;

    cs.style &= ~LVS_TYPEMASK;
    cs.style |= LVS_REPORT;
    return TRUE;
```

```
}
/////////////////////////////////////////////////////////////////////////////
// CFileView drawing
void CFileView::OnDraw(CDC* pDC)
{
    CWandererDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
}
void CFileView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    //
    // Initialize the image list.
    //
    m_ilLarge.Create (IDB_LARGEDOC, 32, 1, RGB (255, 0, 255));
    m_ilSmall.Create (IDB_SMALLDOC, 16, 1, RGB (255, 0, 255));

    GetListCtrl ().SetImageList (&m_ilLarge, LVSIL_NORMAL);
    GetListCtrl ().SetImageList (&m_ilSmall, LVSIL_SMALL);
    //
    // Add columns to the list view.
    //
    GetListCtrl ().InsertColumn (0, _T ("File Name"), LVCFMT_LEFT, 192);
    GetListCtrl ().InsertColumn (1, _T ("Size"), LVCFMT_RIGHT, 96);
    GetListCtrl ().InsertColumn (2, _T ("Last Modified"), LVCFMT_CENTER,
        128);
    //
    // Populate the list view with items.
    //
    TCHAR szPath[MAX_PATH];
    ::GetCurrentDirectory (sizeof (szPath) / sizeof (TCHAR), szPath);
    Refresh (szPath);
}
/////////////////////////////////////////////////////////////////////////////
// CFileView diagnostics
#ifdef _DEBUG
void CFileView::AssertValid() const
{
    CListView::AssertValid();
}
void CFileView::Dump(CDumpContext& dc) const
{
    CListView::Dump(dc);
}
CWandererDoc* CFileView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CWandererDoc)));
    return (CWandererDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CFileView message handlers
int CFileView::Refresh(LPCTSTR pszPath)
{
```

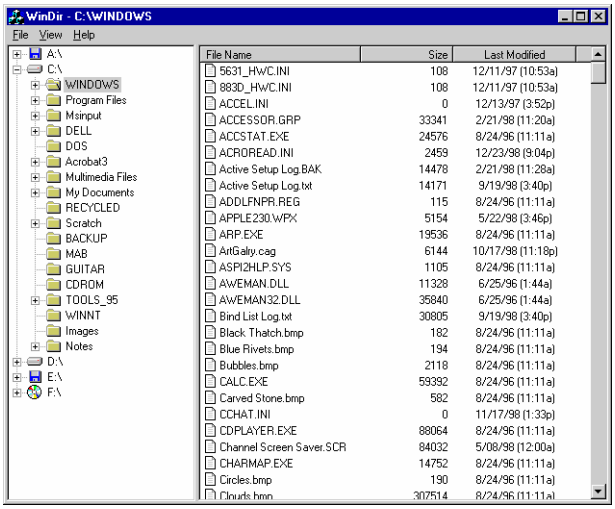
```
    CString strPath = pszPath;
    if (strPath.Right (1) != _T ("\\"))
        strPath += _T ("\\");
    strPath += _T ("*.*");

    HANDLE hFind;
    WIN32_FIND_DATA fd;
    int nCount = 0;

    if ((hFind = ::FindFirstFile (strPath, &fd)) != INVALID_HANDLE_VALUE) {
        //
        // Delete existing items (if any).
        //
        GetListCtrl ().DeleteAllItems ();
        //
        // Show the path name in the frame window's title bar.
        //
        TCHAR szFullPath[MAX_PATH];
        ::GetFullPathName (pszPath, sizeof (szFullPath) / sizeof (TCHAR),
            szFullPath, NULL);
        m_strPath = szFullPath;

        CString strTitle = _T ("WinDir - ");
        strTitle += szFullPath;
        AfxGetMainWnd ()->SetWindowText (strTitle);
        //
        // Add items representing files to the list view.
        //
        if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
            AddItem (nCount++, &fd);

        while (::FindNextFile (hFind, &fd)) {
            if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
                if (!AddItem (nCount++, &fd))
                    break;
        }
        ::FindClose (hFind);
    }
    return nCount;
}
BOOL CFileView::AddItem(int nIndex, WIN32_FIND_DATA *pfd)
{
    //
    // Allocate a new ITEMINFO structure and initialize it with information
    // about the item.
    //
    ITEMINFO* pItem;
    try {
        pItem = new ITEMINFO;
    }
    catch (CMemoryException* e) {
        e->Delete ();
        return FALSE;
    }
    pItem->strFileName = pfd->cFileName;
```

```
    pItem->nFileSizeLow = pfd->nFileSizeLow;
    pItem->ftLastWriteTime = pfd->ftLastWriteTime;
    //
    // Add the item to the list view.
    //
    LV_ITEM lvi;
    lvi.mask = LVIF_TEXT | LVIF_IMAGE | LVIF_PARAM;
    lvi.iItem = nIndex;
    lvi.iSubItem = 0;
    lvi.iImage = 0;
    lvi.pszText = LPSTR_TEXTCALLBACK;
    lvi.lParam = (LPARAM) pItem;

    if (GetListCtrl ().InsertItem (&lvi) == -1)
        return FALSE;

    return TRUE;
}
void CFileView::FreeItemMemory()
{
    int nCount = GetListCtrl ().GetItemCount ();
    if (nCount) {
        for (int i=0; i<nCount; i++)
            delete (ITEMINFO*) GetListCtrl ().GetItemData (i);
    }
}
void CFileView::OnDestroy()
{
    FreeItemMemory ();
    CListView::OnDestroy ();
}
void CFileView::OnGetDispInfo(NMHDR* pNMHDR, LRESULT* pResult)
{
    CString string;
    LV_DISPINFO* pDispInfo = (LV_DISPINFO*) pNMHDR;

    if (pDispInfo->item.mask & LVIF_TEXT) {
        ITEMINFO* pItem = (ITEMINFO*) pDispInfo->item.lParam;

        switch (pDispInfo->item.iSubItem) {
        case 0: // File name
            ::lstrcpy (pDispInfo->item.pszText, pItem->strFileName);
            break;
        case 1: // File size
            string.Format (_T ("%u"), pItem->nFileSizeLow);
            ::lstrcpy (pDispInfo->item.pszText, string);
            break;
        case 2: // Date and time
            CTime time (pItem->ftLastWriteTime);

            BOOL pm = FALSE;
            int nHour = time.GetHour ();
            if (nHour == 0)
                nHour = 12;
            else if (nHour == 12)
```

```
                pm = TRUE;
            else if (nHour > 12) {
                nHour -= 12;
                pm = TRUE;
            }

            string.Format (_T ("%d/%0.2d/%0.2d (%d:%0.2d%c)"),
                time.GetMonth (), time.GetDay (), time.GetYear () % 100,
                nHour, time.GetMinute (), pm ? _T (`p') : _T (`a'));
            ::lstrcpy (pDispInfo->item.pszText, string);
            break;
        }
    }
    *pResult = 0;
}
void CFileView::OnColumnClick(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_LISTVIEW* pNMListView = (NM_LISTVIEW*) pNMHDR;
    GetListCtrl ().SortItems (CompareFunc, pNMListView->iSubItem);
    *pResult = 0;
}
int CALLBACK CFileView::CompareFunc (LPARAM lParam1, LPARAM lParam2,
    LPARAM lParamSort)
{
    ITEMINFO* pItem1 = (ITEMINFO*) lParam1;
    ITEMINFO* pItem2 = (ITEMINFO*) lParam2;
    int nResult;

    switch (lParamSort) {
    case 0: // File name
        nResult = pItem1->strFileName.CompareNoCase (pItem2->strFileName);
        break;
    case 1: // File size
        nResult = pItem1->nFileSizeLow - pItem2->nFileSizeLow;
        break;
    case 2: // Date and time
        nResult = ::CompareFileTime (&pItem1->ftLastWriteTime,
            &pItem2->ftLastWriteTime);
        break;    }
    return nResult;
}
void CFileView::OnViewLargeIcons()
{
    ModifyStyle (LVS_TYPEMASK, LVS_ICON);
}
void CFileView::OnViewSmallIcons()
{
    ModifyStyle (LVS_TYPEMASK, LVS_SMALLICON);
}
void CFileView::OnViewList()
{
    ModifyStyle (LVS_TYPEMASK, LVS_LIST);
}
void CFileView::OnViewDetails()
{
```

```
    ModifyStyle (LVS_TYPEMASK, LVS_REPORT);
}
void CFileView::OnUpdateViewLargeIcons(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_ICON);
}
void CFileView::OnUpdateViewSmallIcons(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_SMALLICON);
}
void CFileView::OnUpdateViewList(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_LIST);
}
void CFileView::OnUpdateViewDetails(CCmdUI* pCmdUI)
{
    DWORD dwCurrentStyle = GetStyle () & LVS_TYPEMASK;
    pCmdUI->SetRadio (dwCurrentStyle == LVS_REPORT);
}
void CFileView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    if (lHint == 0x5A) {
        FreeItemMemory ();
        GetListCtrl ().DeleteAllItems ();
        Refresh ((LPCTSTR) pHint);
        return;
    }
    CListView::OnUpdate (pSender, lHint, pHint);
}
```

Màn hình kết quả như sau:



2.4 TOOLBAR VÀ STATUSBAR

2.4.1 Vấn đề quan tâm

- Hiểu và sử dụng được các class về Toolbar, StatusBar.

2.4.2 ToolBar

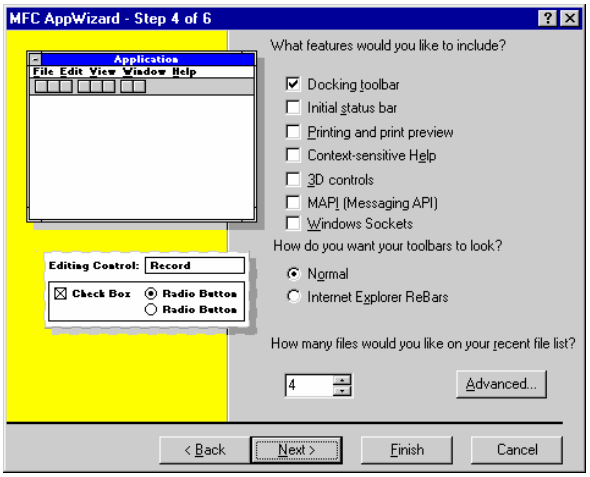
ToolBar là công cụ liên kết giữa các hình bitmap với các hàm chức năng tương ứng để thể hiện vai trò tương tự như 1 button

MFC cung cấp lớp CToolBar để giúp thao tác với toolbar thuận lợi hơn. Tuy nhiên class CToolBarCtrl còn giúp mở rộng khả năng thao tác với toolbar control (trong comctl32.dll)

Để tạo và khai báo 1 toolbar, cần thực hiện theo trình tự sau đây:

- Vào Resources View, thêm 1 toolbal vào và thêm/bớt các hình bitmap tương ứng.
- Có thể tạo 1 class liên kết với toolbar này hay sử dụng class CToolBar cơ sở
- Vào class CMainFrame
  - Thêm biến liên kết với toolbar(có kiểu là CToolBar hay class tự tạo)
  - Thêm lệnh khởi tạo cho toolbar này trong hàm OnCreate và thiết lập các đặc tính tương ứng.
  - Liên kết với document bởi việc gọi hàm EnableDocking và DockControlBar
  - Thêm các hàm chức năng tương ứng với các bitmap trong toolbar để đáp ứng các sự kiện tương tác với chúng.

Sử dụng sự hỗ trợ từ AppWizard như sau:



Ví dụ 1:

Trong Resources View

```
// Tạo 1 toolbar
IDR_MYTOOLBAR BITMAP MyToolbar.bmp
// Các thành phần của toolbar này
IDR_MYTOOLBAR TOOLBAR 16, 15
BEGIN
  BUTTON ID_CHAR_BOLD
  BUTTON ID_CHAR_ITALIC
  BUTTON ID_CHAR_UNDERLINE
  SEPARATOR
  BUTTON ID_PARA_LEFT
  BUTTON ID_PARA_CENTER
  BUTTON ID_PARA_RIGHT
END
```

// Trong file H

```
class CMainFrame : public CFrameWnd
{
…
protected:
CToolBar m_wndMyToolBar;
…
}
```

Trong file CPP

```
BOOL CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
m_wndMyToolBar.Create(this, TBSTYLE_FLAT ¦ WS_CHILD ¦ WS_VISIBLE ¦ CBRS_BOTTOM ¦
CBRS_TOOLTIPS | CBRS_FLYBY);
m_wndMyToolBar.LoadToolBar(IDR_MYTOOLBAR);

m_wndMyToolBar.SetButtonText(0, _T("My Bold"));
m_wndMyToolBar.SetButtonText(1, _T("My Italic"));
m_wndMyToolBar.SetButtonText(2, _T("My Underline"));
m_wndMyToolBar.SetButtonText(4, _T("My Left"));
m_wndMyToolBar.SetButtonText(5, _T("My Center"));
m_wndMyToolBar.SetButtonText(6, _T("My Right"));
m_wndMyToolBar.SetSizes(CSize(48, 42), CSize(40, 19));

m_wndMyToolBar.SetButtonStyle(0, TBBS_CHECKBOX);
m_wndMyToolBar.SetButtonStyle(1, TBBS_CHECKBOX);
m_wndMyToolBar.SetButtonStyle(2, TBBS_CHECKBOX);
m_wndMyToolBar.SetButtonStyle(4, TBBS_CHECKGROUP);
m_wndMyToolBar.SetButtonStyle(5, TBBS_CHECKGROUP);
m_wndMyToolBar.SetButtonStyle(6, TBBS_CHECKGROUP);

m_wndMyToolBar.EnableDocking(CBRS_ALIGN_TOP ¦ CBRS_ALIGN_BOTTOM);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndMyToolBar);

//FloatControlBar(&m_wndMyToolBar, CPoint(50, 100));
}
```

Ví dụ 2:

Trong file RC

```
IDR_TOOLBAR BITMAP Toolbar.bmp

IDR_TOOLBAR TOOLBAR 16, 15
BEGIN
    BUTTON ID_FILE_NEW
    BUTTON ID_FILE_OPEN
    BUTTON ID_FILE_SAVE
    SEPARATOR
    BUTTON ID_EDIT_CUT
    BUTTON ID_EDIT_COPY
    BUTTON ID_EDIT_PASTE
    BUTTON ID_EDIT_UNDO
    SEPARATOR
    BUTTON ID_FILE_PRINT
END
```

Trong file Cpp

```
m_wndToolBar.Create (this);
m_wndToolBar.LoadToolBar (IDR_TOOLBAR);
```

Trong file RC

```
IDR_TOOLBAR BITMAP Toolbar.bmp
```

```
IDR_TOOLBAR TOOLBAR 40, 19
```

Trong file Cpp

```
m_wndToolBar.Create (this);
m_wndToolBar.LoadToolBar (IDR_TOOLBAR);

m_wndToolBar.SetButtonText (0, _T ("New"));
m_wndToolBar.SetButtonText (1, _T ("Open"));
m_wndToolBar.SetButtonText (2, _T ("Save"));
m_wndToolBar.SetButtonText (4, _T ("Cut"));
m_wndToolBar.SetButtonText (5, _T ("Copy"));
m_wndToolBar.SetButtonText (6, _T ("Paste"));
m_wndToolBar.SetButtonText (7, _T ("Undo"));
m_wndToolBar.SetButtonText (9, _T ("Print"));

m_wndToolBar.SetSizes (CSize (48, 42), CSize (40, 19));
```

2.4.3 StatusBar

StatusBar là công cụ thể hiện thông tin trong cửa sổ ứng dụng.

MFC cung cấp class CStatusBar để giúp thao tác với statusbar thuận lợi hơn.

Để tạo và khai báo 1 statusbar, cần thực hiện theo trình tự sau đây:

- Xác định danh sách các phần(pane) trong statusbar.
- Vào class CMainFrame
  - Thêm biến liên kết với statusbar(có kiểu là CStatusBar)
  - Thêm lệnh khởi tạo cho toolbar này trong hàm OnCreate và thiết lập các đặc tính tương ứng.
  - Gọi hàm SetIndicators để khai báo số lượng pane trong statusbar
  - Để tương tác(điều khiển, hiển thị thông tin…) , truy xuất thông qua biến liên kết các hàm như SetPaneText

Ví dụ 1:

Trong Resources View

```
Khai báo các thành phần trong statusbar
static UINT nIndicators[] = {
  ID_SEPARATOR,
  ID_SEPARATOR,
  ID_SEPARATOR
};

STRINGTABLE DISCARDABLE DISCARDABLE
BEGIN
  ID_INDICATOR_EXT    "EXT"
  ID_INDICATOR_CAPS    "CAP"
  ID_INDICATOR_NUM    "NUM"
  ID_INDICATOR_SCRL    "SCRL"
  ID_INDICATOR_OVR    "OVR"
  ID_INDICATOR_REC    "REC"
END
```

Trong file H

```
class CMainFrame : public CFrameWnd
{
…
protected:
CStatusBar m_wndMyStatusBar;
…
```

```
}
```

Trong file CPP

```
// In the message map
ON_UPDATE_COMMAND_UI(ID_INDICATOR_TIME, OnUpdateTime)

BOOL CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
       …
m_wndMyStatusBar.Create(this);
m_wndSMytatusBar.SetIndicators(nIndicators, 3);

m_wndMyStatusBar.SetPaneInfo(0, ID_SEPARATOR, SBPS_NOBORDERS, 64);
m_wndMyStatusBar.SetPaneInfo(1, ID_SEPARATOR, SBPS_POPOUT, 64);
m_wndMyStatusBar.SetPaneInfo(2, ID_SEPARATOR, SBPS_NORMAL ¦
       SBPS_STRETCH, 0);
…
SetTimer(ID_TIMER, 200, NULL);
       …
}

void CMainFrame::OnTimer(UINT nTimerID)
{
  CTime time = CTime::GetCurrentTime();
  int nSecond = time.GetSecond();
  int nMinute = time.GetMinute();
  int nHour = time.GetHour() % 12;

  CString string;
  string.Format(_T("%0.2d:%0.2d:%0.2d"), nHour, nMinute, nSecond);
  m_wndMyStatusBar.SetPaneText(2, string);
}

void CMainFrame::OnUpdateTime(CCmdUI* pCmdUI)
{
  CTime time = CTime::GetCurrentTime();
  int nSecond = time.GetSecond();
  int nMinute = time.GetMinute();
  int nHour = time.GetHour() % 12;

  CString string;
  string.Format(_T("%0.2d:%0.2d:%0.2d"), nHour, nMinute, nSecond);
  pCmdUI->SetText(string);
}
```

Ví dụ tổng hợp

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_MAINFRM_H__C85C9089_A154_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__C85C9089_A154_11D2_8E53_006008A82731__INCLUDED_

#include "StyleBar.h"    // Added by ClassView
#if _MSC_VER > 1000
```

```
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
    CStyleBar   m_wndStyleBar;
    CStatusBar  m_wndStatusBar;
    CToolBar    m_wndToolBar;
// Generated message map functions
protected:
    BOOL CreateToolBar ();
    BOOL CreateStyleBar ();
    BOOL CreateStatusBar ();
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnClose();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(AFX_MAINFRM_H__C85C9089_A154_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "MyWord.h"
#include "MainFrm.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    ON_WM_CLOSE()
    //}}AFX_MSG_MAP

ON_COMMAND_EX (IDW_STYLE_BAR, OnBarCheck)
    ON_UPDATE_COMMAND_UI (IDW_STYLE_BAR, OnUpdateControlBarMenu)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    //
    // Tell the frame window to permit docking.
    //
    EnableDocking (CBRS_ALIGN_ANY);
    //
    // Create the toolbar, style bar, and status bar.
    //
    if (!CreateToolBar () ||
        !CreateStyleBar () ||
        !CreateStatusBar ())
        return -1;
    //
    // Load the saved bar state (if any).
    //
    LoadBarState (_T ("MainBarState"));
    return 0;
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
```

```
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
void CMainFrame::OnClose()
{
    SaveBarState (_T ("MainBarState"));
    CFrameWnd::OnClose();
}
BOOL CMainFrame::CreateToolBar()
{
    if (!m_wndToolBar.Create (this) ||
        !m_wndToolBar.LoadToolBar (IDR_MAINFRAME))
        return FALSE;

    m_wndToolBar.SetBarStyle (m_wndToolBar.GetBarStyle () |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

    m_wndToolBar.SetWindowText (_T ("Main"));
    m_wndToolBar.EnableDocking (CBRS_ALIGN_ANY);
    DockControlBar (&m_wndToolBar);
    return TRUE;
}
BOOL CMainFrame::CreateStyleBar()
{
    if (!m_wndStyleBar.Create (this, WS_CHILD | WS_VISIBLE | CBRS_TOP |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC, IDW_STYLE_BAR))
        return FALSE;

    m_wndStyleBar.SetWindowText (_T ("Styles"));
    m_wndStyleBar.EnableDocking (CBRS_ALIGN_TOP | CBRS_ALIGN_BOTTOM);
    DockControlBar (&m_wndStyleBar);
    return TRUE;
}
BOOL CMainFrame::CreateStatusBar()
{
    static UINT nIndicators[] = {
        ID_SEPARATOR,
        ID_INDICATOR_LINE,
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM
    };

    if (!m_wndStatusBar.Create (this))
        return FALSE;

    m_wndStatusBar.SetIndicators (nIndicators, 4);
```

```
    return TRUE;
}
```

MyWordDoc.h

```
// MyWordDoc.h : interface of the CMyWordDoc class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_MYWORDDOC_H__C85C908B_A154_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MYWORDDOC_H__C85C908B_A154_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMyWordDoc : public CRichEditDoc
{
protected: // create from serialization only
    CMyWordDoc();
    DECLARE_DYNCREATE(CMyWordDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyWordDoc)
    public:

    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL
    virtual CRichEditCntrItem* CreateClientItem(REOBJECT* preo) const;

// Implementation
public:
    virtual ~CMyWordDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CMyWordDoc)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_MYWORDDOC_H__C85C908B_A154_11D2_8E53_006008A82731__INCLUDED_)
```

MyWordDoc.cpp

```
// MyWordDoc.cpp : implementation of the CMyWordDoc class
//
#include "stdafx.h"
#include "MyWord.h"

#include "MyWordDoc.h"
#include "CntrItem.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMyWordDoc
IMPLEMENT_DYNCREATE(CMyWordDoc, CRichEditDoc)

BEGIN_MESSAGE_MAP(CMyWordDoc, CRichEditDoc)
    //{{AFX_MSG_MAP(CMyWordDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Enable default OLE container implementation
    ON_UPDATE_COMMAND_UI(ID_OLE_EDIT_LINKS,
        CRichEditDoc::OnUpdateEditLinksMenu)
    ON_COMMAND(ID_OLE_EDIT_LINKS, CRichEditDoc::OnEditLinks)
    ON_UPDATE_COMMAND_UI_RANGE(ID_OLE_VERB_FIRST,
        ID_OLE_VERB_LAST, CRichEditDoc::OnUpdateObjectVerbMenu)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMyWordDoc construction/destruction
CMyWordDoc::CMyWordDoc()
{
}
CMyWordDoc::~CMyWordDoc()
{
}
BOOL CMyWordDoc::OnNewDocument()
{
    if (!CRichEditDoc::OnNewDocument())
        return FALSE;
    return TRUE;
}
CRichEditCntrItem* CMyWordDoc::CreateClientItem(REOBJECT* preo) const
{
    return new CMyWordCntrItem(preo, (CMyWordDoc*) this);
}
/////////////////////////////////////////////////////////////////////////////
// CMyWordDoc serialization
```

```
void CMyWordDoc::Serialize(CArchive& ar)
{
    CRichEditDoc::Serialize(ar);
}
/////////////////////////////////////////////////////////////////////////////
// CMyWordDoc diagnostics
#ifdef _DEBUG
void CMyWordDoc::AssertValid() const
{
    CRichEditDoc::AssertValid();
}
void CMyWordDoc::Dump(CDumpContext& dc) const
{
    CRichEditDoc::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMyWordDoc commands
```

MyWordView.h

```
// MyWordView.h : interface of the CMyWordView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_MYWORDVIEW_H__C85C908D_A154_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MYWORDVIEW_H__C85C908D_A154_11D2_8E53_006008A82731__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMyWordCntrItem;
class CMyWordView : public CRichEditView
{
protected: // create from serialization only
    CMyWordView();
    DECLARE_DYNCREATE(CMyWordView)
// Attributes
public:
    CMyWordDoc* GetDocument();

// Operations
public:
    void GetFontInfo (LPTSTR pszFaceName, int& nSize);
    void ChangeFont (LPCTSTR pszFaceName);
    void ChangeFontSize (int nSize);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyWordView)
    public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
```

```
public:
    virtual ~CMyWordView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CMyWordView)
    afx_msg void OnDestroy();
    afx_msg void OnCharBold();
    afx_msg void OnCharItalic();
    afx_msg void OnCharUnderline();
    afx_msg void OnParaLeft();
    afx_msg void OnParaCenter();
    afx_msg void OnParaRight();
    afx_msg void OnUpdateCharBold(CCmdUI* pCmdUI);
    afx_msg void OnUpdateCharItalic(CCmdUI* pCmdUI);
    afx_msg void OnUpdateCharUnderline(CCmdUI* pCmdUI);
    afx_msg void OnUpdateParaLeft(CCmdUI* pCmdUI);
    afx_msg void OnUpdateParaCenter(CCmdUI* pCmdUI);
    afx_msg void OnUpdateParaRight(CCmdUI* pCmdUI);
    //}}AFX_MSG
    afx_msg void OnUpdateLineNumber (CCmdUI* pCmdUI);
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in MyWordView.cpp
inline CMyWordDoc* CMyWordView::GetDocument()
    { return (CMyWordDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_MYWORDVIEW_H__C85C908D_A154_11D2_8E53_006008A82731__INCLUDED_)
```

MyWordView.cpp

```
// MyWordView.cpp : implementation of the CMyWordView class
//
#include "stdafx.h"
#include "MyWord.h"
#include "MyWordDoc.h"
#include "CntrItem.h"
#include "MyWordView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////////////////////
// CMyWordView
IMPLEMENT_DYNCREATE(CMyWordView, CRichEditView)
BEGIN_MESSAGE_MAP(CMyWordView, CRichEditView)
    //{{AFX_MSG_MAP(CMyWordView)
    ON_WM_DESTROY()
    ON_COMMAND(ID_CHAR_BOLD, OnCharBold)
    ON_COMMAND(ID_CHAR_ITALIC, OnCharItalic)
    ON_COMMAND(ID_CHAR_UNDERLINE, OnCharUnderline)
    ON_COMMAND(ID_PARA_LEFT, OnParaLeft)
    ON_COMMAND(ID_PARA_CENTER, OnParaCenter)
    ON_COMMAND(ID_PARA_RIGHT, OnParaRight)
    ON_UPDATE_COMMAND_UI(ID_CHAR_BOLD, OnUpdateCharBold)
    ON_UPDATE_COMMAND_UI(ID_CHAR_ITALIC, OnUpdateCharItalic)
    ON_UPDATE_COMMAND_UI(ID_CHAR_UNDERLINE, OnUpdateCharUnderline)
    ON_UPDATE_COMMAND_UI(ID_PARA_LEFT, OnUpdateParaLeft)
    ON_UPDATE_COMMAND_UI(ID_PARA_CENTER, OnUpdateParaCenter)
    ON_UPDATE_COMMAND_UI(ID_PARA_RIGHT, OnUpdateParaRight)
    //}}AFX_MSG_MAP
    ON_UPDATE_COMMAND_UI(ID_INDICATOR_LINE, OnUpdateLineNumber)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMyWordView construction/destruction
CMyWordView::CMyWordView()
{
}
CMyWordView::~CMyWordView()
{
}
BOOL CMyWordView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CRichEditView::PreCreateWindow(cs);
}
void CMyWordView::OnInitialUpdate()
{
    CRichEditView::OnInitialUpdate();

    CHARFORMAT cf;
    cf.cbSize = sizeof (CHARFORMAT);
    cf.dwMask = CFM_BOLD | CFM_ITALIC | CFM_UNDERLINE |
        CFM_PROTECTED | CFM_STRIKEOUT | CFM_FACE | CFM_SIZE;
    cf.dwEffects = 0;
    cf.yHeight = 240; // 240 twips == 12 points
    ::lstrcpy (cf.szFaceName, _T ("Times New Roman"));
    SetCharFormat (cf);
}
void CMyWordView::OnDestroy()
{
    // Deactivate the item on destruction; this is important
    // when a splitter view is being used.
   CRichEditView::OnDestroy();
   COleClientItem* pActiveItem = GetDocument()->GetInPlaceActiveItem(this);
   if (pActiveItem != NULL && pActiveItem->GetActiveView() == this)
   {
       pActiveItem->Deactivate();
```

```
       ASSERT(GetDocument()->GetInPlaceActiveItem(this) == NULL);
   }
}
/////////////////////////////////////////////////////////////////////////////
// CMyWordView diagnostics
#ifdef _DEBUG
void CMyWordView::AssertValid() const
{
    CRichEditView::AssertValid();
}
void CMyWordView::Dump(CDumpContext& dc) const
{
    CRichEditView::Dump(dc);
}
CMyWordDoc* CMyWordView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CMyWordDoc)));
    return (CMyWordDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMyWordView message handlers
void CMyWordView::OnCharBold()
{
    CHARFORMAT cf;
    cf = GetCharFormatSelection ();

    if (!(cf.dwMask & CFM_BOLD) || !(cf.dwEffects & CFE_BOLD))
        cf.dwEffects = CFE_BOLD;
    else
        cf.dwEffects = 0;

    cf.dwMask = CFM_BOLD;
    SetCharFormat (cf);
}
void CMyWordView::OnCharItalic()
{
    CHARFORMAT cf;
    cf = GetCharFormatSelection ();

    if (!(cf.dwMask & CFM_ITALIC) || !(cf.dwEffects & CFE_ITALIC))
        cf.dwEffects = CFE_ITALIC;
    else
        cf.dwEffects = 0;

    cf.dwMask = CFM_ITALIC;
    SetCharFormat (cf);
}
void CMyWordView::OnCharUnderline()
{
    CHARFORMAT cf;
    cf = GetCharFormatSelection ();

    if (!(cf.dwMask & CFM_UNDERLINE) || !(cf.dwEffects & CFE_UNDERLINE))
        cf.dwEffects = CFE_UNDERLINE;
```

```
    else
        cf.dwEffects = 0;

    cf.dwMask = CFM_UNDERLINE;
    SetCharFormat (cf);
}
void CMyWordView::OnParaLeft()
{
    OnParaAlign (PFA_LEFT);
}
void CMyWordView::OnParaCenter()
{
    OnParaAlign (PFA_CENTER);
}
void CMyWordView::OnParaRight()
{
    OnParaAlign (PFA_RIGHT);
}
void CMyWordView::OnUpdateCharBold(CCmdUI* pCmdUI)
{
    OnUpdateCharEffect (pCmdUI, CFM_BOLD, CFE_BOLD);
}
void CMyWordView::OnUpdateCharItalic(CCmdUI* pCmdUI)
{
    OnUpdateCharEffect (pCmdUI, CFM_ITALIC, CFE_ITALIC);
}
void CMyWordView::OnUpdateCharUnderline(CCmdUI* pCmdUI)
{
    OnUpdateCharEffect (pCmdUI, CFM_UNDERLINE, CFE_UNDERLINE);
}
void CMyWordView::OnUpdateParaLeft(CCmdUI* pCmdUI)
{
    OnUpdateParaAlign (pCmdUI, PFA_LEFT);
}
void CMyWordView::OnUpdateParaCenter(CCmdUI* pCmdUI)
{
    OnUpdateParaAlign (pCmdUI, PFA_CENTER);
}
void CMyWordView::OnUpdateParaRight(CCmdUI* pCmdUI)
{
    OnUpdateParaAlign (pCmdUI, PFA_RIGHT);
}
void CMyWordView::OnUpdateLineNumber(CCmdUI* pCmdUI)
{
    int nLine = GetRichEditCtrl ().LineFromChar (-1) + 1;

    CString string;
    string.Format (_T ("Line %d"), nLine);
    pCmdUI->Enable (TRUE);
    pCmdUI->SetText (string);
}
void CMyWordView::ChangeFont(LPCTSTR pszFaceName)
{
    CHARFORMAT cf;
    cf.cbSize = sizeof (CHARFORMAT);
```

```
    cf.dwMask = CFM_FACE;
    ::lstrcpy (cf.szFaceName, pszFaceName);
    SetCharFormat (cf);
}
void CMyWordView::ChangeFontSize(int nSize)
{
    CHARFORMAT cf;
    cf.cbSize = sizeof (CHARFORMAT);
    cf.dwMask = CFM_SIZE;
    cf.yHeight = nSize;
    SetCharFormat (cf);
}
void CMyWordView::GetFontInfo(LPTSTR pszFaceName, int& nSize)
{
    CHARFORMAT cf = GetCharFormatSelection ();
    ::lstrcpy (pszFaceName,
        cf.dwMask & CFM_FACE ? cf.szFaceName : _T (""));
    nSize = cf.dwMask & CFM_SIZE ? cf.yHeight : -1;
}
```

StyleBar.h

```
#if !defined(
    AFX_STYLEBAR_H__C85C9099_A154_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_STYLEBAR_H__C85C9099_A154_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// StyleBar.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CStyleBar command target
class CStyleBar : public CToolBar
{
// Attributes
public:

// Operations
public:
    static int CALLBACK EnumFontNameProc (ENUMLOGFONT* lpelf,
    NEWTEXTMETRIC* lpntm, int nFontType, LPARAM lParam);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CStyleBar)
    //}}AFX_VIRTUAL
    virtual void OnUpdateCmdUI (CFrameWnd* pTarget,
        BOOL bDisableIfNoHndler);
// Implementation
protected:
    void InitTypefaceList (CDC* pDC);
    CFont m_font;
    CComboBox m_wndFontNames;
    CComboBox m_wndFontSizes;
    // Generated message map functions
    //{{AFX_MSG(CStyleBar)
```

```
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    afx_msg void OnSelectFont ();
    afx_msg void OnSelectSize ();
    afx_msg void OnCloseUp ();
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_STYLEBAR_H__C85C9099_A154_11D2_8E53_006008A82731__INCLUDED_)
```

StyleBar.cpp

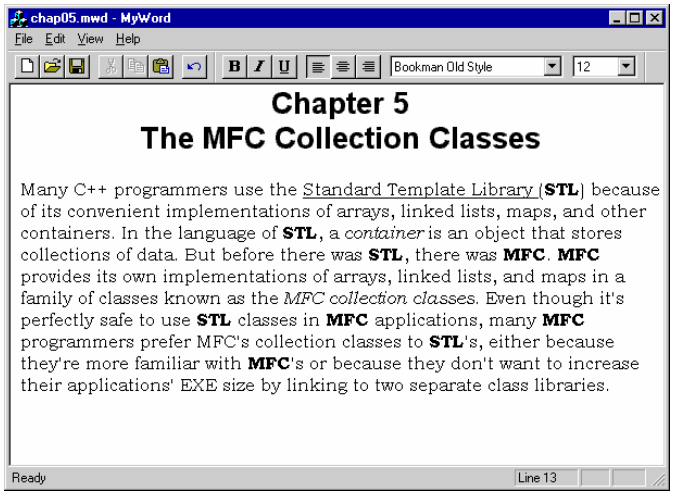
```
// StyleBar.cpp : implementation file
//
#include "stdafx.h"
#include "MyWord.h"
#include "MyWordDoc.h"
#include "MyWordView.h"
#include "StyleBar.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CStyleBar
BEGIN_MESSAGE_MAP(CStyleBar, CToolBar)
    //{{AFX_MSG_MAP(CStyleBar)
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
    ON_CBN_SELENDOK (IDC_FONTNAMES, OnSelectFont)
    ON_CBN_SELENDOK (IDC_FONTSIZES, OnSelectSize)
    ON_CBN_CLOSEUP (IDC_FONTNAMES, OnCloseUp)
    ON_CBN_CLOSEUP (IDC_FONTSIZES, OnCloseUp)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CStyleBar message handlers
int CStyleBar::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    static int nFontSizes[] = {
        8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 32, 36, 48, 72
    };
    if (CToolBar::OnCreate(lpCreateStruct) == -1)
        return -1;
    //
    // Load the toolbar.
    //
    if (!LoadToolBar (IDR_STYLE_BAR))
        return -1;
    //
    // Create an 8-point MS Sans Serif font for the combo boxes.
```

```
    //
    CClientDC dc (this);
    m_font.CreatePointFont (80, _T ("MS Sans Serif"));
    CFont* pOldFont = dc.SelectObject (&m_font);

    TEXTMETRIC tm;
    dc.GetTextMetrics (&tm);
    int cxChar = tm.tmAveCharWidth;
    int cyChar = tm.tmHeight + tm.tmExternalLeading;

    dc.SelectObject (pOldFont);
    //
    // Add the font name combo box to the toolbar.
    //
    SetButtonInfo (8, IDC_FONTNAMES, TBBS_SEPARATOR, cxChar * 32);

    CRect rect;
    GetItemRect (8, &rect);
    rect.bottom = rect.top + (cyChar * 16);

    if (!m_wndFontNames.Create (WS_CHILD | WS_VISIBLE | WS_VSCROLL |
        CBS_DROPDOWNLIST | CBS_SORT, rect, this, IDC_FONTNAMES))
        return -1;

    m_wndFontNames.SetFont (&m_font);
    InitTypefaceList (&dc);
    //
    // Add the font size combo box to the toolbar.
    //
    SetButtonInfo (10, IDC_FONTSIZES, TBBS_SEPARATOR, cxChar * 12);

    GetItemRect (10, &rect);
    rect.bottom = rect.top + (cyChar * 14);

    if (!m_wndFontSizes.Create (WS_CHILD | WS_VISIBLE | WS_VSCROLL |
        CBS_DROPDOWNLIST, rect, this, IDC_FONTSIZES))
        return -1;

    m_wndFontSizes.SetFont (&m_font);
    CString string;
    int nCount = sizeof (nFontSizes) / sizeof (int);
    for (int i=0; i<nCount; i++) {
        string.Format (_T ("%d"), nFontSizes[i]);
        m_wndFontSizes.AddString (string);
    }
    return 0;
}
void CStyleBar::OnSelectFont ()
{
    TCHAR szFaceName[LF_FACESIZE];
    int nIndex = m_wndFontNames.GetCurSel ();
    m_wndFontNames.GetLBText (nIndex, szFaceName);

    CMyWordView* pView =
        (CMyWordView*) ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView ();
```

```
    pView->ChangeFont (szFaceName);
}
void CStyleBar::OnSelectSize ()
{
    TCHAR szSize[8];
    int nIndex = m_wndFontSizes.GetCurSel ();
    m_wndFontSizes.GetLBText (nIndex, szSize);

    int nSize = atoi (szSize) * 20; // Need twips

    CMyWordView* pView =
        (CMyWordView*) ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView ();
    pView->ChangeFontSize (nSize);
}
void CStyleBar::OnCloseUp ()
{
    ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView ()->SetFocus ();
}
void CStyleBar::InitTypefaceList (CDC* pDC)
{
    ::EnumFontFamilies (pDC->m_hDC, NULL,
        (FONTENUMPROC) EnumFontNameProc, (LPARAM) this);
}
int CALLBACK CStyleBar::EnumFontNameProc (ENUMLOGFONT* lpelf,
    NEWTEXTMETRIC* lpntm, int nFontType, LPARAM lParam)
{
    CStyleBar* pWnd = (CStyleBar*) lParam;
    if (nFontType & TRUETYPE_FONTTYPE)
        pWnd->m_wndFontNames.AddString (lpelf->elfLogFont.lfFaceName);
    return 1;
}
void CStyleBar::OnUpdateCmdUI (CFrameWnd* pTarget, BOOL bDisableIfNoHndler)
{
    CToolBar::OnUpdateCmdUI (pTarget, bDisableIfNoHndler);

    CWnd* pWnd = GetFocus ();
    if ((pWnd == &m_wndFontNames) || (pWnd == &m_wndFontSizes))
        return;
    //
    // Get the font name and size.
    //
    int nTwips;
    TCHAR szFaceName[LF_FACESIZE];

    CMyWordView* pView =
        (CMyWordView*) ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView ();
    pView->GetFontInfo (szFaceName, nTwips);
    //
    // Update the font name combo box.
    //
    TCHAR szSelection[LF_FACESIZE];
    m_wndFontNames.GetWindowText (szSelection,
        sizeof (szSelection) / sizeof (TCHAR));

    if (::lstrcmp (szFaceName, szSelection) != 0) {
```

```
        if (szFaceName[0] == 0)
            m_wndFontNames.SetCurSel (-1);
        else {
            if (m_wndFontNames.SelectString (-1, szFaceName) == CB_ERR)
                m_wndFontNames.SetCurSel (-1);
        }
    }
    //
    // Update the font size combo box.
    //
    TCHAR szSize[4];
    m_wndFontSizes.GetWindowText (szSize,
        sizeof (szSize) / sizeof (TCHAR));
    int nSizeFromComboBox = atoi (szSize);
    int nSizeFromView = nTwips / 20;

    if (nSizeFromComboBox != nSizeFromView) {
        if (nTwips == -1)
            m_wndFontSizes.SetCurSel (-1);
        else {
            CString string;
            string.Format (_T ("%d"), nSizeFromView);
            if (m_wndFontSizes.SelectString (-1, string) == CB_ERR)
                m_wndFontSizes.SetCurSel (-1);
        }
    }
}
```

Màn hình kết quả như sau:



CHƯƠNG 3. XỬ LÝ HỆ THỐNG

3.1 TIMER/IDLE

3.1.1 Vấn đề quan tâm

- Hiểu và sử dụng được các class về Timer/Idles.

3.1.2 Timer

Việc tạo sự kiện thực thi lặp đi lặp lại theo chu kỳ nhất định được giải quyết bởi việc dùng WM_TIMER

Để tạo sự kiện dạng Timer, cần thực hiện theo trình tự sau:

- Vào *Add Windows Message Handler* thêm message **WM_TIMER** vào ứng dụng, hàm OnTimer xuất hiện sau khi thêm.
- Trong hàm OnCreate của class, thêm hàm **SetTimer** để khai báo thời gian(chu kỳ) lặp lại(thời gian tối đa là 2^32-1 milliseconds)
- Hiện thực các thao tác trong hàm **OnTimer** để đáp ứng sự kiện WM_TIMER.

Ví dụ 1:

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
…
  ON_WM_CREATE()
  ON_WM_TIMER()
…
END_MESSAGE_MAP()

int CMainFrame::OnCreate(LPCREATESTRUCT lpcs)
{
  if(CFrameWnd::OnCreate(lpcs) == -1)
    return -1;

      // ID_TIMER_ELLIPSE là timerID của sự kiện Timer này
  if(!SetTimer(ID_TIMER_ELLIPSE, 100, NULL)) {
    MessageBox(_T("Error: SetTimer failed"));
    return -1;
  }
  return 0;
}
```

hay:

```
void CMainFrame::OnTimer(UINT nTimerID)
{
  CRect rect;
  GetClientRect(&rect);

  int x1 = rand() % rect.right;
  int x2 = rand() % rect.right;
  int y1 = rand() % rect.bottom;
  int y2 = rand() % rect.bottom;

  CClientDC dc(this);
  CBrush brush(RGB(rand() % 255, rand() % 255,
    rand() % 255));
  CBrush* pOldBrush = dc.SelectObject(&brush);
  dc.Ellipse(min(x1, x2), min(y1, y2), max(x1, x2),
    max(y1, y2));
  dc.SelectObject(pOldBrush);
}
```

Để ngừng sự kiện dạng Timer, dung hàm **KillTimer(timerID)**

Ví dụ 2:

```
KillTimer(ID_TIMER_ELLIPSE);
```

3.1.3 Idles

Việc làm xử lý gián đoạn trong hệ thống được giải quyết bởi hàm Idle hay sự kiện OnIdle

Ví dụ 1:

```
BOOL CMyApp::OnIdle(LONG lCount)
{
  BOOL bMFCContinue = CWinApp::OnIdle(lCount);
  BOOL bAppContinue = TRUE;
  if(lCount >= 2)
    bAppContinue = DoIdleWork(); // Do custom idle processing.
  return(bMFCContinue œœ bAppContinue);
}
```

Xử lý Idle là trái ngược với xử lý đa luồng (*multithreads*)

Ví dụ tổng hợp:

Resource.h

```
#define IDM_SYSMENU_FULL_WINDOW      16
#define IDM_SYSMENU_STAY_ON_TOP      32
#define IDI_APPICON                 100
```

Clock.rc

```
#include <afxres.h>
#include "Resource.h"

IDI_APPICON ICON Clock.ico
```

Clock.h

```
class CMyApp : public CWinApp
{
public:
    virtual BOOL InitInstance ();
};
class CMainWindow : public CFrameWnd
{
protected:
    BOOL m_bFullWindow;
    BOOL m_bStayOnTop;

    int m_nPrevSecond;
    int m_nPrevMinute;
    int m_nPrevHour;

    void DrawClockFace (CDC* pDC);
    void DrawSecondHand (CDC* pDC, int nLength, int nScale, int nDegrees,
        COLORREF clrColor);
    void DrawHand (CDC* pDC, int nLength, int nScale, int nDegrees,
        COLORREF clrColor);

    void SetTitleBarState ();
    void SetTopMostState ();
    void SaveWindowState ();
    void UpdateSystemMenu (CMenu* pMenu);

public:
    CMainWindow ();
```

```
    virtual BOOL PreCreateWindow (CREATESTRUCT& cs);
    BOOL RestoreWindowState ();

protected:
    afx_msg int OnCreate (LPCREATESTRUCT lpcs);
    afx_msg void OnGetMinMaxInfo (MINMAXINFO* pMMI);
    afx_msg void OnTimer (UINT nTimerID);
    afx_msg void OnPaint ();
    afx_msg UINT OnNcHitTest (CPoint point);
    afx_msg void OnSysCommand (UINT nID, LPARAM lParam);
    afx_msg void OnContextMenu (CWnd* pWnd, CPoint point);
    afx_msg void OnEndSession (BOOL bEnding);
    afx_msg void OnClose ();

    DECLARE_MESSAGE_MAP ()
};
```

Clock.cpp

```
#include <afxwin.h>
#include <math.h>
#include "Clock.h"
#include "Resource.h"
#define SQUARESIZE 20
#define ID_TIMER_CLOCK 1

CMyApp myApp;
////////////////////////////////////////////////////////////////////////
// CMyApp member functions

BOOL CMyApp::InitInstance ()
{
    SetRegistryKey (_T ("Programming Windows with MFC"));
    m_pMainWnd = new CMainWindow;
    if (!((CMainWindow*) m_pMainWnd)->RestoreWindowState ())
        m_pMainWnd->ShowWindow (m_nCmdShow);
    m_pMainWnd->UpdateWindow ();
    return TRUE;
}
////////////////////////////////////////////////////////////////////////
// CMainWindow message map and member functions
BEGIN_MESSAGE_MAP (CMainWindow, CFrameWnd)
    ON_WM_CREATE ()
    ON_WM_PAINT ()
    ON_WM_TIMER ()
    ON_WM_GETMINMAXINFO ()
    ON_WM_NCHITTEST ()
    ON_WM_SYSCOMMAND ()
    ON_WM_CONTEXTMENU ()
    ON_WM_ENDSESSION ()
    ON_WM_CLOSE ()
END_MESSAGE_MAP ()

CMainWindow::CMainWindow ()
{
    m_bAutoMenuEnable = FALSE;
```

```
    CTime time = CTime::GetCurrentTime ();
    m_nPrevSecond = time.GetSecond ();
    m_nPrevMinute = time.GetMinute ();
    m_nPrevHour = time.GetHour () % 12;

    CString strWndClass = AfxRegisterWndClass (
        CS_HREDRAW œ CS_VREDRAW,
        myApp.LoadStandardCursor (IDC_ARROW),

        (HBRUSH) (COLOR_3DFACE + 1),
        myApp.LoadIcon (IDI_APPICON)    );

    Create (strWndClass, _T ("Clock"));
}
BOOL CMainWindow::PreCreateWindow (CREATESTRUCT& cs)
{
    if (!CFrameWnd::PreCreateWindow (cs))
        return FALSE;

    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    return TRUE;
}
int CMainWindow::OnCreate (LPCREATESTRUCT lpcs)
{
    if (CFrameWnd::OnCreate (lpcs) == -1)
        return -1;
    //
    // Set a timer to fire at 1-second intervals.
    //
    if (!SetTimer (ID_TIMER_CLOCK, 1000, NULL)) {
        MessageBox (_T ("SetTimer failed"), _T ("Error"),
            MB_ICONSTOP œ MB_OK);
        return -1;
    }
    //
    // Customize the system menu.
    //
    CMenu* pMenu = GetSystemMenu (FALSE);
    pMenu->AppendMenu (MF_SEPARATOR);
    pMenu->AppendMenu (MF_STRING, IDM_SYSMENU_FULL_WINDOW,
        _T ("Remove &Title"));
    pMenu->AppendMenu (MF_STRING, IDM_SYSMENU_STAY_ON_TOP,
        _T ("Stay on To&p"));
    return 0;
}
void CMainWindow::OnClose ()
{
    SaveWindowState ();
    KillTimer (ID_TIMER_CLOCK);
    CFrameWnd::OnClose ();
}
void CMainWindow::OnEndSession (BOOL bEnding)
{
    if (bEnding)
        SaveWindowState ();
```

```
    CFrameWnd::OnEndSession (bEnding);
}
void CMainWindow::OnGetMinMaxInfo (MINMAXINFO* pMMI)
{
    pMMI->ptMinTrackSize.x = 120;
    pMMI->ptMinTrackSize.y = 120;
}
UINT CMainWindow::OnNcHitTest (CPoint point)
{
    UINT nHitTest = CFrameWnd::OnNcHitTest (point);
    if ((nHitTest == HTCLIENT) && (::GetAsyncKeyState (MK_LBUTTON) < 0))
        nHitTest = HTCAPTION;
    return nHitTest;
}
void CMainWindow::OnSysCommand (UINT nID, LPARAM lParam)
{
    UINT nMaskedID = nID & 0xFFF0;

    if (nMaskedID == IDM_SYSMENU_FULL_WINDOW) {
        m_bFullWindow = m_bFullWindow ? 0 : 1;
        SetTitleBarState ();
        return;
    }
    else if (nMaskedID == IDM_SYSMENU_STAY_ON_TOP) {
        m_bStayOnTop = m_bStayOnTop ? 0 : 1;
        SetTopMostState ();
        return;
    }
    CFrameWnd::OnSysCommand (nID, lParam);
}
void CMainWindow::OnContextMenu (CWnd* pWnd, CPoint point)
{
    CRect rect;
    GetClientRect (&rect);
    ClientToScreen (&rect);

    if (rect.PtInRect (point)) {
        CMenu* pMenu = GetSystemMenu (FALSE);
        UpdateSystemMenu (pMenu);

        int nID = (int) pMenu->TrackPopupMenu (TPM_LEFTALIGN œ
            TPM_LEFTBUTTON œ TPM_RIGHTBUTTON œ TPM_RETURNCMD, point.x,
            point.y, this);

        if (nID > 0)
            SendMessage (WM_SYSCOMMAND, nID, 0);
        return;
    }
    CFrameWnd::OnContextMenu (pWnd, point);
}
void CMainWindow::OnTimer (UINT nTimerID)
{
    //
    // Do nothing if the window is minimized.
    //
```

```
    if (IsIconic ())
        return;
    //
    // Get the current time and do nothing if it hasn't changed.
    //
    CTime time = CTime::GetCurrentTime ();
    int nSecond = time.GetSecond ();
    int nMinute = time.GetMinute ();
    int nHour = time.GetHour () % 12;

    if ((nSecond == m_nPrevSecond) &&
        (nMinute == m_nPrevMinute) &&
        (nHour == m_nPrevHour))
        return;
    //
    // Center the origin and switch to the MM_ISOTROPIC mapping mode.
    //
    CRect rect;
    GetClientRect (&rect);

    CClientDC dc (this);
    dc.SetMapMode (MM_ISOTROPIC);
    dc.SetWindowExt (1000, 1000);
    dc.SetViewportExt (rect.Width (), -rect.Height ());
    dc.SetViewportOrg (rect.Width () / 2, rect.Height () / 2);
    //
    // If minutes have changed, erase the hour and minute hands.
    //
    COLORREF clrColor = ::GetSysColor (COLOR_3DFACE);

    if (nMinute != m_nPrevMinute) {
        DrawHand (&dc, 200, 4, (m_nPrevHour * 30) + (m_nPrevMinute / 2),
            clrColor);
        DrawHand (&dc, 400, 8, m_nPrevMinute * 6, clrColor);
        m_nPrevMinute = nMinute;
        m_nPrevHour = nHour;
    }
    //
    // If seconds have changed, erase the second hand and redraw all hands.
    //
    if (nSecond != m_nPrevSecond) {
        DrawSecondHand (&dc, 400, 8, m_nPrevSecond * 6, clrColor);
        DrawSecondHand (&dc, 400, 8, nSecond * 6, RGB (0, 0, 0));
        DrawHand (&dc, 200, 4, (nHour * 30) + (nMinute / 2),
            RGB (0, 0, 0));
        DrawHand (&dc, 400, 8, nMinute * 6, RGB (0, 0, 0));
        m_nPrevSecond = nSecond;
    }
}
void CMainWindow::OnPaint ()
{
    CRect rect;
    GetClientRect (&rect);

    CPaintDC dc (this);
```

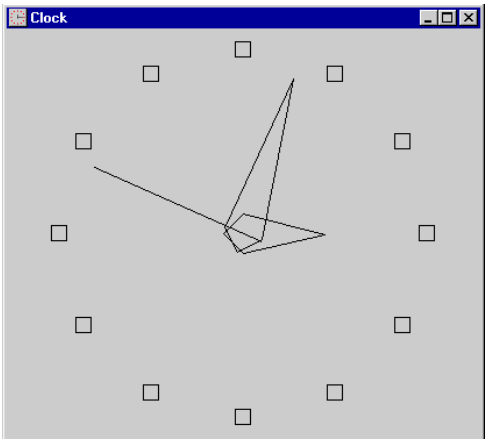
```
    dc.SetMapMode (MM_ISOTROPIC);
    dc.SetWindowExt (1000, 1000);
    dc.SetViewportExt (rect.Width (), -rect.Height ());
    dc.SetViewportOrg (rect.Width () / 2, rect.Height () / 2);

    DrawClockFace (&dc);
    DrawHand (&dc, 200, 4, (m_nPrevHour * 30) +
        (m_nPrevMinute / 2), RGB (0, 0, 0));
    DrawHand (&dc, 400, 8, m_nPrevMinute * 6, RGB (0, 0, 0));
    DrawSecondHand (&dc, 400, 8, m_nPrevSecond * 6, RGB (0, 0, 0));
}
void CMainWindow::DrawClockFace (CDC* pDC)
{
    static CPoint point[12] = {
        CPoint (   0,  450),    // 12 o'clock
        CPoint ( 225,  390),    //  1 o'clock
        CPoint ( 390,  225),    //  2 o'clock
        CPoint ( 450,    0),    //  3 o'clock
        CPoint ( 390, -225),    //  4 o'clock
        CPoint ( 225, -390),    //  5 o'clock
        CPoint (   0, -450),    //  6 o'clock
        CPoint (-225, -390),    //  7 o'clock
        CPoint (-390, -225),    //  8 o'clock
        CPoint (-450,    0),    //  9 o'clock
        CPoint (-390,  225),    // 10 o'clock
        CPoint (-225,  390),    // 11 o'clock
    };

    pDC->SelectStockObject (NULL_BRUSH);

    for (int i=0; i<12; i++)
        pDC->Rectangle (point[i].x - SQUARESIZE,
            point[i].y + SQUARESIZE, point[i].x + SQUARESIZE,
            point[i].y - SQUARESIZE);
}
void CMainWindow::DrawHand (CDC* pDC, int nLength, int nScale,
    int nDegrees, COLORREF clrColor)
{
    CPoint point[4];
    double nRadians = (double) nDegrees * 0.017453292;

    point[0].x = (int) (nLength * sin (nRadians));
    point[0].y = (int) (nLength * cos (nRadians));

    point[2].x = -point[0].x / nScale;
    point[2].y = -point[0].y / nScale;

    point[1].x = -point[2].y;
    point[1].y = point[2].x;

    point[3].x = -point[1].x;
    point[3].y = -point[1].y;

    CPen pen (PS_SOLID, 0, clrColor);
    CPen* pOldPen = pDC->SelectObject (&pen);
```

```
    pDC->MoveTo (point[0]);
    pDC->LineTo (point[1]);
    pDC->LineTo (point[2]);
    pDC->LineTo (point[3]);
    pDC->LineTo (point[0]);

    pDC->SelectObject (pOldPen);
}
void CMainWindow::DrawSecondHand (CDC* pDC, int nLength, int nScale,
    int nDegrees, COLORREF clrColor)
{
    CPoint point[2];
    double nRadians = (double) nDegrees * 0.017453292;

    point[0].x = (int) (nLength * sin (nRadians));
    point[0].y = (int) (nLength * cos (nRadians));

    point[1].x = -point[0].x / nScale;
    point[1].y = -point[0].y / nScale;

    CPen pen (PS_SOLID, 0, clrColor);
    CPen* pOldPen = pDC->SelectObject (&pen);

    pDC->MoveTo (point[0]);
    pDC->LineTo (point[1]);

    pDC->SelectObject (pOldPen);
}
void CMainWindow::SetTitleBarState ()
{
    CMenu* pMenu = GetSystemMenu (FALSE);

    if (m_bFullWindow ) {
        ModifyStyle (WS_CAPTION, 0);
        pMenu->ModifyMenu (IDM_SYSMENU_FULL_WINDOW, MF_STRING,
            IDM_SYSMENU_FULL_WINDOW, _T ("Restore &Title"));
    }
    else {
        ModifyStyle (0, WS_CAPTION);
        pMenu->ModifyMenu (IDM_SYSMENU_FULL_WINDOW, MF_STRING,
            IDM_SYSMENU_FULL_WINDOW, _T ("Remove &Title"));
    }
    SetWindowPos (NULL, 0, 0, 0, 0, SWP_NOMOVE œ SWP_NOSIZE œ
        SWP_NOZORDER œ SWP_DRAWFRAME);
}
void CMainWindow::SetTopMostState ()
{
    CMenu* pMenu = GetSystemMenu (FALSE);

    if (m_bStayOnTop) {
        SetWindowPos (&wndTopMost, 0, 0, 0, 0, SWP_NOMOVE œ SWP_NOSIZE);
        pMenu->CheckMenuItem (IDM_SYSMENU_STAY_ON_TOP, MF_CHECKED);
    }
    else {
```

```
        SetWindowPos (&wndNoTopMost, 0, 0, 0, 0, SWP_NOMOVE œ SWP_NOSIZE);
        pMenu->CheckMenuItem (IDM_SYSMENU_STAY_ON_TOP, MF_UNCHECKED);
    }
}
BOOL CMainWindow::RestoreWindowState ()
{
    CString version = _T ("Version 1.0");
    m_bFullWindow = myApp.GetProfileInt (version, _T ("FullWindow"), 0);
    SetTitleBarState ();
    m_bStayOnTop = myApp.GetProfileInt (version, _T ("StayOnTop"), 0);
    SetTopMostState ();

    WINDOWPLACEMENT wp;
    wp.length = sizeof (WINDOWPLACEMENT);
    GetWindowPlacement (&wp);

    if (((wp.flags =
            myApp.GetProfileInt (version, _T ("flags"), -1)) != -1) &&
        ((wp.showCmd =
            myApp.GetProfileInt (version, _T ("showCmd"), -1)) != -1) &&
        ((wp.rcNormalPosition.left =
            myApp.GetProfileInt (version, _T ("x1"), -1)) != -1) &&
        ((wp.rcNormalPosition.top =
            myApp.GetProfileInt (version, _T ("y1"), -1)) != -1) &&
        ((wp.rcNormalPosition.right =
            myApp.GetProfileInt (version, _T ("x2"), -1)) != -1) &&
        ((wp.rcNormalPosition.bottom =
            myApp.GetProfileInt (version, _T ("y2"), -1)) != -1)) {

        wp.rcNormalPosition.left = min (wp.rcNormalPosition.left,
            ::GetSystemMetrics (SM_CXSCREEN) -
            ::GetSystemMetrics (SM_CXICON));
        wp.rcNormalPosition.top = min (wp.rcNormalPosition.top,
            ::GetSystemMetrics (SM_CYSCREEN) -
            ::GetSystemMetrics (SM_CYICON));
        SetWindowPlacement (&wp);
        return TRUE;
    }
    return FALSE;
}
void CMainWindow::SaveWindowState ()
{
    CString version = _T ("Version 1.0");
    myApp.WriteProfileInt (version, _T ("FullWindow"), m_bFullWindow);
    myApp.WriteProfileInt (version, _T ("StayOnTop"), m_bStayOnTop);

    WINDOWPLACEMENT wp;
    wp.length = sizeof (WINDOWPLACEMENT);
    GetWindowPlacement (&wp);

    myApp.WriteProfileInt (version, _T ("flags"), wp.flags);
    myApp.WriteProfileInt (version, _T ("showCmd"), wp.showCmd);
    myApp.WriteProfileInt (version, _T ("x1"), wp.rcNormalPosition.left);
    myApp.WriteProfileInt (version, _T ("y1"), wp.rcNormalPosition.top);
    myApp.WriteProfileInt (version, _T ("x2"), wp.rcNormalPosition.right);
```

```
    myApp.WriteProfileInt (version, _T ("y2"), wp.rcNormalPosition.bottom);
}
void CMainWindow::UpdateSystemMenu (CMenu* pMenu)
{
    static UINT nState[2][5] = {
        { MFS_GRAYED,  MFS_ENABLED, MFS_ENABLED,
          MFS_ENABLED, MFS_DEFAULT },
        { MFS_DEFAULT, MFS_GRAYED,  MFS_GRAYED,
          MFS_ENABLED, MFS_GRAYED  }
    };
    if (IsIconic ()) // Shouldn't happen, but let's be safe
        return;
    int i = 0;
    if (IsZoomed ())
        i = 1;
    CString strMenuText;
    pMenu->GetMenuString (SC_RESTORE, strMenuText, MF_BYCOMMAND);
    pMenu->ModifyMenu (SC_RESTORE, MF_STRING œ nState[i][0], SC_RESTORE,
        strMenuText);

    pMenu->GetMenuString (SC_MOVE, strMenuText, MF_BYCOMMAND);
    pMenu->ModifyMenu (SC_MOVE, MF_STRING œ nState[i][1], SC_MOVE,
        strMenuText);

    pMenu->GetMenuString (SC_SIZE, strMenuText, MF_BYCOMMAND);
    pMenu->ModifyMenu (SC_SIZE, MF_STRING œ nState[i][2], SC_SIZE,
        strMenuText);

    pMenu->GetMenuString (SC_MINIMIZE, strMenuText, MF_BYCOMMAND);
    pMenu->ModifyMenu (SC_MINIMIZE, MF_STRING œ nState[i][3], SC_MINIMIZE,
        strMenuText);

    pMenu->GetMenuString (SC_MAXIMIZE, strMenuText, MF_BYCOMMAND);
    pMenu->ModifyMenu (SC_MAXIMIZE, MF_STRING œ nState[i][4], SC_MAXIMIZE,
        strMenuText);

    SetMenuDefaultItem (pMenu->m_hMenu, i ? SC_RESTORE :
        SC_MAXIMIZE, FALSE);
}
```

Màn hình kết quả như sau:



3.2 THREADS

3.2.1 Vấn đề quan tâm

- Hiểu và sử dụng được kỹ thuật lập trình về thread.

3.2.2 Giới thiệu

Thread (*tiểu trình/luồng*) là một phần lệnh của chương trình ứng dụng thực thi với mục đích độc lập và không phân chia nhỏ hơn nữa.

Một quá trình trong ứng dụng dạng Win-32 bit được bắt đầu như đơn luồng (single thread) nhưng cũng có thể thêm nhiều luồng khác để trở thành đa luồng (multi threads)

Việc xây dựng ứng dụng dạng multi threads không đơn giản và không dành cho tất cả các dạng ứng dụng mà chỉ dàng cho các ứng dụng dạng lập trình/xử lý song song (*paralell processing*)

MFC hỗ trợ việc lập trình với thread bởi lớp **CWinThread**

3.2.3 Threads

Để tạo một thread, gọi hàm **AfxBeginThread**

Ví dụ 1:

```
CWinThread* pThread = AfxBeginThread(ThreadFunc, &threadInfo);
Và
UINT ThreadFunc(LPVOID pParam)
{
  UINT nIterations =(UINT) pParam;
  for(UINT i=0; i<nIterations; i++);
  return 0;
}
```

Dạng tổng quát của **AfxBeginThread** là:

```
CWinThread* AfxBeginThread(AFX_THREADPROC pfnThreadProc,
 LPVOID pParam, int nPriority = THREAD_PRIORITY_NORMAL,
 UINT nStackSize = 0, DWORD dwCreateFlags = 0,
 LPSECURITY_ATTRIBUTES lpSecurityAttrs = NULL)
```

Có thể dùng hàm **SetThreadPriority** để đặt độ ưu tiên cho mỗi thread

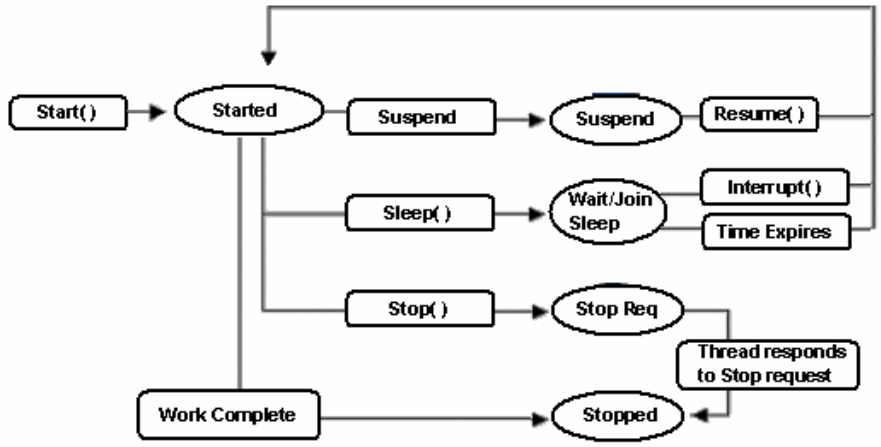
Có thể tạo 1 class liên kết với mỗi thread

Ví dụ 2:

```
// The CUIThread class
class CUIThread : public CWinThread
{
  DECLARE_DYNCREATE(CUIThread)

public:
  virtual BOOL InitInstance();
};
IMPLEMENT_DYNCREATE(CUIThread, CWinThread)

BOOL CUIThread::InitInstance()
{
  m_pMainWnd = new CMainWindow;
  m_pMainWnd->ShowWindow(SW_SHOW);
  m_pMainWnd->UpdateWindow();
  return TRUE;
}
// The CMainWindow class
class CMainWindow : public CFrameWnd
{
public:
```

```
  CMainWindow();

protected:
  afx_msg void OnLButtonDown(UINT, CPoint);
  DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CMainWindow, CFrameWnd)
  ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()

CMainWindow::CMainWindow()
{
  Create(NULL, _T("UI Thread Window"));
}
void CMainWindow::OnLButtonDown(UINT nFlags, CPoint point)
{
  PostMessage(WM_CLOSE, 0, 0);
}
```

Mỗi thread có thể chuyển đổi qua nhiều trạng thái theo như sơ đồ sau:



Có thể tạm ngừng (*Suspend*) và khôi phục (*Resume*) việc hoạt động của thread bởi các lệnh **SuspendThread** và **ResumeThread**

Có thể đặt thread vào trạng thái tạm ngừng trong môt thời gian hạn định bằng việc dùng lệnh **Sleep**

Kết thúc một thread bằng việc dùng lệnh **AfxEndThread**

Đặt độ ưu tiên cho thread bằng SetPriorityClass, với các giá trị có thể chọn như sau:

Độ ưu tiên	Mô tả
IDLE_PRIORITY_CLASS	The process runs only when the system is idle—for example, when no other thread is waiting for a given CPU.
NORMAL_PRIORITY_CLASS	The default process priority class. The process has no special scheduling needs.
HIGH_PRIORITY_CLASS	The process receives priority over IDLE_PRIORITY_CLASS and NORMAL_PRIORITY_CLASS processes.
REALTIME_PRIORITY_CLASS	The process must have the highest possible priority, and its threads should preempt even threads belonging to HIGH_PRIORITY_CLASS processes.

Và:

Mức ưu tiên	Mô tả
THREAD_PRIORITY_IDLE	The thread's base priority level is 1 if the process's priority class is HIGH_PRIORITY_CLASS or lower, or 16 if the process's priority class is REALTIME_PRIORITY_CLASS.
THREAD_PRIORITY_LOWEST	The thread's base priority level is equal to the

	process's priority class minus 2.
THREAD_PRIORITY_BELOW_NORMAL	The thread's base priority level is equal to the process's priority class minus 1.
THREAD_PRIORITY_NORMAL	The default thread priority value. The thread's base priority level is equal to the process's priority class.
THREAD_PRIORITY_ABOVE_NORMAL	The thread's base priority level is equal to the process's priority class plus 1.
THREAD_PRIORITY_HIGHEST	The thread's base priority level is equal to the process's priority class plus 2.
THREAD_PRIORITY_TIME_CRITICAL	The thread's base priority level is 15 if the process's priority class is HIGH_PRIORITY_CLASS or lower, or 31 if the process's priority class is REALTIME_PRIORITY_CLASS.

Ví dụ 3:

Trong file MyDlg.h

```
#define WM_USER_THREAD_FINISHED WM_USER+0x100

UINT ThreadFunc(LPVOID pParam);
int MyFunc(int nMax);

typedef struct tagTHREADPARMS {
  int nMax;
  HWND hWnd;
} THREADPARMS;

class CMyDlg : public CDialog
{
…
// Implementation
protected:
  HICON m_hIcon;

  // Generated message map functions
  //{{AFX_MSG(CMyDlg)
  virtual BOOL OnInitDialog();
  afx_msg void OnPaint();
  afx_msg HCURSOR OnQueryDragIcon();
  afx_msg void OnStart();
  //}}AFX_MSG
  afx_msg LONG OnThreadFinished(WPARAM wParam, LPARAM lParam);
  DECLARE_MESSAGE_MAP()
…
};
```

Trong file MyDlg.cpp

```
BEGIN_MESSAGE_MAP(CMyDlg, CDialog)
  //{{AFX_MSG_MAP(CMyDlg)
  ON_BN_CLICKED(IDC_START, OnStart)
  //}}AFX_MSG_MAP
  ON_MESSAGE(WM_USER_THREAD_FINISHED, OnThreadFinished)
END_MESSAGE_MAP()
…
void CMyDlg::OnStart()
{
  int nMax = GetDlgItemInt(IDC_MAX);
```

```
  if(nMax < 10) {
    MessageBox(_T("The number you enter must be 10 or higher"));
    GetDlgItem(IDC_MAX)->SetFocus();
    return;
  }

  SetDlgItemText(IDC_RESULT, _T(""));
  GetDlgItem(IDC_START)->EnableWindow(FALSE);

  THREADPARMS* ptp = new THREADPARMS;
  ptp->nMax = nMax;
  ptp->hWnd = m_hWnd;
  AfxBeginThread(ThreadFunc, ptp);
}
LONG CMyDlg::OnThreadFinished(WPARAM wParam, LPARAM lParam)
{
  SetDlgItemInt(IDC_RESULT,(int) wParam);
  GetDlgItem(IDC_START)->EnableWindow(TRUE);
  return 0;
}
////////////////////////////////////////////////////////////////////////////
// Global functions
UINT ThreadFunc(LPVOID pParam)
{
  THREADPARMS* ptp =(THREADPARMS*) pParam;
  int nMax = ptp->nMax;
  HWND hWnd = ptp->hWnd;
  delete ptp;

  int nCount = MyFunction(nMax);
  ::PostMessage(hWnd, WM_USER_THREAD_FINISHED,(WPARAM) nCount, 0);
  return 0;
}
int MyFunction(int nMax)
{
  PBYTE pBuffer = new BYTE[nMax + 1];
  ::FillMemory(pBuffer, nMax + 1, 1);

  int nLimit = 2;
  while(nLimit * nLimit < nMax)
    nLimit++;

  for(int i=2; i<=nLimit; i++) {
    if(pBuffer[i]) {
      for(int k=i + i; k<=nMax; k+=i)
        pBuffer[k] = 0;
    }
  }

  int nCount = 0;
  for(i=2; i<=nMax; i++)
    if(pBuffer[i])
      nCount++;

  delete[] pBuffer;
```

```
  return nCount;
}
```

Ví dụ tổng hợp:

Sieve.h

```
// Sieve.h : main header file for the SIEVE application
//
#if !defined(AFX_SIEVE_H__6DF40C9B_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_)
#define AFX_SIEVE_H__6DF40C9B_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
    #error include `stdafx.h' before including this file for PCH
#endif
#include "resource.h"        // main symbols
/////////////////////////////////////////////////////////////////////////////
// CSieveApp:
// See Sieve.cpp for the implementation of this class
//
class CSieveApp : public CWinApp
{
public:
    CSieveApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSieveApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation

    //{{AFX_MSG(CSieveApp)
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
// immediately before the previous line.

#endif
// !defined(AFX_SIEVE_H__6DF40C9B_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_)
```

Sieve.cpp

```
// Sieve.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "Sieve.h"
#include "SieveDlg.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSieveApp
BEGIN_MESSAGE_MAP(CSieveApp, CWinApp)
    //{{AFX_MSG_MAP(CSieveApp)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSieveApp construction
CSieveApp::CSieveApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CSieveApp object
CSieveApp theApp;
/////////////////////////////////////////////////////////////////////////////
// CSieveApp initialization
BOOL CSieveApp::InitInstance()
{
    // Standard initialization
    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

    CSieveDlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with Cancel
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    //  application, rather than start the application's message pump.
    return FALSE;
}
```

SieveDlg.h

```
// SieveDlg.h : header file
//
```

```
#if !defined(
    AFX_SIEVEDLG_H__6DF40C9D_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_)
#define AFX_SIEVEDLG_H__6DF40C9D_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#define WM_USER_THREAD_FINISHED WM_USER+0x100
UINT ThreadFunc (LPVOID pParam);
int Sieve (int nMax);
typedef struct tagTHREADPARMS {
    int nMax;
    HWND hWnd;
} THREADPARMS;
/////////////////////////////////////////////////////////////////////////////
// CSieveDlg dialog
class CSieveDlg : public CDialog
{
// Construction
public:
    CSieveDlg(CWnd* pParent = NULL);    // standard constructor
// Dialog Data
    //{{AFX_DATA(CSieveDlg)
    enum { IDD = IDD_SIEVE_DIALOG };
        // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSieveDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    HICON m_hIcon;

    // Generated message map functions
    //{{AFX_MSG(CSieveDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnStart();
    //}}AFX_MSG
    afx_msg LONG OnThreadFinished (WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_SIEVEDLG_H__6DF40C9D_7EA1_11D1_8E53_E4D9F9C00000__INCLUDED_)
```

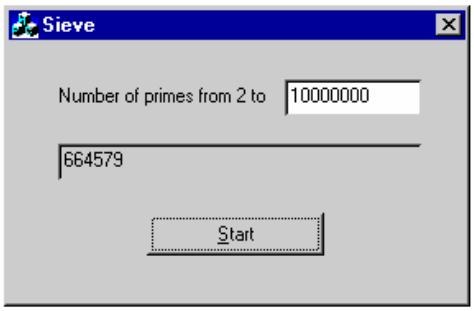
SieveDlg.cpp

```
// SieveDlg.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "Sieve.h"
#include "SieveDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CSieveDlg dialog
CSieveDlg::CSieveDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CSieveDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CSieveDlg)
        // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent
    // DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CSieveDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSieveDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSieveDlg, CDialog)
    //{{AFX_MSG_MAP(CSieveDlg)
    ON_BN_CLICKED(IDC_START, OnStart)
    //}}AFX_MSG_MAP
    ON_MESSAGE (WM_USER_THREAD_FINISHED, OnThreadFinished)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSieveDlg message handlers
BOOL CSieveDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    SetIcon(m_hIcon, TRUE);
    SetIcon(m_hIcon, FALSE);
    return TRUE;
}
void CSieveDlg::OnStart()
{
    int nMax = GetDlgItemInt (IDC_MAX);
    if (nMax < 10) {
        MessageBox (_T ("The number you enter must be 10 or higher"));
        GetDlgItem (IDC_MAX)->SetFocus ();
        return;
    }

    SetDlgItemText (IDC_RESULT, _T (""));
    GetDlgItem (IDC_START)->EnableWindow (FALSE);
```

```
    THREADPARMS* ptp = new THREADPARMS;
    ptp->nMax = nMax;
    ptp->hWnd = m_hWnd;
    AfxBeginThread (ThreadFunc, ptp);
}
LONG CSieveDlg::OnThreadFinished (WPARAM wParam, LPARAM lParam)
{
    SetDlgItemInt (IDC_RESULT, (int) wParam);
    GetDlgItem (IDC_START)->EnableWindow (TRUE);
    return 0;
}
////////////////////////////////////////////////////////////////////////
// Global functions
UINT ThreadFunc (LPVOID pParam)
{
    THREADPARMS* ptp = (THREADPARMS*) pParam;
    int nMax = ptp->nMax;
    HWND hWnd = ptp->hWnd;
    delete ptp;

    int nCount = Sieve (nMax);
    ::PostMessage (hWnd, WM_USER_THREAD_FINISHED, (WPARAM) nCount, 0);
    return 0;
}
int Sieve(int nMax)
{
    PBYTE pBuffer = new BYTE[nMax + 1];
    ::FillMemory (pBuffer, nMax + 1, 1);

    int nLimit = 2;
    while (nLimit * nLimit < nMax)
        nLimit++;

    for (int i=2; i<=nLimit; i++) {
        if (pBuffer[i]) {
            for (int k=i + i; k<=nMax; k+=i)
                pBuffer[k] = 0;
        }
    }
    int nCount = 0;
    for (i=2; i<=nMax; i++)
        if (pBuffer[i])
            nCount++;

    delete[] pBuffer;
    return nCount;
}
```

Màn hình kết quả như sau:



3.2.4 Đồng bộ các threads(Thread Synchronization)

(*Xem Chương 17, phần Thread Synchronization, sách Programming Windows with MFC, 2nd edition*)

Windows cung cấp 4 kiểu của đối tượng đồng bộ mà có thể dùng để đồng bộ các hành động của các thread hoạt động song hành như sau:

- Critical sections
- Mutexes
- Events
- Semaphores

Việc đồng bộ các thread bởi các đối tượng này thông qua việc xử lý hiện thực tác vụ **Lock** và **Unlock** để điều phối quyền truy xuất dử liệu/bộ nhớ đồng thời hay không.

Ngoài ra MFC còn cung cấp lớp ***CSingleLock*** and ***CMultiLock*** để trợ gúp việc xử lý hiện thực tác vụ Lock và Unlock của các đối tượng trên một cách đơn giản hơn.

```
CCriticalSection g_cs;

CSingleLock lock (&g_cs); // Wrap it in a CSingleLock.
lock.Lock ();            // Lock the critical section.
```

hay

```
CMutex g_mutex;
CEvent g_event[2];
CSyncObject* g_pObjects[3] = { &g_mutex, &g_event[0], &g_event[1] };

// Block until all three objects become signaled.
CMultiLock multiLock (g_pObjects, 3);
multiLock.Lock ();

// Block until one of the three objects becomes signaled.
CMultiLock multiLock (g_pObjects, 3);
multiLock.Lock (INFINITE, FALSE);
```

Ví dụ tổng hợp:

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_MAINFRM_H__9D77AEE8_AA14_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__9D77AEE8_AA14_11D2_8E53_006008A82731__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)
```

```
// Attributes
public:
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
    CSpecialStatusBar m_wndStatusBar;
// Generated message map functions protected:
    int m_nPercentDone;
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg BOOL OnQueryNewPalette();
    afx_msg void OnPaletteChanged(CWnd* pFocusWnd);
    //}}AFX_MSG
    afx_msg LRESULT OnUpdateImageStats (WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnThreadUpdate (WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnThreadFinished (WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnThreadAborted (WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(AFX_MAINFRM_H__9D77AEE8_AA14_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "ImageEdit.h"
#include "ImageEditDoc.h"
#include "SpecialStatusBar.h"
#include "MainFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
```

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)

    ON_WM_CREATE()
    ON_WM_QUERYNEWPALETTE()
    ON_WM_PALETTECHANGED()
    //}}AFX_MSG_MAP
    ON_MESSAGE (WM_USER_UPDATE_STATS, OnUpdateImageStats)
    ON_MESSAGE (WM_USER_THREAD_UPDATE, OnThreadUpdate)
    ON_MESSAGE (WM_USER_THREAD_FINISHED, OnThreadFinished)
    ON_MESSAGE (WM_USER_THREAD_ABORTED, OnThreadAborted)
END_MESSAGE_MAP()
static UINT indicators[] =
{
    ID_SEPARATOR,
    ID_SEPARATOR,
    ID_SEPARATOR
};
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
    m_nPercentDone = -1;
}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndStatusBar.Create(this))
    {
        TRACE0("Failed to create status bar\n");
        return -1;      // fail to create
    }
    return 0;
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
```

```
    CFrameWnd::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
BOOL CMainFrame::OnQueryNewPalette()
{
    CDocument* pDoc = GetActiveDocument ();
    if (pDoc != NULL)
        GetActiveDocument ()->UpdateAllViews (NULL);
    return TRUE;
}
void CMainFrame::OnPaletteChanged(CWnd* pFocusWnd)
{
    if (pFocusWnd != this) {
        CDocument* pDoc = GetActiveDocument ();
        if (pDoc != NULL)
            GetActiveDocument ()->UpdateAllViews (NULL);
    }
}
LRESULT CMainFrame::OnUpdateImageStats (WPARAM wParam, LPARAM lParam)
{
    m_wndStatusBar.SetImageStats ((LPCTSTR) lParam);
    return 0;
}
LRESULT CMainFrame::OnThreadUpdate (WPARAM wParam, LPARAM lParam)
{
    int nPercentDone = ((int) wParam * 100) / (int) lParam;
    if (nPercentDone != m_nPercentDone) {
        m_wndStatusBar.SetProgress (nPercentDone);
        m_nPercentDone = nPercentDone;
    }
    return 0;
}
LRESULT CMainFrame::OnThreadFinished (WPARAM wParam, LPARAM lParam)
{
    CImageEditDoc* pDoc = (CImageEditDoc*) GetActiveDocument ();
    if (pDoc != NULL) {
        pDoc->ThreadFinished ();
        m_wndStatusBar.SetProgress (0);
        m_nPercentDone = -1;
    }
    return 0;
}
LRESULT CMainFrame::OnThreadAborted (WPARAM wParam, LPARAM lParam)
{
    CImageEditDoc* pDoc = (CImageEditDoc*) GetActiveDocument ();
    if (pDoc != NULL) {
        pDoc->ThreadAborted ();
        m_wndStatusBar.SetProgress (0);
        m_nPercentDone = -1;
    }
    return 0;
}
```

ImageEditDoc.h

```
// ImageEditDoc.h : interface of the CImageEditDoc class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_IMAGEEDITDOC_H__9D77AEEA_AA14_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_IMAGEEDITDOC_H__9D77AEEA_AA14_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
UINT ThreadFunc (LPVOID pParam);
LOGPALETTE* CreateGrayScale ();
class CImageEditDoc : public CDocument
{
protected: // create from serialization only
    CImageEditDoc();
    DECLARE_DYNCREATE(CImageEditDoc)
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CImageEditDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
    virtual void DeleteContents();
    //}}AFX_VIRTUAL

// Implementation
public:
    void ThreadAborted();
    void ThreadFinished();
    CPalette* GetPalette();
    CBitmap* GetBitmap();
    virtual ~CImageEditDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
    CCriticalSection m_cs;
    CEvent m_event;
    HANDLE m_hThread;
    BOOL m_bWorking;
    CPalette m_palette;
    CBitmap m_bitmap;
    //{{AFX_MSG(CImageEditDoc)
    afx_msg void OnGrayScale();
    afx_msg void OnUpdateGrayScale(CCmdUI* pCmdUI);
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_IMAGEEDITDOC_H__9D77AEEA_AA14_11D2_8E53_006008A82731__INCLUDED_)
```

ImageEditDoc.cpp

```
// ImageEditDoc.cpp : implementation of the CImageEditDoc class
//
#include "stdafx.h"
#include "ImageEdit.h"
#include "ImageEditDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CImageEditDoc
IMPLEMENT_DYNCREATE(CImageEditDoc, CDocument)
BEGIN_MESSAGE_MAP(CImageEditDoc, CDocument)
    //{{AFX_MSG_MAP(CImageEditDoc)
    ON_COMMAND(ID_EFFECTS_GRAY_SCALE, OnGrayScale)
    ON_UPDATE_COMMAND_UI(ID_EFFECTS_GRAY_SCALE, OnUpdateGrayScale)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CImageEditDoc construction/destruction
CImageEditDoc::CImageEditDoc() :
    m_event (FALSE, TRUE) // Manual-reset event, initially unowned
{
    m_hThread = NULL;
    m_bWorking = FALSE;
}
CImageEditDoc::~CImageEditDoc()
{
}
BOOL CImageEditDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;
    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CImageEditDoc diagnostics
#ifdef _DEBUG
void CImageEditDoc::AssertValid() const
{
    CDocument::AssertValid();
}
```

```
void CImageEditDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG
////////////////////////////////////////////////////////////////////////
// CImageEditDoc commands
BOOL CImageEditDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
    //
    // Return now if an image is being processed.
    //
    if (m_bWorking) {
        AfxMessageBox (_T ("You can't open an image while another is " \
            "being converted"));
        return FALSE;
    }
    //
    // Let the base class do its thing.
    //
    if (!CDocument::OnOpenDocument (lpszPathName))
        return FALSE;
    //
    // Open the file and create a DIB section from its contents.
    //
    HBITMAP hBitmap = (HBITMAP) ::LoadImage (NULL, lpszPathName,
        IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE œ LR_CREATEDIBSECTION);

    if (hBitmap == NULL) {
        CString string;
        string.Format (_T ("%s does not contain a DIB"), lpszPathName);
        AfxMessageBox (string);
        return FALSE;
    }
    m_bitmap.Attach (hBitmap);
    //
    // Return now if this device doesn't support palettes.
    //
    CClientDC dc (NULL);
    if ((dc.GetDeviceCaps (RASTERCAPS) & RC_PALETTE) == 0)
        return TRUE;
    //
    // Create a palette to go with the DIB section.
    //
    if ((HBITMAP) m_bitmap != NULL) {
        DIBSECTION ds;
        m_bitmap.GetObject (sizeof (DIBSECTION), &ds);

        int nColors;
        if (ds.dsBmih.biClrUsed != 0)
            nColors = ds.dsBmih.biClrUsed;
        else
            nColors = 1 << ds.dsBmih.biBitCount;
        //
        // Create a halftone palette if the DIB section contains more
```

```
        // than 256 colors.
        //
        if (nColors > 256)
            m_palette.CreateHalftonePalette (&dc);
        //
        // Create a custom palette from the DIB section's color table
        // if the number of colors is 256 or less.
        //
        else {
            RGBQUAD* pRGB = new RGBQUAD[nColors];

            CDC memDC;
            memDC.CreateCompatibleDC (&dc);
            CBitmap* pOldBitmap = memDC.SelectObject (&m_bitmap);
            ::GetDIBColorTable ((HDC) memDC, 0, nColors, pRGB);
            memDC.SelectObject (pOldBitmap);

            UINT nSize = sizeof (LOGPALETTE) +
                (sizeof (PALETTEENTRY) * (nColors - 1));
            LOGPALETTE* pLP = (LOGPALETTE*) new BYTE[nSize];

            pLP->palVersion = 0x300;
            pLP->palNumEntries = nColors;

            for (int i=0; i<nColors; i++) {
                pLP->palPalEntry[i].peRed = pRGB[i].rgbRed;
                pLP->palPalEntry[i].peGreen = pRGB[i].rgbGreen;
                pLP->palPalEntry[i].peBlue = pRGB[i].rgbBlue;
                pLP->palPalEntry[i].peFlags = 0;
            }

            m_palette.CreatePalette (pLP);
            delete[] pLP;
            delete[] pRGB;
        }
    }
    return TRUE;
}
void CImageEditDoc::DeleteContents()
{
    if ((HBITMAP) m_bitmap != NULL)
        m_bitmap.DeleteObject ();

    if ((HPALETTE) m_palette != NULL)
        m_palette.DeleteObject ();
    CDocument::DeleteContents();
}
CBitmap* CImageEditDoc::GetBitmap()
{
    return ((HBITMAP) m_bitmap == NULL) ? NULL : &m_bitmap;
}
CPalette* CImageEditDoc::GetPalette()
{
    return ((HPALETTE) m_palette == NULL) ? NULL : &m_palette;
}
```

```
void CImageEditDoc::ThreadFinished()
{
    ASSERT (m_hThread != NULL);
    ::WaitForSingleObject (m_hThread, INFINITE);
    ::CloseHandle (m_hThread);
    m_hThread = NULL;
    m_bWorking = FALSE;
    //
    // Replace the current palette with a gray scale palette.
    //
    if ((HPALETTE) m_palette != NULL) {
        m_palette.DeleteObject ();
        LOGPALETTE* pLP = CreateGrayScale ();
        m_palette.CreatePalette (pLP);
        delete[] pLP;
    }
    //
    // Tell the view to repaint.
    //
    UpdateAllViews (NULL);
}
void CImageEditDoc::ThreadAborted()
{
    ASSERT (m_hThread != NULL);
    ::WaitForSingleObject (m_hThread, INFINITE);
    ::CloseHandle (m_hThread);
    m_hThread = NULL;
    m_bWorking = FALSE;
}
void CImageEditDoc::OnGrayScale()
{
    if (!m_bWorking) {
        m_bWorking = TRUE;
        m_event.ResetEvent ();
        //
        // Package data to pass to the image processing thread.
        //
        THREADPARMS* ptp = new THREADPARMS;
        ptp->pWnd = AfxGetMainWnd ();
        ptp->pBitmap = &m_bitmap;
        ptp->pPalette = &m_palette;
        ptp->pCriticalSection = &m_cs;
        ptp->pEvent = &m_event;
        //
        // Start the image processing thread and duplicate its handle.
        //
        CWinThread* pThread = AfxBeginThread (ThreadFunc, ptp,
            THREAD_PRIORITY_NORMAL, 0, CREATE_SUSPENDED);

        ::DuplicateHandle (GetCurrentProcess (),
            pThread->m_hThread, GetCurrentProcess (), &m_hThread,
            0, FALSE, DUPLICATE_SAME_ACCESS);

        pThread->ResumeThread ();
    }
```

```
    else
        //
        // Kill the image processing thread.
        //
        m_event.SetEvent ();
}
void CImageEditDoc::OnUpdateGrayScale(CCmdUI* pCmdUI)
{
    if (m_bWorking) {
        pCmdUI->SetText (_T ("Stop &Gray Scale Conversion"));
        pCmdUI->Enable ();
    }
    else {
        pCmdUI->SetText (_T ("Convert to &Gray Scale"));
        pCmdUI->Enable ((HBITMAP) m_bitmap != NULL);
    }
}
////////////////////////////////////////////////////////////////////////
// Thread function and other globals
UINT ThreadFunc (LPVOID pParam)
{
    THREADPARMS* ptp = (THREADPARMS*) pParam;
    CWnd* pWnd = ptp->pWnd;
    CBitmap* pBitmap = ptp->pBitmap;
    CPalette* pPalette = ptp->pPalette;
    CCriticalSection* pCriticalSection = ptp->pCriticalSection;
    CEvent* pKillEvent = ptp->pEvent;
    delete ptp;

    DIBSECTION ds;
    pBitmap->GetObject (sizeof (DIBSECTION), &ds);
    int nWidth = ds.dsBm.bmWidth;
    int nHeight = ds.dsBm.bmHeight;
    //
    // Initialize one memory DC (memDC2) to hold a color copy of the
    // image and another memory DC (memDC1) to hold a gray scale copy.
    //
    CClientDC dc (pWnd);
    CBitmap bitmap1, bitmap2;
    bitmap1.CreateCompatibleBitmap (&dc, nWidth, nHeight);
    bitmap2.CreateCompatibleBitmap (&dc, nWidth, nHeight);

    CDC memDC1, memDC2;
    memDC1.CreateCompatibleDC (&dc);
    memDC2.CreateCompatibleDC (&dc);
    CBitmap* pOldBitmap1 = memDC1.SelectObject (&bitmap1);
    CBitmap* pOldBitmap2 = memDC2.SelectObject (&bitmap2);

    CPalette* pOldPalette1 = NULL;
    CPalette* pOldPalette2 = NULL;
    CPalette grayPalette;

    if (pPalette->m_hObject != NULL) {
        LOGPALETTE* pLP = CreateGrayScale ();
        grayPalette.CreatePalette (pLP);
```

```
        delete[] pLP;

        pOldPalette1 = memDC1.SelectPalette (&grayPalette, FALSE);
        pOldPalette2 = memDC2.SelectPalette (pPalette, FALSE);
        memDC1.RealizePalette ();
        memDC2.RealizePalette ();
    }
    //
    // Copy the bitmap to memDC2.
    //
    CDC memDC3;
    memDC3.CreateCompatibleDC (&dc);
    pCriticalSection->Lock ();
    CBitmap* pOldBitmap3 = memDC3.SelectObject (pBitmap);
    memDC2.BitBlt (0, 0, nWidth, nHeight, &memDC3, 0, 0, SRCCOPY);
    memDC3.SelectObject (pOldBitmap3);
    pCriticalSection->Unlock ();
    //
    // Convert the colors in memDC2 to shades of gray in memDC1.
    //
    int x, y;
    COLORREF crColor;
    BYTE grayLevel;

    for (y=0; y<nHeight; y++) {
        for (x=0; x<nWidth; x++) {
            crColor = memDC2.GetPixel (x, y);
            grayLevel = (BYTE)
                (((((UINT) GetRValue (crColor)) * 30) +
                (((UINT) GetGValue (crColor)) * 59) +
                (((UINT) GetBValue (crColor)) * 11)) / 100);
            memDC1.SetPixel (x, y,
                PALETTERGB (grayLevel, grayLevel, grayLevel));
        }
        //
        // Kill the thread if the pKillEvent event is signaled.
        //
        if (::WaitForSingleObject (pKillEvent->m_hObject, 0) ==
            WAIT_OBJECT_0) {
            memDC1.SelectObject (pOldBitmap1);
            memDC2.SelectObject (pOldBitmap2);

            if (pPalette->m_hObject != NULL) {
                memDC1.SelectPalette (pOldPalette1, FALSE);
                memDC2.SelectPalette (pOldPalette2, FALSE);
            }
            pWnd->PostMessage (WM_USER_THREAD_ABORTED, y + 1, 0);
            return (UINT) -1;
        }
        pWnd->SendMessage (WM_USER_THREAD_UPDATE, y + 1, nHeight);
    }
    //
    // Copy the gray scale image over the original bitmap.
    //
    CPalette* pOldPalette3 = NULL;
```

```
    if (pPalette->m_hObject != NULL) {
        pOldPalette3 = memDC3.SelectPalette (&grayPalette, FALSE);
        memDC3.RealizePalette ();
    }
    pCriticalSection->Lock ();
    pOldBitmap3 = memDC3.SelectObject (pBitmap);
    memDC3.BitBlt (0, 0, nWidth, nHeight, &memDC1, 0, 0, SRCCOPY);
    memDC3.SelectObject (pOldBitmap3);
    pCriticalSection->Unlock ();
    //
    // Clean up the memory DCs.
    //
    memDC1.SelectObject (pOldBitmap1);
    memDC2.SelectObject (pOldBitmap2);

    if (pPalette->m_hObject != NULL) {
        memDC1.SelectPalette (pOldPalette1, FALSE);
        memDC2.SelectPalette (pOldPalette2, FALSE);
        memDC3.SelectPalette (pOldPalette3, FALSE);
    }
    //
    // Tell the frame window we're done.
    //
    pWnd->PostMessage (WM_USER_THREAD_FINISHED, 0, 0);
    return 0;
}
LOGPALETTE* CreateGrayScale ()
{
    UINT nSize = sizeof (LOGPALETTE) + (sizeof (PALETTEENTRY) * 63);
    LOGPALETTE* pLP = (LOGPALETTE*) new BYTE[nSize];

    pLP->palVersion = 0x300;
    pLP->palNumEntries = 64;

    for (int i=0; i<64; i++) {
        pLP->palPalEntry[i].peRed = i * 4;
        pLP->palPalEntry[i].peGreen = i * 4;
        pLP->palPalEntry[i].peBlue = i * 4;
        pLP->palPalEntry[i].peFlags = 0;
    }
    return pLP;
}
```

ImageEditView.h

```
// ImageEditView.h : interface of the CImageEditView class
//
/////////////////////////////////////////////////////////////////////////////
#if !defined(
    AFX_IMAGEEDITVIEW_H__9D77AEEC_AA14_11D2_8E53_006008A82731__INCLUDED_)
#define
    AFX_IMAGEEDITVIEW_H__9D77AEEC_AA14_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CImageEditView : public CScrollView
```

```
{
protected: // create from serialization only
    CImageEditView();
    DECLARE_DYNCREATE(CImageEditView)

// Attributes
public:
    CImageEditDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CImageEditView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CImageEditView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
    //{{AFX_MSG(CImageEditView)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifndef _DEBUG  // debug version in ImageEditView.cpp
inline CImageEditDoc* CImageEditView::GetDocument()
   { return (CImageEditDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//    AFX_IMAGEEDITVIEW_H__9D77AEEC_AA14_11D2_8E53_006008A82731__INCLUDED_)
```

ImageEditView.cpp

```
// ImageEditView.cpp : implementation of the CImageEditView class
//
#include "stdafx.h"
#include "ImageEdit.h"
#include "ImageEditDoc.h"
#include "ImageEditView.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CImageEditView
IMPLEMENT_DYNCREATE(CImageEditView, CScrollView)
BEGIN_MESSAGE_MAP(CImageEditView, CScrollView)
    //{{AFX_MSG_MAP(CImageEditView)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CImageEditView construction/destruction
CImageEditView::CImageEditView()
{
}
CImageEditView::~CImageEditView()
{
}
BOOL CImageEditView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CImageEditView drawing
void CImageEditView::OnDraw(CDC* pDC)
{
    CImageEditDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    CBitmap* pBitmap = pDoc->GetBitmap ();

    if (pBitmap != NULL) {
        CPalette* pOldPalette;
        CPalette* pPalette = pDoc->GetPalette ();

        if (pPalette != NULL) {
            pOldPalette = pDC->SelectPalette (pPalette, FALSE);
            pDC->RealizePalette ();
       }

        DIBSECTION ds;
        pBitmap->GetObject (sizeof (DIBSECTION), &ds);

        CDC memDC;
        memDC.CreateCompatibleDC (pDC);
        CBitmap* pOldBitmap = memDC.SelectObject (pBitmap);

        pDC->BitBlt (0, 0, ds.dsBm.bmWidth, ds.dsBm.bmHeight, &memDC,
            0, 0, SRCCOPY);

        memDC.SelectObject (pOldBitmap);

        if (pPalette != NULL)
```

```
            pDC->SelectPalette (pOldPalette, FALSE);
    }
}
void CImageEditView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate ();

    CString string;
    CSize sizeTotal;
    CBitmap* pBitmap = GetDocument ()->GetBitmap ();
    //
    // If a bitmap is loaded, set the view size equal to the bitmap size.
    // Otherwise, set the view's width and height to 0.
    //
    if (pBitmap != NULL) {
        DIBSECTION ds;
        pBitmap->GetObject (sizeof (DIBSECTION), &ds);
        sizeTotal.cx = ds.dsBm.bmWidth;
        sizeTotal.cy = ds.dsBm.bmHeight;
        string.Format (_T ("\t%d x %d, %d bpp"), ds.dsBm.bmWidth,
            ds.dsBm.bmHeight, ds.dsBmih.biBitCount);
    }
    else {
        sizeTotal.cx = sizeTotal.cy = 0;
        string.Empty ();
    }

    AfxGetMainWnd ()->SendMessage (WM_USER_UPDATE_STATS, 0,
        (LPARAM) (LPCTSTR) string);
    SetScrollSizes (MM_TEXT, sizeTotal);
}
/////////////////////////////////////////////////////////////////////////////
// CImageEditView diagnostics
#ifdef _DEBUG
void CImageEditView::AssertValid() const
{
    CScrollView::AssertValid();
}
void CImageEditView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}
CImageEditDoc* CImageEditView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CImageEditDoc)));
    return (CImageEditDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CImageEditView message handlers
```

SpecialStatusBar.h

```
// SpecialStatusBar.h: interface for the CSpecialStatusBar class.
//
//////////////////////////////////////////////////////////////////////
#if !defined(
```

```
   AFX_SPECIALSTATUSBAR_H__4BA7D301_AA24_11D2_8E53_006008A82731__INCLUDED_)
#define
   AFX_SPECIALSTATUSBAR_H__4BA7D301_AA24_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CSpecialStatusBar : public CStatusBar
{
public:
    void SetProgress (int nPercent);
    void SetImageStats(LPCTSTR pszStats);
    CSpecialStatusBar();
    virtual ~CSpecialStatusBar();

protected:
    CProgressCtrl m_wndProgress;
    afx_msg int OnCreate (LPCREATESTRUCT lpcs);
    afx_msg void OnSize (UINT nType, int cx, int cy);
    DECLARE_MESSAGE_MAP ()
};
#endif
// !defined(
// AFX_SPECIALSTATUSBAR_H__4BA7D301_AA24_11D2_8E53_006008A82731__INCLUDED_)
```

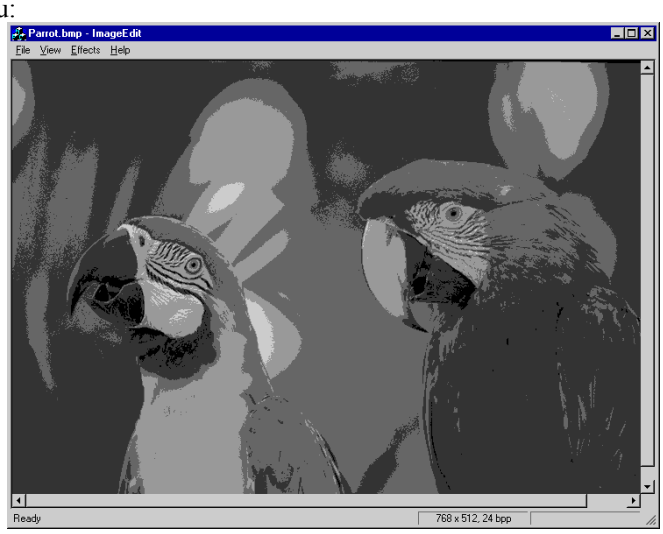
SpecialStatusBar.cpp

```
// SpecialStatusBar.cpp: implementation of the CSpecialStatusBar class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "ImageEdit.h"
#include "SpecialStatusBar.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

BEGIN_MESSAGE_MAP(CSpecialStatusBar, CStatusBar)
    ON_WM_CREATE ()
    ON_WM_SIZE ()
END_MESSAGE_MAP()
CSpecialStatusBar::CSpecialStatusBar()
{
}
CSpecialStatusBar::~CSpecialStatusBar()
{
}
int CSpecialStatusBar::OnCreate (LPCREATESTRUCT lpcs)
{
    static UINT nIndicators[] =
    {
        ID_SEPARATOR,
```

```
        ID_SEPARATOR,
        ID_SEPARATOR
    };
    if (CStatusBar::OnCreate (lpcs) == -1)
        return -1;
    //
    // Add panes to the status bar.
    //
    SetIndicators (nIndicators, sizeof (nIndicators) / sizeof (UINT));
    //
    // Size the status bar panes.
    //
    TEXTMETRIC tm;
    CClientDC dc (this);
    CFont* pFont = GetFont ();

    CFont* pOldFont = dc.SelectObject (pFont);
    dc.GetTextMetrics (&tm);
    dc.SelectObject (pOldFont);

    int cxWidth;
    UINT nID, nStyle;
    GetPaneInfo (1, nID, nStyle, cxWidth);
    SetPaneInfo (1, nID, nStyle, tm.tmAveCharWidth * 24);
    GetPaneInfo (2, nID, nStyle, cxWidth);
    SetPaneInfo (2, nID, SBPS_NOBORDERS, tm.tmAveCharWidth * 24);
    //
    // Place a progress control in the rightmost pane.
    //
    CRect rect;
    GetItemRect (2, &rect);
    m_wndProgress.Create (WS_CHILD œ WS_VISIBLE œ PBS_SMOOTH,
        rect, this, -1);
    m_wndProgress.SetRange (0, 100);
    m_wndProgress.SetPos (0);
    return 0;
}
void CSpecialStatusBar::OnSize (UINT nType, int cx, int cy)
{
    CStatusBar::OnSize (nType, cx, cy);
    //
    // Resize the rightmost pane to fit the resized status bar.
    //
    CRect rect;
    GetItemRect (2, &rect);
    m_wndProgress.SetWindowPos (NULL, rect.left, rect.top,
        rect.Width (), rect.Height (), SWP_NOZORDER);
}
void CSpecialStatusBar::SetImageStats(LPCTSTR pszStats)
{
    SetPaneText (1, pszStats, TRUE);
}
void CSpecialStatusBar::SetProgress(int nPercent)
{
    ASSERT (nPercent >= 0 && nPercent <= 100);
```

```
    m_wndProgress.SetPos (nPercent);
}
```

Màn hình kết quả như sau:



CHƯƠNG 4. LẬP TRÌNH CƠ SỞ DỮ LIỆU VỚI MFC

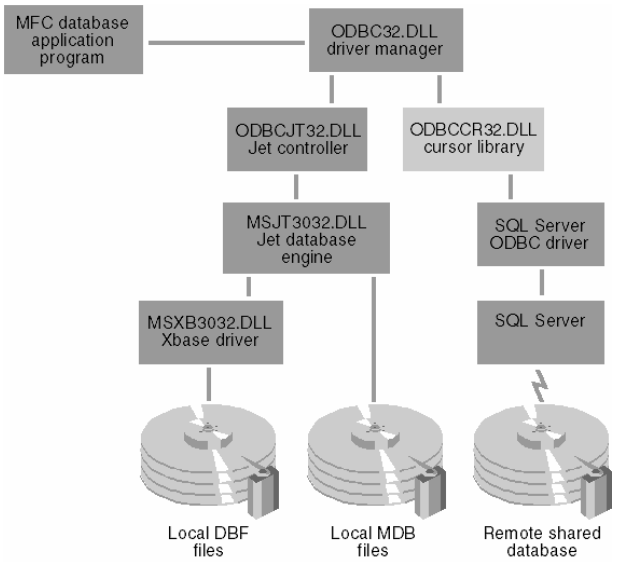
4.1 TRUY XUẤT VỚI ODBC/DAO

4.1.1 Vấn đề quan tâm

- Hiểu và sử dụng được các class về truy xuất dữ liệu mà MFC hỗ trợ.
- Hiểu và lập trình được các sự kiện tương tác dữ liệu.

4.1.2 Giới thiệu

Cấu trúc ODBC được mô tả như sau:



Tổ chức các class giúp truy xuất dữ liệu bao gồm:

Class	Use
CDaoWorkspace	Giao diện quản lý các đơn kết nối.
CDaoDatabase	Giao diện làm việc với database
CDaoRecordset	Giao diện làm việc với tập các mẩu tin (kiểu table, dynaset, snapshot)
CDaoTableDef	Giao diện thao tác định nghĩa hay truy xuất bảng dữ liệu.
CDaoQueryDef	Giao diện truy vấn dữ liệu

Trong quá trình truy xuất dữ liệu, dữ liệu lấy từ cơ sở dữ liệu về thường được lưu trữ và quản lý bởi class CRecordset. Thành phần của class CRecordset như sau:

Tên hàm	Mô tả
Open	Mở recordset
AddNew	Chuẩn bị để thêm mới mẫu dữ liệu vào bảng dữ liệu
Update	Hoàn tất thao tác AddNew hay Edit bởi thực hiện lưu vào cơ sở dữ liệu.
Delete	Xoá mẫu dữ liệu hiện hành
Edit	Chuẩn bị để thay đổi mẫu dữ liệu vào bảng dữ liệu
IsBOF	Nhận biết vị trí đứng đầu trong tập dữ liệu
IsEOF	Nhận biết vị trí đứng cuối trong tập dữ liệu
MoveNext	Di chuyển sang mẫu dữ liệu kế tiếp
MoveFirst	Di chuyển sang mẫu dữ liệu đầu tiên
MoveLast	Di chuyển sang mẫu dữ liệu cuối cùng
MovePrev	Di chuyển sang mẫu dữ liệu liền trước
GetDefaultConnect	Lấy về chuỗi kết nối cơ sở dữ liệu mặc định
GetDefaultSQL	Lấy về chuỗi truy vấn cơ sở dữ liệu mặc định
DoFieldExchange	Thực hiện trao đổi dữ liệu giữa thành phần dữ liệu trong tập dữ liệu với biến liên kết tương ứng
GetStatus	Lấy về chỉ mục của mẫu dữ liệu hiện hành và trạng thấy của nó
GetRecordCount	Lấy về số lượng mẫu dữ liệu trong tập dữ liệu
GetODBCFieldCount	Lấy về số lượng thành phần dữ liệu trong tập dữ liệu
GetODBCFieldInfo	Lấy về thông tin thành phần dữ liệu trong tập dữ liệu

Việc nhận biết trạng thái và số lượng "mẫu tin" (row/record) trong quá trình truy vấn dữ liệu từ đối tượng CRecordset được thực hiện bởi hàm **GetRecordCount**() và **GetStatus**()

Việc di chuyển giữa các record bởi hàm **MoveNext**(), **MovePrevious**(), **MoveFirst**(), **MoveLast**().

Để xử lý bẫy lỗi trong quá trình truy xuất, dùng cơ chế **try…catch** theo ví dụ sau:

Ví dụ 1:

```
try {
  m_pSet->Delete();        // thao tác cập nhật - xoá dữ liệu
}
catch(CDBException* e) {
  AfxMessageBox(e->m_strError);
  e->Delete();
  m_pSet->MoveFirst(); // lost our place!
  UpdateData(FALSE);
  return;
}
m_pSet->MoveNext();
```

Để duyệt tập hợp dữ liệu trong đối tượng CRecordset, dùng vòng lặp theo ví dụ sau:

Ví dụ 2:

Tạo 1 class CMySet như sau:

```
class CMySet: public CRecordset
{
       int          MyID;
       CString      MyName;
…
       CString GetDefaultConnect() {return _T("ODBC;DSN=MyDabase");};
       CString GetDefaultSQL() { return _T("[MyTable]");};
…
}
```

và truy cập dữ liệu này như sau:

- Mở recordset:

```
CMySet* m_pSet;
…
if(m_pSet->IsOpen()) {
   m_pSet->Close();
 }
m_pSet->Open();
```

- Kiểm tra recordset có chứa dữ liệu không

```
if(m_pSet->IsBOF())       // detects empty recordset
{
        return;
}
```

- Duyệt qua các mẫu dữ liệu trong recordset:

```
m_pSet->MoveFirst();  // fails if recordset is empty
while(!m_pSet->IsEOF())
{
        str.Format("%ld", m_pSet->m_MyID);
        pDC->TextOut(10, 20, str);
        pDC->TextOut(10, 50, m_pSet->m_MyName);
```

```
        m_pSet->MoveNext();
}
m_pSet->Close();
```

- Chuyển sang chế độ thêm mới một mẫu tin:

```
if (AfxMessageBox("Are you  want to add new record?",1,1) = IFYES ){
        m_pSet->AddNew();
}
```

- Chuyển sang chế độ cập nhật một mẫu tin:

```
if (AfxMessageBox("Are you  want to edit this record?",1,1) = IFYES ){
        m_pSet->Edit();
}
```

- Thực hiện lưu thông tin vào cơ sở dữ liệu

```
m_pSet->Update();
```

- Thực hiện xoá 1 mẫu tin

```
if (AfxMessageBox("Are you  want to delete this record?",1,1) = IFYES ){
        m_pSet->Delete();
}
```

Chú ý:

Để sử dụng các lớp truy xuất dữ liệu, cần thêm vào hàng cuối cùng trong file **StdAfx.h** hàng khai báo như sau:

```
#include <afxdb.h>
```

Trong file *.RC, thêm hàng khai báo

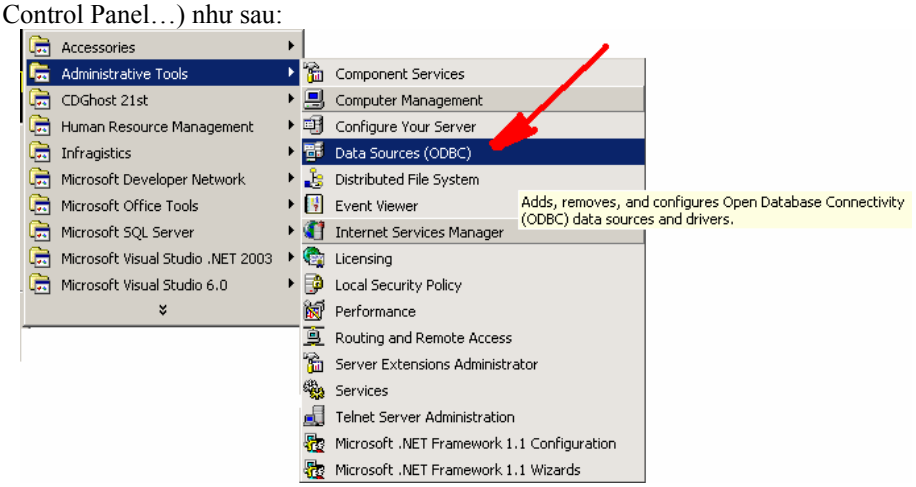
```
"#include ""afxdb.rc""  // database resources\r\n"
```

sau hàng lệnh
"#include ""afxprint.rc"" // printing print preview resources\r\n"
thêm hàng khai báo

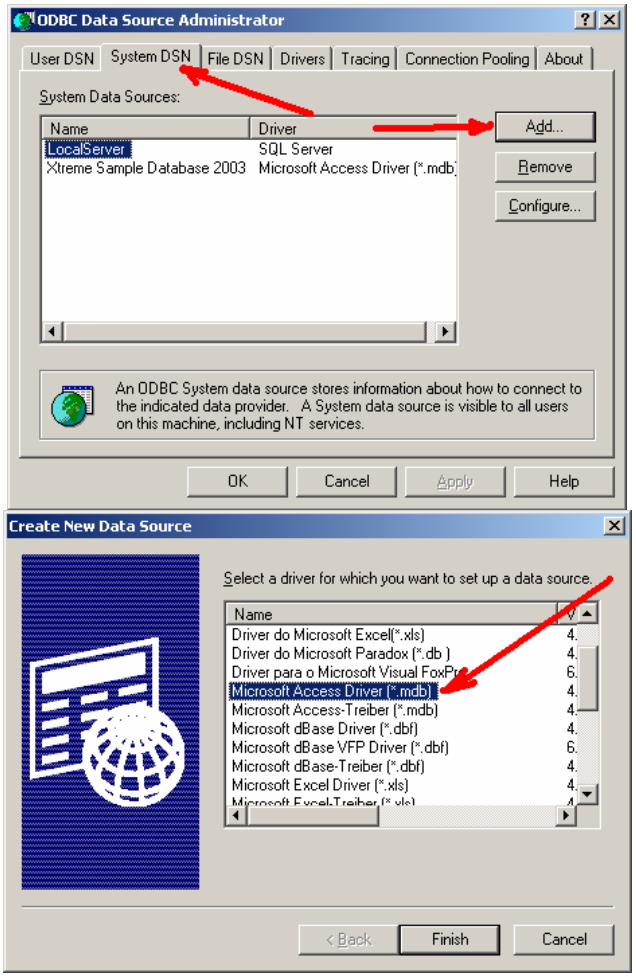
```
#include "afxdb.rc"    // database resources
```

sau hàng lệnh
#include "afxprint.rc" // printing print preview resources

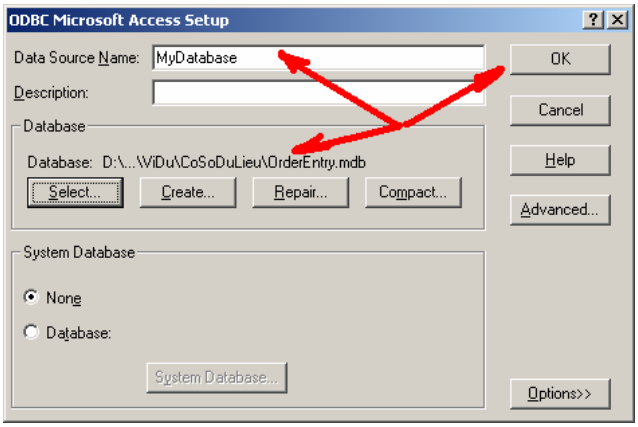
- Để khai báo nguồn dữ liệu, cần sử dụng tiện ích DataSource(ODBC) có trong chương trình Windows (tại Control Panel…) như sau:



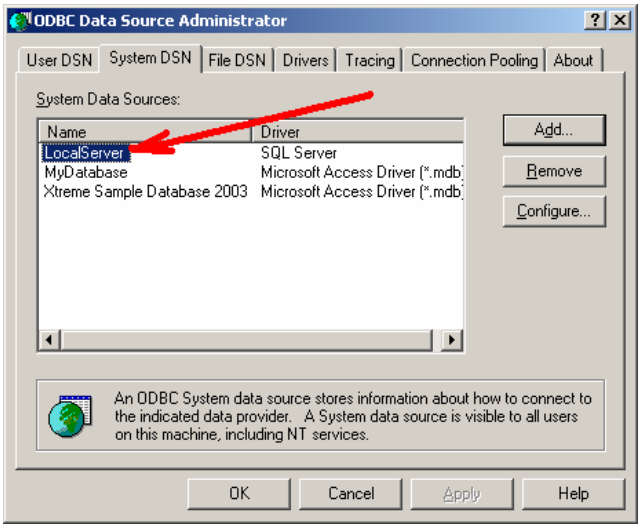
và quá trình khai báo tên nguồn dữ liệu được tiến hành trong ứng dụng này:


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
System Data Sources:
Name
Driver
LocalServer
SQL Server
Xtreme Sample Database 2003
Microsoft Access Driver (*.mdb)
Add...
Remove
Configure...
An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.
OK
Cancel
Apply
Help
Create New Data Source
Select a driver for which you want to set up a data source.
Name
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
Microsoft Access Driver (*.mdb)
Microsoft Access-Treiber (*.mdb)
Microsoft dBase Driver (*.dbf)
Microsoft dBase VFP Driver (*.dbf)
Microsoft dBase-Treiber (*.dbf)
Microsoft Excel Driver (*.xls)
< Back
Finish
Cancel

và:


ODBC Microsoft Access Setup
Data Source Name:
MyDatabase
Description:
Database
Database: D:\...\ViDu\CoSoDuLieu\OrderEntry.mdb
Select...
Create...
Repair...
Compact...
System Database
None
Database:
System Database...
OK
Cancel
Help
Advanced...
Options>>

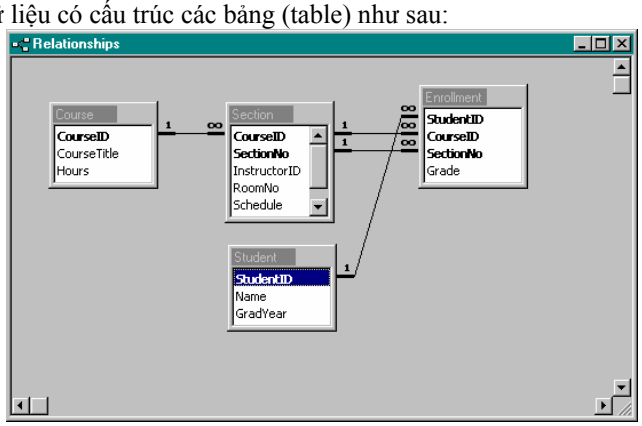
và:



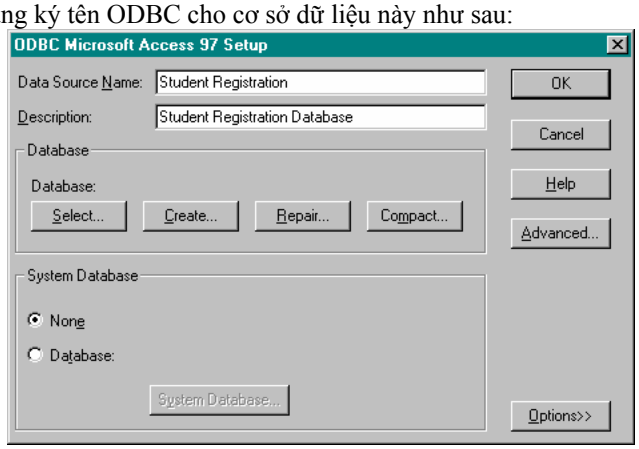
4.1.3 Ví dụ tổng hợp

4.1.3.1 Chương trình 1:

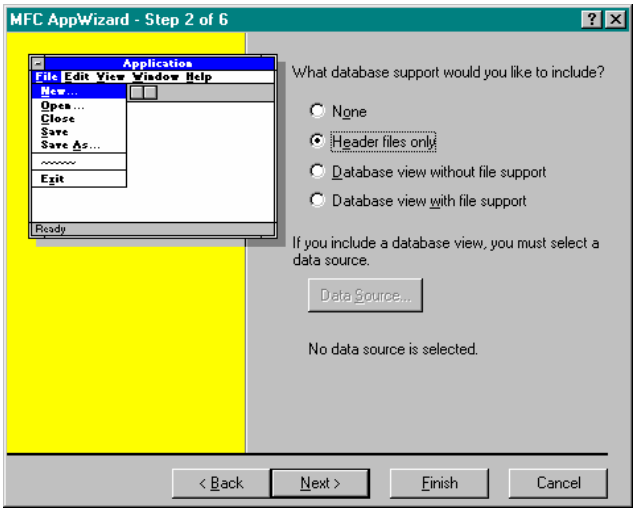
- B1: Cho cơ sở dữ liệu có cấu trúc các bảng (table) như sau:



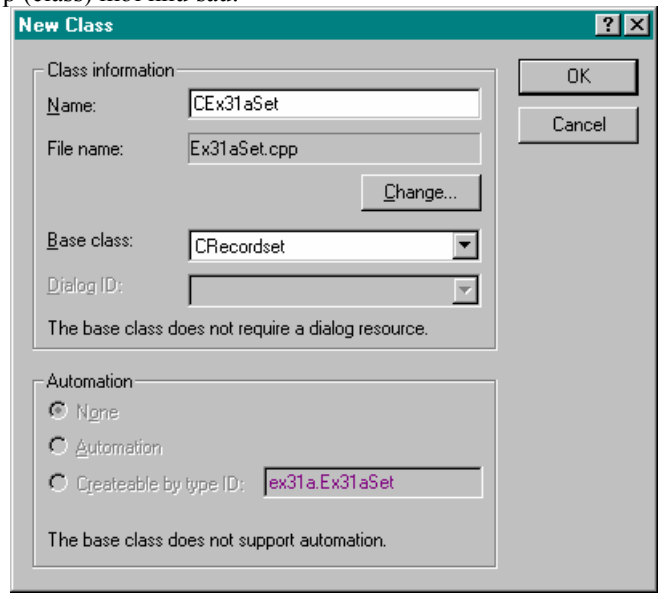
- B2: Thực hiện đăng ký tên ODBC cho cơ sở dữ liệu này như sau:



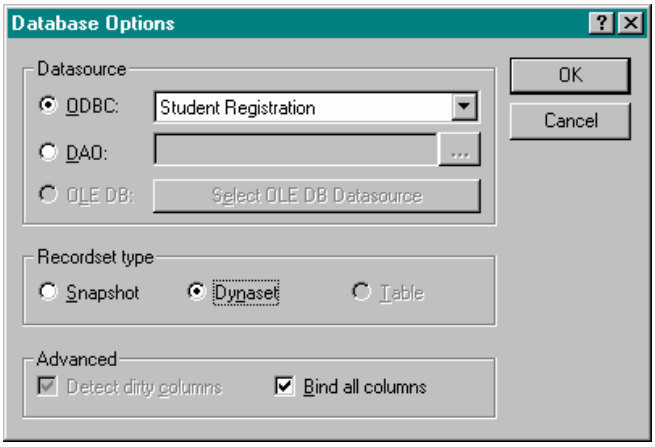
- B3: Tạo ứng dụng trong môi trường Visual C++ như sau:



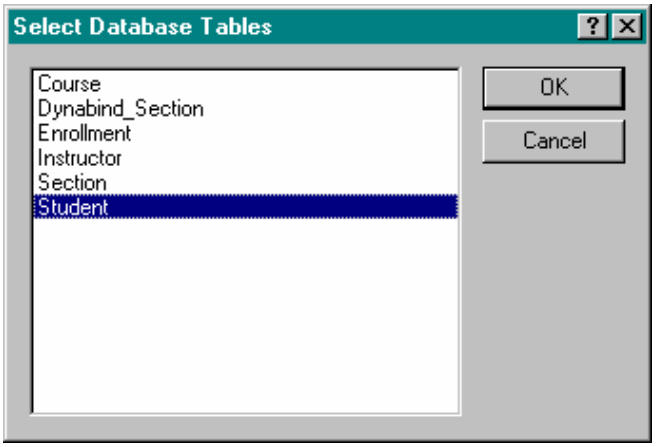
- B4: Tạo thêm lớp (class) mới như sau:



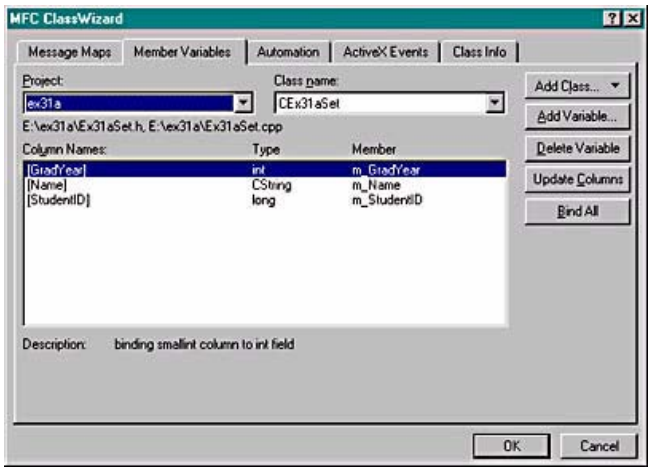
và:



và:



- B5: Tạo các biến liên kết như sau:



- B6: Cập nhật chương trình như sau:

Trong lớp (class) CEx31aDoc của file ex31aDoc.h, thêm:

```
CEx31aSet m_ex31aSet;
```

Trong file ex31aDoc.cpp, thêm:

```
#include "ex31aSet.h"
```

Trong lớp (class) CEx31aView của file ex31aView.h, thêm:

```
CEx31aSet* m_pSet;
```

Cập nhật lại các hàm trong file ex31aView.cpp như sau:

```
void CEx31aView::OnDraw(CDC* pDC)
{
    TEXTMETRIC tm;
    pDC->GetTextMetrics(&tm);
    int nLineHeight=tm.tmHeight+tm.tmExternalLeading;
    CPoint pText(0,0);

    int y = 0;
    CString str;
    if (m_pSet->IsBOF()) { // detects empty recordset
        return;
    }
    m_pSet->MoveFirst();   // fails if recordset is empty
    while (!m_pSet->IsEOF()) {
        str.Format("%ld", m_pSet->m_StudentID);
        pDC->TextOut(pText.x, pText.y, str);
        pDC->TextOut(pText.x+1000, pText.y, m_pSet->m_Name);
        str.Format("%d", m_pSet->m_GradYear);
```

```
        pDC->TextOut(pText.x+4000, pText.y, str);
        m_pSet->MoveNext();
        pText.y -= nLineHeight;
    }
}
void CEx31aView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
    CSize sizeTotal(8000, 10500);
    SetScrollSizes(MM_HIENGLISH, sizeTotal);

    m_pSet = &GetDocument()->m_ex31aSet;
    // Remember that documents/views are reused in SDI applications!
    if (m_pSet->IsOpen()) {
        m_pSet->Close();
    }
    m_pSet->Open();
}
```

Trong file ex31aView.cpp, thêm:

```
#include "ex31aSet.h"
```

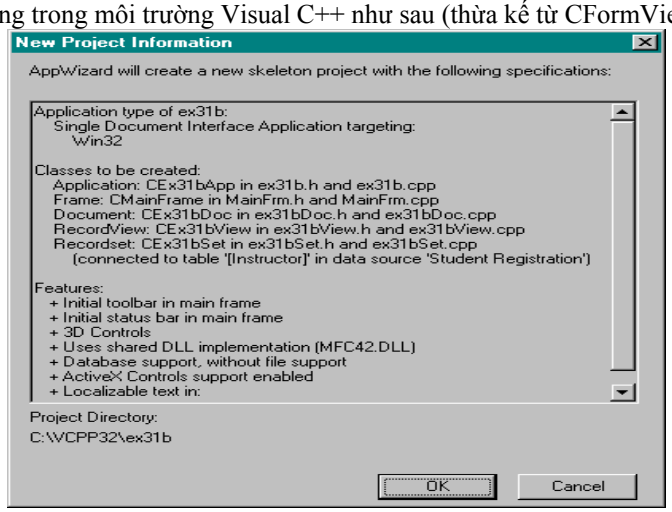
Trong file ex31a.cpp, thêm:

```
#include "ex31aSet.h"
```

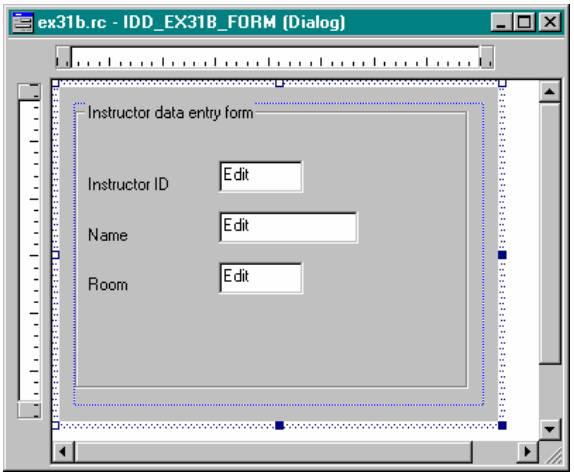
- B7: Biên dịch và thực thi chương trình, kết quả như sau:

4.1.3.2 Chương trình 2:

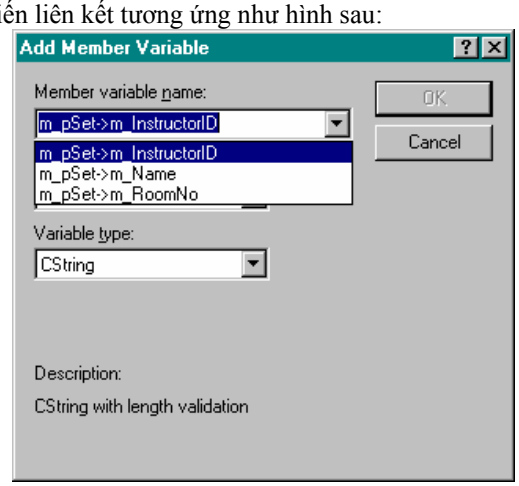
- B1: Tạo ứng dụng trong môi trường Visual C++ như sau (thừa kế từ CFormView):



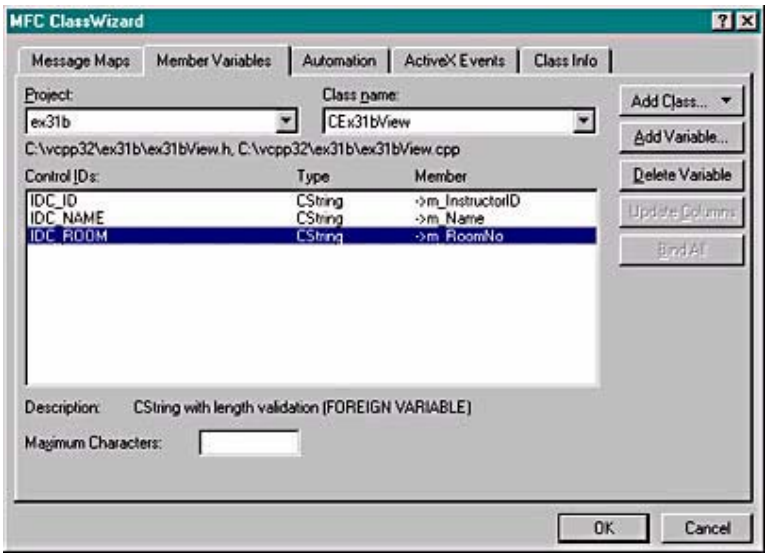
- B2: Tạo giao diện chương trình như sau (với các edit box có tên lần lượt là IDC_NAME, and IDC_ROOM):



- B3: Lần lượt tạo các biến liên kết tương ứng như hình sau:



và:



- B4: Biên dịch và thực thi chương trình
- B5: Tạo các hàm liên kết các icon (next, back, first, last) có trên toolbar nhuư sau:

Menu Command	Command ID	Command Handler	Update Command UI Handler
Add Record	ID_RECORD_ADD	OnRecordAdd	
Clear Fields	ID_RECORD_CLEARFIELDS	OnRecordClearfields	
Delete	ID_RECORD_DELETE	OnRecordDelete	OnUpdateRecordDelete

Record			
Update Record	ID_RECORD_UPDATE	OnRecordUpdate	OnUpdateRecordUpdate

- B6: Cập nhật hàm OnMove trong lớp CEx31bView như sau:

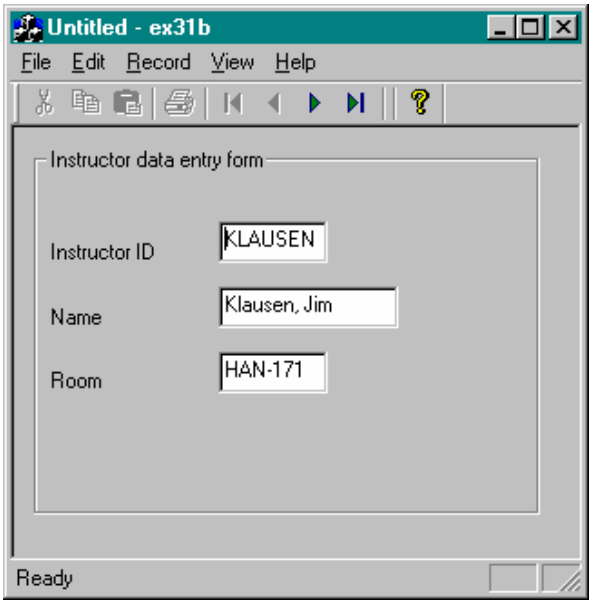
```
BOOL CEx31bView::OnMove(UINT nIDMoveCommand)
{
    switch (nIDMoveCommand)
    {
    case ID_RECORD_PREV:
        m_pSet->MovePrev();
        if (!m_pSet->IsBOF())
            break;

    case ID_RECORD_FIRST:
        m_pSet->MoveFirst();
        break;

    case ID_RECORD_NEXT:
        m_pSet->MoveNext();
        if (!m_pSet->IsEOF())
            break;
        if (!m_pSet->CanScroll()) {
            // Clear screen since we're sitting on EOF
            m_pSet->SetFieldNull(NULL);
            break;
        }

    case ID_RECORD_LAST:
        m_pSet->MoveLast();
        break;

    default:
        // unexpected case value
        ASSERT(FALSE);
    }

    // Show results of Move operation
    UpdateData(FALSE);
    return TRUE;
}
```

và:

```
void CEx31bView::OnRecordAdd()
{
    m_pSet->AddNew();
    UpdateData(TRUE);
    if (m_pSet->CanUpdate()) {
        m_pSet->Update();
    }
    if (!m_pSet->IsEOF()) {
        m_pSet->MoveLast();
    }
    m_pSet->Requery(); // for sorted sets
    UpdateData(FALSE);
}
```

```
void CEx31bView::OnRecordClearfields()
{
    m_pSet->SetFieldNull(NULL);
    UpdateData(FALSE);
}

void CEx31bView::OnRecordDelete()
{
    CRecordsetStatus status;
    try {
        m_pSet->Delete();
    }
    catch(CDBException* e) {
        AfxMessageBox(e->m_strError);
        e->Delete();
        m_pSet->MoveFirst(); // lost our place!
        UpdateData(FALSE);
        return;
    }
    m_pSet->GetStatus(status);
    if (status.m_lCurrentRecord == 0) {
        // We deleted last of 2 records
        m_pSet->MoveFirst();
    }
    else {
        m_pSet->MoveNext();
    }
    UpdateData(FALSE);
}

void CEx31bView::OnUpdateRecordDelete(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(!m_pSet->IsEOF());
}

void CEx31bView::OnRecordUpdate()
{
    m_pSet->Edit();
    UpdateData(TRUE);
    if (m_pSet->CanUpdate()) {
        m_pSet->Update();
    }
// should requery if key field changed
}

void CEx31bView::OnUpdateRecordUpdate(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(!m_pSet->IsEOF());
}
```

- B7: Biên dịch và thực thi chương trình, kết quả như sau:



4.2 TRUY XUẤT VỚI OLEDB

4.2.1 Vấn đề quan tâm

- Hiểu về bản chất của ADO và sử dụng được các class về truy xuất dữ liệu mà MFC hỗ trợ.

4.2.2 Giới thiệu OLEDB

OLEDB là một tập hợp các giao diện truy xuất dữ liệu thông qua COM.

Cấu trúc của OLEDB bao gồm:

- *Enumerators*: có tác vụ tìm các nguồn dữ liệu khả dụng.
- *Data source objects*: Data source objects chứa các cơ chế kết nối tới nguồn dữ liệu - một phiên làm việc (*session*) được tạo ra khi chương trình kết nối đến một nguồn dữ liệu.
- *Sessions*: Sessions thể hiện một kết nối(phiên làm việc) đến nguồn dữ liệu – có thể nguồn dữ liệu có thể tạo nhiều sessions. Mỗi Sessions có thể tạo ra transactions, commands, và rowsets.
- *Transaction*: là đối tượng quản lý các thao tác truy xuất dữ liệu và bảo đảm an toàn dữ liệu.
- *Commands*: là đối tượng cho phép thực thi các lệnh SQL. Nếu lệnh SQL là lệnh SELECT, thì đối tượng này sẽ tạo ra (nhận về) rowsets. Một seesion có thể tạo sử dụng với commands.
- *Rowsets*: là tập dữ liệu dạng bảng (tabular). Rowsets có thể được tạo từ session hay command.
- *Errors*: Errors có thể được tạo ra bởi bất kỳ giao diện của đối tượng OLE DB nào. Errors chứa các thông tin về lỗi.

4.2.3 Thực hiện tác vụ truy xuất dữ liệu với OLEDB:

MFC cung cấp một số class giúp thao tác với OLEDB như:

Class	Use
CDataSource	This class represents the data source component and manages the connection to a data source.
CEnumerator	This class provides a way to select a provider by cycling through a list of providers. Its functionality is equivalent to the SQLBrowseConnect and SQLDriverConnect functions.
CSession	This class handles transactions. You can use this class to create rowsets, commands, and many other objects. A CDataSource object creates a CSession object using the CSession::Open method
CAccessor	This class is used when a record is statically bound to a data source—it contains the pre-existing data buffer and understands the data format up front. CAccessor is used when you know the structure and the type of the database ahead of time.
CDynamicAccessor	This class is used for retrieving data from a source whose structure is not known at design time. This class uses

	IColumnsInfo::GetColumnInfo to get the database column information. CDynamicAccessor creates and manages the data buffer.
CDynamicParameterAccessor	This class is similar to CDynamicAccessor except that it's used with commands. When used to prepare commands, CDynamicParameterAccessor can get parameter information from the ICommandWithParameters interface, which is especially useful for handling unknown command types.
CManualAccessor	This class lets you access whatever data types you want as long as the provider can convert the type. CManualAccessor handles both result columns and command parameters.
CTable	The CTable class is a minimal class implementation that opens a table on a data source(which you can specify programmatically). Use this class when you need bare-bones access to a source, since CTable is designed for simple providers that do not support commands
CCommand	CCommand is used mostly for executing commands. This class has a function named Open that executes singular commands. This class also has a function named Prepare for setting up a command to execute multiple times

Một số giao diện được cung cấp để tương tác với OLEDB như:

Interface	Required?	Implemented?
IDBInitialize	Mandatory	Yes
IDBCreateSession	Mandatory	Yes
IDBProperties	Mandatory	Yes
IPersist	Mandatory	Yes
IDBDataSourceAdmin	Optional	No
IDBInfo	Optional	No
IPersistFile	Optional	No
ISupportErrorInfo	Optional	No
IDBInitialize	Mandatory	Yes
IDBCreateSession	Mandatory	Yes
IDBProperties	Mandatory	Yes
IPersist	Mandatory	Yes
IDBDataSourceAdmin	Optional	No
IDBInfo	Optional	No
IPersistFile	Optional	No
ISupportErrorInfo	Optional	No
IGetDataSource	Mandatory	Yes
IOpenRowset	Mandatory	Yes
ISessionProperties	Mandatory	Yes
IDBCreateCommand	Optional	Yes
IDBSchemaRowset	Optional	Yes
IIndexDefinition	Optional	No
ISupportErrorInfo	Optional	No
ITableDefinition	Optional	No
ITransactionJoin	Optional	No
ITransactionLocal	Optional	No
ITransactionObject	Optional	No
IAccessor	Mandatory	Yes
IColumnsInfo	Mandatory	Yes
IConvertType	Mandatory	Yes
IRowset	Mandatory	Yes
IRowsetInfo	Mandatory	Yes
IColumnsRowset	Optional	No
IConnectionPointContainer	Optional	Yes, through ATL
IRowsetChange	Optional	No
IRowsetIdentity	Required for Level 0	Yes
IRowsetLocate	Optional	No

IRowsetResynch	Optional	No
IRowsetScroll	Optional	No
IRowsetUpdate	Optional	No
ISupportErrorInfo	Optional	No

Ví dụ 1:

```
class CMainFrame : public CFrameWnd
{
…
    CDataSource m_db;
    bool m_bConnectionValid;
    CSession m_session;
…
}
…

BOOL CMainFrame::OpenConnection()
{
    HRESULT hr = m_db.Open();

    if(SUCCEEDED(hr))
        m_bConnectionValid = true;

    if(SUCCEEDED(hr) && m_session.Open(m_db) != S_OK){
        AfxMessageBox("Could not open a Session to the database");
        return FALSE;
    }

    return TRUE;
}
```

Ví dụ 2:

```
CCommand<CDynamicAccessor, CRowset> dbCommand;
try {
    Recordset20Ptr spRs;
    ADORecordsetConstructionPtr spADOsCt;

    CDBPropSet propset(DBPROPSET_ROWSET);
    propset.AddProperty(DBPROP_CLIENTCURSOR, true);
    propset.AddProperty(DBPROP_IRowsetChange, true);
    propset.AddProperty(DBPROP_UPDATABILITY, DBPROPVAL_UP_CHANGE |
        DBPROPVAL_UP_INSERT | DBPROPVAL_UP_DELETE);

    CString sCommand;
    sCommand.Format("SELECT * FROM [%s]", sTableName);

    HRESULT hr = dbCommand.Create(
        pMainFrame->m_session,(LPCTSTR)sCommand);
    if(FAILED(hr))
        _com_issue_error(hr);

    hr = dbCommand.Open(&propset, NULL, true);
    if(FAILED(hr))
        _com_issue_error(hr);

    hr = spRs.CreateInstance(__uuidof(Recordset));
    if(FAILED(hr))
```

```
        _com_issue_error(hr);

    hr = spRs->QueryInterface(
        __uuidof(ADORecordsetConstruction),(void **)&spADOsCt);
    if(FAILED(hr))
        _com_issue_error(hr);

    hr = spADOsCt->put_Rowset(dbCommand.m_spRowset);
    if(FAILED(hr))
        _com_issue_error(hr);

    //Demonstrates, how to populate DataGrid by assigning it a Recordset object.
    m_ctlDataGrid.SetCaption(sTableName);
    m_ctlDataGrid.SetRefDataSource(NULL);
    m_ctlDataGrid.SetRefDataSource((LPUNKNOWN) spRs );
          m_ctlDataGrid.Refresh();
}
catch(_com_error &e) {
    AfxMessageBox(GetErrorDescription(e));
}

UpdateData(FALSE);
```

Ví dụ 3:

```
try {
    CTables tableSet;
    HRESULT hr = tableSet.Open(pMainFrame->m_session);
    if(SUCCEEDED(hr)) {
        CString sName, sNameShort, sSchema;
        int nPos = -1;
        HRESULT hr = S_OK;
        int nIndex = 0;

        hTreeRoot = GetTreeCtrl().InsertItem("Tables", 0, 0);

        while(tableSet.MoveNext() == S_OK) {
            sName = tableSet.m_szName;
            sNameShort = sName;
            nPos = sName.Find(';');
            if(nPos != -1)
                sName = sName.Left(nPos);
            if(sName.Find(' ') != -1) // MS SQL Server scenario
                sName = "[" + sName + "]";
            // Alternatively...
            sSchema = tableSet.m_szSchema;

            HTREEITEM hTreeSPRoot = GetTreeCtrl().InsertItem(
                sName, 1, 1, hTreeRoot);
        }
    }
    else
    {
        if(FAILED(hr))
            _com_issue_error(hr);
    }
}
```

```
catch(_com_error e) {
    bRet = false;
    AfxMessageBox(GetErrorDescription(e));
}
catch(...) {
    bRet = false;
    AfxMessageBox("UnKnown Error");
}
```

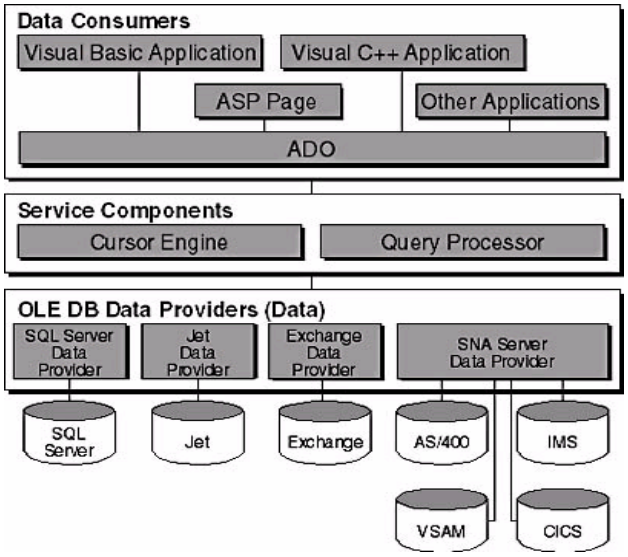
4.3 TRUY XUẤT VỚI ADO

4.3.1 Vấn đề quan tâm

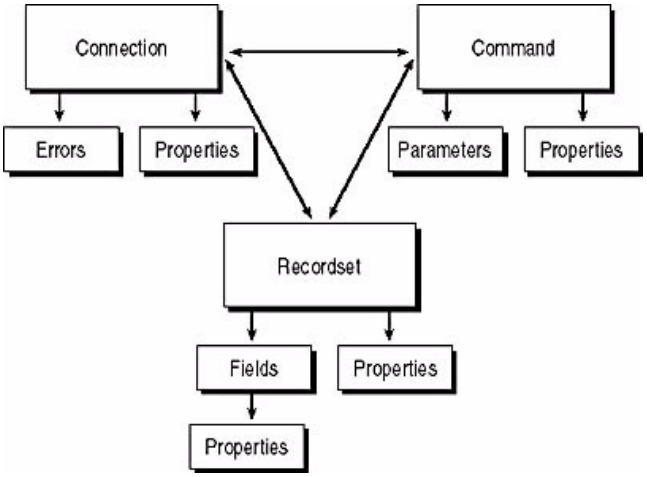
- Hiểu về bản chất của ADO và sử dụng được các class về truy xuất dữ liệu mà MFC hỗ trợ.

4.3.2 Giới thiệu ADO

ADO là một tập hợp các giao diện truy xuất dữ liệu thông qua COM (tương tự như OLEDB nhưng sử dụng đơn giản hơn). ADO đóng vai trò trung gian tương tác với chương trình ứng dụng và cơ sở dữ liệu, được thể hiện như hình sau:



Cấu trúc của ADO bao gồm:



- **Connection**: là đối tượng quản lý các tác vụ truy cập/kết nối cơ sở dữ liệu.
- **Command**: là đối tượng quản lý các tác vụ cập nhật dữ liệu
- **Recordset**: là đối tượng quản lý dữ liệu truy vấn được
- **Error**: là đối tượng quản lý lỗi xãy ra

4.3.3 Thực hiện truy xuất cơ sở dữ liệu với ADO:

Cần thêm khai báo về thư viện ADO vào hàng cuối cùng của file StdAfx.h như sau:

```
#import "c:\program files\common  files\system\ado\msado15.dll" \
no_namespace \
rename ("EOF", "adoEOF")
```

Cần thêm lệnh ::CoInitialize(NULL); trước khi tạo kết nối

Việc truy xuất dữ liệu với ADO được thực hiện thông qua các ví dụ minh hoạ sau đây:

Khai báo biến cần thiết:

```
_ConnectionPtr  m_pConnection;
_CommandPtr             m_pCommand;
```

Khai báo kết nối đến cơ sở dữ liệu:

```
// Kết nối đến Access:
m_szConnection="Provider=Microsoft.JET.OLEDB.4.0;Data source=xxx";
```

(trong đó xxx là tên file Access có đầy đủ đường dẫn)

```
// Kết nối đến SQL Server:
m_szConnection="Provider=SQLOLEDB;Data source=my_server_name;Initial
Catalog=my_database_name;User ID=my_user;Password=my_password;";
```

và:

```
// Kết nối tổng quát:
m_szConnection="File name=xxx";
```

*(trong đó xxx là tên file *.UDL có đầy đủ đường dẫn)*

Kết nối đến cơ sở dữ liệu:

```
// Khoi tao moi truong COM <-- lenh nay bat buoc phai co
::CoInitialize(NULL);
try {
      m_pConnection.CreateInstance(__uuidof(Connection));
      m_pConnection-> Open((_bstr_t)m_szConnectionString,"","",-1);
}
catch(_com_error *e)     {
      CString Error = e->ErrorMessage();
      AfxMessageBox(e->ErrorMessage());
      return FALSE;
}
catch(...)
{
      AfxMessageBox("Loi bat ky");
      return FALSE;
}
```

Đóng kết nối:

```
if (m_pConnection->GetState() == adStateOpen)
      m_pConnection->Close();
m_pConnection = NULL;
```

Truy xuất dữ liệu:

```
if (m_pConnection->GetState() == adStateOpen) {
```

```
    // Reset noi dung dang co trong listbox
    m_lstData.ResetContent();
    //
    m_pRecordset.CreateInstance(__uuidof(Recordset));
    _variant_t myValue;
    _variant_t myKey((short)4);
    int nCount = 0;
    try {
        m_pRecordset->Open(
            "SELECT * FROM Products", m_pConnection.GetInterfacePtr(),
            adOpenDynamic,
            adLockOptimistic,
            adCmdText);
        while(!m_pRecordset->adoEOF) {
            myValue = m_pRecordset->GetCollect("ProductName");
            if (myValue.vt!=VT_NULL) {
                m_lstData.AddString((char*)_bstr_t(myValue));
                myKey = m_pRecordset->GetCollect("ProductID");
                m_lstData.SetItemData(nCount++, (int)myKey.iVal);
            }
            m_pRecordset->MoveNext();
        }
        m_pRecordset->Close();
    }
    catch(_com_error *e) {
        CString Error = e->ErrorMessage();
        AfxMessageBox(e->ErrorMessage());
    }
    catch(...) {
        MessageBox("Whoa");
    }
    // Always set these pointers to null when you are done with them!
    m_pRecordset = NULL;
    UpdateData(FALSE);
}
```

Cập nhật dữ liệu:

```
if (AfxMessageBox("Delete it?", 1,0)!=IDOK)
    return;
if (m_pConnection->GetState() == adStateOpen) {
    //
    UpdateData(TRUE);
    //
    CString szSQL;
```

```
    szSQL.Format("DELETE FROM Products WHERE ProductID=%d", m_nID);
    //
    try {
        m_pConnection->Execute((_bstr_t)szSQL, NULL, 0);
    }
    catch(_com_error *e) {
        CString Error = e->ErrorMessage();
    }
    catch(...)    {
        MessageBox("Whoa");
    }
    OnButtonLoad();
}
```

Ví dụ tổng hợp:

MyADO.h

```
// MyADO.h: interface for the CMyADO class.
//
//////////////////////////////////////////////////////////////////////
#if !defined(AFX_MYADO_H__E144D98C_2388_4800_BC00_F7E963816A73__INCLUDED_)
#define AFX_MYADO_H__E144D98C_2388_4800_BC00_F7E963816A73__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#import "C:\Program Files\Common Files\System\ADO\msado15.dll"
no_namespace
rename( "EOF", "adoEOF" )

class CMyADO
{
public:
      CMyADO();
      virtual ~CMyADO();

      HRESULT Open(_bstr_t btConnectionString,_bstr_t btUserID,
                                                  _bstr_t btPassword);
      HRESULT Close();
      HRESULT AddParameterReturnValue();
      HRESULT AddParameterInputLong(_bstr_t btParameterName, long lValue);
      HRESULT AddParameterInputText(_bstr_t btParameterName, _bstr_t btValue);
      HRESULT AddParameterInputOutputLong(_bstr_t btParameterName, long lValue);
      HRESULT AddParameterInputOutputText(_bstr_t btParameterName,
                                    _bstr_t btValue, DWORD dwMaxTextSize );
      HRESULT AddParameterOutputLong(_bstr_t btParameterName);
      HRESULT AddParameterOutputText(_bstr_t btParameterName,DWORD dwMaxTextSize);
      HRESULT Execute();
      HRESULT GetFieldLong(_bstr_t btFieldName, long* plValue);
      HRESULT GetFieldText(_bstr_t btFieldName,char* szText, DWORD dwMaxTextSize);
      HRESULT GetParameterReturnValue(long* plReturnValue);
      HRESULT GetParameterLong(_bstr_t btParameterName, long* plValue);
```

```
	HRESULT GetParameterText(_bstr_t btParameterName, char* szText,
                                                    DWORD dwMaxTextSize);
	HRESULT Initialize(_bstr_t btStoredProcedureName);
	BOOL IsEOF();
	HRESULT MoveNext();
protected:
	HRESULT GetRecordCount(long* lRecordCount);
private:
	HRESULT AddParameter( _bstr_t btParameterName, DataTypeEnum enDataType,
                              ParameterDirectionEnum enParameterDirection,
                              long lSize, _variant_t vtValue);
	HRESULT GetField(_variant_t vtFieldName, _variant_t& vtValue);
	HRESULT GetParameter(_variant_t vtParameterName, _variant_t& vtValue);
	BOOL IsConnected();
	BOOL IsInitialized();

	_ConnectionPtr m_pConnectionPtr;
	_CommandPtr m_pCommandPtr;
	_RecordsetPtr m_pRecordsetPtr;
};
#endif // !defined(AFX_MYADO_H__E144D98C_2388_4800_BC00_F7E963816A73__INCLUDED_)
```

MyADO.cpp

```
// MyADO.cpp: implementation of the CMyADO class.
//////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <string.h>
#include "MyADO.h"
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
//     Class Constructor
CMyADO::CMyADO()
{
	m_pCommandPtr = NULL;
}
//     Class Destructor
CMyADO::~CMyADO()
{
	if( m_pRecordsetPtr )
	{
		if( m_pRecordsetPtr->State == adStateOpen )
			m_pRecordsetPtr->Close();
		m_pRecordsetPtr = NULL;
	}
	m_pCommandPtr = NULL;
}
//Create a Parameter and Add it to the CommandPtr Object
//(Which will be used to Execute the Stored Procedure)
HRESULT CMyADO::AddParameter( _bstr_t btParameterName, DataTypeEnum enDataType,
ParameterDirectionEnum enParameterDirection, long lSize, _variant_t vtValue )
{
	HRESULT hReturn = S_FALSE;
	if( IsConnected() && IsInitialized())
	{
		try
```

```
            {
                  _ParameterPtr pParameterPtr = m_pCommandPtr->CreateParameter(
btParameterName, enDataType, enParameterDirection, lSize, vtValue );
                  m_pCommandPtr->Parameters->Append( pParameterPtr );
                  hReturn = S_OK;
            }
            catch( _com_error& eComError )
            {
                  char szErrorMsg[256];
                  _snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::Execute() - %s\n", eComError.ErrorMessage());
                  OutputDebugString( szErrorMsg );
            }
            catch( ... )
            {
            }
      }
      return hReturn;
}
//Add Parameter Heler Function for Long Type and Input Direction
HRESULT CMyADO::AddParameterInputLong( _bstr_t btParameterName, long lValue )
{
      return AddParameter( btParameterName, adInteger, adParamInput, sizeof(long),
_variant_t( lValue ));
}
//Add Parameter Helper Function for Text Type and Input Direction
HRESULT CMyADO::AddParameterInputText( _bstr_t btParameterName, _bstr_t btValue )
{
      return AddParameter( btParameterName, adVarChar, adParamInput,
btValue.length(), _variant_t( btValue ));
}
//Add Parameter Helper Function for Long Type and Input/Output Direction
HRESULT CMyADO::AddParameterInputOutputLong( _bstr_t btParameterName, long lValue )
{
      return AddParameter( btParameterName, adInteger, adParamInputOutput, sizeof(
long ), _variant_t( lValue ));
}
//Add Parameter Helper Function for Text Type and Input/Output Direction
HRESULT CMyADO::AddParameterInputOutputText( _bstr_t btParameterName, _bstr_t
btValue, DWORD dwMaxTextSize )
{
      return AddParameter( btParameterName, adVarChar, adParamInputOutput,
dwMaxTextSize, _variant_t( btValue ));
}
//Add Parameter Helper Function for Long Type and Output Direction
HRESULT CMyADO::AddParameterOutputLong( _bstr_t btParameterName )
{
      _variant_t vtNull;
      return AddParameter( btParameterName, adInteger, adParamOutput, 0, vtNull );
}
//Add Parameter Helper Function for Text Type and Output Direction
HRESULT CMyADO::AddParameterOutputText( _bstr_t btParameterName, DWORD
dwMaxTextSize )
{
      _variant_t vtNull;
```

```
	return AddParameter( btParameterName, adVarChar, adParamOutput,
dwMaxTextSize, vtNull );
}
//Add Parameter Helper Function for Return Value
HRESULT CMyADO::AddParameterReturnValue()
{
	_variant_t vtNull;
	return AddParameter( "RETURN_VALUE", adInteger, adParamReturnValue, 0,
vtNull );
}
//Close the Current ADO Connection
HRESULT CMyADO::Close()
{
	HRESULT hReturn = S_FALSE;
	if( m_pConnectionPtr )
	{
		if( m_pConnectionPtr->State == adStateOpen )
		{
			try
			{
				hReturn = m_pConnectionPtr->Close();
				m_pConnectionPtr = NULL;
			}
			catch( _com_error& eComError )
			{
				char szErrorMsg[256];
				_snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::Close() - %s\n", eComError.ErrorMessage());
				OutputDebugString( szErrorMsg );
			}
			catch( ... )
			{
			}
		}
		else
			hReturn = S_OK;
	}
	else
		hReturn = S_OK;
	return hReturn;
}
//Execute the Stored Procedure using the CommandPtr Object
HRESULT CMyADO::Execute()
{
	HRESULT hReturn = S_FALSE;
	if( IsConnected() && IsInitialized())
	{
		try
		{
			m_pRecordsetPtr = m_pCommandPtr->Execute( NULL, NULL,
adCmdStoredProc );
			hReturn = S_OK;
		}
		catch( _com_error& eComError )
		{
```

```
                    char szErrorMsg[256];
                    _snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::Execute() - %s\n", eComError.ErrorMessage());
                    OutputDebugString( szErrorMsg );
             }
             catch( ... )
             {
             }
      }
      return hReturn;
}
//Retrieve a Value from the Recordset (which was created during Stored Procedure
Execution)
HRESULT CMyADO::GetField( _variant_t vtFieldName, _variant_t& vtValue )
{
      HRESULT hReturn = S_FALSE;
      if( IsConnected() && IsInitialized())
      {
             try
             {
                    vtValue = m_pRecordsetPtr->Fields->GetItem(vtFieldName)->Value;
                    hReturn = S_OK;
             }
             catch( _com_error& eComError )
             {
                    char szErrorMsg[256];
                    _snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::GetField() - %s\n", eComError.ErrorMessage());
                    OutputDebugString( szErrorMsg );
             }
             catch( ... )
             {
             }
      }
      return hReturn;
}
//Get Field Helper Function for Long Type
HRESULT CMyADO::GetFieldLong( _bstr_t btFieldName, long* plValue )
{
      _variant_t vtValue;
      HRESULT hReturn = GetField( btFieldName, vtValue );
      if( hReturn == S_OK )
             *plValue = ( long )vtValue;
      return hReturn;
}
//Get Field Helper Function for Text Type
HRESULT CMyADO::GetFieldText( _bstr_t btFieldName, char* szText, DWORD
dwMaxTextSize )
{
      _variant_t vtValue;
      HRESULT hReturn = GetField( btFieldName, vtValue);
      if( hReturn == S_OK )
      {
             _bstr_t btValue = ( _bstr_t )vtValue;
             if( dwMaxTextSize < btValue.length())
```

```
                  hReturn = S_FALSE;
            else
                  strcpy( szText, btValue );
      }
      return hReturn;
}
//Retrieve a Parameter (which was previously set up as either an Output or
InputOutput Direction and is set during Stored Procedure Execution)
HRESULT CMyADO::GetParameter( _variant_t vtParameterName, _variant_t& vtValue )
{
      HRESULT hReturn = S_FALSE;
      if( IsConnected() && IsInitialized())
      {
            try
            {
                  vtValue = m_pCommandPtr->Parameters->GetItem(vtParameterName)
->Value;
                  hReturn = S_OK;
            }
            catch( _com_error& eComError )
            {
                  char szErrorMsg[256];
                  _snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::GetParameter() - %s\n", eComError.ErrorMessage());
                  OutputDebugString( szErrorMsg );
            }
            catch( ... )
            {
            }
      }
      return hReturn;
}
//Retrieve Parameter Helper Function for Long Type
HRESULT CMyADO::GetParameterLong( _bstr_t btParameterName, long* plValue )
{
      _variant_t vtValue;
      HRESULT hReturn = GetParameter( btParameterName, vtValue );
      if( hReturn == S_OK )
            *plValue = ( long )vtValue;
      return hReturn;
}
//Retrieve Parameter Helper Function for Return Value
HRESULT CMyADO::GetParameterReturnValue( long* plReturnValue )
{
      return GetParameterLong( "RETURN_VALUE", plReturnValue );
}
//Retrieve Parameter Helper Function for Text Type
HRESULT CMyADO::GetParameterText( _bstr_t btParameterName, char* szText, DWORD
dwMaxTextSize )
{
      _variant_t vtValue;
      HRESULT hReturn = GetParameter( btParameterName, vtValue );
      if( hReturn == S_OK )
      {
            _bstr_t btValue = ( _bstr_t )vtValue;
```

```
		if( dwMaxTextSize < btValue.length())
			hReturn = S_FALSE;
		else
			strcpy( szText, btValue );
	}
	return hReturn;
}
//Retrieve the Record Count for the Recordset (which was created during Stored
Procedure Execution)
HRESULT CMyADO::GetRecordCount( long* lRecordCount )
{
	HRESULT hReturn = S_FALSE;
	if( m_pRecordsetPtr )
	{
		try
		{
			*lRecordCount = m_pRecordsetPtr->RecordCount;
		}
		catch( _com_error& eComError )
		{
			char szErrorMsg[256];
			_snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::GetParameter() - %s\n", eComError.ErrorMessage());
			OutputDebugString( szErrorMsg );
		}
		catch( ... )
		{
		}
	}
	return hReturn;
}
//Close the Recordset and Initialize the CommandPtr Object
HRESULT CMyADO::Initialize( _bstr_t btStoredProcedureName )
{
	HRESULT hReturn = S_FALSE;
	if( IsConnected())
	{
		m_pCommandPtr = NULL;

		if( m_pRecordsetPtr )
		{
			if( m_pRecordsetPtr->State == adStateOpen )
				m_pRecordsetPtr->Close();
			m_pRecordsetPtr = NULL;
		}
		m_pCommandPtr.CreateInstance( __uuidof( Command ));
		m_pCommandPtr->ActiveConnection = m_pConnectionPtr;
		m_pCommandPtr->CommandText = btStoredProcedureName;
		m_pCommandPtr->CommandType = adCmdStoredProc;

		hReturn = S_OK;
	}
	return hReturn;
}
//Check for Connection Status
```

```
BOOL CMyADO::IsConnected()
{
	return ( m_pConnectionPtr );
}
//Check for EOF on the Recordset
//(which was created during Stored Procedure Execution)
BOOL CMyADO::IsEOF()
{
	BOOL bReturn = TRUE;
	if( m_pRecordsetPtr )
		if(m_pRecordsetPtr->State == adStateOpen && !m_pRecordsetPtr->adoEOF)
			bReturn = FALSE;
	return bReturn;
}
//Check for Initialization Status (CommandPtr Object is valid)
BOOL CMyADO::IsInitialized()
{
	return ( m_pCommandPtr );
}
//Open a new ADO Connection
HRESULT CMyADO::Open( _bstr_t btConnectionString, _bstr_t btUserID, _bstr_t
btPassword )
{
	HRESULT hReturn = S_FALSE;
	if( m_pConnectionPtr == NULL )
	{
		m_pConnectionPtr.CreateInstance( __uuidof( Connection ));
		try
		{
			hReturn = m_pConnectionPtr->Open( btConnectionString, btUserID,
btPassword, 0 );
			if( hReturn == S_OK )
				m_pConnectionPtr->CursorLocation = adUseClient;
		}
		catch( _com_error& eComError )
		{
			char szErrorMsg[256];
			_snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::Open( '%s', '%s', '%s' ) - %s\n", ( char* )btConnectionString, ( char*
)btUserID, ( char* )btPassword, eComError.ErrorMessage());
			OutputDebugString( szErrorMsg );
		}
		catch( ... )
		{
		}
	}
	else
		hReturn = S_OK;
	return hReturn;
}
//Move to the Next Record in the Recordset (which was created during Stored
Procedure Execution)
HRESULT CMyADO::MoveNext()
{
	HRESULT hResult = S_FALSE;
```

```
	if( !IsEOF())
	{
		try
		{
			hResult = m_pRecordsetPtr->MoveNext();
		}
		catch( _com_error& eComError )
		{
			char szErrorMsg[256];
			_snprintf( szErrorMsg, sizeof( szErrorMsg ), "ERROR in
CMyADO::MoveNext() - %s\n", eComError.ErrorMessage());
			OutputDebugString( szErrorMsg );
		}
		catch( ... )
		{
		}
	}
	return hResult;
}
```

Main.cpp

```
#include <stdio.h>
#include "MyADO.h"

void main()
{
	// Initialize COM
	CoInitialize( NULL );

	// Instantiate the Object
	CMyADO MyADOObject;

	// Open( ConnectionString, UserID, Password ) the Connection
	if( MyADOObject.Open( "TestMyADO", "", "" ) == S_OK )
	{
		// Some Example Strings
		char* pNames[] = { "First Name", "Second Name", "Third Name", "Fourth
Name", "Fifth Name", "Sixth Name", "Seventh Name", "Eighth Name", "Nineth Name",
"Tenth Name" };
		char* pValues[] = { "Value One", "Value Two", "Value Three", "Value
Four", "Value Five", "Value Six", "Value Seven", "Value Eight", "Value Nine",
"Value Ten" };
		// The Index
		DWORD dwIndex = 0;

		// The Insert Loop
		while( dwIndex < 10 )
		{
			// Initialize( StoredProcedureName ) the Stored Procedure
			if( MyADOObject.Initialize( "InsertMyADO" ) == S_OK )
			{
				// Add the Return Value Parameter
				if( MyADOObject.AddParameterReturnValue() == S_OK )
				{
					// Add the Output Long Parameter
```

```
                                    if( MyADOObject.AddParameterOutputLong( "ID" ) ==
S_OK )
                                    {
                                          // Add the Input Text Parameters
                                          if( MyADOObject.AddParameterInputText(
"Name", pNames[dwIndex] ) == S_OK &&
                                                MyADOObject.AddParameterInputText(
"Value", pValues[dwIndex] ) == S_OK )
                                          {
                                                // Execute the Stored Procedure
                                                if( MyADOObject.Execute() == S_OK )
                                                {
                                                      long lReturnValue = 0;
                                                      long lID = 0;

                                                      // Retrieve the Return Value
and the Output Paramter set up above
                                                      if(
MyADOObject.GetParameterReturnValue( &lReturnValue ) == S_OK &&

      MyADOObject.GetParameterLong( "ID", &lID ) == S_OK )
                                                      {
                                                            // Sanity check that
does nothing : )
                                                            if( lReturnValue == lID
)
                                                                  printf(
"Inserted Record with ID: %2d, Name: %15s, Value: %15s\n", lID, pNames[dwIndex],
pValues[dwIndex] );
                                                      }
                                                      else
                                                            printf( "ERR: Unable to
Retrieve Return Value And ID\n" );
                                                }
                                                else
                                                      printf( "ERR: Unable to
Execute InsertMyADO\n" );
                                          }
                                          else
                                                printf( "ERR: Unable to Add the
Input Text Values\n" );
                                    }
                                    else
                                          printf( "ERR: Unable to Add the Output Long
Value\n" );
                              }
                              else
                                    printf( "ERR: Unable to Add the Return Value\n" );
                        }
                        else
                              printf( "ERR: Unable to Initialize InsertMyADO\n" );

                        // Increment the Index
                        dwIndex++;
                  }
```

```
            // Initialize( StoredProcedureName ) the Stored Procedure
            if( MyADOObject.Initialize( "SelectMyADO" ) == S_OK )
            {
                  // Execute the Stored Procedure
                  if( MyADOObject.Execute() == S_OK )
                  {
                        // Ensure there are more Records to Retrieve
                        while( !MyADOObject.IsEOF())
                        {
                              long lID = 0;
                              char szName[15];
                              char szValue[15];

                              // Retrieve the Record Fields
                              if( MyADOObject.GetFieldLong( "ID", &lID ) == S_OK
&&
                                    MyADOObject.GetFieldText( "Name", szName,
sizeof( szName )) == S_OK &&
                                    MyADOObject.GetFieldText( "Value", szValue,
sizeof( szValue )) == S_OK )
                                    printf( "Selected Record with ID: %2d,
Name: %15s, Value: %15s\n", lID, szName, szValue );

                              // Move to the Next Record
                              MyADOObject.MoveNext();
                        }
                  }
                  else
                        printf( "ERR: Unable to Execute SelectMyADO\n" );
            }
            else
                  printf( "ERR: Unable to Initialize SelectMyADO\n" );

            // Close the Connection
            MyADOObject.Close();
      }

      // For every action there must be an opposite and equal reaction
      CoUninitialize();
}
```

CHƯƠNG 5. MFC VÀ ACTIVE X

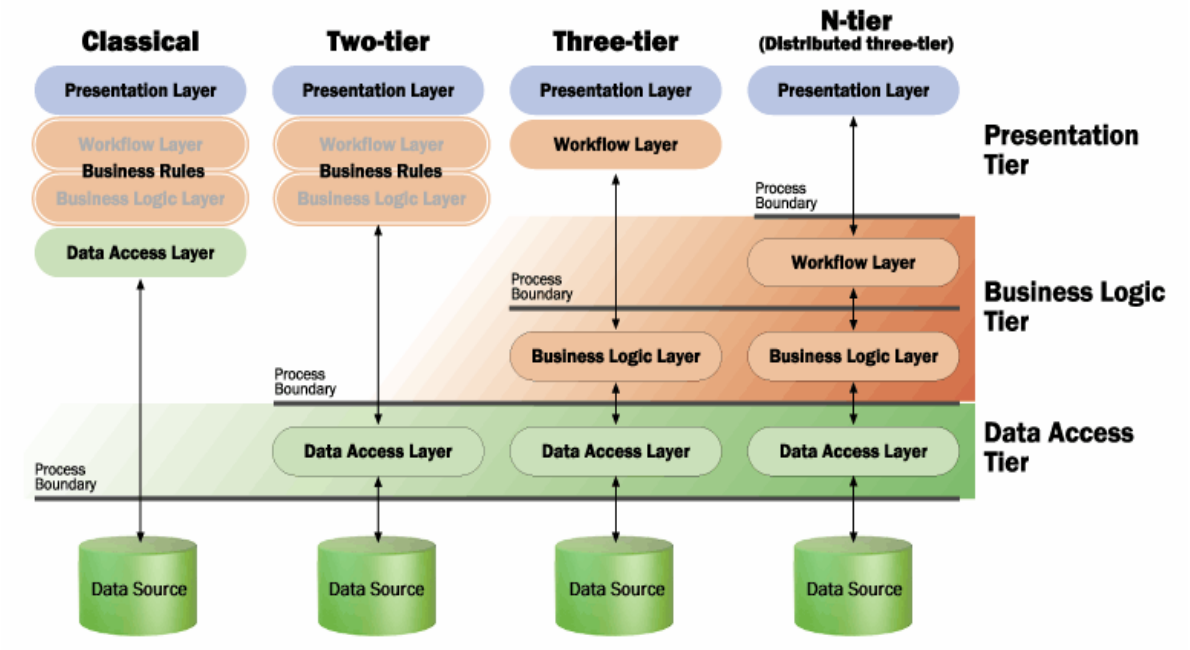
5.1 COMPONENT OBJECT MODEL (COM)

5.1.1 Vấn đề quan tâm

- Hiểu cấu trúc và vai trò của mô hình COM
- Khai thác được ưu điểm của COM trong lập trình ứng dụng.

5.1.2 Giới thiệu mô hình 3-lớp và n-lớp

Là cấu trúc ứng dụng mà trong đó các nhóm ứng dụng được phân chia theo tính năng và vai trò theo từng lớp (*tier*).



Mỗi mô hình có từng ưu/khuyết điểm khác nhau, được sử dụng trong các ngữ cảnh khác nhau.

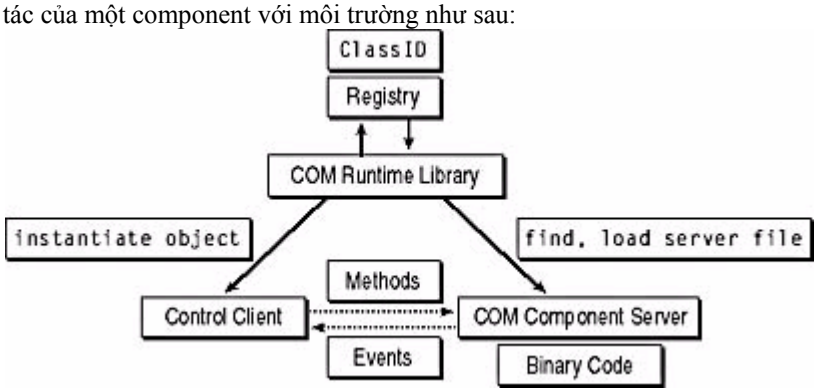
5.1.3 Giới thiệu COM

COM là mô hình đối tượng thành phần, được sử dụng trong các ứng dụng 3-lớp (hoặc n-lớp) nhằm tăng hiệu suất hoạt động, tính bảo mật của hệ thống và tính linh hoạt cao.

Với COM, chúng ta có thể mở rộng khả năng liên kết và tích hợp giữa các thành phần (độc lập ngôn ngữ) trong hệ thống vì bản thân COM được thể hiện ở dạng mã nhị phân.

Có thể dùng VC++ hay VB để tạo COM, và bất cứ ứng dụng nào cũng có thể tương tác với COM (nếu được hỗ trợ).

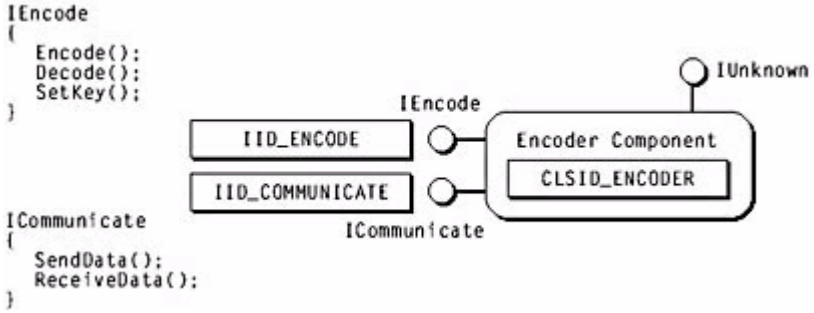
Tổ chức và tương tác của một component với môi trường như sau:



5.1.4 Cấu trúc COM

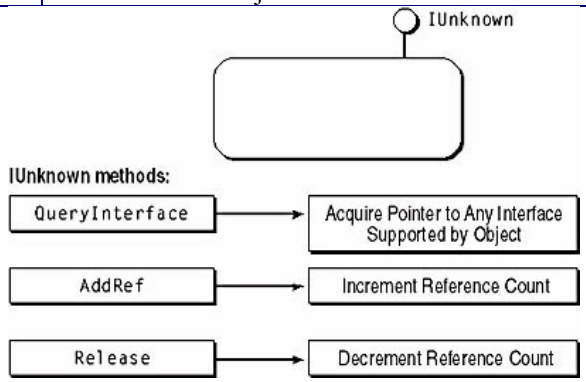
COM được tổ chức dưới dạng tập các class – trong đó mỗi class bao gồm các ***method*** (hàm thành viên), ***attribute*** (biến thành viên) và các ***property***.

Các method và các property được nhóm vào trong các interface (giao diện) – các interface có (như là vỏ bọc của các class) có nhiệm vụ giao tiếp với các ứng dụng khác nhằm che dấu/ngăn chặn sự truy cập trực tiếp vào các thành phần của class trong COM - đồng thời cung cấp các phương thức truy cập các method và property này. Microsoft đã định nghĩa sẵn khoảng 100 interface để hỗ trợ cho COM, nhưng người sử dụng có thể tự tạo interface riêng biệt.

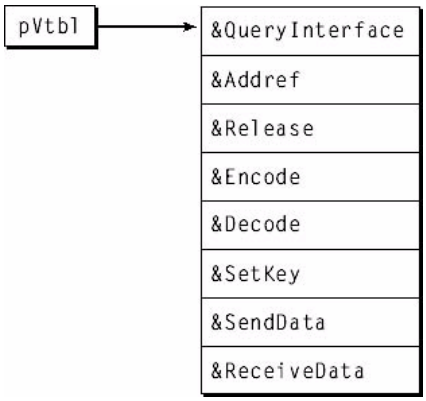


Mỗi COM khi tạo ra luôn có sẵn interface IUnknown và các method như:

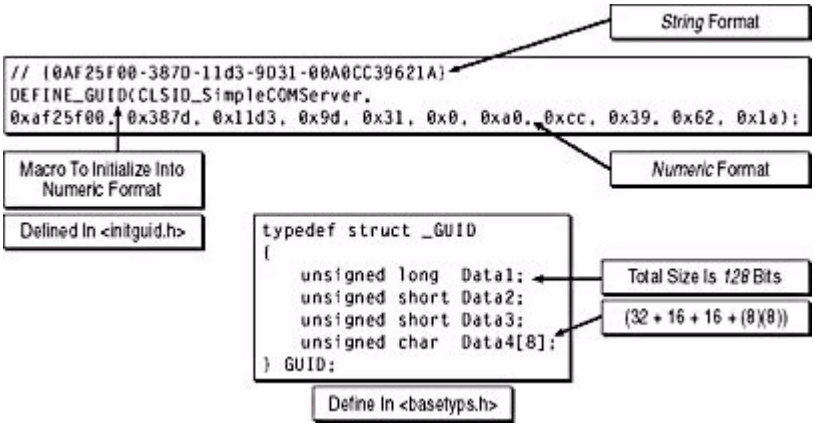
Phương thức	Mô tả
QueryInterface	Returns a pointer to another interface
AddRef	Increments the object's reference count
Release	Decrements the object's reference count



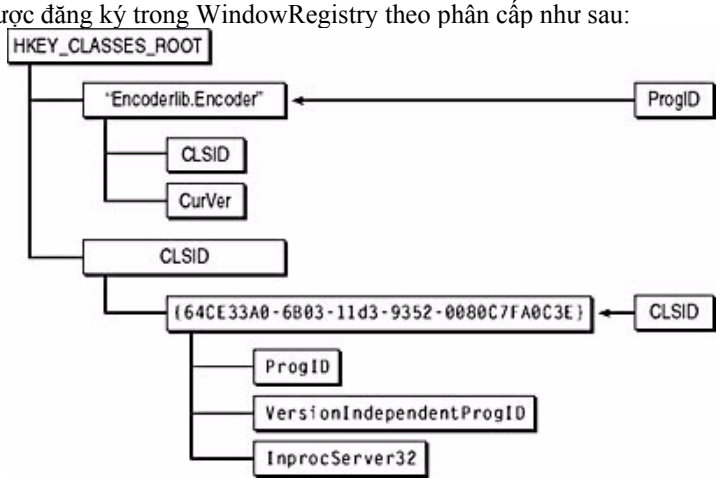
và tổ chức lưu trữ như sau:



Mỗi class trong COM khi tạo ra cần có class ID (hay CLSID) để khai báo với hệ thống, giá trị này được thể hiện bởi chuỗi định nghĩa 128-bit, trong đó thông tin liên quan gồm:



Mỗi component đều được đăng ký trong WindowRegistry theo phân cấp như sau:



Ví dụ 1:

Để khởi tạo đối tượng của COM, gọi hàm kích hoạt **CoCreateInstance** cùng với CLSID của đối tượng đó, kết quả nhận về là một pointer quản lý của đối tượng đó.

Ví dụ 2:

```
Khởi tạo một COM class có CLSID là CLSID_Object và liên kết tới interface IMath bởi
pointer pMath
IMath* pMath;
CoCreateInstance (CLSID_Object, NULL, CLSCTX_SERVER, IID_IMath, (void**) &pMath);
```

Đối tượng COM được khởi tạo bởi thao tác new nhưng được tự động hủy (mà không dùng delete). Có thể một interface hỗ trợ nhiều truy xuất đồng thời, số lượng truy xuất có thể được nhận biết và quản lý bởi một biến thành viên và các sự kiện AddRef, Release:

Ví dụ 3:

```
ULONG __stdcall CComClass::AddRef()
{
    return ++m_lRef;
}
ULONG __stdcall CComClass::Release()
{
    if (--m_lRef == 0) {
        delete this;
        return 0;
    }
    return m_lRef;
}
```

Vấn đề truy xuất đồng thời nhiều đối tượng được xử lý theo trình tự sau:

Ví dụ 4:

```
IMath* pMath;
HRESULT hr = CoCreateInstance (CLSID_Object, NULL,
```

```
    CLSCTX_SERVER, IID_IMath, (void**) &pMath);

if (SUCCEEDED (hr)) { // CoCreateInstance worked.

    ISpelling* pSpelling;
    hr = pMath->QueryInterface (IID_ISpelling, (void**) &pSpelling);
    if (SUCCEEDED (hr)) {
        // Got the interface pointer!
        …
        pSpelling->Release();
    }
    pMath->Release();
}
```

5.1.5 COM server

Một file đối tượng thực thi mà hiện thực một đối tượng COM được gọi là COM server.

Section HKEY_CLASSES_ROOT\CLSID trong registry sẽ chứa các thông tin về CLSID và các đối tượng thực thi liên quan.

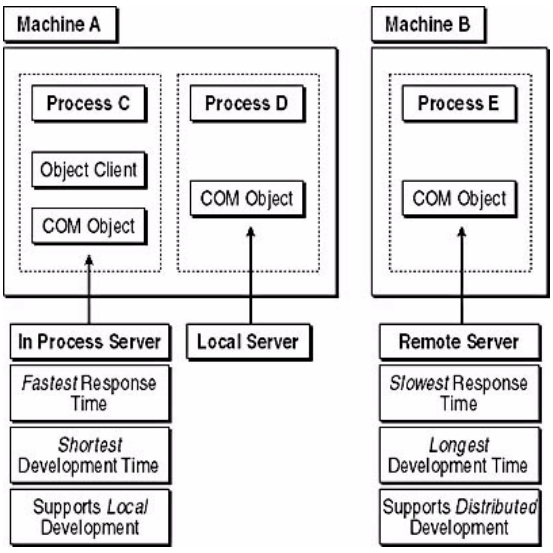
Ví dụ 5:

```
Nếu server file là MathSvr.exe hiện thực cho đối tượng Math objects, và người dùng
gọI CoCreateInstance vớI CLSID của Math, thì COM sẽ tìm CLSID trong registry, và
lấy được đường dẫn của MathSvr.exe rồi thực thi file EXE này.
```

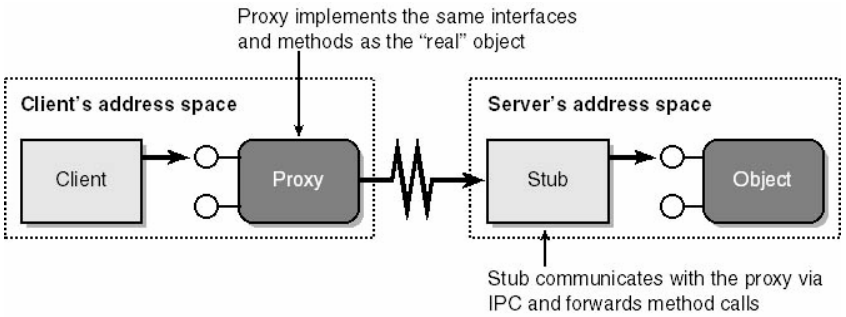
Có 2 kiểu COM-server là:

- **in-process**: là COM có dạng file là DLL, khi hoạt động thì được nạp (cất/lưu) trong cùng không gian địa chỉ của ứng dụng client → tối ưu hơn trong hiện thực.
- **out-of-process**: là COM có dạng file là EXE, khi hoạt động thì được nạp vào một không gian địa chỉ khác với ứng dụng client → có khuyết điểm là ảnh hưởng đến tốc độ thực thi của ứng dụng.

Mô hình DCOM (*Distributed COM*) được giới thiệu với mô hình dạng out-of-process chạy tự do trên các máy server của môi trường mạng.



Một ưu điểm của COM là tính chất "trong suốt vị trí" (*Location Transparency*), tức là chương trình client không quan tâm đến vị trí của các object của COM, và COM sẽ quản lý tất cả thao tác hỗ trợ việc kết nối này. Việc tương tác này thể hiện qua cấu trúc proxy/stub mà COM đã hỗ trợ sẵn.



5.1.6 MFC và COM

Interface có thể được khai báo kế thừa nhau.

Ví dụ 6:

```
interface IUnknown
{
    virtual HRESULT __stdcall QueryInterface(REFIID riid, void** ppv)= 0;
    virtual ULONG __stdcall AddRef()=0;
    virtual ULONG __stdcall Release()=0;
};
…
interface IMath : public IUnknown
{
    virtual HRESULT __stdcall Add(int a, int b, int* pResult)=0;
    virtual HRESULT __stdcall Subtract(int a, int b, int* pResult)=0;
};
…
class CComClass : public IMath
{
protected:
    long m_lRef;    // Reference count
public:
    CComClass ();
    virtual ~CComClass ();
    // IUnknown methods
    virtual HRESULT __stdcall QueryInterface (REFIID riid, void** ppv);
    virtual ULONG __stdcall AddRef();
    virtual ULONG __stdcall Release();
    // IMath methods
    virtual HRESULT __stdcall Add (int a, int b, int* pResult);
    virtual HRESULT __stdcall Subtract (int a, int b, int* pResult);
};
…
class CComClass : public IMath, public ISpelling
{
protected:
    long m_lRef;    // Reference count
public:
    CComClass ();
    virtual ~CComClass();
    // IUnknown methods
    virtual HRESULT __stdcall QueryInterface(REFIID riid, void** ppv);
    virtual ULONG __stdcall AddRef ();
    virtual ULONG __stdcall Release();
    // IMath methods
    virtual HRESULT __stdcall Add(int a, int b, int* pResult);
```

```
    virtual HRESULT __stdcall Subtract(int a, int b, int* pResult);
    // ISpelling methods
    virtual HRESULT __stdcall CheckSpelling(wchar_t* pString);
};
```

và có thể lấy pointer interface của IMath bởi:

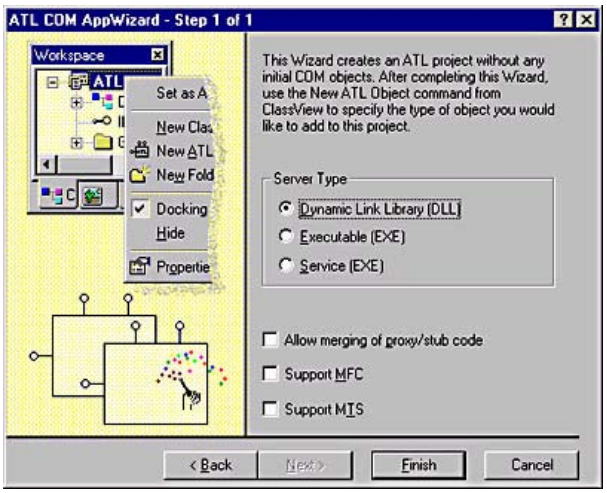
```
*ppv = (IMath*) this;
```

hay:

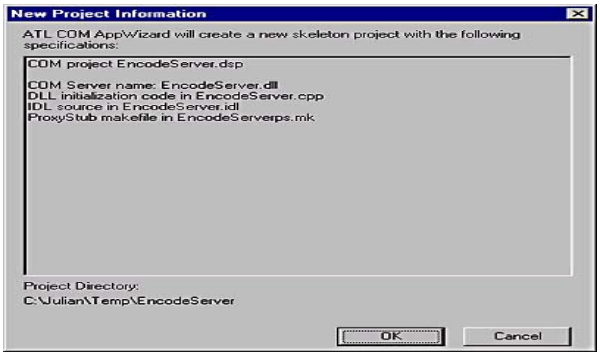
```
*ppv = (ISpelling*) this;
```

Ví dụ tổng hợp:

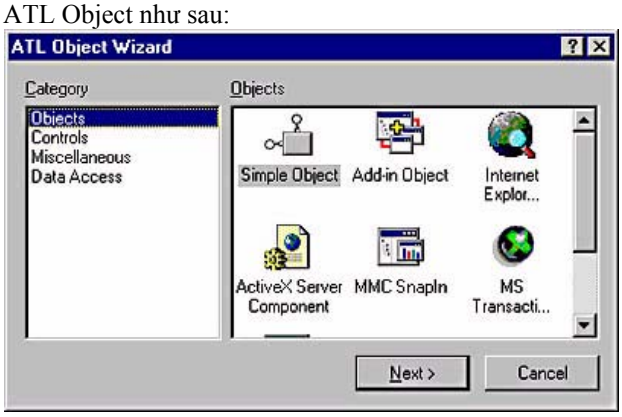
- B1:



và:

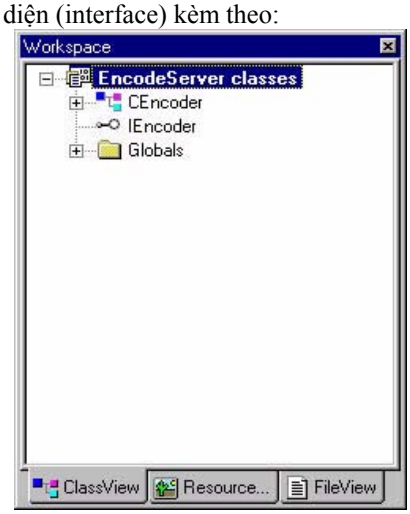


- B2: Chọn tạo mới ATL Object như sau:

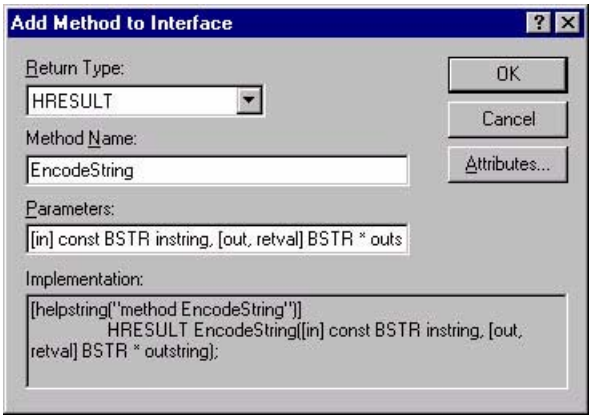


Đối tượng cần tạo có thôntg tin như sau:

Đối tượng được tạo ra bao gồm giao diện (interface) kèm theo:



Chọn để thêm một method vào đối tượng như sau:



trong đó hàm EncodeString cần được cập nhật như sau:

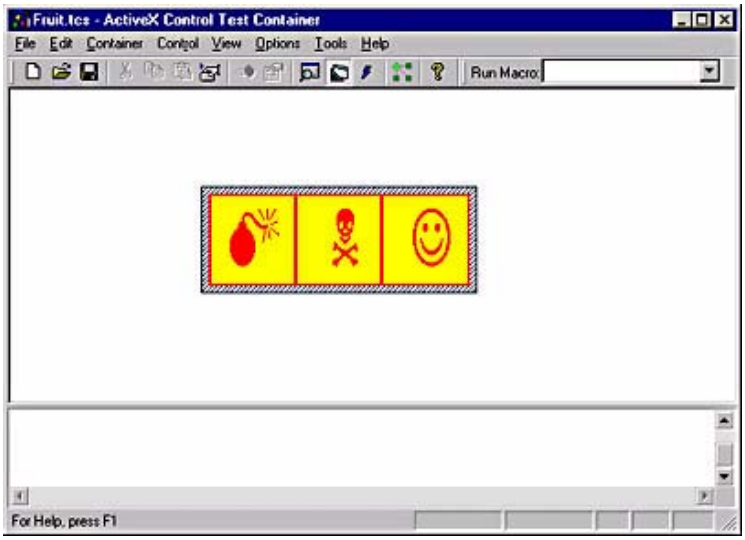
```
STDMETHODIMP CEncoder::EncodeString(const BSTR instring, BSTR *outstring)
{
     BSTR tempstring = ::SysAllocString(instring);
     wcscpy(tempstring, instring);

     for(UINT i = 0; i < ::SysStringLen(tempstring); i++)
          tempstring[i] += m_Key;

     *outstring = ::SysAllocString(tempstring);

     ::SysFreeString(tempstring);

     return S_OK;
}
```

Thực hiện tương tự với **Add Property** để thêm một property *Key* như sau:

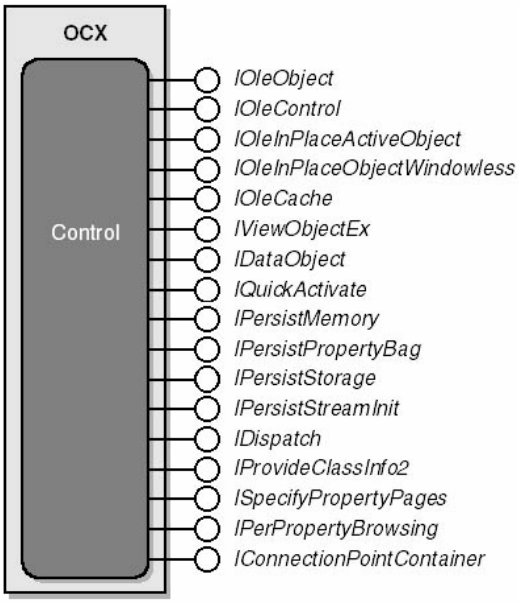
```
STDMETHODIMP CEncoder::get_Key(short *pVal)
{
     *pVal = m_Key;

     return S_OK;
}
STDMETHODIMP CEncoder::put_Key(short newVal)
{
     newVal = newVal > 5 ? 5 : newVal;
     newVal = newVal < -5 ? -5 : newVal;
     m_Key = newVal;

     return S_OK;
}
```

5.2 ACTIVE X CONTROL

ActiveX Control là một điều khiển được xây dựng sẵn hoặc được phát triển theo ý muốn người lập trình, ví dụ như ActiveX control sau đây:



Cấu trúc của một ActiveX Control như sau:

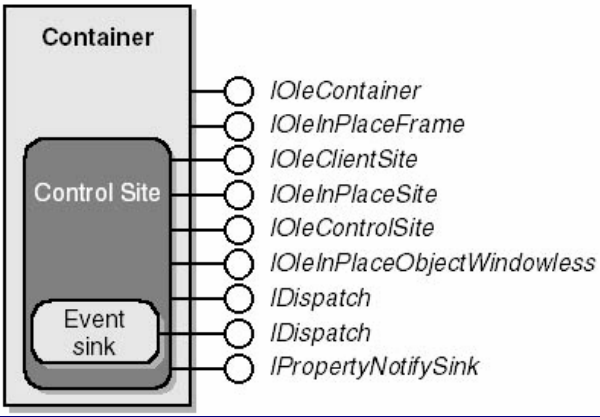


trong đó các giao diện thành phần như sau:

Interface	Comments
IConnectionPointContainer	Exposes connection points for event interfaces
IDataObject	Makes presentation data available to the control container
IDispatch	Exposes the control's methods and properties
IOleCache	Controls the presentation data cache
IOleControl	Base interface for ActiveX controls
IOleInPlaceActiveObject	Base interface for embedded objects that support in-place activation
IOleInPlaceObjectWindowless	Allows the container to manage the activation and deactivation of both windowed and windowless controls
IOleObject	Base interface for embedded objects
IQuickActivate	Speeds control creation in containers that recognize this interface
IPerPropertyBrowsing	Allows containers to acquire information about control properties, such as each property's name
IPersistMemory	Allows the control to write property values to memory and read them back
IPersistPropertyBag	Allows the control to save property values in "property bag" objects provided by the container
IPersistStorage	Allows the control to save property values in storage objects
IPersistStreamInit	Allows the control to save property values in stream objects

IProvideClassInfo2	Makes type information available to the control container
ISpecifyPropertyPages	Allows the control to add pages to property sheets displayed by the container
IViewObjectEx	Allows the container to acquire images of inactive controls and paint windowless controls

Tổ chức lớp chứa và sự kiện như sau:



Interface	Comments
IOleContainer	Base interface for embedding containers
IOleInPlaceFrame	Base interface for OLE containers that support in-place activation
IOleClientSite	Base interface for OLE containers
IOleInPlaceSite	Base interface for OLE containers that support in-place activation
IOleControlSite	Base interface for ActiveX control sites
IDispatch	Exposes the container's ambient properties
IDispatch	Traps events fired by a control
IPropertyNotifySink	Allows the control to notify the container about property changes and to ask permission before changing them

Các hàm ảo hỗ trợ trong COleControl như sau:

Function	Description
OnDraw	Called to paint the control. Override to add control-specific painting logic.
DoPropExchange	Called to save or load a control's persistent properties. Override to support persistent control properties.

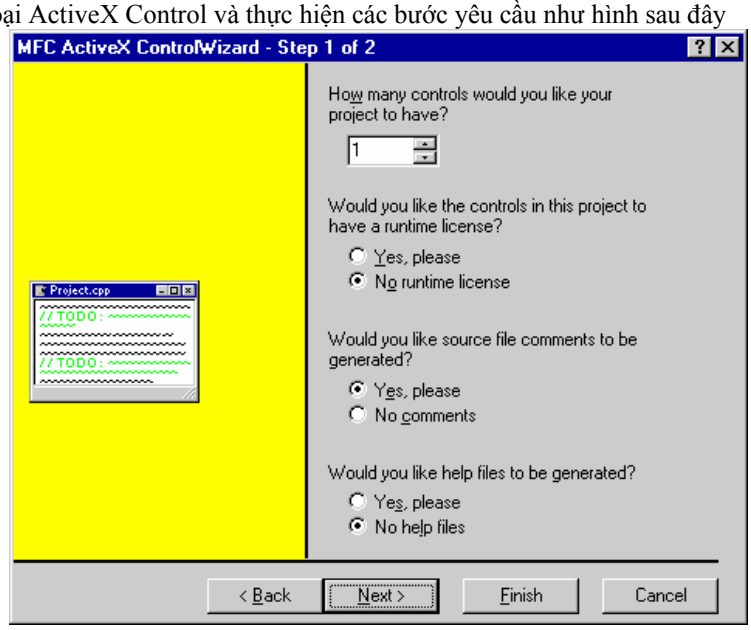
Các hàm hỗ trợ trong lớp COleControl như sau:

Function	Description
Ambientxxx	Retrieves an ambient property value from the container (for example, AmbientBackColor)
Firexxx	Fires a stock event (for example, FireClick)
GetAmbientProperty	Retrieves the values of an ambient property for which no Ambientxxx function is defined
Getxxx	Retrieves the value of a stock property (for example, GetBackColor)
InitializeIIDs	Makes the IDs of the control's event interface and IDispatch interface known to MFC; normally called from the class constructor
InvalidateControl	Repaints the control
SerializeStockProps	Serializes the control's stock properties
SetModifiedFlag	Marks the control as dirty or not dirty (A "dirty" control is one that contains unsaved property changes.)
SetNotSupported	Generates an error when a client attempts to write to a read-only property

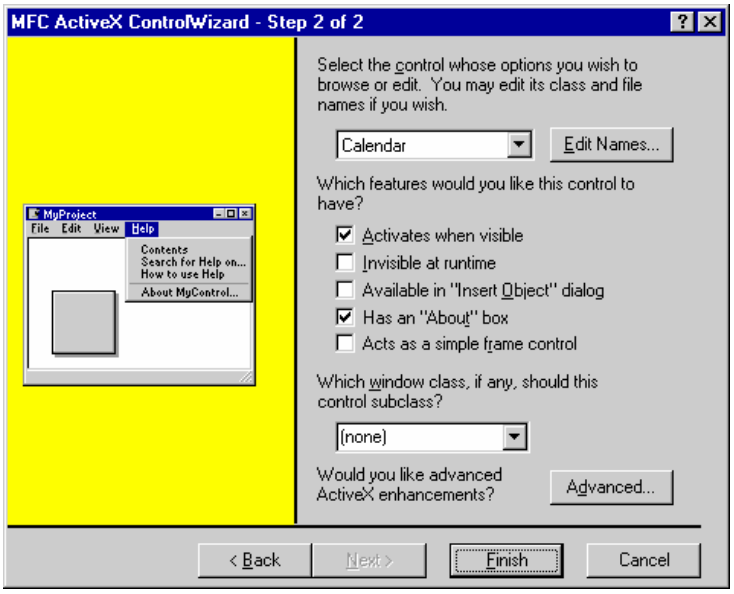
ThrowError	Signals that an error occurred; used in method implementations and property accessor functions
TranslateColor	Translates an OLE_COLOR value into a COLORREF value

Thao tác tạo ActiveX Control:

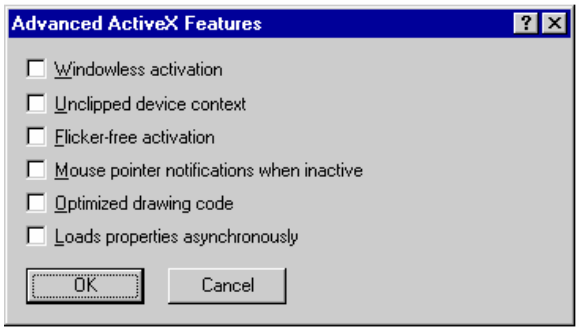
- B1: Chọn loại ActiveX Control và thực hiện các bước yêu cầu như hình sau đây



và:



và:



Ví dụ tổng hợp:

CalendarCtl.h

```
#if !defined(
    AFX_CALENDARCTL_H__68932D29_CFE2_11D2_9282_00C04F8ECF0C__INCLUDED_)
#define AFX_CALENDARCTL_H__68932D29_CFE2_11D2_9282_00C04F8ECF0C__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// CalendarCtl.h : Declaration of the CCalendarCtrl ActiveX Control class.
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl : See CalendarCtl.cpp for implementation.
class CCalendarCtrl : public COleControl
{
    DECLARE_DYNCREATE(CCalendarCtrl)
// Constructor
public:
    CCalendarCtrl();
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CCalendarCtrl)
    public:
    virtual void OnDraw(CDC* pdc, const CRect& rcBounds,
        const CRect& rcInvalid);
    virtual void DoPropExchange(CPropExchange* pPX);
    virtual void OnResetState();
    //}}AFX_VIRTUAL

// Implementation
protected:
    BOOL LeapYear(int nYear);
    static const int m_nDaysPerMonth[];
    int m_nDay;
    int m_nMonth;
    int m_nYear;
    ~CCalendarCtrl();

    DECLARE_OLECREATE_EX(CCalendarCtrl)    // Class factory and guid
    DECLARE_OLETYPELIB(CCalendarCtrl)      // GetTypeInfo
    DECLARE_PROPPAGEIDS(CCalendarCtrl)     // Property page IDs
    DECLARE_OLECTLTYPE(CCalendarCtrl)      // Type name and misc status

// Message maps
    //{{AFX_MSG(CCalendarCtrl)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

// Dispatch maps
    //{{AFX_DISPATCH(CCalendarCtrl)
    BOOL m_bRedSundays;
    afx_msg void OnRedSundaysChanged();
    afx_msg DATE GetDate();
    afx_msg BOOL SetDate(short nYear, short nMonth, short nDay);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()

    afx_msg void AboutBox();

// Event maps
    //{{AFX_EVENT(CCalendarCtrl)
    void FireNewDay(short nDay)
        {FireEvent(eventidNewDay,EVENT_PARAM(VTS_I2), nDay);}
    //}}AFX_EVENT
    DECLARE_EVENT_MAP()

// Dispatch and event IDs
public:
    enum {
    //{{AFX_DISP_ID(CCalendarCtrl)
    dispidRedSundays = 1L,
    dispidGetDate = 2L,
    dispidSetDate = 3L,
    eventidNewDay = 1L,
    //}}AFX_DISP_ID
    };
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.
#endif
// !defined(
//     AFX_CALENDARCTL_H__68932D29_CFE2_11D2_9282_00C04F8ECF0C__INCLUDED)
```

CalendarCtl.cpp

```
// CalendarCtl.cpp : Implementation of the
// CCalendarCtrl ActiveX Control class.

#include "stdafx.h"
#include "Calendar.h"
#include "CalendarCtl.h"
#include "CalendarPpg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CCalendarCtrl, COleControl)
```

```
const int CCalendarCtrl::m_nDaysPerMonth[] = {
    31,         // January
    28,         // February
    31,         // March
    30,         // April
    31,         // May
    30,         // June
    31,         // July
    31,         // August
    30,         // September
    31,         // October
    30,         // November
    31,         // December
};
////////////////////////////////////////////////////////////////////////////
// Message map
BEGIN_MESSAGE_MAP(CCalendarCtrl, COleControl)
    //{{AFX_MSG_MAP(CCalendarCtrl)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
    ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////////
// Dispatch map
BEGIN_DISPATCH_MAP(CCalendarCtrl, COleControl)
    //{{AFX_DISPATCH_MAP(CCalendarCtrl)
    DISP_PROPERTY_NOTIFY(CCalendarCtrl, "RedSundays", m_bRedSundays,
        OnRedSundaysChanged, VT_BOOL)
    DISP_FUNCTION(CCalendarCtrl, "GetDate", GetDate, VT_DATE, VTS_NONE)
    DISP_FUNCTION(CCalendarCtrl, "SetDate", SetDate, VT_BOOL,
        VTS_I2 VTS_I2 VTS_I2)
    DISP_STOCKPROP_BACKCOLOR()
    //}}AFX_DISPATCH_MAP
    DISP_FUNCTION_ID(CCalendarCtrl, "AboutBox", DISPID_ABOUTBOX,
        AboutBox, VT_EMPTY, VTS_NONE)
END_DISPATCH_MAP()
////////////////////////////////////////////////////////////////////////////
// Event map
BEGIN_EVENT_MAP(CCalendarCtrl, COleControl)
    //{{AFX_EVENT_MAP(CCalendarCtrl)
    EVENT_CUSTOM("NewDay", FireNewDay, VTS_I2)
    //}}AFX_EVENT_MAP
END_EVENT_MAP()
////////////////////////////////////////////////////////////////////////////
// Property pages
// TODO: Add more property pages as needed.
// Remember to increase the count!
BEGIN_PROPPAGEIDS(CCalendarCtrl, 2)
    PROPPAGEID(CCalendarPropPage::guid)
    PROPPAGEID (CLSID_CColorPropPage)
END_PROPPAGEIDS(CCalendarCtrl)

////////////////////////////////////////////////////////////////////////////
// Initialize class factory and guid
IMPLEMENT_OLECREATE_EX(CCalendarCtrl, "CALENDAR.CalendarCtrl.1",
```

```
    0xed780d6b, 0xcc9f, 0x11d2, 0x92, 0x82, 0, 0xc0, 0x4f, 0x8e, 0xcf, 0xc)

////////////////////////////////////////////////////////////////////////////
// Type library ID and version
IMPLEMENT_OLETYPELIB(CCalendarCtrl, _tlid, _wVerMajor, _wVerMinor)
////////////////////////////////////////////////////////////////////////////
// Interface IDs
const IID BASED_CODE IID_DCalendar =
        { 0x68932d1a, 0xcfe2, 0x11d2,
            { 0x92, 0x82, 0, 0xc0, 0x4f, 0x8e, 0xcf, 0xc } };
const IID BASED_CODE IID_DCalendarEvents =
        { 0x68932d1b, 0xcfe2, 0x11d2,
            { 0x92, 0x82, 0, 0xc0, 0x4f, 0x8e, 0xcf, 0xc } };
////////////////////////////////////////////////////////////////////////////
// Control type information
static const DWORD BASED_CODE _dwCalendarOleMisc =
    OLEMISC_ACTIVATEWHENVISIBLE œ
    OLEMISC_SETCLIENTSITEFIRST œ
    OLEMISC_INSIDEOUT œ
    OLEMISC_CANTLINKINSIDE œ
    OLEMISC_RECOMPOSEONRESIZE;
IMPLEMENT_OLECTLTYPE(CCalendarCtrl, IDS_CALENDAR, _dwCalendarOleMisc)
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::CCalendarCtrlFactory::UpdateRegistry -
// Adds or removes system registry entries for CCalendarCtrl
BOOL CCalendarCtrl::CCalendarCtrlFactory::UpdateRegistry(BOOL bRegister)
{
    // TODO: Verify that your control follows apartment-model
    // threading rules. Refer to MFC TechNote 64 for more information.
    // If your control does not conform to the apartment-model rules, then
    // you must modify the code below, changing the 6th parameter from
    // afxRegApartmentThreading to 0.
    if (bRegister)
        return AfxOleRegisterControlClass(
            AfxGetInstanceHandle(),
            m_clsid,
            m_lpszProgID,
            IDS_CALENDAR,
            IDB_CALENDAR,
            afxRegApartmentThreading,
            _dwCalendarOleMisc,
            _tlid,
            _wVerMajor,
            _wVerMinor);
    else
        return AfxOleUnregisterClass(m_clsid, m_lpszProgID);
}
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::CCalendarCtrl - Constructor
CCalendarCtrl::CCalendarCtrl()
{
    InitializeIIDs(&IID_DCalendar, &IID_DCalendarEvents);

    CTime time = CTime::GetCurrentTime ();
    m_nYear = time.GetYear ();
```

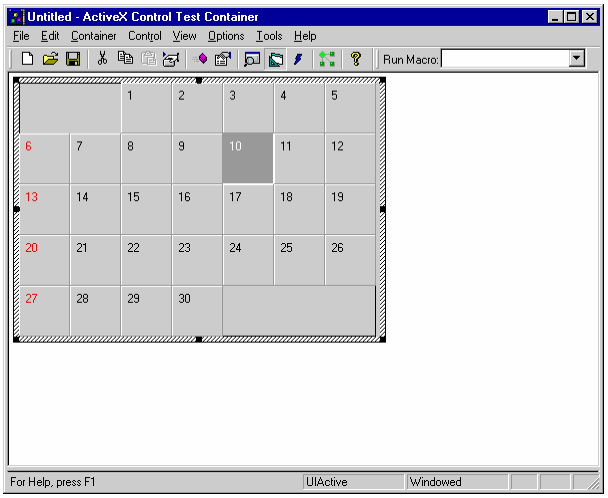
```
    m_nMonth = time.GetMonth ();
    m_nDay = time.GetDay ();
}
////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::~CCalendarCtrl - Destructor
CCalendarCtrl::~CCalendarCtrl()
{
    // TODO: Cleanup your control's instance data here.
}
////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::OnDraw - Drawing function
void CCalendarCtrl::OnDraw(
            CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
    //
    // Paint the control's background.
    //
    CBrush brush (TranslateColor (GetBackColor ()));
    pdc->FillRect (rcBounds, &brush);
    //
    // Compute the number of days in the month, which day of the week
    // the first of the month falls on, and other information needed to
    // draw the calendar.
    //
    int nNumberOfDays = m_nDaysPerMonth[m_nMonth - 1];
    if (m_nMonth == 2 && LeapYear (m_nYear))
        nNumberOfDays++;
    CTime time (m_nYear, m_nMonth, 1, 12, 0, 0);
    int nFirstDayOfMonth = time.GetDayOfWeek ();
    int nNumberOfRows = (nNumberOfDays + nFirstDayOfMonth + 5) / 7;

    int nCellWidth = rcBounds.Width () / 7;
    int nCellHeight = rcBounds.Height () / nNumberOfRows;

    int cx = rcBounds.left;
    int cy = rcBounds.top;
    //
    // Draw the calendar rectangle.
    //
    CPen* pOldPen = (CPen*) pdc->SelectStockObject (BLACK_PEN);
    CBrush* pOldBrush = (CBrush*) pdc->SelectStockObject (NULL_BRUSH);

    pdc->Rectangle (rcBounds.left, rcBounds.top,
        rcBounds.left + (7 * nCellWidth),
        rcBounds.top + (nNumberOfRows * nCellHeight));
    //
    // Draw rectangles representing the days of the month.
    //
    CFont font;
    font.CreatePointFont (80, _T ("MS Sans Serif"));
    CFont* pOldFont = pdc->SelectObject (&font);

    COLORREF clrOldTextColor = pdc->SetTextColor (RGB (0, 0, 0));
    int nOldBkMode = pdc->SetBkMode (TRANSPARENT);
```

```
    for (int i=0; i<nNumberOfDays; i++) {
        int nGridIndex = i + nFirstDayOfMonth - 1;
        int x = ((nGridIndex % 7) * nCellWidth) + cx;
        int y = ((nGridIndex / 7) * nCellHeight) + cy;
        CRect rect (x, y, x + nCellWidth, y + nCellHeight);

        if (i != m_nDay - 1) {
            pdc->Draw3dRect (rect, RGB (255, 255, 255),
                RGB (128, 128, 128));
            pdc->SetTextColor (RGB (0, 0, 0));
        }
        else {
            pdc->SelectStockObject (NULL_PEN);
            pdc->SelectStockObject (GRAY_BRUSH);
            pdc->Rectangle (rect);
            pdc->Draw3dRect (rect, RGB (128, 128, 128),
                RGB (255, 255, 255));
            pdc->SetTextColor (RGB (255, 255, 255));
        }
        CString string;
        string.Format (_T ("%d"), i + 1);
        rect.DeflateRect (nCellWidth / 8, nCellHeight / 8);

        if (m_bRedSundays && nGridIndex % 7 == 0)
            pdc->SetTextColor (RGB (255, 0, 0));

        pdc->DrawText (string, rect, DT_SINGLELINE œ DT_LEFT œ DT_TOP);
    }
    //
    // Clean up and exit.
    //
    pdc->SetBkMode (nOldBkMode);
    pdc->SetTextColor (clrOldTextColor);
    pdc->SelectObject (pOldFont);
    pdc->SelectObject (pOldBrush);
    pdc->SelectObject (pOldPen);
}
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::DoPropExchange - Persistence support
void CCalendarCtrl::DoPropExchange(CPropExchange* pPX)
{
    ExchangeVersion(pPX, MAKELONG(_wVerMinor, _wVerMajor));
    COleControl::DoPropExchange(pPX);
    PX_Bool (pPX, _T ("RedSundays"), m_bRedSundays, TRUE);
}
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::OnResetState - Reset control to default state
void CCalendarCtrl::OnResetState()
{
    COleControl::OnResetState(); // Resets defaults found in DoPropExchange

    // TODO: Reset any other control state here.
}
////////////////////////////////////////////////////////////////////////////
// CCalendarCtrl::AboutBox - Display an "About" box to the user
```

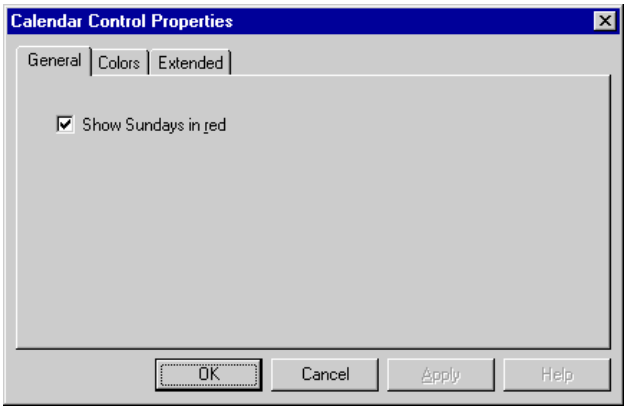
```
void CCalendarCtrl::AboutBox()
{
    CDialog dlgAbout(IDD_ABOUTBOX_CALENDAR);
    dlgAbout.DoModal();
}
////////////////////////////////////////////////////////////////////////
// CCalendarCtrl message handlers
BOOL CCalendarCtrl::LeapYear(int nYear)
{
    return (nYear % 4 == 0) ^ (nYear % 400 == 0) ^ (nYear % 100 == 0);
}
void CCalendarCtrl::OnRedSundaysChanged()
{
    InvalidateControl ();
    SetModifiedFlag();
}
DATE CCalendarCtrl::GetDate()
{
    COleDateTime date (m_nYear, m_nMonth, m_nDay, 12, 0, 0);
    return (DATE) date;
}
BOOL CCalendarCtrl::SetDate(short nYear, short nMonth, short nDay)
{
    //
    // Make sure the input date is valid.
    //
    if (nYear < 1970 œœ nYear > 2037)
        return FALSE;

    if (nMonth < 1 œœ nMonth > 12)
        return FALSE;

    int nNumberOfDays = m_nDaysPerMonth[m_nMonth - 1];
    if (nMonth == 2 && LeapYear (nYear))
        nNumberOfDays++;

    if (nDay < 1 œœ nDay > nNumberOfDays)
        return FALSE;
    //
    // Update the date, repaint the control, and fire a NewDay event.
    //
    m_nYear = nYear;
    m_nMonth = nMonth;
    m_nDay = nDay;
    InvalidateControl ();
    return TRUE;
}
void CCalendarCtrl::OnLButtonDown(UINT nFlags, CPoint point)
{
    int nNumberOfDays = m_nDaysPerMonth[m_nMonth - 1];
    if (m_nMonth == 2 && LeapYear (m_nYear))
        nNumberOfDays++;

    CTime time (m_nYear, m_nMonth, 1, 12, 0, 0);
    int nFirstDayOfMonth = time.GetDayOfWeek ();
```

```
    int nNumberOfRows = (nNumberOfDays + nFirstDayOfMonth + 5) / 7;

    CRect rcClient;
    GetClientRect (&rcClient);
    int nCellWidth = rcClient.Width () / 7;
    int nCellHeight = rcClient.Height () / nNumberOfRows;

    for (int i=0; i<nNumberOfDays; i++) {
        int nGridIndex = i + nFirstDayOfMonth - 1;
        int x = rcClient.left + (nGridIndex % 7) * nCellWidth;
        int y = rcClient.top + (nGridIndex / 7) * nCellHeight;
        CRect rect (x, y, x + nCellWidth, y + nCellHeight);

        if (rect.PtInRect (point)) {
            m_nDay = i + 1;
            FireNewDay (m_nDay);
            InvalidateControl ();
        }
    }
    COleControl::OnLButtonDown(nFlags, point);
}
```

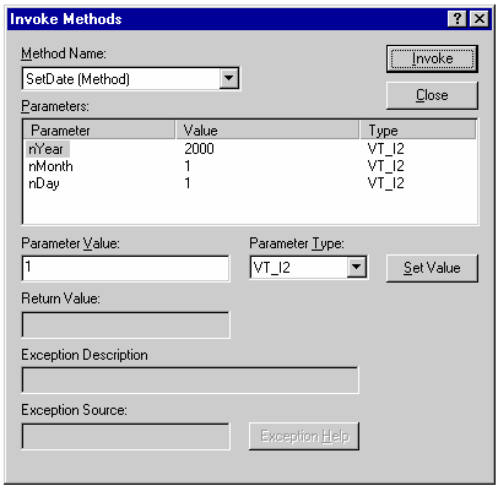
Màn hình kết quả:



và:



và:



Đăng ký OCX:

```
Regsvr32 Calendar.ocx
```

Hủy đăng ký:

```
Regsvr32 /U Calendar.ocx
```

CHƯƠNG 6. BÀI THỰC HÀNH

6.1 Bài thực hành 1

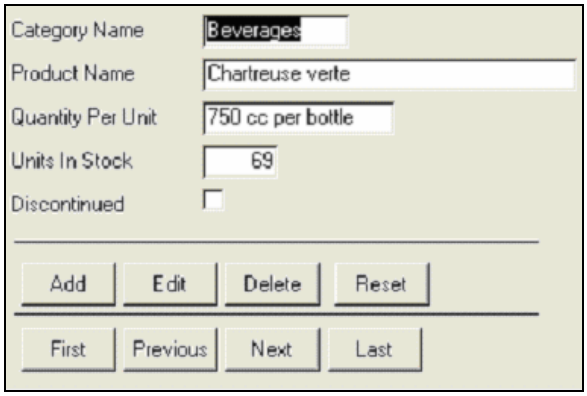
- Yêu cầu: Tìm hiểu môi trường làm việc VC++ và tạo ứng dụng đơn giản.
  a. Tìm hiểu các thành phần của Visual C++.
  b. Dùng MFC App Framework tạo chương trình ứng dụng theo các kiểu: Dialog (tên ExDlg), SDI (tên ExSDI) và MDI (tên ExMDI) (*ExSDI và ExMDI có lớp view cơ sở ở bước 6 là CFormView*).
  c. Lần lượt biên dịch ExDlg, ExSDI và ExMDI với chế độ Release và Debug, quan sát file EXE kết quả (về kích thước) và cho nhận xét.
  d. Mở ứng dụ ExSDI, thêm 1 button (đặt ID là ***IDC_BUTTON_RED*** và caption là ***Mau Do***) sao cho khi bấm vào button này, chương trình sẽ hiện ra thông báo "Nut Mau Do" bởi việc gọi hàm AfxMessageBox.
  e. Làm tương tự(d) cho button ***Mau Xanh***.
  f. Thêm 1 hàm tên ***MyCommonFuntion(BOOL bType)*** kiểu ***void*** để sử dụng chung cho câu (d) và (e) cùng gọi hàm này để thực hiện các công việc (nhằm tránh việc trùng lắp code ở câu d và e)
  g. Thêm 1 editbox (đặt ID là IDC_EDIT_TEXT) và tạo 1 biến liên kết với editbox này. [*HD: Bấm vào menu View → ClassWizard →Member Variables, sau đó chọn ID của editbox này và nhấn button "Add Variable"*] có tên là ***m_szMessage***.
  h. Sửa lại hàm ở (f) sao cho khi bấm vào nút ***Mau Do*** (hay **Mau Xanh**) thông báo hiện lên có dạng "Ban nhap noi dung: xxx, va bam nut Mau Do" hay "Ban nhap noi dung: xxx, va bam nut Mau Xanh" (*với xxx là nội dung có trong editbox ở câu g do người dùng nhập vào)*
  i. Thêm 1 static textbox(đặt ID là IDC_STATIC_MSGSHOW), tạo 1 biến liên kết với static textbox này [làm tương tự như(g)] có tên là m_szMsgShow.
  j. Xử lý tương tự như (h) nhưng không hiện thông báo mà sẽ hiện nội dung này trong static textbox vừa thêm ở (i).

6.2 Bài thực hành 2

- Yêu cầu: Xử lý xuất nội dung, vẽ hình, xử lý thao tác từ bàn phím, chuột và xuất nhập với file.
  a. Tạo mới ứng dụng ExSDI có lớp view cơ sở ở bước 6 là CView
  b. Xuất 1 chuỗi "MSSV: xxx, TENSV: yyy" tại toạ độ (10,10) và vẽ 1 hình elip tuỳ ý không có màu nền *(HD: thao tác trong hàm OnDraw trong lớp View)*
  c. Xử lý vẽ hình con lật đật đứng yên có kích thước và màu nền tuỳ ý tại toạ độ (200, 200)
  d. Xử lý vẽ con lật đật tại vị trí bấm chuột phải bất kỳ (trong vùng client của cửa sổ).
  e. Mở rộng câu d bằng ách xử lý dời con lật đật theo quỹ đạo hình tròn bán kính 20pixel.
  f. Xử lý vẽ đường tự do khi bấm rê chuột trái và phím Control (trong vùng client của cửa sổ).
  g. Hiển thị nội dung ký tự (bao gồm mẫu tự, ký số và các ký hiệu hiển thị được) ở góc phải trên trong vùng client của cửa sổ.
  h. Lập lại tương tự các câu từ (c) đến (e) cho ngôi sao 6 cánh.

6.3 Bài thực hành 3

- Yêu cầu: Tạo và sử dụng các điều khiển Windows và sử dụng hộp thoại (File, Color, Font, Print/Print ...)
  a. Tạo mới ứng dụng ExDlg có lớp view cơ sở ở bước 6 là CFormView
  b. Tạo 1 menu có item "Font…", "Color…", khi nhấn vào lần lượt hiện ra các dialog cho chọn font chữ, màu.
  c. Cập nhật giao diện thành dạng như sau:



d. Tìm hiểu các thành phần View/Document của project ExSDI.

e. Kết hợp với sự hỗ trợ của các lớp CArray, hãy xử lý sao cho khi nhập thông tin và nhấn button "Add" thì thông tin này sẽ được lưu vào bộ nhớ, tương tự cho phép cập nhật, xóa hay reset khi nhấn vào các button Edit/Delete hay Reset.

f. Xử lý tương tự với các button First/Previous/Next/Last cho phép hiển thị các mẫu tin mong muốn.

g. Thêm 1 editbox (đặt ID là IDC_EDIT_NUMBER)

h. Cập nhật lại chương trình sao cho khi bấm di chuyển đến mẫu tin nào thì số thứ tự mẫu tin đó sẽ hiện ra tương ứng trong editbox ở câu g.

i. Lập lại các bước (e) đến (f) với CList.

6.4 Bài thực hành 4

- Yêu cầu: Phát triển từ các Window App ở bài TH3 nêu trên: mở rộng sử dụng các View class, xử lý truyền/nhận giữa Document và View, bổ sung/điều khiển Toolbar, StatusBar.

a. Mở bài thực hành số 3 (ExSDI), sau đó thêm 1 button "Save File"(đặt ID là IDC_BUTTON_SAVEFILE), và cập nhật giao diện lại giống như hình bên dưới đây với combo-box có ID là IDC_COMBOX_CATEGORY

b. Thực hiện thao tác lưu danh sách Product đang có vào file có tên "ProductList.dat" khi bấm nút "Save File"

c. Thực hiện thao tác lưu nội dung đang có trong file "ProductList.dat" vào danh sách Product khi bấm nút "Load File"

d. Thực hiện việc bẫy lỗi với cơ chế ***try…catch***, xuất thông báo lỗi(trong trường hợp có lỗi), xuất thông báo "Du lieu duoc luu xong" (trong trường hợp Add/Edit), xuất thông báo "Ban muon xoa du lieu khong?" (trong trường hợp vừa bấm Delete) và xuất thông báo "Du lieu duoc xoa xong" (sau khi thực hiện Delete xong)

e. Tạo Toolbar (nếu chưa có) hay thêm vào Toolbar hiện hành các biểu tượng tương ứng với các button đã có (Add/Edit/Delete/Save File,First/Previous/Next/Last) sao cho chúng thực thi tương tự như khi bấm các button đã có này.

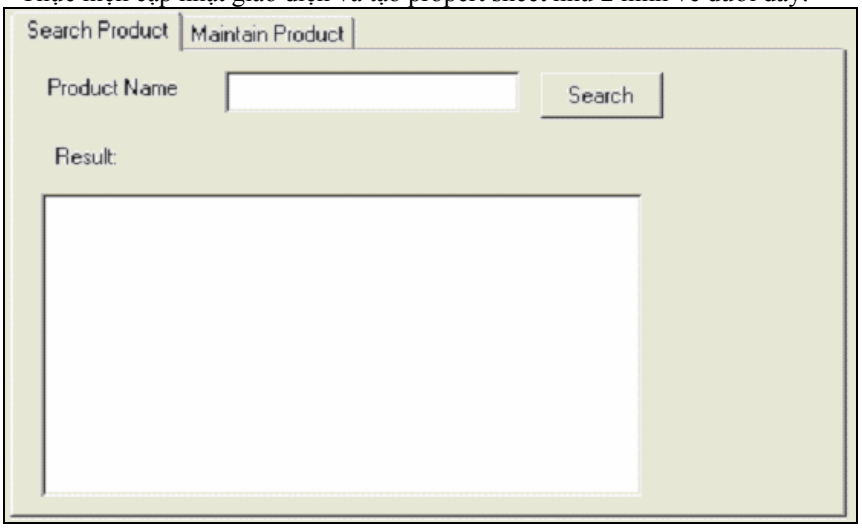
f. Thực hiện tương tự câu (c) nhưng hiện thông báo ra StatusBar phía dưới.

6.5 Bài thực hành 5

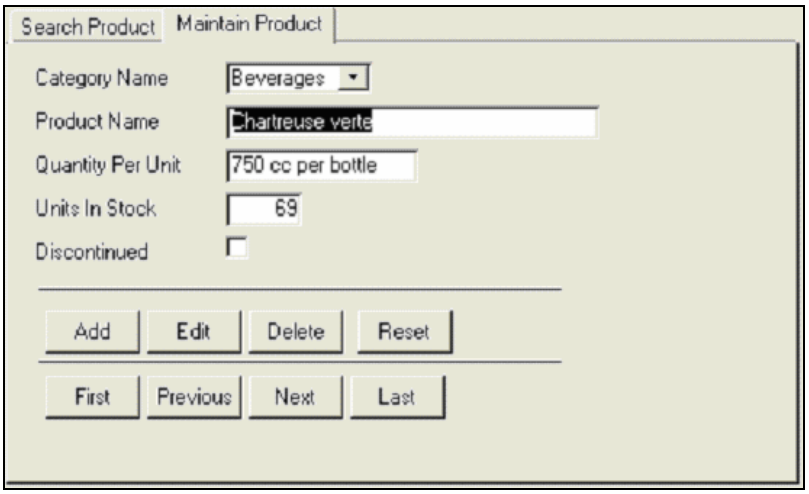
- Yêu cầu: Phát triển từ các Window App ở bài TH1 nêu trên: xây dựng hệ thống đồng hồ theo các địa điểm (timer/idle), và xử lý song hành (threads).
    a. Tạo mới ứng dụng ExSDI có lớp view cơ sở ở bước 6 là CFormView. Tạo giao diện tương tự đồng hồ điện tử với xử lý hiện thời gian trên giao diện và cả StatusBar (tương tự như đồng hồ điện tử ở taskbar)
    b. Thực hiện chương trình vẽ con lật đật tự do – có lắc lư - (dựa theo ví dụ trong phần Timer) tại vị trí bất ký, màu bất kỳ, kích thước bất kỳ với yêu cầu cung cấp 1 menu cho phép khởi tạo bộ định thời (*timer*), ngừng bộ định thời và có thể nhập 1 chu kỳ thời gian (*theo đơn vị millisecond*) tùy ý.
    c. Tìm hiểu chương trình mẫu Chat Client/Server.

6.6 Bài thực hành 6

- Yêu cầu: Xây dựng chương trình quản lý dữ liệu với Northwind database dùng ADO.
    a. Tạo mới ứng dụng ExDlg có lớp view cơ sở ở bước 6 là CFormView
    b. Thực hiện cập nhật giao diện và tạo propert sheet như 2 hình vẽ dưới đây:



trong đó editbox có ID là IDC_EDIT_ SEARCH, button có ID là IDC_BUTTON_SEARCH và combox-box có ID là IDC_LIST_ SEARCH.



và combo-box có ID là IDC_COMBO_CATEGORY

    c. Truy cập cơ sở dữ liệu Northwind (dạng Access hay SQL Server nếu có) để:
        - Trong property page (tab) "Maintain Product", thực hiện nạp dữ liệu từ bảng Categories vào combo-box(dropdown listbox) Category Name sao cho CategoryID được lưu ngầm và CategoryName hiện lên trong combo-box
        - Trong property page (tab) "Maintain Product", thực hiện tương tự câu (e) và (f) của bài 3 nhưng truy xuất trực tiếp cơ sở dữ liệu.

- Trong property page (tab) "Search Product", thực hiện việc tìm kiếm khi bấm button Search và xuất kết quả ra danh sách Result bên dưới.

6.7 Bài thực hành 7

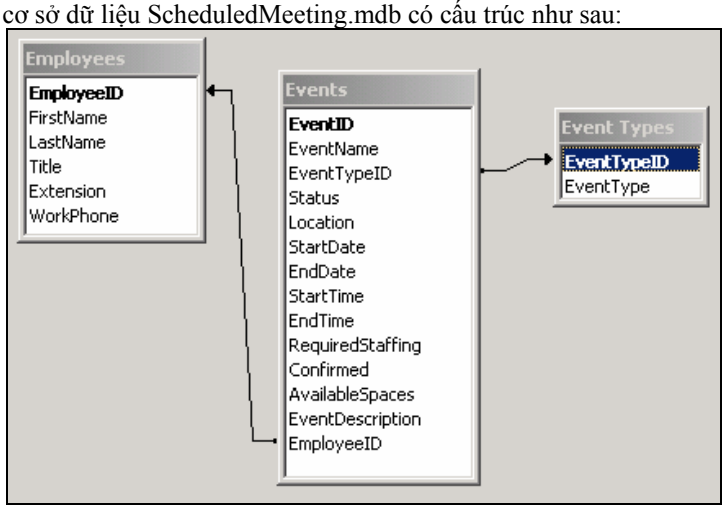
- Yêu cầu: Tạo COM object liên kết trong việc truy xuất dữ liệu.
  a. Tạo mới project có tên ExCOM và thực hiện xây dựng các hàm thư viện hỗ trợ truy cập cơ sở dữ liệu Northwind như thêm/cập nhật/xoá pProduct, tìm kiếm/lấy danh sách Product, lấy danh sách Category.
  b. Mở bài thực hành số 6 (ExSDI), thực hiện liên kết với xxxExCOM và thay thế các thao tác(i), (ii) và (iii) bởi các hàm thư viện ở câu (a).

6.8 Bài thực hành 8

- Yêu cầu: Phát triển từ các Window App ở bài TH6/TH7 nêu trên: xây dựng ActiveX control hỗ trợ các thành phần của chương trình.
  a. Xem chương trình mẫu về ActiveXControl
  b. Tạo ActiveX Control về máy tính cá nhân
  c. Xem chương trình mẫu về Windows NT Service.
  d. Tạo Windows NT Service về thông báo nhắc nhở thời gian 1000ms

6.9 Bài thực hành 9

- Yêu cầu: bài tập tổng hợp: xây dựng chương trình để thông báo lịch họp.
  a. Tạo cơ sở dữ liệu ScheduledMeeting.mdb có cấu trúc như sau:



  b. Xây dựng chương trình quản lý nội dung lịch họp của nhân viên trong 1 cơ quan (không dùng COM) có giao diện tùy chọn (*giao diện xây dựng tùy ý*)

6.10 Bài thực hành 10

- Yêu cầu: bài tập tổng hợp: xây dựng chương trình để thông báo lịch họp.
  a. Phát triển từ bài 9 nhưng dùng COM làm thư việc truy xuất dữ liệu

CHƯƠNG 7. BÀI TẬP TỔNG HỢP

7.1 Bài 1

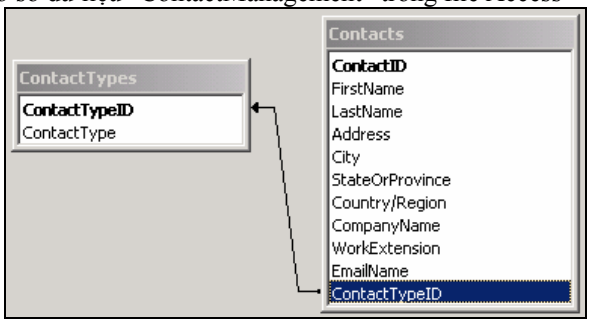
Thực hiện chương trình tương tự như ứng dụng Windows Explorer.

7.2 Bài 2

Thực hiện chương trình tương tự như ứng dụng Microsoft Internet Explorer.

7.3 Bài 3

a. Xem cấu trúc của cơ sở dữ liệu "ContactManagement" trong file Access



b. Xây dựng chương trình ứng dụng dạng SDI trong đó có menu Tools bao gồm các lệnh (menu item) theo cấu trúc sau sau:

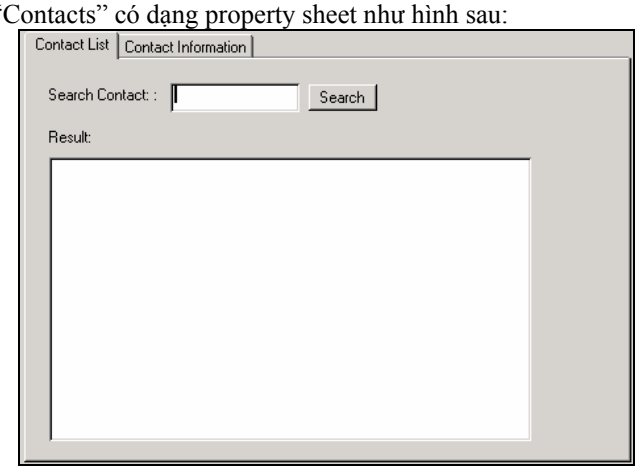
Tools

|- "Enter-View Contacts"

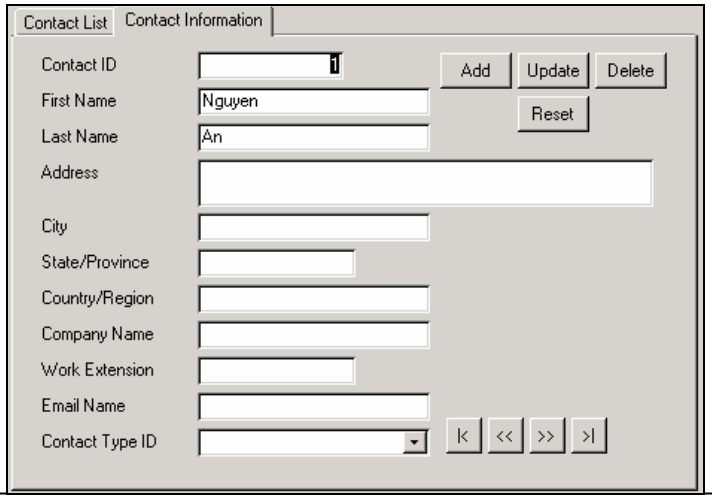
|- "Enter-View Contact Types"

sao cho khi bấm vào từng lệnh sẽ mở các form tương ứng (màn hình như ở câu 3 và 5)

c. Xây dựng form "Contacts" có dạng property sheet như hình sau:

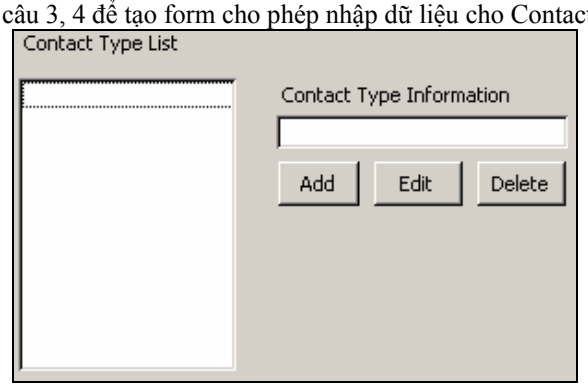


và



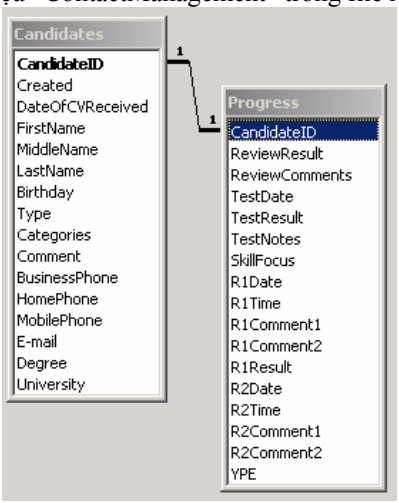
d. Thực hiện truy xuất cơ sở dữ liệu "ContactManagement" và làm các thao tác sau:
   i) Cho phép tìm kiếm dữ liệu Contact
   ii) Cho phép xem thông tin của Contact
   iii) Cho phép thêm, cập nhập, xoá contact. Cho phép reset màn hình. Cho phép thực hiện di chuyển first, last, next, previous để xem thông tin Contact.

e. Thực hiện tương tự câu 3, 4 để tạo form cho phép nhập dữ liệu cho ContactType với màn hình như sau:



7.4 Bài 4

a. Xem cấu trúc của cơ sở dữ liệu "ContactManagement" trong file Access



b. Xây dựng chương trình ứng dụng dạng SDI trong đó có menu Tools bao gồm các lệnh (menu item) theo cấu trúc sau sau:

Tools
|- "Enter-View Candidates"
|- "Enter-View Interview Progress"

c. Xây dựng chương trình quản lý các ứng viên và quá trình phỏng vấn tuyển dụng của công ty ABC (dựa theo bài tổng hợp số 3).

7.5 Bài 5

Viết chương trình MyPainBrush, tương tự như ứng dụng PainBrush của Windows.

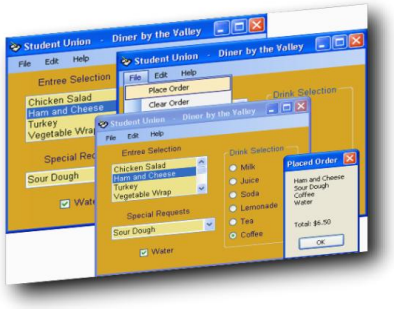
7.6 Bài 6

Viết chương trình MyGraph hỗ trợ vẽ đồ thị của một đường bậc nhất (dạng y=ax+b) hay bậc hai (dạng $y=ax^2+bx+c$), dạng y=(ax+b)/(cx+d), dạng $y=(ax^2+bx+c)/(dx+e)$ với các giá trị a, b, c, ... do người dùng nhập vào.

7.7 Bài 7

Viết chương trình MyPainBrush, tương tự như ứng dụng WordPad của Windows.

Windows Form



Nguyễn Văn Phong

Nội Dung

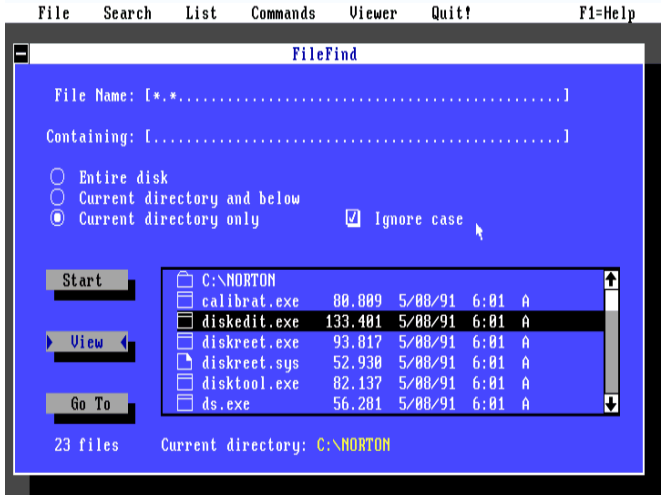
- Graphical User Interface (GUI)
- Event Driven Programming
- Ứng dụng Windows Form dùng C#
- Khuôn mẫu của ứng dụng Windows Form chuẩn
- Cách tạo ứng dụng Windows Form trong VS 2005
  - Tạo ứng dụng Form
  - Chỉnh sửa form
  - Thêm component vào form
  - Viết phần xử lý cơ bản

GUI

Command line interface: CLI

```
--- rr.chtpa.wikinedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1               /mnt/usbkey
/dev/sda2               /mnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ lsmod
Module                  Size  Used by
joydev                  8256  0
ipw2200               175112  0
ieee80211              44228  1 ipw2200
ieee80211_crypt         4872  2 ipw2200,ieee80211
e1000                  84468  0
bash-2.05b$
```

Text user interface: TUI



Tương tác qua keyboard
Thực thi tuần tự

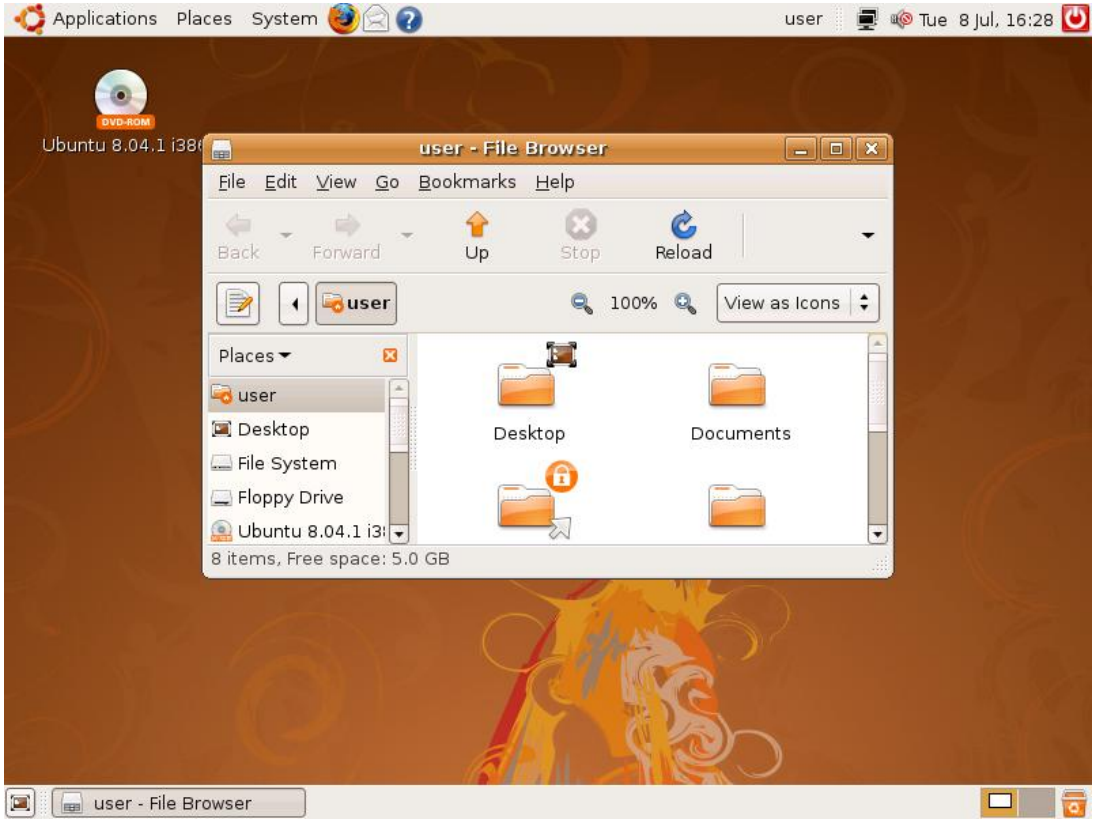
GUI dựa trên text
Mức độ tương tác cao hơn

Graphical User Interface: GUI

Tương tác qua giao diện đồ họa độ phân giải cao

Đa số các hệ OS hiện đại đều dùng GUI

Cho phép user dễ dàng thao tác



GUIs

- Chương trình hiện đại đều dùng GUI
- Graphical: text, window, menu, button…
- User: người sử dụng chương trình
- Interface: cách tương tác chương trình

- Thành phần đồ họa điển hình
  - Window: một vùng bên trong màn hình chính
  - Menu: liệt kê những chức năng
  - Button: nút lệnh cho phép click vào
  - TextBox: cho phép user nhập dữ liệu text

GUI Application

- Windows Form là nền tảng GUI cho ứng dụng desktop
  - (Ngược với Web Form ứng dụng cho Web)
  - Single Document Interface (SDI)
  - Multiple Document Interface (MDI)
- Các namespace chứa các lớp hỗ trợ GUI trong .NET
  - System.Windows.Forms:
    - Chứa GUI components/controls và form
  - System.Drawing:
    - Chức năng liên quan đến tô vẽ cho thành phần GUI
    - Cung cấp chức năng truy cập đến GDI+ cơ bản

Event- Driven Programming

Cách truyền thống

Danh sách các lệnh thực thi tuần tự

Việc kế tiếp xảy ra chính là lệnh tiếp theo trong danh sách

Chương trình được thực thi bởi máy tính

Event-Driven Programming

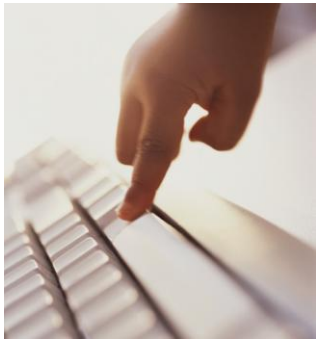
Các đối tượng có thể kích hoạt sự kiện và các đối tượng khác phản ứng với những sự kiện đó

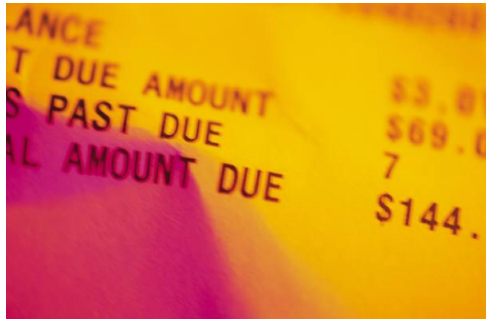
Việc kế tiếp xảy ra phụ thuộc vào sự kiện kế tiếp

Luồng chương trình được điều kiển bở sự tương tác User-Computer

Event-Driven Programming

- Chương trình GUI thường dùng Event-Drive Programming
- Chương trình chờ cho event xuất hiện và xử lý
- Ví dụ sự kiện:

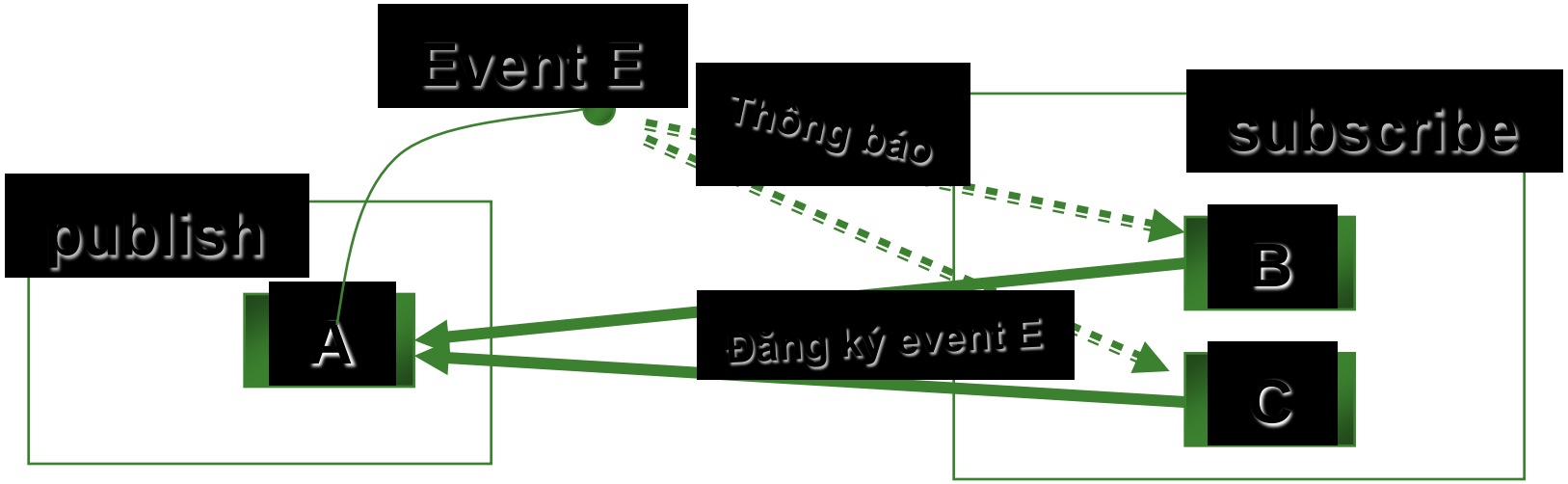




- Firing an event: khi đối tượng khởi tạo sự kiện
- Listener: đối tượng chờ cho sự kiện xuất hiện
- Event handler: phương thức phản ứng lại sự kiện

Event-Driven Programming

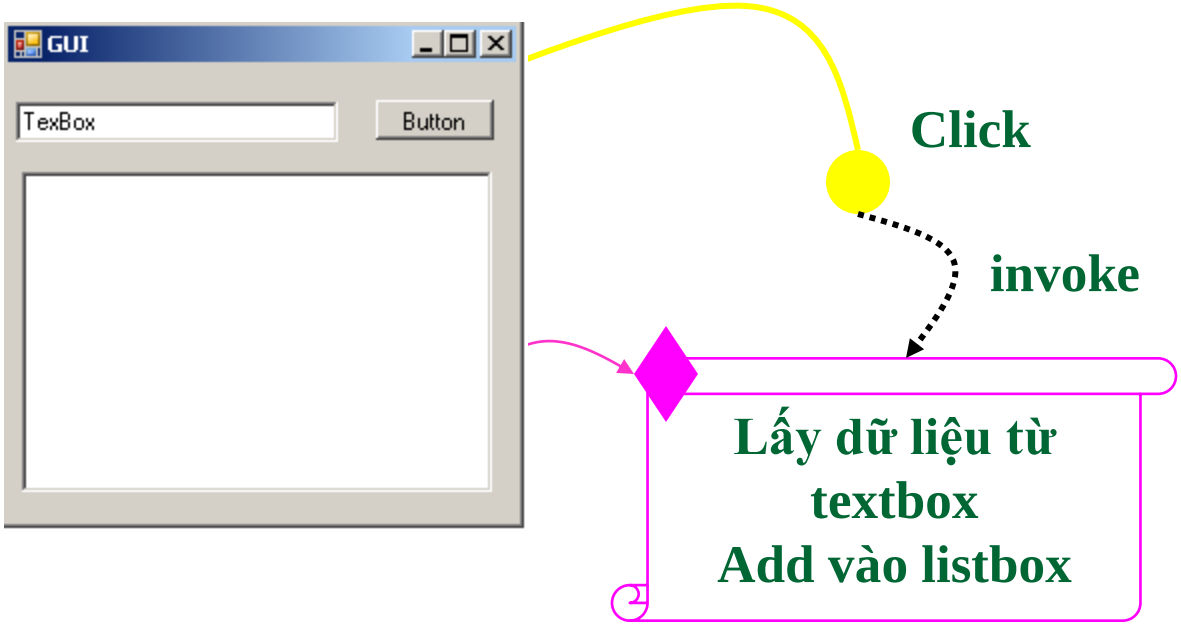
- Trong C#, Event-Driven Programming được thực thi bởi event (*xem slide Delegate & Event*)
-



Event-Driven Programming

- Minh họa xử lý trong form

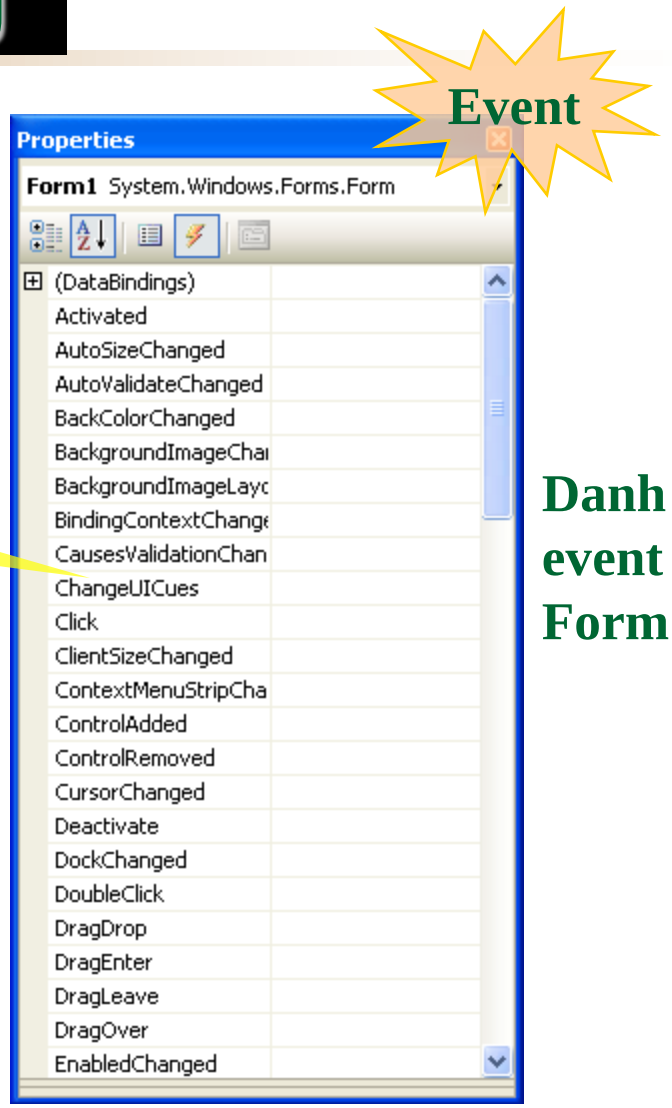
User nhập text vào texbox -> click Button để add chuỗi nhập vào listbox



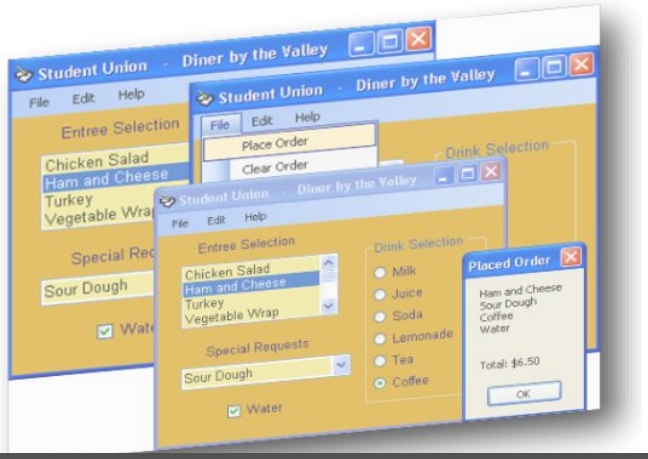
Button đưa ra sự kiện click
Form có event handler cho click của button

Event-Driven Programming

- GUI-based events
  - Mouse move
  - Mouse click
  - Mouse double-click
  - Key press
  - Button click
  - Menu selection
  - Change in focus
  - Window activation
  - …



Danh sách event cho Form



Windows Forms Application

Windows Form App

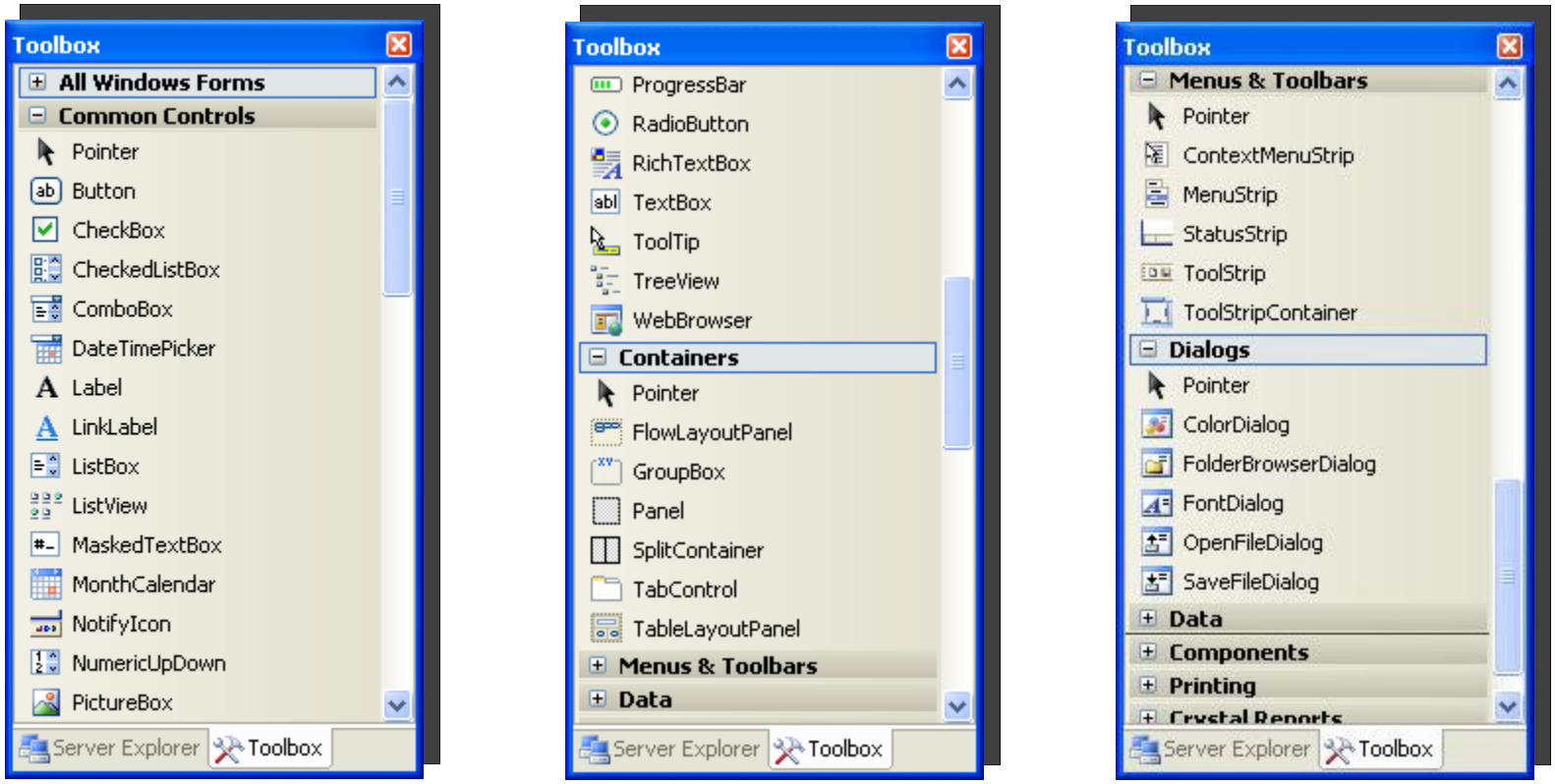
- Sử dụng GUI làm nền tảng
- Event-driven programming cho các đối tượng trên form
- Ứng dụng dựa trên một "form" chứa các thành phần
  - Menu
  - Toolbar
  - StatusBar
  - TextBox, Label, Button…
- Lớp cơ sở cho các form của ứng dụng là Form

Minh họa WinForm App

GUI Components/Controls

- Components/controls được tổ chức vào các lớp thừa kế, cho phép dễ dàng chia sẻ các thuộc tính
- Mỗi component/control định nghĩa các
  - Thuộc tính
  - Phương thức
  - Sự kiện
- Cách dễ nhất là sử dụng VS .NET Toolbox để thêm control và component vào form

Components and Controls cho Windows Form



Toolbox của Visual Studio .NET 2005

UD WinForm đơn giản

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace HaGiang
{
    public class Form1 : Form
    {
        Label title;
        public Form1()
        {
            this.Text = "Hello World!";
            this.Size = new Size(400, 200);
            title = new Label();
            title.Text = "Hello World";
            title.Location = new Point(50, 50);
            this.Controls.Add(title);
        }
        public static void Main(string[] argv)
        {
            Application.Run(new Form1());
        }
    }
}
```

Lớp Form cơ sở

Control kiểu Label

Thiết kế form & control

Add control vào form

Chạy ứng dụng với Form1 làm form chính

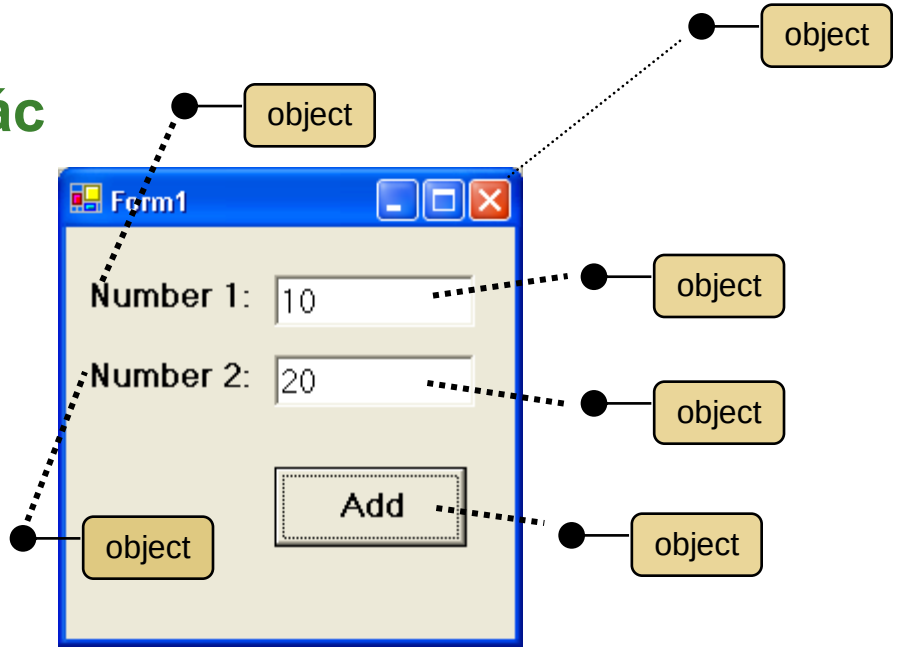
Các bước tạo UD WinForm cơ bản

- Tạo lớp kế thừa từ lớp Form cơ sở
- Bổ sung các control vào form
  - Thêm các label, menu, button, textbox…
- Thiết kế layout cho form (bố trí control)
  - Hiệu chỉnh kích thước, trình bày, giao diện cho
    - form
    - Control chứa trong form
- Viết các xử lý cho các control trên form và các xử lý khác
- Hiển thị Form
  - Thông qua lớp Application gọi phương thức Run

Nên sử dụng IDE hỗ trợ thiết kế GUI!

Form và control

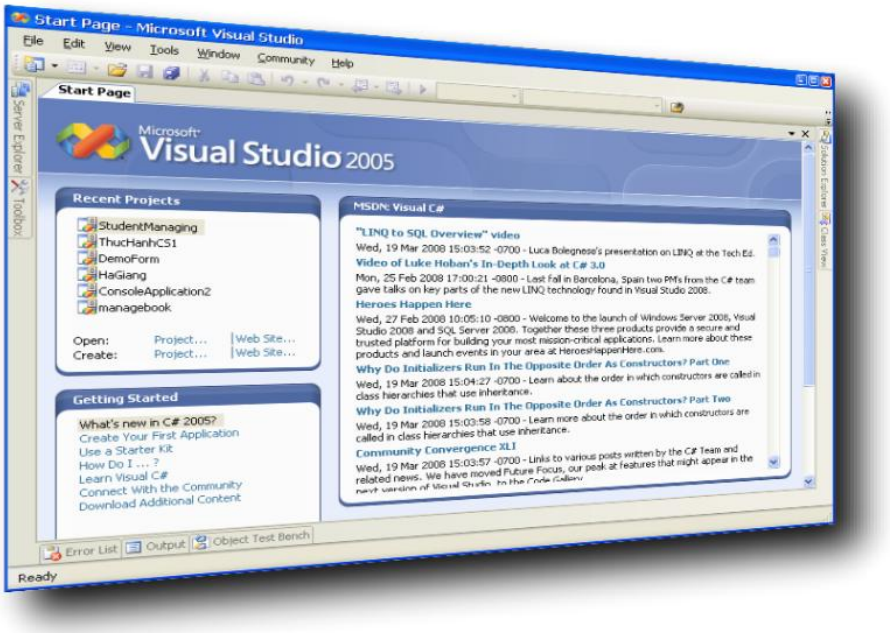
- Tất cả các thành phần trên form đều là đối tượng
- Các control là những lớp của FCL
  - System.Windows.Forms.Label
  - System.Windows.Forms.TextBox
  - System.Windows.Forms.Button
  - …
- Các control là instance của các lớp trên.



Các thuộc tính của Form

Property	Description	Default
Name	Tên của form sử dụng trong project	Form1,Form2…
AcceptButton	Thiết lập button là click khi user nhấn Enter	
CancelButton	Thiết lập button là click khi user nhấn Esc	
ControlBox	Hiển thị control box trong caption bar	True
FormBorderStyle	Biên của form: none, single, 3D, sizable	Sizable
StartPosition	Xác định vị trí xuất hiện của form trên màn hình	WindowsDefaultLocation
Text	Nội dung hiển thị trên title bar	Form1, Form2, Form3
Font	Font cho form và mặc định cho các control	
Method	**Description**	
Close	Đóng form và free resource	
Hide	ẩn form	
Show	Hiển thị form đang ẩn	
Event	**Description**	
Load	Xuất hiện trước khi form show	

Minh họa tạo ứng dụng Windows Form từ Visual Studio .NET



Tạo WinForm App từ VS. 2005

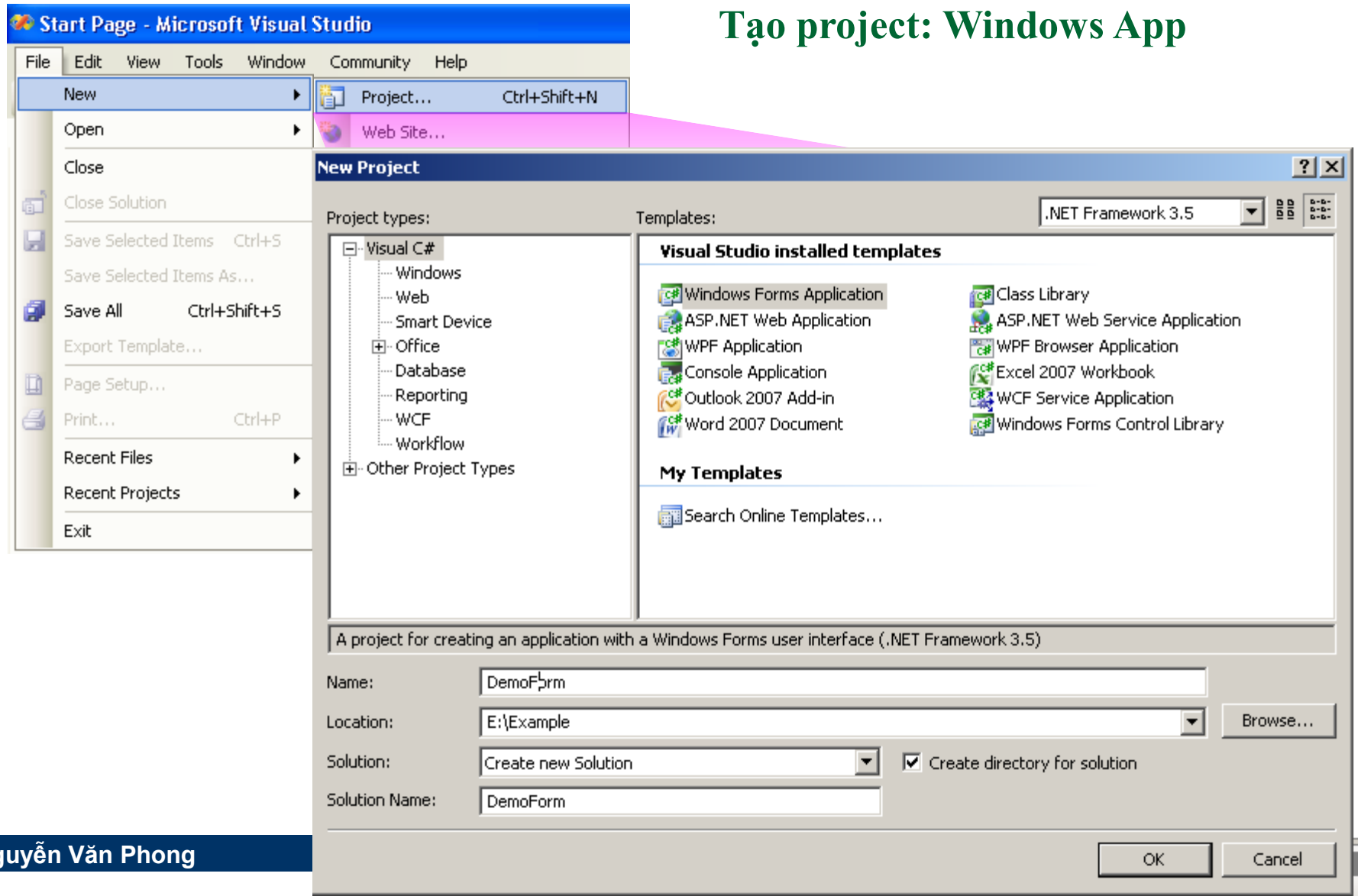
Cơ chế xử lý sự kiện code behind

Hỗ trợ WYSISYG cho GUI design

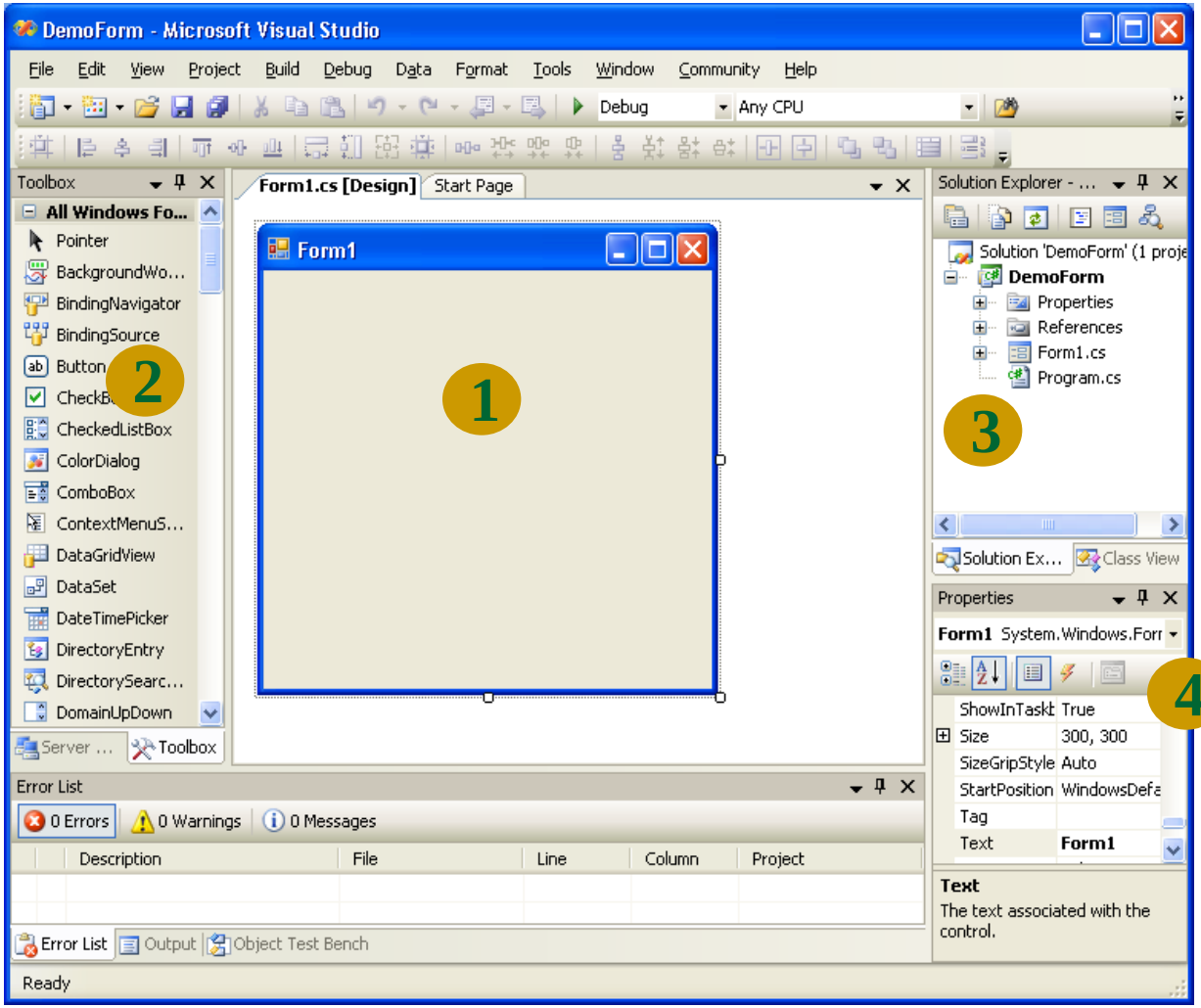
Nhanh chóng & dễ dàng tạo UD Windows Form

Tạo WinForm App từ VS. 2005 (2)

Tạo project: Windows App



Tạo WinForm App từ VS. 2005 (3)



Windows App do VS.2005 khởi tạo

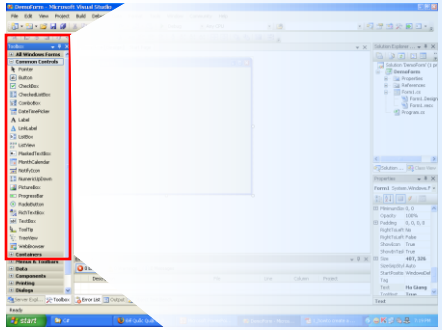
1: form ứng dụng
2: control toolbox
3: Solution Explorer
4: Form properties

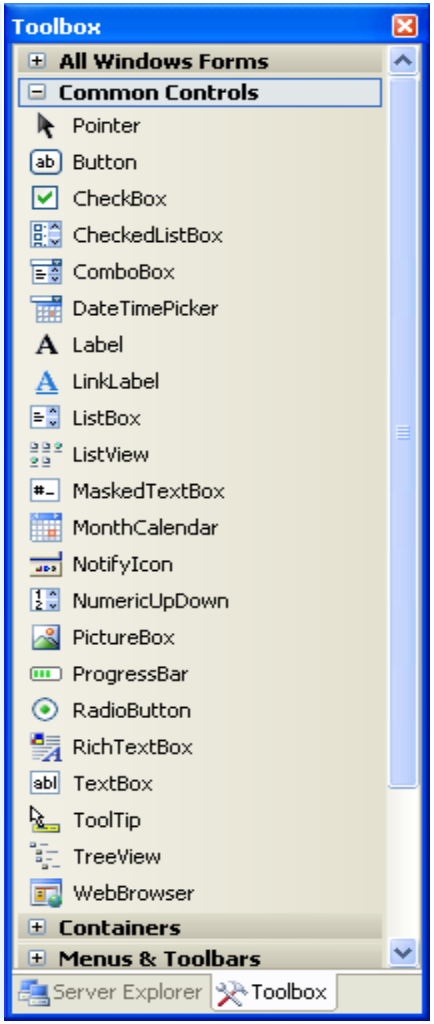
Tạo WinForm App từ VS. 2005 (4)

- Màn hình thiết kế Form, cho phép người lập trình kéo thả những control vào trong form
  - Tất cả những code được tạo tự động dựa trên sự thao tác thiết kế form của user
  - Rút ngắn nhiều thời gian cho việc thao tác giao diện form
  - Tính năng trực quan WYSIWYG

Có được ứng dụng form mặc dù chưa viết code!

Toolbox



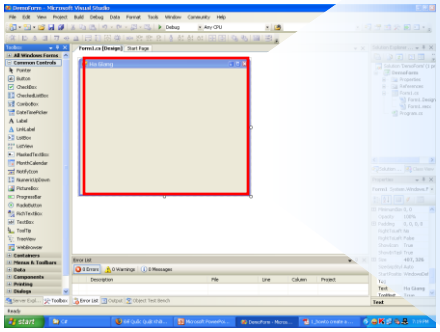


Toolbox

-Kéo thả control lên form

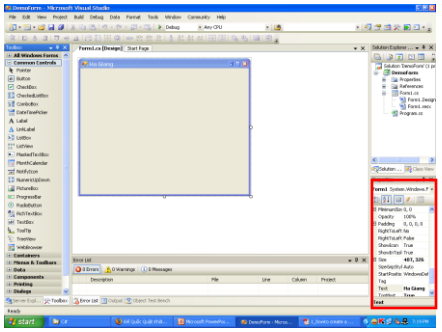
-Code được phát sinh tự động

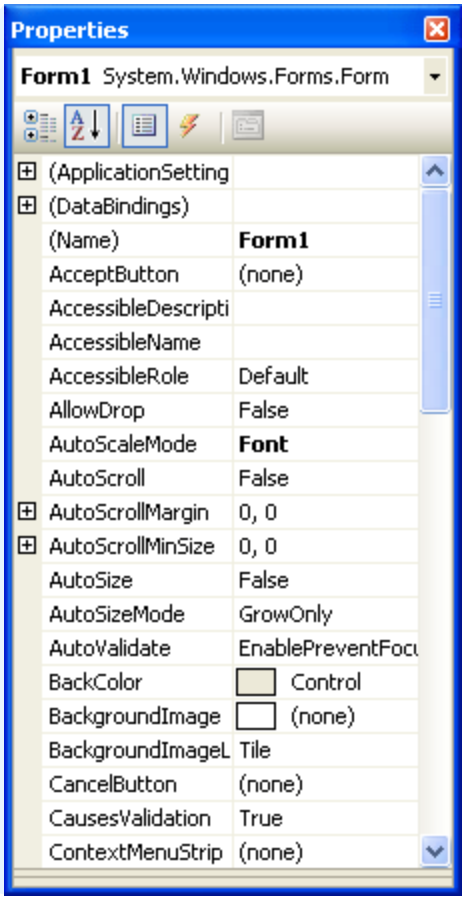
Giao diện thiết kế form



Form chính của ứng dụng

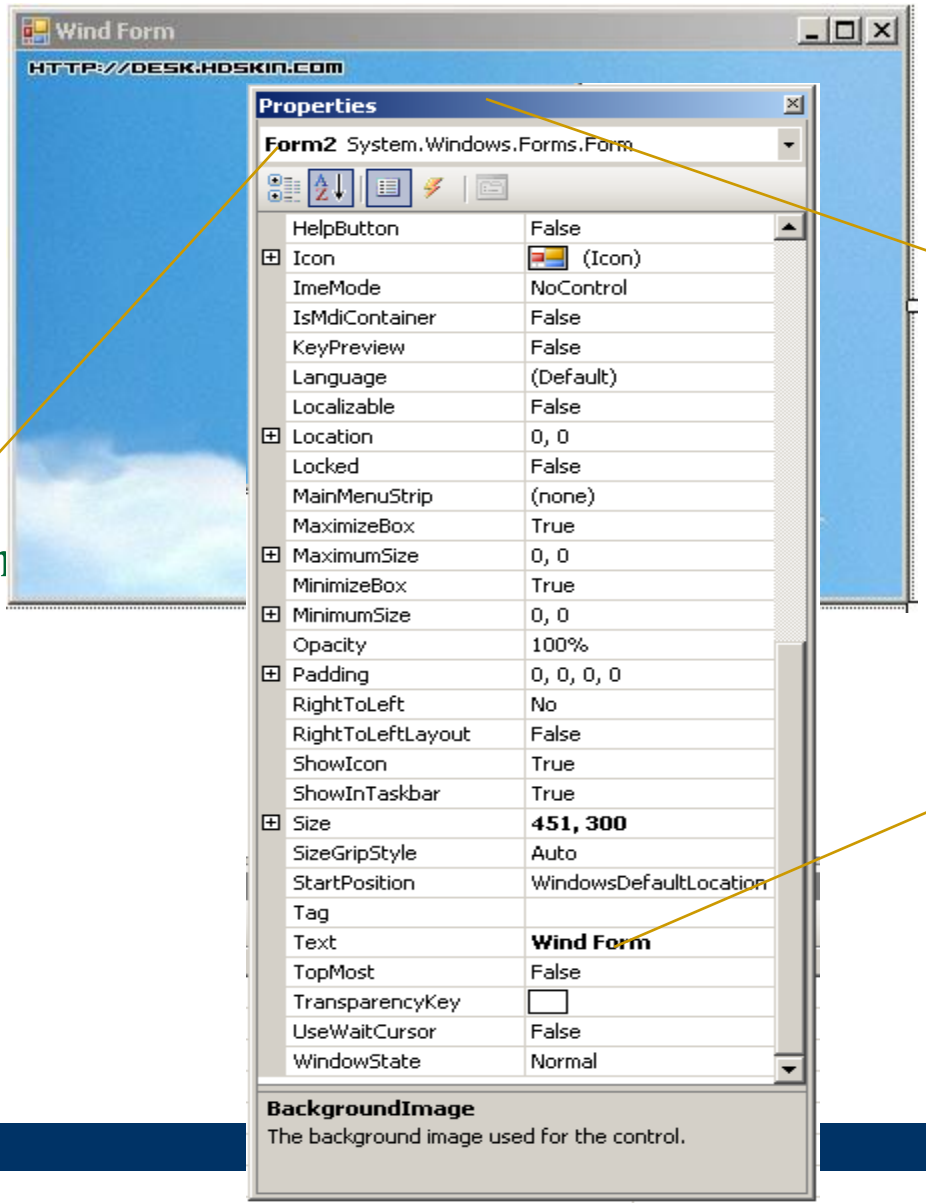
Cửa sổ properties





Cửa sổ properties của form

Cửa sổ properties



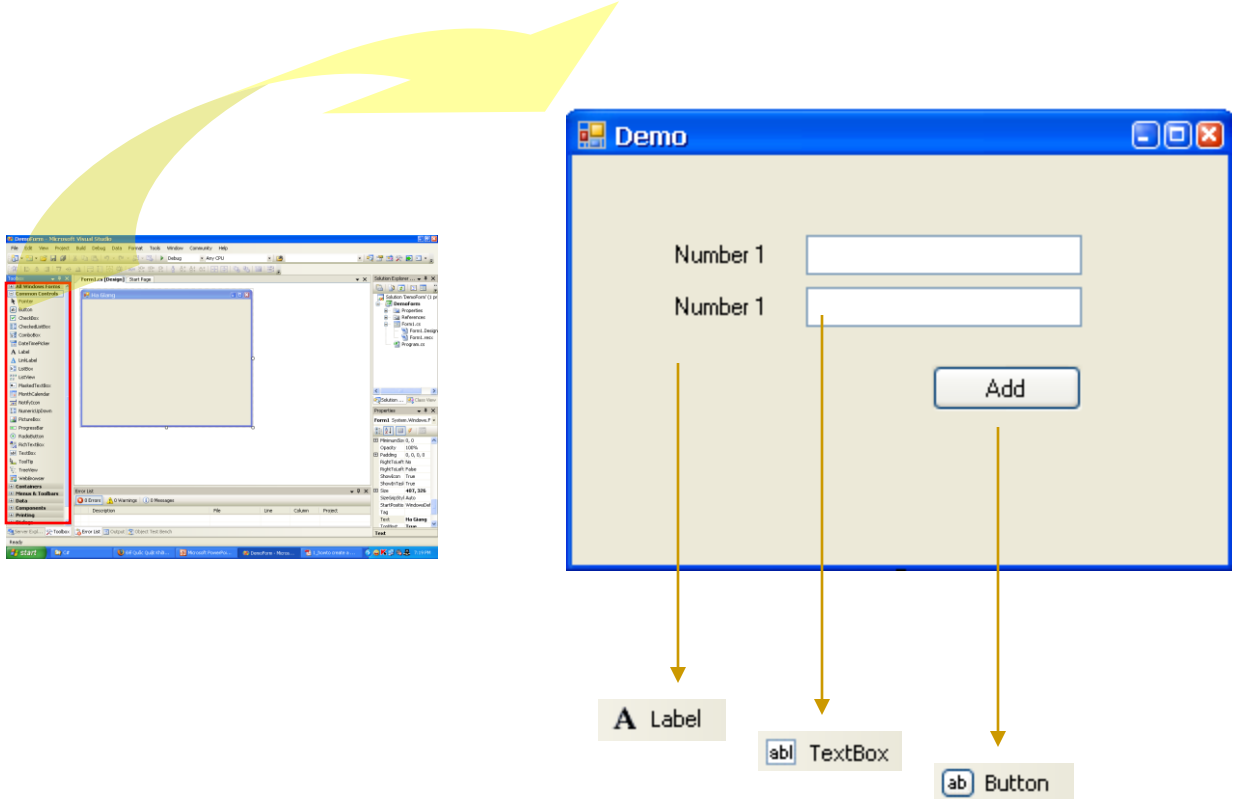
Tên của form chính là tên lớp

Dễ dàng hiệu chỉnh form thông qua cửa sổ Properties

Thay đổi title

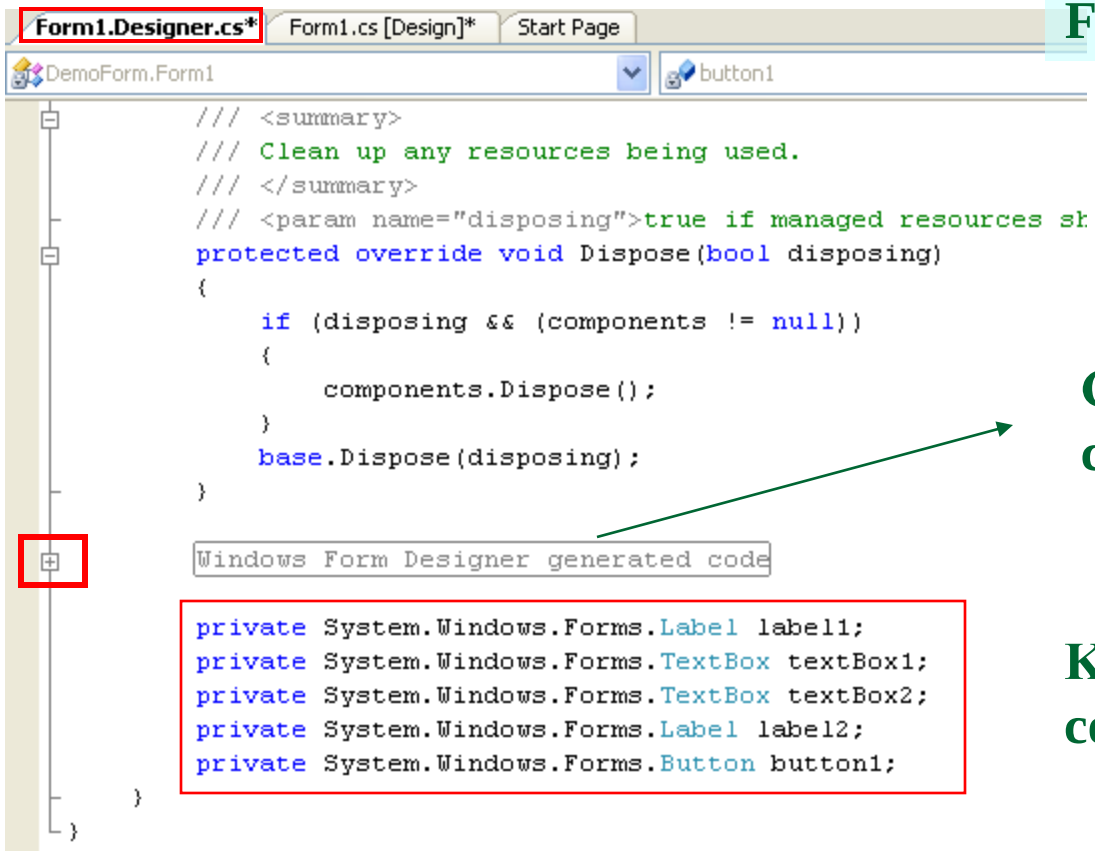
Thêm control vào form

- Kéo thả control vào form



Code của phần design

- Phần code thiết kế Form1 được tạo tự động



Form1.Designer.cs

Chứa code khởi tạo control

Khai báo các đối tượng control trên Form1

Code của phần design

InitializeComponent

Form1.Designer.cs* | Form1.cs [Design]* | Start Page

DemoForm.Form1 | button1

```
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(48, 43);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(53, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Number 1";
            //
            // textBox1
            //
```

Tạo đối tượng

Lần lượt khai báo các thuộc tính cho các control

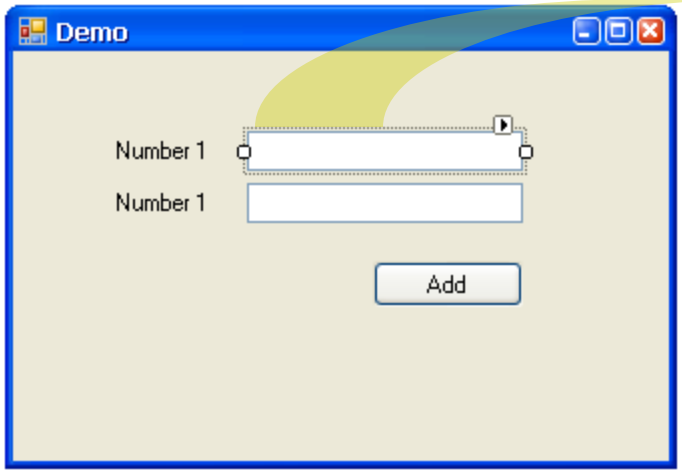
Code của phần design

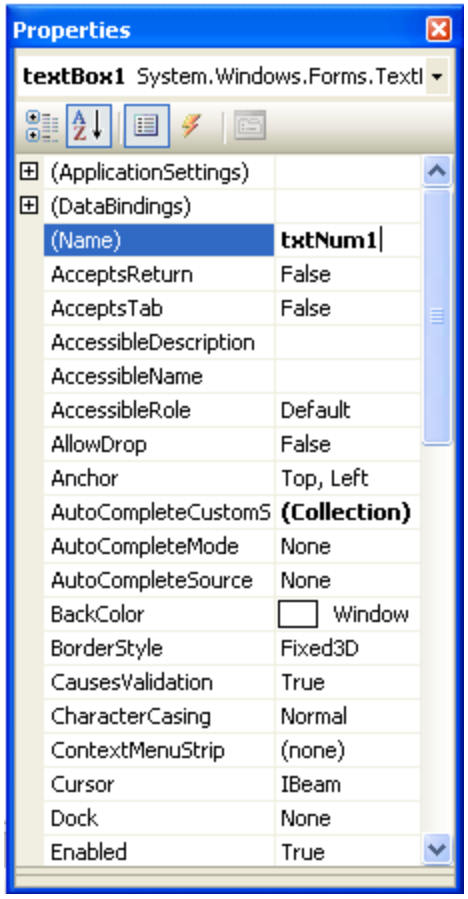
InitializeComponent

```
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(328, 205);
this.Controls.Add(this.BtnAdd);
this.Controls.Add(this.txtNum2);
this.Controls.Add(this.label2);
this.Controls.Add(this.txtNum1);
this.Controls.Add(this.label1);
this.Name = "Form1";
this.Text = "Demo";
this.TopMost = true;
this.ResumeLayout(false);
this.PerformLayout();
```

Đưa các control vào danh sách control của Form1

Sửa thuộc tính của control





Đổi tên thành txtNum1

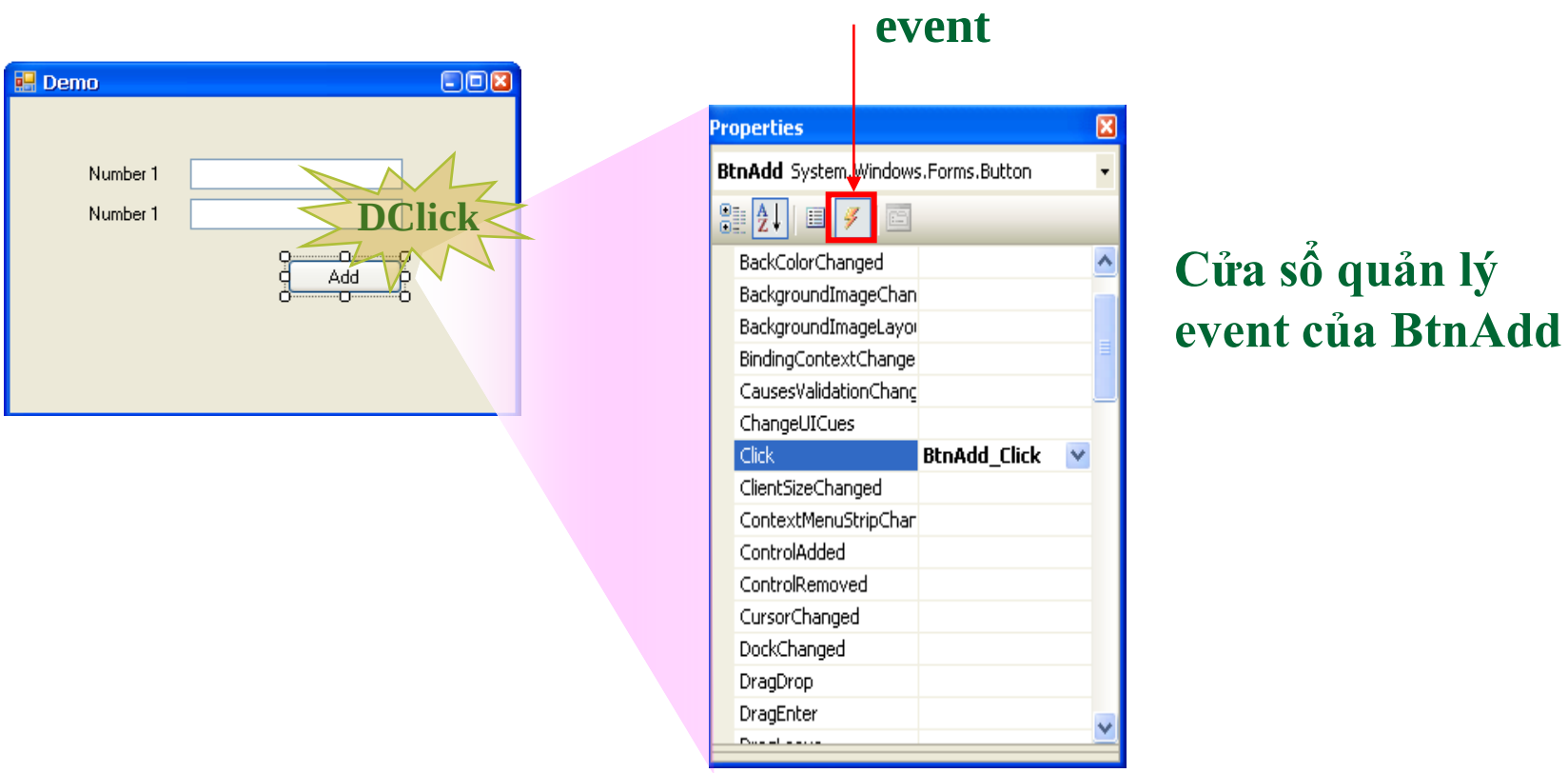
Thay đổi các giá trị qua cửa sổ properties -> VS tự cập nhật code

Phần xử lý

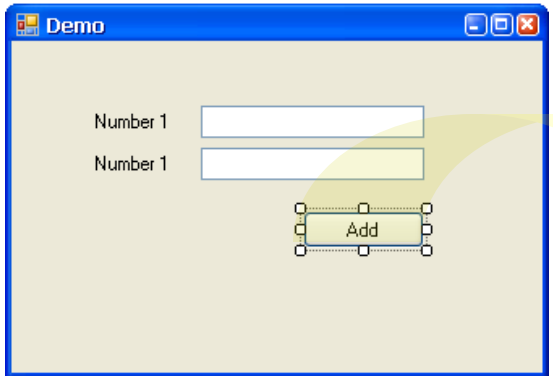
- Khi click vào Add -> cộng 2 giá trị và xuất kết quả
- Thực hiện
  - Button Add cung cấp sự kiện click
  - Form sẽ được cảnh báo khi Add được click
  - Form sẽ lấy dữ liệu từ 2 textbox và cộng -> kết quả
- Cơ chế event
  - Button đưa ra sự kiện click: đối tượng publish
  - Form quan tâm đến sự kiện click của button, Form có sẽ phần xử lý ngay khi button click.
  - Phần xử lý của form gọi là Event Handler
  - Form đóng vai trò là lớp subscribe

Khai báo event handler

- Kích đúp vào button Add trên màn hình thiết kế cho phép tạo event handler cho sự kiện này.



Khai báo event handler



Event handler cho button Add

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void BtnAdd_Click(object sender, EventArgs e)
    {
        |
    }
}
```

Cùng signature method với System.EventHandler

Khai báo event handler

InitializeComponent

```
//
// BtnAdd
//
this.BtnAdd.Location = new System.Drawing.Point(180, 105);
this.BtnAdd.Name = "BtnAdd";
this.BtnAdd.Size = new System.Drawing.Size(75, 23);
this.BtnAdd.TabIndex = 4;
this.BtnAdd.Text = "Add";
this.BtnAdd.UseVisualStyleBackColor = true;
this.BtnAdd.Click += new System.EventHandler(this.BtnAdd_Click);
```

Sự kiện click

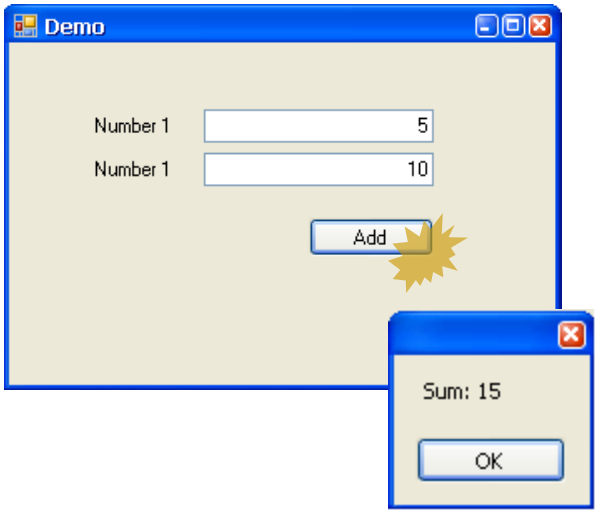
Trình xử lý được gọi khi event xảy ra

Delegate chuẩn cho event handler

Viết phần xử lý

- **Phần xử lý của Form1 khi button click**
  - **Lấy giá trị của 2 textbox, cộng kết quả và xuất ra MeesageBox**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void BtnAdd_Click(object sender, EventArgs e)
    {
        int sum;
        sum = int.Parse(txtNum1.Text) + int.Parse(txtNum2.Text);
        MessageBox.Show("Sum: " + sum.ToString());
    }
}
```

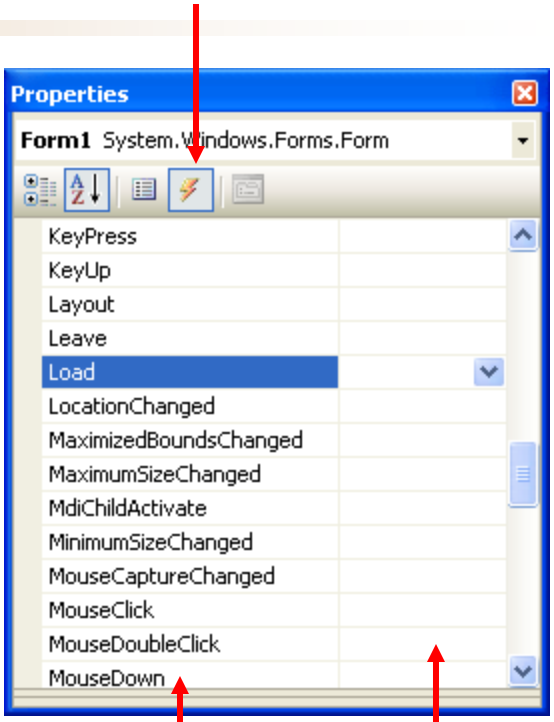


Phương thức của lớp Form

- Các hành động có thể thực hiện trên form
  - Activate: cho form nhận focus
  - Close: đóng và giải phóng resource
  - Hide: ẩn form
  - Refresh: tô vẽ lại
  - Show: cho form show ra màn hình (modeless) và activate
  - ShowDialog: hiển thị dạng modal
    - Find Dialog chính là dạng modeless
    - Font dialog dạng modal

Event của Form

- Tạo xử lý cho event
  - Trong cửa sổ properties
  - Chọn biểu tượng event
  - Kích đúp vào tên event
- Event thường dùng
  - Load: xuất hiện trước khi form xuất hiện lần đầu tiên
  - Closing: xuất hiện khi form đang chuẩn bị đóng
  - Closed: xuất hiện khi form đã đóng
  - Resize: xuất hiện sau khi user resize form
  - Click: xuất hiện khi user click lên nền form
  - KeyPress: xuất hiện khi form có focus và user nhấn phím



Tên event

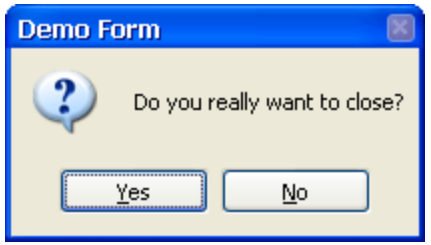
Trình xử lý nếu có

Event của Form

- Ví dụ chương trình sẽ hỏi user xác nhận trước khi đóng ứng dụng.
  - Kích đúp vào item FormClosing trong cửa sổ event
  - Hàm Form1_FormClosing được tạo và gắn với sự kiện FormClosing
  - Viết code cho event handler Form1_FormClosing

```
this.FormClosing += new FormClosingEventHandler( this.Form1_FormClosing );
```

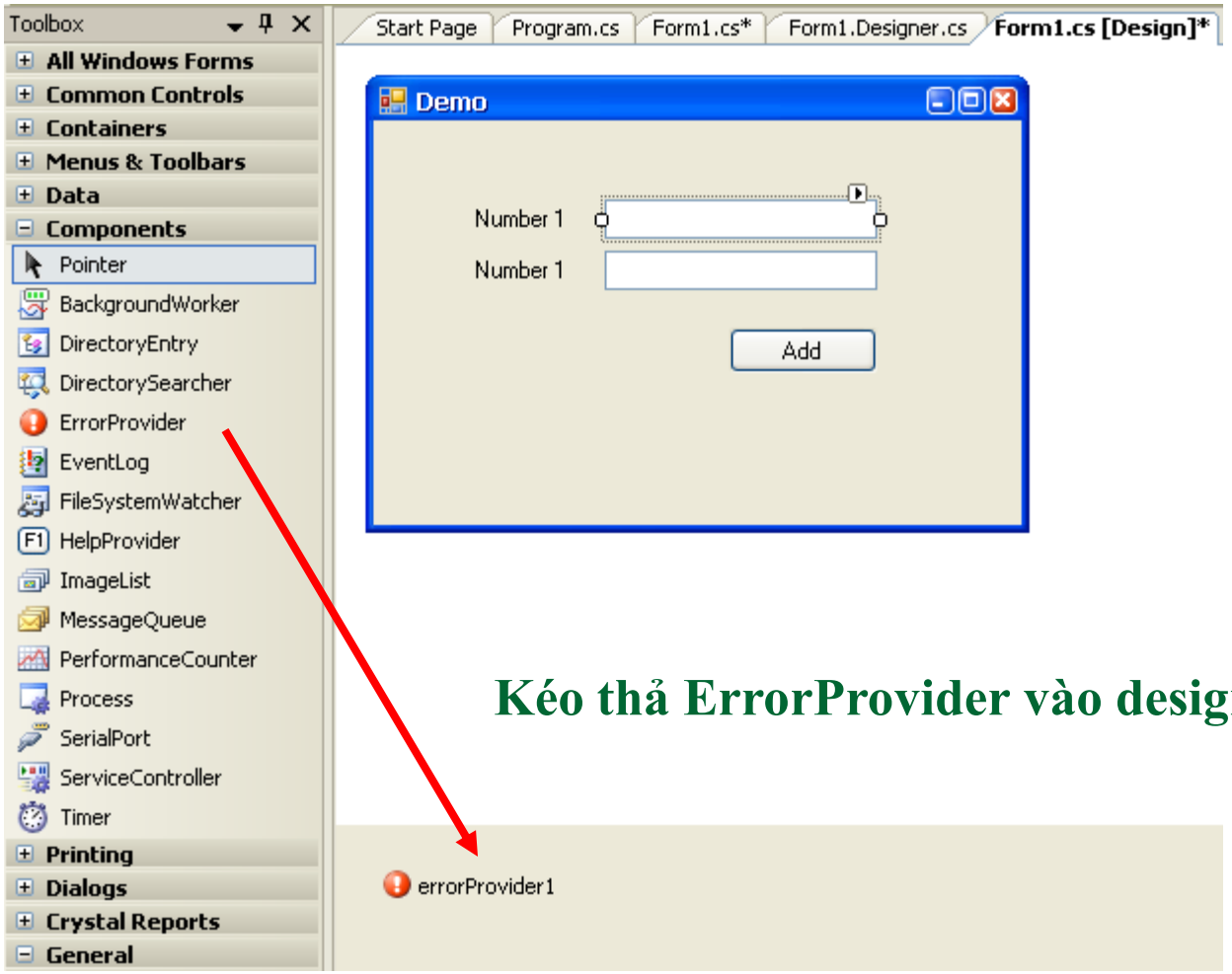
```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult r;
    r = MessageBox.Show("Do you really want to close?","Demo Form",
                        MessageBoxButtons.YesNo,
                        MessageBoxIcon.Question,
                        MessageBoxDefaultButton.Button1);
    if (r == DialogResult.No)
        e.Cancel = true;

}
```



Kiểm tra dữ liệu nhập

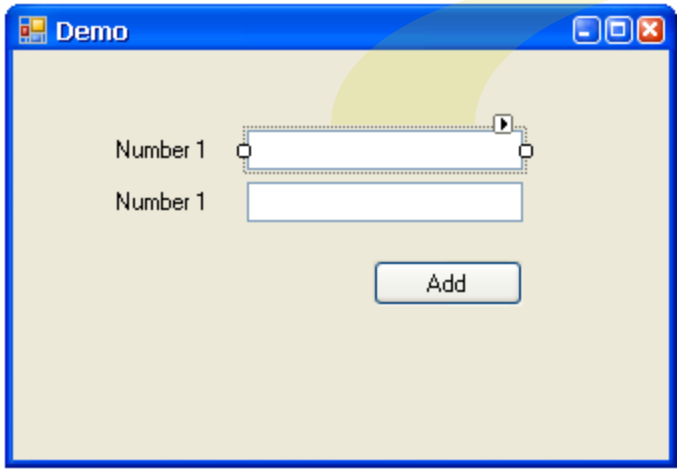
- Nếu user nhập vào chuỗi thì chương trình trên sẽ lỗi!
- Khắc phục:
  - Cảnh báo user nhập không đúng dạng
  - Xóa những ký tự không hợp lệ đó
- Sử dụng control ErrorProvider để cảnh báo lỗi khi user nhập không đúng
  - Trong Design View: kéo ErrorProvider từ ToolBox/Component vào form
  - Chặn xử lý sự kiện TextChanged khi user nhập liệu vào textbox
  - Nếu nhập sai thiết lập lỗi cho control ErrorProvider cảnh báo!

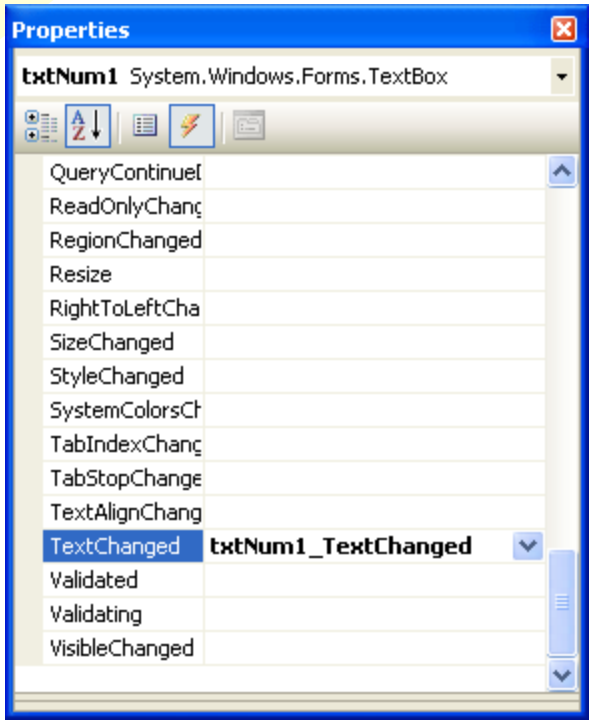
Bổ sung ErrorProvider



Kéo thả ErrorProvider vào design view

Xử lý sự kiện TextChanged của textBox

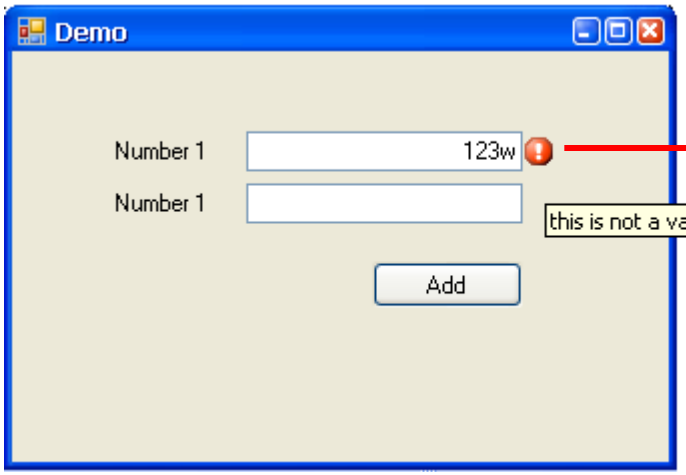





```
private void txtNum1_TextChanged(object sender, EventArgs e)
{
    Control control = (Control)sender;

    if (! Char.IsDigit(control.Text[control.Text.Length-1]))
        this.errorProvider1.SetError(control, "this is not a valid number");
    else
        this.errorProvider1.Clear();
}
```

Phần kiểm tra

ErrorProvider cảnh báo



Icon hiển thị lỗi

Di chuyển chuột vào icon, tooltip xuất hiện

Tóm tắt

- Tổng quan lập trình GUI
- Cơ chế Event Driven Programming
- Ứng dụng Windows Form cơ bản
- Sử dụng Visual Studio .NET 2005 tạo ứng dụng WF
  - Windows Form Application
  - Sử dụng control: text, label, button
  - Xử lý sự kiện cho button, form
  - Sử dụng ErrorProvider

Q&A

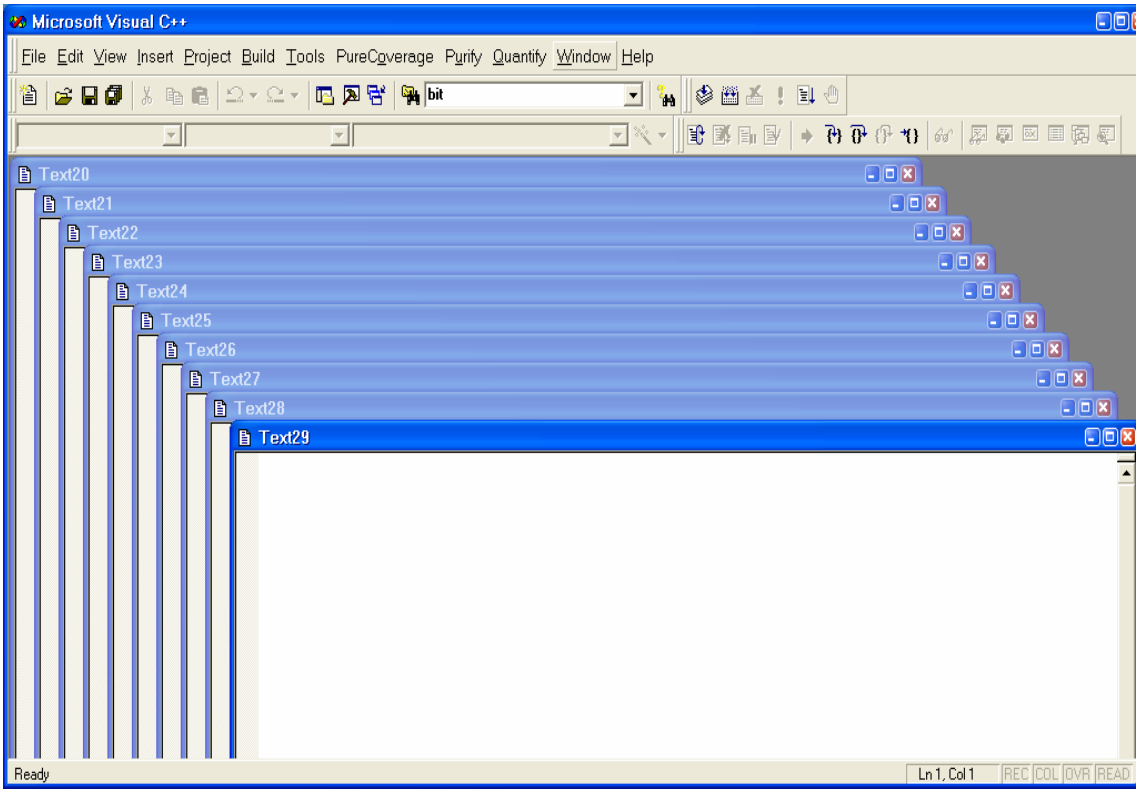
Giáo trình

Lập trình Windows

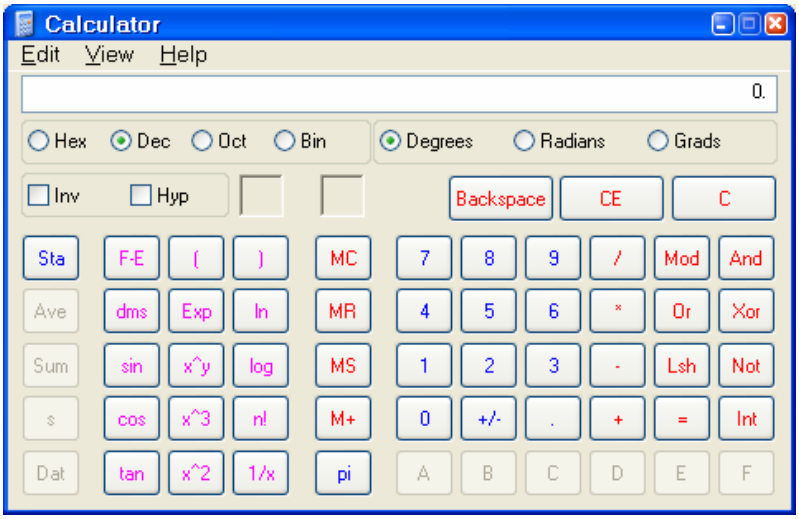
Mục lục

Trang

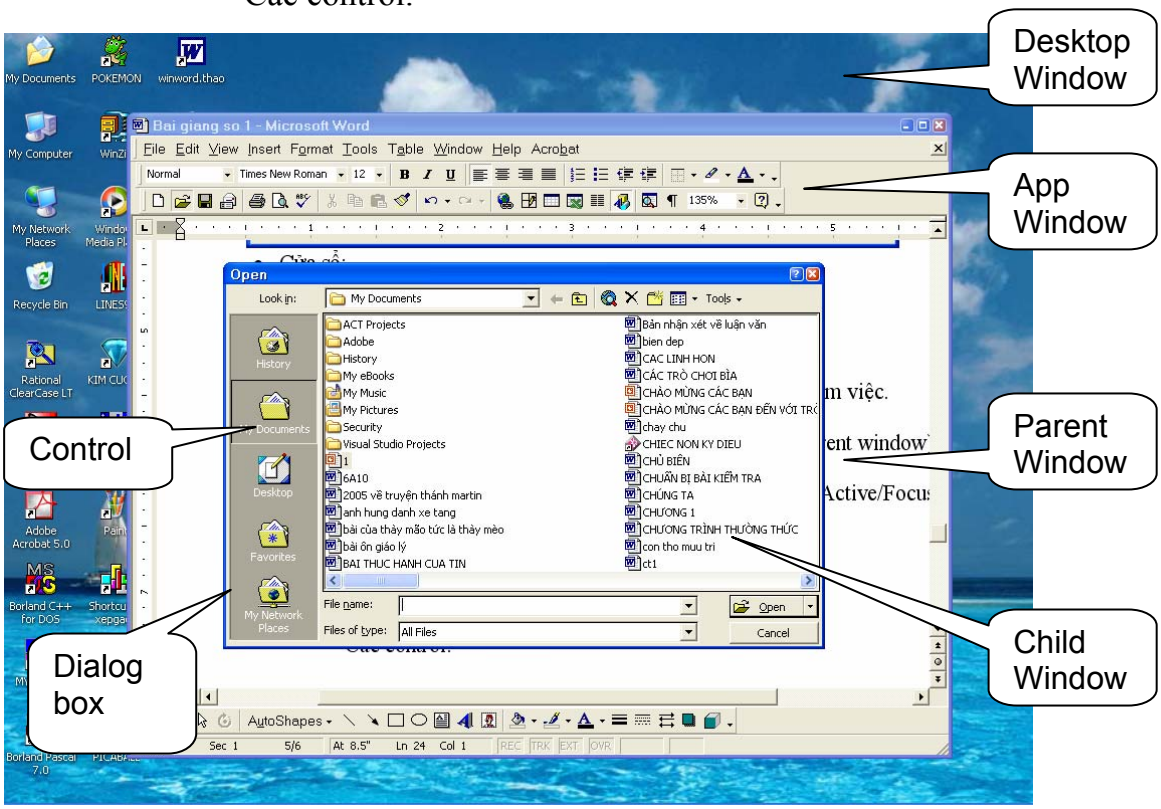
- ✓ Cho phép thay đổi kích thước cửa sổ (Resizeable).
- ✓ Cho phép Maximize/Minimize/Close các cửa sổ con.
- ✓ Ví dụ: Word, Excel, VC++,…

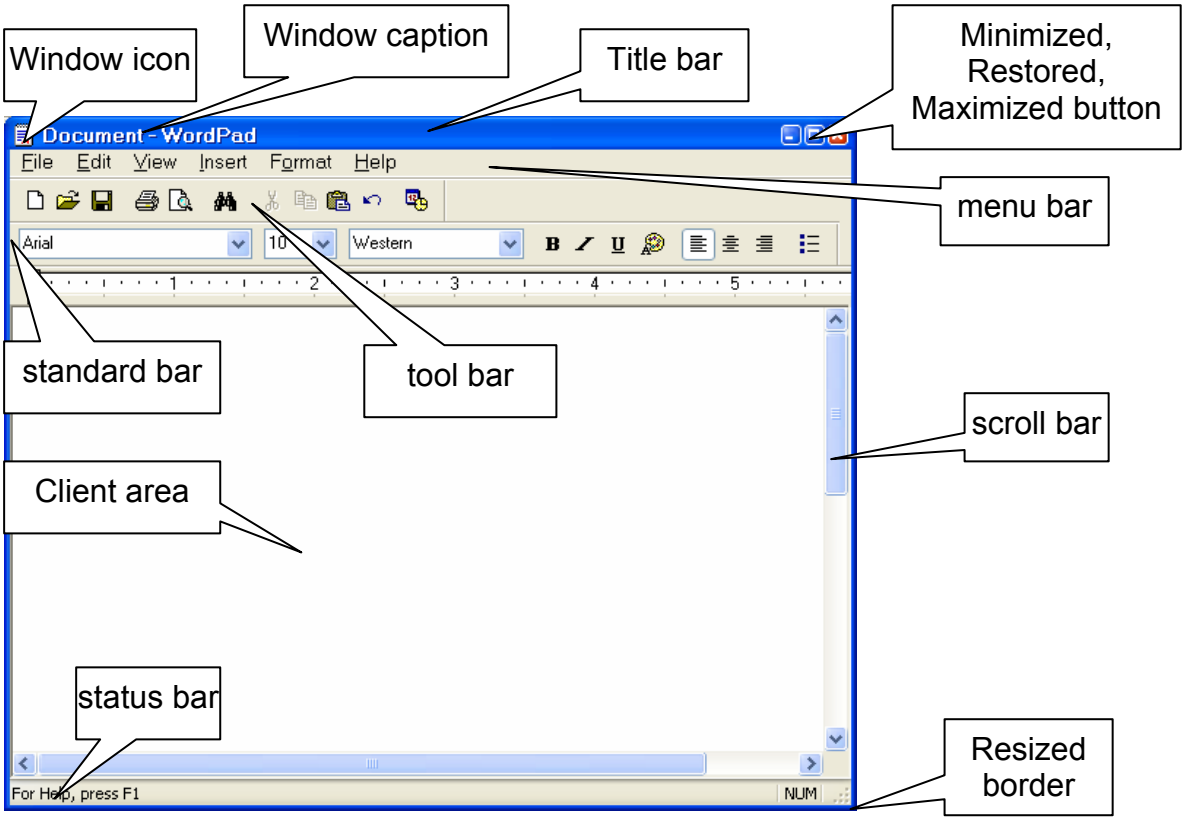


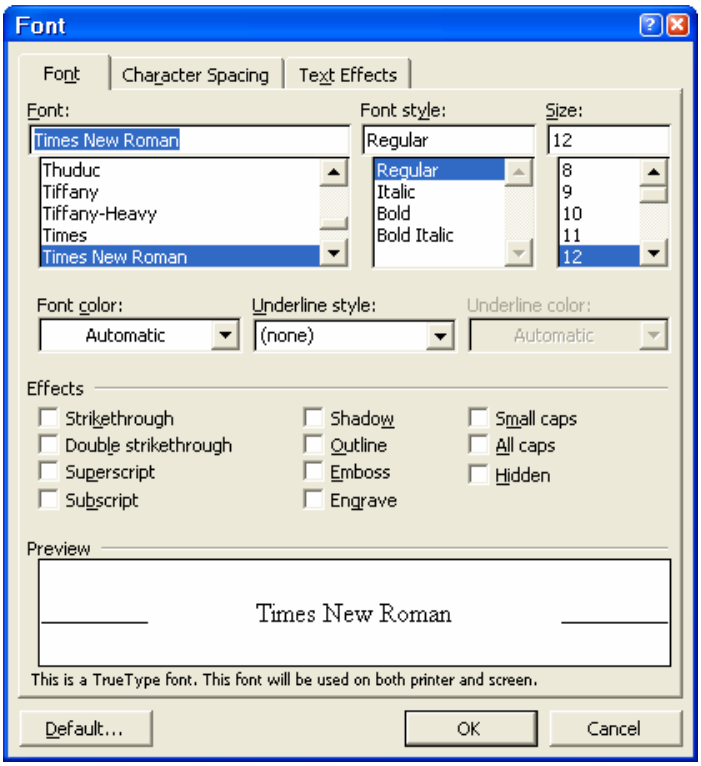
- Dialog:
  - ✓ Một cửa sổ làm việc.
  - ✓ Thường có kích thước cố định.
  - ✓ Thường không có menu bar.
  - ✓ Thường có các button, edit box, list-box,…
  - ✓ Ví dụ: Calculator, CD Player,…



- **Cửa sổ:**
  - ✓ **Định nghĩa:**
    - Là 1 vùng chữ nhật trên màn hình.
    - Dùng để hiển thị kết quả output.
    - Và nhận các input từ người dùng
  - ✓ **Công việc đầu tiên của 1 ứng dụng GUI là tạo 1 cửa sổ làm việc.**
  - ✓ **Nguyên tắc quản lý:**
    - Mô hình phân cấp: mỗi cửa sổ đều có 1 cửa sổ cha (parent window), ngoại trừ cửa sổ nền Desktop.
    - Tại mỗi thời điểm, chỉ có 1 cửa sổ nhận input từ user (Active/Focused window).
  - ✓ **Phân loại:**
    - Cửa sổ Desktop.
    - Cửa sổ tiêu chuẩn.
    - Cửa sổ hộp thoại (Dialog box).
    - Các control.




Window icon
Window caption
Title bar
Minimized, Restored, Maximized button
Document - WordPad
File Edit View Insert Format Help
menu bar
Arial
10
Western
standard bar
tool bar
scroll bar
Client area
status bar
For Help, press F1
NUM
Resized border


Font
Font
Character Spacing
Text Effects
Font:
Times New Roman
Thuduc
Tiffany
Tiffany-Heavy
Times
Times New Roman
Font style:
Regular
Regular
Italic
Bold
Bold Italic
Size:
12
8
9
10
11
12
Font color:
Automatic
Underline style:
(none)
Underline color:
Automatic
Effects
Strikethrough
Double strikethrough
Superscript
Subscript
Shadow
Outline
Emboss
Engrave
Small caps
All caps
Hidden
Preview
Times New Roman
This is a TrueType font. This font will be used on both printer and screen.
Default...
OK
Cancel

```
4   int WINAPI WinMain (HANDLE hInst, HANDLE hPrevInst,
5                       LPSTR lpszCmdLine, int nCmdShow)
6   {
7       HWND hwnd;      MSG msg;
8       WNDCLASSEX wndclass;
9       wndclass.cbSize = sizeof(wndclass);
10      wndclass.style = CS_HREDRAW | CS_VREDRAW;
11      wndclass.lpfnWndProc = XulyMessage;
12      wndclass.cbClsExtra = 0;
13      wndclass.cbWndExtra = 0;
14      wndclass.hInstance = hInst;
15      wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION);
16      wndclass.hCursor = LoadCursor (NULL, IDC_ARROW);
17      wndclass.hbrBackground = GetStockObject (WHITE_BRUSH);
18      wndclass.lpszMenuName = NULL;
19      wndclass.lpszClassName = szAppName;
20      wndclass.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
21      RegisterClassEx(&wndclass);
22      hwnd = CreateWindow(szAppName,
23                          "Vi du mo dau",
24                          WS_OVERLAPPEDWINDOW,
25                          CW_USEDEFAULT, CW_USEDEFAULT,
26                          CW_USEDEFAULT, CW_USEDEFAULT,
27                          HWND_DESKTOP,
28                          NULL,
29                          hInst,
30                          NULL);
31      ShowWindow (hwnd, nCmdShow);
32      UpdateWindow (hwnd);
33      while (GetMessage (&msg, NULL, 0, 0))
34      {
35          TranslateMessage (&msg);
36          DispatchMessage (&msg);
37      }
38      return msg.wParam;
39  }
40  LRESULT CALLBACK XulyMessage (HWND hwnd, UINT iMsg,
41                                WPARAM wParam, LPARAM lParam)
42  {
43      HDC hdc;
44      PAINTSTRUCT ps;
45      RECT rect;
46      switch (iMsg)
47      {
48          case WM_PAINT:
49              hdc = BeginPaint (hwnd, &ps);
```

```
50                      GetClientRect (hwnd, &rect);
51                      DrawText (hdc, "Lap trinh C for Win", -1, &rect,
52                          DT_SINGLELINE | DT_CENTER | DT_VCENTER);
53                      EndPaint (hwnd, &ps);
54                      break;
55              case WM_DESTROY:
56                      PostQuitMessage(0);
57                      break;
58              default:
59                      return DefWindowProc (hwnd, iMsg, wParam, lParam);
60      }
61      return 0;
62  }
```

Ta sẽ khảo sát ví dụ trên để nắm được nguyên lý hoạt động của chúng. Trên đây là đoạn chương trình đơn giản trên Windows, chương trình chỉ hiển thị 1 khung cửa sổ và 1 dòng chữ nhưng có rất nhiều lệnh mà cú pháp rất khó nhớ. Do vậy, nguyên tắc lập trình trên Windows chủ yếu là sao chép và chỉnh sửa những nơi cần thiết dựa vào một chương trình mẫu có sẵn.

a. Hàm WinMain() được thực hiện đầu tiên hay còn gọi là điểm vào của chương trình.

   ❖ Ta thấy hàm này có 4 tham số:

      o hInst, hPrevinst: Chỉ số chương trình khi chúng đang chạy. Vì Windows là hệ điều hành đa nhiệm, có thể có nhiều bản của cùng một chương trình cùng chạy vào cùng một thời điểm nên phải quản lý chặt chẽ chúng. hInst là chỉ số bản chương trình vừa khởi động, hPrevinst là chỉ số của bản đã được khởi động trước đó và chúng luôn có giá trị NULL.

      o lpszCmdLine: chứa địa chỉ đầu của xâu ký tự các đối số dòng lệnh.

      o nCmdShow: Cho biết cách thức hiển thị cửa sổ khi chương trình khởi động. Windows có thể gán giá trị SW_SHOWNORMAL hay SW_SHOWMINNOACTIVE.

      Các tham số trên do hệ điều hành truyền vào.

   ❖ Định nghĩa lớp cửa sổ và đăng ký với Windows

      o Lớp cửa sổ (window class):

- Là một tập các thuộc tính mà HĐH Windows sử dụng làm khuôn mẫu (template) khi tạo lập cửa sổ.
- Mỗi lớp cửa sổ được đặc trưng bằng 1 tên (class-name) dạng chuỗi.
- Phân loại class:
  - Lớp cửa sổ của hệ thống (System class):
    Được định nghĩa trước bởi HĐH Windows.
    Các ứng dụng không thể hủy bỏ.

Class	Description
Button	The class for a button
ComboBox	The class for a combo box
Edit	The class for an edit control
ListBox	The class for a list box
MDIClient	The class for a MDI client window
ScrollBar	The class for a scroll bar
Static	The class for a static control

  - Lớp cửa sổ do ứng dụng định nghĩa:
    Được đăng ký bởi ứng dụng.
    Có thể hủy bỏ khi không còn sử dụng nữa.
    Lớp toàn cục của ứng dụng (Application global class).
    Lớp cục bộ của ứng dụng (Application local class).
  - Mỗi cửa sổ đều thuộc một lớp xác định.

o Khi lần đầu chạy, ứng dụng phải định nghĩa và đăng ký lớp với cửa sổ (Window Class). Đây là cấu trúc dữ liệu mô tả tính chất của cửa sổ, lần lượt ta gán các giá trị ban đầu cho các thành phần của cấu trúc lớp cửa sổ, bao gồm: Kích thước, kiểu, địa chỉ hàm xử lý thông điệp cửa sổ, định nghĩa hình dạng cho con trỏ chuột (cursor) và biểu tượng (Icon), màu nền, tên lớp cửa sổ.

Macro	Màu nền cửa sổ

BLACK_BRUSH	Đen
DKGRAY_BRUSH	Xám đen
HOLLOW_BRUSH	Không tô
LTGRAY_BRUSH	Xám nhạt
WHITE_BRUSH	Trắng

```
struct WNDCLASSEX {
        UINT cbSize;
        UINT style;
        WNDPROC lpfnWndProc;
        int cbClsExtra;
        int cbWndExtra;
        HINSTANCE hInstance;
        HICON hIcon;
        HCURSOR hCursor;
        HBRUSH hbrBackground;
        LPCTSTR lpszMenuName;
        LPCTSTR lpszClassName;
        HICON hIconSm;
};
```

- Sau khi đã định nghĩa các thành phần lớp cửa sổ ta phải đăng ký lớp cửa sổ với hệ điều hành (RegisterClassEX).

  *ATOM RegisterClassEx (CONST WNDCLASSEX *lpWClass);*

  với: Kiểu giá trị của ATOM được định nghĩa trong window.h là WORD; lpWClass là con trỏ đến cấu trúc lớp cửa sổ; hàm này trả về chỉ số của lớp cửa sổ.
- Có hai nguyên nhân dẫn đến việc đăng ký cửa sổ thất bại:
  - Trùng tên giữa các ứng dụng trong hệ điều hành.
  - Không đủ bộ nhớ.

❖ Tạo lập cửa sổ làm việc (Frame Window)

- Sau khi đăng ký thành công ta có thể tạo lập cửa sổ thông qua hàm CreateWindow().

HWND CreateWindow (

LPCSTR lpClassName,

LPCSTR lpWinName,

DWORD dwStyle,

int X, int Y,

int Width, int Height,

HWND hParent,

HMENU hMenu,

HINSTANCE hInst,

LPVOID lpszAdditional);

Kiểu	Mô tả
WS_MAXIMIZEBOX	Cửa sổ có phím dãn to trên thanh tiêu đề
WS_MINIMIZEBOX	Cửa sổ có phím co nhỏ trên thanh tiêu đề
WS_OVERLAPPED	Cửa sổ maximize và không có cửa sổ cha
WS_SYSMENU	Cửa sổ có hộp thực đơn hệ thống
WS_VSCROLL	Cửa sổ có thanh trượt dọc
WS_HSCROLL	Cửa sổ có thanh trượt ngang

- Gọi hàm ShowWindow()để hiển thị cửa sổ

BOOL ShowWindow (HWND hwnd, int nShow);

với: hwnd chỉ số cửa sổ cần hiển thị.

nShow cách thức hiển thị của cửa sổ, tham số này được nhận giá trị lần đầu tiên của hàm WinMain(), chúng có thể nhận các giá trị sau:

Macro	Cách thức hiển thị
SW_HIDE	Dấu cửa sổ
SW_MINIMIZE	Thu nhỏ cửa sổ
SW_MAXIMIZE	Phóng to cửa sổ toàn màn hình
SW_RESTORE	Trở lại kích thước thông thường

- Để thông báo cho ứng dụng biết là phải vẽ lại vùng làm việc của cửa sổ, ta phải gọi hàm UpdateWindow() yêu cầu Windows gửi thông điệp đến hàm xử lý thông điệp cửa sổ.

❖ Vòng lặp thông điệp

- Khi nhấn phím hay chuột, Windows chuyển đổi sự kiện này thành các thông điệp và đặt vào hàng đợi thông điệp. Vòng lặp thông điệp có nhiệm vụ nhận và xử lý các thông điệp trong hàng đợi.
- TranslateMessage: Dịch thông điệp sang dạng tiêu chuẩn.
- DispatchMessage: Phân phối thông điệp đến hàm xử lý thông điệp tương ứng.

b. Thủ tục xử lý thông điệp

❖ Nhận và xử lý thông điệp của chương trình.

❖ Một chương trình có thể có nhiều thủ tục window.

❖ Một lớp cửa sổ sẽ khai báo 1 thủ tục window.

❖ Các thông điệp sau khi xử lý nên trả về giá trị 0.

❖ Dạng tổng quát:

```
LRESULT CALLBACK WndProc(
    HWND hWnd,        //handle của window nhận message
    UINT message,     //ID của thông điệp (tên thông điệp)
    WPARAM wParam,//thamsố thứ nhất của message (WORD)
    LPARAM lParam)   //thamsố thứ hai của message (LONG)
{
    switch (message)
    {
            case WM_COMMAND:
                return 0;
            case WM_PAINT:
                return 0;
            case WM_DESTROY:
                PostQuitMessage(0);
                return 0;
            default:
                return DefWindowProc(hWnd, message, wParam,
                lParam);
    }
}
```

Thông điệp WM_PAINT:

❖ Cập nhật lại thông tin vẽ trên màn hình.

```
TCHAR szWindowClass[MAX_LOADSTRING];// The title bar text
// Foward declarations of functions included in this code module:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine, int       nCmdShow)
{
        // TODO: Place code here.
        MSG msg;
        HACCEL hAccelTable;
        // Initialize global strings
        LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
        LoadString(hInstance, IDC_BT1, szWindowClass, MAX_LOADSTRING);
        MyRegisterClass(hInstance);
        // Perform application initialization:
        if (!InitInstance (hInstance, nCmdShow))
        {
                return FALSE;
        }
        hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_BT1);
        // Main message loop:
        while (GetMessage(&msg, NULL, 0, 0))
        {
                if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
                {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
        }
        return msg.wParam;
}
//  FUNCTION: MyRegisterClass()
//  PURPOSE: Registers the window class.
//  COMMENTS:
//    This function and its usage is only necessary if you want this code
//    to be compatible with Win32 systems prior to the 'RegisterClassEx'
//    function that was added to Windows 95. It is important to call this function
//    so that the application will get 'well formed' small icons associated
//    with it.
ATOM MyRegisterClass(HINSTANCE hInstance)
{
        WNDCLASSEX wcex;
        wcex.cbSize = sizeof(WNDCLASSEX);
        wcex.style          = CS_HREDRAW | CS_VREDRAW;
```

```
54          wcex.lpfnWndProc   = (WNDPROC)WndProc;
55          wcex.cbClsExtra    = 0;
56          wcex.cbWndExtra    = 0;
57          wcex.hInstance     = hInstance;
58          wcex.hIcon         = LoadIcon(hInstance, (LPCTSTR)IDI_BT1);
59          wcex.hCursor       = LoadCursor(NULL, IDC_ARROW);
60          wcex.hbrBackground      = (HBRUSH)(COLOR_WINDOW+1);
61          wcex.lpszMenuName       = (LPCSTR)IDC_BT1;
62          wcex.lpszClassName      = szWindowClass;
63          wcex.hIconSm       = LoadIcon(wcex.hInstance,(LPCTSTR)IDI_SMALL);
64          return RegisterClassEx(&wcex);
65  }
66  //  FUNCTION: InitInstance(HANDLE, int)
67  //  PURPOSE: Saves instance handle and creates main window
68  //  COMMENTS:
69  //       In this function, we save the instance handle in a global variable and
70  //       create and display the main program window.
71  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
72  {
73          HWND hWnd;
74          hInst = hInstance; // Store instance handle in our global variable
75          hWnd = CreateWindow(szWindowClass,
76                                  szTitle,
77                                  WS_OVERLAPPEDWINDOW,
78                                  CW_USEDEFAULT,
79                                  0,
80                                  CW_USEDEFAULT,
81                                  0,
82                                  NULL,
83                                  NULL,
84                                  hInstance,
85                                  NULL);
86          if (!hWnd)
87          {
88                  return FALSE;
89           }
90          ShowWindow(hWnd, nCmdShow);
91          UpdateWindow(hWnd);
92           return TRUE;
93  }
94  // FUNCTION: WndProc(HWND, unsigned, WORD, LONG)
95  // PURPOSE:  Processes messages for the main window.
96  // WM_COMMAND- process the application menu
97  // WM_PAINT        - Paint the main window
98  // WM_DESTROY   - post a quit message and return
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
        int wmId, wmEvent,x,y;
        PAINTSTRUCT ps;
        HDC hdc;
        TCHAR szHello[MAX_LOADSTRING];
        LoadString(hInst, IDS_HELLO, szHello, MAX_LOADSTRING);
        switch (message)
        {
                case WM_COMMAND:
                        wmId    = LOWORD(wParam);
                        wmEvent = HIWORD(wParam);
                        // Parse the menu selections:
                        switch (wmId)
                        {
                                case IDM_ABOUT:
                                        DialogBox(hInst,(LPCTSTR)IDD_ABOUTBOX,
                                        hWnd, (DLGPROC)About);
                                        break;
                                case IDM_EXIT:
                                        DestroyWindow(hWnd);
                                        break;
                                default:
                                        return DefWindowProc(hWnd, message,
                                        wParam, lParam);
                        }
                        break;
                case WM_LBUTTONDOWN:
                        hdc = GetDC(hWnd);
                        // TODO: Add any drawing code here...
                        x=LOWORD(lParam);
                        y=HIWORD(lParam);
                        TextOut(hdc,x,y,(LPCTSTR)szHello, strlen(szHello));
                        break;
                case WM_PAINT:
                        hdc = BeginPaint(hWnd, &ps);
                        // TODO: Add any drawing code here...
                        RECT rt;
                        GetClientRect(hWnd, &rt);
                        DrawText(hdc, szHello, strlen(szHello), &rt, DT_CENTER);
                        EndPaint(hWnd, &ps);
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);                     break;
                default:
```

```
145                     return DefWindowProc(hWnd, message, wParam, lParam);
146         }
147         return 0;
148 }
149 // Mesage handler for about box.
150 LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam,
151 LPARAM lParam)
152 {
153         switch (message)
154         {
155                 case WM_INITDIALOG:             return TRUE;
156                 case WM_COMMAND:
157                         if (LOWORD(wParam)== IDOK||LOWORD(wParam)==
158                         IDCANCEL)
159                         {
160                                 EndDialog(hDlg, LOWORD(wParam));return TRUE;
161                         }
162                         break;
163         }
164         return FALSE;
165 }
```

- ❖ Vị trí hiển thị ký tự TextOut() là tọa độ tương đối trong cửa sổ (tọa độ logic).
- ❖ Windows sẽ ánh xạ đơn vị này thành pixel khi hiển thị ký tự.
- ❖ Ở chế độ mặc định tọa độ logic ≈ pixel.

c) Mô hình màu RGB (Red – Green – Blue)

Byte 3	Byte 2	Byte 1	Byte 0
0	Blue	Green	Red

- ❖ Có giá trị từ 0 – 255

  $(0, 0, 0)_{\text{đen}} \rightarrow (255, 255, 255)_{\text{trắng}}$
- ❖ Các hàm API liên quan đến màu đều sử dụng mô hình RGB.
- ❖ Định nghĩa màu COLORREF RGB (int red, int green, int blue).

Ví dụ 1 : Vẽ hình chữ nhật

```
HDC hDC;
HPEN hPen, oldHPen;
hDC=GetDC(hWnd);
hPen=CreatePen(PS_SOLID, 5, RGB(0, 0, 255));
oldHPen=(HPEN)SelectObject(hDC, hPen);
Rectangle(hDC, 20, 20, 100, 100);
SelectObject(hDC, oldHPen);
DeleteObject(hPen);
ReleaseDC(hWnd, hDC);
```

d) Tạo lập và giải phóng memory device context

- ❖ Memory device context (MDC) là một device context ảo không gắn với một thiết bị xuất cụ thể nào. Muốn kết quả kết xuất ra thiết bị vật lý ta phải chép MDC lên một device context thật sự(device context có liên kết với thiết bị vật lý). MDC thường được dùng như một device context trung gian để vẽ trước khi thực sự xuất ra thiết bị, nhằm giảm sự chớp giật nếu thiết bị xuất là window hay màn hình.
- ❖ Để tạo MDC tương thích với một hDC cụ thể, sử dụng hàm CreateCompatibleDC:

  ```
  HDC hMemDC;
  hMemDC = CreateCompatibleDC(hDC);
  ```
- ❖ Đơn giản hơn, có thể đặt NULL vào vị trí hDC, Windows sẽ tạo một device context tương thích với màn hình.

- Hủy MDC bằng hàm DeleteDC.
- MDC có bề mặt hiển thị như một thiết bị thật. Tuy nhiên, bề mặt hiển thị này lúc đầu rất nhỏ, chỉ là một pixel đơn sắc. Không thể làm gì với một bề mặt hiển thị chỉ gồm 1 bit như vậy. Do đó cần làm cho bề mặt hiển thị này rộng hơn bằng cách chọn một đối tượng bitmap GDI vào MDC:

  SelectObject(hMemDC, hBitmap);
- Chỉ có thể chọn đối tượng bitmap vào MDC, không thể chọn vào một device context cụ thể được.
- Sau khi chọn một đối tượng bitmap cho MDC, có thể dùng MDC như một device context thật sự.
- Sau khi được hoàn tất trong MDC, ảnh được đưa ra device context thật sự bằng hàm BitBlt:

  BitBlt(hDC, xDest, yDest, nWidth, nHeight, hMemDC, xSource, ySource);
- Ví dụ : Chuẩn bị ảnh trước khi đưa ra màn hình, tránh gây chớp màn hình trong thông điệp WM_PAINT.

```
case WM_PAINT:
hdc = BeginPaint(hWnd, &ps);
// Lấy về kích thước vùng client của cửa sổ hiện hành
RECT rect;
GetClientRect(hWnd, &rect);
// Tạo MDC tương thích với DC của cửa sổ
HDC hMemDC;
hMemDC = CreateCompatibleDC(hdc);
// Chọn một đối tượng bitmap để mở rộng vùng hiển thị cho MDC
HBITMAP bitmap,oBitmap;
bitmap = CreateCompatibleBitmap(hdc, rect.right, rect.bottom);
oBitmap = (HBITMAP)SelectObject(hMemDC, bitmap);
// Vẽ lại nền MDC
FillRect(hMemDC, &rect, HBRUSH (GetBkColor(hMemDC)));
// Xuất hình ảnh, text ra MDC
SetPixel(hMemDC, 0, 0, RGB(255,0,0));
MoveToEx(hMemDC, 50, 50, NULL);
LineTo(hMemDC, 100, 100);
Rectangle(hMemDC, 10, 10, 100, 100);
TextOut(hMemDC, 15 ,15, "Testing MDC", 11);
```

- **BOOL Rectangle(HDC** hDC, **int** left, **int** top, **int** right, **int** bottom);
  Vẽ hình chữ nhật có tọa độ là left, top, right, bottom lên hDC.
- **HPEN CreatePen(int** penStyle, **int** penWidth, **COLORREF** penColor);
  Tạo bút vẽ có kiểu penStyle, độ dày nét vẽ là penWidth, màu penColor. Hàm trả về handle của bút vẽ nếu thành công và trả về NULL nếu thất bại. Các giá trị của penStyle như sau :

Giá trị	Giải thích
PS_SOLID	
PS_DASH	
PS_DOT	
PS_DASHDOT	
PS_DASHDOTDOT	
PS_NULL	Không hiển thị
PS_INSIDEFRAME	

Các kiểu bút vẽ penStyle

Ví dụ : Tạo bút vẽ mới và dùng bút vẽ này vẽ một số đường cơ sở.

```
HDC hDC;
POINT PointArr[3];
HPEN hPen, hOldPen;
hDC = GetDC(hWnd);
PointArr[0].x = 50;
PointArr[0].y = 10;
PointArr[1].x = 250;
PointArr[1].y = 50;
PointArr[2].x = 125;
PointArr[2].y = 130;
Polyline(hDC, PointArr, 3);
hPen = (HPEN)CreatePen(PS_SOLID, 1, RGB(0, 0, 255));
hOldPen = SelectObject(hDC, hPen);
MoveToEx(hDC, 100, 100, NULL);
LineTo(hDC, 200, 150);
SelectObject(hDC, hOldPen);
DeleteObject(hPen);
ReleaseDC(hWnd, hDC);
```

b) Nhóm hàm miền

- **HBRUSH CreateSolidBrush(COLORREF cRef);**

  Tạo mẫu tô đặc với màu cRef.

WM_CHAR	Thông điệp này được gởi tới cửa sổ có sự quan tâm khi thông điệp WM_KEYDOWN đã được dịch từ hàm TranslateMessage. Thông điệp WM_CHAR có chứa mã kí tự của phím được nhấn.
WM_DEADCHAR	Thông điệp này được gởi tới cửa sổ có sự quan tâm khi thông điệp WM_KEYUP đã được xử lý từ hàm TranslateMessage. Thông điệp này xác nhận mã kí tự khi một phím dead key được nhấn. Phím dead key là phím kết hợp để tạo ra kí tự ngôn ngữ không có trong tiếng anh (xuất hiện trong bàn phím hỗ trợ ngôn ngữ khác tiếng Anh).
WM_GETHOTKEY	Ứng dụng gởi thông điệp này để xác định một phím nóng liên quan đến một cửa sổ. Để gởi thông điệp này thì dùng hàm SendMessage.
WM_HOTKEY	Thông điệp này được gởi khi người dùng nhấn một phím nóng được đăng kí trong RegisterHotKey.
WM_KEYDOWN	Thông điệp này được gởi cho cửa sổ nhận được sự quan tâm khi người dùng nhấn một phím trên bàn phím. Phím này không phải phím hệ thống (Phím không có nhấn phím Alt).
WM_KEYUP	Thông điệp này được gởi cho cửa sổ nhận được sự quan tâm khi người dùng nhả một phím đã được nhấn trước đó.Phím này không phải phím hệ thống (Phím không có nhấn phím Alt).
WM_KILLFOCUS	Thông điệp này được gởi tới cửa sổ đang nhận được sự quan tâm trước khi nó mất quyền này.
WM_SETFOCUS	Thông điệp này được gởi tới cửa sổ sau khi cửa sổ nhận được sự quan tâm của Windows
WM_SETHOTKEY	Ứng dụng sẽ gởi thông điệp này đến cửa sổ liên quan đến phím nóng, khi người dùng nhấn một phím nóng thì cửa sổ tương ứng liên quan tới phím nóng này sẽ được kích hoạt.
WM_SYSCHAR	Thông điệp này sẽ được gởi tới cửa sổ nhận được sự quan tâm khi hàm TranslateMesage xử lý xong thông điệp WM_SYSKEYDOWN.

	Thông điệp WM_SYSCHAR chứa mã cửa phím hệ thống. Phím hệ thống là phím có chứa phím Alt và tổ hợp phím khác.
WM_SYSDEADCHAR	Thông điệp này được gởi tới cửa sổ nhận được sự quan tâm khi một thông điệp WM_SYSKEYDOWN được biên dịch trong hàm TranslateMessage. Thông điệp này xác nhận mã kí tự của phím hệ thống deadkey được nhấn.
WM_SYSKEYDOWN	Thông điệp này được gởi tới cửa sổ nhận được sự quan tâm khi người dùng nhấn phím hệ thống.

d. Ví dụ

```
1   #define BUFSIZE 65535
2   #define SHIFTED 0x8000
3
4   LONG APIENTRY MainWndProc(HWND hwndMain, UINT uMsg,
5   WPARAM wParam, LPARAM lParam)
6   {
7      HDC hdc;                // handle to device context
8      TEXTMETRIC tm;          // structure for text metrics
9      static DWORD dwCharX;   // average width of characters
10     static DWORD dwCharY;   // height of characters
11     static DWORD dwClientX; // width of client area
12     static DWORD dwClientY; // height of client area
13     static DWORD dwLineLen; // line length
14     static DWORD dwLines;   // text lines in client area
15     static int nCaretPosX = 0; // horizontal position of caret
16     static int nCaretPosY = 0; // vertical position of caret
17     static int nCharWidth = 0; // width of a character
18     static int cch = 0;     // characters in buffer
19     static int nCurChar = 0; // index of current character
20     static PTCHAR pchInputBuf; // input buffer
21     int i, j;               // loop counters
22     int cCR = 0;            // count of carriage returns
23     int nCRIndex = 0;       // index of last carriage return
24     int nVirtKey;           // virtual-key code
25     TCHAR szBuf[128];       // temporary buffer
26     TCHAR ch;               // current character
27     PAINTSTRUCT ps;         // required by BeginPaint
28     RECT rc;                // output rectangle for DrawText
29     SIZE sz;                // string dimensions
30     COLORREF crPrevText;    // previous text color
```

```
31          COLORREF crPrevBk;        // previous background color
32           switch (uMsg)
33           {
34                case WM_CREATE:
35                 // Get the metrics of the current font.
36                 hdc = GetDC(hwndMain);
37                GetTextMetrics(hdc, &tm);
38                ReleaseDC(hwndMain, hdc);
39                 // Save the average character width and height.
40                 dwCharX = tm.tmAveCharWidth;
41                dwCharY = tm.tmHeight;
42                 // Allocate a buffer to store keyboard input.
43                 pchInputBuf = (LPTSTR) GlobalAlloc(GPTR,
44                   BUFSIZE * sizeof(TCHAR));
45                return 0;
46              case WM_SIZE:
47                 // Save the new width and height of the client area.
48                 dwClientX = LOWORD(lParam);
49                dwClientY = HIWORD(lParam);
50                 // Calculate the maximum width of a line and the
51                // maximum number of lines in the client area.
52                dwLineLen = dwClientX - dwCharX;
53                dwLines = dwClientY / dwCharY;
54                break;
55             case WM_SETFOCUS:
56                 // Create, position, and display the caret when the
57                // window receives the keyboard focus.
58                CreateCaret(hwndMain, (HBITMAP) 1, 0, dwCharY);
59                SetCaretPos(nCaretPosX, nCaretPosY * dwCharY);
60                ShowCaret(hwndMain);
61                break;
62              case WM_KILLFOCUS:
63                 // Hide and destroy the caret when the window loses the
64                // keyboard focus.
65                HideCaret(hwndMain);
66                DestroyCaret();
67                break;
68              case WM_CHAR:
69                switch (wParam)
70                {
71                   case 0x08:  // backspace
72                   case 0x0A:  // linefeed
73                   case 0x1B:  // escape
74                      MessageBeep((UINT) -1);
75                      return 0;
76                    case 0x09:  // tab
```

```
77                      // Convert tabs to four consecutive spaces.
78                      for (i = 0; i < 4; i++)
79                         SendMessage(hwndMain, WM_CHAR, 0x20, 0);
80                      return 0;
81                    case 0x0D:  // carriage return
82                      // Record the carriage return and position the
83                      // caret at the beginning of the new line.
84                      pchInputBuf[cch++] = 0x0D;
85                      nCaretPosX = 0;
86                      nCaretPosY += 1;
87                      break;
88                    default:    // displayable character
89                      ch = (TCHAR) wParam;
90                      HideCaret(hwndMain);
91                      // Retrieve the character's width and output
92                      // the character.
93                      hdc = GetDC(hwndMain);
94                      GetCharWidth32(hdc, (UINT) wParam, (UINT) wParam,
95                         &nCharWidth);
96                      TextOut(hdc, nCaretPosX, nCaretPosY * dwCharY,
97                         &ch, 1);
98                      ReleaseDC(hwndMain, hdc);
99                      // Store the character in the buffer.
100                     pchInputBuf[cch++] = ch;
101                     // Calculate the new horizontal position of the
102                     // caret. If the position exceeds the maximum,
103                     // insert a carriage return and move the caret
104                     // to the beginning of the next line.
105                     nCaretPosX += nCharWidth;
106                     if ((DWORD) nCaretPosX > dwLineLen)
107                     {
108                        nCaretPosX = 0;
109                        pchInputBuf[cch++] = 0x0D;
110                        ++nCaretPosY;
111                     }
112                     nCurChar = cch;
113                     ShowCaret(hwndMain);
114                     break;
115                 }
116                 SetCaretPos(nCaretPosX, nCaretPosY * dwCharY);
117                 break;
118               case WM_KEYDOWN:
119                 switch (wParam)
120                 {
121                    case VK_LEFT:   // LEFT ARROW
122                       // The caret can move only to the beginning of
```

```
123                     // the current line.
124                      if (nCaretPosX > 0)
125                      {
126                         HideCaret(hwndMain);
127                          // Retrieve the character to the left of
128                         // the caret, calculate the character's
129                         // width, then subtract the width from the
130                         // current horizontal position of the caret
131                         // to obtain the new position.
132                          ch = pchInputBuf[--nCurChar];
133                         hdc = GetDC(hwndMain);
134                         GetCharWidth32(hdc, ch, ch, &nCharWidth);
135                         ReleaseDC(hwndMain, hdc);
136                         nCaretPosX = max(nCaretPosX - nCharWidth, 0);
137                         ShowCaret(hwndMain);
138                      }
139                      break;
140                   case VK_RIGHT:  // RIGHT ARROW
141                      // Caret moves to the right or, when a carriage
142                      // return is encountered, to the beginning of
143                      // the next line.
144                      if (nCurChar < cch)
145                      {
146                         HideCaret(hwndMain);
147                          // Retrieve the character to the right of
148                         // the caret. If it's a carriage return,
149                         // position the caret at the beginning of
150                         // the next line.
151                          ch = pchInputBuf[nCurChar];
152                         if (ch == 0x0D)
153                         {
154                            nCaretPosX = 0;
155                            nCaretPosY++;
156                         }
157                          // If the character isn't a carriage
158                         // return, check to see whether the SHIFT
159                         // key is down. If it is, invert the text
160                         // colors and output the character.
161                          else
162                         {
163                            hdc = GetDC(hwndMain);
164                            nVirtKey = GetKeyState(VK_SHIFT);
165                            if (nVirtKey & SHIFTED)
166                            {
167                               crPrevText = SetTextColor(hdc,
168                                  RGB(255, 255, 255));
```

```
169                         crPrevBk = SetBkColor(hdc,
170                            RGB(0,0,0));
171                         TextOut(hdc, nCaretPosX,
172                            nCaretPosY * dwCharY,
173                            &ch, 1);
174                         SetTextColor(hdc, crPrevText);
175                         SetBkColor(hdc, crPrevBk);
176                      }
177                       // Get the width of the character and
178                      // calculate the new horizontal position of the caret.
179                       GetCharWidth32(hdc, ch, ch, &nCharWidth);
180                      ReleaseDC(hwndMain, hdc);
181                      nCaretPosX = nCaretPosX + nCharWidth;
182                   }
183                   nCurChar++;
184                   ShowCaret(hwndMain);
185                   break;
186                }
187                break;
188             case VK_UP:     // UP ARROW
189             case VK_DOWN:   // DOWN ARROW
190                MessageBeep((UINT) -1);
191                return 0;
192              case VK_HOME:   // HOME
193                 // Set the caret's position to the upper left
194                // corner of the client area.
195                nCaretPosX = nCaretPosY = 0;
196                nCurChar = 0;
197                break;
198              case VK_END:    // END
199                 // Move the caret to the end of the text.
200                 for (i=0; i < cch; i++)
201                 {
202                    // Count the carriage returns and save the
203                    // index of the last one.
204                     if (pchInputBuf[i] == 0x0D)
205                     {
206                        cCR++;
207                        nCRIndex = i + 1;
208                     }
209                }
210                nCaretPosY = cCR;
211
212                // Copy all text between the last carriage
213                // return and the end of the keyboard input
214                // buffer to a temporary buffer.
```

```
  int nIndex   // system metric or configuration setting
);
```

fMouse = GetSystemMetrics(SM_MOUSEPRESENT);

Giá trị trả về fMouse là TRUE (1) nếu có thiết bị chuột được cài đặt, và ngược lại bằng FALSE (0) nếu thiết bị chuột không được cài đặt vào máy.

b. Trong lớp cửa sổ ta định nghĩa con trỏ chuột cho ứng dụng

wndclass.hCursor = **LoadCursor** (NULL, IDC_ARROR);

wndclass.style = CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS;

Với thiết bị chuột ta có thể có các hành động như sau:

- *Kích chuột* : nhấn và thả một nút chuột.
- *Kích đúp chuột* : nhấn và thả chuột nhanh (nhấn 2 lần nhanh).
- *Kéo* : di chuyển chuột trong khi vẫn nắm giữ một nút.

c. Thông điệp chuột trong vùng làm việc

Nút	Nhấn	Thả	Nhấn đúp
Trái	WM_LBUTTONDOWN	WM_LBUTTONUP	WM_LBUTTONDBLCLK
Giữa	WM_MBUTTONDOWN	WM_MBUTTONUP	WM_MBUTTONDBLCLK
Phải	WM_RBUTTONDOWN	WM_MBUTTONUP	WM_RBUTTONDBLCLK

d. Giá trị wParam sẽ cho biết trạng thái của nút nhấn, phím Shift, và phím Ctrl.

MK_LBUTTON	Nút chuột trái nhấn
MK_MBUTTON	Nút chuột giữa nhấn
MK_RBUTTON	Nút chuột phải nhấn
MK_SHIFT	Phím Shift được nhấn
MK_CONTROL	Phím Ctrl được nhấn

e. Giá trị lParam sẽ cho biết vị trí chuột tại thời điểm phát sinh message.

- *2 bytes thấp: tọa độ x*
- *2 bytes cao: tọa độ y*

f. Ví dụ

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    static POINT oldPoint;
    static int iC;
    int WIDTH_PEN = 2;
    HPEN oPen,pen;
    COLORREF Col [ ] ={ RGB (0, 0, 0) , RGB (255 ,0 ,0),
    RGB (0, 255, 0), RGB (0, 0, 255), RGB (255, 255, 0)};
    POINT point;
    TCHAR str [255];
    switch ( message ) // Xử lý thông điệp
    {
        case WM_LBUTTONDOWN:
        /* Vẽ đường thẳng từ vị trí trước đó đến vị trí chuột hiện tại*/
            hdc = GetDC ( hWnd );
            pen = CreatePen ( PS_SOLID,WIDTH_PEN,Col [
            iC] );
            oPen = ( HPEN ) SelectObject ( hdc,pen );
            point.x = LOWORD ( lParam );
            point.y = HIWORD ( lParam );
            MoveToEx ( hdc, oldPoint.x, oldPoint.y, NULL );
            LineTo ( hdc, point.x, point.y );
            oldPoint = point;
            /* Chọn lại bút vẽ trước đó và hủy bút vẽ vừa tạo*/
            SelectObject ( hdc, oPen );
            DeleteObject ( pen );
            ReleaseDC ( hWnd, hdc );
            break;
        case WM_RBUTTONDOWN:
        /* Chuyển index của bảng màu sang vị trí tiếp theo, nếu
        cuối bảng màu thì quay lại màu đầu tiên*/
            iC = ( iC+1 ) % ( sizeof ( Col ) / sizeof (
            COLORREF ) );
            break;
        case WM_MOUSEMOVE:
        /* Xuất toạ độ chuột hiện thời lên thanh tiêu đề*/
            sprintf ( str,"Toa do chuot x = %d, To do y = %d",
            LOWORD(lParam), HIWORD(lParam));
            SetWindowText ( hWnd, str );
            /* Kiểm tra xem có giữ phím chuột trái hay không*/
            if ( wParam & MK_LBUTTON )
            {
                hdc = GetDC ( hWnd );
```

```
#include "stdio.h"
#define MAX_POINT 10000
#define IDT_TIMER1 1
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    PAINTSTRUCT ps;
    HDC hdc;
    static int NumCir = 0;
    static POINT point [ MAX_POINT ];
    int r = 5, i;
    HPEN pen, oldPen;
    RECT rc;
    TCHAR str [255];
    /* Xử lý thông điệp*/
    switch ( message )
    {
        case WM_CREATE:
            SetTimer(hWnd, IDT_TIMER1, 500,
            (TIMERPROC) NULL);
            srand ( (unsigned) time( NULL ) );
            break;
        case WM_PAINT:
            hdc = BeginPaint ( hWnd, &ps );
            pen = CreatePen ( PS_SOLID, 2, RGB (255,0,0) );
            oldPen = (HPEN) SelectObject ( hdc, pen );
            for( i=0; i < NumCir; i++ )
                Arc ( hdc, point[i].x-r, point[i].y-r,
                point[i].x+r, point[i].y+r, point[i].x+r,
                point[i].y,point[i].x+r,point[i].y);
            SelectObject ( hdc, oldPen );
            DeleteObject ( pen );
            EndPaint ( hWnd, &ps );
            break;
        case WM_TIMER:
            GetClientRect ( hWnd, &rc );
            point [NumCir].x = rand( ) % (rc.right - rc.left);
            point [NumCir].y = rand( ) % (rc.bottom - rc.top);
            NumCir++;
            sprintf ( str,"So vong tron : %d", NumCir);
            SetWindowText ( hWnd, str );
            InvalidateRect ( hWnd, &rc, FALSE);
            break;
        case WM_DESTROY:
            KillTimer ( hWnd, IDT_TIMER1 );
            PostQuitMessage ( 0 );
```

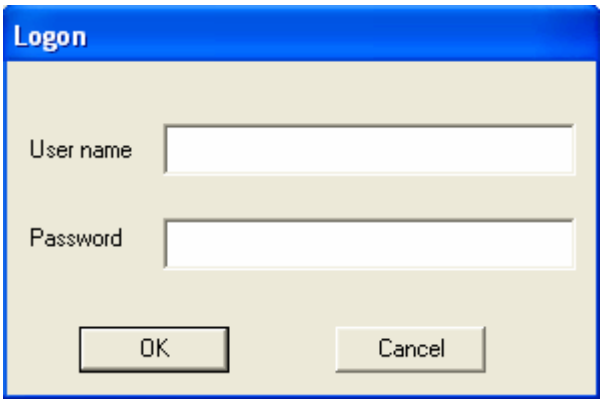
```
                    break;
                default:
                    return DefWindowProc ( hWnd, message, wParam,
                    lParam );
            }
            return 0;
        }
```

d. Ví dụ 2

```
#include <time.h>
#include "stdio.h"
#define IDT_TIMER1 1
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    PAINTSTRUCT ps;
    HDC hdc;
    /* Khai báo biến lưu các giá trị không gian*/
    struct tm *newtime;
    time_t CurTime;
    TCHAR str [255];
    RECT rc;
    /* Biến LOGFONT để tạo font mới*/
    LOGFONT lf;
    HFONT oldFont, font;
    COLORREF color = RGB (255, 0, 0), oldColor;
    switch ( message )
    {
    case WM_CREATE:
    /* khởi tạo bộ định thời gian, và khai báo hàm xử lý Timer*/
    SetTimer ( hWnd, IDT_TIMER1, 1000, ( TIMERPROC )
    TimerProc );
    break;
    case WM_PAINT:
        hdc = BeginPaint ( hWnd, &ps );
        time( &CurTime );
        newtime = localtime ( &CurTime );
        GetClientRect ( hWnd, &rc );
        sprintf(str,"Gio hien tai : %d gio: %d phut: %d giay",
        newtime->tm_hour,newtime->tm_min, newtime-
        >tm_sec);
        oldColor = SetTextColor ( hdc, color );
        memset ( &lf, 0, sizeof ( LOGFONT ) );
        lf.lfHeight = 50;
        strcpy ( lf.lfFaceName, "Tahoma" );
```

```
37                      font = CreateFontIndirect ( &lf );
38                      oldFont = ( HFONT ) SelectObject ( hdc,font );
39                      DrawText ( hdc, str, strlen(str), &rc, DT_CENTER |
40                      DT_VCENTER | DT_SINGLELINE );
41                      SetTextColor ( hdc,oldColor );
42                      SelectObject ( hdc,oldFont );
43                      DeleteObject ( font );
44                      EndPaint ( hWnd, &ps );
45                      break;
46              case WM_DESTROY:
47                      PostQuitMessage ( 0 );
48                      break;
49              default:
50                      return DefWindowProc ( hWnd, message, wParam,
51                      lParam );
52              }
53              return 0;
54      }
55      VOID CALLBACK TimerProc( HWND hwnd, UINT uMsg,
56      UINT_PTR idEvent, DWORD dwTime)
57      {
58              RECT rc;
59              GetClientRect ( hwnd, &rc );
60              InvalidateRect ( hwnd, &rc, TRUE );
61      }
```

Ví dụ:




```
IDD_DIALOG1 DIALOG DISCARDABLE  0, 0, 196, 102
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
CAPTION "Logon"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,24,81,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,109,81,50,14
    LTEXT           "User name",IDC_STATIC,7,23,40,15
    LTEXT           "Password",IDC_STATIC,7,50,40,16
    EDITTEXT DC_EDT_NAME,52,19,137,16,ES_AUTOHSCROLL
    EDITTEXT IDC_EDT_PASSWORD, 52, 48, 137, 16, ES_AUTOHSCROLL
END
```

Kiểu điều khiển	Lớp cửa sổ	Kiểu
PUSHBUTTON	Button	BS_PUSHBUTTON
DEFPUSHBUTTON	Button	BS_DEFBUSHBUTTON | WS_TABSTOP
CHECKBOX	Button	BS_CHECKBOX | WS_TABSTOP
RADIOBUTTON	Button	BS_RADIOBUTTON | WS_TABSTOP
GROUPBOX	Button	BS_GROUPBOX | WS_TABSTOP
LTEXT	Static	SS_LEFT | WS_GROUP
CTEXT	Static	SS_CENTER | WS_GROUP
RTEXT	Static	SS_RIGHT | WS_GROUP
ICON	Static	SS_ICON
EDITTEXT	Edit	ES_LEFT | WS_BORDER |

		WS_STABSTOP
SCROLLBAR	Scrollbar	SBS_HORZ
LISTBOX	Listbox	LBS_NOTIFY | WS_BORDER | WS_VSCROLL
COMBOBOX	Combobox	CBS_SIMPLE | WS_TABSTOP

Các kiểu điều khiển

*Các kiểu điều khiển được khai báo trong resource script có dạng như sau, ngoại trừ kiểu điều khiển **LISTBOX**, **COMBOBOX**, **SCROLLBAR**, **EDITTEXT**.*

***Control-type** "text", id, xPos, yPos, xWidth, yHeight, iStyle*

*Các kiểu điều khiển **LISTBOX**, **COMBOBOX**, **SCROLLBAR**, **EDITTEXT** được khai báo trong **resource script** với cấu trúc như trên nhưng không có trường **"text"**.*

*Thêm thuộc tính cho các kiểu điều khiển bằng cách thay đổi tham số iStyle. Ví dụ ta muốn tạo **radio button** với chuỗi diễn đạt nằm ở bên trái của nút thì ta gán trường iStyle bằng **BS_LEFTTEXT** cụ thể như sau.*

***RADIOBUTTON** Radio1",IDC_RADIO1,106,10,53,15,BS_LEFTTEXT*

b) Thủ tục xử lý hộp thoại

❖ Đặc điểm

- Mỗi hộp thoại cần có một thủ tục xử lý riêng.
- Các thông điệp không được gửi tới hàm xử lý cửa sổ chính.
- Là một hàm xử lý cửa sổ.

❖ Mẫu hàm

```
BOOL CALLBACK Tên hàm (HWND, UINT, WPARAM, LPARAM) ;
```

- Có nhiều thông điệp khác nhau.
- Không cần xử lý WM_PAINT và WM_DESTROY.
- Xử lý thông điệp nào thì trả về TRUE, nếu không trả về FALSE.

- Thường phải xử lý hai thông điệp chính: WM_INITDIALOG và WM_COMMAND: LOWORD(WPARAM) chứa ID các điều khiển.

Ví dụ:

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM,
LPARAM);
BOOL CALLBACK DialogProc (HWND, UINT, WPARAM,
LPARAM) ;
LRESULT CALLBACK WndProc (HWND hwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
        static HINSTANCE hInstance ;
        switch (message)
        {
                case WM_CREATE :
                hInstance = ((LPCREATESTRUCT) lParam)->hInstance ;
                return 0 ;
                case WM_COMMAND :
                        switch (LOWORD (wParam))
                        {
                        case IDC_SHOW :
                        DialogBox (hInstance, TEXT ("DIALOG1"),
                        hwnd, DialogProc) ;
                        break;
                        }
                        return 0 ;
                case WM_DESTROY :
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
/*----------------------hàm xử lý thông điệp hộp thoại----------------------*/
BOOL CALLBACK DialogProc (HWND hDlg, UINT message,
WPARAM wParam, LPARAM lParam)
{
        switch (message)
        {
                case WM_INITDIALOG:  return TRUE ;
                case WM_COMMAND:
                        switch (LOWORD (wParam))
                        {
                                case IDOK :
                                EndDialog (hDlg, 0) ;
```

```
41                                  return TRUE ;
42                              }
43                              break ;
44                      }
45                      return FALSE ;
46              }
```

c) Hộp thoại trạng thái

❖ Hiển thị hộp thoại

```
INT_PTR DialogBox(
  HINSTANCE hInstance,  // handle to module
  LPCTSTR lpTemplate,   // dialog box template
  HWND hWndParent,      // handle to owner window
  DLGPROC lpDialogFunc  // dialog box procedure
);
```

Ví dụ:

DialogBox (hInstance, TEXT ("DIALOG1"), hwnd, DialogProc) ;

❖ Gởi thông điệp đến hàm **WndProc** yêu cầu xử lý ngay cả khi hộp thoại đang mở nhờ hàm **SendMessage**:

SendMessage(**GetParent**(hDlg), message, wParam, lParam);

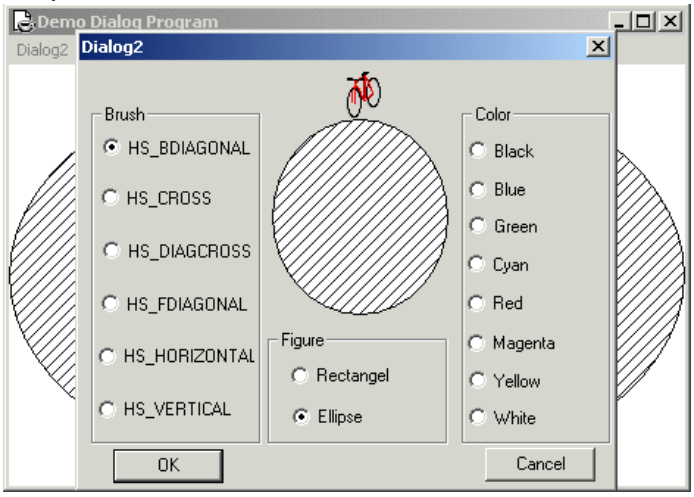
❖ Thêm tiêu đề cho hộp thoại:

SetWindowText(hDlg,**TEXT**("Hello Dialog")); trong xử lý thông điệp **WM_INITDIALOG**

❖ Đóng hộp thoại

```
BOOL EndDialog(
  HWND hDlg,        // handle to dialog box
  INT_PTR nResult   // value to return
);
```

❖ Ví dụ



```
1  LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
2  BOOL CALLBACK DialogProc (HWND, UINT, WPARAM, LPARAM);
3  int iCurrentColor = IDC_BLACK, iCurrentFigure = IDC_RECT;
4  int iCurrenBrush = IDC_HS_BDIAGONAL;
5  void PaintWindow(HWND hwnd, int iColor, int iFigure, int iBrush)
6  {
7          static COLORREF crColor[8] = { RGB(0, 0, 0), RGB(0, 0, 255),
8          RGB(0, 255, 0), RGB(0, 255, 255), RGB(255, 0, 0), RGB(255, 0, 255),
9          RGB(255, 255, 0), RGB(255, 255, 255) } ;
10         HBRUSH hBrush,hbrush;
11         HDC hdc ;
12         RECT rect ;
13         hdc = GetDC (hwnd) ;
14         GetClientRect (hwnd, &rect) ;
15         if(iBrush==IDC_HS_BDIAGONAL)
16                 hbrush=CreateHatchBrush(HS_BDIAGONAL,
17                 crColor[iColor-IDC_BLACK]);
18         if(iBrush == IDC_HS_CROSS)
19                 hbrush=CreateHatchBrush(HS_CROSS,
20                 crColor[iColor - IDC_BLACK]);
21         if(iBrush == IDC_HS_DIAGCROSS)
22                 hbrush=CreateHatchBrush(HS_DIAGCROSS,
23                 crColor[iColor - IDC_BLACK]);
24         if(iBrush == IDC_HS_FDIAGONAL)
25                 hbrush=CreateHatchBrush(HS_FDIAGONAL,
26                 crColor[iColor - IDC_BLACK]);
27         if(iBrush == IDC_HS_HORIZONTAL)
28                 hbrush=CreateHatchBrush(HS_HORIZONTAL,
29                 crColor[iColor - IDC_BLACK]);
30         if(iBrush == IDC_HS_VERTICAL)
31                 hbrush=CreateHatchBrush(HS_BDIAGONAL,
32                 crColor[iColor - IDC_BLACK]);
33                 hBrush = (HBRUSH) SelectObject (hdc, hbrush) ;
34         if (iFigure == IDC_RECT)
35                 Rectangle (hdc, rect.left, rect.top, rect.right, rect.bottom) ;
36         else
37                 Ellipse(hdc, rect.left, rect.top, rect.right, rect.bottom) ;
38         DeleteObject (SelectObject (hdc, hBrush)) ;
39         ReleaseDC (hwnd, hdc) ;
40 }
41 void PaintTheBlock(HWND hCtrl, int iColor, int iFigure, int iBrush)
42 {
43         InvalidateRect (hCtrl, NULL, TRUE) ;
44         UpdateWindow (hCtrl) ;
45         PaintWindow (hCtrl, iColor, iFigure,iBrush) ;
46 }
```

```
47  LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM
48  wParam, LPARAM lParam)
49  {
50          static HINSTANCE hInstance ;
51          PAINTSTRUCT ps ;
52          switch (message)
53          {
54          case WM_CREATE:
55                  hInstance = ((LPCREATESTRUCT) lParam)->hInstance ;
56                  return 0 ;
57                  case WM_COMMAND:
58                          switch (LOWORD (wParam))
59                          {
60                                  case IDC_SHOW:
61                                  if (DialogBox (hInstance, TEXT ("DIALOG"),
62                                  hwnd, DialogProc))
63                                          InvalidateRect (hwnd, NULL, TRUE) ;
64                                  return 0 ;
65                          }
66                          break;
67                  case WM_PAINT:
68                          BeginPaint (hwnd, &ps) ;
69                          EndPaint (hwnd, &ps) ;
70                          PaintWindow (hwnd, iCurrentColor, iCurrentFigure,
71                          iCurrenBrush) ;
72                          return 0 ;
73                  case WM_DESTROY:
74                          PostQuitMessage (0) ;
75                          return 0 ;
76          }
77          return DefWindowProc (hwnd, message, wParam, lParam) ;
78  }
79  BOOL CALLBACK DialogProc (HWND hDlg, UINT message, WPARAM
80  wParam, LPARAM lParam)
81  {
82          static HWND hCtrlBlock ;
83          static int iColor, iFigure,iBrush;
84          switch (message)
85          {
86                  case WM_INITDIALOG:
87                          iColor = iCurrentColor ;
88                          iFigure = iCurrentFigure ;
89                          iBrush = iCurrenBrush;
90                          CheckRadioButton(hDlg,IDC_BLACK,IDC_WHITE,
91                          iColor);
```

```
92                  CheckRadioButton(hDlg,IDC_RECT,IDC_ELLIPSE,iFig
93                  ure);CheckRadioButton (hDlg, IDC_HS_BDIAGONAL,
94                  IDC_HS_VERTICAL, iBrush);
95                  hCtrlBlock = GetDlgItem (hDlg, IDC_PAINT) ;
96                  SetFocus (GetDlgItem (hDlg, iColor)) ;
97                  return FALSE ;
98          case WM_COMMAND:
99          switch (LOWORD (wParam))
100         {
101                 case IDOK:
102                         iCurrentColor = iColor ;
103                         iCurrentFigure = iFigure ;
104                         iCurrenBrush = iBrush;
105                         EndDialog (hDlg, TRUE) ;
106                         return TRUE ;
107                 case IDCANCEL:
108                         EndDialog (hDlg, FALSE) ;
109                         return TRUE ;
110                 case IDC_BLACK:
111                 case IDC_RED:
112                 case IDC_GREEN:
113                 case IDC_YELLOW:
114                 case IDC_BLUE:
115                 case IDC_MAGENTA:
116                 case IDC_CYAN:
117                 case IDC_WHITE:
118                         iColor = LOWORD (wParam) ;
119                         CheckRadioButton (hDlg, IDC_BLACK,
120                         IDC_WHITE, LOWORD (wParam)) ;
121                         PaintTheBlock (hCtrlBlock, iColor,
122                         iFigure,iBrush);
123                         return TRUE ;
124                 case IDC_RECT:
125                 case IDC_ELLIPSE:
126                         iFigure = LOWORD (wParam) ;
127                         CheckRadioButton (hDlg, IDC_RECT,
128                         IDC_ELLIPSE, LOWORD (wParam)) ;
129                         PaintTheBlock (hCtrlBlock, iColor,
130                         iFigure,iBrush);
131                         return TRUE ;
132                 case IDC_HS_BDIAGONAL:
133                 case IDC_HS_CROSS:
134                 case IDC_HS_DIAGCROSS:
135                 case IDC_HS_FDIAGONAL:
136                 case IDC_HS_HORIZONTAL:
137                 case IDC_HS_VERTICAL:
```

```
                    iBrush = LOWORD (wParam)
                    CheckRadioButton(hDlg,IDC_HS_BDIAGONAL,
                    IDC_HS_VERTICAL, LOWORD (wParam)) ;
                    PaintTheBlock (hCtrlBlock, iColor,
                    iFigure,iBrush);
                    return TRUE ;
                }
                break;
        case WM_PAINT:
                PaintTheBlock (hCtrlBlock, iColor, iFigure,iBrush) ;
                break ;
        }
        return FALSE ;
}
```

d) Hộp thoại không trạng thái

❖ Hiển thị hộp thoại

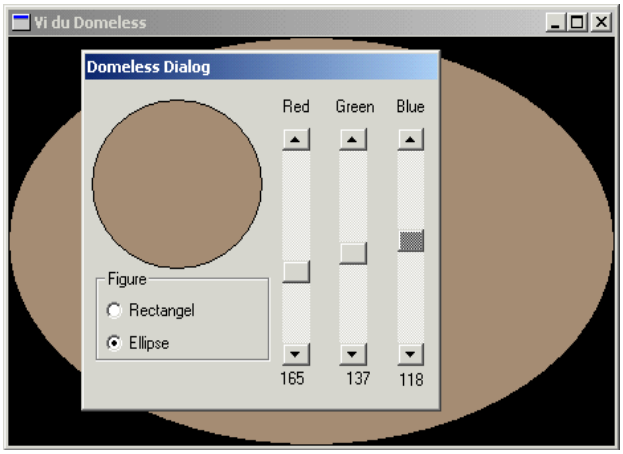
```
HWND hDlgModeless=CreateDialog(hInstance, szTemplate,
hwndParent, DialogProc);
ShowWindow(hDlgModeless,SW_SHOW);

while(GetMessage(&msg, NULL, 0, 0))
{
        if (hDlgModeless==0 || !IsDialogMessage
        (hDlgModeless, &msg);
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
}

while(GetMessage(&msg, NULL, 0, 0))
{
        if (hDlgModeless==0 || !IsDialogMessage(hDlgModeless,
        &msg);
        {
                if(TranslateAccelerator (hwnd, hAccel, &msg)
                {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
        }
}
```

❖ Đóng hộp thoại

Đặt hDlgModeless về giá trị 0.

```
BOOL DestroyWindow(
  HWND hWnd   // handle to window to destroy
);
```

❖ Ví dụ




```
1   void PaintWindow (HWND hwnd, int iColor[], int iFigure)
2   {
3       HBRUSH hBrush ;
4       HDC hdc ;
5       RECT rect ;
6       hdc = GetDC(hwnd) ;
7       GetClientRect (hwnd, &rect) ;
8       hBrush = CreateSolidBrush(RGB(iColor[0], iColor[1],
9       iColor[2]));
10      hBrush = (HBRUSH) SelectObject (hdc, hBrush) ;
11      if (iFigure == IDC_RECT)
12          Rectangle (hdc, rect.left, rect.top, rect.right,
13          rect.bottom) ;
14      else
15          Ellipse(hdc, rect.left, rect.top, rect.right,
16          rect.bottom) ;
17      DeleteObject (SelectObject (hdc, hBrush)) ;
18      ReleaseDC (hwnd, hdc) ;
19  }
20  LRESULT CALLBACK WndProc (HWND hwnd, UINT
21  message, WPARAM wParam, LPARAM lParam)
22  {
23      switch (message)
24      {
25          case WM_PAINT:
26              PaintTheBlock(hwnd, iColor, iFigure) ;
27              return 0 ;
28          case WM_DESTROY :
```

```
29                              DeleteObject((HGDIOBJ)SetClassLong(hw
30                              nd, GCL_HBRBACKGROUND,(LONG)
31                              GetStockObject (WHITE_BRUSH))) ;
32                              PostQuitMessage (0) ;
33                              return 0 ;
34                      }
35                      return DefWindowProc (hwnd, message, wParam,
36                      lParam);
37              }
38              void PaintTheBlock (HWND hCtrl, int iColor[], int iFigure)
39              {
40                      InvalidateRect (hCtrl, NULL, TRUE);
41                      UpdateWindow (hCtrl) ;
42                      PaintWindow (hCtrl, iColor, iFigure) ;
43              }
44              BOOL CALLBACK ColorScrDlg (HWND hDlg, UINT
45              message, WPARAM wParam, LPARAM lParam)
46              {
47                      HWND hwndParent, hCtrl ;
48                      static HWND hCtrlBlock ;
49                      int iCtrlID, iIndex ;
50                      switch (message)
51                      {
52                              case WM_INITDIALOG :
53                                      hCtrlBlock = GetDlgItem (hDlg,
54                                      IDC_PAINT) ;
55                                      for (iCtrlID = 10 ; iCtrlID < 13 ; iCtrlID++)
56                                      {
57                                              hCtrl = GetDlgItem (hDlg, iCtrlID) ;
58                                              PaintTheBlock (hCtrlBlock, iColor,
59                                              iFigure) ;
60                                              PaintTheBlock (hwndParent, iColor,
61                                              iFigure) ;
62                                              SetScrollRange (hCtrl, SB_CTL, 0,
63                                              255, FALSE) ;
64                                              SetScrollPos(hCtrl, SB_CTL, 0,
65                                              FALSE) ;
66                                      }
67                                      return TRUE ;
68                              case WM_COMMAND:
69                                      {
70                                      switch( LOWORD(wParam))
71                                      {
72                                              case IDC_RECT:
73                                              case IDC_ELLIPSE:
74                                                      iFigure = LOWORD(wParam);
```

```
                                        hwndParent =
                                        GetParent(hDlg);
                                        CheckRadioButton(hDlg,
                                        IDC_RECT, IDC_ELLIPSE,
                                        LOWORD (wParam)) ;
                                        PaintTheBlock(hCtrlBlock,
                                        iColor, iFigure) ;
                                        PaintTheBlock (hwndParent,
                                        iColor, iFigure) ;
                                        return TRUE ;
                        }
                        break;
                }
                case WM_VSCROLL :
                        hCtrl = (HWND) lParam ;
                        iCtrlID = GetWindowLong (hCtrl,
                        GWL_ID) ;
                        iIndex = iCtrlID - 10 ;
                        hwndParent = GetParent (hDlg) ;
                        PaintTheBlock (hCtrlBlock, iColor, iFigure);
                        PaintTheBlock (hwndParent, iColor,
                        iFigure) ;
                        switch (LOWORD (wParam))
                        {
                                case SB_PAGEDOWN :
                                        iColor[iIndex] += 15 ;
                                case SB_LINEDOWN :
                                        iColor[iIndex] = min (255,
                                        iColor[iIndex] + 1) ;
                                        break;
                                case SB_PAGEUP :
                                        iColor[iIndex] -= 15 ;
                                        case SB_LINEUP :
                                        iColor[iIndex] = max (0,
                                        iColor[iIndex] - 1);
                                        break;
                                case SB_TOP :
                                        iColor[iIndex] = 0 ;
                                        break;
                                case SB_BOTTOM :
                                        iColor[iIndex] = 255 ;
                                        break;
                                case SB_THUMBPOSITION :
                                case SB_THUMBTRACK :
                                        iColor[iIndex] = HIWORD
                                        (wParam) ;
```

```
            MENUITEM "&White", IDM_BKGND_WHITE,
            CHECKED
            MENUITEM "&Light Gray", IDM_BKGND_LTGRAY
            MENUITEM "&Gray", IDM_BKGND_GRAY
            MENUITEM "&Dark Gray", IDM_BKGND_DKGRAY
            MENUITEM "&Black", IDM_BKGND_BLACK
        END
        POPUP "&Help"
        BEGIN
            MENUITEM "&Help...", IDM_APP_HELP
            MENUITEM "&About ...", IDM_APP_ABOUT
        END
    END
```

b) Thiết lập Menu

wndclass.lpszMenuName = "MENU1";

hoặc:

hMenu = **LoadMenu** (hInstance, **TEXT**("MENU1"));

hwnd = **CreateWindow** (**TEXT**("MyClass"), **TEXT**("Window Caption"), WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, NULL, hMenu, hInstance, NULL);

SetMenu(hWnd, hMenu);

LOWORD(WPARAM) chứa ID các điều khiển.

c) Ví dụ

```
1   LRESULT CALLBACK WndProc (HWND, UINT, WPARAM,
2   LPARAM);
3   /* Khai báo tên dùng chung cho cáctài nguyên trong chương trình.*/
4   TCHAR szAppName[] = TEXT ("MenuDemo") ;
5   int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
6   hPrevInstance, PSTR szCmdLine, int iCmdShow)
7   {
8       HWND hwnd;
9       MSG msg;
10      WNDCLASS wndclass;
11      wndclass.style = CS_HREDRAW | CS_VREDRAW;
12      wndclass.lpfnWndProc = WndProc ;
13      wndclass.cbClsExtra = 0 ;
14      wndclass.cbWndExtra = 0 ;
15      wndclass.hInstance = hInstance ;
16      wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
```

```
17              wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
18              wndclass.hbrBackground =
19              (HBRUSH)GetStockObject(WHITE_BRUSH) ;
20              wndclass.lpszMenuName = szAppName ;
21              wndclass.lpszClassName = szAppName ;
22              if (!RegisterClass (&wndclass))
23              {
24                      MessageBox(NULL, TEXT("This program requires
25                      Windows "), szAppName, MB_ICONERROR) ;
26                      return 0 ;
27              }
28              hwnd = CreateWindow (szAppName, TEXT("Menu
29              Demonstration"), WS_OVERLAPPEDWINDOW,
30              CW_USEDEFAULT, CW_USEDEFAULT,
31              CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL,
32              hInstance, NULL) ;
33              ShowWindow (hwnd, iCmdShow) ;
34              UpdateWindow (hwnd) ;
35              while (GetMessage(&msg, NULL, 0, 0))
36              {
37                      TranslateMessage (&msg) ;
38                      DispatchMessage (&msg) ;
39              }
40              return msg.wParam ;
41      }
42      LRESULT CALLBACK WndProc (HWND hwnd, UINT message,
43      WPARAM wParam, LPARAM lParam)
44      {
45          /* Khao báo danh sách các màu chổi tô, các hằng này được định
46          nghĩa trong file WINGDI.H */
47          static int idColor[5] = { WHITE_BRUSH, LTGRAY_BRUSH,
48          GRAY_BRUSH, DKGRAY_BRUSH, BLACK_BRUSH } ;
49          static int iSelection = IDM_BKGND_WHITE ;
50          HMENU hMenu ;
51          switch (message)
52          {
53                  case WM_COMMAND:
54                  hMenu = GetMenu (hwnd) ; // Lấy định danh của menu
55                  switch (LOWORD (wParam)) //Kiểm tra định danh mục chọn
56                  {
57                          case IDM_FILE_NEW:
58                          case IDM_FILE_OPEN:
59                          case IDM_FILE_SAVE:
60                          case IDM_FILE_SAVE_AS:
61                                  MessageBeep(0) ; //Phát ra tiếng kêu bíp
62                                  return 0 ;
```

```
63                      case IDM_APP_EXIT:
64                      /*Gởi thông điệp để đóng ứng dụng lại*/
65                              SendMessage (hwnd, WM_CLOSE, 0, 0) ;
66                              return 0 ;
67                      case IDM_EDIT_UNDO:
68                      case IDM_EDIT_CUT:
69                      case IDM_EDIT_COPY:
70                      case IDM_EDIT_PASTE:
71                      case IDM_EDIT_CLEAR:
72                              MessageBeep (0) ;
73                              return 0 ;
74                      case IDM_BKGND_WHITE:
75                      case IDM_BKGND_LTGRAY:
76                      case IDM_BKGND_GRAY:
77                      case IDM_BKGND_DKGRAY:
78                      case IDM_BKGND_BLACK:
79                              /* Bỏ check của mục chọn trước đó*/
80                              CheckMenuItem(hMenu,iSelection,
81                              MF_UNCHECKED);
82                              iSelection = LOWORD (wParam) ; /*Lấy ID
83                              mục mới*/
84                              /* Check mục chọn mới*/
85                              CheckMenuItem (hMenu, iSelection,
86                              MF_CHECKED) ;
87                              /* Thiết lập màu tương ứng với mục chọn
88                              mới*/
89                              SetClassLong(hwnd,GCL_HBRBACKGRO
90                              UND, (LONG)
91                              GetStockObject(idColor[iSelection-
92                              IDM_BKGND_WHITE]));
93                              InvalidateRect (hwnd, NULL, TRUE) ;
94                              return 0 ;
95                      case IDM_APP_HELP:
96                              MessageBox(hwnd, TEXT("Help not yet
97                              implemented!"), szAppName,
98                              MB_ICONEXCLAMATION | MB_OK) ;
99                              return 0 ;
100                     case IDM_APP_ABOUT:
101                             MessageBox (hwnd, TEXT ("Menu
102                             Demonstration Program\n (c) Charles
103                             Petzold, 1998"), szAppName,
104                             MB_ICONINFORMATION | MB_OK) ;
105                             return 0 ;
106             }
107                     break;
108             case WM_DESTROY:
```

```
109                         PostQuitMessage(0) ;
110                         return 0 ;
111                 }
112                 return DefWindowProc(hwnd, message, wParam, lParam) ;
113         }
```

lpnTabStopPositions bằng 30, ta sẽ có dãy tab dừng tại vị trí 30, 60, 90, ... pixels.

Trường nTabOrigin xác định tọa độ theo trục x của điểm bắt đầu tính khoảng cách tới các tab. Giá trị này không nhất thiết phải là vị trí đầu tiên của chuỗi, có thể chọn trùng hoặc không.

Hàm trả về kích thước chuỗi hiển thị, theo đơn vị logic, nếu thành công. Ngược lại, hàm trả về 0. Trong đó, chiều cao chuỗi là WORD cao của biến kiểu LONG, chiều rộng là WORD thấp.

- *int DrawText(HDC hDC, LPCTSTR lpString, int nCount, LPRECT lpRect, UINT uFormat);*

  Cũng như các hàm xuất văn bản khác, hàm DrawText xuất chuỗi xác định bởi con trỏ lpString có độ dài nCount. Tuy nhiên, với chuỗi có ký tự kết thúc là NULL, nếu nCount bằng -1, hàm sẽ tự động tính toán chiều dài của chuỗi.

  Biến lpRect trỏ đến cấu trúc RECT của hình chữ nhật (theo toạ độ logic) mà trong đó văn bản thể hiện theo định dạng được thiết lập trong uFormat.

  Nếu uFormat bằng 0, nội dung văn bản sẽ được hiển thị theo từng dòng từ trên xuống dưới. Mỗi dòng mới được xác định thông qua ký tự về đầu dòng CR (carriage return, bằng '\r' hoặc 0x0D) hoặc ký tự xuống dòng LF (linefeed, bằng '\n' hoặc 0x0A) có trong văn bản. Phần văn bản bên ngoài hình chữ nhật lpRect sẽ bị cắt bỏ.

  Giá trị uFormat bằng 0 cũng chính là giá trị cờ canh lề trái (DT_LEFT). Ngoài ra, ta có thể thiết lập các cờ canh lề phải (DT_RIGHT), và canh lề giữa (DT_CENTER) cho văn bản.

  Để loại bỏ chức năng điều khiển của các ký tự CR và LF, cần thêm vào cờ DT_SINGLELINE. Nếu thiết lập DT_SINGLELINE, ta cũng có thể chỉ định vị trí của dòng hiển thị ở phía trên (DT_TOP), phía dưới (DT_BOTTOM), hoặc ở chính giữa (DT_VCENTER) trong vùng hình chữ nhật.

	bị. Tuy nhiên, đối với một số thiết bị, font được cài đặt ngay trên thiết bị. Ví dụ, đối với máy in, các font thiết bị cài sẵn thực hiện thao tác in nhanh hơn so với việc load bitmap ảnh về từ máy tính.
DEFAULT_GUI_FONT	Font của giao diện đồ họa được thiết lập mặc định.
OEM_FIXED_FONT	Font chữ cố định, dựa trên bộ ký tự OEM. Ví dụ, đối với máy IBM®, font OEM dựa trên bộ ký tự IBM PC.
SYSTEM_FONT	Font hệ thống của Windows. Được hệ điều hành dùng để trình bày các thành phần giao diện như thanh tiêu đề, menu, nội dung văn bản trong các hộp thoại thông điệp. Các font hệ thống này luôn có sẵn khi cài hệ điều hành, trong khi các font khác cần phải cài thêm tùy theo ứng dụng sau này.
SYSTEM_FIXED_FONT	Font Windows được sử dụng như font hệ thống trong các phiên bản trước 3.0.

Macro các font định nghĩa sẵn.

- Nạp: HGDIOBJ GetStockObject(int fnObject) → Nếu thành công, trả về handle font chữ. Ngược lại, giá trị trả về là NULL.

  Trong đó, kiểu HGDIOBJ là HFONT, biến fnObject là một trong các macro ở bảng trên.
- Gán chỉ số cho DC: HGDIOBJ SelectObject(HDC hDC, HGDIOBJ hGDIObj) → Trả về handle font chữ vừa sử dụng trước, lỗi trả về GDI_ERROR

Hoặc gọn hơn, ta có thể gọi :

SelectObject(hDC.**GetStockObject**(fnObject));

DeleteObject (Đối tượng): để hủy.

Ví dụ:

```
HFONT hfnt, hOldFont;
hfnt = GetStockObject(ANSI_VAR_FONT);
if (hOldFont = SelectObject(hdc, hfnt))
{
```

```
        TextOut(hdc, 10, 50, "Sample ANSI_VAR_FONT text.", 26);
        SelectObject(hdc, hOldFont);
}
```

❖ Xác định kích thước font

BOOL GetTextMetrics(HDC hdc, LPTEXTMETRIC lptm);

```
typedef struct tagTEXTMETRIC // tm
{
        LONG tmHeight;
        LONG tmAscent;
        LONG tmDescent;
        LONG tmInternalLeading;
        LONG tmExternalLeading;
        LONG tmAveCharWidth;
        LONG tmMaxCharWidth;
        LONG tmWeight;
        LONG tmOverhang;
        LONG tmDigitizedAspectX;
        LONG tmDigitizedAspectY;
        BCHAR tmFirstChar;
        BCHAR tmLastChar;
        BCHAR tmDefaultChar;
        BCHAR tmBreakChar;
        BYTE tmItalic;
        BYTE tmUnderlined;
        BYTE tmStruckOut;
        BYTE tmPitchAndFamily;
        BYTE tmCharSet;
} TEXTMETRIC;
```

Cấu trúc TEXTMETRIC gồm 20 thành phần, một số thành phần quan trọng gồm:

- tmHeight: Chiều cao ký tự tính bằng pixel.
- tmInternalLeading: Vùng chứa dấu trọng âm.
- tmExternalLeading: Không gian giữa 2 dòng.
- tmAveCharWidth: Bề rộng trung bình mỗi ký tự.
- tmPitchAndFamily: Họ c*ủa font (8 bit).*

Ví dụ:

```
static int cxchar, cychar;
```

```
TEXTMETRIC tm;
case WM_CREATE:
    {
            hdc = GetDC(hwnd);
            GetTextMetrics(hdc, &tm);
            cxchar=tm.tmInternalLeading+tm.tmExternal;
            cychar=tm.tmAveCharWidth;
            ReleaseDC(hwnd, hdc);
            return 0;
    }
case WM_PAINT:
{
        for(int i=0; i<10; i++)
            TextOut(hdc, cxchar, cychar*i, "aaa", 3);
    }
```

❖ Tính độ dài của xâu ký tự

- Các ký tự hiển thị có bề rộng khác nhau do vậy không nên dùng hàm strlen() để lấy số ký tự → độ dài.
- Dùng hàm: BOOL GetTextExtentPoint32 (HDC hdc, LPCSTR lpszString, int len, LPSIZE lpSize);

```
typedef struct tagSIZE
{
        long cx;
        long cy;        //Tính theo đơn vị logic
} SIZE;
```

len: Tổng số ký tự.

Tạo lập đặc tính mới cho font chữ

HFONT CreateFont (int Height, int Width, int Escapement, int Orientation, int fnWeight, DWORD Italic, DWORD Underline, DWORD StrikeOut, DWORD CharSet, DWORD outputPrecision, DWORD ClipPrecision, DWORD Quality, DWORD PitchAndFamily, LPCSTR lpszFontName)

Với: