Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

GIỚI THIỆU MÔN HỌC

Giới thiệu chung

- Đối tượng: Sinh viên năm nhất
- Thời gian (15 tuần): 45 tiết LT + 45 tiết TH
- Môn học tiên quyết: Không có
- Hình thức kiểm tra: LT (7đ), TH (1đ + 2đ)
- Giảng viên lý thuyết
  - Đặng Bình Phương dbphuong@fit.hcmus.edu.vn
    dang2@vnn.vn
- Nhóm giảng viên hướng dẫn thực hành
  - ...

Nội dung môn học

- Tuần 1: Các khái niệm cơ bản về lập trình
  - Các khái niệm cơ bản: thuật toán, lưu đồ, …
  - Giới thiệu ngôn ngữ lập trình C.
  - Cấu trúc một chương trình viết bằng ngôn ngữ lập trình cấp cao (C/C++).
- Tuần 2:
  - Kiểu dữ liệu và các phép toán số học, luận lý.
  - Nhập xuất dữ liệu.

Nội dung môn học

- Tuần 3: Cấu trúc điều kiện, rẽ nhánh
  - if … else …
  - switch
- Tuần 4: Cấu trúc lặp
  - for
  - while
  - do … while …

Nội dung môn học

- Tuần 5, 6: Chương trình con
  - Khái niệm về chương trình con.
  - Chương trình con trong NNLT C/C++: Hàm.
  - Tham số và truyền tham số (Call-by-Value).

- Tuần 7, 8: Kiểu dữ liệu mảng
  - Mảng 1 chiều, 2 chiều(tham khảo).
  - Kỹ thuật lập trình với mảng 1 chiều.

Nội dung môn học

- Tuần 9, 10: Biến con trỏ và các kỹ thuật cơ bản
  - Khái niệm con trỏ, địa chỉ vùng nhớ.
  - Các phép toán số học trên con trỏ.
  - Con trỏ và mảng một chiều.
  - Cấp phát bộ nhớ động.

- Tuần 11, 12: Kỹ thuật lập trình trên chuỗi ký tự
  - Ký tự và chuỗi.
  - Các hàm cơ bản trên chuỗi ký tự.

Nội dung môn học

- Tuần 13, 14: Kiểu dữ liệu cấu trúc
  - Khái niệm.
  - Truy xuất và truyền cấu trúc cho hàm.
  - Mảng cấu trúc.

- Tuần 15: Một số kỹ thuật lập trình hữu ích khác
  - Kỹ thuật lập trình đệ quy cơ bản.
  - Kỹ thuật lập trình với tập tin.
  - Kỹ thuật lập trình trên bit.

Tài liệu tham khảo

- Tự học lập trình C trong 21 ngày, NXB Đà Nẵng.
- Slides bài giảng, code mẫu, tài liệu tham khảo: www.mediafire.com/**dang2**

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KHÁI NIỆM CƠ BẢN VỀ MÁY TÍNH

Nội dung

1. Vài nét lịch sử máy tính
2. Các thế hệ máy tính điện tử
3. Phân loại
4. Các thành phần cơ bản

Vài nét lịch sử máy tính

1642

Blaise Pascal (1623 – 1662)
Máy cộng cơ học đầu tiên trên thế giới

1670

Gottfried Leibritz (1646 – 1716)
Cải tiến máy của Pascal để +, -, *, /

1833

Charle Babbage
Không nên phát triển máy cơ học
Máy tính với chương trình bên ngoài

1945

John von Neumann
Nguyên lý có tính chất quyết định
. Chương trình lưu trữ trong máy
. Sự gián đoạn quá trình tuần tự

5 thế hệ máy tính điện tử

Thế hệ thứ nhất (1950 – 1958)
Sử dụng đèn chân không
Tốc độ thấp: 10^3 phép tính/s
Chtrình viết bằng ngôn ngữ máy
Máy ENIAC nặng 30 tấn!

5 thế hệ máy tính điện tử

Thế hệ thứ hai (1959 – 1963)
Sử dụng đèn bán dẫn
Tốc độ nhanh: 10^6 phép tính/s
Chtrình viết bằng COBOL, ALGOL
Máy IBM151 (Mỹ), MINSK22 (LX)

5 thế hệ máy tính điện tử

Thế hệ thứ ba (1964 – 1977)
Sử dụng mạch tích hợp IC
Tốc độ cao: 10^9 tính toán/s
Ngôn ngữ lập trình cấp cao & các phần mềm ứng dụng
IBM360 (Mỹ), MINSK32 (LX)

5 thế hệ máy tính điện tử

Thế hệ thứ tư (1978 - 1983)

Mạch tích hợp quy mô lớn LSI

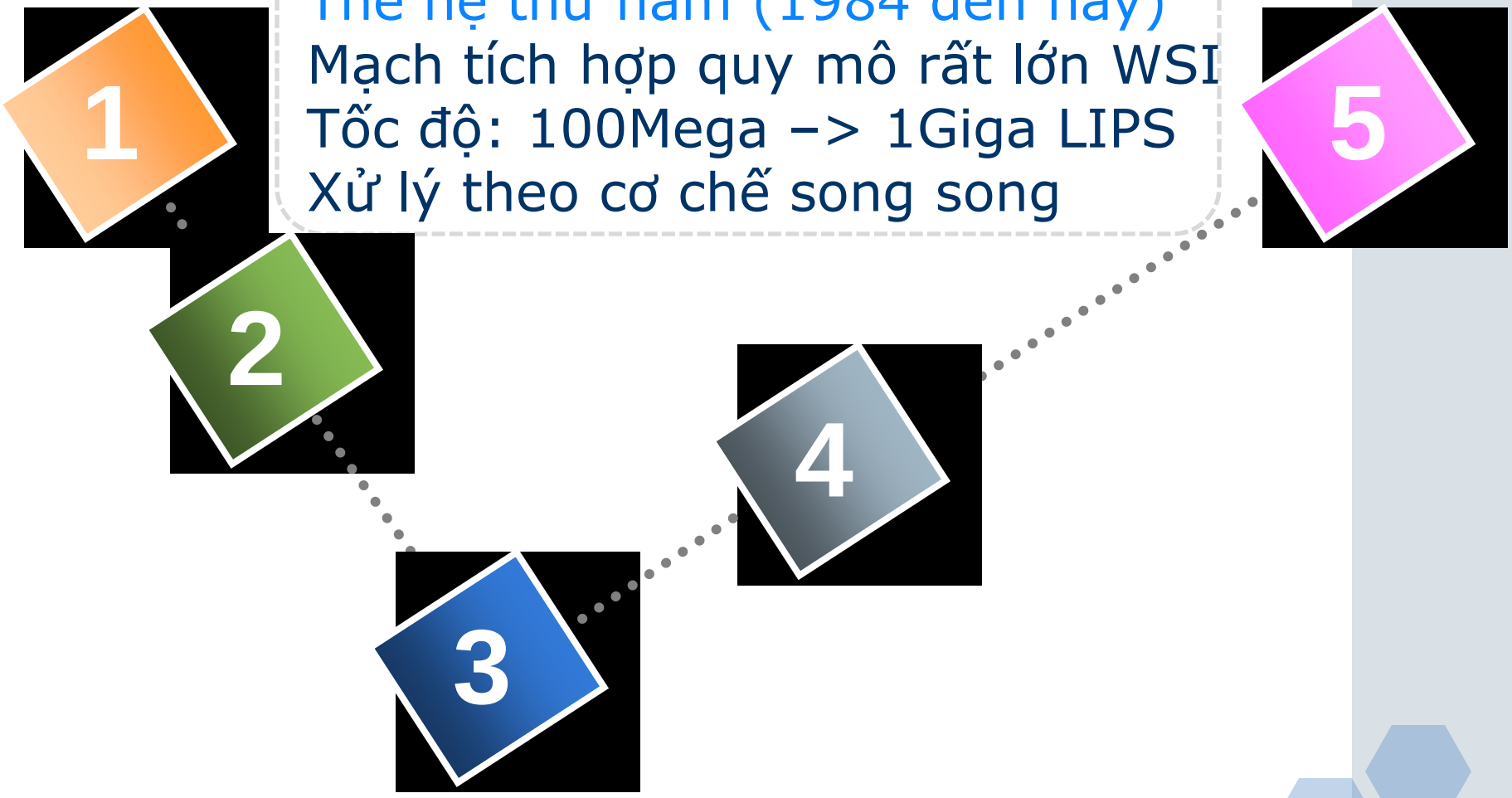
Tốc độ cao: 10^{12} phép tính/s

Nhỏ gọn và bộ nhớ tăng dần

Phần mềm phong phú, đa dạng

Mạng máy tính ra đời

5 thế hệ máy tính điện tử



Thế hệ thứ năm (1984 đến nay)
Mạch tích hợp quy mô rất lớn WSI
Tốc độ: 100Mega –> 1Giga LIPS
Xử lý theo cơ chế song song

Phân loại

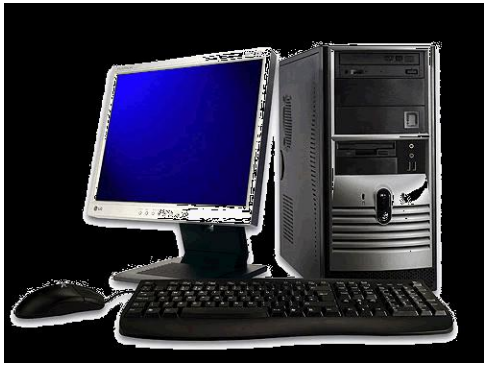
Máy tính lớn (Mainframe)

Kích thước vật lý lớn.
Thực hiện hàng tỉ phép tính/s
Phục vụ tính toán phức tạp.
Trong cơ quan nhà nước.

Siêu máy tính (Super Computer)

Nhiều máy lớn ghép song song.
Tốc độ tính toán cực lớn.
Dùng trong lĩnh vực đặc biệt như quân sự, vũ trụ.

Phân loại



Máy tính cá nhân
(Personal Computer - PC)
Còn gọi là máy tính để bàn (Desktop)
Dùng ở văn phòng, gia đình.

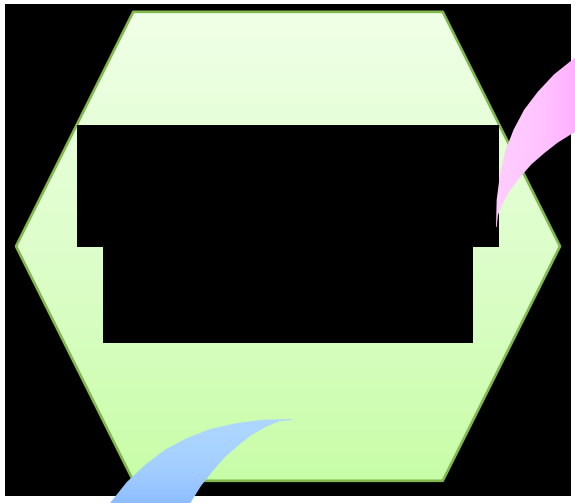
Máy tính xách tay (Laptop)
Còn gọi là “Notebook”.
Là loại máy tính nhỏ, có thể mang theo người.
Chạy bằng pin.

Phân loại

Máy tính bỏ túi (Pocket PC)

Thiết bị kỹ thuật số cá nhân có chức năng rất phong phú như kiểm tra email, xem phim, nghe nhạc, duyệt web. Nhiều máy còn tính hợp chức năng điện thoại di động.

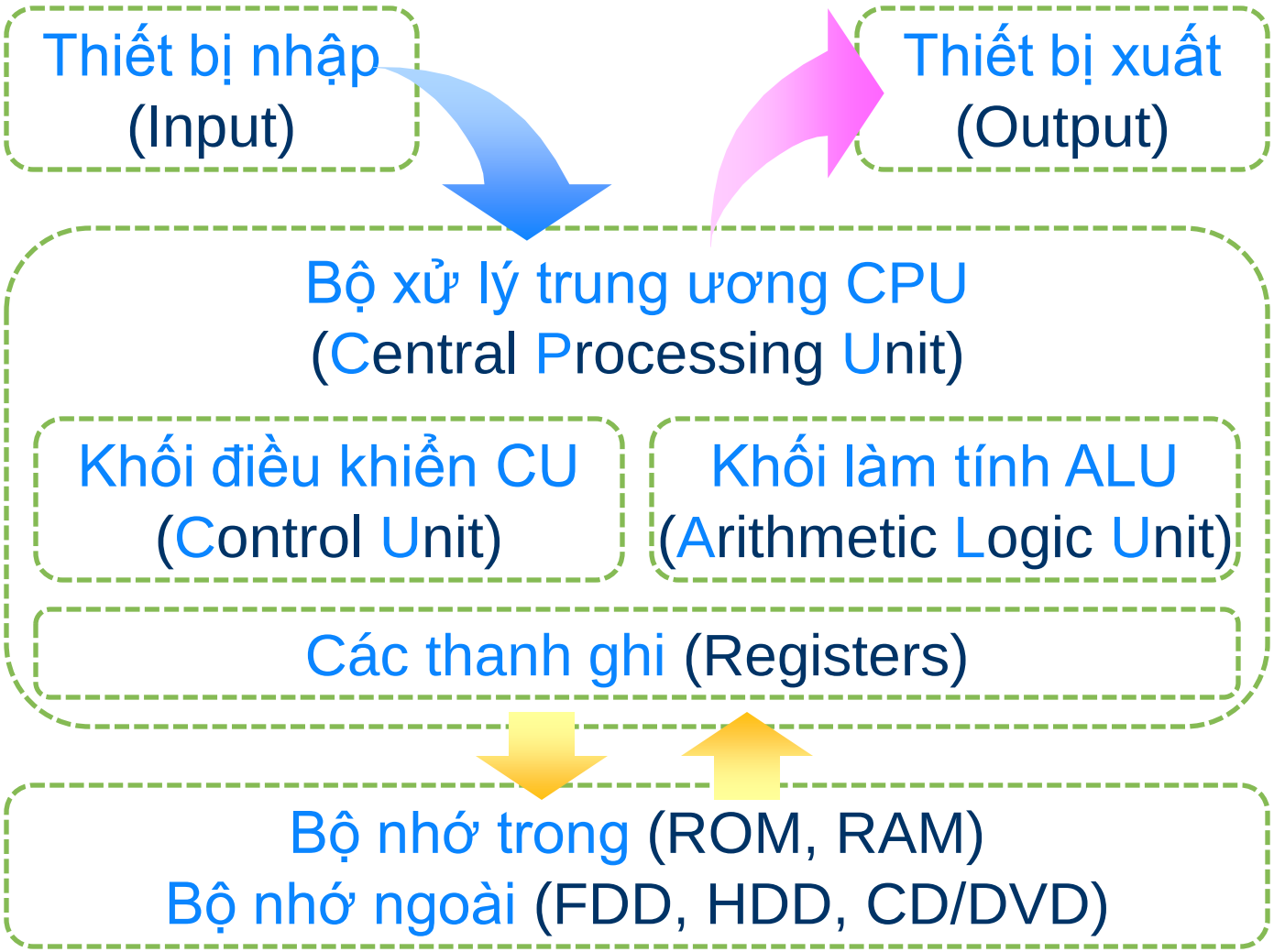
Các thành phần cơ bản



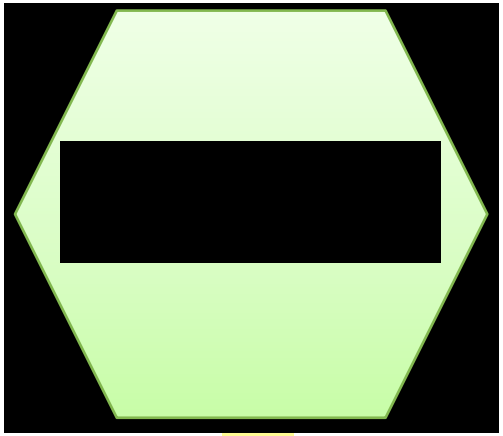
Phần mềm (Software)
- Phần mềm hệ thống
- Phần mềm ứng dụng

Phần cứng (Hardware)
- Bộ nhớ (Memory)
- Đơn vị xử lý trung ương CPU (Central Processing Unit)
- Thiết bị nhập xuất (Input/Ouput Device).

Phần cứng - Cấu trúc

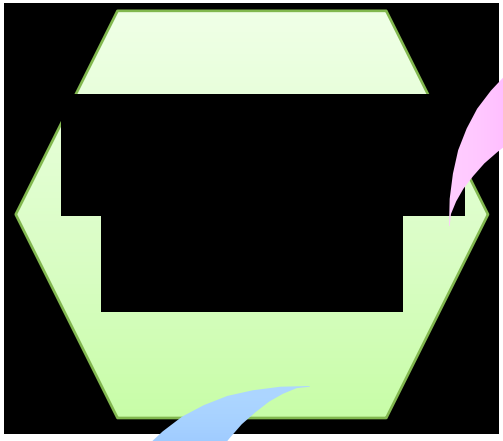


Phần cứng - Bộ nhớ



Bộ nhớ (Memory)
Thiết bị lưu trữ thông tin trong quá trình máy tính xử lý.

Phần cứng - Bộ nhớ trong



ROM (Read Only Memory)
- Chỉ đọc thông tin
- Lưu chương trình hệ thống
- Không mất khi mất điện.

RAM (Random Access Memory)
- Bộ nhớ truy xuất ngẫu nhiên.
- Bị mất khi mất điện.

Phần cứng - Bộ nhớ ngoài

Đĩa mềm (Floppy Disk)
Đường kính 3.5”
Dung lượng 1.44 MB.

Đĩa cứng (Hard Disk)
Dung lượng lớn khoảng:
20 GB, 30 GB, 750 GB...

Đĩa quang (Compact Disk)
CD (700 MB)
DVD (4.7 GB)

Phần cứng - Bộ nhớ ngoài

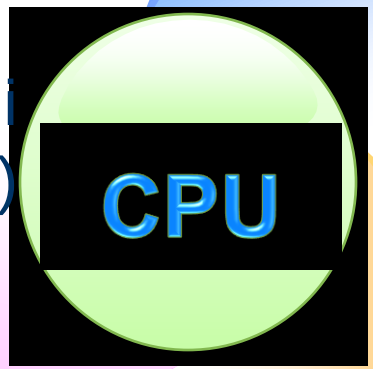
Thẻ nhớ (Memory Stick hay Compact Flash Card)
Dung lượng khoảng
32 MB, 64 MB, 128 MB…

USB Flash Drive
Dung lượng khoảng
256 MB, 512 MB, 1GB…

Phần cứng - CPU

Khối điều khiển
(CU – Control Unit)

Các thanh ghi
(Registers)



Khối tính toán số học và logic
(ALU – Arithmetic Logic Unit)

Phần cứng - CPU

- Đơn vị xử lý trung ương CPU:
  - Gắn với một đồng hồ (clock) hay còn gọi là bộ xung nhịp. Tần số đồng hồ càng cao thì tốc độ xử lý thông tin càng nhanh.
  - Pentium 4/D, Dual Core, Core 2 Duo, Core 2 Quad. Tốc độ: 2.0 GHz, …, 3.0 GHz…

Phần cứng - Thiết bị nhập

Bàn phím (Keyboard)
Nhập dữ liệu và câu lệnh
Loại phổ biến có 104 phím

Chuột (Mouse)
Kích thước vừa nắm tay
Dùng để di chuyển
con trỏ chuột
trong môi trường đồ họa.

Phần cứng - Thiết bị nhập

Máy quét hình (Scanner)
Nhập văn bản hay hình vẽ, hình chụp vào máy tính.

Camera & Webcam
Quay hình ảnh bên ngoài đưa vào máy tính

Máy chụp hình kỹ thuật số
Chụp hình ảnh bên ngoài đưa vào máy tính.

Phần cứng - Thiết bị xuất

Màn hình (Screen hay Moniter)
Thể hiện thông tin ra màn hình bằng kỹ thuật ánh xạ bộ nhớ (memory mapping)
Các loại màn hình phổ biến hiện nay là SVGA 15”, 17”, 19”…

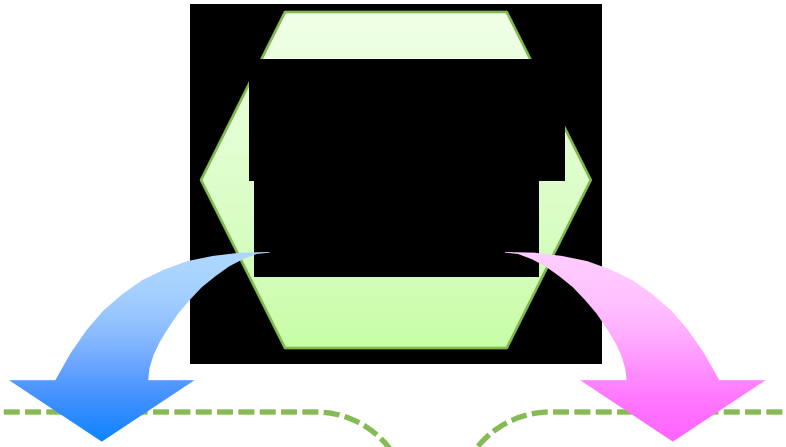
Phần cứng - Thiết bị xuất

Máy chiếu (Projector)
Tương tự như màn hình nhưng phóng to hình ảnh.

Máy in (Printer)
Xuất thông tin ra giấy.

Loa (Speaker)
Phát âm thanh.

Phần mềm



Phần mềm hệ thống

- Hệ điều hành (OS)
- PM đi kèm thiết bị phần cứng (Driver)
- Ví dụ: MSDOS, Linux, Windows…

Phần mềm ứng dụng

- Soạn thảo văn bản
- Tính toán, phân tích
- Đồ họa
- Bảo mật
- Trò chơi

Bài tập

1. Nêu vài nét lịch sử phát triển máy tính và phân loại máy tính điện tử.
2. Mô tả cấu tạo và chức năng CPU?
3. Phân biệt bộ nhớ trong và bộ nhớ ngoài. Kể tên và mô tả một số bộ nhớ ngoài mà bạn biết.
4. Kể tên và mô tả một số thiết bị nhập và thiết bị xuất mà bạn biết.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

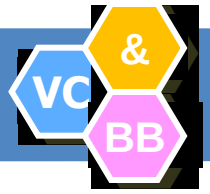
NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmuns.edu.vn

GIỚI THIỆU
NGÔN NGỮ LẬP TRÌNH C

Nội dung

1. Giới thiệu
2. Bộ từ vựng của C
3. Cấu trúc chương trình C
4. Một số ví dụ minh họa



Giới thiệu

Giới thiệu

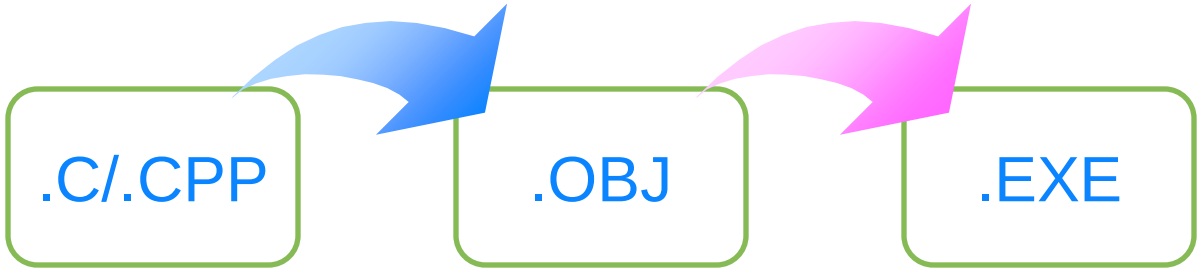
- Dennis Ritchie tại Bell Telephone năm 1972.
- Tiền thân của ngôn ngữ B, KenThompson, cũng tại Bell Telephone.
- Là ngôn ngữ lập trình có cấu trúc và phân biệt chữ Hoa - thường (case sensitive)
- ANSI C.

Giới thiệu

- Ưu điểm của C
  - Rất mạnh và linh động, có khả năng thể hiện bất cứ ý tưởng nào.
  - Được sử dụng rộng rãi bởi các nhà lập trình chuyên nghiệp.
  - Có tính khả chuyển, ít thay đổi trên các hệ thống máy tính khác nhau.
  - Rõ ràng, cô đọng.
  - Lập trình đơn thể, tái sử dụng thông qua hàm.

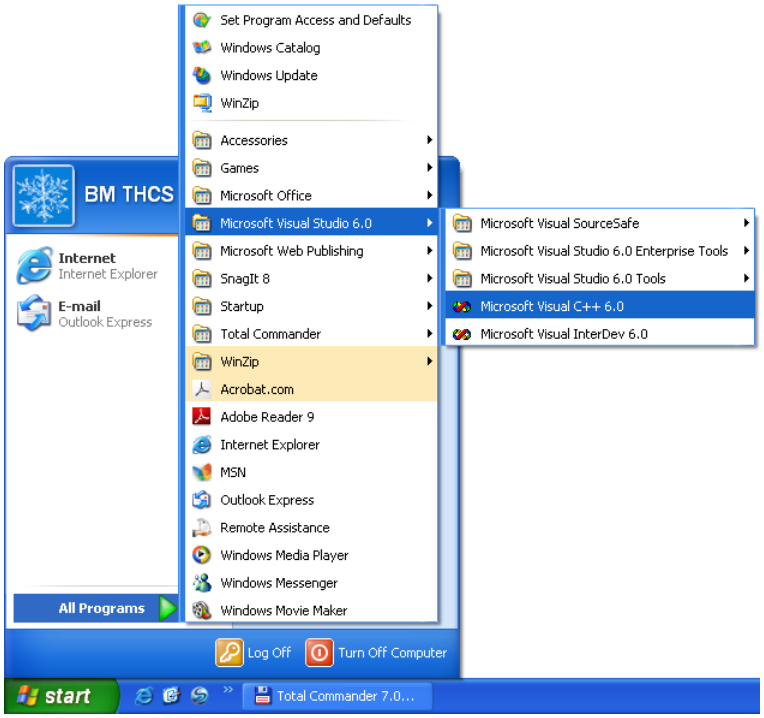
Giới thiệu

- Môi trường phát triển tích hợp IDE (Integrated Development Environment)
  - Biên tập chương trình nguồn (Trình EDIT).
  - Biên dịch chương trình (Trình COMPILE).
  - Chạy chương trình nguồn (Trình RUNTIME).
  - Sửa lỗi chương trình nguồn (Trình DEBUG).



Giới thiệu

- Môi trường lập trình
  - Borland C++ 3.1 for DOS.
  - Visual C++ 6.0, Win32 Console Application.



Bộ từ vựng của C

- Các ký tự được sử dụng
  - Bộ chữ cái 26 ký tự Latinh A, B, C, …, Z, a, b, c, …, z
  - Bộ chữ số thập phân : 0, 1, 2, …, 9
  - Các ký hiệu toán học : + – * / = < > ( )
  - Các ký tự đặc biệt : . , : ; [ ] % \ # $ ‘
  - Ký tự gạch nối _ và khoảng trắng ‘ ’

Bộ từ vựng của C

- Từ khóa (keyword)
  - Các từ dành riêng trong ngôn ngữ.
  - Không thể sử dụng từ khóa để đặt tên cho biến, hàm, tên chương trình con.
  - Một số từ khóa thông dụng:
    - const, enum, signed, struct, typedef, unsigned…
    - char, double, float, int, long, short, void
    - case, default, else, if, switch
    - do, for, while
    - break, continue, goto, return

Bộ từ vựng của C

- Tên/Định danh (Identifier)
  - Một dãy ký tự dùng để chỉ tên một hằng số, hằng ký tự, tên một biến, một kiểu dữ liệu, một hàm một hay thủ tục.
  - Không được trùng với các từ khóa và được tạo thành từ các chữ cái và các chữ số nhưng bắt buộc chữ đầu phải là chữ cái hoặc _.
  - Số ký tự tối đa trong một tên là 255 ký tự và được dùng ký tự _ chen trong tên nhưng không cho phép chen giữa các khoảng trắng.

Bộ từ vựng của C

- Ví dụ Tên/Định danh (Identifier)
  - Các tên hợp lệ: GiaiPhuongTrinh, Bai_Tap1
  - Các tên không hợp lệ: 1A, Giai Phuong Trinh
  - Phân biệt chữ hoa chữ thường, do đó các tên sau đây khác nhau:
    - A, a
    - BaiTap, baitap, BAITAP, bAItaP, …

Bộ từ vựng của C

- Dấu chấm phẩy ;
  - Dùng để phân cách các câu lệnh.
  - Ví dụ: printf(“Hello World!”); printf(“\n”);
- Câu chú thích
  - Đặt giữa cặp dấu /* */ hoặc // (C++)
  - Ví dụ: /*Ho & Ten: NVA*/, // MSSV: 0712078
- Hằng ký tự và hằng chuỗi
  - Hằng ký tự: ‘A’, ‘a’, …
  - Hằng chuỗi: “Hello World!”, “Nguyen Van A”
  - Chú ý: ‘A’ khác “A”

Cấu trúc chương trình C

```
#include "…";   // Khai báo file tiêu đề

int x;          // Khai báo biến hàm
void Nhap();    // Khai báo hàm

void main()     // Hàm chính
{
     // Các lệnh và thủ tục
}
```

Ví dụ

```
#include <stdio.h>
#include <conio.h>

void main()
{
      int x, y, tong;
      printf("Nhap hai so nguyen: ");
      scanf("%d%d", &x, &y);
      tong = x + y;
      printf("Tong hai so la %d", tong);
      getch();
}
```

Bài tập lý thuyết

1. Tên (định danh) nào sau đây đặt không hợp lệ, tại sao?
   - Tin hoc co SO A, 1BaiTapKHO
   - THucHaNH, NhapMon_L@pTrinH
2. Câu ghi chú dùng để làm gì? Cách sử dụng ra sao? Cho ví dụ minh họa.
3. Trình bày cấu trúc của một chương trình. Giải thích ý nghĩa của từng phần trong cấ trúc.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KHÁI NIỆM CƠ BẢN VỀ HỆ ĐIỀU HÀNH

Nội dung

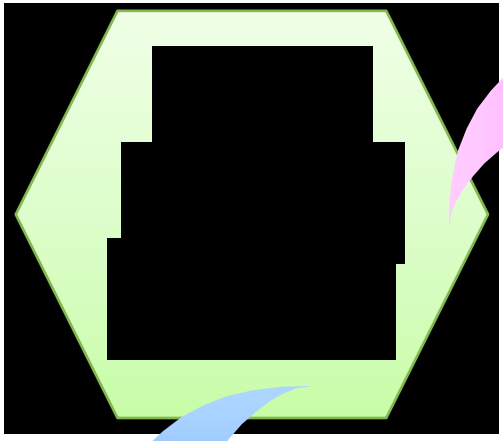
1. Khái niệm & các chức năng chính
2. Phân loại
3. Hệ thống tập tin
4. Một số hệ điều hành thông dụng

Khái niệm Hệ điều hành

Khái niệm

- Một chương trình chạy trên máy tính, dùng để điều hành, quản lý các thiết bị phần cứng và các tài nguyên phần mềm.
- Vai trò trung gian trong việc giao tiếp giữa người sử dụng và phần cứng máy tính.
- Cung cấp môi trường cho phép người sử dụng phát triển và thực hiện các ứng dụng của họ một cách dễ dàng.

Các chức năng chính của HĐH



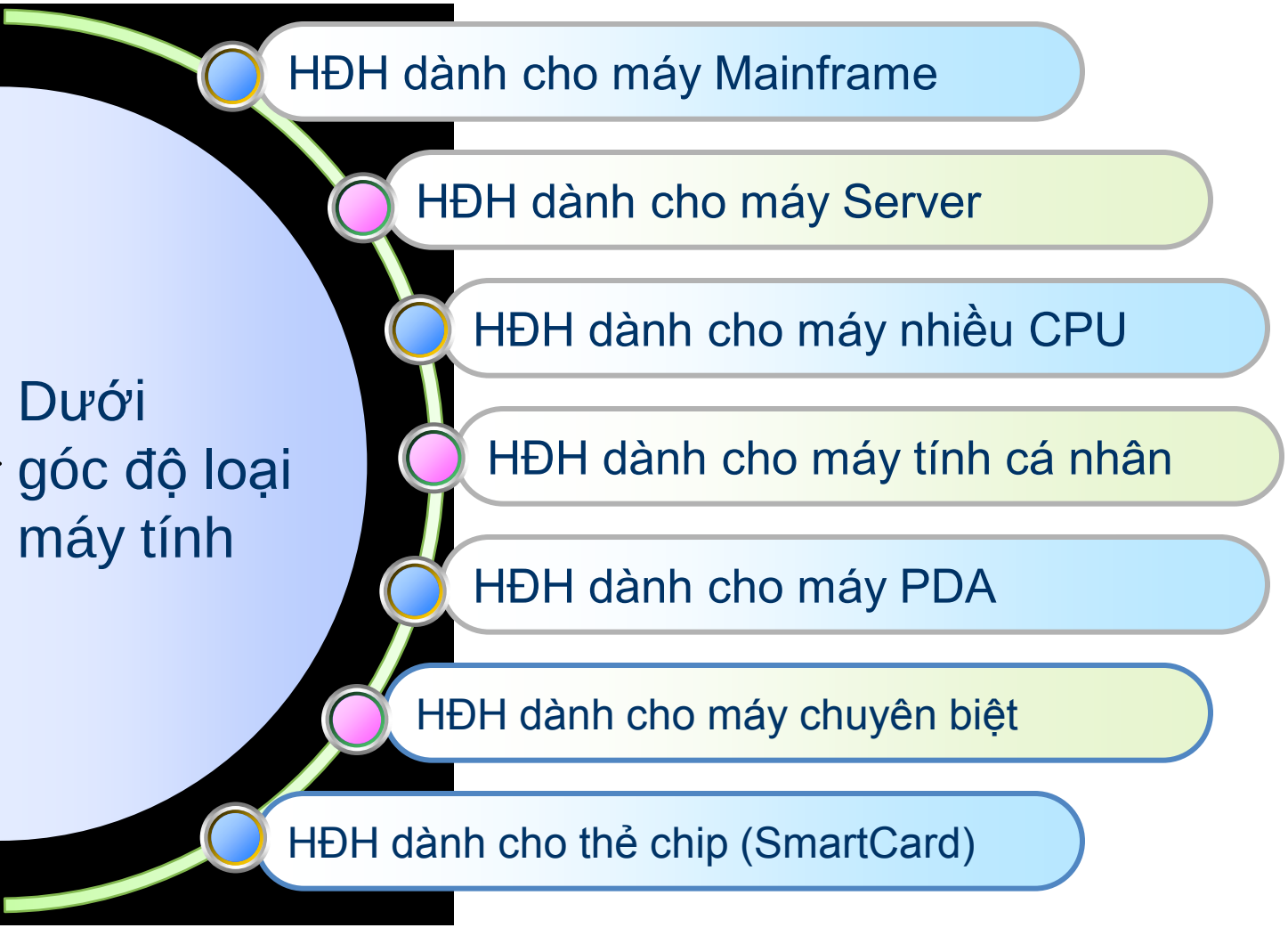
Quản lý chia sẻ tài nguyên

- Tài nguyên là hữu hạn.
- Người sử dụng yêu cầu nhiều tài nguyên đồng thời.
- Chia sẻ tài nguyên phần mềm (thông tin) với nhau.

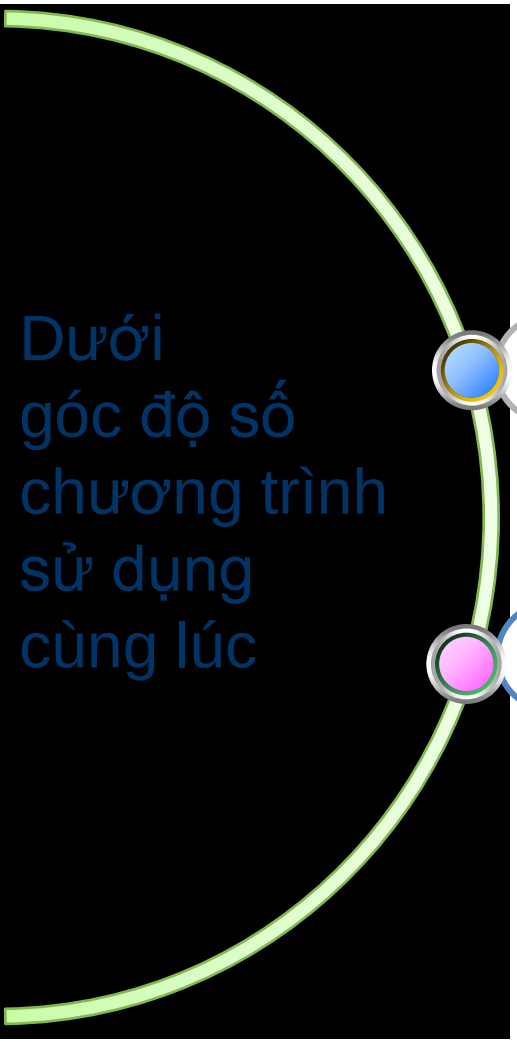
Giả lập một máy tính mở rộng

- Hệ thống nhiều máy tính trừu tượng xếp thành nhiều lớp chồng lên nhau.
- Ẩn đi các chi tiết phần cứng qua giao diện làm việc đơn giản và không phụ thuộc vào thiết bị.

Phân loại Hệ điều hành

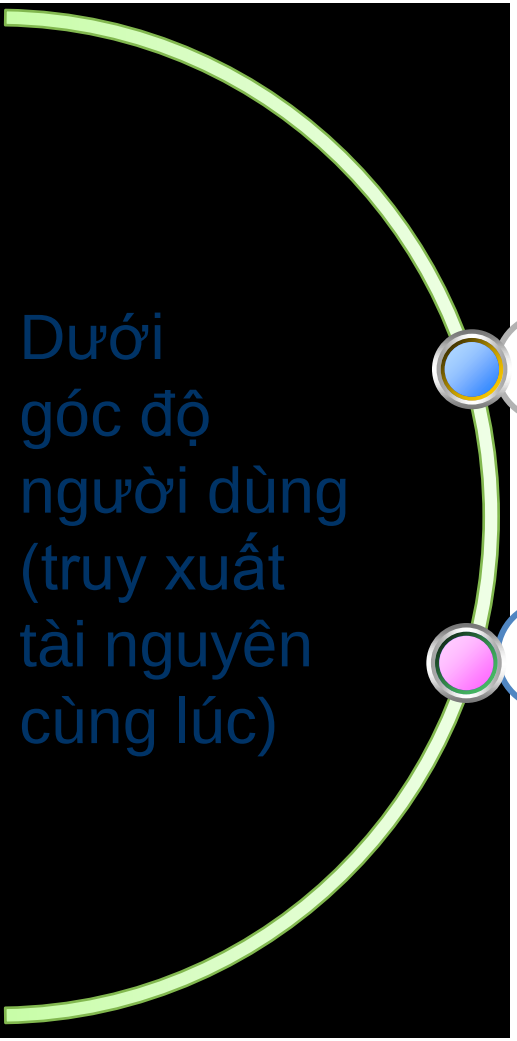


Phân loại Hệ điều hành

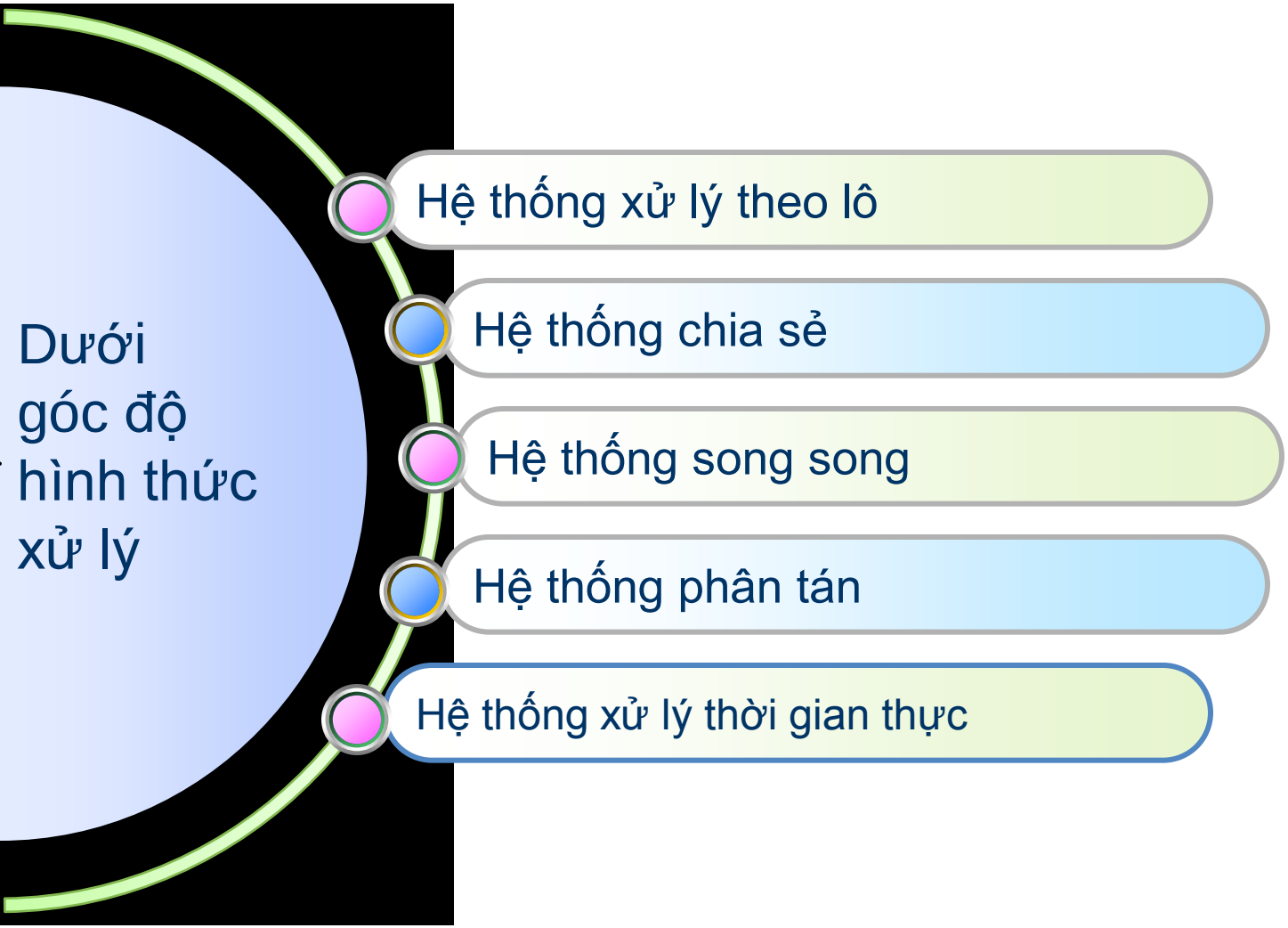


- HĐH đơn nhiệm
- HĐH đa nhiệm

Phân loại Hệ điều hành



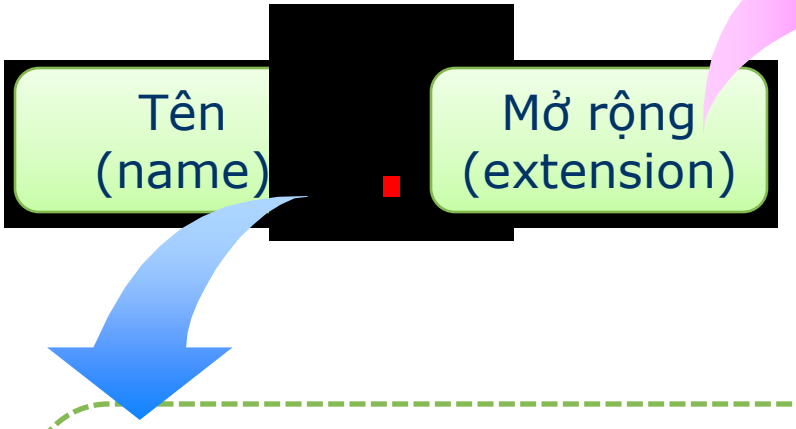
Phân loại Hệ điều hành



Hệ thống tập tin – Tập tin

- Khái niệm
  - Viết tắt của tập thông tin.
  - Còn gọi là tệp, tệp tin, file.
  - Tập hợp của thông tin (dữ liệu) được tổ chức theo một cấu trúc nào đó.
  - Nội dung có thể là chương trình, dữ liệu, văn bản…

Hệ thống tập tin - Tập tin



- Không bắt buộc.
- Thường có 3 ký tự.
- Thường do chương trình ứng dụng tạo tập tin tự đặt

- Bắt buộc phải có.
- Hệ điều hành MS-DOS: dài tối đa 8 ký tự.
- Hệ điều hành Windows: dài tối đa 128 ký tự.
- Gồm các ký tự A đến Z, số 0 đến 9, ký tự khác như #, $, %, ~, ^, @, (,), !, _, khoảng trắng.

Hệ thống tập tin – Tập tin

- Ví dụ
  - Có phần mở rộng: TinA.bat, Bai Tap.pas, …
  - Không có phần mở rộng: TinA, Bai Tap, …
- Các phần mở rộng thông dụng
  - COM, EXE
  - TXT, DOC, PDF
  - PAS, BAS, CPP
  - WK1, XLS
  - BMP, GIF, JPG
  - MP3, DAT, WMA

Hệ thống tập tin – Tập tin

- Ký tự đại diện (Wildcard)
  - Chỉ một nhóm tập tin.
  - Sử dụng 2 ký hiệu:
    - Dấu ? đại diện cho một ký tự bất kỳ.
    - Dấu * đại diện một chuỗi ký tự bất kỳ (có thể rỗng).
  - Ví dụ:
    - Bai?.doc: Bai1.doc, Bai9.doc, Bain.doc, ~~Bai.doc~~, ~~Bai10.doc~~, ~~Bai1.txt~~, …
    - Bai*.doc: Bai.doc, Bai9.doc, Bai10.doc, Bai Tap.doc, …

Hệ thống tập tin – Tập tin

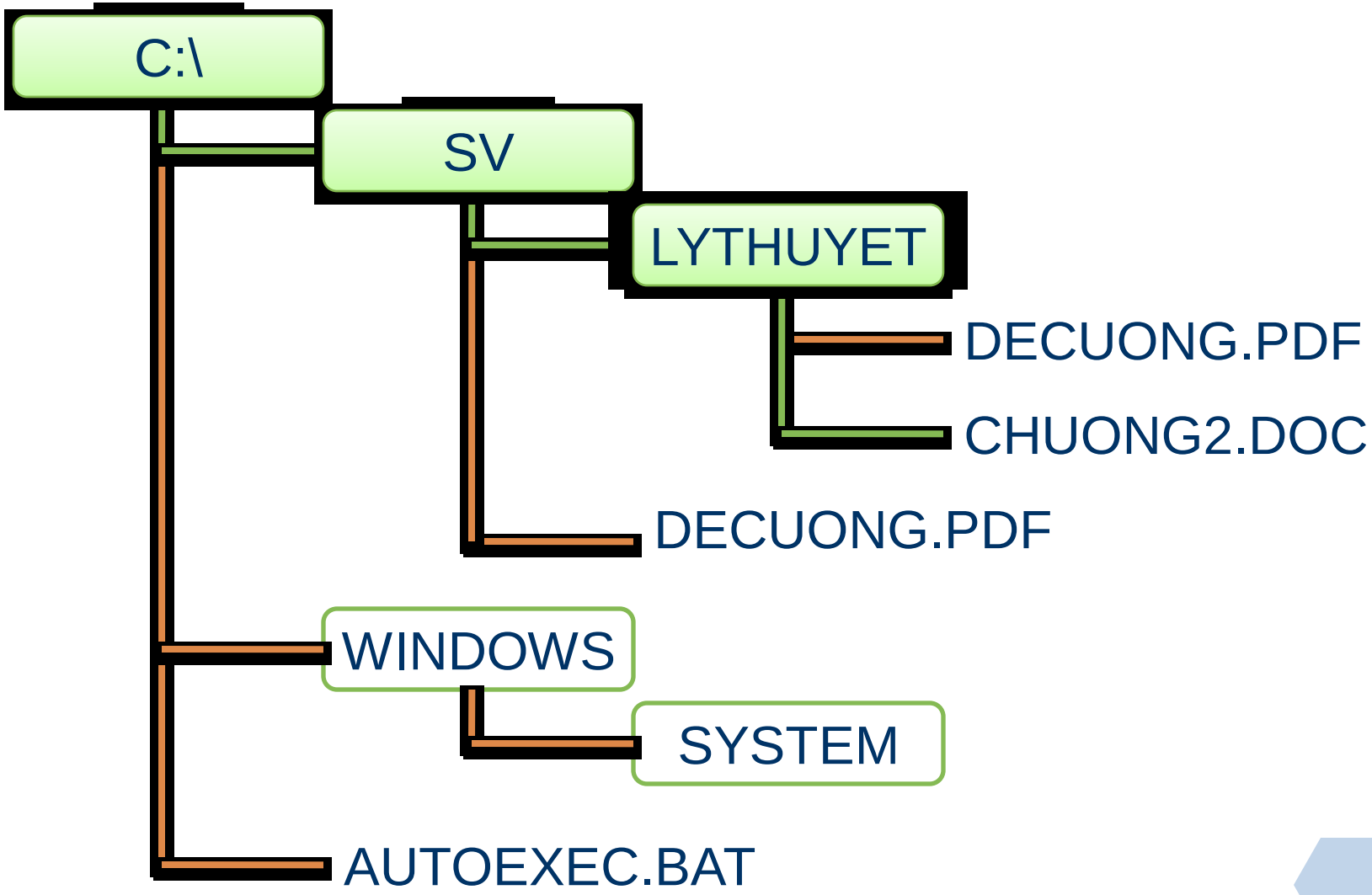
- Thuộc tính
  - Là đặc tính và giới hạn của tập tin.
  - Khác nhau tùy hệ thống
  - Ví dụ trong hệ thống tập tin FAT:
    - Archive (lữu trữ).
    - Hidden (ẩn).
    - Read-only (chỉ đọc).
    - System (thuộc về hệ thống).
    - Sub-directory/directory (thư mục con/thư mục).

Hệ thống tập tin – Thư mục

Một số lưu ý

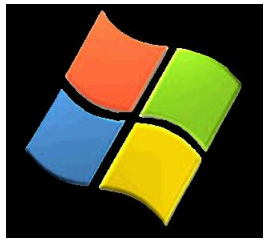
- Thư mục là tập tin ở dạng đặc biệt.
- Không chứa dữ liệu thông thường mà chứa các tập tin và các thư mục khác.
- Thư mục cấp cao nhất trên đĩa là thư mục gốc, ký hiệu \
- Thư mục đang làm việc là thư mục hiện hành.
- Tên thư mục tuân thủ quy tắc đặt tên của tập tin nhưng không có phần mở rộng.
- Ví dụ: BAITAP, BAI THI, THUC HANH…

Cây thư mục và đường dẫn



Một số hệ điều hành thông dụng

- MS-DOS
- Microsoft Windows
  - 3.x (1980), 95 (1995), 98 (1998), Me (2000), 2000 Pro (2000), XP (2001), Vista (2007).
  - NT 4.0 (1996), 2000 Server (2000), 2003 Server (2003): máy chủ - hệ điều hành mạng.
- Linux, Unix, OS/2



Bài tập

1. Hệ điều hành là gì? Hãy nêu các chức năng chính yếu của một hệ điều hành.
2. Hãy nêu vài ví dụ của tập tin có dạng đại diện như sau:
   a) Bai??p*.??s
   b) *T?p.doc
   c) *.?x?
   d) *.*

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmuns.edu.vn

CÁC KIỂU DỮ LIỆU CƠ SỞ

Nội dung

1. Các kiểu dữ liệu cơ sở
2. Biến, Hằng, Câu lệnh & Biểu thức
3. Các lệnh nhập xuất
4. Một số ví dụ minh họa

Các kiểu dữ liệu cơ sở

- Turbo C có 4 kiểu cơ sở như sau:
  - Kiểu số nguyên: giá trị của nó là các số nguyên như 2912, -1706, …
  - Kiểu số thực: giá trị của nó là các số thực như 3.1415, 29.12, -17.06, …
  - Kiểu luận lý: giá trị đúng hoặc sai.
  - Kiểu ký tự: 256 ký tự trong bảng mã ASCII.

Kiểu số nguyên

- Các kiểu số nguyên (có dấu)
  - n bit có dấu: $-2^{n-1} \ldots +2^{n-1} - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
char	1	–128 … +127
int	2	–32.768 … +32.767
short	2	–32.768 … +32.767
long	4	–2.147.483.648 … +2.147.483.647

Kiểu số nguyên

- Các kiểu số nguyên (không dấu)
  - n bit không dấu: 0 … $2^n - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
unsigned char	1	0 … 255
unsigned int	2	0 … 65.535
unsigned short	2	0 … 65.535
unsigned long	4	0 … 4.294.967.295

Kiểu số thực

Các kiểu số thực (floating-point)

- Ví dụ
  - 17.06 = 1.706*10 = 1.706*10^1

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
float (*)	4	$3.4*10^{-38}$ … $3.4*10^{38}$
double (**)	8	$1.7*10^{-308}$ … $1.7*10^{308}$

- (*) Độ chính xác đơn (Single-precision) chính xác đến 7 số lẻ.
- (**) Độ chính xác kép (Double-precision) chính xác đến 19 số lẻ.

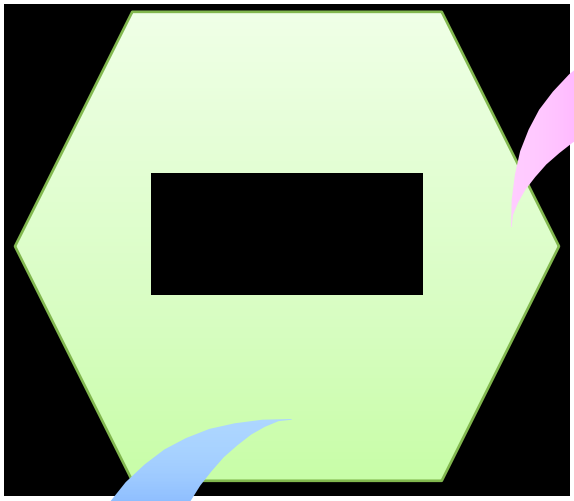
Kiểu luận lý

- Đặc điểm
  - C ngầm định một cách không tường minh:
    - false (sai): giá trị 0.
    - true (đúng): giá trị khác 0, thường là 1.
  - C++: bool
- Ví dụ
  - 0 (false), 1 (true), 2 (true), 2.5 (true)
  - 1 > 2 (0, false), 1 < 2 (1, true)

Kiểu ký tự

- Đặc điểm
  - Tên kiểu: char
  - Miền giá trị: 256 ký tự trong bảng mã ASCII.
  - Chính là kiểu số nguyên do:
    - Lưu tất cả dữ liệu ở dạng số.
    - Không lưu trực tiếp ký tự mà chỉ lưu mã ASCII của ký tự đó.
- Ví dụ
  - Lưu số 65 tương đương với ký tự ‘A’…
  - Lưu số 97 tương đương với ký tự ‘a’.

Biến



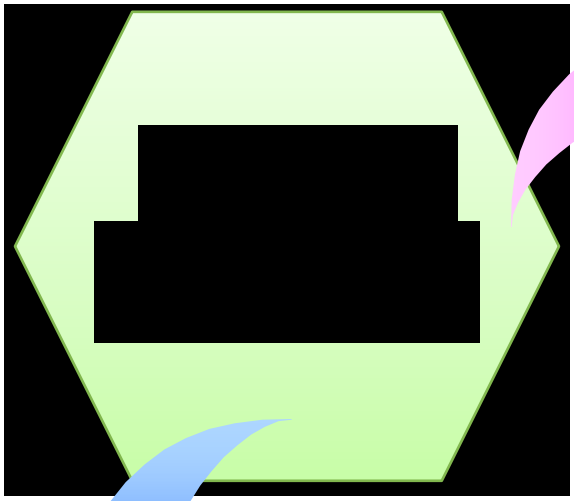
Ví dụ

```
int i;
int j, k;
unsigned char dem;
float ketqua, delta;
```

Cú pháp

```
<kiểu> <tên biến>;
<kiểu> <tên biến 1>, <tên biến 2>;
```

Hằng số



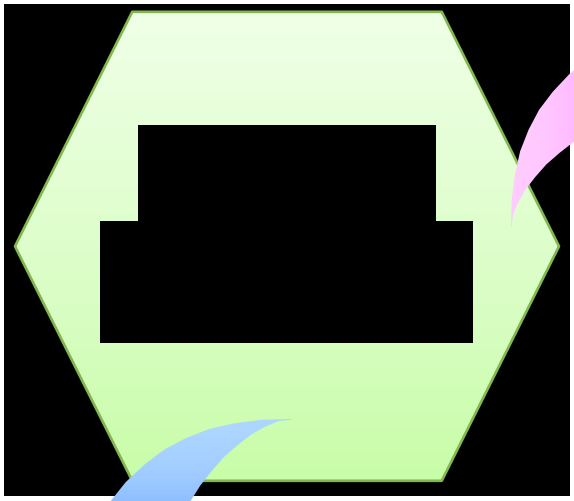
Cú pháp

<kiểu> <tênhằng> = <giá trị>;

Ví dụ

```
int a = 1506;          // 1506(10)
int b = 01506;         // 1506(8)
int c = 0x1506;        // 1506(16) (0x hay 0X)
float d = 15.06e-3;    // 15.06*10^-3 (e hay E)
```

Hằng số



Cú pháp

#define <tênhằng> <giá trị>
hoặc sử dụng từ khóa const.

Ví dụ

```
#define MAX 100                  // Không có ;
#define PI 3.14                  // Không có ;
const int MAX = 100;
const float PI = 3.14;
```

Biểu thức

- Khái niệm
  - Tạo thành từ các toán tử (Operator) và các toán hạng (Operand).
  - Toán tử tác động lên các giá trị của toán hạng và cho giá trị có kiểu nhất định.
  - Toán tử: +, –, *, /, %….
  - Toán hạng: hằng, biến, lời gọi hàm...
- Ví dụ
  - 2 + 3, a / 5, (a + b) * 5, …

Toán tử gán

- Khái niệm
  - Thường được sử dụng trong lập trình.
  - Gán giá trị cho biến.
- Cú pháp
  - <biến> = <giá trị>;
  - <biến> = <biến>;
  - <biến> = <biểu thức>;
  - Có thể thực hiện liên tiếp phép gán.

Toán tử gán

Ví dụ

```
void main()
{
        int a, b, c, d, e, thuong;
        a = 10;
        b = a;
        thuong = a / b;
        a = b = c = d = e = 156;
        e = 156;
        d = e;
        c = d;
        b = c;
        a = b;
}
```

Các toán tử toán học

- Toán tử 1 ngôi
  - Chỉ có một toán hạng trong biểu thức.
  - ++ (tăng 1 đơn vị), -- (giảm 1 đơn vị)
  - Đặt trước toán hạng
    - Ví dụ ++x hay --x: thực hiện tăng/giảm trước.
  - Đặt sau toán hạng
    - Ví dụ x++ hay x--: thực hiện tăng/giảm sau.
- Ví dụ
  - x = 10; y = x++; // y = 10 và x = 11
  - x = 10; y = ++x; // x = 11 và y = 11

Các toán tử toán học

- Toán tử 2 ngôi
  - Có hai toán hạng trong biểu thức.
  - +, –, *, /, % (chia lấy phần dư)
  - x = x + y ⇔ x += y;
- Ví dụ
  - a = 1 + 2; b = 1 – 2; c = 1 * 2; d = 1 / 2;
  - e = 1*1.0 / 2; f = float(1) / 2; g = float(1 / 2);
  - h = 1 % 2;
  - x = x * (2 + 3*5); ⇔ x *= 2 + 3*5;

Các toán tử trên bit

- Các toán tử trên bit
  - Tác động lên các bit của toán hạng (nguyên).
  - & (and), | (or), ^ (xor), ~ (not hay lấy số bù 1)
  - >> (shift right), << (shift left)
  - Toán tử gộp: &=, |=, ^=, ~=, >>=, <<=

&	0	1
0	0	0
1	0	**1**

|	0	1
0	0	**1**
1	**1**	**1**

^	0	1
0	0	**1**
1	**1**	0

~	0	1
	1	0

Các toán tử trên bit

Ví dụ

```
void main()
{
        int a = 5;   // 0000 0000 0000 0101
        int b = 6;   // 0000 0000 0000 0110

        int z1, z2, z3, z4, z5, z6;
        z1 = a & b; // 0000 0000 0000 0100
        z2 = a | b; // 0000 0000 0000 0111
        z3 = a ^ b; // 0000 0000 0000 0011
        z4 = ~a;    // 1111 1111 1111 1010
        z5 = a >> 2;// 0000 0000 0000 0001
        z6 = a << 2;// 0000 0000 0001 0100
}
```

Các toán tử quan hệ

- Các toán tử quan hệ
  - So sánh 2 biểu thức với nhau
  - Cho ra kết quả 0 (hay false nếu sai) hoặc 1 (hay true nếu đúng)
  - ==, >, <, >=, <, <=, !=
- Ví dụ
  - s1 = (1 == 2); s2 = (1 != 2);
  - s3 = (1 > 2); s4 = (1 >= 2);
  - s5 = (1 < 2); s6 = (1 <= 2);

Các toán tử luận lý

Các toán tử luận lý

- Tổ hợp nhiều biểu thức quan hệ với nhau.
- && (and), || (or), ! (not)

&&	0	1
0	0	0
1	0	**1**

||	0	1
0	0	**1**
1	**1**	**1**

- Ví dụ
  - s1 = (1 > 2) && (3 > 4);
  - s2 = (1 > 2) || (3 > 4);
  - s3 = !(1 > 2);

Toán tử điều kiện

Toán tử điều kiện

- Đây là toán tử 3 ngôi (gồm có 3 toán hạng)
- <biểu thức 1> ? <biểu thức 2> : <biểu thức 3>
  - <biểu thức 1> đúng thì giá trị là <biểu thức 2>.
  - <biểu thức 1> sai thì giá trị là <biểu thức 3>.

Ví dụ

- s1 = (1 > 2) ? 2912 : 1706;
- int s2 = 0;
- 1 < 2 ? s2 = 2912 : s2 = 1706;

Toán tử phẩy

- Toán tử phẩy
  - Các biểu thức đặt cách nhau bằng dấu ,
  - Các biểu thức con lần lượt được tính từ trái sang phải.
  - Biểu thức mới nhận được là giá trị của biểu thức bên phải cùng.
- Ví dụ
  - x = (a++, b = b + 2);
  - ⇔ a++; b = b + 2; x = b;

Độ ưu tiên của các toán tử

Toán tử	Độ ưu tiên
() [] -> .	→
! ++ -- - + * (cast) & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= > >=	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %= &= …	←
,	←

Độ ưu tiên của các toán tử

- Quy tắc thực hiện
  - Thực hiện biểu thức trong ( ) sâu nhất trước.
  - Thực hiện theo thứ tự ưu tiên các toán tử.

  => Tự chủ động thêm ( )
- Ví dụ
  - n = 2 + 3 * 5;

    => n = 2 + (3 * 5);
  - a > 1 && b < 2

    => (a > 1) && (b < 2)

Viết biểu thức cho các mệnh đề

- x lớn hơn hay bằng 3

  x >= 3

- a và b cùng dấu

  ((a>0) && (b>0)) || ((a<0) && (b<0))

  (a>0 && b>0) || (a<0 && b<0)

- p bằng q bằng r

  (p == q) && (q == r) hoặc (p == q && q == r)

- –5 < x < 5

  (x > –5) && (x < 5) hoặc (x > –5 && x < 5)

Câu lệnh

- Khái niệm
  - Là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó.
  - Trình biên dịch bỏ qua các khoảng trắng (hay tab hoặc xuống dòng) chen giữa lệnh.
- Ví dụ

```
a=2912;
a = 2912;
a
=
2912;
```

Câu lệnh

- Phân loại
  - Câu lệnh đơn: chỉ gồm một câu lệnh.
  - Câu lệnh phức (khối lệnh): gồm nhiều câu lệnh đơn được bao bởi { và }
- Ví dụ

```
a = 2912;                // Câu lệnh đơn

{                        // Câu lệnh phức/khối lệnh
      a = 2912;
      b = 1706;
}
```

Câu lệnh xuất

- Thư viện
  - #include <stdio.h> (standard input/output)
- Cú pháp
  - printf(<chuỗi định dạng>[, <đs1>, <đs2>, …]);
  - <chuỗi định dạng> là cách trình bày thông tin xuất và được đặt trong cặp nháy kép “ ”.
    - Văn bản thường (literal text)
    - Ký tự điều khiển (escape sequence)
    - Đặc tả (conversion specifier)

Chuỗi định dạng

- Văn bản thường (literal text)
  - Được xuất y hệt như lúc gõ trong chuỗi định dạng.
- Ví dụ
  - Xuất chuỗi Hello World
    - ➔ printf(“Hello ”); printf(“World”);
    - ➔ printf(“Hello World”);
  - Xuất chuỗi a + b
    - ➔ printf(“a + b”);

Chuỗi định dạng

- Ký tự điều khiển (escape sequence)
  - Gồm dấu \ và một ký tự như trong bảng sau:

Ký tự điều khiển	Ý nghĩa
\a	Tiếng chuông
\b	Lùi lại một bước
\n	Xuống dòng
\t	Dấu tab
\\	In dấu \
\?	In dấu ?
\”	In dấu “

- Ví dụ
  - printf(“\t”); printf(“\n”);
  - printf(“\t\n”);

Chuỗi định dạng

- Đặc tả (conversion specifier)
  - Gồm dấu % và một ký tự.
  - Xác định kiểu của biến/giá trị muốn xuất.
  - Các đối số chính là các biến/giá trị muốn xuất, được liệt kê theo thứ tự cách nhau dấu phẩy.

Đặc tả	Ý nghĩa	
%c	Ký tự	char
%d, %ld	Số nguyên có dấu	int, short, long
%f, %lf	Số thực	float, double
%s	Chuỗi ký tự	char[], char*
%u	Số nguyên không dấu	unsigned int/short/long

Chuỗi định dạng

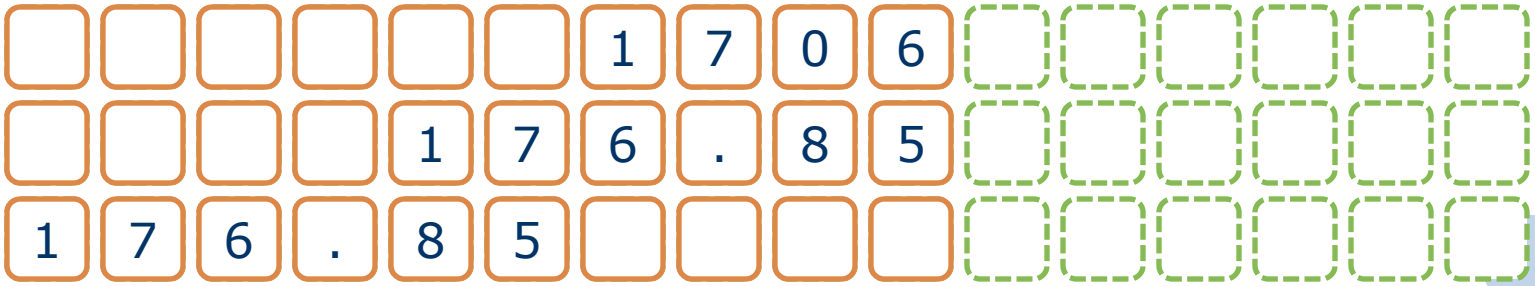
- Ví dụ
  - int a = 10, b = 20;
  - printf(“%d”, a); ➔ Xuất ra 10
  - printf(“%d”, b); ➔ Xuất ra 20
  - printf(“%d %d”, a, b); ➔ Xuất ra 10 20

  - float x = 15.06;
  - printf(“%f”, x); ➔ Xuất ra 15.060000
  - printf(“%f”, 1.0/3); ➔ Xuất ra 0.333333

Định dạng xuất

- Cú pháp
  - Định dạng xuất số nguyên: %nd
  - Định dạng xuất số thực: %n.kd

```
int a = 1706;
float x = 176.85;
printf("%10d", a);printf("\n");
printf("%10.2f", x);printf("\n");
printf("%.2f", x);printf("\n");
```



Chuỗi định dạng

- Phối hợp các thành phần
  - int a = 1, b = 2;
  - Xuất 1 cong 2 bang 3 và xuống dòng.
    - printf("%d", a); // Xuất giá trị của biến a
    - printf(" cong "); // Xuất chuỗi " cong "
    - printf("%d", b); // Xuất giá trị của biến b
    - printf(" bang "); // Xuất chuỗi " bang "
    - printf("%d", a + b); // Xuất giá trị của a + b
    - printf("\n"); // Xuất điều khiển xuống dòng \n

  ➔ printf("%d cong %d bang %d\n", a, b, a+b);

Câu lệnh nhập

- Thư viện
  - #include <stdio.h> (standard input/output)
- Cú pháp
  - scanf(<chuỗi định dạng>[, <đs1>, <đs1>, …]);
  - <chuỗi định dạng> giống định dạng xuất nhưng chỉ có các đặc tả.
  - Các đối số là tên các biến sẽ chứa giá trị nhập và được đặt trước dấu &

Câu lệnh nhập

- Ví dụ, cho a và b kiểu số nguyên
  - scanf("%d", &a); // Nhập giá trị cho biến a
  - scanf("%d", &b); // Nhập giá trị cho biến b
  - ➔ scanf("%d%d", &a, &b);
  - Các câu lệnh sau đây sai
    - scanf("%d", a); // Thiếu dấu &
    - scanf("%d", &a, &b);// Thiếu %d cho biến b
    - scanf("%f", &a); // a là biến kiểu số nguyên
    - scanf("%9d", &a); // không được định dạng
    - scanf("a = %d, b = %d", &a, &b");

Một số hàm hữu ích khác

- Các hàm trong thư việc toán học
  - #include <math.h>
  - 1 đầu vào: double, Trả kết quả: double
    - acos, asin, atan, cos, sin, …
    - exp, log, log10
    - sqrt
    - ceil, floor
    - abs, fabs
  - 2 đầu vào: double, Trả kết quả: double
    - double pow(double x, double y)

Một số hàm hữu ích khác

- Ví dụ
  - int x = 4, y = 3, z = -5;
  - float t = -1.2;
  - float kq1 = sqrt(x1);
  - int kq2 = pow(x, y);
  - float kq3 = pow(x, 1/3);
  - float kq4 = pow(x, 1.0/3);
  - int kq5 = abs(z);
  - float kq6 = fabs(t);

Bài tập lý thuyết

1. Trình bày các kiểu dữ liệu cơ sở trong C và cho ví dụ.
2. Trình bày khái niệm về biến và cách sử dụng lệnh gán.
3. Phân biệt hằng thường và hằng ký hiệu. Cho ví dụ minh họa.
4. Trình bày khái niệm về biểu thức.

   Tại sao nên sử dụng cặp ngoặc đơn.
5. Trình bày cách định dạng xuất.

Bài tập thực hành

4. Nhập năm sinh của một người và tính tuổi của người đó.
5. Nhập 2 số a và b. Tính tổng, hiệu, tính và thương của hai số đó.
6. Nhập tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:
   a. tiền = số lượng * đơn giá
   b. thuế giá trị gia tăng = 10% tiền

Bài tập thực hành

7. Nhập điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.
8. Nhập bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.
9. Nhập vào số xe (gồm 4 chữ số) của bạn. biết số xe của bạn được mấy nút?

Bài tập 4

```
#include <stdio.h>
#include <conio.h>

void main()
{
	int NamSinh, Tuoi;
	printf("Nhap nam sinh: ");
	scanf("%d", &NamSinh);
	Tuoi = 2007 - NamSinh;
	printf("Tuoi cua ban la %d", Tuoi);
	getch();
}
```

Bài tập 5

```
#include <stdio.h>
#include <conio.h>

void main()
{
        int a, b;
        printf("Nhap hai so nguyen: ");
        scanf("%d%d", &a, &b);
        Tong = a + b; Hieu = a - b;
        Tich = a * b; Thuong = a / b;
        printf("Tong cua a va b: %d", Tong);
        printf("Hieu cua a va b: %d", Hieu);
        printf("Tich cua a va b: %d", Tich);
        printf("Thuong cua a va b: %d", Thuong);
}
```

Bài tập 6

```
#include <stdio.h>
#include <conio.h>

void main()
{
        int SoLuong, DonGia, Tien;
        float VAT;

        printf("Nhap so luong va don gia: ");
        scanf("%d%d", &SoLuong, &DonGia);
        Tien = SoLuong * DonGia;
        VAT = Tien * 0.1;
        printf("Tien phai tra: %d", Tien);
        printf("Thue phai tra: %.2f", VAT);
}
```

Bài tập 7

```
#include <stdio.h>
#include <conio.h>

void main()
{
	float T, L, H, DTB;
	int HsT, HsL, HsH;
	printf("Nhap diem Toan, Ly, Hoa: ");
	scanf("%f%f%f", &T, &L, &H);
	printf("Nhap he so Toan, Ly, Hoa: ");
	scanf("%d%d%d", &HsT, &HsL, &HsH);
	DTB = (T * HsT + L * HsL + H * HsH) /
		(HsT + HsL + HsH);
	printf("DTB cua ban la: %.2f", DTB);
}
```

Bài tập 8

```
#include <stdio.h>
#include <conio.h>
#define PI 3.14

void main()
{
	float R, ChuVi, DienTich;
	printf("Nhap ban kinh duong tron: ");
	scanf("%f", &R);
	ChuVi = 2*PI*R;
	DienTich = PI*R*R;
	printf("Chu vi: %.2f", ChuVi);
	printf("Dien tich: %.2f", DienTich);
}
```

Bài tập 9

```
#include <stdio.h>
#include <conio.h>

void main()
{
        int n;
        int n1, n2, n3, n4, SoNut;
        printf("Nhap bien so xe (4 so): ");
        scanf("%d", &n);
        n4 = n % 10; n = n / 10;
        n3 = n % 10; n = n / 10;
        n2 = n % 10; n = n / 10;
        n1 = n;
        SoNut = (n1 + n2 + n3 + n4) % 10;
        printf("So nut la: %d", SoNut);
}
```

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KHÁI NIỆM CƠ BẢN VỀ MẠNG MÁY TÍNH

Nội dung

1. Khái niệm mạng máy tính
2. Phân loại
3. Các lợi ích của mạng máy tính
4. Một số ứng dụng

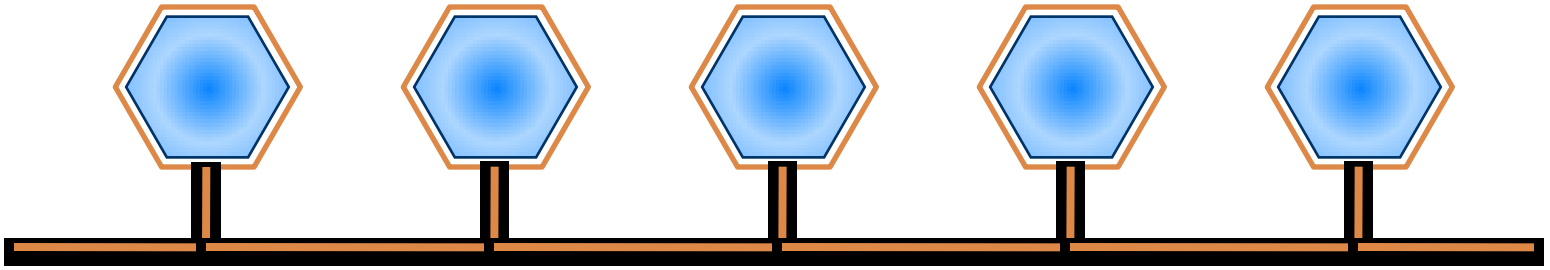
Khái niệm mạng máy tính

- Khái niệm
  - Computer Network hay Network System.
  - Liên kết nhiều máy tính lại với nhau nhằm:
    - Trao đổi thông tin
    - Chia sẻ tài nguyên phần cứng, phần mềm.
    - Tạo điều kiện làm việc theo hình thức kết hợp.
- Các thành phần
  - Các thiết bị đầu cuối (end system).
  - Môi trường truyền (media).
  - Giao thức (protocol).

Phân loại theo quy mô

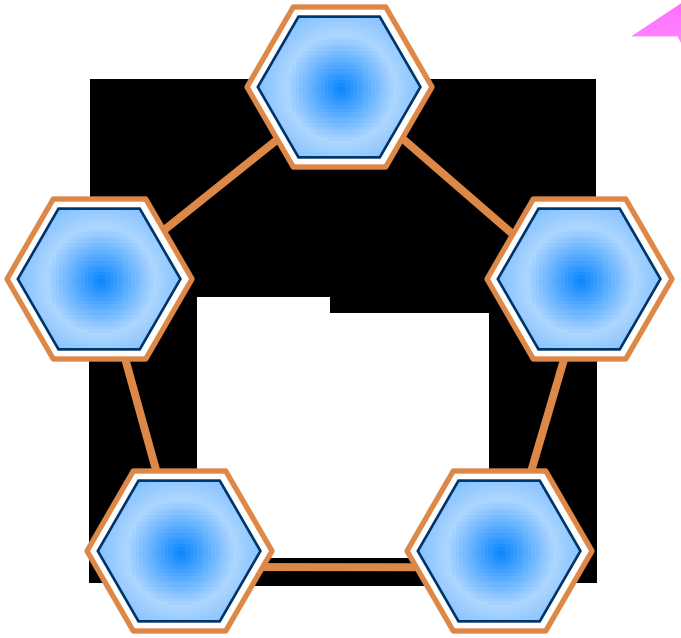
- Mạng cục bộ - LAN (Local Area Network)
  - Mạng tư nhân trong một tòa nhà, một khu vực (trường học, cơ quan).
  - Phạm vi từ vài mét đến 1 km.
  - Một đường dây cáp (cable) nối tất cả máy.
  - Tốc độ truyền: 10 Mbps, 100 Mbps, 1 Gbps, gần đây là 10 Gbps.
  - Kiến trúc mạng thông dụng: mạng bus (tuyến tính) và mạng vòng.

Kiến trúc mạng LAN



Mạng tuyến tính

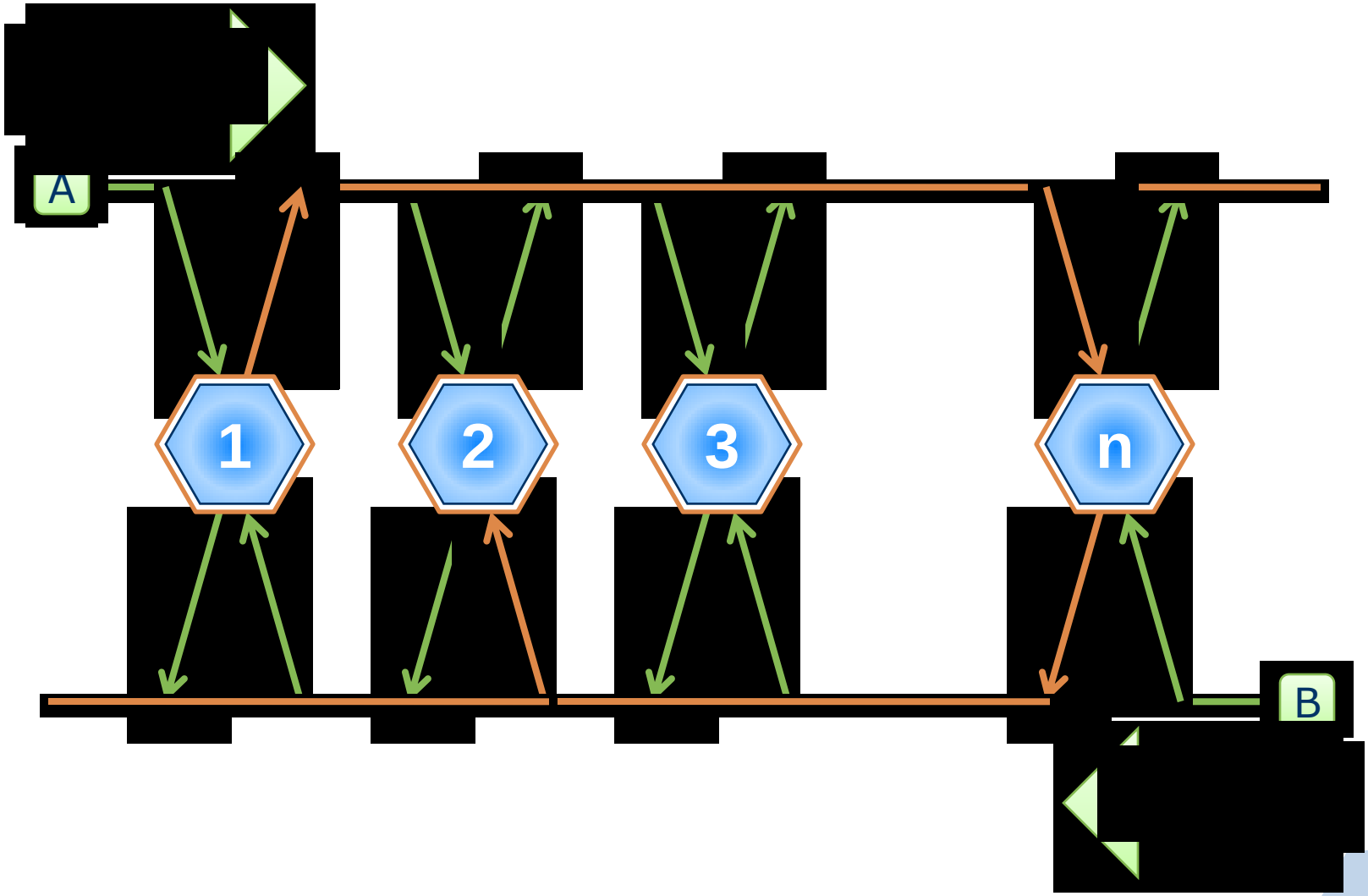
Mạng vòng



Phân loại theo quy mô

- Mạng đô thị - MAN (Metropolitan Area Network)
  - Lớn hơn LAN, nhóm các văn phòng gần nhau trong phạm vi vài km.
  - Tối đa hai dây cáp nối.
  - Không dùng kỹ thuật nối chuyển.
  - Hỗ trợ vận chuyển dữ liệu và đàm thoại, truyền hình. Có thể dùng cáp quang (fiber optical) để truyền tín hiệu.
  - Tốc độ hiện nay đạt đến 10 Gbps.

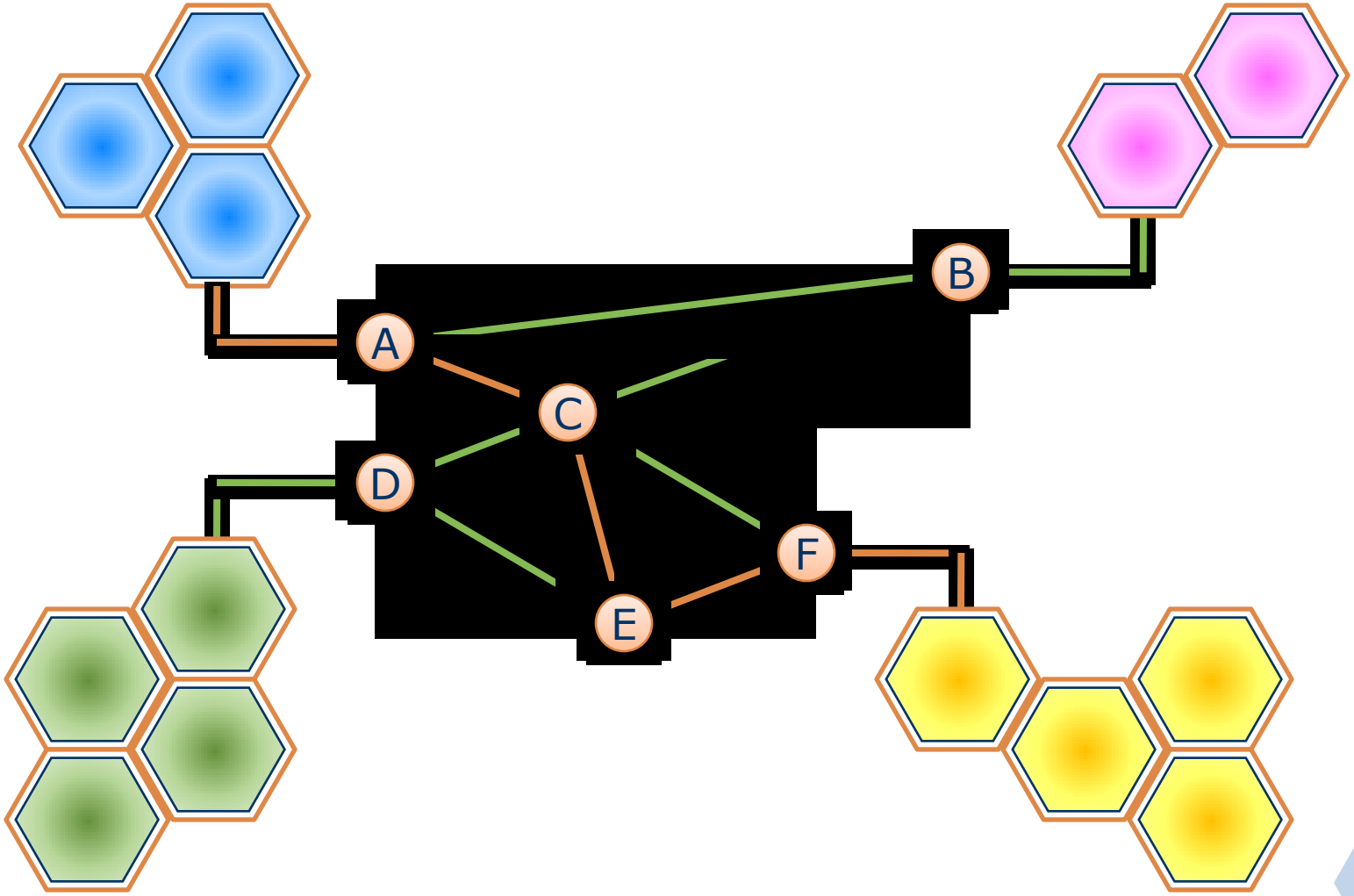
Kiến trúc mạng MAN



Phân loại theo quy mô

- Mạng diện rộng – WAN (Wide Area Network)
  - Dùng trong vùng địa lý lớn như quốc gia, châu lục.
  - Phạm vi vài trăm đến vài ngàn km.

Kiến trúc mạng WAN



Lợi ích của mạng máy tính

- Trong các tổ chức
  - Chia sẻ tài nguyên.
  - Độ tin cậy và an toàn của thông tin cao hơn.
  - Tiết kiệm.
- Cho nhiều người
  - Cung cấp thông tin từ xa giữa các cá nhân.
  - Liên lạc trực tiếp và riêng tư giữa các cá nhân
  - Phương tiên giải trí chung: trò chơi, chia sẻ phim ảnh qua mạng.

Một số ứng dụng của mạng

- Thư tín điện tử (email)
  - Chuyển phát nhanh chóng, không phụ thuộc vị trí, sao lưu và hiệu chỉnh dễ dàng.
- Tham chiếu từ xa (e-conference, chat…)
  - Đối thoại từ xa, chủ động về thời gian.
- Các ứng dụng khác
  - Thông báo, quảng cáo điện tử.
  - Thương mại điện tử.
  - Truyền thông multimedia.
  - …

Bài tập

1. Mạng máy tính là gì? Tại sao cần thiết phải có hệ thống mạng máy tính?.
2. Hãy phân loại khái quát các hệ thống mạng máy tính (theo quy mô).
3. Hãy nêu một số lợi ích của mạng máy tính.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

BIỂU DIỄN THÔNG TIN BÊN TRONG MÁY TÍNH

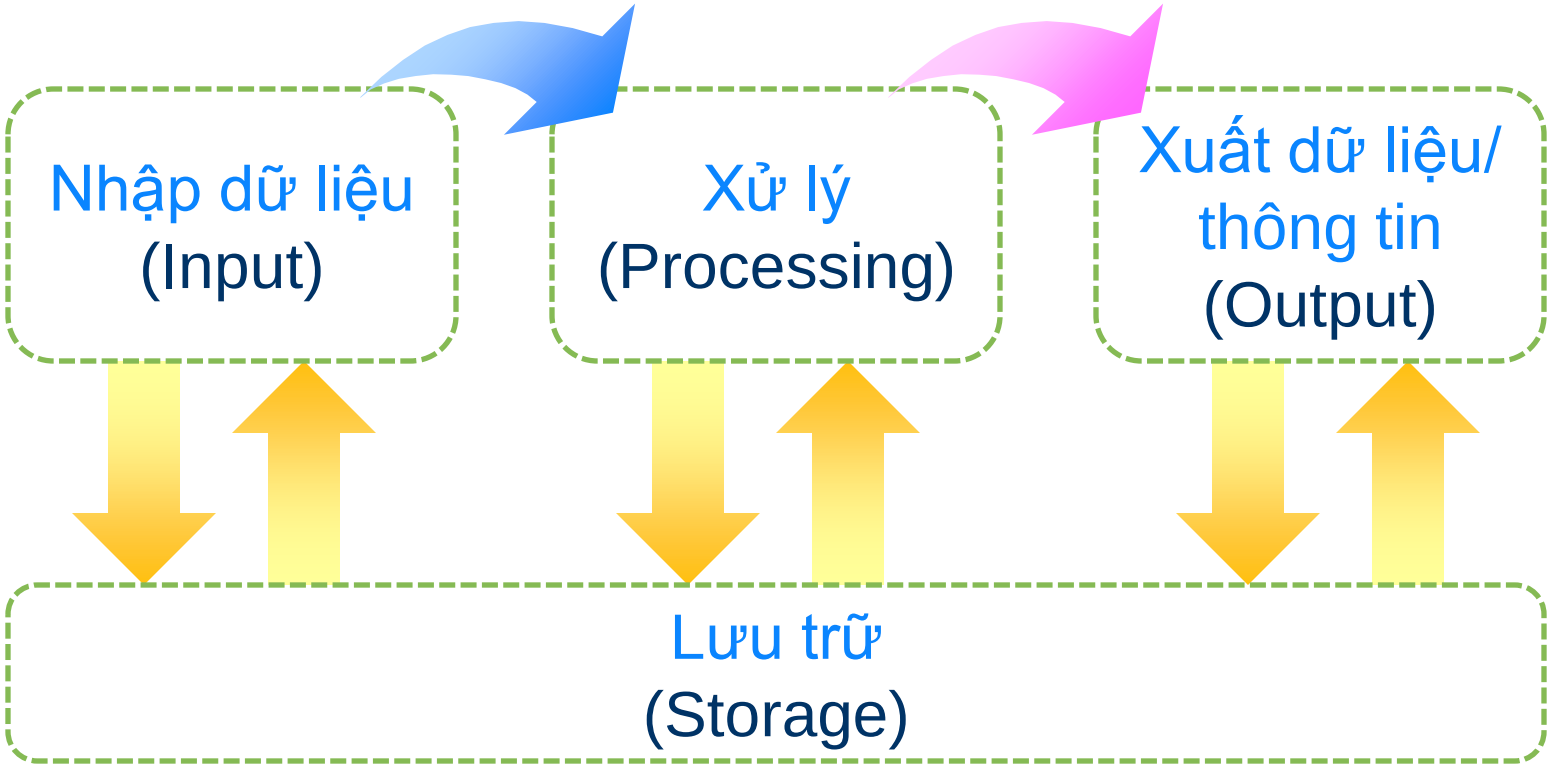
Nội dung

1. Khái niệm thông tin
2. Đơn vị đo thông tin
3. Hệ thống số đếm
4. Biểu diễn thông tin trong MTĐT

Khái niệm

- Thông tin (information)
  - Khái niệm sử dụng thường ngày.
  - Thông qua báo chí, phim ảnh, giao tiếp...
- Dữ liệu (data)
  - Biểu diễn thông tin bằng các tín hiệu vật lý.
  - Không có ý nghĩa nếu chúng không được tổ chức và xử lý.
- Hệ thống thông tin (information system)
  - Hệ thống ghi nhận dữ liệu, xử lý nó để tạo thông tin có ý nghĩa hoặc dữ liệu mới.

Quá trình xử lý thông tin



Đơn vị đo thông tin

- Số học nhị phân sử dụng hai ký số 0 và 1.
- Bit (Binary Digit)
- Đơn vị chứa thông tin nhỏ nhất.
- Các đơn vị đo thông tin lớn hơn:

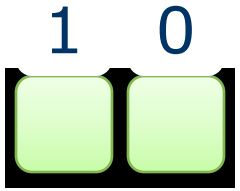
Tên gọi	Ký hiệu	Giá trị
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Byte
MegaByte	MB	2^{10} KB = 2^{20} Byte
GigaByte	GB	2^{10} MB = 2^{30} Byte
TeraByte	TB	2^{10} GB = 2^{40} Byte

Đơn vị đo thông tin

1 bit

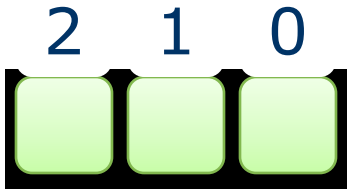
2

2 bit



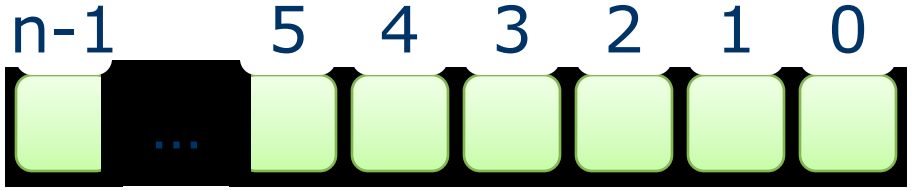
2^2

3 bit



2^3

n bit



2^n

$0\ldots000 \rightarrow 1\ldots111 = 2^n - 1$

Hệ thống số đếm tổng quát

- a ∈ N* biểu diễn duy nhất dưới dạng:
  - $a = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0$

    hay $a = (a_n a_{n-1} \ldots a_1 a_0)_b$
  - Trong đó:
    - b là cơ sở của biểu diễn, b ∈ N, b ≥ 2.
    - a_i là các ký số và a_i ∈ N, 0 ≤ i ≤ n, 0 ≤ a_i < b.
    - Cách viết trên được gọi là biểu diễn cơ sở b của a.
    - Chiều dài của biểu diễn bằng n + 1.
    - Nếu có số lẻ thì vị trí đầu tiên sau dấu phẩy là -1, các vị trí tiếp theo là -2, -3, …

Hệ thập phân – DECimal

- Hệ đếm quen thuộc của con người.
- Sử dụng 10 ký số từ 0 đến 9.
- Ví dụ
  - $1208_{10} = 1*10^3 + 2*10^2 + 0*10^1 + 8*10^0$
    $= 1*1000 + 2*100 + 0*10 + 8*1$
    $= 1000 + 200 + 0 + 8 = 1208_{10}$
  - $12.08_{10} = 1*10^1 + 2*10^0 + 0*10^{-1} + 8*10^{-2}$
    $= 1*10 + 2*1 + 0*1/10 + 8*1/100$
    $= 10 + 2 + 0 + 0.08 = 12.08_{10}$

Hệ nhị phân – BINary

- Hệ đếm sử dụng trong máy tính điện tử.
- Sử dụng 2 ký số là 0 và 1.
- Ví dụ
  - $10110_2 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$
    $= 1*16 + 0*8 + 1*4 + 1*2 + 0*1$
    $= 16 + 0 + 4 + 2 + 0 = 22_{10}$
  - $10.110_2 = 1*2^1 + 0*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3}$
    $= 1*2 + 0*1 + 1*1/2 + 1*1/4 + 0*1/8$
    $= 2 + 0 + 0.5 + 0.25 + 0 = 2.75_{10}$

Hệ bát phân – OCTal

- Sử dụng 8 ký số từ 0 đến 7.
- Ví dụ
  - $2270_8 = 2*8^3 + 2*8^2 + 7*8^1 + 0*8^0$
    $= 2*512 + 2*64 + 7*8 + 0*1$
    $= 1024 + 128 + 56 + 0 = 1208_{10}$
  - $22.70_8 = 2*8^1 + 2*8^0 + 7*8^{-1} + 0*8^{-2}$
    $= 2*8 + 2*1 + 7*1/8 + 0*1/64$
    $= 16 + 2 + 0.875 + 0 = 18.875_{10}$

Hệ thập lục phân – HEXadecimal

- Sử dụng 16 ký số từ 0 đến 9 và từ A đến F
- Ví dụ
  - $4B8_{16} = 4*16^2 + B*16^1 + 8*16^0$
    $= 4*256 + 11*16 + 8*1$
    $= 1024 + 176 + 8 = 1208_{10}$
  - $4B.8_{16} = 4*16^1 + B*16^0 + 8*16^{-1}$
    $= 4*16 + 11*1 + 8*1/16$
    $= 64 + 11 + 0.5 = 75.5_{10}$

Chuyển đổi giữa các hệ đếm

- Đặc điểm
  - Con người sử dụng hệ thập phân.
  - Máy tính sử dụng hệ nhị phân, bát phân, thập lục phân.
- Nhu cầu
  - Chuyển đổi qua lại giữa các hệ đếm.
    - Hệ khác sang hệ thập phân (... ~> dec)
    - Hệ thập phân sang hệ khác (dec ~> ...)
    - Hệ nhị phân sang hệ khác và ngược lại (bin <> …)
    - …

Chuyển từ hệ cơ sở b -> DEC

- Cách 1
  - Khai triển biểu diễn và tính giá trị biểu thức.
  - Ví dụ chuyển từ hệ nhị phân sang thập phân
    - $1011.01_2 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2}$
      $= 8 + 0 + 2 + 1 + 0 + 0.25 = 11.25_{10}$
- Cách 2
  - Nhân/Chia lồng nhau.
  - Ví dụ
    - $\underline{1011}.01_2 = \underline{((1*2 + 0)*2 + 1)*2 + 1} + (1/2 + 0)/2$
      $= \underline{11} + 0.25 = 11.25_{10}$

Chuyển từ DEC -> hệ cơ sở b

- Đổi phần nguyên
  - Chia phần nguyên của số đó cho b và tiếp tục lấy phần nguyên của kết quả chia cho b.
  - Dãy các số dư ở mỗi lần chia là a_0, a_1, ..., a_n.
  - Phần nguyên của số hệ cở sở b là $(a_n \ldots a_1 a_0)$.
- Đổi phần lẻ
  - Nhân phần lẻ của số đó cho b và tiếp tục lấy phần lẻ của kết quả nhân cho b.
  - Dãy các số nguyên ở mỗi lần nhân là a_{-1}, a_{-2}, …, a_{-m} tạo thành phần lẻ ở hệ cơ sở b.

Chuyển từ DEC -> hệ cơ sở b

- Đổi 11.25_{10} sang hệ nhị phân (b = 2)
  - Đổi phần nguyên 11_{10}
    - 11 : 2 = 5 dư 1, vậy $a_0 = 1$
      5 : 2 = 2 dư 1, vậy $a_1 = 1$
      2 : 2 = 1 dư 0, vậy $a_2 = 0$
      1 : 2 = 0 dư 1, vậy $a_3 = 1$
      => phần nguyên $11_{10} = 1011_2$
  - Đổi phần lẻ 0.25_{10}
    - 0.25 * 2 = 0.5, vậy $a_{-1} = 0$
      0.5 * 2 = 1.0, vậy $a_{-2} = 1$
      => phần lẻ $0.25_{10} = .01_2$
  - Vậy $11.25_{10} = 1011.01_2$

Chuyển từ DEC -> hệ cơ sở b

- Đổi 1208.676_{10} sang hệ 16 (lấy 2 số lẻ).
  - Đổi phần nguyên 1208_{10}
    - 1208 : 16 = 75 dư 8, vậy $a_0 = 8$

      75 : 16 = 4 dư 11, vậy $a_1 = B$

      4 : 16 = 0 dư 4, vậy $a_2 = 4$

      => phần nguyên $1208_{10} = 4B8_{16}$
  - Đổi phần lẻ 0.676_{10}
    - 0.676 * 16 = 10.816, vậy $a_{-1} = A$

      0.816 * 16 = 13.056, vậy $a_{-2} = D$

      do ta chỉ muốn lấy 2 số lẻ nên không nhân tiếp.

      => phần lẻ $0.676_{10} = .AD_{16}$
  - Vậy $1208.676_{10} = 4B8.AD_{16}$

Chuyển từ BIN <-> hệ cơ sở b

- Từ hệ nhị phân sang thập lục phân (2^4)
  - Nhóm từng bộ 4 bit trong biểu diễn nhị phân rồi chuyển sang ký số tương ứng trong hệ thập lục phân (0000 ~> 0,…, 1111 ~> F)
  - Ví dụ
    - $1001011.1_2 = 0100\ 1011\ .\ 1000 = 4B.8_{16}$

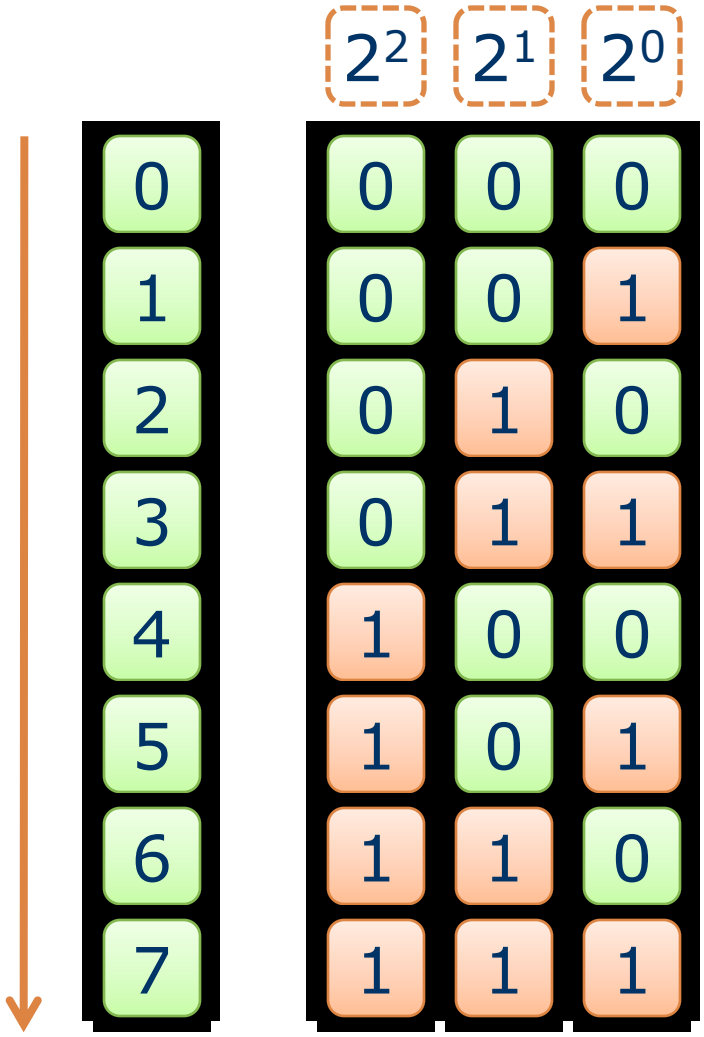
HEX	BIN	HEX	BIN	HEX	BIN	HEX	BIN
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Chuyển từ BIN <-> hệ cơ sở b

- Từ hệ nhị phân sang thập bát phân (2^3)
  - Nhóm từng bộ 3 bit trong biểu diễn nhị phân rồi chuyển sang ký số tương ứng trong hệ bát phân (000 ~> 0,…, 111 ~> 7).
  - Ví dụ
    - $1101.11_2 = 001\ 101\ .\ 110 = 15.6_8$

OCT	BIN	OCT	BIN
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Lập bảng chuyển đổi

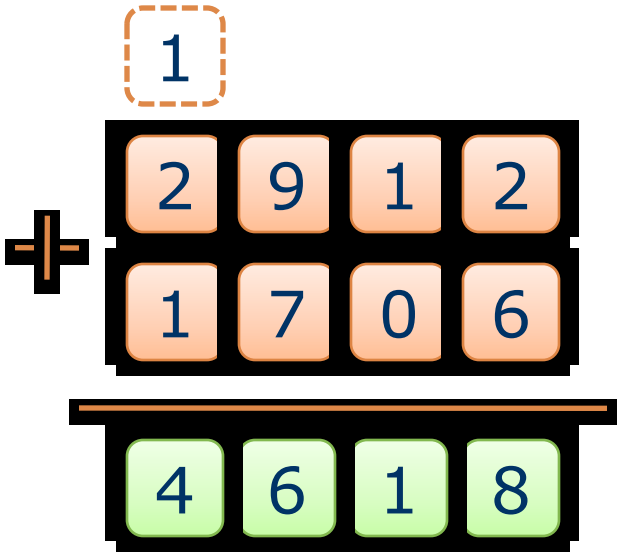


Bảng tổng hợp

Từ hệ	Sang hệ	Cách thực hiện
b bất kỳ	10	Khai triển theo cơ sở b Phần nguyên: nhân lồng ->>>>>>>>>> Phần lẻ: chia lồng <-
10	b bất kỳ	Phần nguyên (lẻ): Chia (nhân) liên tục phần nguyên (lẻ) cho b và giữa lại phần dư (phần nguyên)
2	8	Nhóm từng bộ 3 bit <- . ->
	16	Nhóm từng bộ 4 bit <- . ->
8	2	1 ký số ứng với 3 bit
	16	Hệ trung gian: nhị phân
16	2	1 ký số ứng với 4 bit
	8	Hệ trung gian: nhị phân

Các phép toán trên các hệ đếm

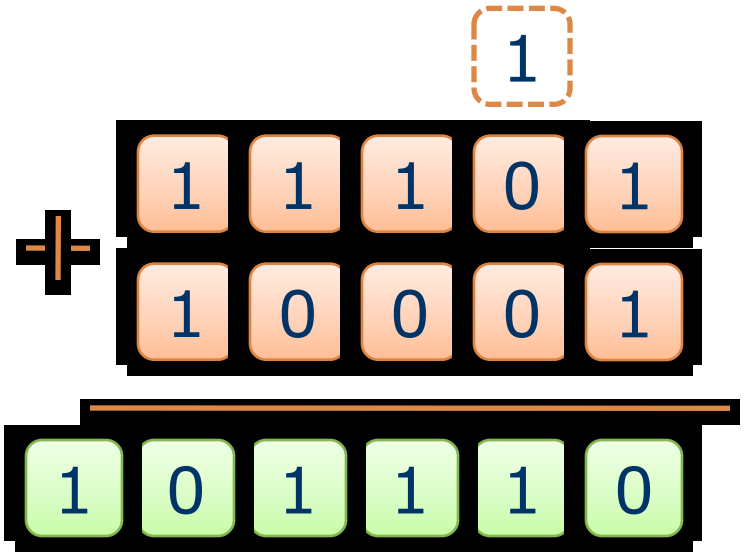
- Phép cộng
  - Ví dụ cộng 2 số thập phân



Các phép toán trên các hệ đếm

+	0	1
0	0	1
1	1	10

- Phép cộng
  - Ví dụ cộng 2 số nhị phân

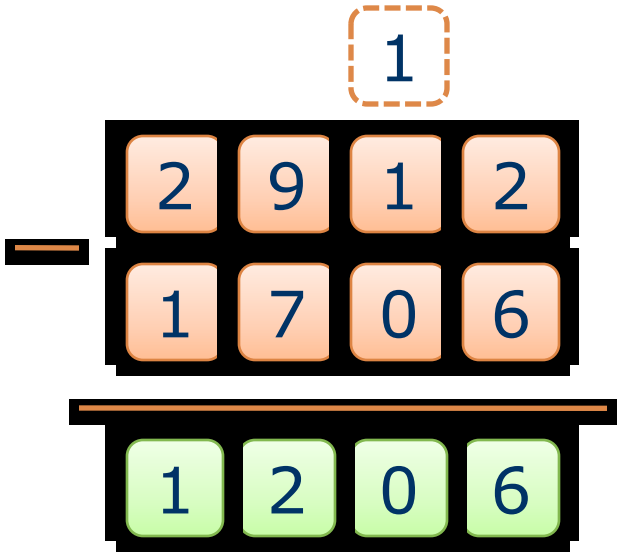


Các phép toán trên các hệ đếm

- Phép cộng
  - Cộng các số ở hệ khác được thực hiện tương tự như ở hệ thập phân.
  - Ở mỗi hệ nên lập bảng cộng các ký số và tra trong bảng này để được ngay kết quả.
  - Bảng cộng số bát phân và số thập lục phân (xem trong giáo trình).

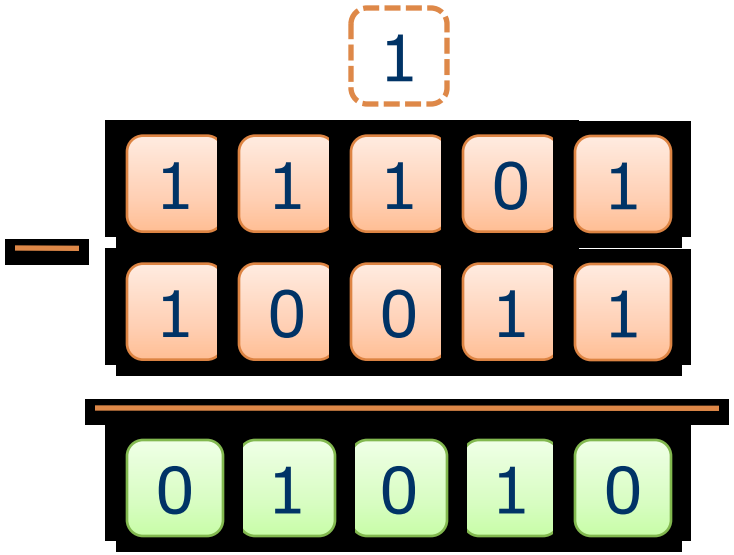
Các phép toán trên các hệ đếm

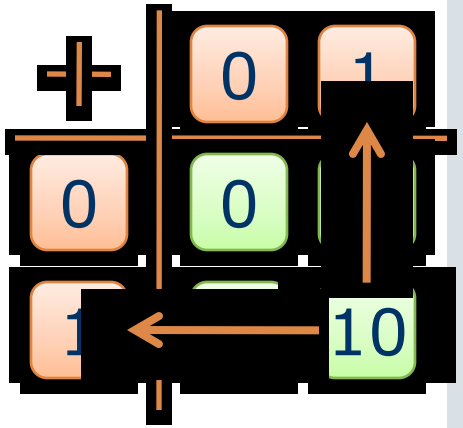
- Phép trừ (kết quả dương)
  - Ví dụ trừ hai số thập phân



Các phép toán trên các hệ đếm

- Phép trừ (kết quả dương)
  - Ví dụ trừ 2 số nhị phân



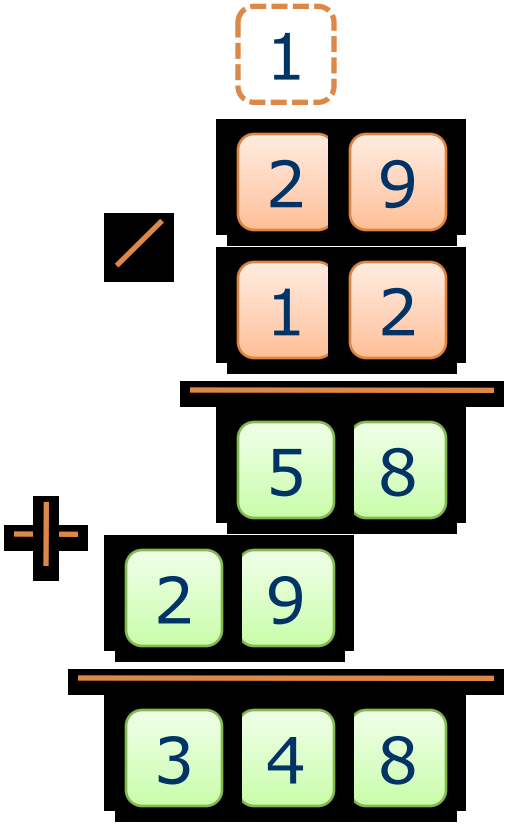


Các phép toán trên các hệ đếm

- Phép trừ
  - Các ví dụ trên số bị trừ nhỏ hơn số trừ, tức là a – b với a < b.
  - Muốn tính a – b mà a > b ta tính b – a rồi đảo dấu kết quả.
  - Tra bảng cộng từng hệ đếm để có kết quả nhanh chóng.

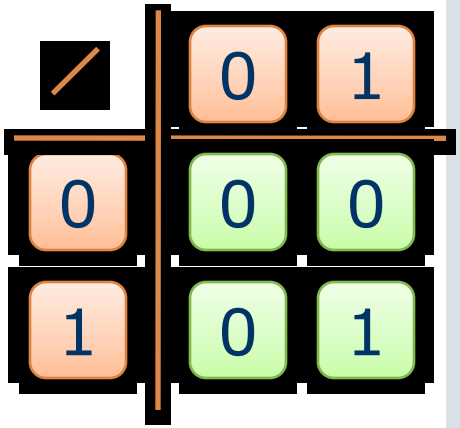
Các phép toán trên các hệ đếm

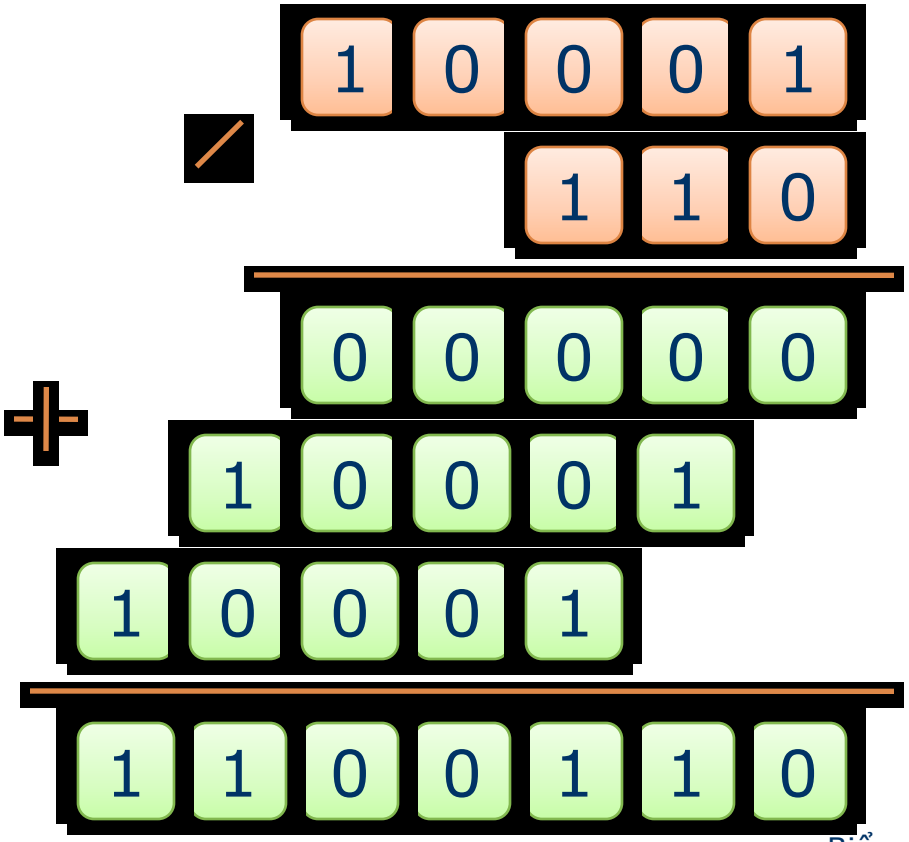
- Phép nhân
  - Ví dụ nhân 2 số thập phân



Các phép toán trên các hệ đếm

- Phép nhân
  - Ví dụ nhân 2 số nhị phân





Các phép toán trên các hệ đếm

- Phép nhân
  - Nhân các số ở hệ khác được thực hiện tương tự như ở hệ thập phân.
  - Ở mỗi hệ nên lập bảng nhân các ký số và tra trong bảng này để được ngay kết quả.
  - Bảng nhân số bát phân và số thập lục phân (xem trong giáo trình).
- Phép chia
  - Tương tự như phép chia trong hệ thập phân (xem trong giáo trình).

Biểu diễn thông tin trong MTĐT

- Đặc điểm
  - Được lưu trong các thanh ghi hoặc trong các ô nhớ. Thanh ghi hoặc ô nhớ có kích thước 1 byte (8 bit) hoặc 1 word (16 bit).
  - Biểu diễn số nguyên không dấu, số nguyên có dấu, số thực và ký tự.
- Hai loại bit đặc biệt
  - msb (most significant bit): bit nặng nhất (bit n)
  - lsb (least significant bit): bit nhẹ nhất (bit 0)

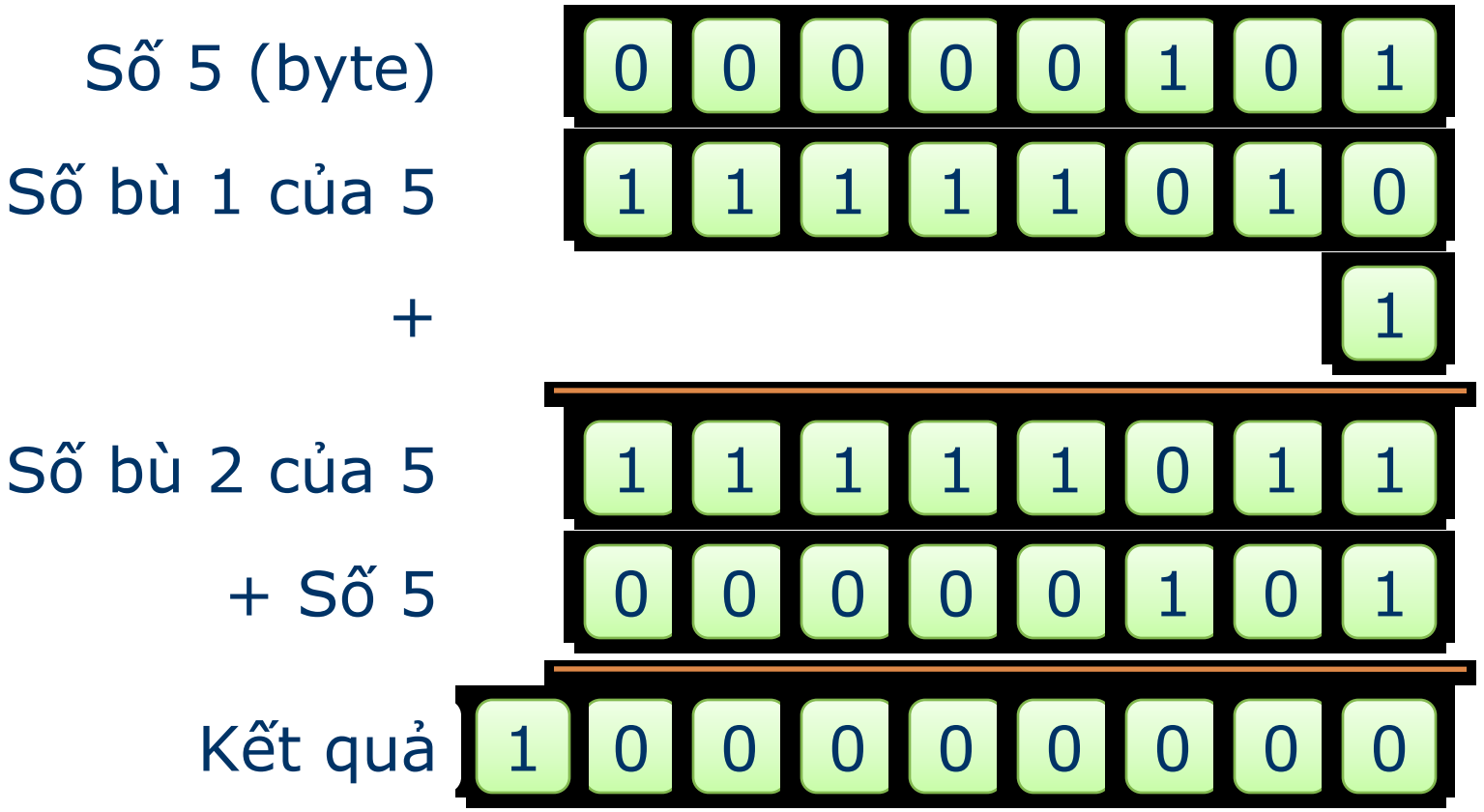
Biểu diễn số nguyên không dấu

- Đặc điểm
  - Biểu diễn các đại lương luôn dương.
  - Ví dụ: chiều cao, cân nặng, mã ASCII…
  - Tất cả bit được sử dụng để biểu diễn giá trị.
  - Số nguyên không dấu 1 byte lớn nhất là $1111\ 1111_2 = 2^8 - 1 = 255_{10}$.
  - Số nguyên không dấu 1 word lớn nhất là $1111\ 1111\ 1111\ 1111_2 = 2^{16} - 1 = 65535_{10}$.
  - Tùy nhu cầu có thể sử dụng số 2, 3… word.
  - lsb = 1 thì số đó là số đó là số lẻ.

Biểu diễn số nguyên có dấu

- Đặc điểm
  - Lưu các số dương hoặc âm.
  - Bit msb dùng để biểu diễn dấu
    - msb = 0 biểu diễn số dương. VD: 0101 0011
    - msb = 1 biểu diễn số âm. VD: 1101 0011
  - Số âm trong máy được biểu diễn ở dạng số bù 2.

Số bù 1 và số bù 2



Biểu diễn số nguyên có dấu

Nhận xét

- Số bù 2 của x cộng với x là một dãy toàn bit 0 (không tính bit 1 cao nhất do vượt quá phạm vi lưu trữ). Do đó số bù 2 của x chính là giá trị âm của x hay – x.
- Đổi số thập phân âm –5 sang nhị phân?
  - => Đổi 5 sang nhị phân rồi lấy số bù 2 của nó.
- Thực hiện phép toán a – b?
  - a – b = a + (–b) => Cộng với số bù 2 của b.

Tính giá trị có dấu và không dấu

- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Số nguyên không dấu ?
    - Tất cả 16 bit lưu giá trị.
    - => giá trị là 52464.
  - Số nguyên có dấu ?
    - Bit msb = 1 do đó số này là số âm.
    - => độ lớn là giá trị của số bù 2.
    - Số bù 2 = 0011 0011 0001 0000 = 13072.
    - => giá trị là –13072.

Tính giá trị có dấu và không dấu

- Nhận xét
  - Bit msb = 0 thì giá trị có dấu bằng giá trị không dấu.
  - Bit msb = 1 thì giá trị có dấu bằng giá trị không dấu trừ đi 256 (byte) hay 65536 (word).
- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Giá trị không dấu là 52464.
  - Giá trị có dấu: vì bit msb = 1 nên giá trị có dấu bằng 52464 – 65536 = –13072.

Biểu diễn số thực

- Khái niệm
  - Để lưu trữ các số lẻ.
  - Sử dụng dấu chấm động (floating-point).
  - Chia làm 3 phần:
    - 1 bit để biểu diễn dấu.
    - Một chuỗi bit để biểu diễn số mũ.
    - Một chuỗi bit để biểu diễn phần định trị.
  - Đọc thêm phần 4.4.1.3 Số thực trong giáo trình Tin học cơ sở A.

Biểu diễn thông tin chữ số

- Khái niệm
  - Để biểu diễn các ký tự như chữ thường, chữ hoa, ký hiệu…
- Các hệ mã
  - Hệ thập phân mã nhị phân BCD (Binary Coded Decimal): dùng số nhị phân 4 bit thay thế một số thập phân.
  - Hệ thập phân mã nhị phân mở rộng EBCDIC (Extended Binary Coded Decimal Interchange Code): dùng 8 bit biểu diễn 1 ký tự.

Biểu diễn thông tin chữ số

Các hệ mã (tiếp theo)

- Hệ chuyển đổi thông tin theo mã chuẩn của Mỹ ASCII (American Standard Code for Information Interchange)
  - 1 – 31: ký tự điều khiển.
  - 32 – 47: khoảng trắng, “ # $ % & ‘ ( ) * +, - . /
  - 48 – 57: ký số từ 0 đến 9
  - 58 – 64: các dấu : ; < = > ? @
  - 65 – 90: các chữ in hoa từ A đến Z
  - 91 – 96: các dấu [\ ] _ `
  - 97 – 122: các chữ thường từ a đến z
  - 123 – 127: các dấu { | } ~ DEL

Bài tập

1. Thông tin là gì? Hãy vẽ mô hình và mô tả khái quát quá trình xử lý thông tin trong máy tính?
2. Đơn vị đo thông tin trong máy tính điện tử là gì? Kể tên một số đơn vị đo thông tin mà bạn biết.
3. Trình bày hệ đếm nhị phân, bát phâ thập phân, thập lục phân.
4. Số nguyên trong máy tính.
5. Bảng mã ASCII.

Bài tập

6. Đổi sang hệ thập phân (lấy 2 số lẻ)
   - e. 3203_{16}
   - f. $80.07A_{16}$
7. Đổi sang hệ thập lục phân
   - a. 19405_{10}
   - b. 194.05_{10}
9. Tính giá trị không dấu, có dấu của v
   - b. $F956_{16}$
10. Thực hiện phép cộng, trừ, nhân
    - c. $C2_{16}$ và $9C_{16}$

Giải bài tập 6e

- $3203_{16} = 3*16^3 + 2*16^2 + 0*16^1 + 3*16^0$
  $= 3*4096 + 2*256 + 0 + 3*1$
  $= 12288 + 512 + 3 = 12803_{10}$

- $3203_{16} = ((3*16 + 2)*16 + 0)*16 + 3$
  $= (50*16 + 0)*16 + 3$
  $= 800*16 + 3 = 12803_{10}$

Giải bài tập 6f

- $80.07A_{16} = 8*16^{1} + 0*16^{0}$
  $+ 0*16^{-1} + 7*16^{-2} + A*16^{-3}$
  $= 8*16 + 7/256 + 10/4096$
  $= 128 + 0.027 + 0.002$
  $= 128.03_{10}$
- $\underline{80}.07A_{16} = \underline{8*16 + 0}$
  $+ ((A/16 + 7)/16 + 0)/16$
  $= 128 + (7.625/16 + 0)/1$
  $= 128 + 0.4766/16$
  $= 128.03_{10}$

Giải bài tập 7a

- Đổi 19405_{10} sang hệ 16
  - 19405 : 16 = 1212 dư 13, vậy a_0 = D
    1212 : 16 = 75 dư 12, vậy a_1 = C
    75 : 16 = 4 dư 11, vậy a_2 = B
    4 : 16 = 0 dư 4, vậy a_3 = 4

  Vậy $19405_{10} = 4BCD_{16}$

Giải bài tập 7b

- Đổi phần nguyên 194_{10} sang hệ 16
  - 194 : 16 = 12 dư 2, vậy $a_0 = 2$

    12 : 16 = 0 dư 12, vậy $a_1 = C$

    Vậy $194_{10} = C2_{16}$
- Đổi phần lẻ 0.05_{10} sang hệ 16
  - 0.05 * 16 = 0.8, vậy $a_{-1} = 0$

    0.8 * 16 = 12.8, vậy $a_{-2} = C$

    Vậy $0.05_{10} = 0.0C_{16}$
- Vậy $194.05_{10} = C2.0C_{16}$

Giải bài tập 9b

- $F956_{16}$ = $1111\ 1001\ 0101\ 0110_2$
- Giá trị không dấu: 63830_{10}
- Giá trị có dấu:
  - Nhận xét: bit msb = 1 nên đây là số âm.
  - Cách 1: Tính số bù 2 của nó.
  - Cách 2: 63830 – 65536 = -1706_{10}

Giải bài tập 10c

	C	2	
+	9	C	
1	5	E	

	1		
	C	2	
−	9	C	
	2	6	

		1		
		C	2	
/		9	C	
	9	1	8	
+	6	D	2	
	7	6	3	8

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

Nội dung

1. Các khái niệm cơ bản
2. Các bước xây dựng chương trình
3. Biểu diễn thuật toán
4. Cài đặt thuật toán bằng NNLT

Các khái niệm cơ bản

- Lập trình máy tính
  - Gọi tắt là lập trình (programming).
  - Nghệ thuật cài đặt một hoặc nhiều thuật toán trừu tượng có liên quan với nhau bằng một ngôn ngữ lập trình để tạo ra một chương trình máy tính.
- Thuật toán
  - Là tập hợp (dãy) hữu hạn các chỉ thị (hành động) được định nghĩa rõ ràng nhằm giải quyết một bài toán cụ thể nào đó.

Các khái niệm cơ bản

Ví dụ

- Thuật toán giải PT bậc nhất: ax + b = 0 (a, b là các số thực).

```
Đầu vào: a, b thuộc R
Đầu ra: nghiệm phương trình ax + b = 0
```

- Nếu a = 0
  - b = 0 thì phương trình có nghiệm bất kì.
  - b ≠ 0 thì phương trình vô nghiệm.
- Nếu a ≠ 0
  - Phương trình có nghiệm duy nhất x = -b/a

Các tính chất của thuật toán

- Bao gồm 5 tính chất sau:
  - Tính chính xác: quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.
  - Tính rõ ràng: các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định.
  - Tính khách quan: được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.
  - Tính phổ dụng: có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
  - Tính kết thúc: hữu hạn các bước tính toán.

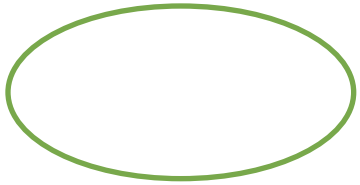
Các bước xây dựng chương trình

Sử dụng ngôn ngữ tự nhiên

```
Đầu vào: a, b thuộc R
Đầu ra: nghiệm phương trình ax + b = 0
```

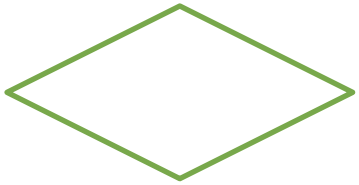
1. Nhập 2 số thực a và b.
2. Nếu a = 0 thì
    - 2.1. Nếu b = 0 thì
        - 2.1.1. Phương trình vô số nghiệm
        - 2.1.2. Kết thúc thuật toán.
    - 2.2. Ngược lại
        - 2.2.1. Phương trình vô nghiệm.
        - 2.2.2. Kết thúc thuật toán.
3. Ngược lại
    - 3.1. Phương trình có nghiệm.
    - 3.2. Giá trị của nghiệm đó là x = -b/a
    - 3.3. Kết thúc thuật toán.

Sử dụng lưu đồ - sơ đồ khối



Khối giới hạn
Chỉ thị bắt đầu và kết thúc.

Khối vào ra
Nhập/Xuất dữ liệu.

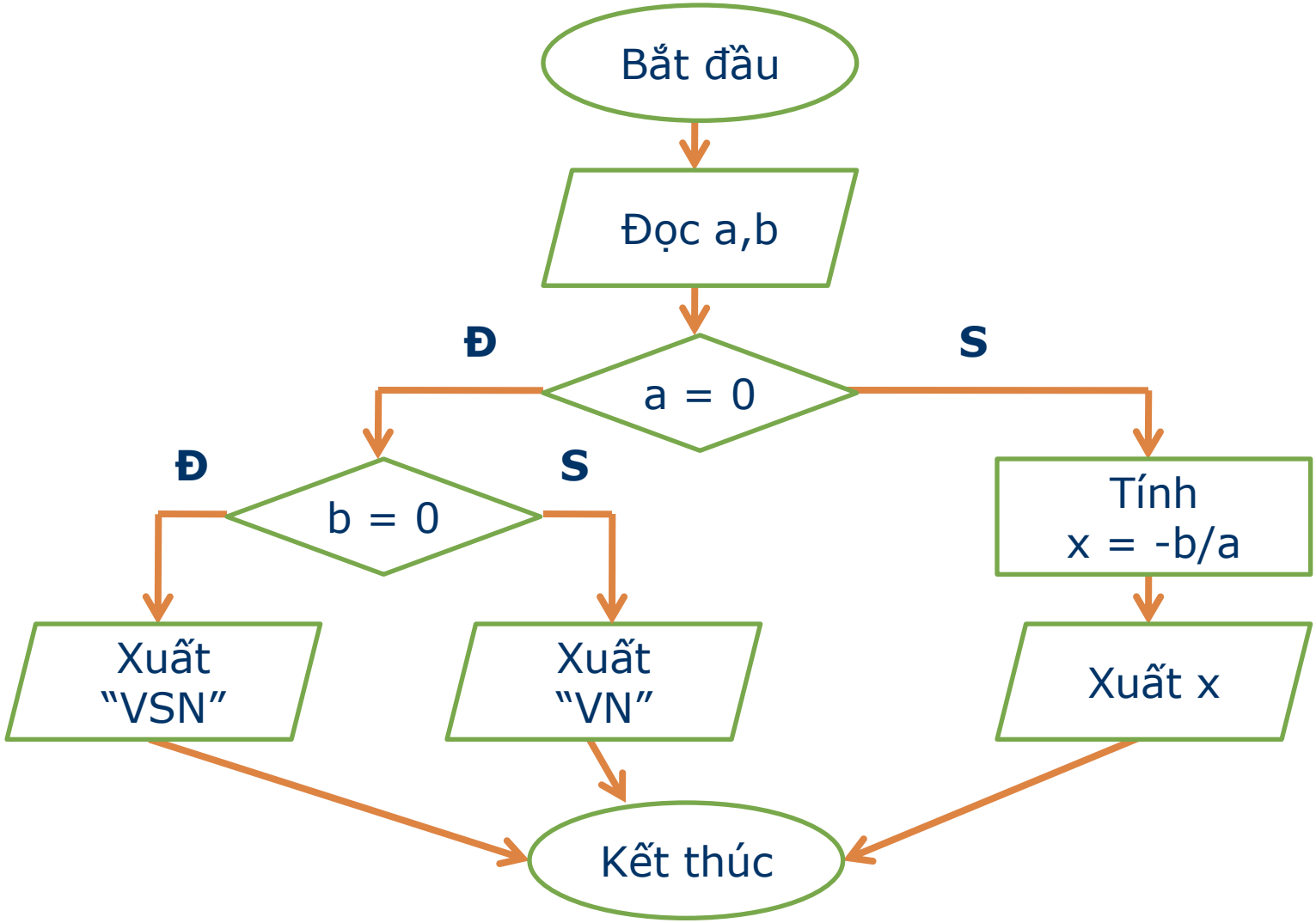


Khối lựa chọn
Tùy điều kiện sẽ rẽ nhánh.

Khối thao tác
Ghi thao tác cần thực hiện.

Đường đi
Chỉ hướng thao tác tiếp theo.

Sử dụng lưu đồ - sơ đồ khối



Sử dụng mã giả

- Vay mượn ngôn ngữ nào đó (ví dụ Pascal) để biểu diễn thuật toán.

```
Đầu vào: a, b thuộc R
Đầu ra: nghiệm phương trình ax + b = 0
```

```
If a = 0 Then
Begin
      If b = 0 Then
            Xuất "Phương trình vô số nghiệm"
      Else
            Xuất "Phương trình vô nghiệm"
End
Else
      Xuất "Phương trình có nghiệm x = -b/a"
```

Cài đặt thuật toán bằng C/C++

```
#include <stdio.h>
#include <conio.h>

void main()
{
	int a, b;
	printf("Nhap a, b: ");
	scanf("%d%d", &a, &b);
	if (a == 0)
		if (b == 0)
			printf("Phương trình VSN");
		else
			printf("Phương trình VN");
	else
		printf("x = %.2f", -float(b)/a);
}
```

Bài tập

1. Thuật toán là gì? Trình bày các tính chất quan trọng của một thuật toán?
2. Các bước xây dựng chương trình?
3. Các cách biểu diễn thuật toán? Ưu và khuyết điểm của từng phương pháp?

   Cho ví dụ minh họa.

Bài tập

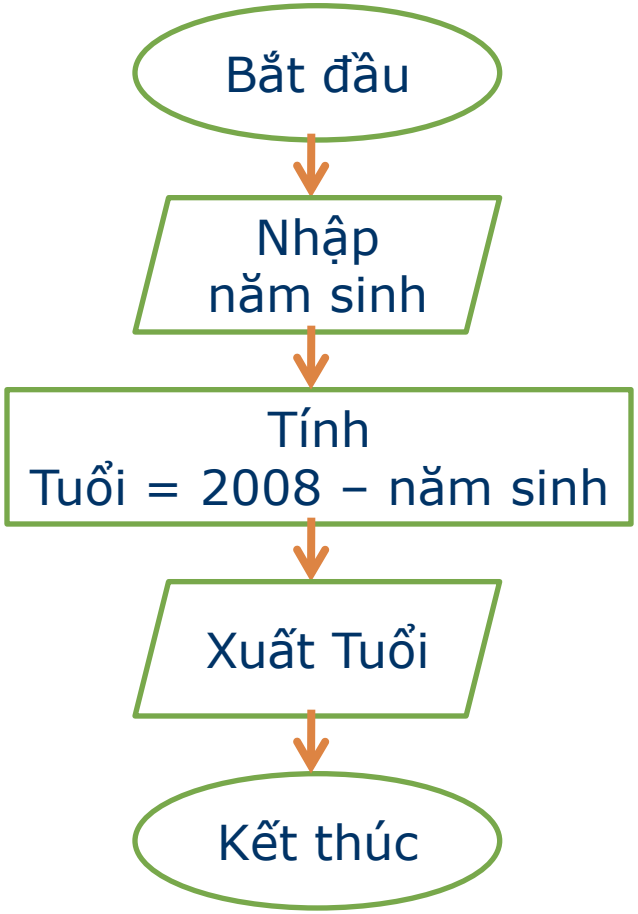
4. Nhập năm sinh của một người. Tính tuổi người đó.
5. Nhập 2 số a và b. Tính tổng, hiệu, tính và thương của hai số đó.
6. Nhập tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:
   a. tiền = số lượng * đơn giá
   b. thuế giá trị gia tăng = 10% tiền

Bài tập

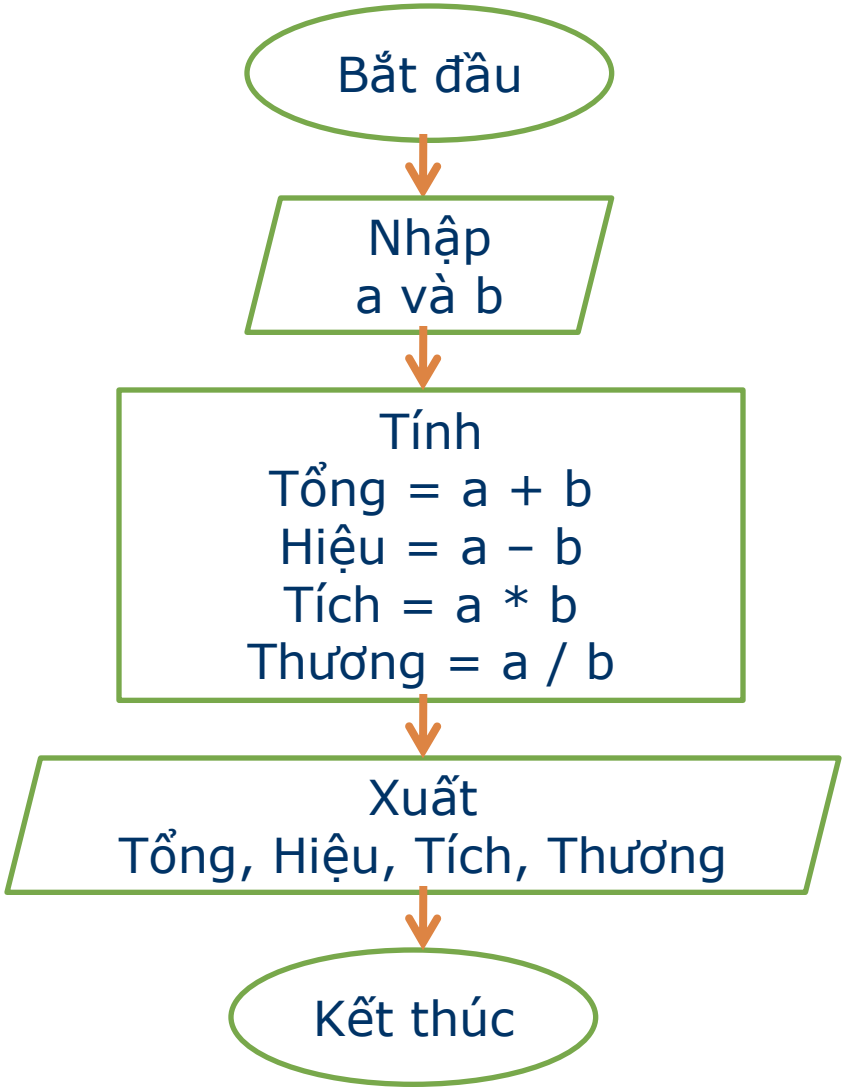
7. Nhập điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.
8. Nhập bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.
9. Nhập vào số xe (gồm 4 chữ số) của bạn. biết số xe của bạn được mấy nút?
10. Nhập vào 2 số nguyên.

    Tính min và max của hai số đó.

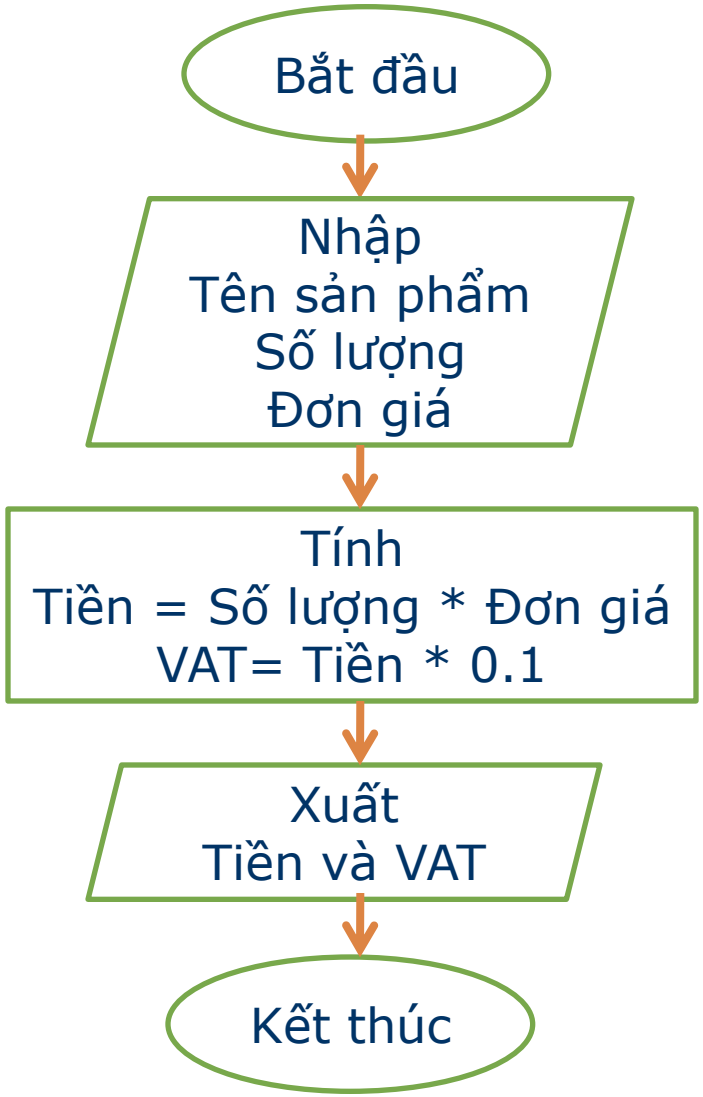
Bài tập 4


Bắt đầu
Nhập
năm sinh
Tính
Tuổi = 2008 – năm sinh
Xuất Tuổi
Kết thúc

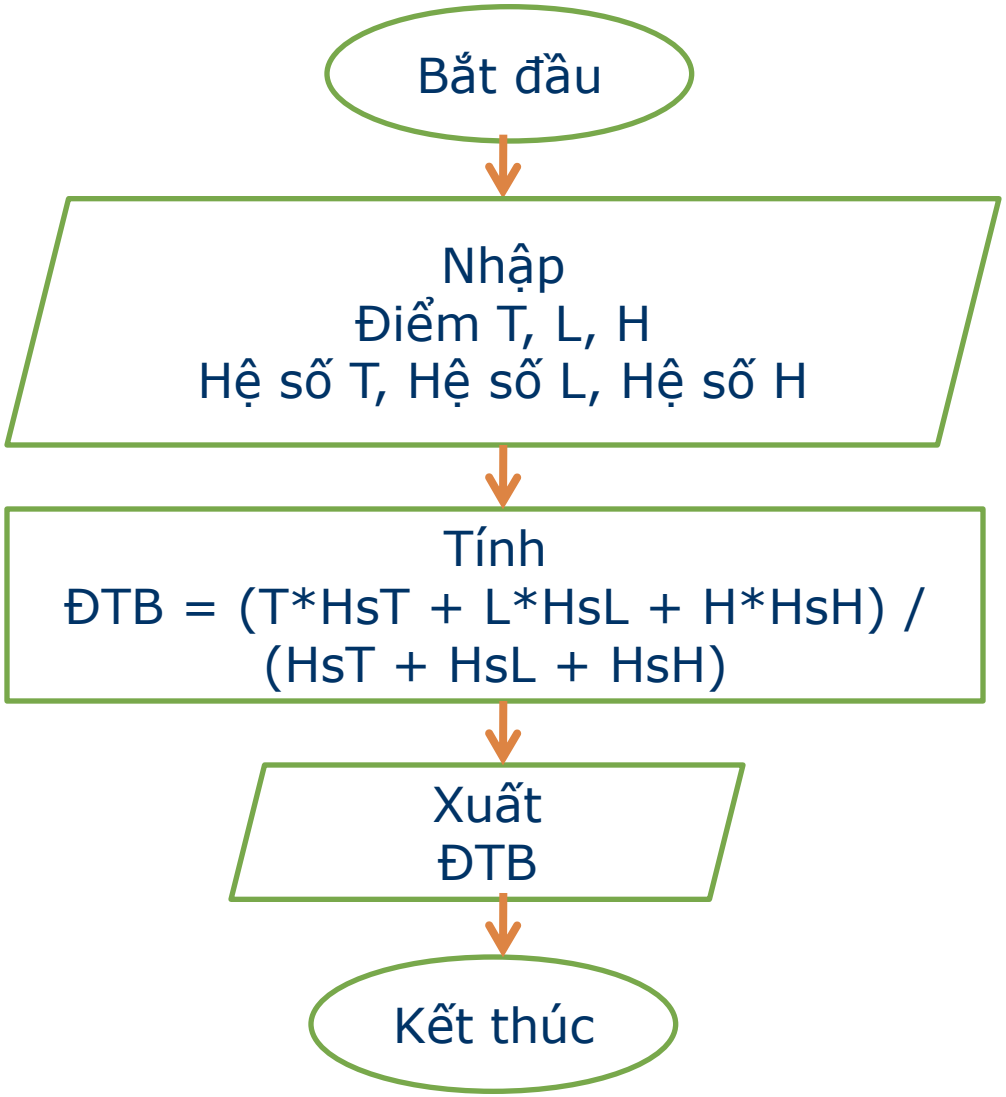
Bài tập 5


Bắt đầu
Nhập
a và b
Tính
Tổng = a + b
Hiệu = a – b
Tích = a * b
Thương = a / b
Xuất
Tổng, Hiệu, Tích, Thương
Kết thúc

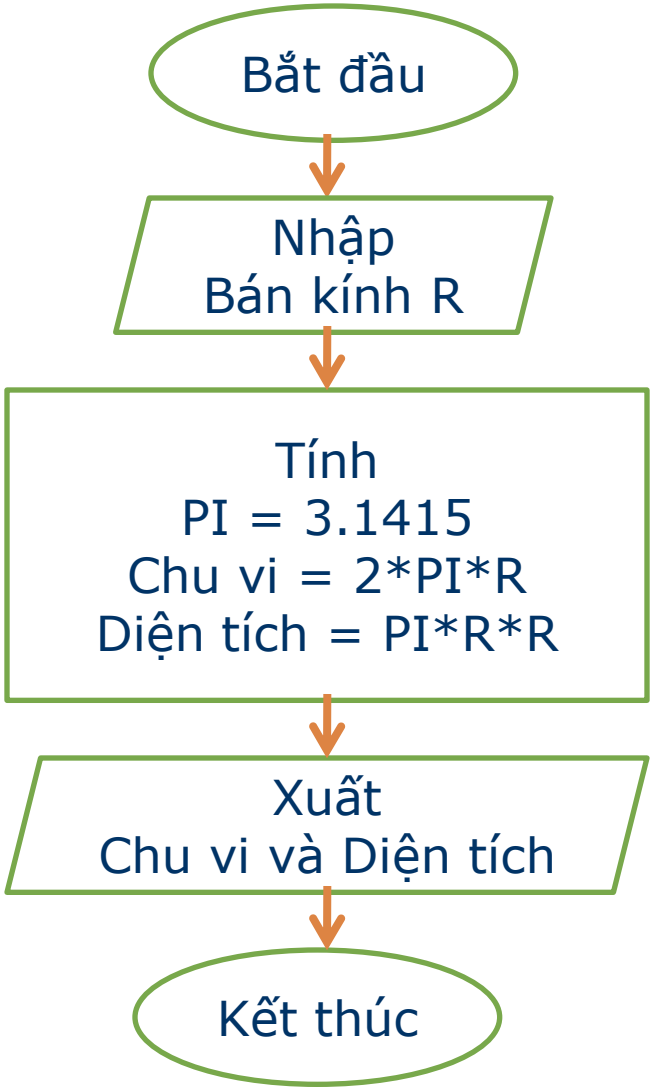
Bài tập 6


Bắt đầu
Nhập
Tên sản phẩm
Số lượng
Đơn giá
Tính
Tiền = Số lượng * Đơn giá
VAT= Tiền * 0.1
Xuất
Tiền và VAT
Kết thúc

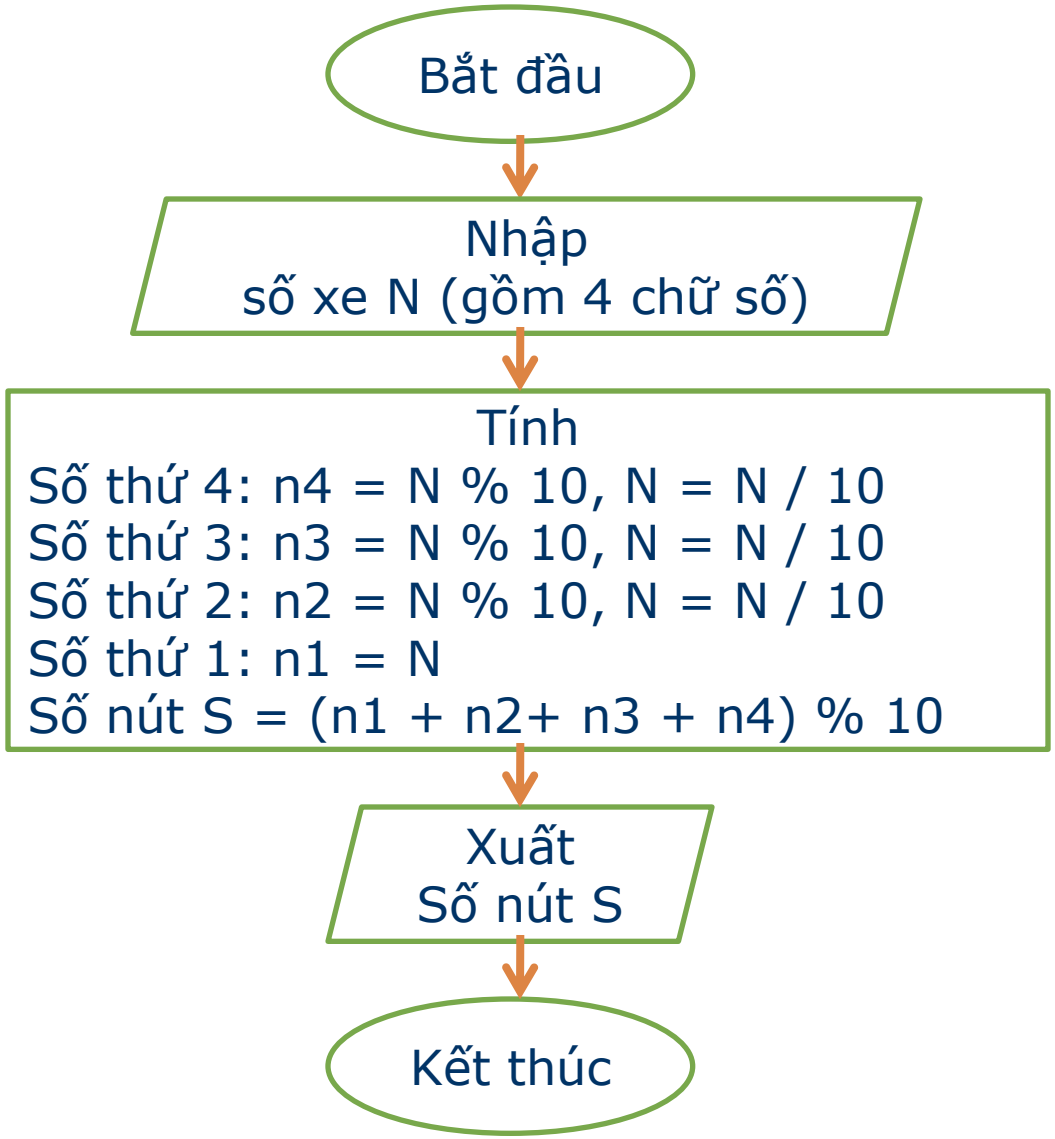
Bài tập 7


Bắt đầu
Nhập
Điểm T, L, H
Hệ số T, Hệ số L, Hệ số H
Tính
ĐTB = (T*HsT + L*HsL + H*HsH) /
(HsT + HsL + HsH)
Xuất
ĐTB
Kết thúc

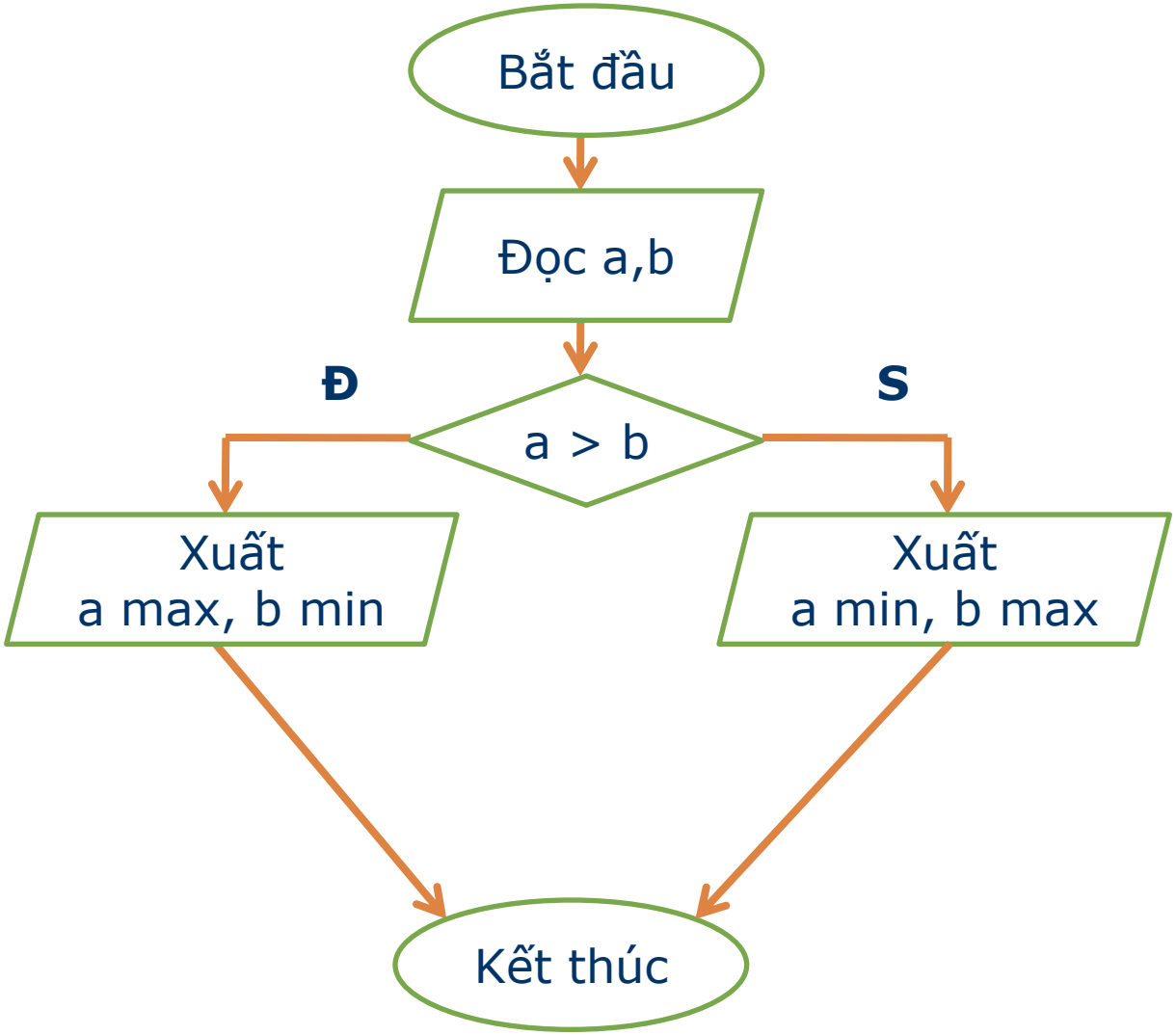
Bài tập 8


Bắt đầu
Nhập
Bán kính R
Tính
PI = 3.1415
Chu vi = 2*PI*R
Diện tích = PI*R*R
Xuất
Chu vi và Diện tích
Kết thúc

Bài tập 9


Bắt đầu
Nhập
số xe N (gồm 4 chữ số)
Tính
Số thứ 4: n4 = N % 10, N = N / 10
Số thứ 3: n3 = N % 10, N = N / 10
Số thứ 2: n2 = N % 10, N = N / 10
Số thứ 1: n1 = N
Số nút S = (n1 + n2+ n3 + n4) % 10
Xuất
Số nút S
Kết thúc

Bài tập 10



Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÂU LỆNH LẶP

Nội dung

1. Câu lệnh for
2. Câu lệnh while
3. Câu lệnh do… while
4. Một số kinh nghiệm lập trình

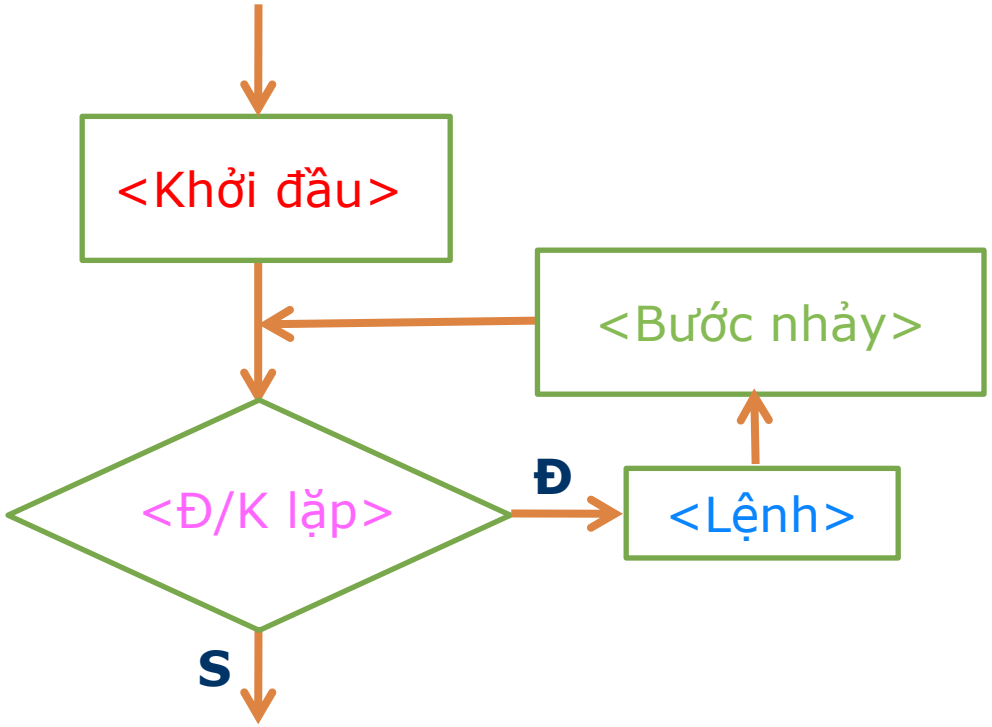
Đặt vấn đề

- Ví dụ
  - Viết chương trình xuất các số từ 1 đến 10

    => Sử dụng 10 câu lệnh printf
  - Viết chương trình xuất các số từ 1 đến 1000

    => Sử dụng 1000 câu lệnh printf !
- Giải pháp
  - Sử dụng cấu trúc lặp lại một hành động trong khi còn thỏa một điều kiện nào đó.
  - 3 lệnh lặp: for, while, do… while

Câu lệnh for



for (<Khởi đầu>; <Đ/K lặp>; <Bước nhảy>)

<Lệnh>;

<Khởi đầu>, <Đ/K lặp>, <Bước nhảy>: là biểu thức C bất kỳ có chức năng riêng

<Lệnh>: đơn hoặc khối lệnh.

Câu lệnh for

```
void main()
{
	int i;
	for (i = 0; i < 10; i++)
		printf("%d\n", i);

	for (int j = 0; j < 10; j = j + 1)
		printf("%d\n", j);

	for (int k = 0; k < 10; k += 2)
	{
		printf("%d", k);
		printf("\n");
	}
}
```

Câu lệnh for - Một số lưu ý

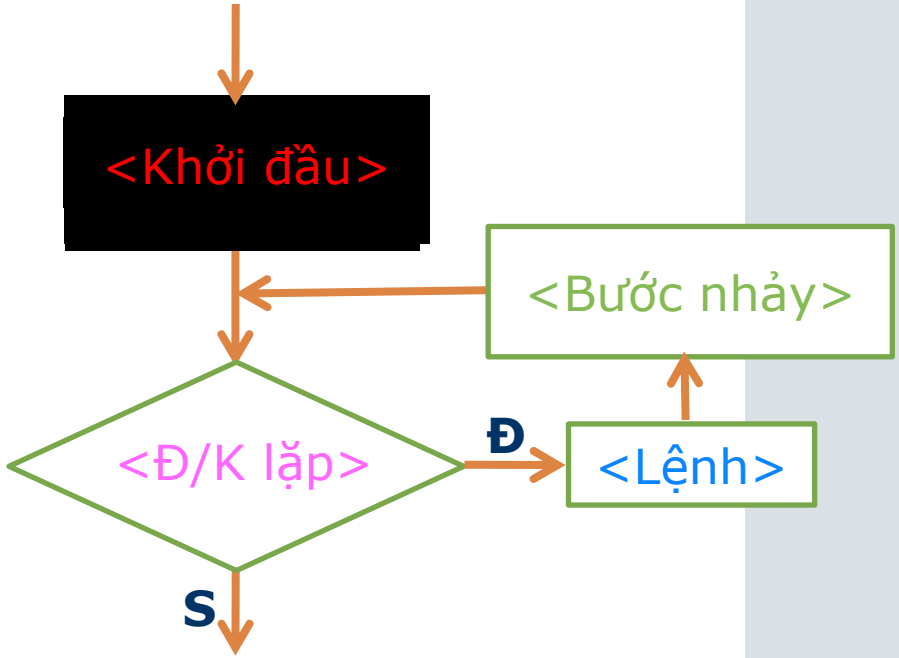
- Câu lệnh for là một câu lệnh đơn và có thể lồng nhau.

```
if (n < 10 && m < 20)
{
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < m; j++)
                {
                        printf("%d", i + j);
                        printf("\n");
                }
        }
}
```

Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Khởi đầu>

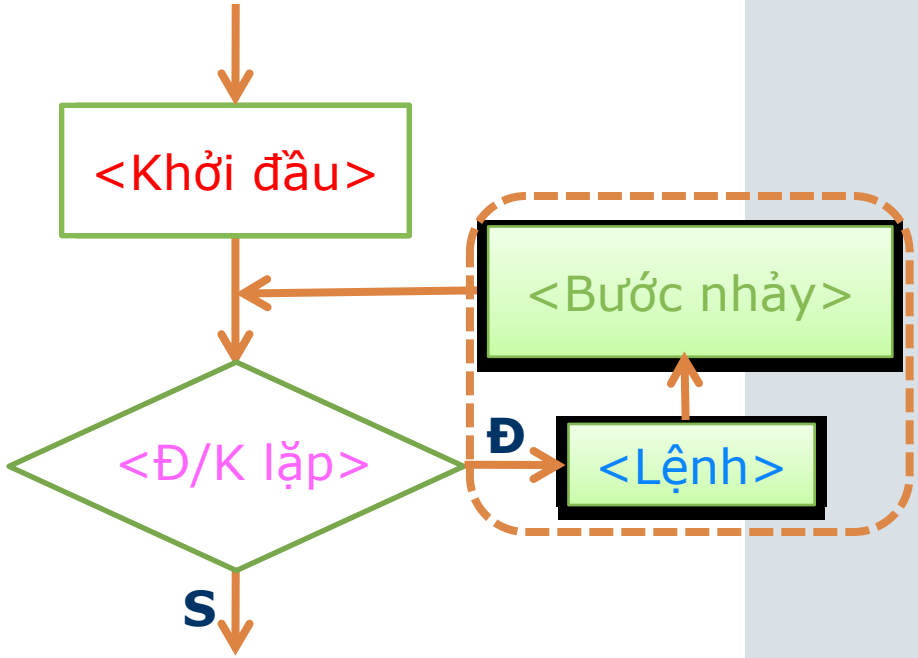
```
int i;
for (i = 0; i < 10; i++)
      printf("%d\n", i);

int i = 0;
for (; i < 10; i++)
      printf("%d\n", i);
```



Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Bước nhảy>

```
int i;
for (i = 0; i < 10; i++)
      printf("%d\n", i);

for (i = 0; i < 10; )
{
      printf("%d\n", i);
      i++;
}
```



Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Đ/K lặp>

```
int i;
for (i = 0; i < 10; i++)
      printf("%d\n", i);

for (i = 0; ; i++)
      printf("%d\n", i);

for (i = 0; ; i++)
{
      if (i >= 10)
            break;
      printf("%d\n", i);
}
```

Câu lệnh for - Một số lưu ý

- Lệnh break làm kết thúc câu lệnh.
- Lệnh continue bỏ qua lần lặp hiện tại.

```
for (i = 0; i < 10; i++)
{
        if (i % 2 == 0)
                break;
        printf("%d\n", i);
}

for (i = 0; i < 10; i++)
{
        if (i % 2 == 0)
                continue;
        printf("%d\n", i);
}
```

Câu lệnh for - Một số lưu ý

- Không được thêm ; ngay sau lệnh lệnh for.

  => Tương đương câu lệnh rỗng.

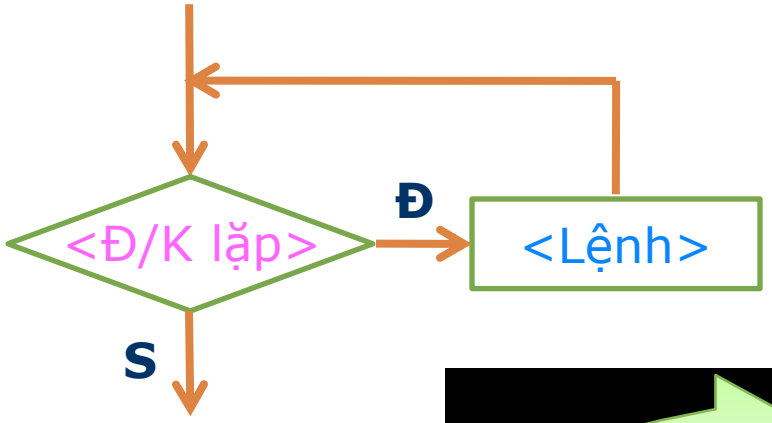
```
for (i = 0; i < 10; i++);
{
      printf("%d", i);
      printf("\n");
}

for (i = 0; i < 10; i++)
{
};
{
      printf("%d", i);
      printf("\n");
}
```

Câu lệnh for - Một số lưu ý

- Các thành phần <Khởi đầu>, <Đ/K lặp>, <Bước nhảy> cách nhau bằng dấu ;
- Nếu có nhiều thành phần trong mỗi phần thì được cách nhau bằng dấu ,

```
for (int i = 1, j = 2; i + j < 10; i++, j += 2)
      printf("%d\n", i + j);
```

Câu lệnh while



while (<Đ/K lặp>**)**

<Lệnh>;

Biểu thức C bất kỳ, thường là biểu thức quan hệ cho kết quả 0 (sai) và != 0 (đúng)

Câu lệnh đơn hoặc Câu lệnh phức (kẹp giữa { và })

Câu lệnh while

```
int i = 0;
while (i < 10)
{
        printf("%d\n", i);
        i++;
}

for (int i = 0; i < 10; i++)
        printf("%d\n", i);

int i = 0;
for (; i < 10; )
{
        printf("%d\n", i);
        i++;
}
```

Câu lệnh while - Một số lưu ý

- Câu lệnh while là một câu lệnh đơn và có thể lồng nhau.

```
if (n < 10 && m < 20)
{
        while (n >= 1)
        {
                while (m >= 1)
                {
                        printf("%d", m);
                        m--;
                }
                n--;
        }
}
```

Câu lệnh while - Một số lưu ý

- Câu lệnh while có thể không thực hiện lần nào do điều kiện lặp ngay từ lần đầu đã không thỏa.

```
void main()
{
        int n = 1;
        while (n > 10)
        {
                printf("%d\n", n);
                n--;
        }
        …
}
```

Câu lệnh for - Một số lưu ý

- Không được thêm ; ngay sau lệnh lệnh while.

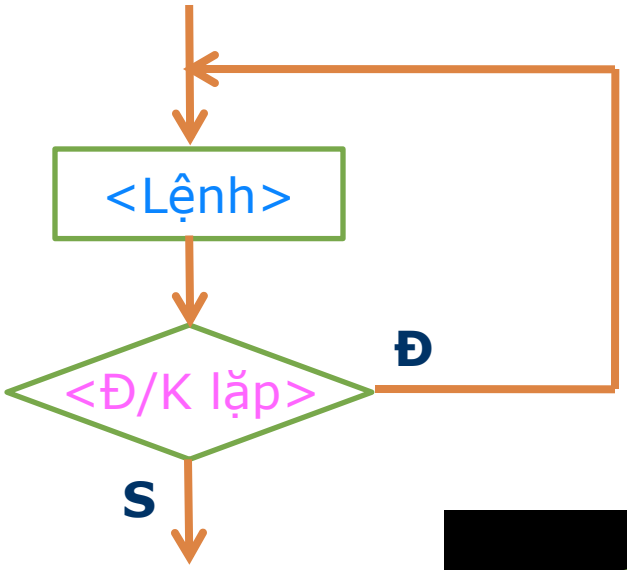
```
int n = 0;
while (n < 10);
{
        printf("%d\n", n);
        n++;
}

while (n < 10)
{
};
{
        printf("%d\n", n);
        n++;
}
```

Câu lệnh while - Một số lưu ý

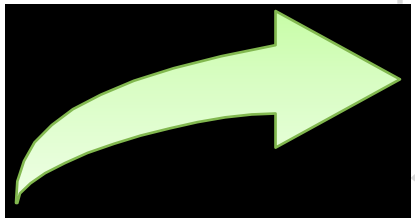
- Câu lệnh while có thể bị lặp vô tận (loop)

```
void main()
{
        int n = 1;
        while (n < 10)
        {
                printf("%d", n);
                n--;
        }

        n = 1;
        while (n < 10)
                printf("%d", n);
}
```

Câu lệnh do… while

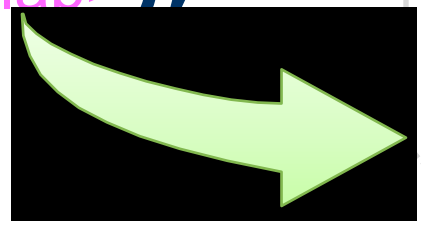


do



<Lệnh>;

while (<Đ/K lặp>**);**



Câu lệnh đơn hoặc Câu lệnh phức (kẹp giữa { và })

Biểu thức C bất kỳ, thường là biểu thức quan hệ cho kết quả 0 (sai) và != 0 (đúng)

Câu lệnh do... while

```
int i = 0;
do
{
        printf("%d\n", i);
        i++;
}
while (i < 10);

int i = 0;
printf("%d\n", i);
i++;
for (; i < 10; )
{
        printf("%d\n", i);
        i++;
}
```

Câu lệnh do… while - Một số lưu ý

- Câu lệnh do… while là một câu lệnh đơn và có thể lồng nhau.

```
int a = 1, b;
do
{
        b = 1;
        do
        {
                printf("%d\n", a + b);
                b = b + 2;
        }
        while (b < 20);
        a++;
}
while (a < 20);
```

Câu lệnh do… while - Một số lưu ý

- Câu lệnh do… while sẽ được thực hiện ít nhất 1 lần do điều kiện lặp được kiểm tra ở cuối.

```
void main()
{
        int n;
        do
        {
                printf("Nhap n: ");
                scanf("%d", &n);
        }
        while (n < 1 || n > 100);
}
```

Câu lệnh do… while - Một số lưu ý

- Câu lệnh do… while có thể bị lặp vô tận (loop)

```
…
        int n = 1;
        do
        {
                printf("%d", n);
                n--;
        }
        while (n < 10);

        n = 1;
        do
                printf("%d", n);
        while (n < 10);
…
```

for, while, do... while

- Đều có khả năng lặp lại nhiều hành động.

```
int n = 10;
for (int i = 1; i <= n; i++)
      printf("%d\n", i);

int i = 1;
while (i <= n)
{
      printf("%d\n", i); i++;
}

int i = 1;
do {
      printf("%d\n", i); i++;
} while (i < n);
```

for, while, do… while

- Số lần lặp xác định ngay trong câu lệnh for

```
int n = 10;
for (int i = 1; i <= n; i++)
        …;

int i = 1;
while (i <= n)
{
        …;
}

int i = 1;
do {
        …;
} while (i > n);
```

while & do… while

- while có thể không thực hiện lần nào.
- do… while sẽ được thực hiện ít nhất 1 lần.

```
int n = 100;
while (n < 10)
{
        …;
}
…
do
{
        printf("Nhap n: ");
        scanf("%d", &n);
}
while (n > 10);
```

Bài tập

1. Nhập một số nguyên dương n (n > 0).

   Hãy cho biết:

   a. Có phải là số đối xứng? Ví dụ: 121, 12321, …
   b. Có phải là số chính phương? Ví dụ: 4, 9, 16, …
   c. Có phải là số nguyên tố? Ví dụ: 2, 3, 5, 7, …
   d. Chữ số lớn nhất và nhỏ nhất?
   e. Các chữ số có tăng dần hay giảm dần kh

Bài tập

2. Nhập một số nguyên dương n. Tính:
   - a. $S = 1 + 2 + \ldots + n$
   - b. $S = 1^2 + 2^2 + \ldots + n^2$
   - c. $S = 1 + 1/2 + \ldots + 1/n$
   - d. $S = 1*2*\ldots*n = n!$
   - e. $S = 1! + 2! + \ldots + n!$
3. Nhập 3 số nguyên a, b và n với a, b < n. Tín
   các số nguyên dương nhỏ hơn n chia hết ch
   nhưng không chia hết cho b.
4. Tính tổng các số nguyên tố nhỏ hơn n (0 < n

Bài tập

5. Nhập một số nguyên dương n. Xuất ra số ngược lại. Ví dụ: Nhập 1706 ➔ Xuất 6071.
6. Tìm và in lên màn hình tất cả các số nguyên trong phạm vi từ 10 đến 99 sao cho tích của 2 chữ số bằng 2 lần tổng của 2 chữ số đó.
7. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b nhập từ bàn phím.
8. Nhập n. In n số đầu tiên trong dãy Fibonacy.
   a. $a_0 = a_1 = 1$
   b. $a_n = a_{n-1} + a_{n-2}$

Bài tập 1a

```
void main()
{
        int n, sogoc, sodao, donvi;
        printf("Nhap n: ");
        scanf("%d", &n);

        sogoc = n; sodao = 0;
        while (sogoc > 0)
        {
                donvi = sogoc % 10;
                sodao = sodao*10 + donvi;
                sogoc = sogoc / 10;
        }
        if (sodao == n) printf("Doi xung");
        else printf("Khong doi xung");
}
```

Bài tập 1b

```
#include <math.h>

void main()
{
	int n, n_can_nguyen;

	printf("Nhap n: ");
	scanf("%d", &n);

	n_can_nguyen = int(sqrt(n));
	if (n_can_nguyen*n_can_nguyen == n)
		printf("%d la so CP", n);
	else
		printf("%d khong la so CP", n);
}
```

Bài tập 1c

```
void main()
{
	int n, i, souoc;

	printf("Nhap n: ");
	scanf("%d", &n);

	souoc = 0;
	for (i = 1; i <= n; i++)
		if (n % i == 0)
			souoc++;

	if (souoc == 2)
		printf("%d la so nguyen to");
	else
		printf("%d ko la so nguyen to", n);
}
```

Bài tập 1d

```
void main()
{
        int n, min, max, donvi;
        …
        min = n % 10;
        max = min;
        n = n / 10;

        while (n>0)
        {
                donvi = n % 10;
                n = n / 10;
                if (donvi < min) min = donvi;
                if (donvi > max) max = donvi;
        }
        printf("So NN = %d, So LN = %d", min, max);
}
```

Bài tập 1e

```
void main()
{
	int n, sotruoc, sosau;
	…	// Nhập n
	sotruoc = n % 10;
	do
	{
		sosau = sotruoc;
		n = n / 10;
		sotruoc = n % 10;
	} while (n != 0 && sotruoc < sosau);

	if (sotruoc < sosau)
		printf("Cac chu so tang dan");
	else
		printf("Cac chu so ko tang dan");
}
```

Bài tập 2a

```
void main()
{
        int n, i, s;

        printf("Nhap n: ");
        scanf("%d", &n);

        s = 0;
        for (i = 1; i <= n; i++)
                s = s + i;

        printf("1 + 2 + … + %d = %d", n, s);
}
```

Bài tập 2b

```
void main()
{
        int n, i, s;

        printf("Nhap n: ");
        scanf("%d", &n);

        s = 0;
        for (i = 1; i <= n; i++)
                s = s + i*i;

        printf("1^2 + 2^2 + … + %d^2 = %d", n, s);
}
```

Bài tập 2c

```
void main()
{
        int n, i;
        float s;

        printf("Nhap n: ");
        scanf("%d", &n);

        s = 0;
        for (i = 1; i <= n; i++)
                s = s + 1.0/i;

        printf("1 + 1/2 + … + 1/%d = %f", n, s);
}
```

Bài tập 2d

```
void main()
{
	int n, i, s;

	printf("Nhap n: ");
	scanf("%d", &n);

	s = 1;
	for (i = 2; i <= n; i++)
		s = s * i;

	printf("%d! = %d", n, s);
}
```

Bài tập 2e

```
void main()
{
	int n, i, j, igt, s;
	printf("Nhap n: ");
	scanf("%d", &n);

	s = 0;
	for (i = 1; i <= n; i++)
	{
		igt = 1;
		for (j = 2; j <= i; j++)
			igt = igt * j;
		s = s + igt;
	}
	printf("1! + 2! + … + %d! = %d", n, s);
}
```

Bài tập 3

```
void main()
{
	int a, b, n, i, s;
	do
	{
		printf("Nhap a, b, n: ");
		scanf("%d%d%d", &a, &b, &n);
	} while (a >= n || b >= n);

	s = 0;
	for (i = 1; i <=  n - 1; i++)
		if (i % a == 0 && i % b != 0)
			s = s + i;

	printf("Tong cac thoa yeu cau la %d", s);
}
```

Bài tập 4

```
void main()
{
	int n, i, j, souoc, s;
	do
	{
		printf("Nhap n: ");
		scanf("%d", &n);
	} while (n <= 0 || n >= 50);
	s = 0;
	for (i = 2; i <= n - 1; i++)
	{
		…	// Đếm số ước của i
		if (souoc == 2)	// Là số nguyên tố
			s = s + i;
	}
	printf("Tong cac so nt < %d la %d", n, s);
}
```

Bài tập 5

```
void main()
{
	int n, donvi;

	printf("Nhap n: ");
	scanf("%d", &n);

	printf("So dao cua %d la ", n);
	while (n > 0)
	{
		donvi = n % 10;
		n = n / 10;
		printf("%d", donvi);
	}
}
```

Bài tập 6

```
void main()
{
	int n, i, donvi, chuc;

	printf("Cac so thoa yeu cau la: ");
	for (i = 10; i <= 99; i++)
	{
		donvi = i % 10;
		chuc = i / 10;
		if (chuc*donvi == 2*(chuc + donvi))
			printf("%d", i);
	}
}
```

Bài tập 7

- Ví dụ: a = 12, b = 8
- Cách 1:
  - Cho 1 biến i chạy từ 8 trở về 1, nếu cả a và b đều chia hết cho i thì dừng và i chính là uscln.
  - 8, 7, 6, 5, 4 => USCLN của 12 và 8 là 4.
- Cách 2:
  - USCLN của a & b (a khác b), ký hiệu (a, b) là:
    - (a – b, b) nếu a > b
    - (a, b – a) nếu b > a
  - (12, 8) = (4, 8) = (4, 4) = 4

Bài tập 7

```
void main()
{
	int a, b, uscln;

	printf("Nhap a va b: ");
	scanf("%d%d", &a, &b);

	if (a < b) uscln = a;
	else uscln = b;

	while (a % uscln != 0 || b % uscln != 0)
		uscln--;

	printf("USCLN cua %d va %d la %d",a, b, uscln);
}
```

Bài tập 7

```
void main()
{
	int a, b;

	printf("Nhap a va b: ");
	scanf("%d%d", &a, &b);

	while (a <> b)
	{
		if (a > b)
			a = a - b;
		else
			b = b - a;
	}
	printf("USCLN cua a va b la %d', a);
}
```

Bài tập 8

- Dãy Fibonacy: $a_0\ a_1\ a_2\ \ldots\ a_{n-2}\ a_{n-1}\ a_n$
  - Với $a_0 = a_1 = 1$, $a_n = a_{n-1} + a_{n-2}$
- Ví dụ: 1 1 2 3 5 8 13 21 …
- Xuất n phần tử đầu tiên của dãy Fibonacy
  - n = 1 => 1, n = 2 => 1 1
  - n > 2
    - Lưu lại 2 phần tử trước nó là a và b
    - Mỗi lần tính xong cập nhật lại a và b.
- Nên thêm 2 phần tử ảo đầu tiên là a_{-2}, a_{-1}
  - 1 0 1 1 2 3 5 8 13 21 …

Bài tập 8

```
void main()
{
	int n, an, an1, an2, i;

	printf("Nhap n: ");
	scanf("%d", &n);

	an2 = 1; an1 = 0;
	printf("%d phan tu dau tien cua day: ", n);
	for (i = 1; i <= n; i++)
	{
		an = an2 + an1;
		printf("%d ", an);
		an2 = an1;
		an1 = an;
	}
}
```

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

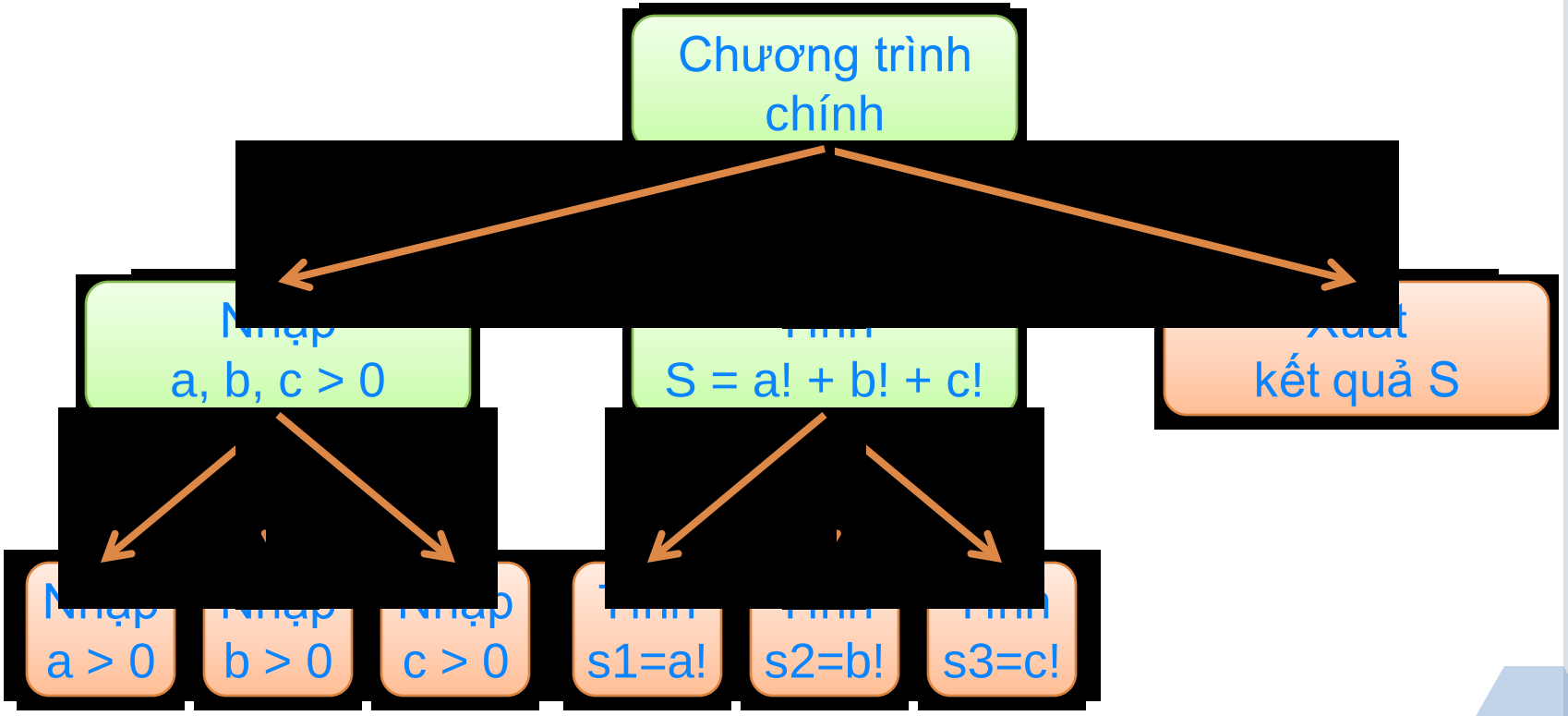
HÀM

Nội dung

1. Khái niệm và cú pháp
2. Tầm vực
3. Tham số và lời gọi hàm
4. Đệ quy

Đặt vấn đề

- Viết chương trình tính S = a! + b! + c! với a, b, c là 3 số nguyên dương nhập từ bàn phím.



Đặt vấn đề

- 3 đoạn lệnh nhập a, b, c > 0

```
do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &a);
} while (a <= 0);

do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &b);
} while (b <= 0);

do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &c);
} while (c <= 0);
```

Đặt vấn đề

- 3 đoạn lệnh tính s1 = a!, s2 = b!, s3 = c!

```
{ Tính s1 = a! = 1 * 2 * … * a }
s1 = 1;
for (i = 2; i <= a ; i++)
      s1 = s1 * i;

{ Tính s2 = b! = 1 * 2 * … * b }
s2 = 1;
for (i = 2; i <= b ; i++)
      s2 = s2 * i;

{ Tính s3 = c! = 1 * 2 * … * c }
s3 = 1;
for (i = 2; i <= c ; i++)
      s3 = s3 * i;
```

Đặt vấn đề

- Giải pháp => Viết 1 lần và sử dụng nhiều lần
  - Đoạn lệnh nhập tổng quát, với n = a, b, c

```
do {
        printf("Nhap mot so nguyen duong: ");
        scanf("%d", &n);
} while (n <= 0);
```

  - Đoạn lệnh tính giai thừa tổng quát, n = a, b, c

```
{ Tính s = n! = 1 * 2 * … * n }
s = 1;
for (i = 2; i <= n ; i++)
        s = s * i;
```

Hàm

Khái niệm

- Một đoạn chương trình có tên, đầu vào và đầu ra.
- Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
- Được gọi nhiều lần với các tham số khác nhau.
- Được sử dụng khi có nhu cầu:
  - Tái sử dụng.
  - Sửa lỗi và cải tiến.

Hàm

Cú pháp

```
<kiểu trả về> <tên hàm>([<danh sách tham số>])
{
        <các câu lệnh>
        [return <giá trị>;]
}
```

- Trong đó
  - <kiểu trả về> : kiểu bất kỳ của C (char, int, long, float,…). Nếu không trả về thì là void.
  - <tên hàm>: theo quy tắc đặt tên định danh.
  - <danh sách tham số> : tham số hình thức đầu vào giống khai báo biến, cách nhau bằng dấu ,
  - <giá trị> : trả về cho hàm qua lệnh return.

Các bước viết hàm

- Cần xác định các thông tin sau đây:
  - Tên hàm.
  - Hàm sẽ thực hiện công việc gì.
  - Các đầu vào (nếu có).
  - Đầu ra (nếu có).

Hàm

- Ví dụ 1
  - Tên hàm: XuatTong
  - Công việc: tính và xuất tổng 2 số nguyên
  - Đầu vào: hai số nguyên x và y
  - Đầu ra: không có

```
void XuatTong(int x, int y)
{
        int s;
        s = x + y;
        printf("%d cong %d bang %d", x, y, s);
}
```

Hàm

- Ví dụ 2
  - Tên hàm: TinhTong
  - Công việc: tính và trả về tổng 2 số nguyên
  - Đầu vào: hai số nguyên x và y
  - Đầu ra: một số nguyên có giá trị x + y

```
int TinhTong(int x, int y)
{
        int s;
        s = x + y;
        return s;
}
```

Chương trình con - Function

- Ví dụ 3
  - Tên hàm: NhapXuatTong
  - Công việc: nhập và xuất tổng 2 số nguyên
  - Đầu vào: không có
  - Đầu ra: không có

```
void NhapXuatTong()
{
        int x, y;
        printf("Nhap 2 so nguyen: ");
        scanf("%d%d", &x, &y);
        printf("%d cong %d bang %d", x, y, x + y);
}
```

Tầm vực

- Khái niệm
  - Là phạm vi hiệu quả của biến và hàm.
  - Biến:
    - Toàn cục: khai báo trong ngoài tất cả các hàm (kể cả hàm main) và có tác dụng lên toàn bộ chương trình.
    - Cục bộ: khai báo trong hàm hoặc khối { } và chỉ có tác dụng trong bản thân hàm hoặc khối đó (kể cả khối con nó). Biến cục bộ sẽ bị xóa khỏi bộ nhớ khi kết thúc khối khai báo nó.

Tầm vực

```
int a;
        int Ham1()
        {
                int a1;
        }

        int Ham2()
        {
                int a2;
                {
                        int a21;
                }
        }

        void main()
        {
                int a3;
        }
```

Một số lưu ý

- Thông thường người ta thường đặt phần tiêu đề hàm/nguyên mẫu hàm (prototype) trên hàm main và phần định nghĩa hàm dưới hàm main.

```
void XuatTong(int x, int y);  // prototype

void main()
{
        …
}

void XuatTong(int x, int y)
{
        printf("%d cong %d bang %d", x, y, x + y);
}
```

Các cách truyền đối số

- Truyền Giá trị (Call by Value)
  - Truyền đối số cho hàm ở dạng giá trị.
  - Có thể truyền hằng, biến, biểu thức nhưng hàm chỉ sẽ nhận giá trị.
  - Được sử dụng khi không có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenGiaTri(int x)
{
        …
        x++;
}
```

Các cách truyền đối số

- Truyền Địa chỉ (Call by Address)
  - Truyền đối số cho hàm ở dạng địa chỉ (con trỏ).
  - Không được truyền giá trị cho tham số này.
  - Được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenDiaChi(int *x)
{
        …
        *x++;
}
```

Các cách truyền đối số

- Truyền Tham chiếu (Call by Reference) (C++)
  - Truyền đối số cho hàm ở dạng địa chỉ (con trỏ). Được bắt đầu bằng & trong khai báo.
  - Không được truyền giá trị cho tham số này.
  - Được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenThamChieu(int &x)
{
        …
        x++;
}
```

Lưu ý khi truyền đối số

- Lưu ý
  - Trong một hàm, các tham số có thể truyền theo nhiều cách.

```
void HonHop(int x, int &y)
{
        …
        x++;
        y++;
}
```

Lưu ý khi truyền đối số

- Lưu ý
  - Sử dụng tham chiếu là một cách để trả về giá trị cho chương trình.

```
int TinhTong(int x, int y)
{
	return x + y;
}
void TinhTong(int x, int y, int &tong)
{
	tong = x + y;
}
void TinhTongHieu(int x, int y, int &tong, int &hieu)
{
	tong = x + y; hieu = x - y;
}
```

Lời gọi hàm

Cách thực hiện

- Gọi tên của hàm đồng thời truyền các đối số (hằng, biến, biểu thức) cho các tham số theo đúng thứ tự đã được khai báo trong hàm.
- Các biến hoặc trị này cách nhau bằng dấu ,
- Các đối số này được được đặt trong cặp dấu ngoặc đơn ( )

<tên hàm> (<đối số 1>,… , <đối số n>);

Lời gọi hàm

- Ví dụ

```
{ Các hàm được khai báo ở đây }
void main()
{
        int n = 9;
        XuatTong(1, 2);
        XuatTong(1, n);
        TinhTong(1, 2);
        int tong = TinhTong(1, 2);
        TruyenGiaTri(1);
        TruyenGiaTri(n);
        TruyenDiaChi(1);
        TruyenDiaChi(&n);
        TruyenThamChieu(1);
        TruyenThamChieu(n);
}
```

Lời gọi chương trình con

- Ví dụ

```
void HoanVi(int &a, int &b);

void main()
{
        HoanVi(2912, 1706);
        int x = 2912, y = 1706;
        HoanVi(x, y);
}
void HoanVi(int &a, int &b)
{
        int tam = a;
        a = b;
        b = tam;
}
```

Đệ quy

- Khái niệm
  - Một chương trình con có thể gọi một chương trình con khác.
  - Nếu gọi chính nó thì được gọi là sự đệ quy.
  - Số lần gọi này phải có giới hạn (điểm dừng)
- Ví dụ
  - Tính S(n) = n! = 1*2*…*(n-1)*n
  - Ta thấy S(n) = S(n-1)*n
  - Vậy thay vì tính S(n) ta sẽ đi tính S(n-1)
  - Tương tự tính S(n-2), …, S(2), S(1), S(0) = 1

Đệ quy

- Ví dụ

```
int GiaiThua(int n)
{
	if (n == 0)
		return 1;
	else
		return GiaiThua(n - 1) * n;
}
int GiaiThua(int n)
{
	if (n > 0)
		return GiaiThua(n - 1) * n;
	else
		return 1;
}
```

Bài tập

1. Bài tập chương câu lệnh điều kiện và rẽ nhánh
   a. Viết hàm đổi một ký tự hoa sang ký tự thường.
   b. Viết thủ tục giải phương trình bậc nhất.
   c. Viết thủ tục giải phương trình bậc hai.
   d. Viết hàm trả về giá trị nhỏ nhất của 4 số nguyên.
   e. Viết thủ tục hoán vị hai số nguyên.
   f. Viết thủ tục sắp xếp 4 số nguyên tăng dần

Bài tập

2. Bài tập chương câu lệnh lặp. Hàm nhận vào một số nguyên dương n và thực hiện:

a. Trả về số đảo của số đó.
b. Có phải là số đối xứng (Trả về True/False)
c. Có phải là số chính phương.
d. Có phải là số nguyên tố.
e. Tổng các chữ số lẻ.
f. Tổng các chữ số nguyên tố.
g. Tổng các chữ số chính phương.

Bài tập

3. Bài tập chương Câu lệnh lặp. Hàm nhận vào một số nguyên dương n và thực hiện:


a. $S = 1 + 2 + \ldots + n$


b. $S = 1^2 + 2^2 + \ldots + n^2$


c. $S = 1 + 1/2 + \ldots + 1/n$


d. $S = 1 * 2 * \ldots * n$


e. $S = 1! + 2! + \ldots + n!$

4. Hàm trả về USCLN của 2 số nguyên.
5. In ra n phần tử của dãy Fibonacy.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

MẢNG MỘT CHIỀU

Nội dung

1. Khái niệm
2. Khai báo
3. Truy xuất dữ liệu kiểu mảng
4. Một số bài toán trên mảng 1 chiều

Đặt vấn đề

- Ví dụ
  - Chương trình cần lưu trữ 3 số nguyên?

    => Khai báo 3 biến int a1, a2, a3;
  - Chương trình cần lưu trữ 100 số nguyên?

    => Khai báo 100 biến kiểu số nguyên!
  - Người dùng muốn nhập n số nguyên?

    => Không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép lưu trữ một dãy các số nguyên và dễ dàng truy xuất.

Dữ liệu kiểu mảng

Khái niệm

- Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
- Biểu diễn một dãy các biến có cùng kiểu. Ví dụ: dãy các số nguyên, dãy các ký tự…
- Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
- NNLT C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

Khai báo biến mảng (tường minh)

Tường minh

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
<kiểu cơ sở> <tên biến mảng>[<N1>][<N2>]…[<Nn>];
```

- <N1>, …, <Nn> : số lượng phần tử của mỗi chiều.

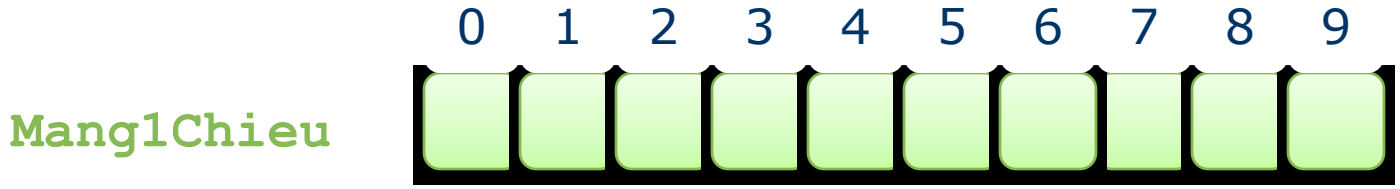
Lưu ý

- Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
- Mảng nhiều chiều: <tổng số phần tử> = N1*N2*…*Nn
- Bộ nhớ sử dụng = <tổng số phần tử>*sizeof(<kiểu cơ sở>)
- Bộ nhớ sử dụng phải ít hơn 64KB (65536 Bytes)
- Một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử>-1

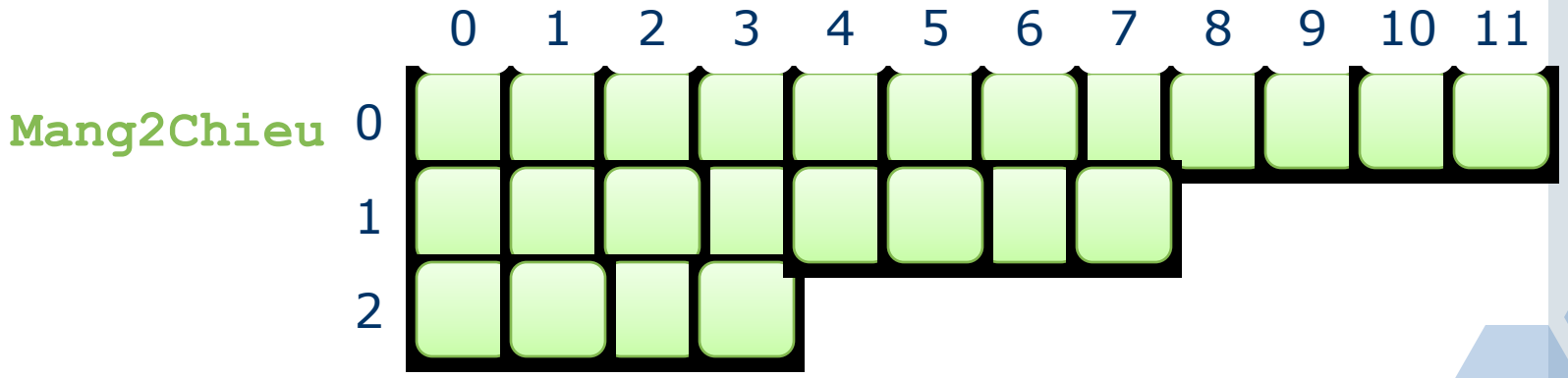
Khai báo biến mảng (tường minh)

Ví dụ

```
int Mang1Chieu[10];
```




```
int Mang2Chieu[3][4];
```



Khai báo biến mảng (kô tường minh)

Cú pháp

- Không tường minh (thông qua khai báo kiểu)

```
typedef <kiểu cơ sở> <tên kiểu mảng>[<số phần tử>];
typedef <kiểu cơ sở> <tên kiểu mảng>[<N1>]…[<Nn>];

<tên kiểu mảng> <tên biến mảng>;
```

Ví dụ

```
typedef int Mang1Chieu[10];
typedef int Mang2Chieu[3][4];

Mang1Chieu m1, m2, m3;
Mang2Chieu m4, m5;
```

Số phần tử của mảng

- Phải xác định cụ thể số phần tử ngay lúc khai báo, không được sử dụng biến hoặc hằng thường

```
int n1 = 10; int a[n1];
const int n2 = 20; int b[n2];
```

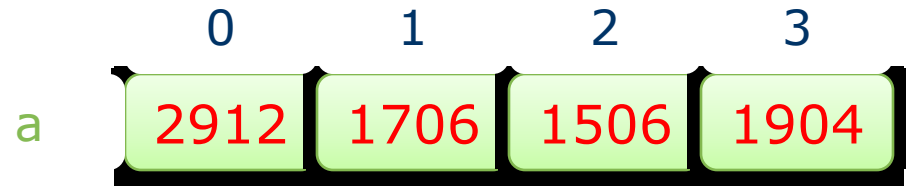
- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#define n1 10
#define n2 20
int a[n1];          // ⇔ int a[10];
int b[n1][n2];      // ⇔ int b[10][20];
```

Khởi tạo giá trị cho mảng lúc khai báo

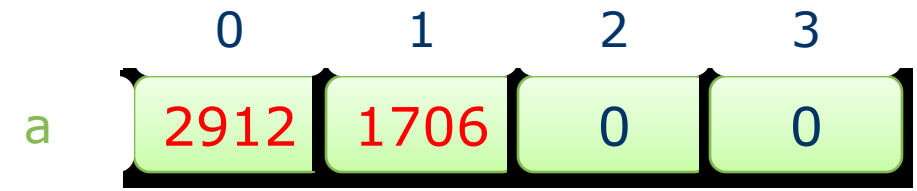
- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```



  - Khởi tạo giá trị cho một số phần tử đầu mảng

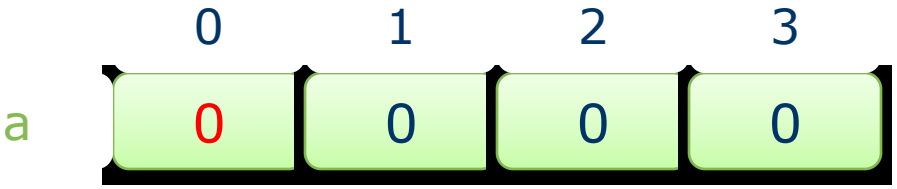
```
int a[4] = {2912, 1706};
```



Khởi tạo giá trị cho mảng lúc khai báo

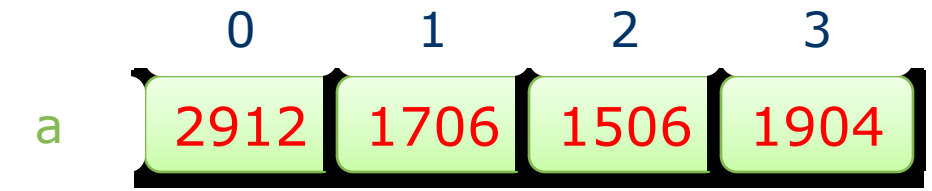
- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int a[4] = {0};
```



  - Tự động xác định số lượng phần tử

```
int a[] = {2912, 1706, 1506, 1904};
```



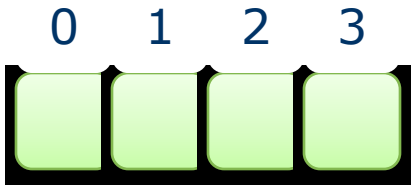
Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<gt cs1>][<gt cs2>]…[<gt csn>]
```

- Ví dụ
  - Cho mảng như sau

```
int a[4];
```



  - Các truy xuất
    - Hợp lệ: a[0], a[1], a[2], a[3]
    - Không hợp lệ: a[-1], a[4], a[5], …

      => Cho kết thường không như mong muốn!

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số thứ i>] = <giá trị>;
```

- Ví dụ

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;

b = a;          // Sai
for (int i = 0; i < 3; i++) b[i] = a[i];
```

Một số lỗi thường gặp

- Khai báo không chỉ rõ số lượng phần tử
  - int a[]; => int a[100];
- Số lượng phần tử liên quan đến biến hoặc hằng
  - int n1 = 10; int a[n1]; => int a[10];
  - const int n2 = 10; int a[n2]; => int a[10];
- Khởi tạo cách biệt với khai báo
  - int a[4]; a = {2912, 1706, 1506, 1904};
    => int a[4] = {2912, 1706, 1506, 1904};
- Chỉ số mảng không hợp lệ
  - int a[4];
  - a[-1] = 1; a[10] = 0;

Truyền mảng cho hàm

Truyền mảng cho hàm

- Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void SapXepTang(int a[100]);
```

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
  - Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void SapXepTang(int a[]);
void SapXepTang(int *a);
```

Truyền mảng cho hàm

- Truyền mảng cho hàm
  - Số lượng phần tử thực sự truyền qua biến khác

```
void SapXepTang(int a[100], int n);
void SapXepTang(int a[], int n);
void SapXepTang(int *a, int n);
```

- Lời gọi hàm

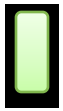
```
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
void main()
{
        int a[100], n;
        NhapMang(a, n);
        XuatMang(a, n);
}
```

Một số bài toán cơ bản

- Viết hàm thực hiện từng yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Tìm kiếm một phần tử trong mảng
  - Kiểm tra tính chất của mảng
  - Tách mảng / Gộp mảng
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - Sắp xếp mảng giảm dần/tăng dần
  - Thêm/Xóa/Sửa một phần tử vào mảng

Một số quy ước

- Số lượng phần tử



```
#define MAX 100
```

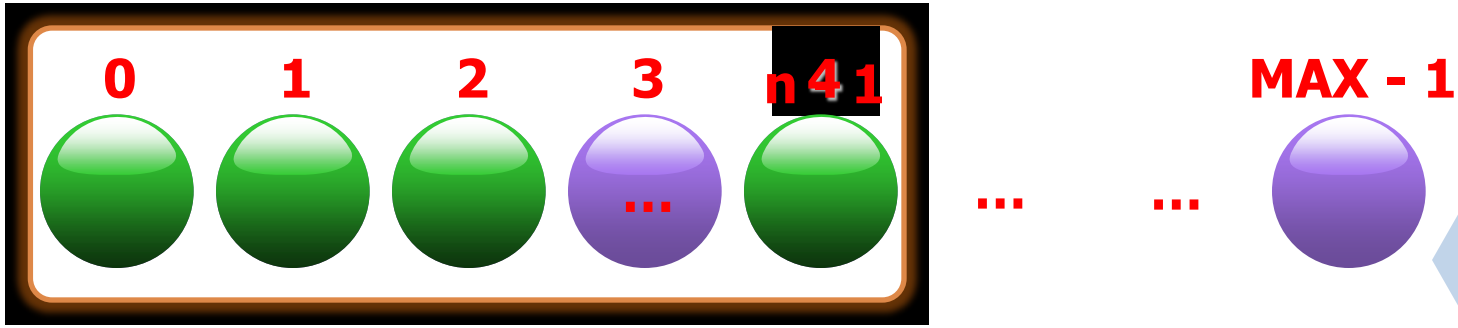
- Các hàm
  - Hàm void HoanVi(int &x, int &y): hoán vị giá trị của hai số nguyên.
  - Hàm int LaSNT(int n): kiểm tra một số có phải là số nguyên tố. Trả về 1 nếu n là số nguyên tố, ngược lại trả về 0.

Thủ tục HoanVi & Hàm LaSNT

```
void HoanVi(int &x, int &y)
{
	int tam = x; x = y; y = tam;
}

int LaSNT(int n)
{
	int i, dem = 0;
	for (i = 1; i <= n; i++)
		if (n%i == 0)
			dem++;

	if (dem == 2)
		return 1;
	else return 0;
}
```

Nhập mảng

- Yêu cầu
  - Cho phép nhập mảng a, số lượng phần tử n
- Ý tưởng
  - Cho trước một mảng có số lượng phần tử là MAX.
  - Nhập số lượng phần tử thực sự n của mảng.
  - Nhập từng phần tử cho mảng từ chỉ số 0 đến n – 1.

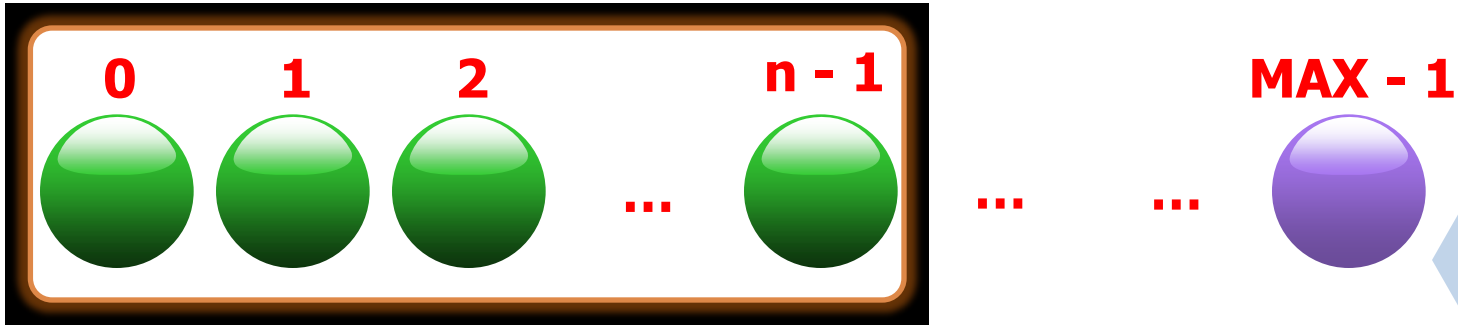


Hàm Nhập Mảng

```
void NhapMang(int a[], int &n)
{
        printf("Nhap so luong phan tu n: ");
        scanf("%d", &n);

        for (int i = 0; i < n; i++)
        {
                printf("Nhap phan tu thu %d: ", i);
                scanf("%d", &a[i]);
        }
}
```

Xuất mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Hãy xuất nội dung mảng a ra màn hình.
- Ý tưởng
  - Xuất giá trị từng phần tử của mảng từ chỉ số 0 đến n-1.



Hàm Xuất Mảng

```
void XuatMang(int a[], int n)
{
        printf("Noi dung cua mang la: ");

        for (int i = 0; i < n; i++)
              printf("%d ", a[i]);

        printf("\n");
}
```

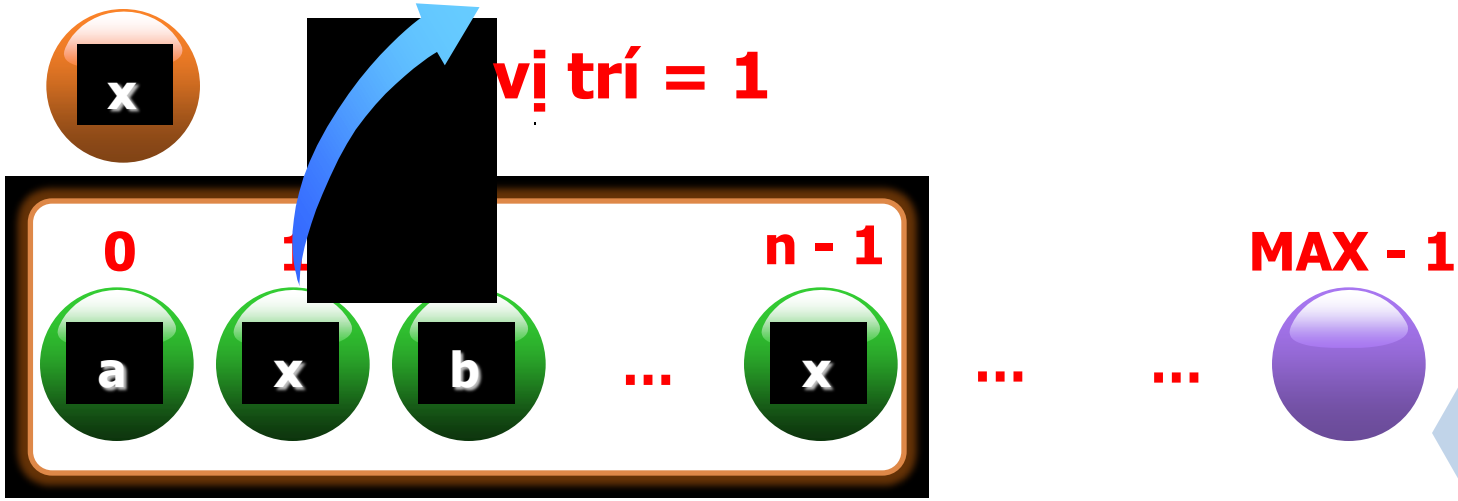
Tìm kiếm một phần tử trong mảng

Yêu cầu

- Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì nó nằm ở vị trí đầu tiên nào.

Ý tưởng

- Xét từng phần của mảng a. Nếu phần tử đang xét bằng x thì trả về vị trí đó. Nếu kô tìm được thì trả về -1.



Hàm Tìm Kiếm (dùng while)

```
int TimKiem(int a[], int n, int x)
{
        int vt = 0;

        while (vt < n && a[vt] != x)
                vt++;

        if (vt < n)
                return vt;
        else
                return -1;
}
```

Hàm Tìm Kiếm (dùng for)

```
int TimKiem(int a[], int n, int x)
{
        for (int vt = 0; vt < n; vt++)
                if (a[vt] == x)
                        return vt;

        return -1;
}
```

Kiểm tra tính chất của mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Mảng a có phải là mảng toàn các số nguyên tố hay không?
- Ý tưởng
  - Cách 1: Đếm số lượng số ngtố của mảng. Nếu số lượng này bằng đúng n thì mảng toàn ngtố.
  - Cách 2: Đếm số lượng số không phải ngtố của mảng. Nếu số lượng này bằng 0 thì mảng toàn ngtố.
  - Cách 3: Tìm xem có phần tử nào không phải số ngtố không. Nếu có thì mảng không toàn số ngtố.

Hàm Kiểm Tra (Cách 1)

```
int KiemTra_C1(int a[], int n)
{
        int dem = 0;

        for (int i = 0; i < n; i++)
              if (LaSNT(a[i]) == 1) // có thể bỏ == 1
                    dem++;

        if (dem == n)
              return 1;
        return 0;
}
```

Hàm Kiểm Tra (Cách 2)

```
int KiemTra_C2(int a[], int n)
{
        int dem = 0;

        for (int i = 0; i < n; i++)
              if (LaSNT(a[i]) == 0) // Có thể sử dụng !
                    dem++;

        if (dem == 0)
              return 1;
        return 0;
}
```

Hàm Kiểm Tra (Cách 3)

```
int KiemTra_C3(int a[], int n)
{
        for (int i = 0; i < n ; i++)
              if (LaSNT(a[i]) == 0)
                    return 0;

        return 1;
}
```

Tách các phần tử thỏa điều kiện

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na. Tách các số nguyên tố có trong mảng a vào mảng b.
- Ý tưởng
  - Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b.

Hàm Tách Số Nguyên Tố

```
void TachSNT(int a[], int na, int b[], int &nb)
{
        nb = 0;

        for (int i = 0; i < na; i++)
                if (LaSNT(a[i]) == 1)
                {
                        b[nb] = a[i];
                        nb++;
                }
}
```

Tách mảng thành 2 mảng con

Yêu cầu

- Cho trước mảng a, số lượng phần tử na. Tách mảng a thành 2 mảng b (chứa số nguyên tố) và mảng c (các số còn lại).

Ý tưởng

- Cách 1: viết 1 hàm tách các số nguyên tố từ mảng a sang mảng b và 1 hàm tách các số không phải nguyên tố từ mảng a sang mảng c.
- Cách 2: Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b, ngược lại đưa vào mảng c.

Hàm Tách 2 Mảng

```
void TachSNT2(int a[], int na,
              int b[], int &nb, int c[], int &nc)
{
      nb = 0;
      nc = 0;

      for (int i = 0; i < na; i++)
            if (LaSNT(a[i]) == 1)
            {
                  b[nb] = a[i]; nb++;
            }
            else
            {
                  c[nc] = a[i]; nc++;
            }
}
```

Gộp 2 mảng thành một mảng

Yêu cầu

- Cho trước mảng a, số lượng phần tử na và mảng b số lượng phần tử nb. Gộp 2 mảng trên theo tứ tự đó thành mảng c, số lượng phần tử nc.

Ý tưởng

- Chuyển các phần tử của mảng a sang mảng c

  => nc = na
- Tiếp tục đưa các phần tử của mảng b sang mảng c

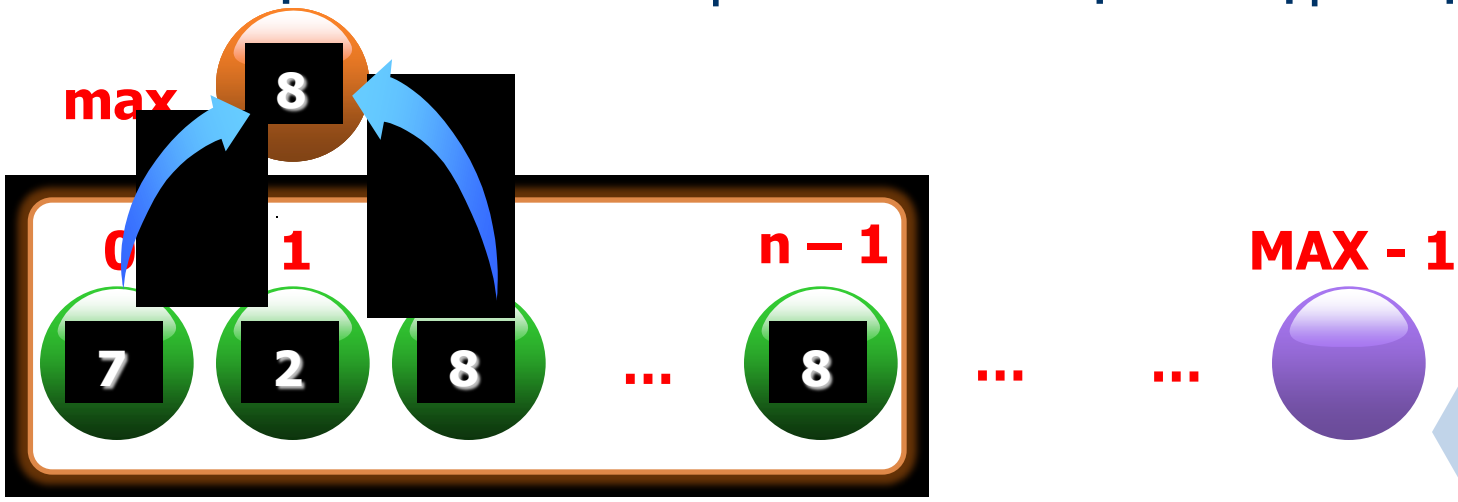
  => nc = nc + nb

Hàm Gộp Mảng

```
void GopMang(int a[], int na, int b[], int nb,
             int c[], int &nc)
{
      nc = 0;

      for (int i = 0; i < na; i++)
      {
            c[nc] = a[i]; nc++; // c[nc++] = a[i];
      }

      for (int i = 0; i < nb; i++)
      {
            c[nc] = b[i]; nc++; // c[nc++] = b[i];
      }
}
```

Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a (gọi là max)
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]
  - Lần lượt kiểm tra các phần tử còn lại để cập nhật max.



Hàm tìm Max

```
int TimMax(int a[], int n)
{
        int max = a[0];

        for (int i = 1; i < n; i++)
              if (a[i] > max)
                    max = a[i];

        return max;
}
```

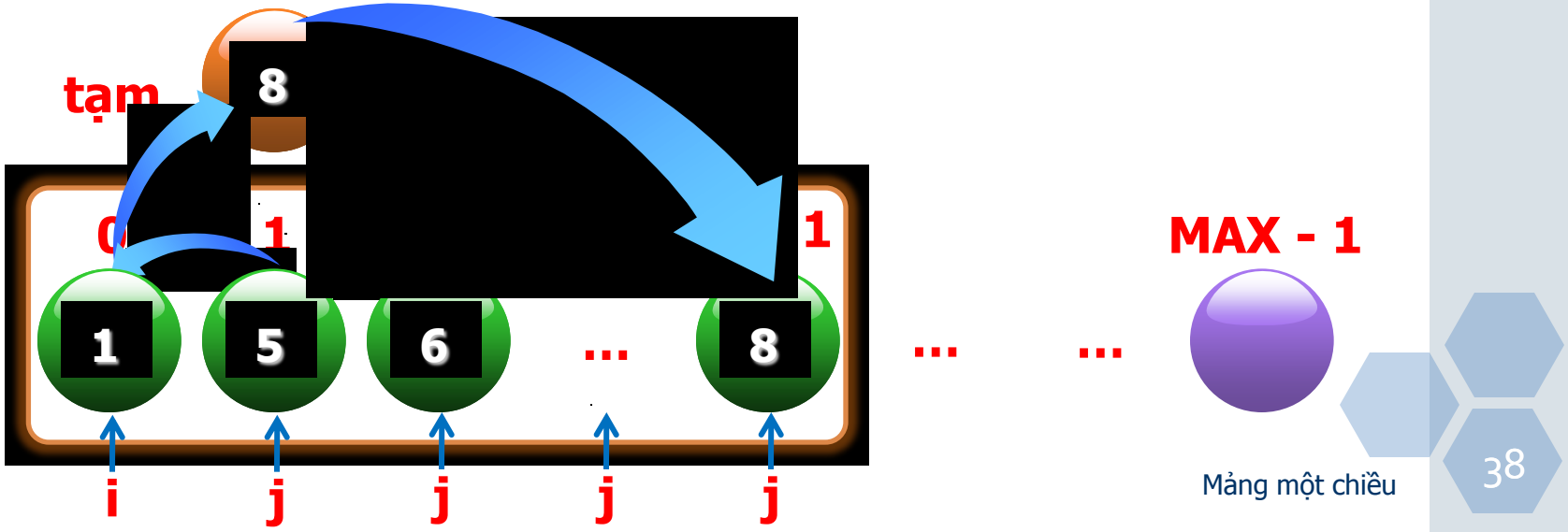
Sắp xếp mảng thành tăng dần

Yêu cầu

- Cho trước mảng a kích thước n. Hãy sắp xếp mảng a đó sao cho các phần tử có giá trị tăng dần.

Ý tưởng

- Sử dụng 2 biến i và j để so sánh tất cả cặp phần tử với nhau và hoán vị các cặp nghịch thế (sai thứ tự).

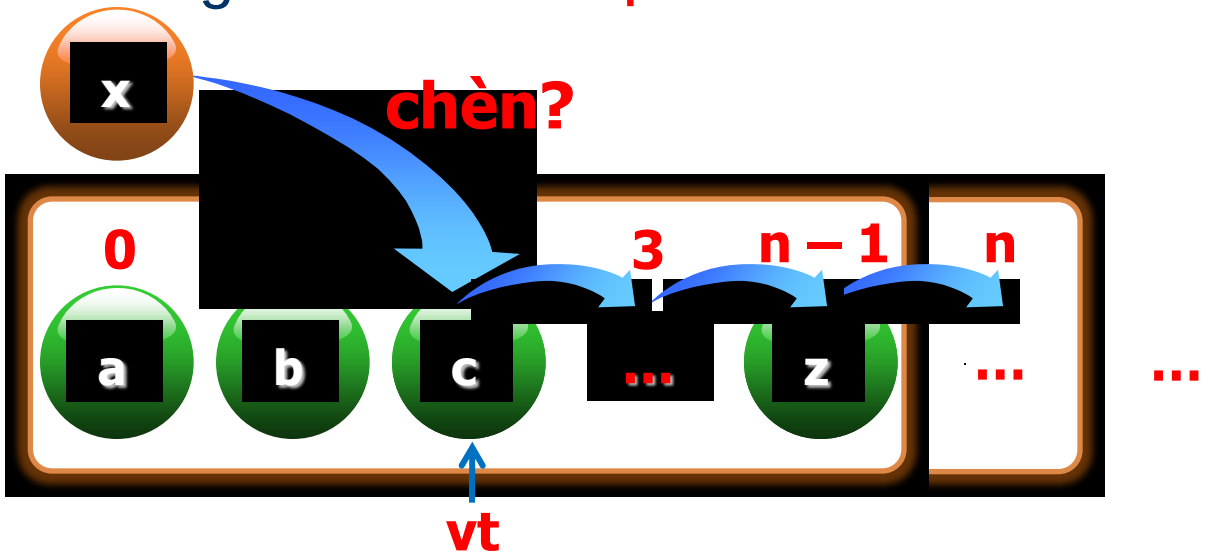


Hàm Sắp Xếp Tăng

```
void SapXepTang(int a[], int n)
{
	int i, j;

	for (i = 0; i < n - 1; i++)
	{
		for (j = i + 1; j < n; j++)
		{
			if (a[i] > a[j])
				HoanVi(a[i], a[j]);
		}
	}
}
```

Thêm một phần tử vào mảng

- Yêu cầu
  - Thêm phần tử x vào mảng a kích thước n tại vị trí vt.
- Ý tưởng
  - "Đẩy" các phần tử bắt đầu tại vị trí vt sang phải 1 vị trí.
  - Đưa x vào vị trí vt trong mảng.
  - Tăng n lên 1 đơn vị.

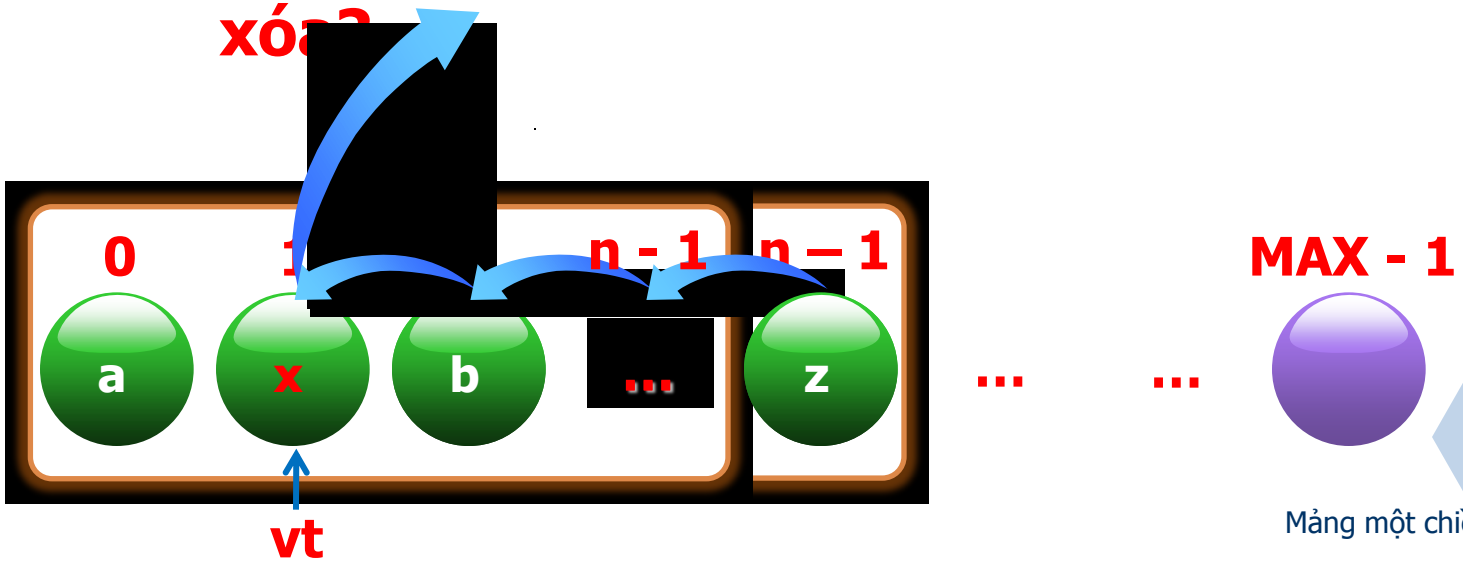


Hàm Thêm

```
void Them(int a[], int &n, int vt, int x)
{
        if (vt >= 0 && vt <= n)
        {
                for (int i = n; i > vt; i--)
                        a[i] = a[i - 1];

                a[vt] = x;
                n++;
        }
}
```

Xóa một phần tử trong mảng

- Yêu cầu
  - Xóa một phần tử trong mảng a kích thước n tại vị trí vt
- Ý tưởng
  - "Kéo" các phần tử bên phải vị trí vt sang trái 1 vị trí.
  - Giảm n xuống 1 đơn vị.



Hàm Xóa

```
void Xoa(int a[], int &n, int vt)
{
        if (vt >= 0 && vt < n)
        {
                for (int i = vt; i < n - 1; i++)
                        a[i] = a[i + 1];

                n--;
        }
}
```

Bài tập

1. Các thao tác nhập xuất
   a. Nhập mảng
   b. Xuất mảng

2. Các thao tác kiểm tra
   a. Mảng có phải là mảng toàn chẵn
   b. Mảng có phải là mảng toàn số nguyên
   c. Mảng có phải là mảng tăng dần

Bài tập

3. Các thao tác tính toán
   - a. Có bao nhiêu số chia hết cho 4 nhưng không chia hết cho 5
   - b. Tổng các số nguyên tố có trong mảng
4. Các thao tác tìm kiếm
   - a. Vị trí cuối cùng của phần tử x trong mả
   - b. Vị trí số nguyên tố đầu tiên trong mảng
   - c. Tìm số nhỏ nhất trong mảng
   - d. Tìm số dương nhỏ nhất trong mảng

Bài tập

5. Các thao tác xử lý

a. Tách các số nguyên tố có trong mảng a đưa vào mảng b.

b. Tách mảng a thành 2 mảng b (chứa các số nguyên dương) và c (chứa các số còn lại)

c. Sắp xếp mảng giảm dần

d. Sắp xếp mảng sao cho các số dương đầu mảng giảm dần, kế đến là các số tăng dần, cuối cùng là các số 0.

Bài tập

6. Các thao tác thêm/xóa/sửa

a. Sửa các số nguyên tố có trong mảng thành số 0

b. Chèn số 0 đằng sau các số nguyên tố trong mảng

c. Xóa tất cả số nguyên tố có trong mảng

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

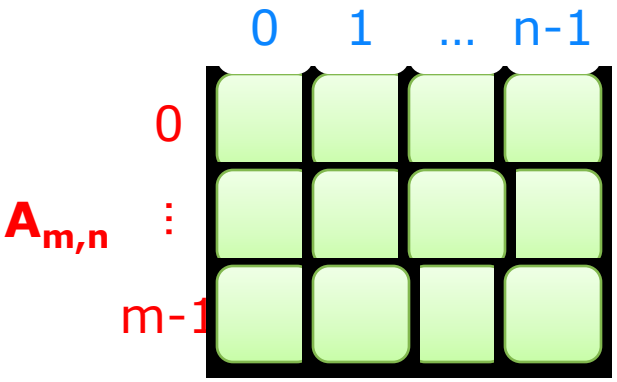
Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

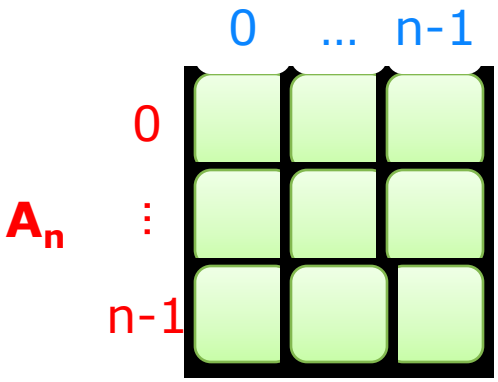
MẢNG HAI CHIỀU

Nội dung

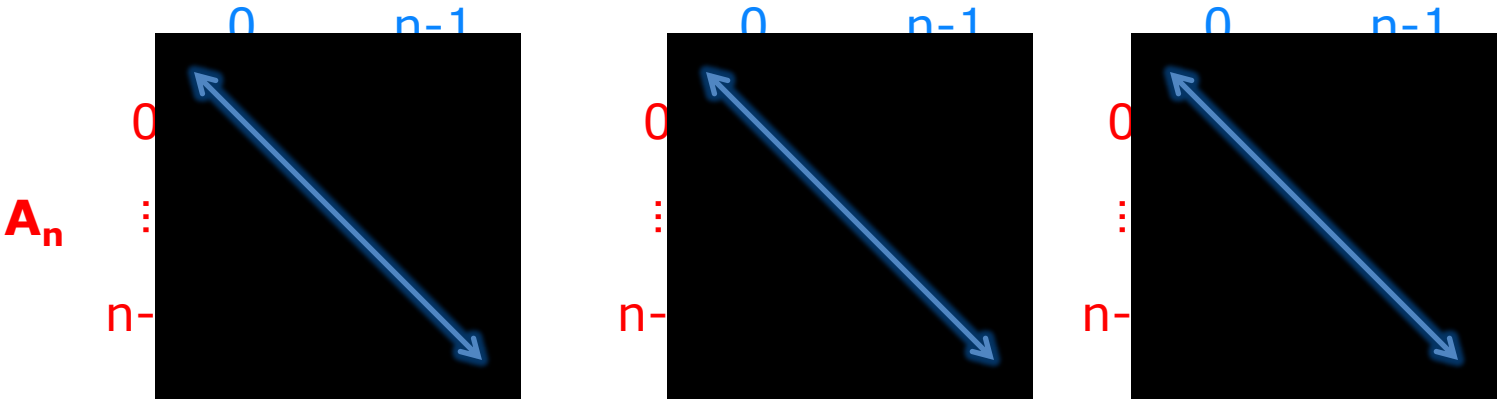
1. Khái niệm
2. Khai báo
3. Truy xuất dữ liệu kiểu mảng
4. Một số bài toán trên mảng 2 chiều

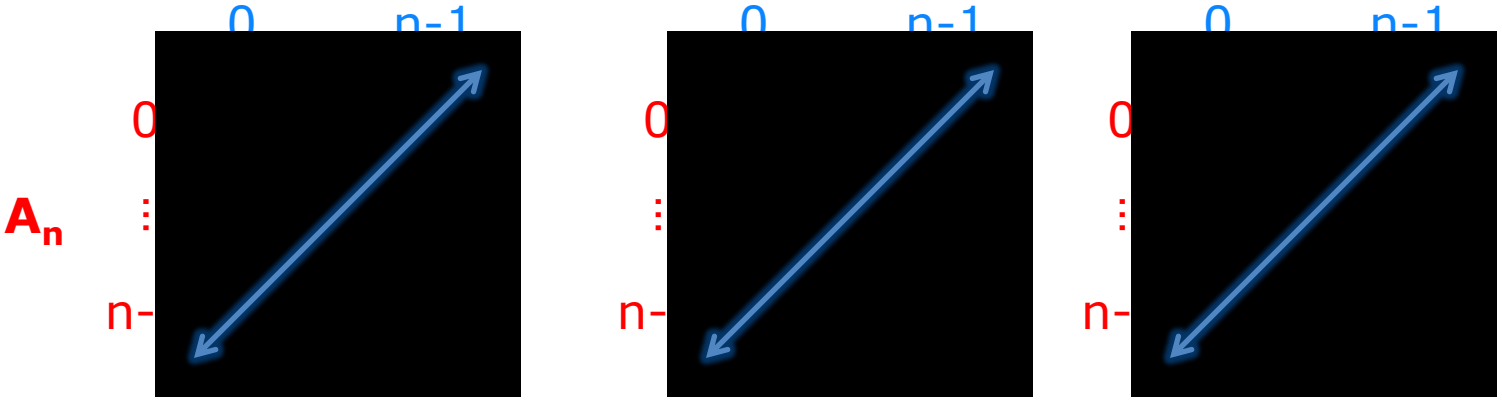
Ma Trận





Ma Trận


A_n
0
n-1
0
n-1
dòng = cột
dòng > cột
dòng < cột


A_n
0
n-1
0
n-1
dòng + cột = n-1
dòng + cột > n-1
dòng + cột < n-1

Khai báo kiểu mảng 2 chiều

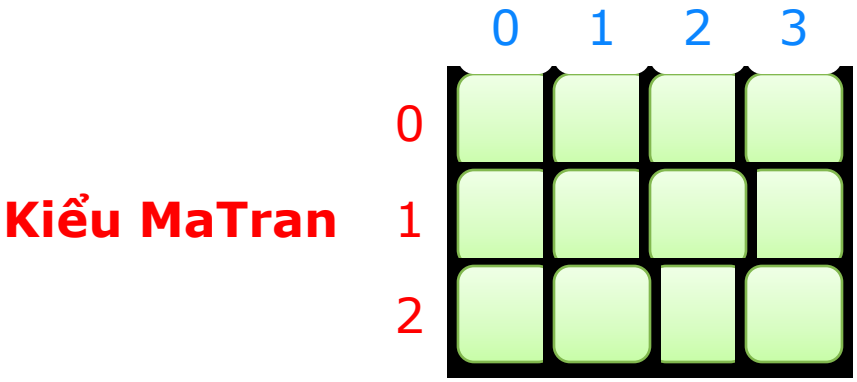
Cú pháp

```
typedef <kiểu cơ sở> <tên kiểu>[<N1>][<N2>];
```

- N1, N2: số lượng phần tử mỗi chiều

Ví dụ

```
typedef int MaTran[3][4];
```



Khai báo biến mảng 2 chiều

Cú pháp

- Tường minh

```
<kiểu cơ sở> <tên biến>[<N1>][<N2>];
```

- Không tường minh (thông qua kiểu)

```
typedef <kiểu cơ sở> <tên kiểu>[<N1>][<N2>];

<tên kiểu> <tên biến>;
<tên kiểu> <tên biến 1>, <tên biến 2>;
```

Khai báo biến mảng 2 chiều

- Ví dụ
  - Tường minh

```
int a[10][20], b[10][20];
int c[5][10];
int d[10][20];
```

  - Không tường minh (thông qua kiểu)

```
typedef int MaTran10x20[10][20];
typedef int MaTran5x10[5][10];

MaTran10x20 a, b;
MaTran11x11 c;
MaTran10x20 d;
```

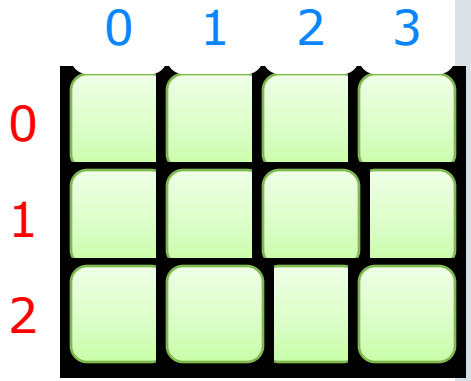
Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<giá trị cs1>][<giá trị cs2>]
```

- Ví dụ
  - Cho mảng 2 chiều như sau

```
int a[3][4];
```



  - Các truy xuất
    - Hợp lệ: a[0][0], a[0][1], …, a[2][2], a[2][3]
    - Không hợp lệ: a[-1][0], a[2][4], a[3][3]

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<giá trị cs1>][giá trị cs2] =
                                        <giá trị>;
```

- Ví dụ

```
int a[5][10], b[5][10];

b = a;          // Sai
int i, j;
for (i = 0; i < 5; i++)
      for (j = 0; j < 10; j++)
            b[i][j] = a[i][j];
```

Truyền mảng cho hàm

Truyền mảng cho hàm

- Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void NhapMaTran(int a[50][100]);
```

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
  - Có thể bỏ số lượng phần tử chiều thứ 2 hoặc con trỏ.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void NhapMaTran(int a[][100]);
void NhapMaTran(int (*a)[100]);
```

Truyền mảng cho hàm

- Truyền mảng cho hàm
  - Số lượng phần tử thực sự truyền qua biến khác

```
void XuatMaTran(int a[50][100], int m, int n);
void XuatMaTran(int a[][100], int m, int n);
void XuatMaTran(int (*a)[100], int m, int n);
```

- Lời gọi hàm

```
void NhapMaTran(int a[][100], int &m, int &n);
void XuatMaTran(int a[][100], int m, int n);
void main()
{
        int a[50][100], m, n;
        NhapMaTran(a, m, n);
        XuatMaTran(a, m, n);
}
```

Một số bài toán cơ bản

- Viết chương trình con thực hiện các yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Tìm kiếm một phần tử trong mảng
  - Kiểm tra tính chất của mảng
  - Tính tổng các phần tử trên dòng/cột/toàn ma trận/đường chéo chính/nửa trên/nửa dưới
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - …

Một số quy ước

- Kiểu dữ liệu

```
#define MAXD 50
#define MAXC 100
```

- Các chương trình con
  - Hàm void HoanVi(int x, int y): hoán vị giá trị của hai số nguyên.
  - Hàm int LaSNT(int n): kiểm tra một số có phải là số nguyên tố. Trả về 1 nếu n là số nguyên tố, ngược lại trả về 0.

Thủ tục HoanVi & Hàm LaSNT

```
void HoanVi(int &x, int &y)
{
	int tam = x; x = y; y = tam;
}

int LaSNT(int n)
{
	int i, dem = 0;
	for (i = 1; i <= n; i++)
		if (n%i == 0)
			dem++;

	if (dem == 2)
		return 1;
	else return 0;
}
```

Nhập Ma Trận

- Yêu cầu
  - Cho phép nhập mảng a, m dòng, n cột
- Ý tưởng
  - Cho trước một mảng 2 chiều có dòng tối đa là MAXD, số cột tối đa là MAXC.
  - Nhập số lượng phần tử thực sự m, n của mỗi chiều.
  - Nhập từng phần tử từ [0][0] đến [m-1][n-1].

Hàm Nhập Ma Trận

```
void NhapMaTran(int a[][MAXC], int &m, int &n)
{
        printf("Nhap so dong, so cot cua ma tran: ");
        scanf("%d%d", &m, &n);

        int i, j;
        for (i=0; i<m; i++)
              for (j=0; j<n; j++)
              {
                    printf("Nhap a[%d][%d]: ", i, j);
                    scanf("%d", &a[i][j]);
              }
}
```

Xuất Ma Trận

- Yêu cầu
  - Cho phép nhập mảng a, m dòng, n cột
- Ý tưởng
  - Xuất giá trị từng phần tử của mảng 2 chiều từ dòng có 0 đến dòng m-1, mỗi dòng xuất giá giá trị của cột 0 đến cột n-1 trên dòng đó.

Hàm Xuất Ma Trận

```
void XuatMaTran(int a[][MAXC], int m, int n)
{
        int i, j;
        for (i=0; i<m; i++)
        {
                for (j=0; j<n; j++)
                        printf("%d ", a[i][j]);

                printf("\n");
        }
}
```

Tìm kiếm một phần tử trong Ma Trận

- Yêu cầu
  - Tìm xem phần tử x có nằm trong ma trận a kích thước mxn hay không?
- Ý tưởng
  - Duyệt từng phần của ma trận a. Nếu phần tử đang xét bằng x thì trả về có (1), ngược lại trả về không có (0).

Hàm Tìm Kiếm

```
int TimKiem(int a[][MAXC], int m, int n, int x)
{
      int i, j;
      for (i=0; i<m; i++)
            for (j=0; j<n; j++)
                  if (a[i][j] == x)
                        return 1;
      return 0;
}
```

Kiểm tra tính chất của mảng

- Yêu cầu
  - Cho trước ma trận a kích thước mxn. Ma trận a có phải là ma trậntoàn các số nguyên tố hay không?
- Ý tưởng
  - Cách 1: Đếm số lượng số ngtố của ma trận. Nếu số lượng này bằng đúng mxn thì ma trận toàn ngtố.
  - Cách 2: Đếm số lượng số không phải ngtố của ma trận. Nếu số lượng này bằng 0 thì ma trận toàn ngtố.
  - Cách 3: Tìm xem có phần tử nào không phải số ngtố không. Nếu có thì ma trận không toàn số ngtố.

Hàm Kiểm Tra (Cách 1)

```
int KiemTra_C1(int a[][MAXC], int m, int n)
{
      int i, j, dem = 0;

      for (i=0; i<m; i++)
            for (j=0; j<n; j++)
                  if (LaSNT(a[i][j]==1)
                        dem++;

      if (dem == m*n)
            return 1;
      return 0;
}
```

Hàm Kiểm Tra (Cách 2)

```
int KiemTra_C2(int a[][MAXC], int m, int n)
{
	int i, j, dem = 0;

	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
			if (LaSNT(a[i][j]==0)
				dem++;

	if (dem == 0)
		return 1;
	return 0;
}
```

Hàm Kiểm Tra (Cách 2)

```
int KiemTra_C3(int a[][MAXC], int m, int n)
{
        int i, j, dem = 0;

        for (i=0; i<m; i++)
              for (j=0; j<n; j++)
                    if (LaSNT(a[i][j]==0)
                          return 0;

        return 1;
}
```

Tính tổng các phần tử

Yêu cầu

- Cho trước ma trận a, kích thước mxn. Tính tổng các phần tử trên:
  - Dòng d, cột c
  - Đường chéo chính, đường chéo phụ (ma trận vuông)
  - Nửa trên/dưới đường chéo chính (ma trận vuông)
  - Nửa trên/dưới đường chéo phụ (ma trận vuông)

Ý tưởng

- Duyệt ma trận và cộng dồn các phần tử có tọa độ (dòng, cột) thỏa yêu cầu.

Hàm tính tổng trên dòng

```
int TongDong(int a[][MAXC], int m, int n, int d)
{
        int j, tong;

        tong = 0;

        for (j=0; j<n; j++)      // Duyệt các cột
              tong = tong + a[d][j];

        return tong;
}
```

Hàm tính tổng trên cột

```
int TongCot(int a[][MAXC], int m, int c)
{
        int i, tong;

        tong = 0;

        for (i=0; i<m; i++)      // Duyệt các dòng
              tong = tong + a[i][c];

        return tong;
}
```

Hàm tính tổng đường chéo chính

```
int TongDCChinh(int a[][MAXC], int n)
{
        int i, tong;

        tong = 0;

        for (i=0; i<n; i++)
                tong = tong + a[i][i];

        return tong;
}
```

Hàm tính tổng trên đường chéo chính

```
int TongTrenDCChinh(int a[][MAXC], int n)
{
        int i, j, tong;

        tong = 0;

        for (i=0; i<n; i++)
              for (j=0; j<n; j++)
                    if (i < j)
                          tong = tong + a[i][j];

        return tong;
}
```

Hàm tính tổng dưới đường chéo chính

```
int TongTrenDCChinh(int a[][MAXC], int n)
{
        int i, j, tong;

        tong = 0;

        for (i=0; i<n; i++)
              for (j=0; j<n; j++)
                    if (i > j)
                          tong = tong + a[i][j];

        return tong;
}
```

Hàm tính tổng trên đường chéo phụ

```
int TongDCPhu(int a[][MAXC], int n)
{
        int i, tong;

        tong = 0;

        for (i=0; i<n; i++)
                tong = tong + a[i][n-i-1];

        return tong;
}
```

Tìm giá trị lớn nhất của Ma Trận

- Yêu cầu
  - Cho trước ma trận a, kích thước mxn. Tìm giá trị lớn nhất trong ma trận a (gọi là max)
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0][0]
  - Lần lượt kiểm tra các phần tử còn lại để cập nhật max.

Hàm tìm Max

```
int TimMax(int a[][MAXC], int m, int n)
{
        int i, j, max;

        max = a[0][0];

        for (i=0; i<m; i++)
              for (j=0; j<n; j++)
                    if (a[i][j] > max)
                          max = a[i][j];

        return max;
}
```

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DỮ LIỆU KIỂU CON TRỎ (CƠ BẢN)

Nội dung

1. Khái niệm và cách sử dụng
2. Các cách truyền đối số cho hàm
3. Con trỏ và mảng một chiều
4. Con trỏ và cấu trúc

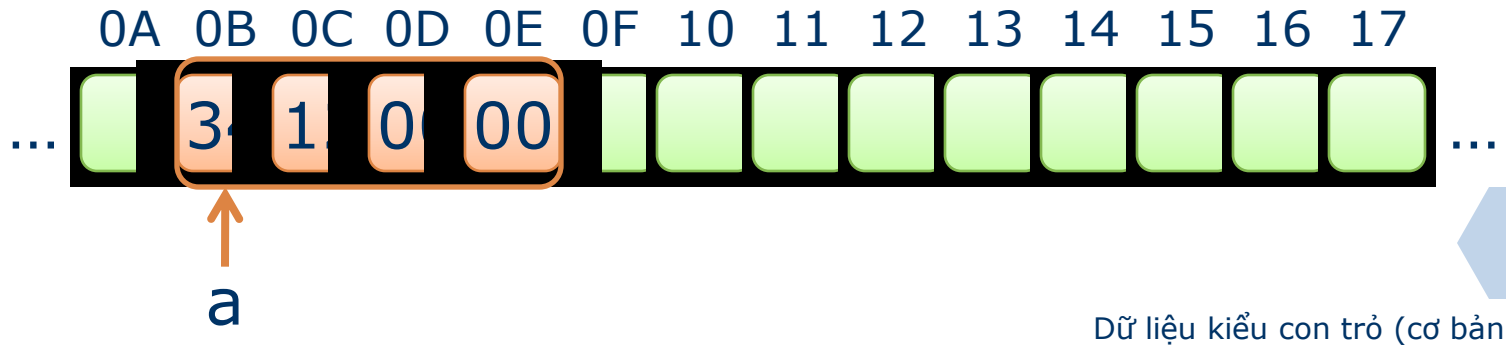
Kiến trúc máy tính

Bộ nhớ máy tính

- Bộ nhớ RAM chứa rất nhiều ô nhớ, mỗi ô nhớ có kích thước 1 byte.
- RAM dùng để chứa một phần hệ điều hành, các lệnh chương trình, các dữ liệu…
- Mỗi ô nhớ có địa chỉ duy nhất và địa chỉ này được đánh số từ 0 trở đi.
- Ví dụ
  - RAM 512MB được đánh địa chỉ từ 0 đến $2^{29} - 1$
  - RAM 2GB được đánh địa chỉ từ 0 đến $2^{31} - 1$

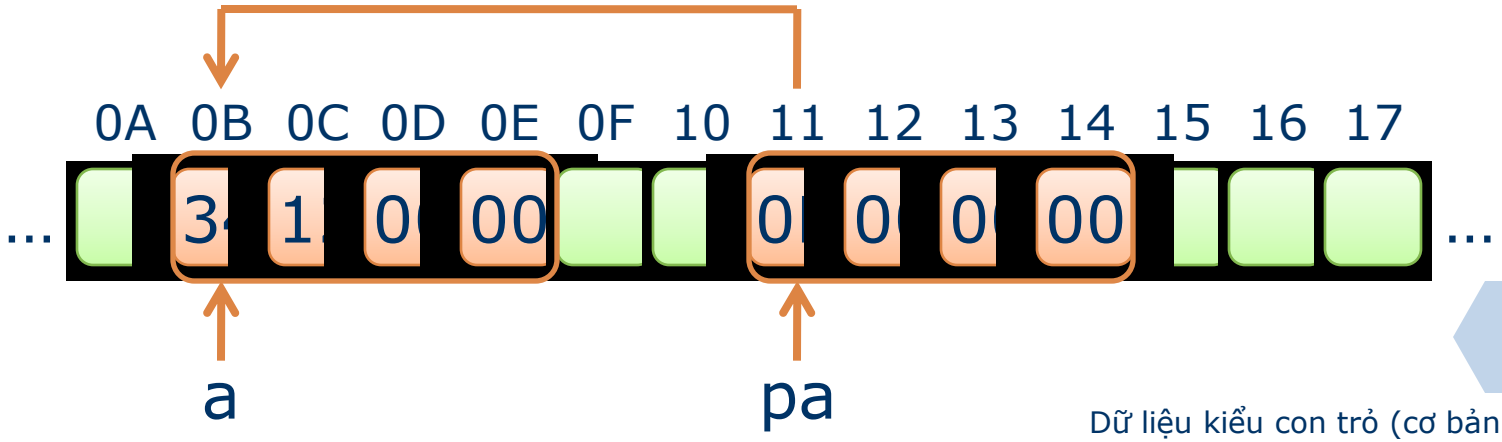
Khai báo biến trong C

- Quy trình xử lý của trình biên dịch
  - Dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến đó.
  - Liên kết địa chỉ ô nhớ đó với tên biến.
  - Khi gọi tên biến, nó sẽ truy xuất tự động đến ô nhớ đã liên kết với tên biến.
- Ví dụ: int a = 0x1234; // Giả sử địa chỉ 0x0B



Khái niệm con trỏ

- Khái niệm
  - Địa chỉ của biến là một con số.
  - Ta có thể tạo biến khác để lưu địa chỉ của biến này ➔ Con trỏ.



Khai báo con trỏ

- Khai báo
  - Giống như mọi biến khác, biến con trỏ muốn sử dụng cũng cần phải được khai báo

```
<kiểu dữ liệu> *<tên biến con trỏ>;
```

- Ví dụ

```
char *ch1, *ch2;
int *p1, p2;
```

  - ch1 và ch2 là biến con trỏ, trỏ tới vùng nhớ kiểu char (1 byte).
  - p1 là biến con trỏ, trỏ tới vùng nhớ kiểu int (4 bytes) còn p2 là biến kiểu int bình thường.

Khai báo con trỏ

- Sử dụng từ khóa typedef

```
typedef <kiểu dữ liệu> *<tên kiểu con trỏ>;

<tên kiểu con trỏ> <tên biến con trỏ>;
```

- Ví dụ

```
typedef int *pint;
int *p1;
pint p2, p3;
```

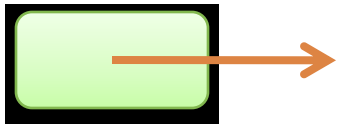
- Lưu ý khi khai báo kiểu dữ liệu mới
  - Giảm bối rối khi mới tiếp xúc với con trỏ.
  - Nhưng dễ nhầm lẫn với biến thường.

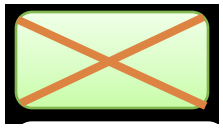
Con trỏ NULL

Khái niệm

- Con trỏ NULL là con trỏ không trỏ và đâu cả.
- Khác với con trỏ chưa được khởi tạo.

```
int n;
int *p1 = &n;
int *p2;     // unreferenced local variable
int *p3 = NULL;
```





NULL

Khởi tạo kiểu con trỏ

- Khởi tạo
  - Khi mới khai báo, biến con trỏ được đặt ở địa chỉ nào đó (không biết trước).
    - ➔ chứa giá trị không xác định
    - ➔ trỏ đến vùng nhớ không biết trước.
  - Đặt địa chỉ của biến vào con trỏ (toán tử &)

```
<tên biến con trỏ> = &<tên biến>;
```

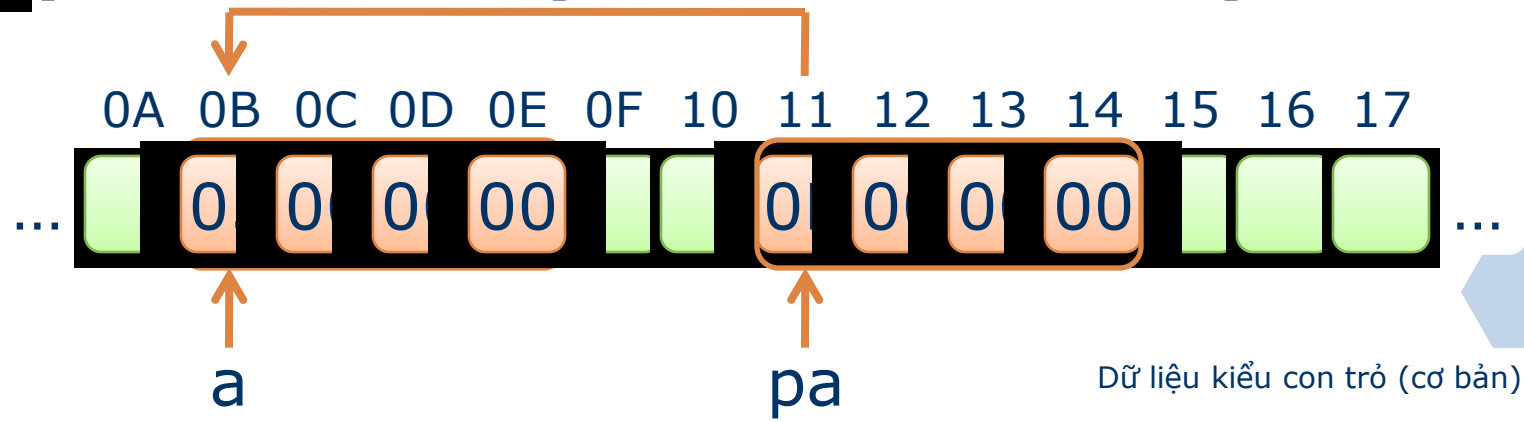
- Ví dụ

```
int a, b;
int *pa = &a, *pb;
pb = &b;
```

Sử dụng con trỏ

- Truy xuất đến ô nhớ mà con trỏ trỏ đến
  - Con trỏ chứa một số nguyên chỉ địa chỉ.
  - Vùng nhớ mà nó trỏ đến, sử dụng toán tử *.
- Ví dụ

```
int a = 5, *pa = &a;
printf("%d\n", pa);  // Giá trị biến pa
printf("%d\n", *pa); // Giá trị vùng nhớ pa trỏ đến
printf("%d\n", &pa); // Địa chỉ biến pa
```



Kích thước của con trỏ

Kích thước của con trỏ

```
char *p1;
int *p2;
float *p3;
double *p4;
…
```

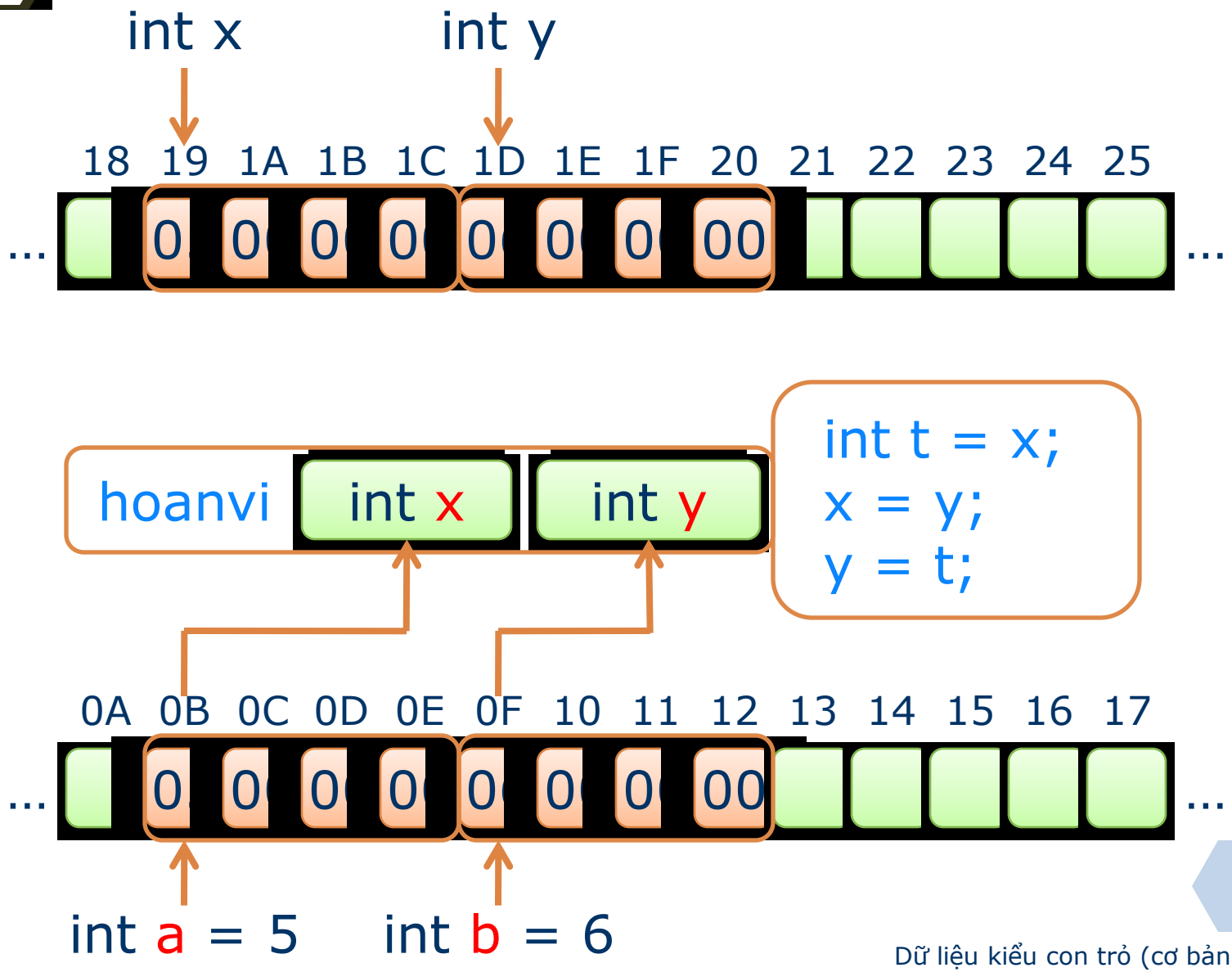
- Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau:
  - Môi trường MD-DOS (16 bit): 2 bytes (64KB)
  - Môi trường Windows (32 bit): 4 bytes (4GB)

Các cách truyền đối số

Truyền giá trị (tham trị)

```
#include <stdio.h>

void hoanvi(int x, int y);

void main()
{
        int a = 5; b = 6;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int x, int y)
{
        int t = x; x = y; y = t;
}
```

Truyền giá trị (tham trị)

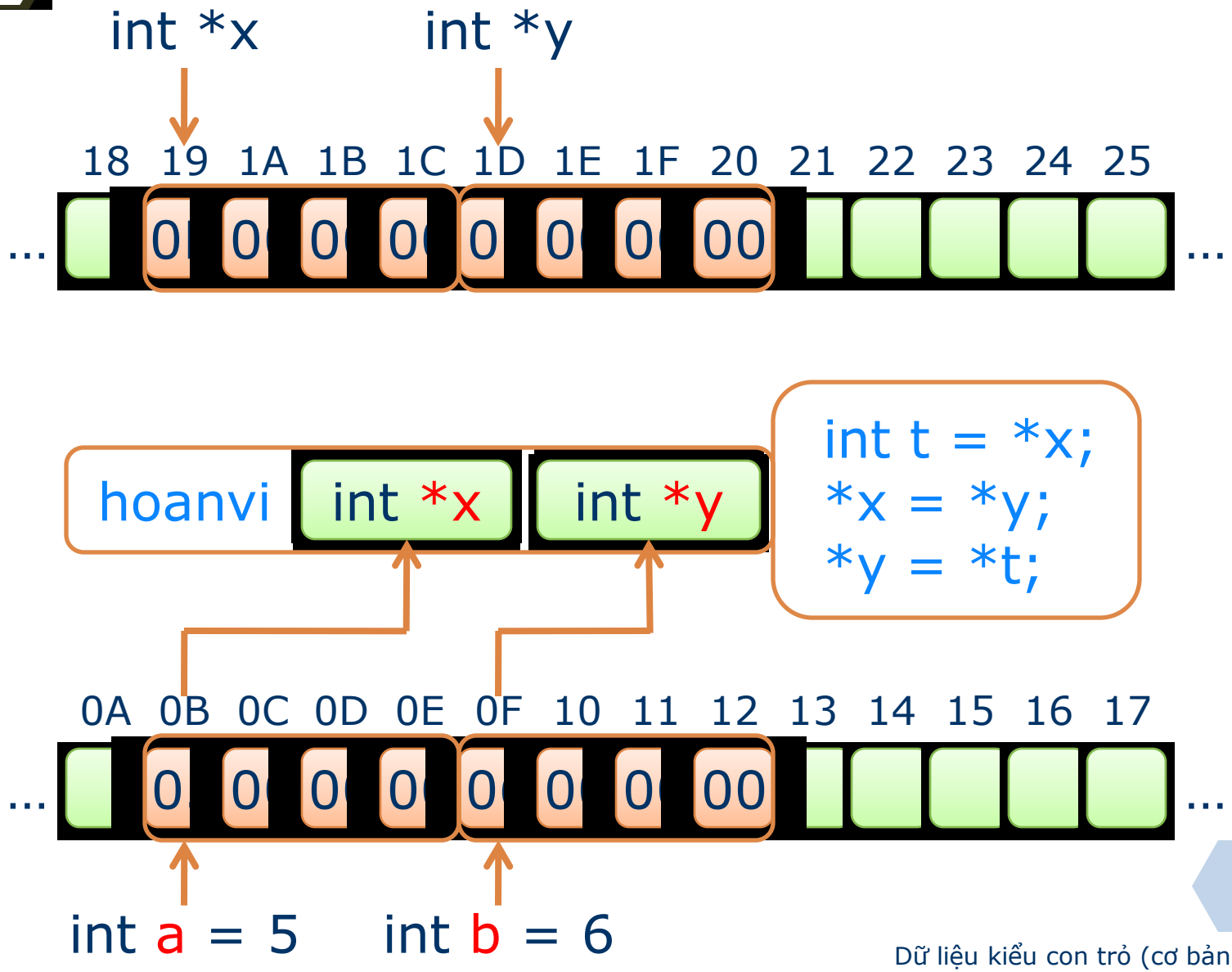


Các cách truyền đối số

Truyền địa chỉ (con trỏ)

```
#include <stdio.h>

void hoanvi(int *x, int *y);

void main()
{
        int a = 2912; b = 1706;
        hoanvi(&a, &b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int *x, int *y)
{
        int t = *x; *x = *y; *y = t;
}
```

Truyền địa chỉ (con trỏ)

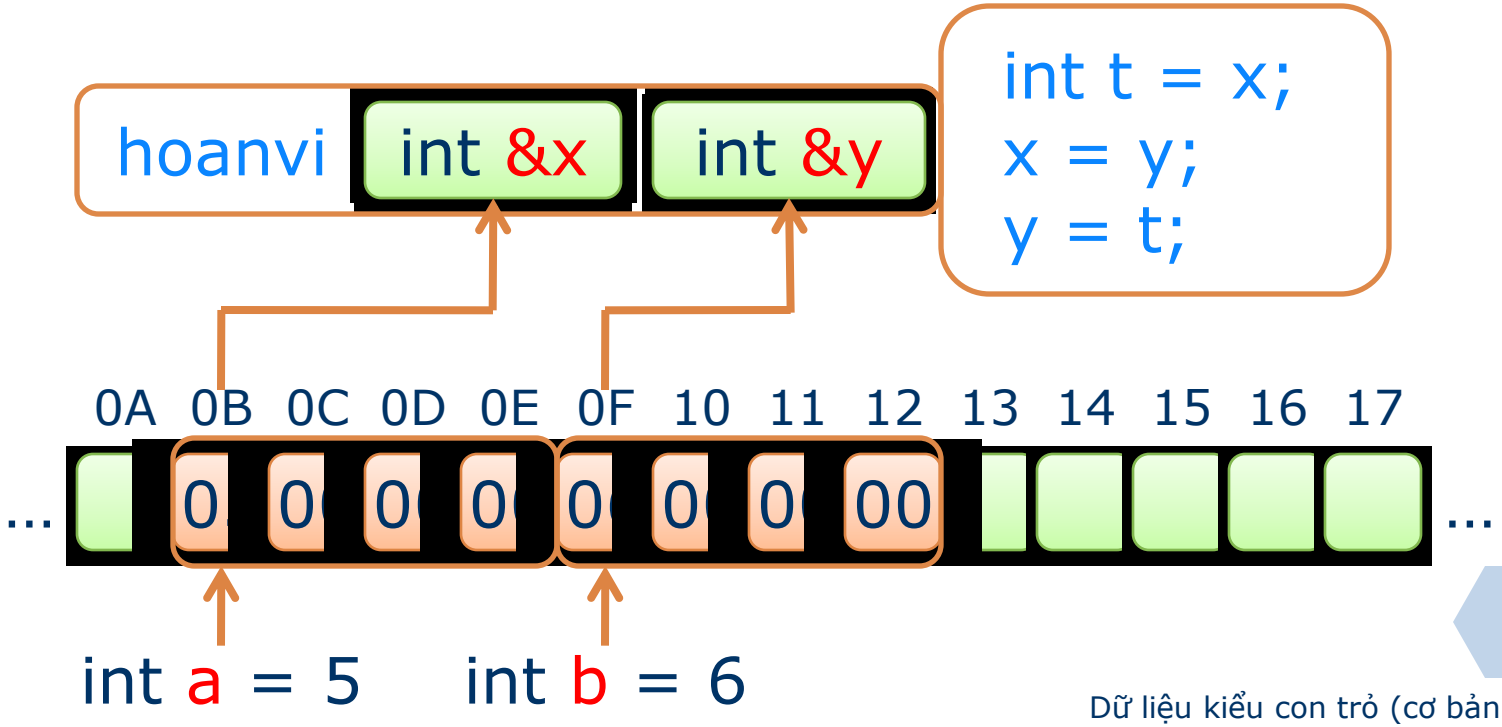


Các cách truyền đối số

- Truyền tham chiếu (C++)

```
#include <stdio.h>

void hoanvi(int &x, int &y);

void main()
{
        int a = 2912; b = 1706;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int &x, int &y)
{
        int t = x; x = y; y = t;
}
```

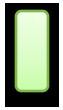
Truyền tham chiếu (C++)



Một số lưu ý

Một số lưu ý

- Con trỏ là khái niệm quan trọng và khó nhất trong C. Mức độ thành thạo C được đánh giá qua mức độ sử dụng con trỏ.
- Nắm rõ quy tắc sau, ví dụ int a, *pa = &a;
  - *pa và a đều chỉ nội dung của biến a.
  - pa và &a đều chỉ địa chỉ của biến a.
- Không nên sử dụng con trỏ khi chưa được khởi tạo. Kết quả sẽ không lường trước được.



```
int *pa; *pa = 1904; // !!!
```

Con trỏ và mảng một chiều

Mảng một chiều

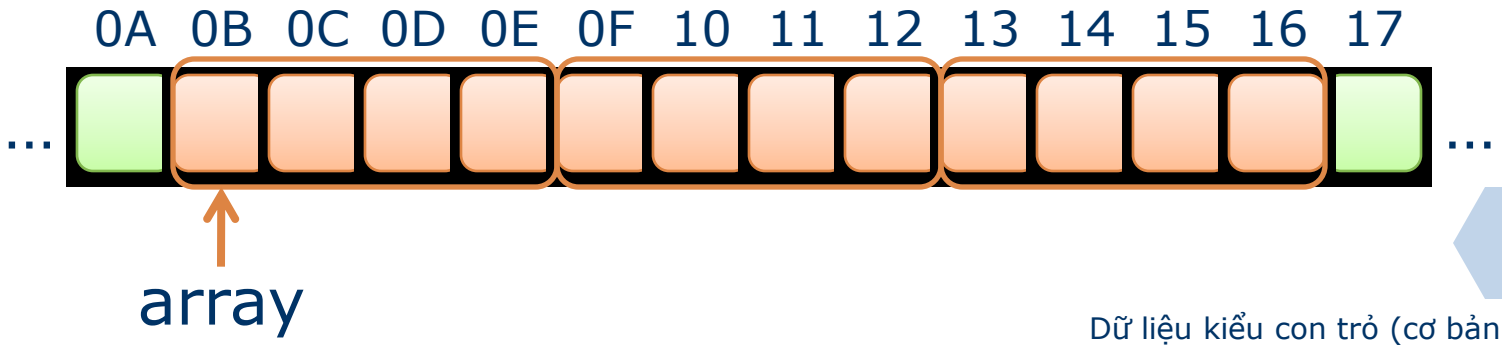
```
int array[3];
```

- Tên mảng array là một hằng con trỏ

  ➔ không thể thay đổi giá trị của hằng này.

- Giá trị của array là địa chỉ phần tử đầu tiên của mảng

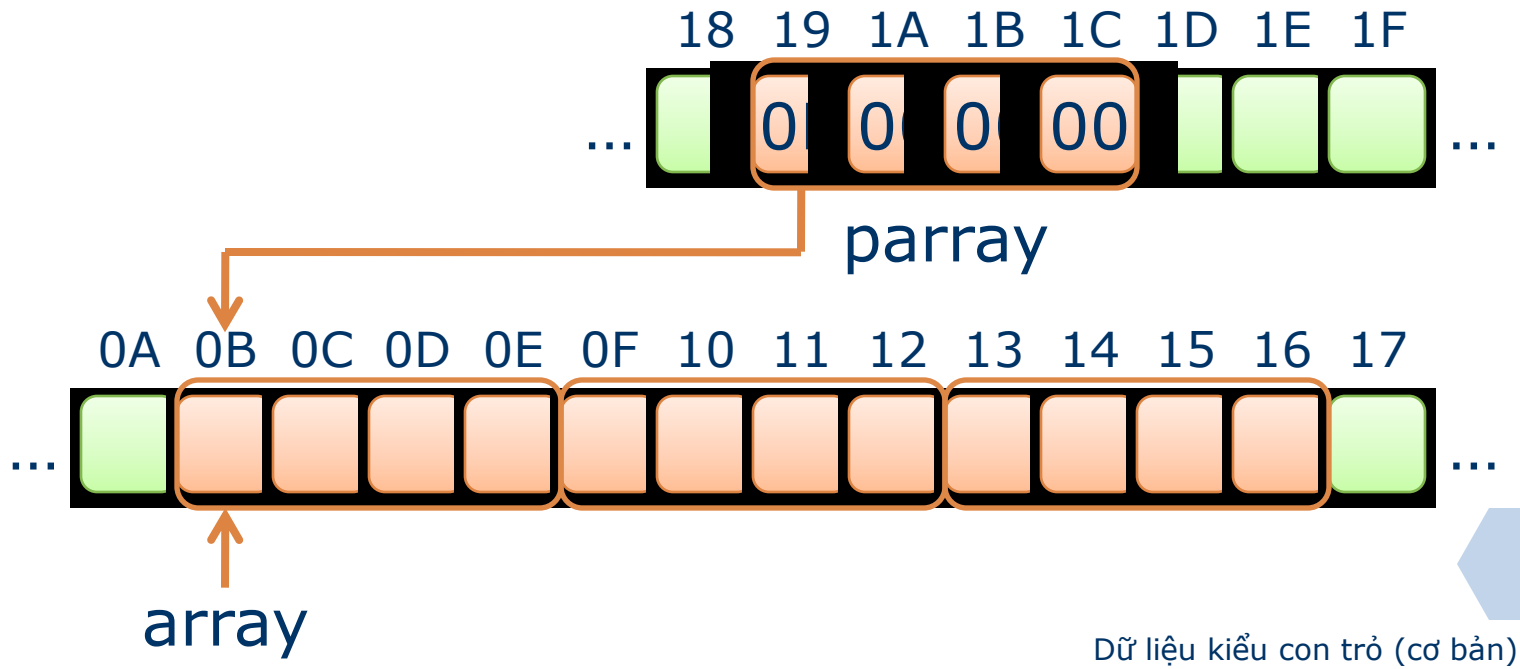
  ➔array == &array[0]



Con trỏ và mảng một chiều

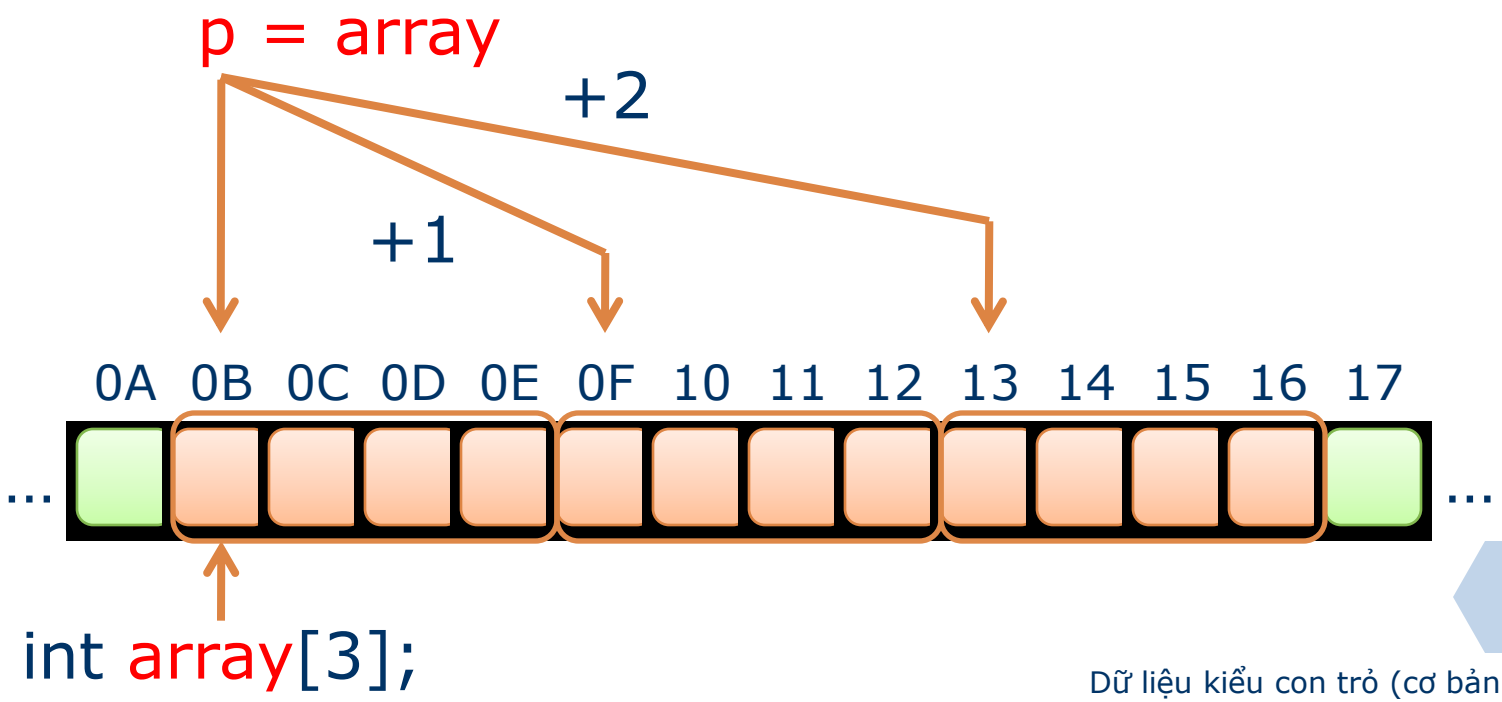
Con trỏ đến mảng một chiều

```
int array[3], *parray;

parray = array;          // Cách 1
parray = &array[0];      // Cách 2
```



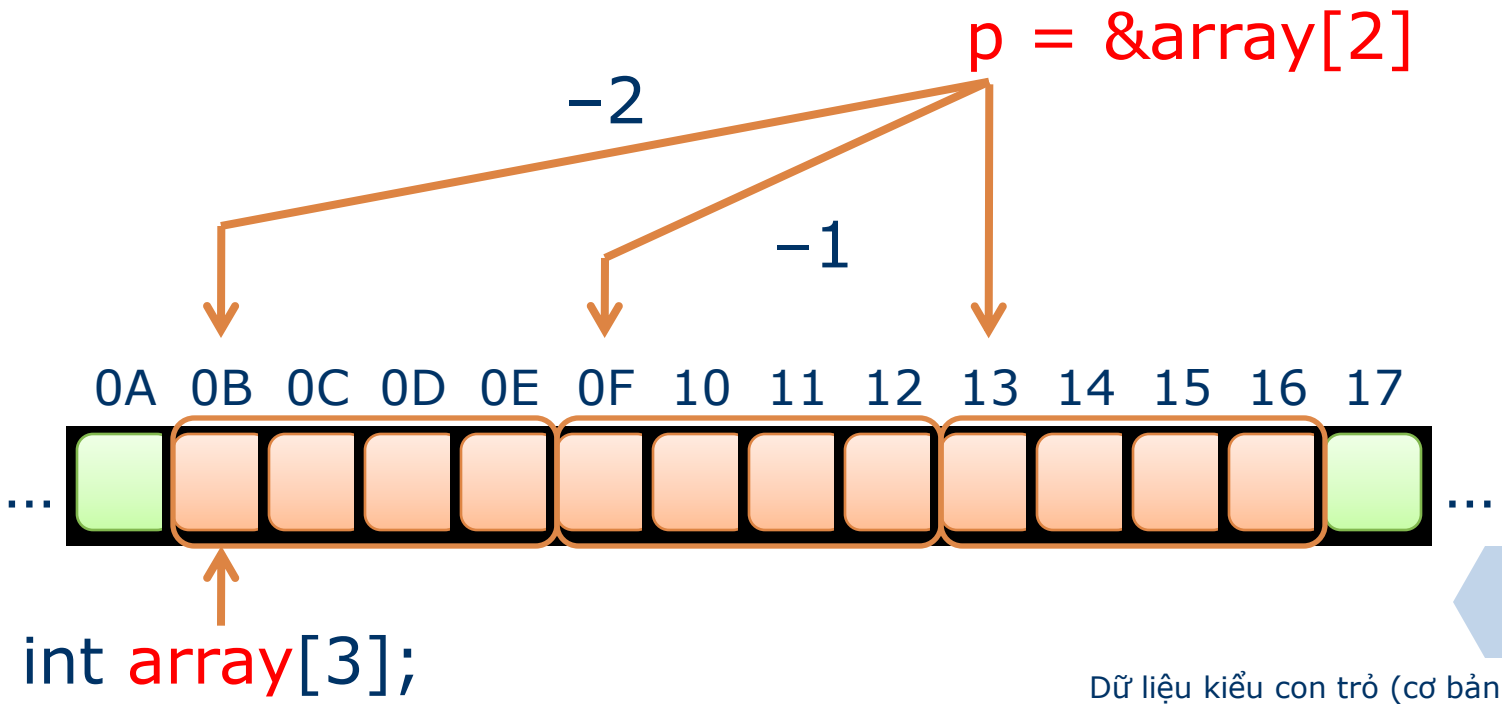
Phép toán số học trên con trỏ

- Phép cộng (tăng)
  - + n ⇔ + n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp += hoặc ++



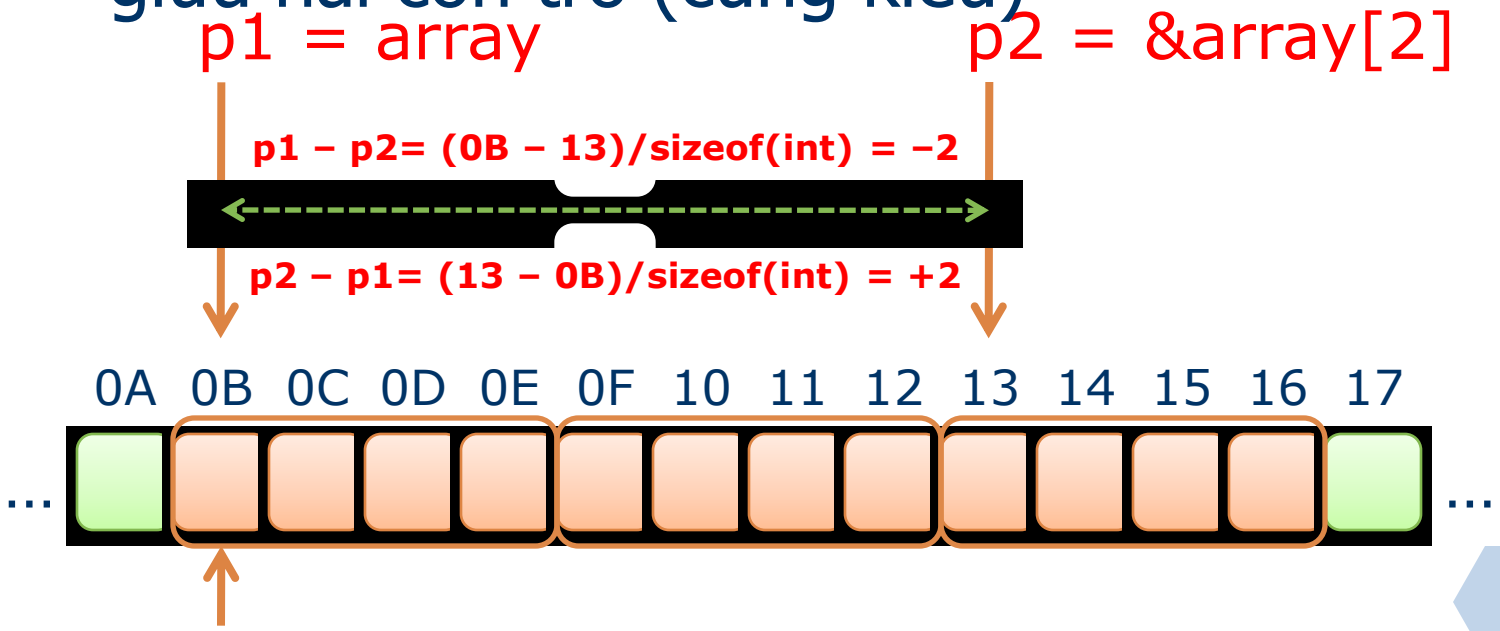
Phép toán số học trên con trỏ

- Phép trừ (giảm)
  - – n ⇔ – n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp –= hoặc – –



Phép toán số học trên con trỏ

- Phép toán tính khoảng cách giữa 2 con trỏ
  - <kiểu dữ liệu> *p1, *p2;
  - p1 – p2 cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)

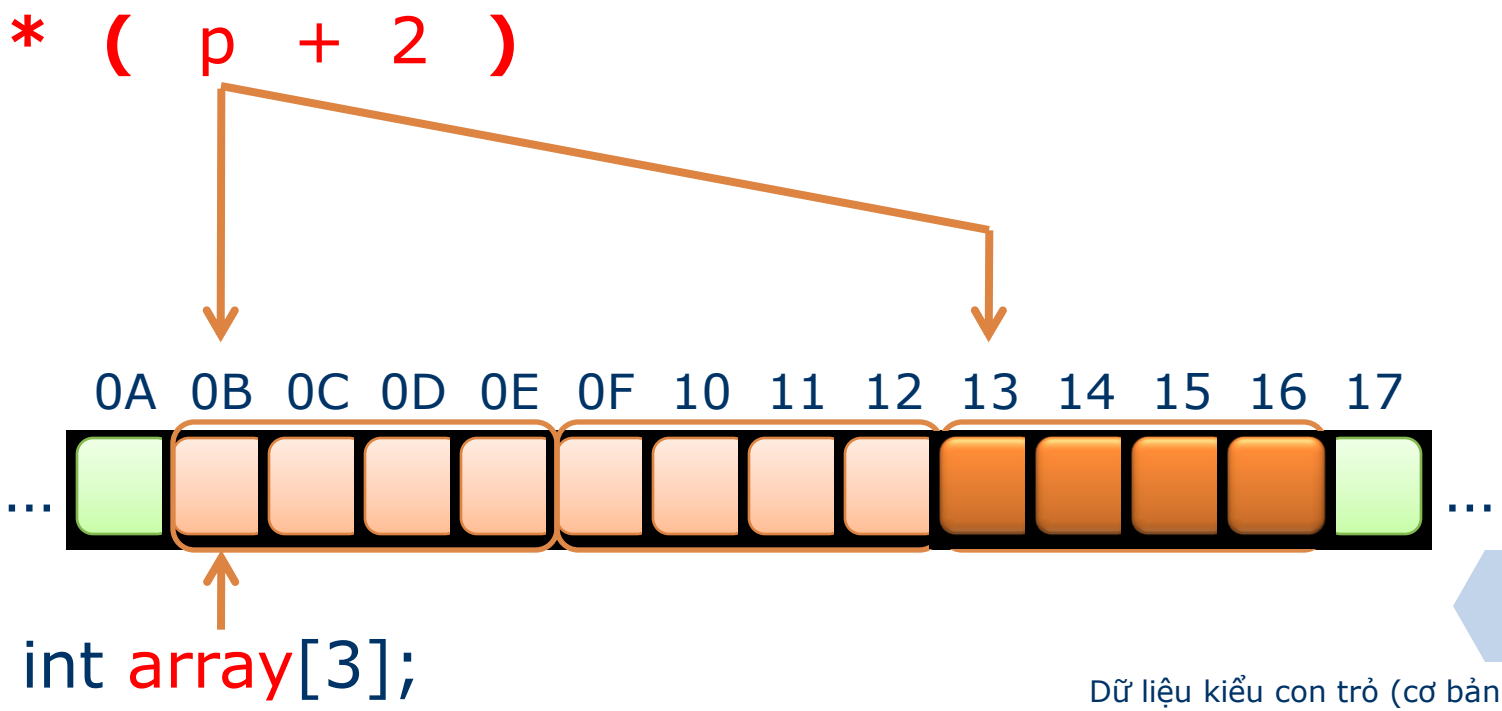


Phép toán số học trên con trỏ

- Các phép toán khác
  - Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)
    - == !=
    - > >=
    - < <=
  - Không thể thực hiện các phép toán: * / %

Con trỏ và mảng một chiều

- Truy xuất đến phần tử thứ n của mảng
  - int array[3], n = 2, *p = array;
  - ➔ array[n] == p[n] == *(p + n)



Con trỏ và mảng một chiều

Ví dụ nhập mảng

```
void main()
{
        int a[10], n = 10, *pa;
        pa = a;      // hoặc pa = &a[0];

        for (int i = 0; i<n; i++)
                scanf("%d", &a[i]);
                scanf("%d", &pa[i]);
                scanf("%d", a + i);
                scanf("%d", pa + i);
                scanf("%d", a++);
                scanf("%d", pa++);
}
```

➔ **&a[i] ⇔ (a + i) ⇔ (pa + i) ⇔ &pa[i]**

Con trỏ và mảng một chiều

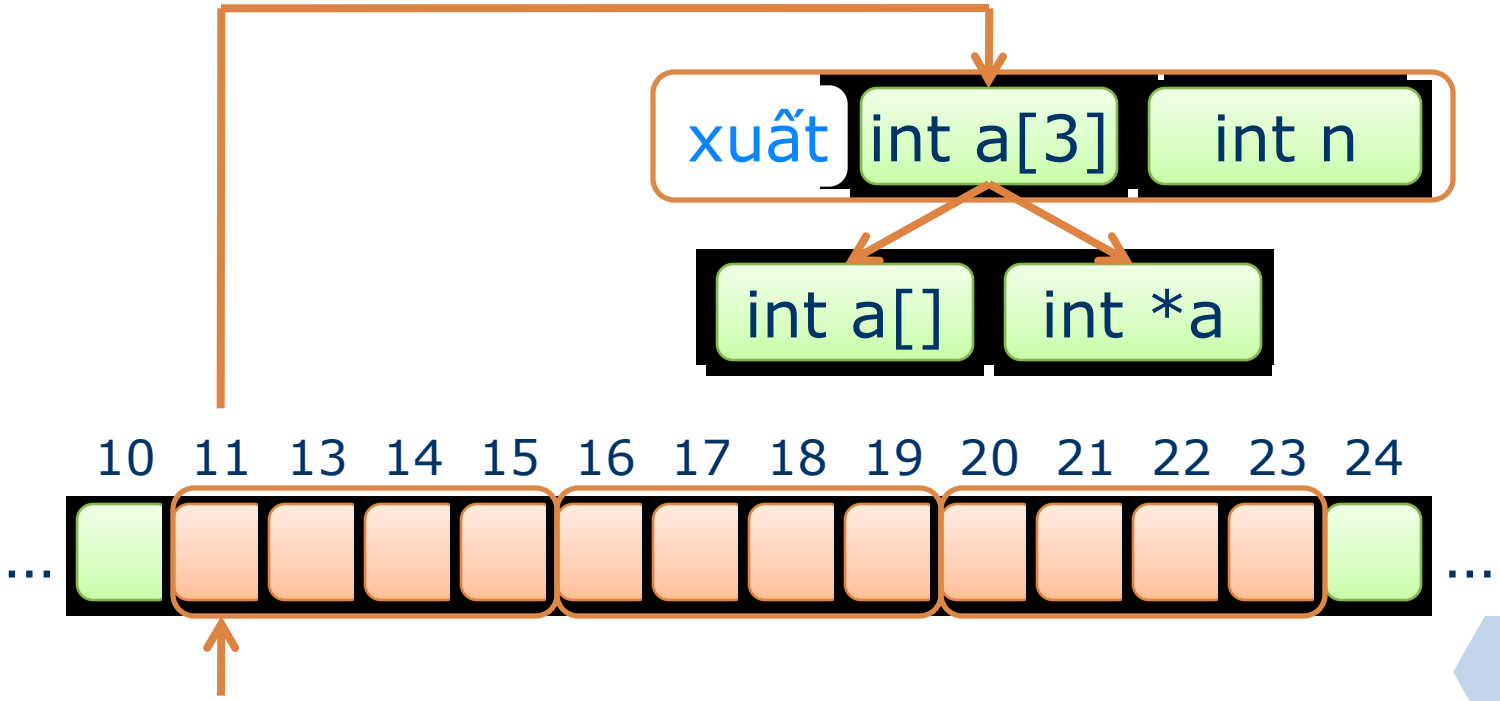
Ví dụ xuất mảng

```
void main()
{
        int a[10], n = 10, *pa;
        pa = a;      // hoặc pa = &a[0];
        …
        for (int i = 0; i<n; i++)
              printf("%d", a[i]);
              printf("%d", pa[i]);
              printf("%d", *(a + i));
              printf("%d", *(pa + i));
              printf("%d", *(a++));
              printf("%d", *(pa++));
}
```

➔ a[i] ⇔ *(a + i) ⇔ *(pa + i) ⇔ pa[i]

Truyền mảng 1 chiều cho hàm

- Chú ý!
  - Mảng một chiều truyền cho hàm là địa chỉ của phần tử đầu tiên chứ không phải toàn mảng.



Đối số mảng truyền cho hàm

- Ví dụ

```
void xuat(int a[10], int n)
{
	for (int i = 0; i<n; i++)
		printf("%d", *(a++));	// OK
}
void main()
{
	int a[10], n = 10;

	for (int i = 0; i<n; i++)
		printf("%d", *(a++));	// Lỗi
}
```

➔ Đối số mảng truyền cho hàm **không phải hằng con trỏ**.

Con trỏ và mảng một chiều

- Lưu ý
  - Không thực hiện các phép toán *, /, %.
  - Tăng/giảm con trỏ n đơn vị có nghĩa là tăng/giảm giá trị của nó n*sizeof(<kiểu dữ liệu mà nó trỏ đến>) (bytes)
  - Không thể tăng/giảm biến mảng (con trỏ hằng). Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm con trỏ đó.
  - Đối số mảng một chiều truyền cho hàm là địa chỉ phần tử đầu tiên của mảng.

Con trỏ và cấu trúc

- Truy xuất bằng 2 cách

```
<tên biến con trỏ cấu trúc>-><tên thành phần>
(*<tên biến con trỏ cấu trúc>).<tên thành phần>
```

- Ví dụ

```
typedef struct
{
        int tu, mau;
} PHANSO;
PHANSO ps1, *ps2 = &ps1; // ps2 là con trỏ

ps1.tu = 1; ps1.mau = 2;
ps2->tu = 1; ps2->mau = 2;
⇔ (*ps2).tu = 1; (*ps2).mau = 2;
```

Bài tập

- Bài 1: Cho đoạn chương trình sau:

```
float pay;
float *ptr_pay;
pay=2313.54;
ptr_pay = &pay;
```

➔ Hãy cho biết giá trị của:

a. pay

b. *ptr_pay

c. *pay

d. &pay

Bài tập

Bài 2: Tìm lỗi

```
#include <stdio.h>
#include <conio.h>

void main()
{
	int *x, y = 2;

	*x = y;
	*x += y++;

	printf("%d %d", *x, y);
	getch();
}
```

Bài tập

❖ Bài 3: Cho đoạn chương trình sau:

```
int  *pint;
float f;
char c;
double *pd;
```

➔ Hãy chọn phát biểu sai cú pháp:

a. f = *pint;

b. c = *pd;

c. *pint = *pd;

d. pd = f;

Bài tập

- Bài 4: Toán tử nào dùng để xác định địa chỉ của một biến?

  ➔ Toán tử &, ví dụ int a; int *p = &a;

- Bài 5: Toán tử nào dùng để xác định giá trị của biến do con trỏ trỏ đến?

  ➔ Toán tử *, ví dụ int a; int *p = &a; *p = 10;

- Bài 6: Phép lấy giá trị gián tiếp là gì?

  ➔ Cách truy xuất giá trị một biến thông qua con trỏ đến biến đó.

  ➔ Ví dụ: int a; int *p = &a; *p = 10;

Bài tập

- Bài 7: Các phần tử trong mảng được sắp xếp trong bộ nhớ như thế nào?

  ➔ Thành một dãy liên tiếp trong bộ nhớ, phần tử mảng chỉ số nhỏ hơn sẽ ở địa chỉ thấp hơn.

- Bài 8: Cho mảng một chiều data. Trình bày 2 cách lấy địa chỉ phần tử đầu tiên của mảng này.

  ➔ int data[10]; int *p;

  ➔ Cách 1: p = &data[0];

  ➔ Cách 2: p = data;

Bài tập

- Bài 9: Trình bày 6 loại phép toán có thể thực hiện trên con trỏ?
  - ➔ 1. Toán tử gán: =
  - ➔ 2. Toán tử lấy địa chỉ: &
  - ➔ 3. Toán tử lấy giá trị gián tiếp: *
  - ➔ 4. Toán tử tăng và giảm: + và –
  - ➔ 5. Toán tử lấy khoảng cách: –
  - ➔ 6. Toán tử so sánh: > >= < <= == !=

Bài tập

- Bài 10: Cho con trỏ p1 trỏ đến phần tử thứ 3 còn con trỏ p2 trỏ đến phần tử thứ 4 của mảng int thì p2 – p1 = ?
  - ➔ int data[10];
  - ➔ int *p1 = &data[2];
  - ➔ int *p2 = &data[3];
  - ➔ p2 – p1 = 1
- Bài 11: Giống như câu trên nhưng đối với mảng float?
  - ➔ 1

Bài tập

- Bài 12: Trình bày khai báo con trỏ pchar trỏ đến kiểu char.

  ➔ char *pchar;

- Bài 13: Cho biến cost kiểu int. Khai báo và khởi tạo con trỏ pcost trỏ đến biến này.

  ➔ int *pcost = &cost;

- Bài 14: Gán giá trị 100 cho biến cost sử dụng hai cách trực tiếp và gián tiếp.

  ➔ Trực tiếp: cost = 100;

  ➔ Gián tiếp: *pcost = 100;

Bài tập

- Bài 15: In giá trị của con trỏ và giá trị của biến mà nó trỏ tới.
  - ➔ printf("%u", pcost);
  - ➔ printf("%d", *pcost);
- Bài 16: Sử dụng con trỏ để làm lại các bài tập về mảng một chiều.
  - Nhập/Xuất mảng
  - Tìm phần tử thỏa yêu cầu
  - Tính tổng/đếm các phần tử thỏa yêu cầu
  - Sắp xếp tăng/giảm

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DỮ LIỆU KIỂU CON TRỎ (NÂNG CAO)

Nội dung

1. Con trỏ cấp 2
2. Con trỏ và mảng nhiều chiều
3. Mảng con trỏ
4. Con trỏ hàm

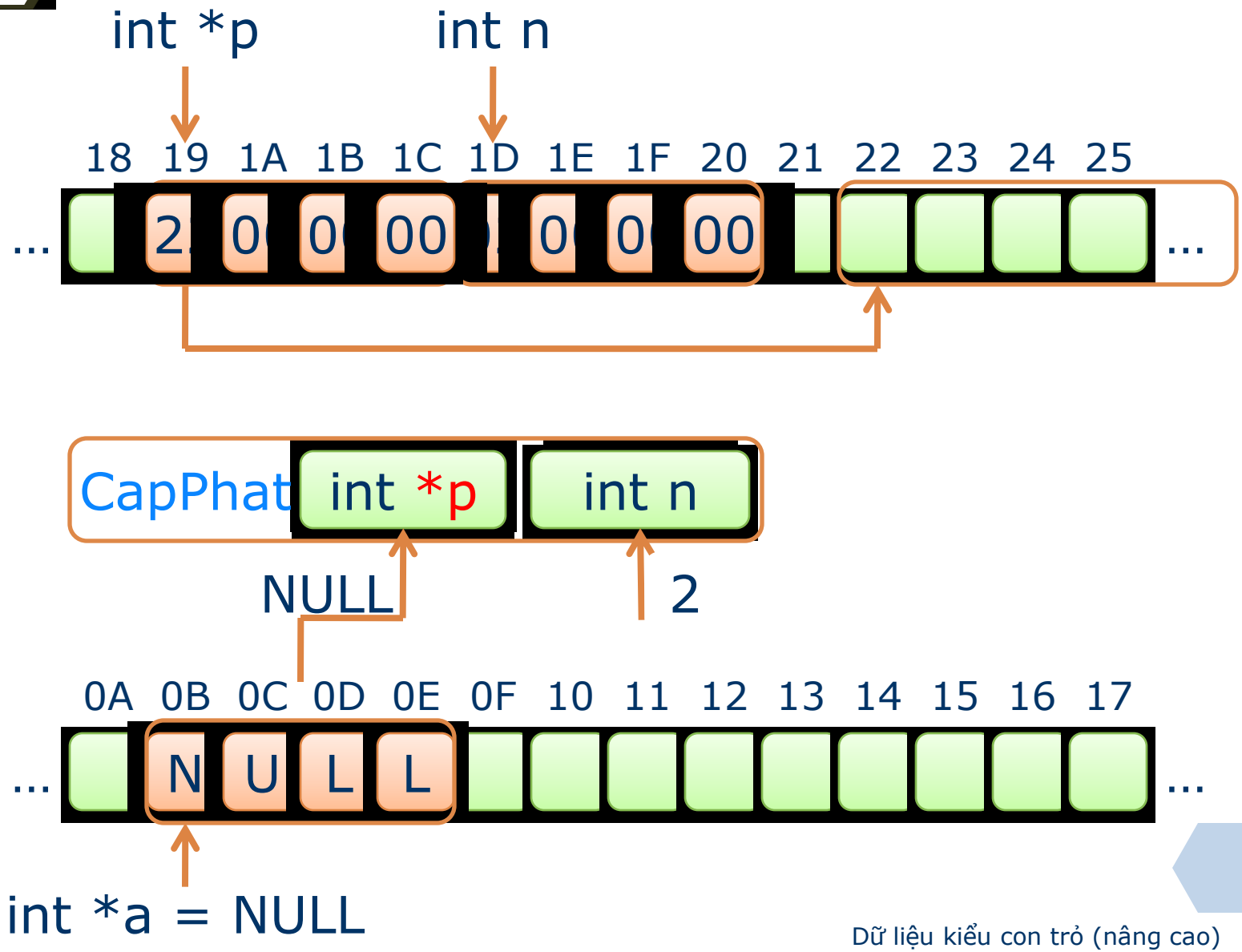
Con trỏ cấp 2 (con trỏ đến con trỏ)

Đặt vấn đề

```
void CapPhat(int *p, int n)
{
	p = (int *)malloc(n * sizeof(int));
}
void main()
{
	int *a = NULL;
	CapPhat(a, 2);
	// a vẫn = NULL
}
```

Làm sao thay đổi giá trị của con trỏ (không phải giá trị mà nó trỏ đến) sau khi gọi hàm?

Con trỏ cấp 2


int *p
int n
18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25
...
2
0
0
00
0
0
00
...
CapPhat
int *p
int n
NULL
2
0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
...
N
U
L
L
...
int *a = NULL

Con trỏ cấp 2

- Giải pháp
  - Sử dụng tham chiếu int *&p (trong C++)

```
void CapPhat(int *&p, int n)
{
        p = (int *)malloc(n * sizeof(int));
}
```

  - Không thay đổi trực tiếp tham số mà trả về

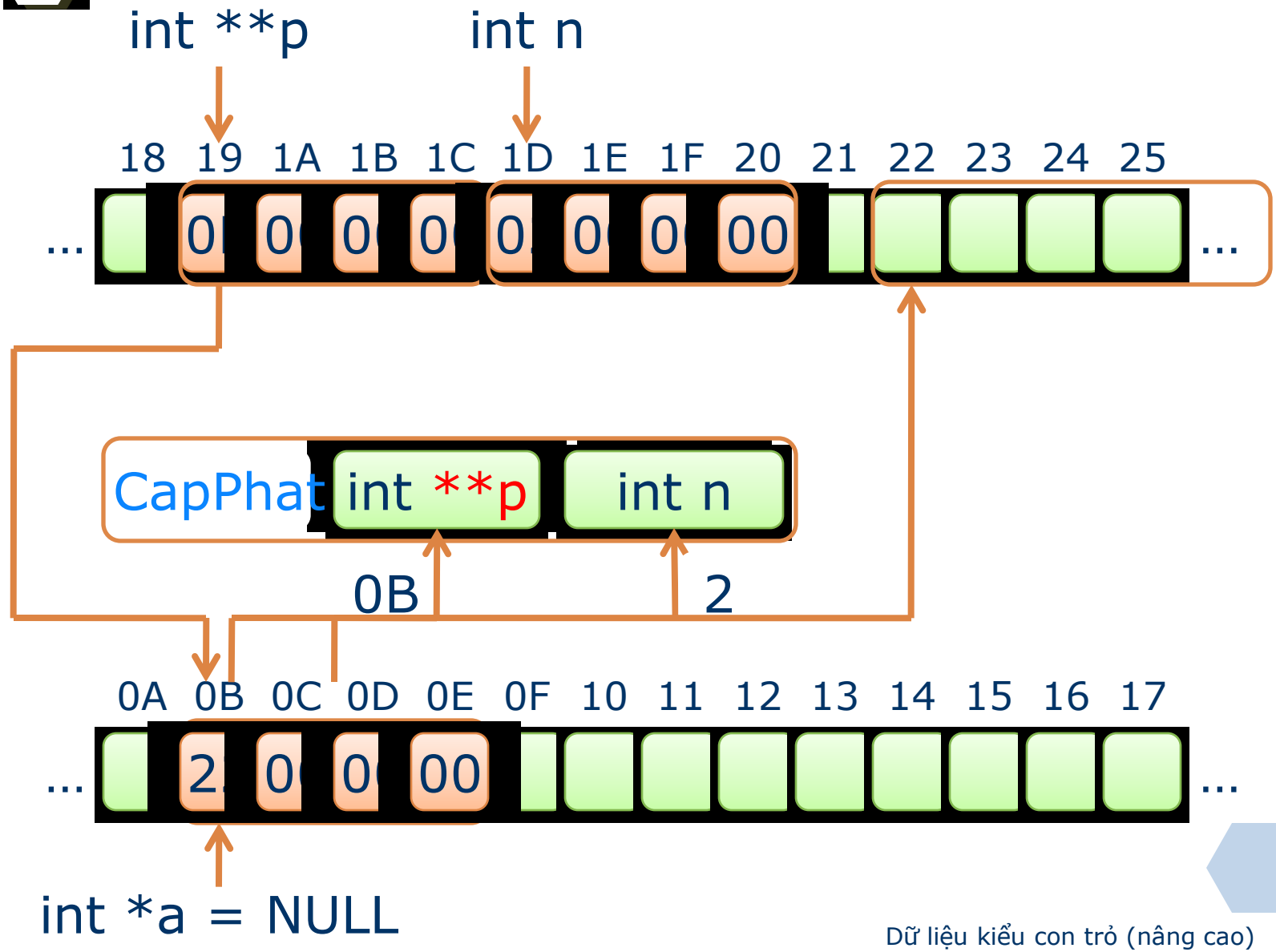
```
int* CapPhat(int n)
{
        int *p = (int *)malloc(n * sizeof(int));
        return p;
}
```

Con trỏ cấp 2

- Giải pháp
  - Sử dụng con trỏ p trỏ đến con trỏ a này. Hàm sẽ thay đổi giá trị của con trỏ â gián tiếp thông qua con trỏ p.

```
void CapPhat(int **p, int n)
{
	*p = (int *)malloc(n * sizeof(int));
}

void main()
{
	int *a = NULL;
	CapPhat(&a, 4);
}
```

Con trỏ cấp 2


int **p
int n
18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25
0 0 0 0 0 0 0 00
CapPhat
int **p
int n
0B
2
0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
2 0 0 00
int *a = NULL

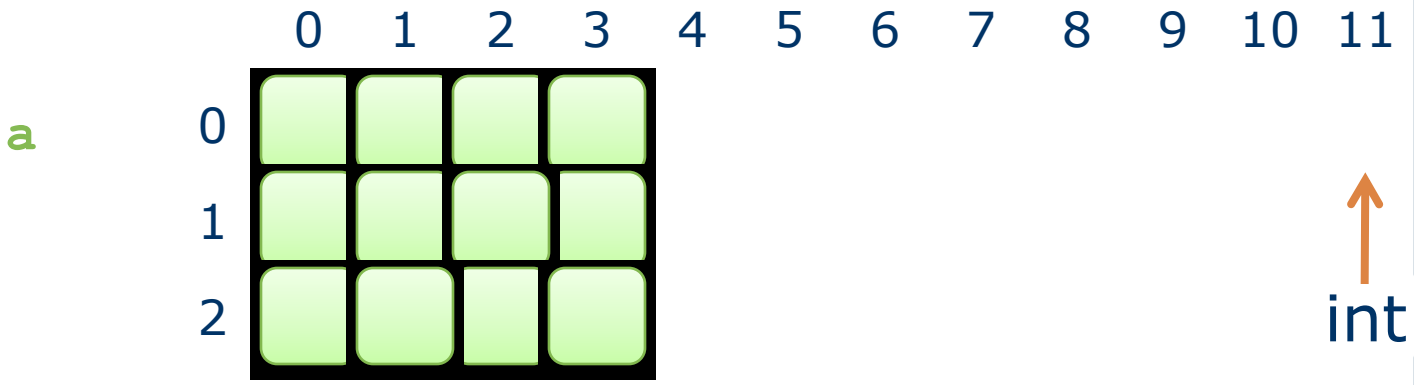
Con trỏ cấp 2

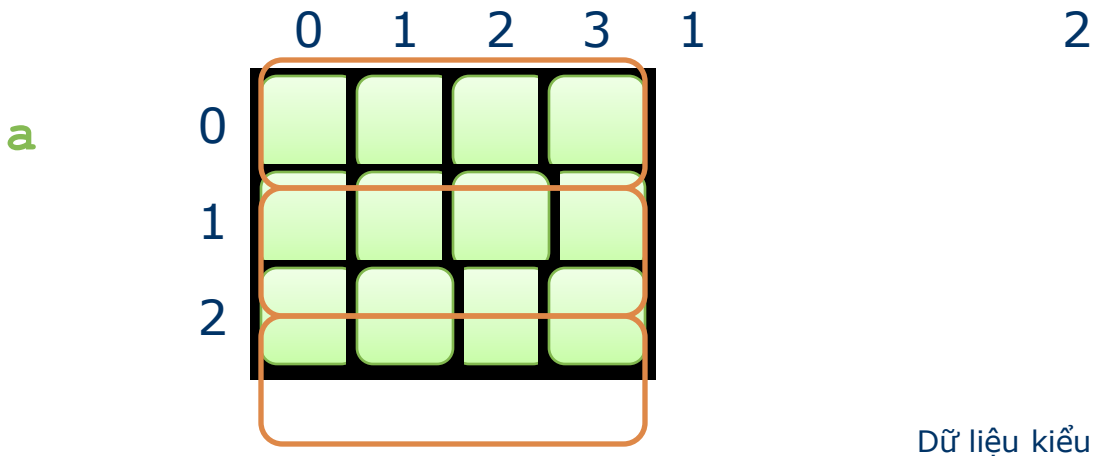
Lưu ý

```
int x = 12;
int *ptr = &x;                  // OK
int k = &x; ptr = k;            // Lỗi

int **ptr_to_ptr = &ptr;        // OK
int **ptr_to_ptr = &x;          // Lỗi

**ptr_to_ptr = 12;              // OK
*ptr_to_ptr = 12;               // Lỗi

printf("%d", ptr_to_ptr);       // Địa chỉ ptr
printf("%d", *ptr_to_ptr);      // Giá trị ptr
printf("%d", **ptr_to_ptr);     // Giá trị x
```

Con trỏ và mảng 2 chiều

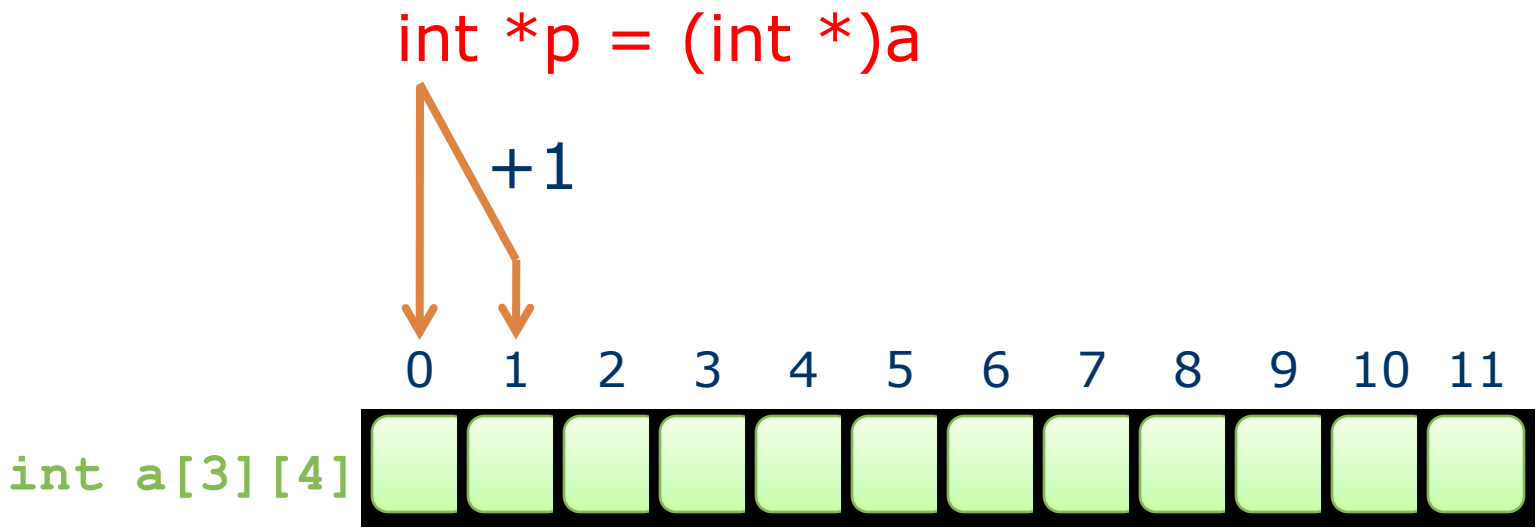
```
int a[3][4];
```





Con trỏ và mảng 2 chiều

- Hướng tiếp cận 1
  - Các phần tử tạo thành mảng 1 chiều
  - Sử dụng con trỏ int * để duyệt mảng 1 chiều



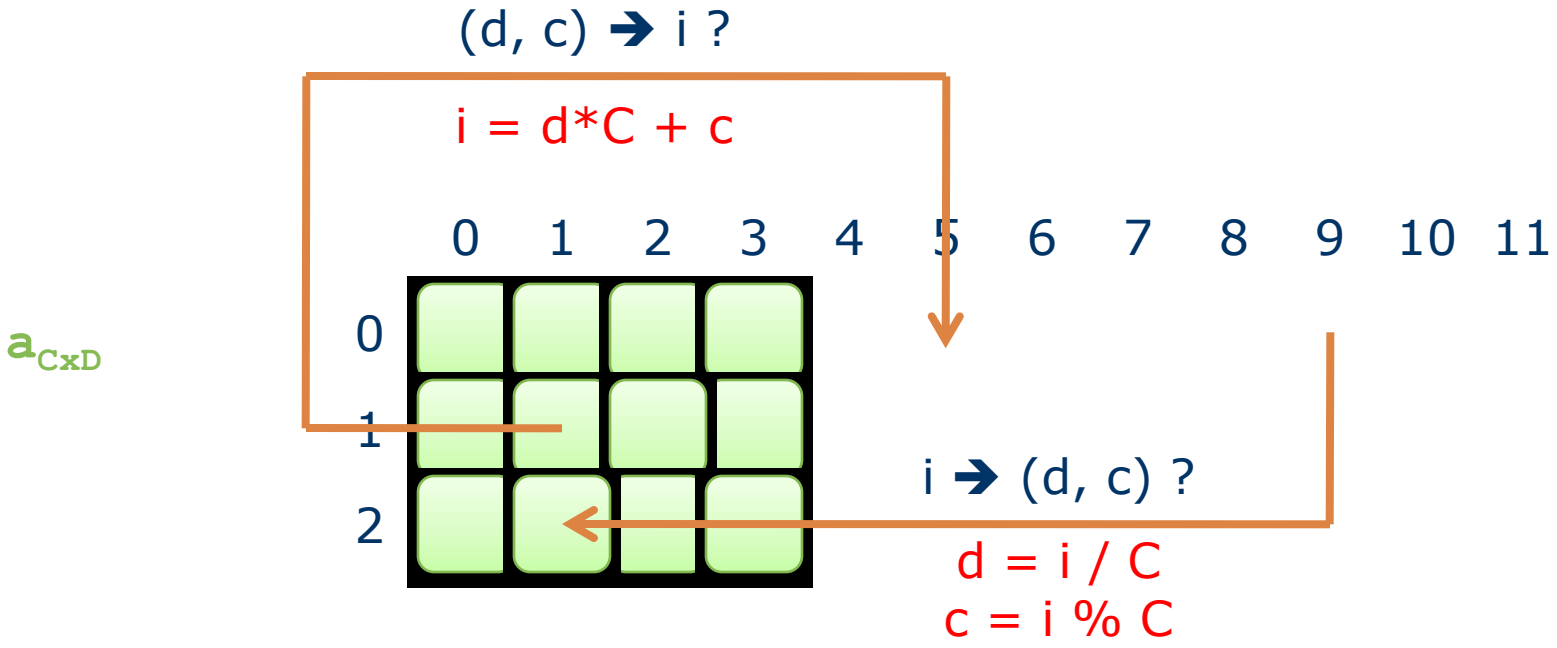
Hướng tiếp cận 1

- Nhập / Xuất theo chỉ số mảng 1 chiều

```
#define D 3
#define C 4
void main()
{
	int a[D][C], i;
	int *p = (int *)a;
	for (i = 0; i < D*C; i++)
	{
		printf("Nhap phan tu thu %d: ", i);
		scanf("%d", p + i);
	}

	for (i = 0; i < D*C; i++)
		printf("%d ", *(p + i));
}
```

Hướng tiếp cận 1

- Liên hệ giữa chỉ số mảng 1 chiều và chỉ số mảng 2 chiều



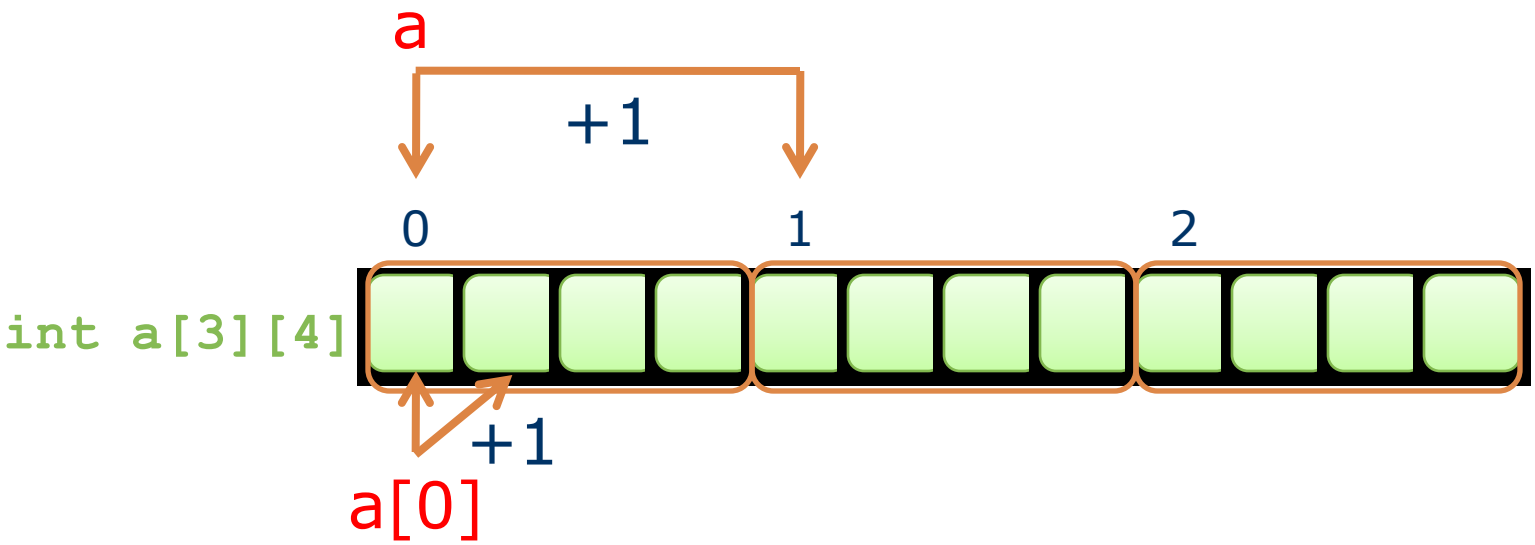
Hướng tiếp cận 1

- Nhập / Xuất theo chỉ số mảng 2 chiều

```
int a[D][C], i, d, c;
int *p = (int *)a;

for (i = 0; i < D*C; i++)
{
      printf("Nhap a[%d][%d]: ", i / C, i % C);
      scanf("%d", p + i);
}
for (d = 0; d < D; d++)
{
      for (c = 0; c < C; c++)
            printf("%d ", *(p + d * C + c));// *p++
      printf("\n";
}
```

Con trỏ và mảng 2 chiều

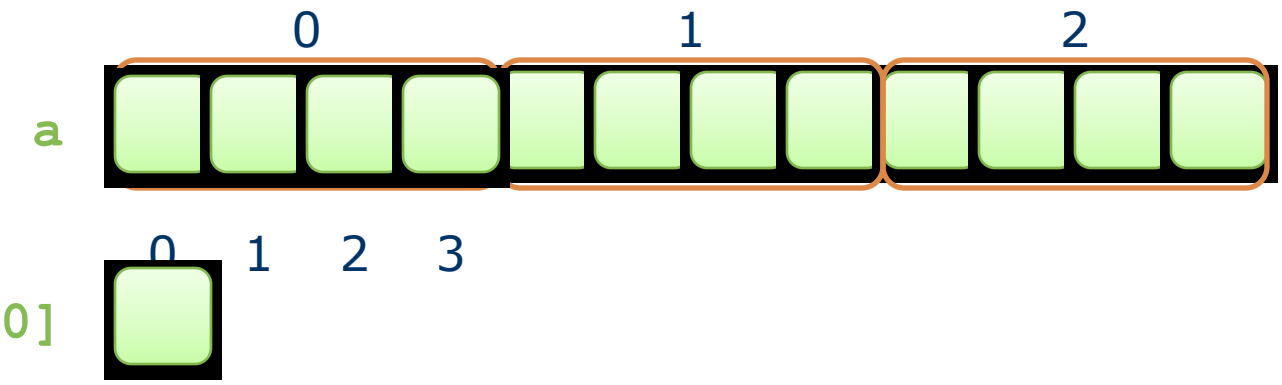
- Hướng tiếp cận 2
  - Mảng 1 chiều, mỗi phần tử là mảng 1 chiều
    - a chứa a[0], a[1], … ➔ a = &a[0]
    - a[0] chứa a[0][0], a[0][1], … ➔ a[0] = &a[0][0]



Hướng tiếp cận 2

Kích thước của mảng

```
void main()
{
        int a[3][4];
        printf("KT của a = %d", sizeof(a));
        printf("KT của a[0] = %d", sizeof(a[0]));
        printf("KT của a[0][0] = %d", sizeof(a[0][0]));
}
```



Hướng tiếp cận 2

- Nhận xét
  - a là con trỏ đến a[0], a[0] là con trỏ đến a[0][0] ➔ a là con trỏ cấp 2.
  - Có thể truy xuất a[0][0] bằng 3 cách:

```
void main()
{
        int a[3][4];
        a[0][0] = 1;
        *a[0] = 1;
        **a = 1;

        a[1][0] = 1; *a[1] = 1; **(a+1) = 1;
        a[1][2] = 1; *(a[1]+2) = 1; *(*(a+1)+2) = 1;
}
```

Hướng tiếp cận 2

- Truyền mảng cho hàm
  - Truyền địa chỉ phần tử đầu tiên cho hàm.
  - Khai báo con trỏ rồi gán địa chỉ mảng cho con trỏ này để nó trỏ đến mảng.
  - Con trỏ này phải cùng kiểu với biến mảng, tức là con trỏ đến vùng nhớ n phần tử (mảng)
- Cú pháp

```
<kiểu dữ liệu> (*<tên con trỏ>)[<số phần tử>];
```

- Ví dụ

```
int (*ptr)[4];
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_1_Mang_C1(int (*ptr)[4])   // ptr[][4]
{
        int *p = (int *)ptr;
        for (int i = 0; i < 4; i++)
              printf("%d ", *p++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;
        for (int i = 0; i < 3; i++)
              Xuat_1_Mang_C1(ptr++);   // hoặc ptr + i
              Xuat_1_Mang_C1(a++);     // sai => a + i
}
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_1_Mang_C2(int *ptr, int n)        // ptr[]
{
        for (int i = 0; i < n; i++)
                printf("%d ", *ptr++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;
        for (int i = 0; i < 3; i++)
                Xuat_1_Mang_C2((int *)ptr++);
                Xuat_1_Mang_C2((int *)(a + i));// a++ sai
}
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_n_Mang_C1(int (*ptr)[4], int n)
{
        int *p = (int *)ptr;
        for (int i = 0; i < n * 4; i++)
              printf("%d ", *p++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;

        Xuat_n_Mang_1(ptr, 3);
        Xuat_n_Mang_1(a, 3);
}
```

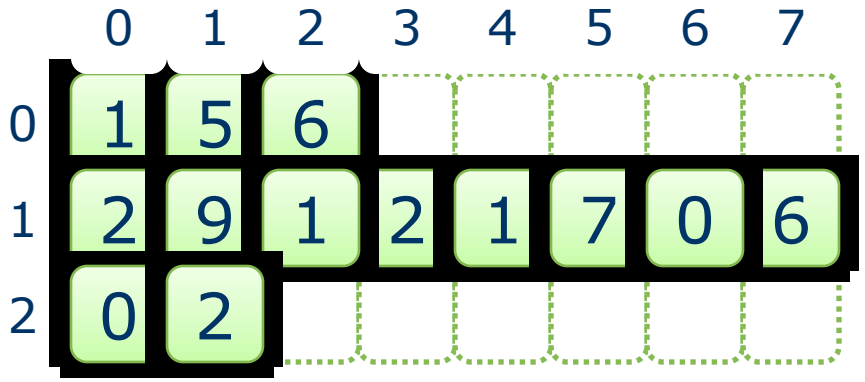
Hướng tiếp cận 2

- Truyền mảng cho hàm

```
void Xuat_n_Mang_C2(int (*ptr)[4], int n)
{
        int *p;
        for (int i = 0; i < n; i++)
        {
                p = (int *)ptr++;

                for (int i = 0; i < 4; i++)
                        printf("%d ", *p++);

                printf("\n");
        }
}
```

Mảng con trỏ

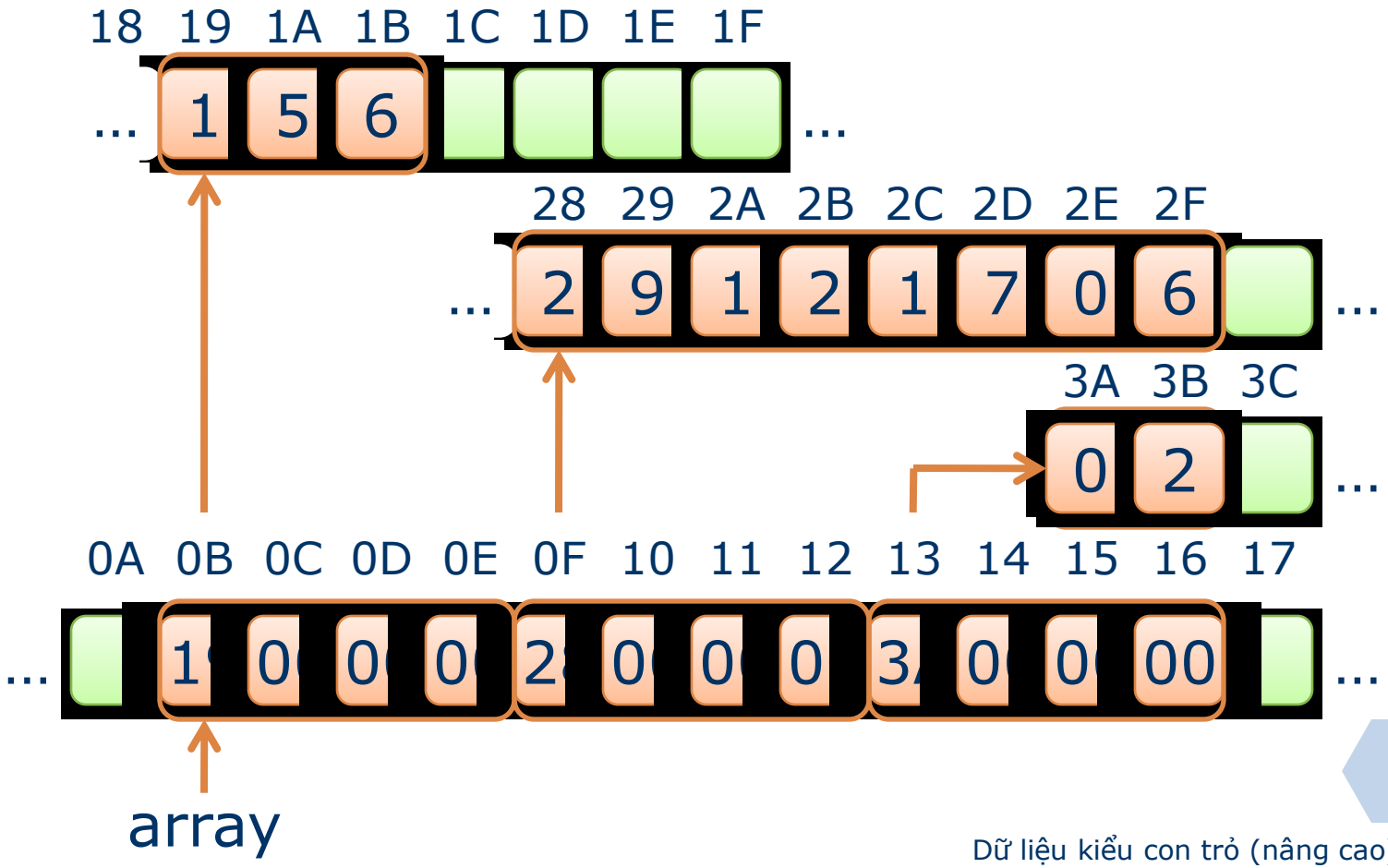
- Đặt vấn đề
  - Sử dụng cấu trúc dữ liệu nào để lưu trữ thông tin sau?



- Giải pháp?
  - Cách 1: Mảng 2 chiều 3x8 (tốn bộ nhớ)

Mảng con trỏ

- Cách 2: Mảng 1 chiều các con trỏ



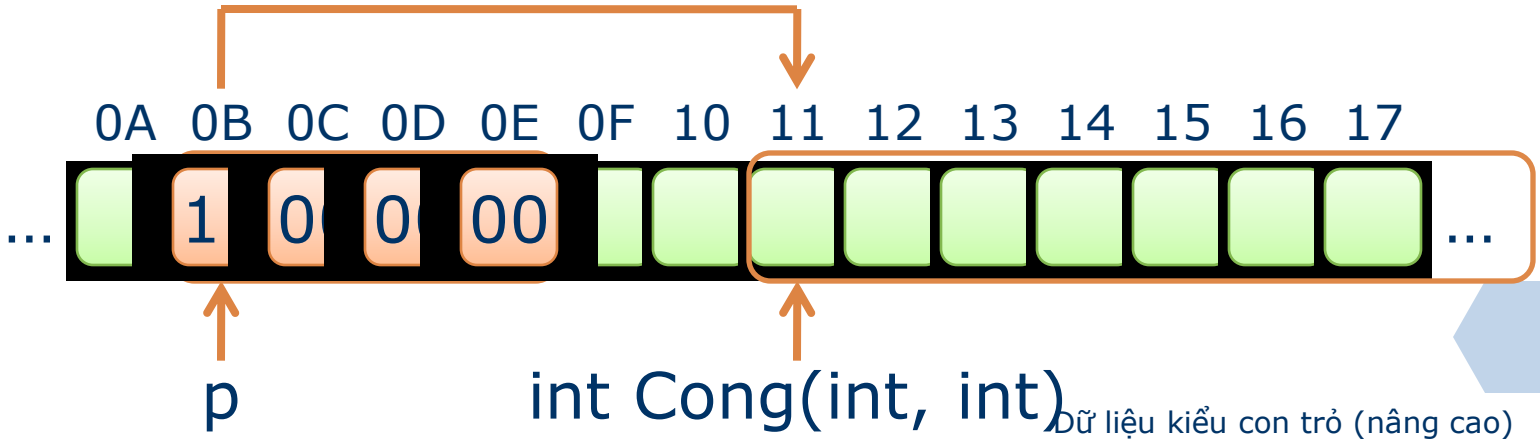
Mảng con trỏ

- Ví dụ

```
void print_strings(char *p[], int n)
{
        for (int i = 0; i<n; i++)
                printf("%s ", p[i]);
}

void main()
{
        char *message[4] = {"Tin", "Hoc", "Co", "So"};

        print_strings(message, 4);
}
```

Con trỏ hàm

- Khái niệm
  - Hàm cũng được lưu trữ trong bộ nhớ, tức là cũng có địa chỉ.
  - Con trỏ hàm là con trỏ trỏ đến vùng nhớ chứa hàm và có thể gọi hàm thông qua con trỏ đó.



Con trỏ hàm

- Khai báo tường minh

```
<kiểu trả về> (* <tên biến con trỏ>)(ds tham số);
```

- Ví dụ

```
// Con trỏ đến hàm nhận đối số int, trả về int
int (*ptof1)(int x);

// Con trỏ đến hàm nhận 2 đối số double, không trả về
void (*ptof2)(double x, double y);

// Con trỏ đến hàm nhận đối số mảng, trả về char
char (*ptof3)(char *p[]);

// Con trỏ đến không nhận đối số và không trả về
void (*ptof4)();
```

Con trỏ hàm

- Khai báo không tường minh (thông qua kiểu)

```
typedef <kiểu trả về> (* <tên kiểu>)(ds tham số);
<tên kiểu> <tên biến con trỏ>;
```

- Ví dụ

```
int (*pt1)(int, int);    // Tường minh

typedef int (*PhepToan)(int, int);

PhepToan pt2, pt3;       // Không tường minh
```

Con trỏ hàm

Gán giá trị cho con trỏ hàm

```
<biến con trỏ hàm> = <tên hàm>;
<biến con trỏ hàm> = &<tên hàm>;
```

- Hàm được gán phải cùng dạng (vào, ra)

Ví dụ

```
int Cong(int x, int y);                    // Hàm
int Tru(int x, int y);                     // Hàm
int (*tinhtoan)(int x, int y);             // Con trỏ hàm

tinhtoan = Cong;   // Dạng ngắn gọn
tinhtoan = &Tru;   // Dạng sử dụng địa chỉ
tinhtoan = NULL;   // Không trỏ đến đâu cả
```

Con trỏ hàm

So sánh con trỏ hàm

```
if  (tinhtoan != NULL)
{
      if (tinhtoan == &Cong)
            printf("Con trỏ đến hàm Cong.");
      else
            if (tinhtoan == &Tru)
                  printf("Con trỏ đến hàm Tru.");
            else
                  printf("Con trỏ đến hàm khác.");
}
else
      printf("Con trỏ chưa được khởi tạo!");
```

Con trỏ hàm

- Gọi hàm thông qua con trỏ hàm
  - Sử dụng toán tử lấy nội dung "*" (chính quy) nhưng trường hợp này có thể bỏ

```
int Cong(int x, int y);
int Tru(int x, int y);

int (*tinhtoan)(int, int);

tinhtoan = Cong;
int kq1 = (*tinhtoan)(1, 2);   // Chính quy
int kq2 = tinhtoan(1, 2);      // Ngắn gọn
```

Con trỏ hàm

Truyền tham số là con trỏ hàm

```
int Cong(int x, int y);
int Tru(int x, int y);
int TinhToan(int x, int y, int (*pheptoan)(int, int))
{
        int kq = (*pheptoan)(x, y);    // Gọi hàm
        return kq;
}

void main()
{
        int (*pheptoan)(int, int) = &Cong;
        int kq1 = TinhToan(1, 2, pheptoan);
        int kq2 = TinhToan(1, 2, &Tru);
}
```

Con trỏ hàm

Trả về con trỏ hàm

```
int (*LayPhepToan(char code))(int, int)
{
	if (code == '+')
		return &Cong;
	return &Tru;
}

void main()
{
	int (*pheptoan)(int, int) = NULL;
	pheptoan = LayPhepToan('+');
	int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm

- Trả về con trỏ hàm (khai báo kiểu)

```
typedef (*PhepToan)(int, int);
PhepToan LayPhepToan(char code)
{
	if (code == '+')
		return &Cong;
	return &Tru;
}

void main()
{
	PhepToan pheptoan = NULL;
	pheptoan = LayPhepToan('+');
	int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm

Mảng con trỏ hàm

```
typedef (*PhepToan)(int, int);
void main()
{
        int (*array1[2])(int, int);    // tường minh
        PhepToan array2[2];            // kô tường minh

        array1[0] = array2[1] = &Cong;
        array1[1] = array2[0] = &Tru;

        printf("%d\n", (*array1[0])(1, 2));
        printf("%d\n", array1[1](1, 2));
        printf("%d\n", array2[0](1, 2));
        printf("%d\n", array2[1](1, 2));
}
```

Con trỏ hàm

- Lưu ý
  - Không được quên dấu () khi khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int *PhepToan(int x, int y);
  - Có thể bỏ tên biến tham số trong khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int (*PhepToan)(int, int);

Bài tập

- Câu 1: Ta có thể khai báo và sử dụng biến con trỏ đến cấp thứ mấy?
- Câu 2: Có sự khác nhau giữa con trỏ đến một chuỗi và con trỏ đến một mảng ký tự không?
- Câu 3: Nếu không sử dụng các kiến thức nâng cao về con trỏ, ta có thể giải quyết một số bài toán nào đó không?
- Câu 4: Hãy nên một số ứng dụng của con trỏ hàm.

Bài tập

- Câu 5: Viết đoạn lệnh khai báo biến x kiểu float, khai báo và khởi tạo con trỏ px đến biến x và khai báo và khởi tạo con trỏ ppx đến con trỏ px.
- Câu 6: Ta muốn gán 100 cho x thông qua con trỏ ppx bằng biểu thức gán "ppx = 100;" có được không?
- Câu 7: Giả sử ta khai báo mảng array 3 chiều int array[2][3][4]. Cho biết cấu trúc của mảng này đối với trình biên dịch C.
- Câu 8: Cho biết array[0][0] có nghĩa là gì?

Bài tập

- Câu 9: Xét xem biểu thức so sánh nào sau đây đúng
  - array[0][0] == & array[0][0][0];
  - array[0][1] == array[0][0][1];
  - array[0][1] == &array[0][1][0];
- Câu 10: Viết nguyên mẫu của một hàm nhận một mảng con trỏ đến kiểu char làm đối số, và giá trị trả về có kiểu void.
- Câu 11: Theo cách viết của câu 10, ta có thể biết được số phần tử của mảng được truyền kô?

Bài tập

- Câu 11: Con trỏ đến hàm là gì?
- Câu 12: Viết khai báo con trỏ đến một hàm mà hàm đó có giá trị trả về kiểu char, nhận đối số là một mảng con trỏ đến kiểu char.
- Câu 13: Ta viết khai báo con trỏ ở câu 12 như vậy có đúng không? char *ptr(char *x[]);
- Câu 14: Cho biết ý nghĩa của các khai báo sau:
  - int *var1;
  - int var2;
  - int **var3;

Bài tập

- Câu 15: Cho biết ý nghĩa của các khai báo sau:
  - int a[3][12];
  - int (*b)[12];
  - int *c[12];
- Câu 16: Cho biết ý nghĩa của các khai báo sau:
  - char *z[10];
  - char *y(int field);
  - char (*x)(int field);

Bài tập

- Câu 17: Viết khai báo con trỏ func đến một hàm nhận đối số là một số nguyên và trả về giá trị kiểu float.
- Câu 18: Viết khai báo một mảng con trỏ đến hàm. Các hàm nhận một chuỗi ký tự làm tham số và trả về giá trị kiểu nguyên. Ta có thể sử dụng mảng này để làm gì?
- Câu 19: Viết câu lệnh khai báo một mảng 10 con trỏ đến kiểu char.

Bài tập

- Câu 20: Tìm lỗi sai trong đoạn lệnh sau
  - int x[3][12];
  - int *ptr[12];
  - ptr = x;
- Câu 21: Viết chương trình khai báo mảng hai chiều có 12x12 phần tử kiểu char. Gán ký tự 'X' cho mọi phần tử của mảng này. Sử dụng con trỏ đến mảng để in giá trị các phần tử mảng lên màn hình ở dạng lưới.

Bài tập

- Câu 22: Viết chương trình khai báo mảng 10 con trỏ đến kiểu float, nhận 10 số thực từ bàn phím, sắp xếp lại và in ra màn hình dãy số đã sắp xếp.
- Câu 23: Sửa lại bài tập 22 để người sử dụng có thể lựa chọn cách sắp xếp theo thứ tự tăng hay giảm dần.

Bài tập

- Câu 24: Chương trình cho phép người dùng nhập các dòng văn bản từ bàn phím đến khi nhập một dòng trống. Chương trình sẽ sắp xếp các dòng theo thứ tự alphabet rồi hiển thị chúng ra màn hình.
- Câu 25: Sử dụng con trỏ hàm để viết các hàm sắp xếp sau
  - Tăng dần
  - Giảm dần
  - Dương giảm rồi âm tăng, cuối cùng là số 0
  - ...

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CHUYỂN ĐỔI KIỂU DỮ LIỆU & CẤP PHÁT BỘ NHỚ ĐỘNG

Nội dung

1. Chuyển đổi kiểu (ép kiểu)
2. Cấu trúc CT C trong bộ nhớ
3. Cấp phát bộ nhớ động
4. Các thao tác trên khối nhớ

Nhu cầu chuyển đổi kiểu

- Mọi đối tượng dữ liệu trong C đều có kiểu xác định
  - Biến có kiểu char, int, float, double, …
  - Con trỏ trỏ đến kiểu char, int, float, double, …
- Xử lý thế nào khi gặp một biểu thức với nhiều kiểu khác nhau?
  - C tự động chuyển đổi kiểu (ép kiểu).
  - Người sử dụng tự chuyển đổi kiểu.

Chuyển đổi kiểu tự động

- Sự tăng cấp (kiểu dữ liệu) trong biểu thức
  - Các thành phần cùng kiểu
    - Kết quả là kiểu chung
    - int / int ➔ int, float / float ➔ float
    - Ví dụ: 2 / 4 ➔ 0, 2.0 / 4.0 ➔ 0.5
  - Các thành phần khác kiểu
    - Kết quả là kiểu bao quát nhất
    - char < int < long < float < double
    - float / int ➔ float / float, …
    - Ví dụ: 2.0 / 4 ➔ 2.0 / 4.0 ➔ 0.5
    - Lưu ý, chỉ chuyển đổi tạm thời (nội bộ).

Chuyển đổi kiểu tự động

- Phép gán <BT vế trái> = <BT vế phải>;
  - BT ở vế phải luôn được tăng cấp (hay giảm cấp) tạm thời cho giống kiểu với BT ở vế trái.

```
int i;
float f = 1.23;

i = f;      // ➔ f tạm thời thành int
f = i;      // ➔ i tạm thời thành float
```

  - Có thể làm mất tính chính xác của số nguyên khi chuyển sang số thực ➔ hạn chế!

```
int i = 3;
float f;
f = i;      // ➔ f = 2.999995
```

Chuyển đổi tường minh (ép kiểu)

Ý nghĩa

- Chủ động chuyển đổi kiểu (tạm thời) nhằm tránh những kết quả sai lầm.

Cú pháp

```
(<kiểu chuyển đổi>)<biểu thức>
```

Ví dụ

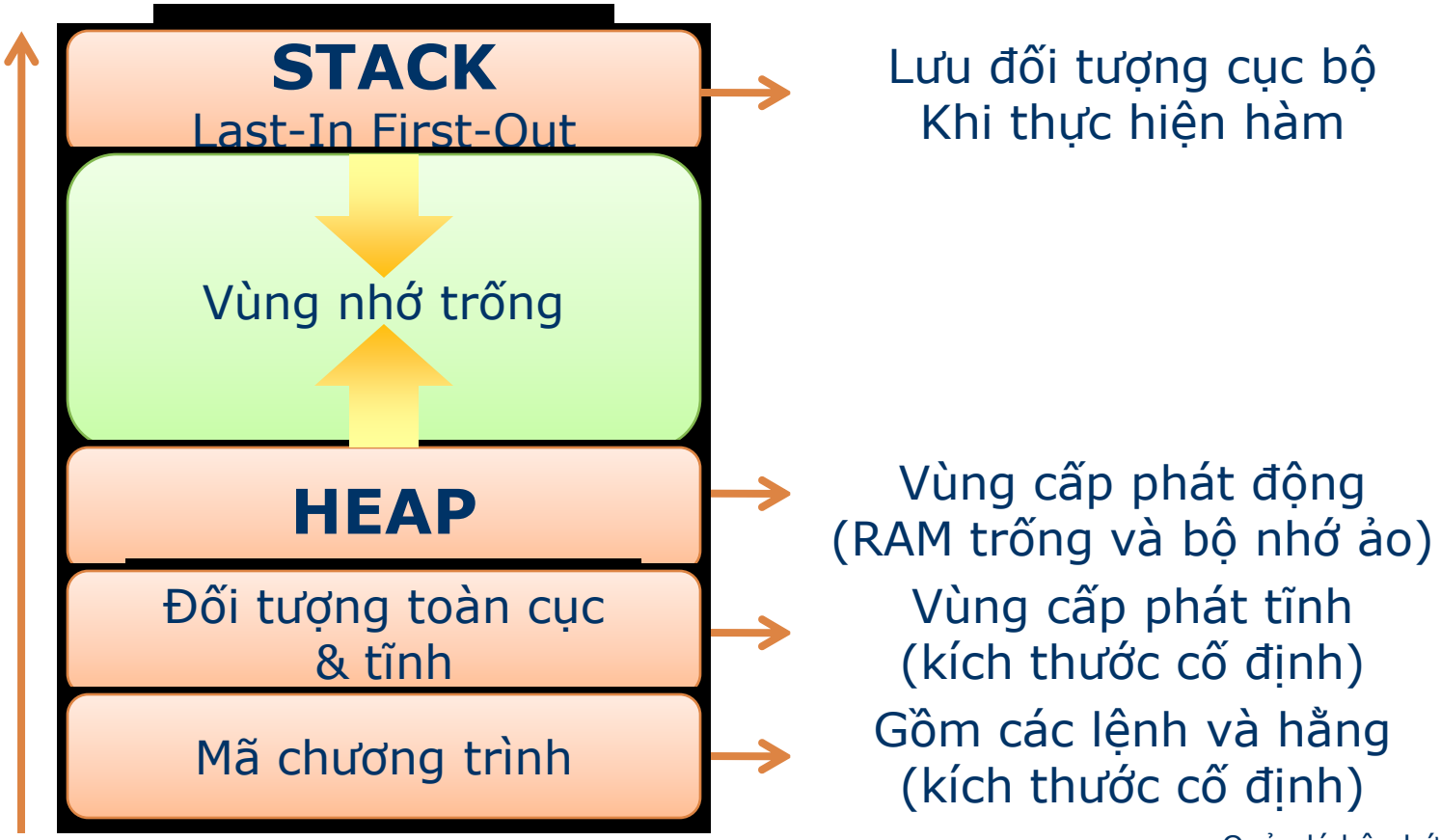
```
int x1 = 1, x2 = 2;
float f1 = x1 / x2;             // → f1 = 0.0
float f2 = (float)x1 / x2;      // → f2 = 0.5
float f3 = (float)(x1 / x2);    // → f3 = 0.0
```

Cấp phát bộ nhớ tĩnh và động

- Cấp phát tĩnh (static memory allocation)
  - Khai báo biến, cấu trúc, mảng, …
  - Bắt buộc phải biết trước cần bao nhiêu bộ nhớ lưu trữ ➔ tốn bộ nhớ, không thay đổi được kích thước, …
- Cấp phát động (dynamic memory allocation)
  - Cần bao nhiêu cấp phát bấy nhiêu.
  - Có thể giải phóng nếu không cần sử dụng.
  - Sử dụng vùng nhớ ngoài chương trình (cả bộ nhớ ảo virtual memory).

Cấu trúc một CT C trong bộ nhớ

- Toàn bộ tập tin chương trình sẽ được nạp vào bộ nhớ tại vùng nhớ còn trống, gồm 4 phần:

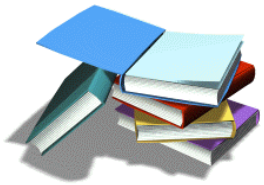


Cấp phát bộ nhớ động

- Thuộc thư viện <stdlib.h> hoặc <alloc.h>
  - malloc
  - calloc
  - realloc
  - free
- Trong C++
  - new
  - delete

Cấp phát bộ nhớ động

void *malloc(size_t size)



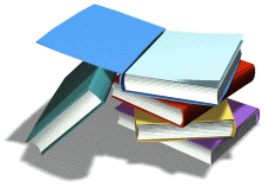
Cấp phát trong HEAP một vùng nhớ size (bytes)
size_t thay cho unsigned (trong <stddef.h>)

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)malloc(10*sizeof(int));
if (p == NULL)
        printf("Khong du bo nho!");
```

Cấp phát bộ nhớ động

void *calloc(size_t num, size_t size)



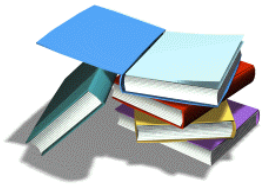
Cấp phát vùng nhớ gồm num phần tử trong HEAP, mỗi phần tử kích thước size (bytes)

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)calloc(10, sizeof(int));
if (p == NULL)
        printf("Khong du bo nho!");
```

Cấp phát bộ nhớ động

void *realloc(void *block, size_t size)



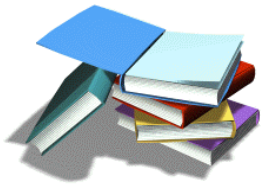
Cấp phát lại vùng nhớ có kích thước size do block trỏ đến trong vùng nhớ HEAP.
block == NULL ➔ sử dụng malloc
size == 0 ➔ sử dụng free

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)malloc(10*sizeof(int));
p = (int *)realloc(p, 20*sizeof(int));
if (p == NULL)
        printf("Khong du bo nho!");
```

Cấp phát bộ nhớ động

void free(void *ptr)



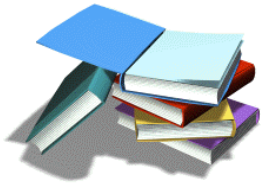
Giải phóng vùng nhớ do ptr trỏ đến, được cấp bởi các hàm malloc(), calloc(), realloc(). Nếu ptr là NULL thì không làm gì cả.

- Không có.

```
int *p = (int *)malloc(10*sizeof(int));
free(p);
```

Cấp phát bộ nhớ động

<pointer_to_datatype> = new <datatype>[size]



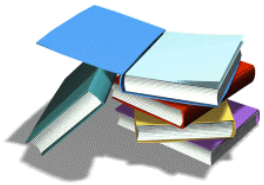
Cấp phát vùng nhớ có kích thước sizeof(<datatype>)*size trong HEAP

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *a1 = (int *)malloc(sizeof(int));
int *a2 = new int;
int *p1 = (int *)malloc(10*sizeof(int));
int *p2 = new int[10];
```

Cấp phát bộ nhớ động

delete []<pointer_to_datatype>



Giải phóng vùng nhớ trong HEAP do <pointer_to_datatype> trỏ đến (được cấp phát bằng new)

- Không có.

```
int *a = new int;
delete a;
int *p = new int[10];
delete []p;
```

Cấp phát bộ nhớ động

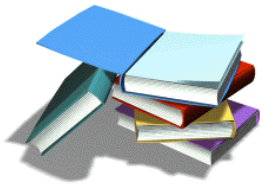
- Lưu ý
  - Không cần kiểm tra con trỏ có NULL hay kô trước khi free hoặc delete.
  - Cấp phát bằng malloc, calloc hay realloc thì giải phóng bằng free, cấp phát bằng new thì giải phóng bằng delete.
  - Cấp phát bằng new thì giải phóng bằng delete, cấp phát mảng bằng new [] thì giải phóng bằng delete [].

Thao tác trên các khối nhớ

- Thuộc thư viện <string.h>
  - memset : gán giá trị cho tất cả các byte nhớ trong khối.
  - memcpy : sao chép khối.
  - memmove : di chuyển thông tin từ khối này sang khối khác.

Thao tác trên các khối nhớ

void *memset(void *dest, int c, size_t count)



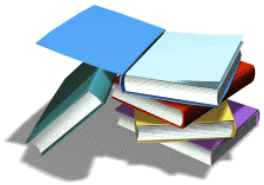
Gán count (bytes) đầu tiên của vùng nhớ mà dest trỏ tới bằng giá trị c (từ 0 đến 255)
Thường dùng cho vùng nhớ kiểu char còn vùng nhớ kiểu khác thường đặt giá trị zero.

- Con trỏ dest.

```
char buffer[] = "Hello world";
printf("Trước khi memset: %s\n", buffer);
memset(buffer, '*', strlen(buffer));
printf("Sau khi memset: %s\n", buffer);
```

Thao tác trên các khối nhớ

void *memcpy(void *dest, void *src, size_t count)



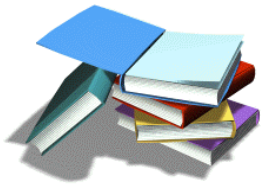
Sao chép chính xác count byte từ khối nhớ src vào khối nhớ dest.
Nếu hai khối nhớ đè lên nhau, hàm sẽ làm việc không chính xác.

- Con trỏ dest.

```
char src[] = "*****";
char dest[] = "0123456789";
memcpy(dest, src, 5);
memcpy(dest + 3, dest + 2, 5);
```

Thao tác trên các khối nhớ

void *memmove(void *dest, void *src, size_t count)



Sao chép chính xác count byte từ khối nhớ src vào khối nhớ dest.
Nếu hai khối nhớ đè lên nhau, hàm vẫn thực hiện chính xác.

- Con trỏ dest.

```
char src[] = "*****";
char dest[] = "0123456789";
memmove(dest, src, 5);
memmove(dest + 3, dest + 2, 5);
```

Bài tập

- Bài 1: Tại sao cần phải giải phóng khối nhớ được cấp phát động?

  ➔ Khối nhớ không tự giải phóng sau khi sử dụng nên sẽ làm giảm tốc độ thực hiện chương trình hoặc tràn bộ nhớ nếu tiếp tục cấp phát

- Bài 2: Điều gì xảy ra nếu ta nối thêm một số ký tự vào một chuỗi (được cấp phát động trước đó) mà không cấp phát lại bộ nhớ cho nó?

  ➔ Nếu chuỗi đủ lớn để chứa thêm thông tin thì không cần cấp phát lại. Ngược lại phải cấp phát lại để có thêm vùng nhớ.

Bài tập

- Bài 3: Ưu điểm của việc sử dụng các hàm thao tác khối nhớ? Ta có thể sử dụng một vòng lặp kết hợp với một câu lệnh gán để khởi tạo hay sao chép các byte nhớ hay không?

  ➔ Việc sử dụng các hàm thao tác khối nhớ như memset, memcpy, memmove giúp khởi tạo hay sao chép/di chuyển vùng nhớ nhanh hơn.

  ➔ Trong một số trường hợp chỉ có thể sử dụng vòng lặp kết hợp với lệnh gán để khởi tạo nếu như các byte nhớ cần khởi tạo khác giá trị.

Bài tập

- Bài 4: Ta thường dùng phép ép kiểu trong những trường hợp nào?

  ➔ Lấy phần nguyên của số thực hoặc lấy phần thực của phép chia hai số nguyên, …
- Bài 5: Giả sử c kiểu char, i kiểu int, l kiểu long. Hãy xác định kiểu của các biểu thức sau:
  - (c + i + l)
  - (i + 'A')
  - (i + 32.0)
  - (100 + 1.0)

Bài tập

- Bài 6: Việc cấp phát động nghĩa là gì?

  ➔ Bộ nhớ được cấp phát động là bộ nhớ được cấp phát trong khi chạy chương trình và có thể thay đổi độ lớn vùng nhớ.

- Bài 7: Cho biết sự khác nhau giữa malloc và calloc?

  ➔ malloc: cấp phát bố nhớ cho một đối tượng.

  ➔ calloc: cấp phát bộ nhớ cho một nhóm đối tượng.

Bài tập

- Bài 8: Viết câu lệnh sử dụng hàm malloc để cấp phát 1000 số kiểu long.
  - ➔ long *ptr;
  - ➔ ptr = (long *)malloc(1000 * sizeof(long));
- Bài 9: Giống bài 7 nhưng dùng calloc
  - ➔ long *ptr;
  - ➔ ptr = (long *)calloc(1000, sizeof(long));
  - ➔ ptr = (long *)calloc(sizeof(long), 1000); !!!

Bài tập

- Bài 10: Cho biết sự khác nhau giữa memcpy và memmove

  ➔ Hàm memmove cho phép sao chép hai vùng nhớ chồng lên nhau trong khi hàm memcpy làm việc không chính xác trong trường hợp này

- Bài 11: Trình bày 2 cách khởi tạo mảng float data[1000]; với giá trị 0.

  ➔ C1: for (int i=0; i<1000; i++) data[i] = 0;

  ➔ C2: memset(data, 0, 1000*sizeof(float));

Bài tập

Bài 12: Kiểm tra lỗi

```
void func()
{
	int n1 = 100, n2 = 3;
	float ketqua = n1 / n2;
	printf("%d / %d = %f", n1, n2, ketqua);
}
```

Bài 13: Kiểm tra lỗi

```
void main()
{
	void *p;
	p = (float *)malloc(sizeof(float));
	*p = 1.23;
}
```

Bài tập

- Bài 14: Viết hàm cấp phát một vùng nhớ đủ chứa n số nguyên với n cho trước và trả về địa chỉ vùng nhớ đó.
- Bài 15: Viết hàm sao chép mảng a, số lượng phần tử n cho trước sang mảng b cho trước (kích thước lớn hơn hay bằng n).
- Bài 16: Viết hàm trả về bản sao của một mảng số nguyên a, số lượng phần tử n cho trước.
- Bài 17: Viết hàm trả về mảng đảo của một mảng số nguyên a, số lượng phần tử n cho trước. Yêu cầu không được thay đổi nội dung mảng a.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DANH SÁCH LIÊN KẾT

Nội dung

1. Các hình thức tổ chức danh sách
2. Các loại danh sách liên kết
3. Thao tác trên DSLK đơn
4. Các ứng dụng của DSLK đơn

Các hình thức tổ chức danh sách

- Mối liên hệ giữa các phần tử được ngầm hiểu
  - Mỗi phần tử có một chỉ số và ngầm hiểu rằng x_{i+1} nằm sau x_i. Do đó các phần tử phải nằm cạnh nhau trong bộ nhớ.
  - Số lượng phần tử cố định. Không có thao tác thêm và hủy mà chỉ có thao tác dời chỗ.
  - Truy xuất ngẫu nhiên đến từng phần tử nhanh chóng.
  - Phí bộ nhớ do không biết trước kích thước.
  - Ví dụ: mảng một chiều.

Các hình thức tổ chức danh sách

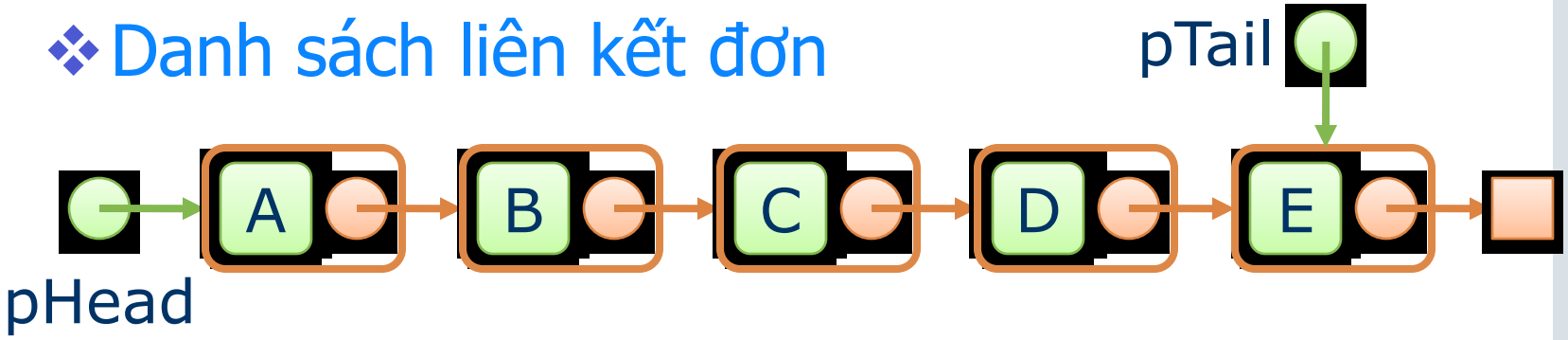
- Mối liên hệ giữa các phần tử rõ ràng
  - Mỗi phần tử ngoài thông tin bản thân còn có thêm liên kết (địa chỉ) đến phần tử kế tiếp.
  - Các phần tử không cần phải sắp xếp cạnh nhau trong bộ nhớ.
  - Việc truy xuất đến một phần tử này đòi hỏi phải thông qua một phần tử khác.
  - Tùy nhu cầu, các phần tử sẽ liên kết theo nhiều cách khác nhau tạo thành danh sách liên kết đơn, kép, vòng.

Danh sách liên kết

- Nhận xét
  - Số nút không cố định, thay đổi tùy nhu cầu nên đây là cấu trúc động.
  - Thích hợp thực hiện các thao tác chèn và hủy vì không cần phải dời nút mà chỉ cần sửa các liên kết cho phù hợp. Thời gian thực hiện không phụ thuộc vào số nút danh sách.
  - Tốn bộ nhớ chứa con trỏ liên kết pNext.
  - Truy xuất tuần tự nên mất thời gian.

Các loại danh sách liên kết

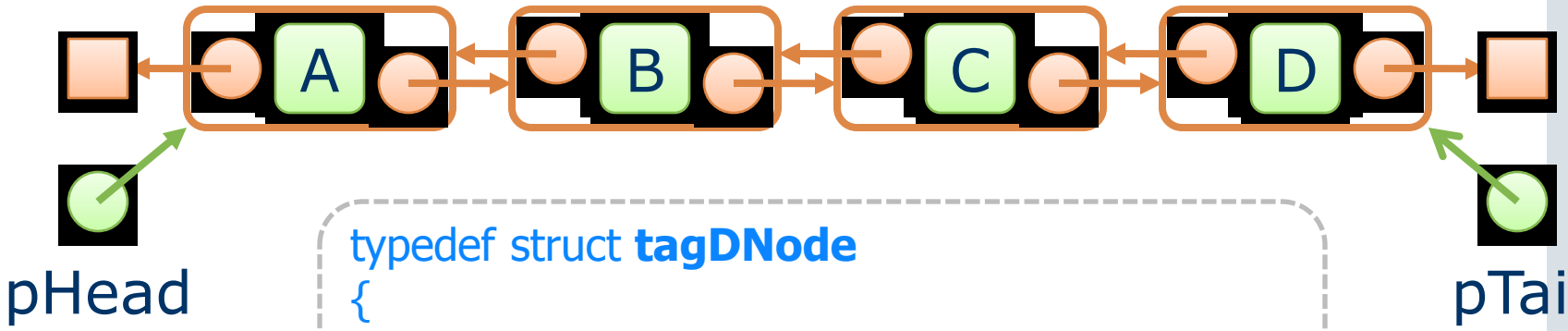
- Danh sách liên kết đơn




```
typedef struct tagNode
{
        Data Info;
        struct tagNode *pNext;
} NODE;
typedef struct tagList
{
        NODE *pHead;
        NODE *pTail;
} LIST;
```

Các loại danh sách liên kết

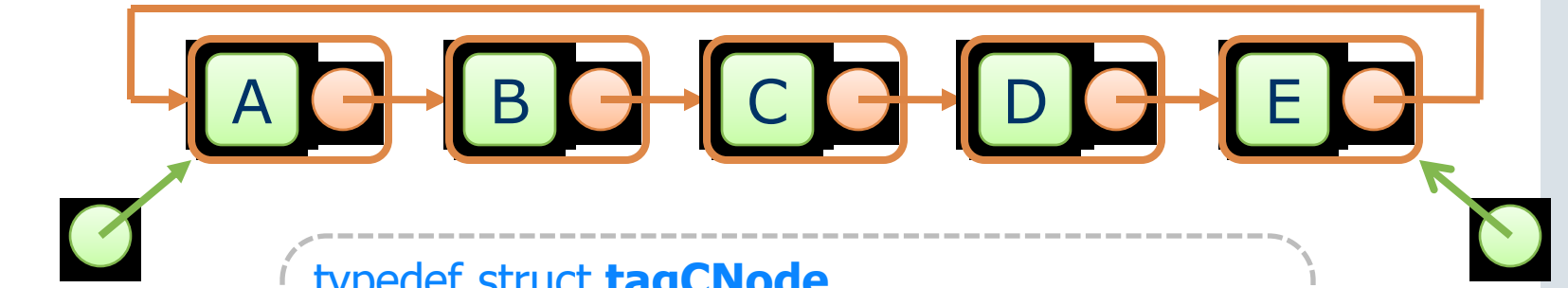
- Danh sách liên kết kép (Doubly Linked List)




```
typedef struct tagDNode
{
        Data Info;
        struct tagDNode *pNext, *pPrev;
} DNODE;
typedef struct tagDList
{
        NODE *pHead;
        NODE *pTail;
} DLIST;
```

Các loại danh sách liên kết

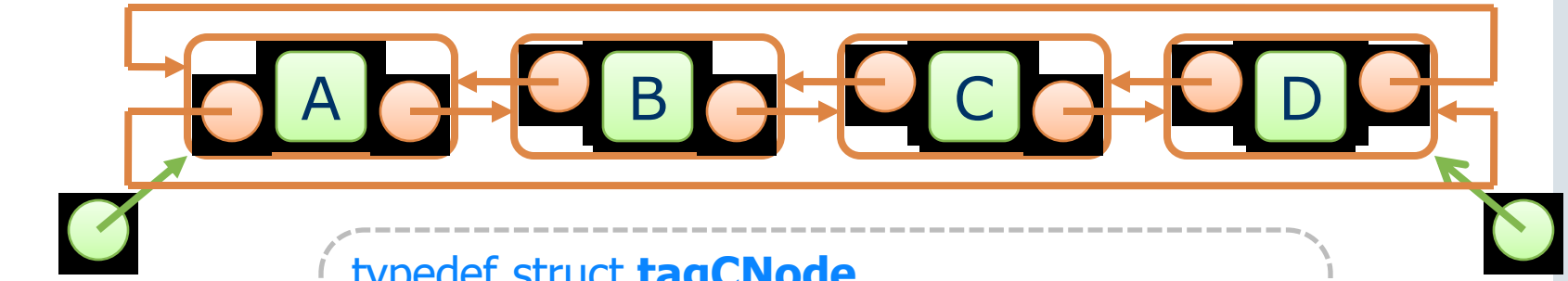
- Danh sách liên kết đơn vòng (Circular Linked List)




```
typedef struct tagCNode
{
        Data Info;
        struct tagCNode *pNext;
} CNODE;
typedef struct tagCList
{
        NODE *pHead;
        NODE *pTail;
} CLIST;
```

Các loại danh sách liên kết

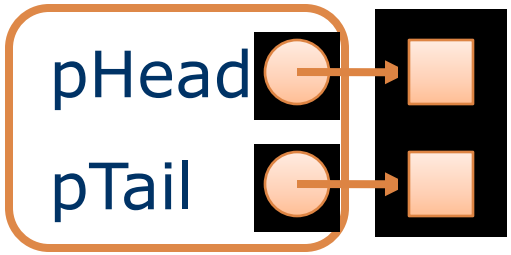
- Danh sách liên kết kép vòng (Circular Linked List)



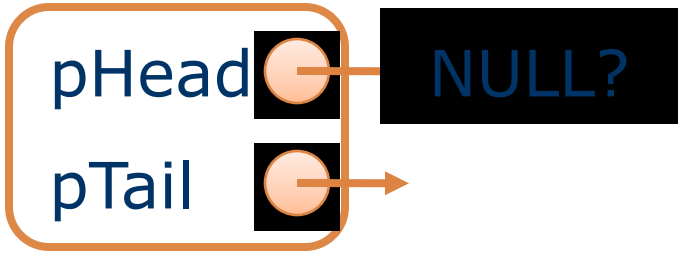

```
typedef struct tagCNode
{
        Data Info;
        struct tagCNode *pNext, *pPrev;
} CNODE;
typedef struct tagCList
{
        NODE *pHead;
        NODE *pTail;
} CLIST;
```

Danh sách liên kết đơn

- Khởi tạo danh sách



- Kiểm tra danh sách có rỗng hay không



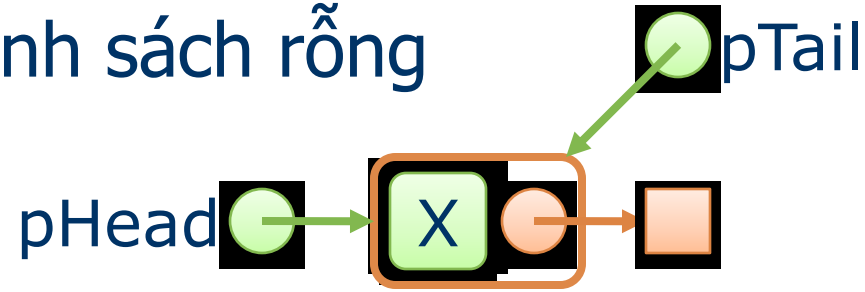
Danh sách liên kết đơn

- Tạo một nút mới

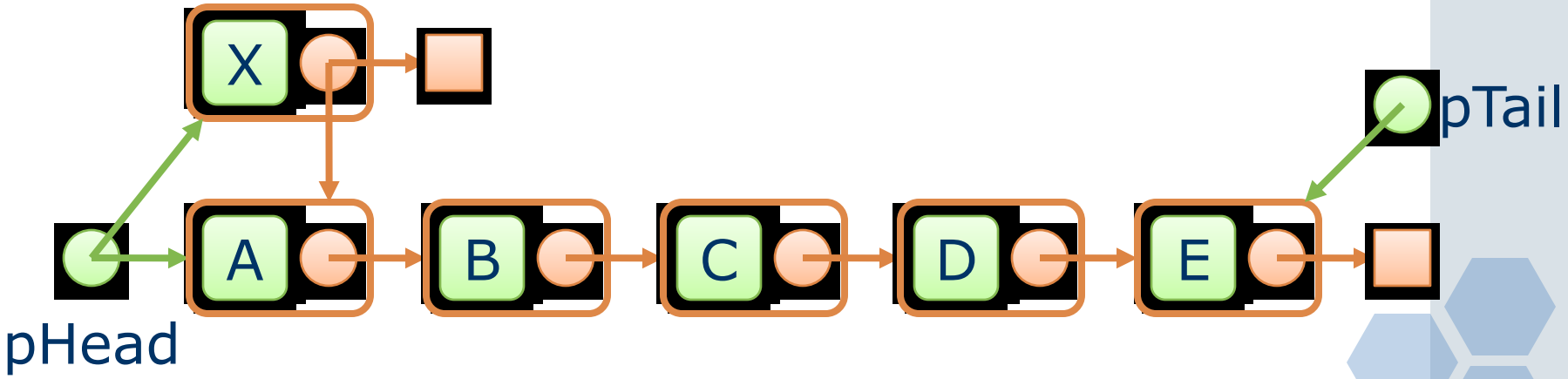
- Xác định con trỏ của nút thứ i trong danh sách
  - p = pHead
  - p = p->pNext i lần trong khi p != NULL rồi return lại con trỏ p hiện tại
- Xác định vị trí của nút p trong danh sách
  - Tương tự như trên nhưng trả lại vị trí

Danh sách liên kết đơn

- Chèn một nút vào đầu danh sách
  - Danh sách rỗng

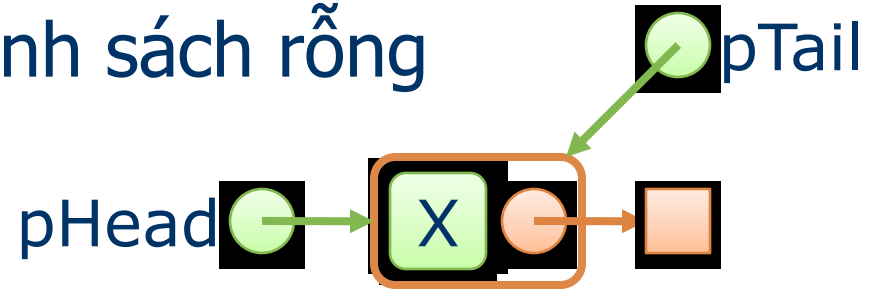


  - Danh sách không rỗng

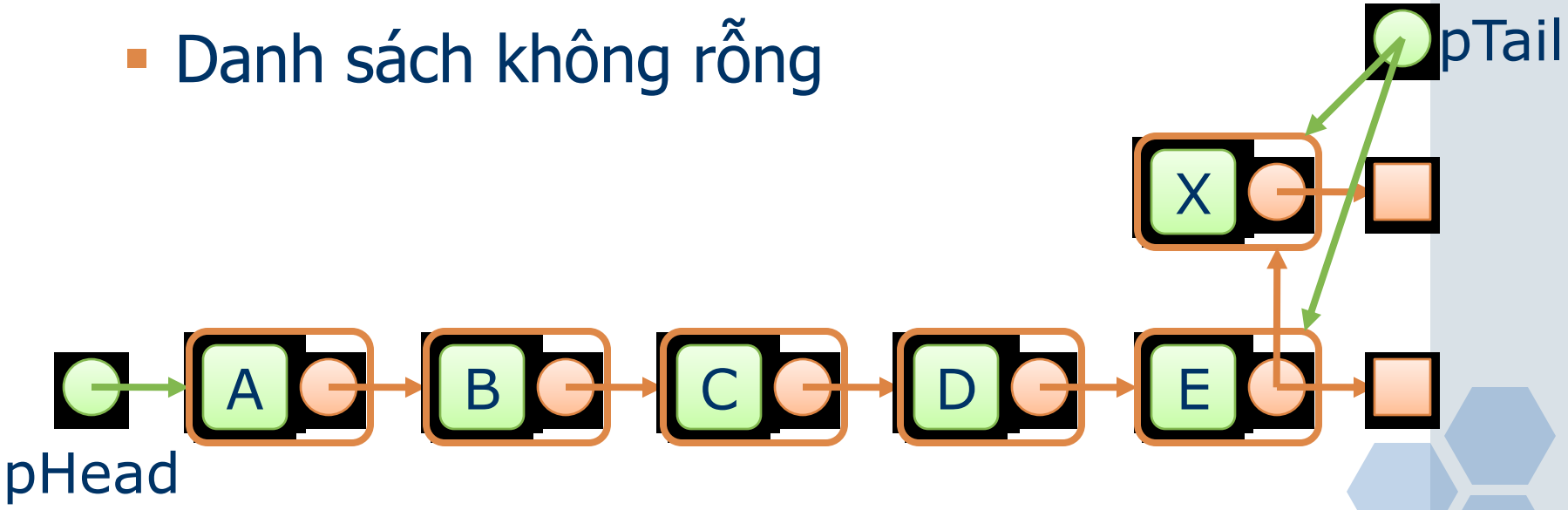


Danh sách liên kết đơn

- Thêm một nút vào cuối danh sách
  - Danh sách rỗng

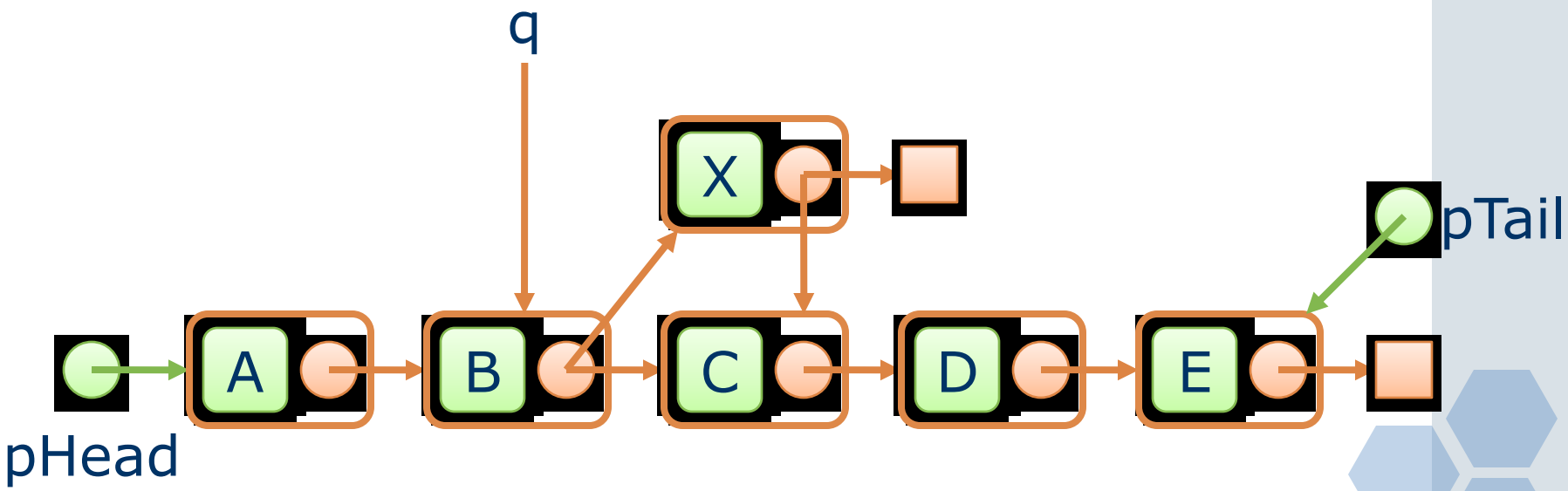


  - Danh sách không rỗng



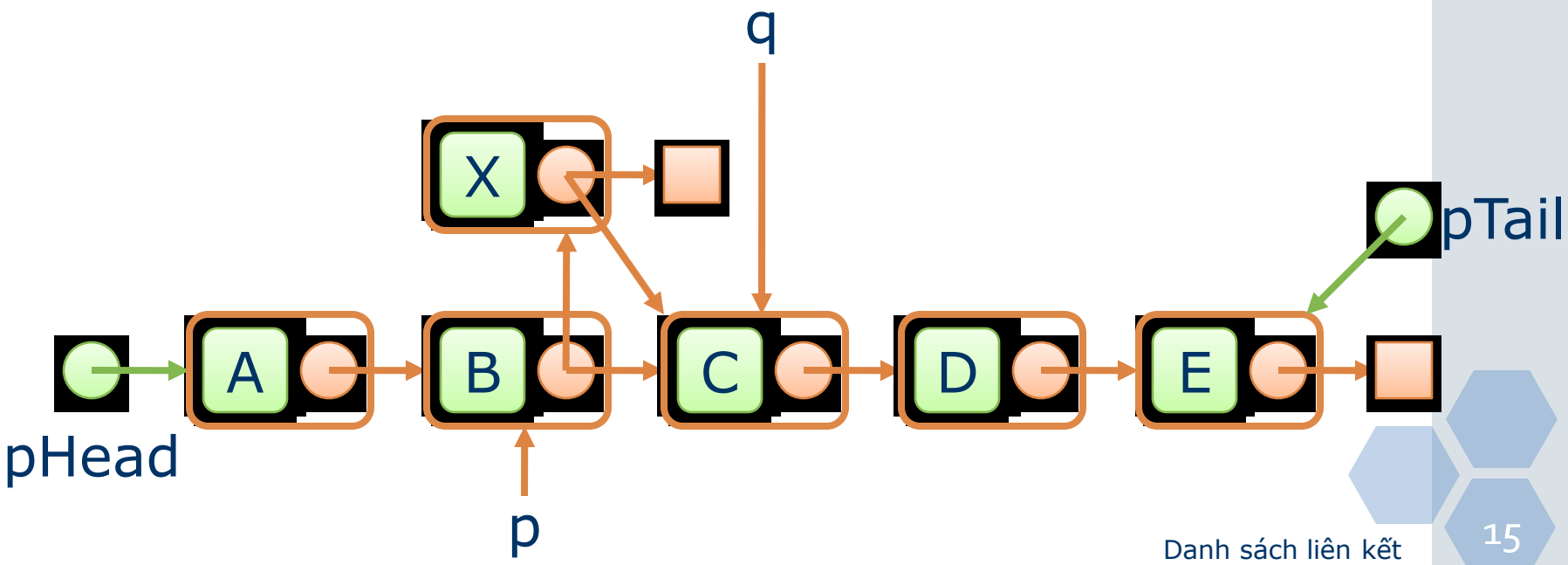
Danh sách liên kết đơn

- Thêm một nút vào sau nút q
  - q == NULL ➔ chèn vào đầu danh sách
  - q != NULL



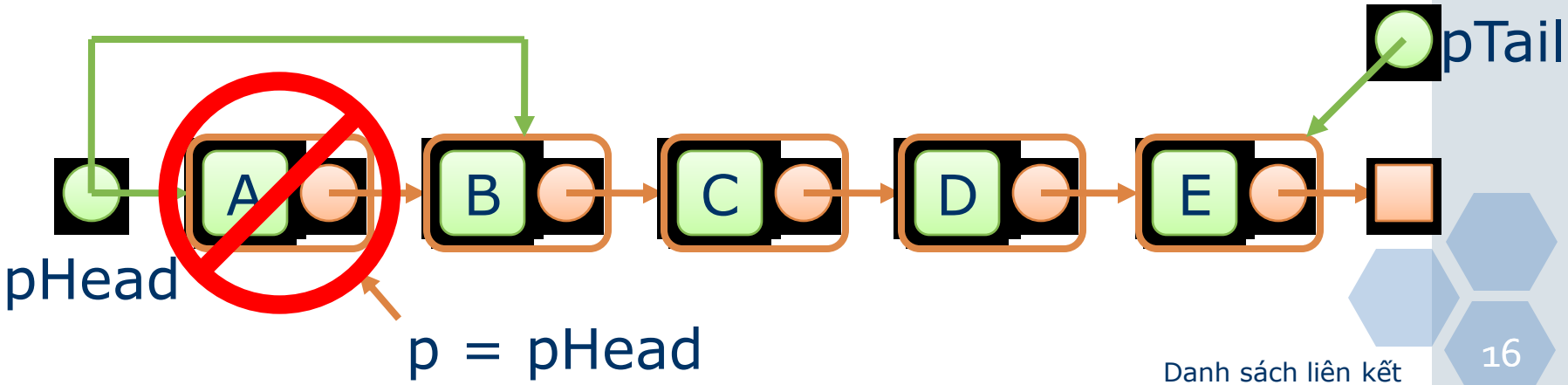
Danh sách liên kết đơn

- Thêm một nút vào trước nút q
  - q == NULL ➔ chèn vào đầu danh sách
  - q != NULL ➔ Tìm nút p trước q rồi thêm vào sau nút p này.



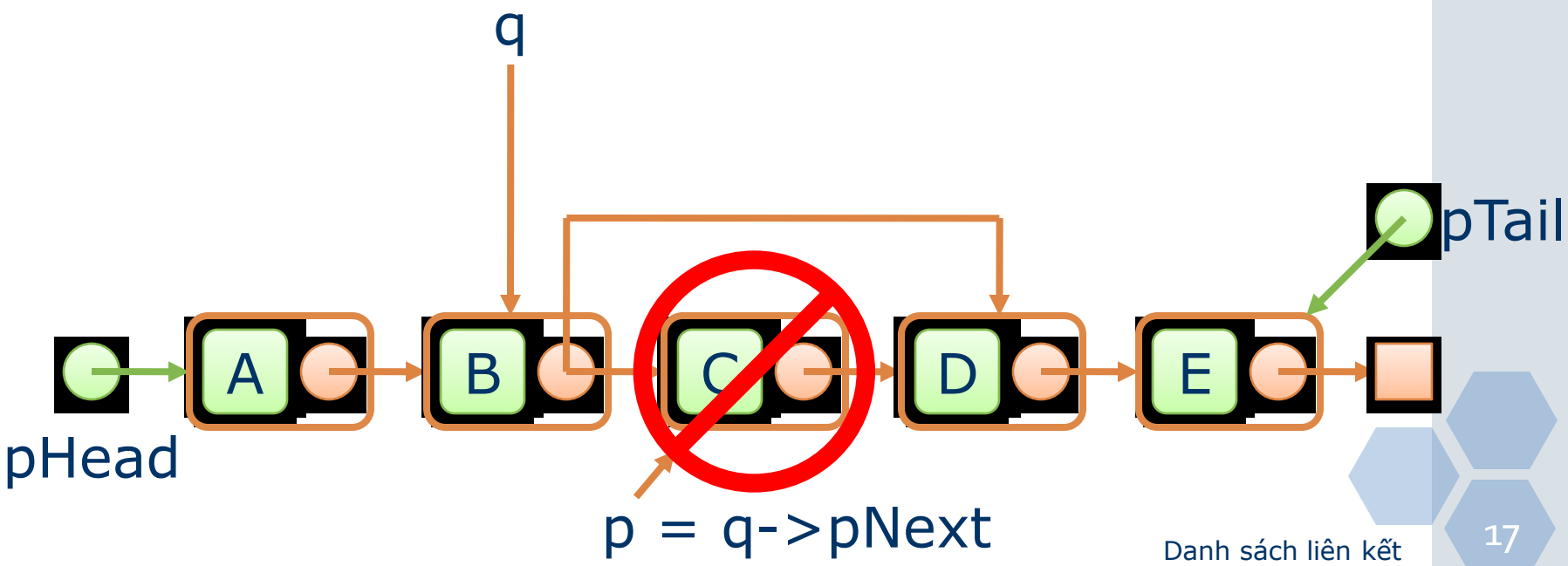
Danh sách liên kết đơn

- Hủy một nút đầu danh sách
  - Danh sách rỗng ➔ không làm gì cả
  - Danh sách không rỗng (nếu sau khi hủy mà pHead = NULL thì pTail = NULL)



Danh sách liên kết đơn

- Hủy một nút sau nút q
  - q == NULL ➔ hủy nút đầu danh sách
  - q != NULL



Danh sách liên kết đơn

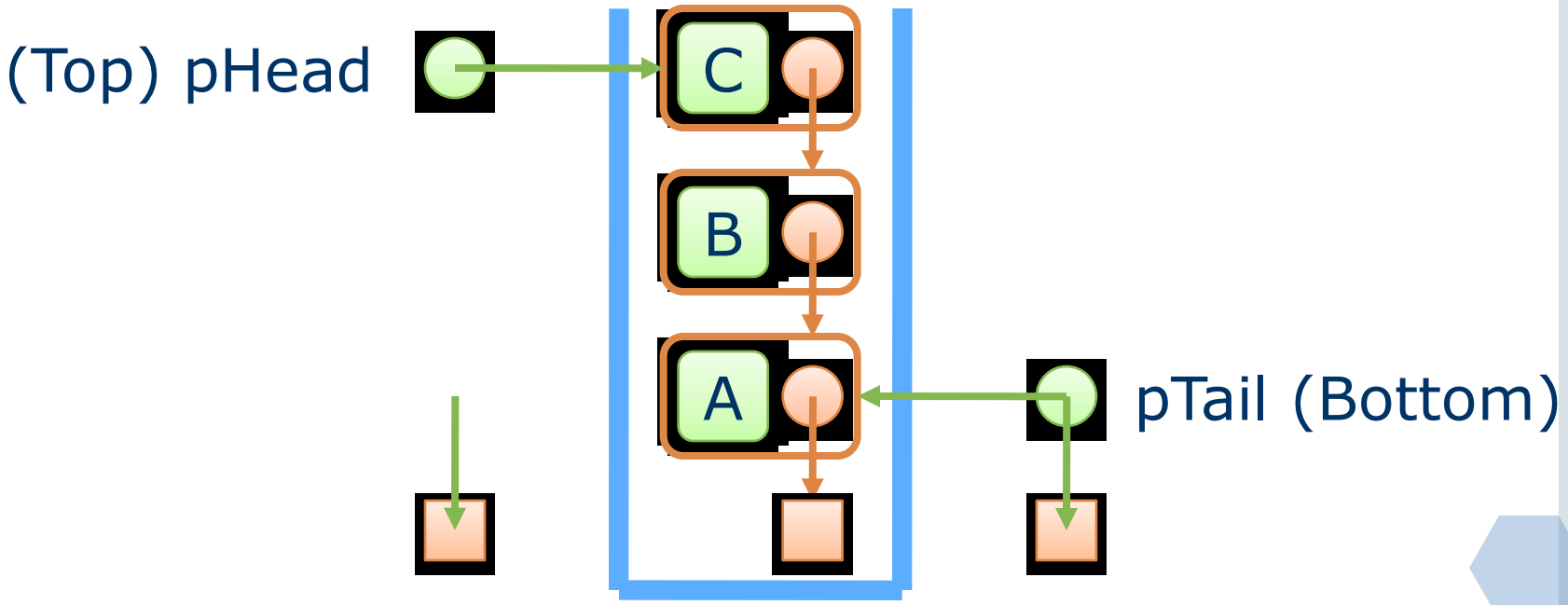
- Hủy một nút cuối danh sách
  - Tìm nút cuối p (có p->pNext == NULL)
  - Tìm nút q trước nút p (nếu có)
  - Hủy nút sau nút q
- Hủy một nút có khóa k (Info = k)
  - Tìm nút p có khóa k và hủy nút q trước đó.
  - Hủy nút sau nút q (nếu có)

Danh sách liên kết đơn

- Duyệt danh sách
  - Đếm/Tìm các phần tử của danh sách thỏa điều kiện
  - Hủy toàn bộ danh sách

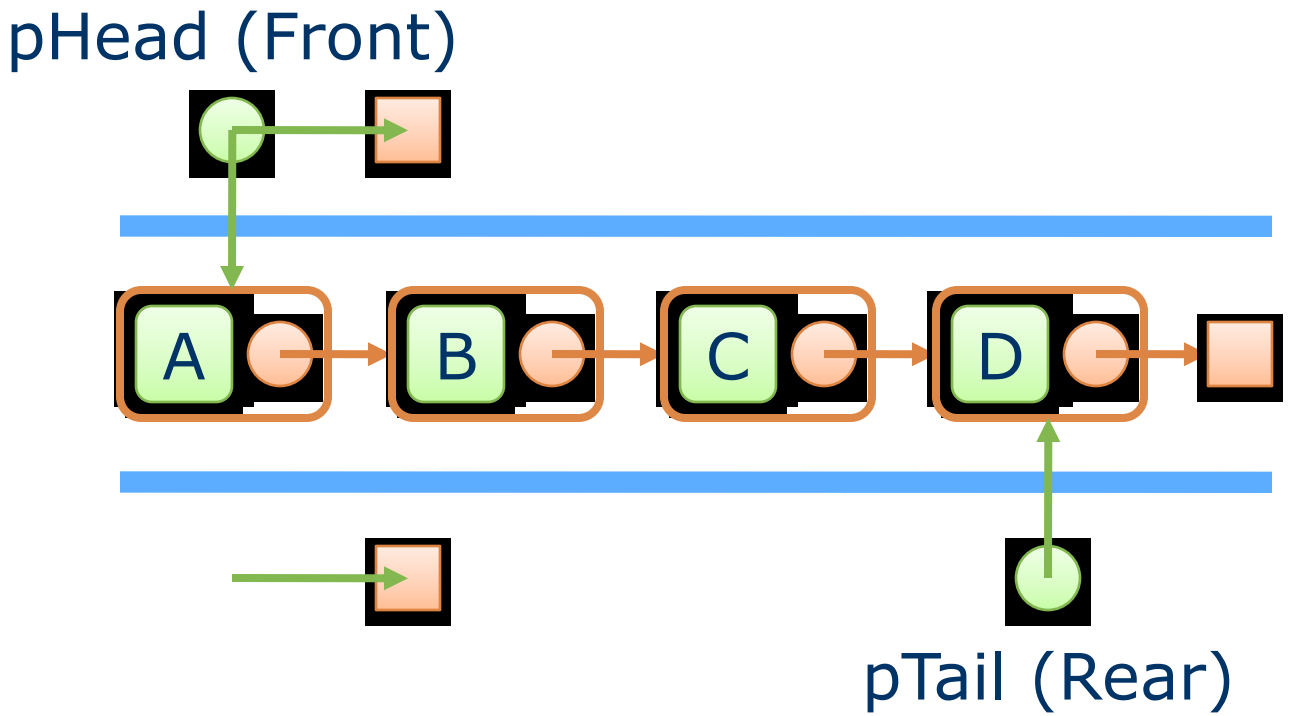
Ứng dụng của DSLK đơn

- Stack (Ngăn xếp)
  - Làm việc theo cơ chế LIFO (Last In First Out)



Ứng dụng của DSLK đơn

- Queue (Hàng đợi)
  - Làm việc theo cơ chế FIFO (First In First Out)



Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CHUỖI KÝ TỰ

Nội dung

1. Khái niệm
2. Khởi tạo
3. Các thao tác trên chuỗi ký tự
4. Bài tập

Khái niệm

Khái niệm

- Kiểu char chỉ chứa được một ký tự. Để lưu trữ một chuỗi (nhiều ký tự) ta sử dụng mảng (một chiều) các ký tự.
- Chuỗi ký tự kết thúc bằng ký tự '\0' (null)

➔ Độ dài chuỗi = kích thước mảng – 1

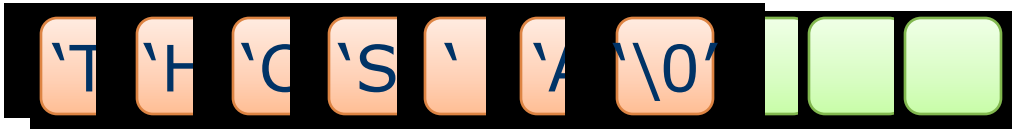
Ví dụ

```
char hoten[30];   // Dài 29 ký tự
char ngaysinh[9]; // Dài 8 ký tự
```

Khởi tạo

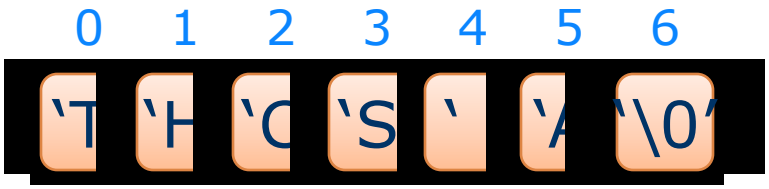
- Khởi tạo như mảng thông thường
  - Độ dài cụ thể

```
char s[10] = {'T', 'H', 'C', 'S', ' ', 'A', '\0'};
char s[10] = "THCS A";  // Tự động thêm '\0'
```



  - Tự xác định độ dài

```
char s[] = {'T', 'H', 'C', 'S', ' ', 'A', '\0'};
char s[] = "THCS A";    // Tự động thêm '\0'
```



Xuất chuỗi

- Sử dụng hàm printf với đặc tả "%s"

```
char monhoc[50] = "Tin hoc co so A";
printf("%s", monhoc);    // Không xuống dòng
```

```
Tin hoc co so A_
```

- Sử dụng hàm puts

```
char monhoc[50] = "Tin hoc co so A";
puts(monhoc);       // Tự động xuống dòng
⇔ printf("%s\n", monhoc);
```

```
Tin hoc co so A
_
```

Nhập chuỗi

- Sử dụng hàm scanf với đặc tả "%s"
  - Chỉ nhận các ký tự từ bàn phím đến khi gặp ký tự khoảng trắng hoặc ký tự xuống dòng.
  - Chuỗi nhận được không bao gồm ký tự khoảng trắng và xuống dòng.

```
char monhoc[50];
printf("Nhap mot chuoi: ");
scanf("%s", monhoc);
printf("Chuoi nhan duoc la: %s", monhoc);
```

```
Nhap mot chuoi: Tin hoc co so A
Chuoi nhan duoc la: Tin_
```

Nhập chuỗi

- Sử dụng hàm gets
  - Nhận các ký tự từ bàn phím đến khi gặp ký tự xuống dòng.
  - Chuỗi nhận được là những gì người dùng nhập (trừ ký tự xuống dòng).

```
char monhoc[50];
printf("Nhap mot chuoi: ");
gets(monhoc);
printf("Chuoi nhan duoc la: %s", monhoc);
```

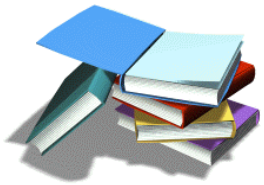
```
Nhap mot chuoi: Tin hoc co so A
Chuoi nhan duoc la: Tin hoc co so A_
```

Một số hàm thao tác trên chuỗi

- Thuộc thư viện <string.h>
  - strlen
  - strcpy
  - strdup
  - strlwr/strupr
  - strrev
  - strcmp/stricmp
  - strcat
  - strstr

Hàm tính độ dài chuỗi

size_t strlen(const char *s)



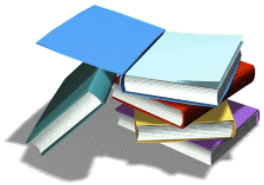
Tính độ dài chuỗi s.
size_t thay cho unsigned (trong <stddef.h>)
dùng để đo các đại lượng không dấu.

- Độ dài chuỗi s (không tính ký tự kết thúc)

```
char s[] = "Visual C++ 6.0";
int len = strlen(s);        // => 14
```

Hàm sao chép chuỗi

char *strcpy(char *dest, const char *src)



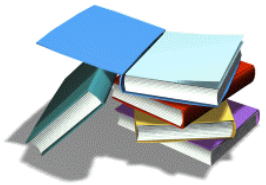
Sao chép chuỗi src sang chuỗi dest, dừng khi ký tự kết thúc chuỗi '\0' vừa được chép.
! dest phải đủ lớn để chứa src

- Con trỏ dest.

```
char s[100];
s = "Visual C++ 6.0";          // sai
strcpy(s, "Visual C++ 6.0");   // đúng
```

Hàm tạo bản sao

char *strdup(const char *s)



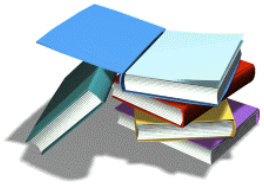
Tạo bản sao của một chuỗi s cho trước. Hàm sẽ tự tạo vùng nhớ dài strlen(s) + 1 (bytes) để chứa chuỗi s. Phải tự hủy vùng nhớ này khi không sử dụng nữa.

- Thành công: trả về con trỏ đến vùng nhớ chứa chuỗi bản sao.
- Thất bại: trả về NULL.

```
char *s;
s = strdup("Visual C++ 6.0");
```

Hàm chuyển thành chuỗi thường

char *strlwr(char *s)



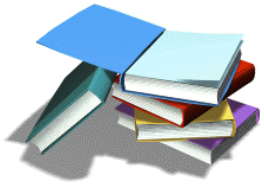
Chuyển chuỗi s thành chuỗi thường ('A' thành 'a', 'B' thành 'b', ..., 'Z' thành 'z')

- Con trỏ đến chuỗi s.

```
char s[] = "Visual C++ 6.0";
strlwr(s);
puts(s);            // visual c++ 6.0
```

Hàm chuyển thành chuỗi IN

char *strupr(char *s)



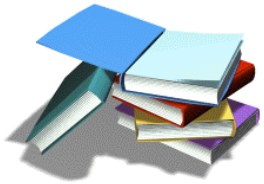
Chuyển chuỗi s thành chuỗi IN ('a' thành 'A', 'b' thành 'B', ..., 'z' thành 'Z')

- Con trỏ đến chuỗi s.

```
char s[] = "Visual C++ 6.0";
strupr(s);
puts(s);            // VISUAL C++ 6.0
```

Hàm đảo ngược chuỗi

char *strrev(char *s)



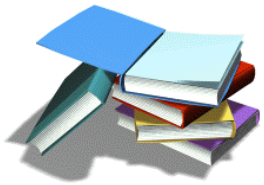
Đảo ngược thứ tự các ký tự trong chuỗi s (trừ ký tự kết thúc chuỗi).

- Con trỏ đến chuỗi kết quả.

```
char s[] = "Visual C++ 6.0";
strrev(s);
puts(s);          // 0.6 ++C lausiV
```

Hàm so sánh hai chuỗi

int strcmp(const char *s1, const char *s2)



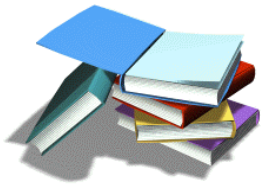
So sánh hai chuỗi s1 và s2 (phân biệt hoa thường).

- < 0 nếu s1 < s2
- == 0 nếu s1 == s2
- >0 nếu s1 > s2

```
char s1[] = "visual C++ 6.0";
char s2[] = "Visual C++ 6.0";
int kq = strcmp(s1, s2); // => kq > 0
```

Hàm so sánh hai chuỗi

int stricmp(const char *s1, const char *s2)



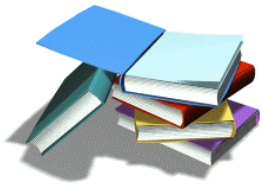
So sánh hai chuỗi s1 và s2 (không phân biệt hoa thường).

- < 0 nếu s1 < s2
- == 0 nếu s1 == s2
- >0 nếu s1 > s2

```
char s1[] = "visual c++ 6.0";
char s2[] = "VISUAL C++ 6.0";
int kq = stricmp(s1, s2);// => kq == 0
```

Hàm nối hai chuỗi

char* strcat(char *dest, const char *src)



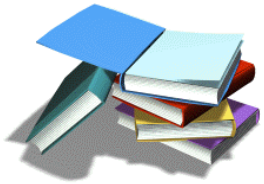
Nối chuỗi src vào sau chuỗi dest.
! Chuỗi dest phải đủ chứa kết quả

- Con trỏ đến chuỗi được nối.

```
char s1[100] = "Visual C++";
char s2[] = "6.0";
strcat(s1, " ");   // => "Visual C++ "
strcat(s1, s2);    // => "Visual C++ 6.0"
```

Hàm tìm chuỗi trong chuỗi

char* strstr(const char *s1, const char *s2)



Tìm vị trí xuất hiện đầu tiên của s2 trong s1

- Thành công: trả về con trỏ đến vị trí xuất hiện đầu tiên của s2 trong s1.
- Thất bại: trả về null.

```
char s1[] = "Visual C++ 6.0";
char s2[] = "C++";
if (strstr(s1, s2) != null)
        printf("Tim thay s2 trong s1...");
```

Bài tập

- Bài 1: Xem thêm một số hàm khác như:
  - atoi, atol, atof : đổi chuỗi thành số.
  - itoa, ltoa, ultoa: đổi số thành chuỗi.
  - strtok
- Bài 2: Viết hàm nhận vào một chuỗi và trả về chuỗi tương ứng (giữ nguyên chuỗi đầu vào):
  - Các ký tự thành ký tự thường (giống strlwr).
  - Các ký tự thành ký tự hoa (giống strupr).
  - Các ký tự đầu tiên thành ký tự hoa.
  - Chuẩn hóa chuỗi (xóa khoảng trắng thừa).

Bài tập

- Bài 3: Viết hàm nhận vào một chuỗi s và trả về chuỗi tương ứng sau khi xóa các khoảng trắng.
- Bài 4: Viết hàm nhận vào một chuỗi s và đếm xem có bao nhiêu từ trong chuỗi đó.
- Bài 5: Viết hàm nhận vào một chuỗi s và xuất các từ trên các dòng liên tiếp.
- Bài 6: Viết hàm tìm từ có chiều dài lớn nhất và xuất ra màn hình từ đó và độ dài tương ứng.
- Bài 7: Viết hàm trích ra n ký tự đầu tiên/cuối cùng/bắt đầu tại vị trí pos của chuỗi s cho trước.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DỮ LIỆU KIỂU CẤU TRÚC

Nội dung

1. Khái niệm kiểu cấu trúc (struct)
2. Khai báo & truy xuất kiểu cấu trúc
3. Kiểu dữ liệu hợp nhất (union)
4. Bài tập

Đặt vấn đề

- Thông tin 1 SV
  - MSSV : kiểu chuỗi
  - Tên SV : kiểu chuỗi
  - NTNS : kiểu chuỗi
  - Phái : kiểu ký tự
  - Điểm Toán, Lý, Hóa : kiểu số thực
- Yêu cầu
  - Lưu thông tin n SV?
  - Truyền thông tin n SV vào hàm?

Đặt vấn đề

- Khai báo các biến để lưu trữ 1 SV
  - char mssv[7]; // “0012078”
  - char hoten[30]; // “Nguyen Van A”
  - char ntns[8]; // “29/12/82”
  - char phai; // ‘n’
  - float toan, ly, hoa;// 8.5 9.0 10.0
- Truyền thông tin 1 SV cho hàm
  - void xuat(char *mssv, char *hoten, char *ntns, char phai, float toan, float ly, float hoa);

Đặt vấn đề

- Nhận xét
  - Đặt tên biến khó khăn và khó quản lý
  - Truyền tham số cho hàm quá nhiều
  - Tìm kiếm, sắp xếp, sao chép,… khó khăn
  - Tốn nhiều bộ nhớ
  - …
- Ý tưởng
  - Gom những thông tin của cùng 1 SV thành một kiểu dữ liệu mới => Kiểu struct

Khai báo kiểu cấu trúc

Cú pháp

```
struct <tên kiểu cấu trúc>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần n>;
};
```

Ví dụ

```
struct DIEM
{
        int x;
        int y;
};
```

Khai báo biến cấu trúc

- Cú pháp tường minh

```
struct <tên kiểu cấu trúc>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần n>;
} <tên biến 1>, <tên biến 2>;
```

- Ví dụ

```
struct DIEM
{
        int x;
        int y;
} diem1, diem2;
```

Khai báo biến cấu trúc

- Cú pháp không tường minh

```
struct <tên kiểu cấu trúc>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần n>;
};
struct <tên kiểu cấu trúc> <tên biến>;
```

- Ví dụ

```
struct DIEM
{
        int x;
        int y;
};
struct DIEM diem1, diem2;// C++ có thể bỏ struct
```

Sử dụng typedef

Cú pháp

```
typedef struct
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần n>;
} <tên kiểu cấu trúc>;
<tên kiểu cấu trúc> <tên biến>;
```

Ví dụ

```
typedef struct
{
        int x;
        int y;
} DIEM;
struct DIEM diem1, diem2;
```

Khởi tạo cho biến cấu trúc

Cú pháp tường minh

```
struct <tên kiểu cấu trúc>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần n>;
} <tên biến> = {<giá trị 1>,…,<giá trị n>};
```

Ví dụ

```
struct DIEM
{
        int x;
        int y;
} diem1 = {2912, 1706}, diem2;
```

Truy xuất dữ liệu kiểu cấu trúc

- Đặc điểm
  - Không thể truy xuất trực tiếp
  - Thông qua toán tử thành phần cấu trúc . hay còn gọi là toán tử chấm (dot operation)

```
<tên biến cấu trúc>.<tên thành phần>
```

- Ví dụ

```
struct DIEM
{
        int x;
        int y;
} diem1;
printf("x = %d, y = %d", diem1.x, diem1.y);
```

Gán dữ liệu kiểu cấu trúc

- Có 2 cách

```
<biến cấu trúc đích> = <biến cấu trúc nguồn>;

<biến cấu trúc đích>.<tên thành phần> = <giá trị>;
```

- Ví dụ

```
struct DIEM
{
        int x, y;
} diem1 = {2912, 1706}, diem2;
…
diem2 = diem1;
diem2.x = diem1.x;
diem2.y = diem1.y * 2;
```

Cấu trúc phức tạp

- Thành phần của cấu trúc là cấu trúc khác

```
struct DIEM
{
        int x;
        int y;
};

struct HINHCHUNHAT
{
        struct DIEM traitren;
        struct DIEM phaiduoi;
} hcn1;
…
hcn1.traitren.x = 2912;
hcn1.traitren.y = 1706;
```

Cấu trúc phức tạp

- Thành phần của cấu trúc là mảng

```
struct SINHVIEN
{
        char hoten[30];
        float toan, ly, hoa;
} sv1;
…
strcpy(sv1.hoten, “Nguyen Van A”);
sv1.toan = 10;
sv1.ly = 6.5;
sv1.hoa = 9;
```

Cấu trúc phức tạp

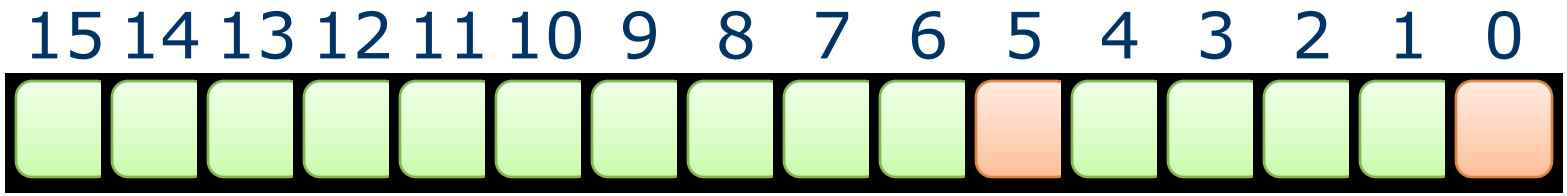
Cấu trúc đệ quy (tự trỏ)

```
struct PERSON
{
        char hoten[30];
        struct PERSON *father, *mother;
};

struct NODE
{
        int value;
        struct NODE *pNext;
};
```

Cấu trúc phức tạp

- Thành phần của cấu trúc có kích thước theo bit

```
struct bit_fields
{
        int bit_0 : 1;
        int bit_1_to_4 : 4;
        int bit_5 : 1;
        int bit_6_to_15 : 10;
};
```



Kích thước của struct

- Ví dụ

```
struct A
{
        int a;
        double b;
};
sizeof(A) = ???
```

```
struct B1
{
        int a;
        int b;
        double c;
};
sizeof(B1) = ???
```

```
struct B2
{
        int a;
        double c;
        int b;
};
sizeof(B2) = ???
```

Chỉ thị #pragma pack

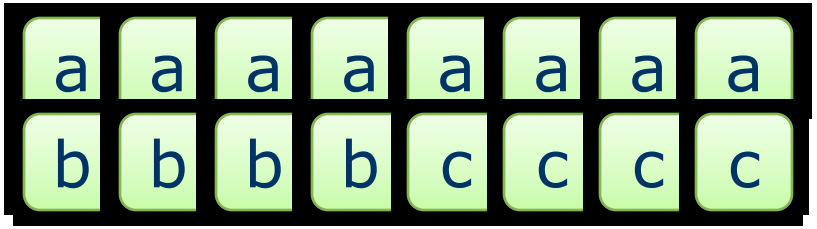
- Chỉ thị #pragma pack (n)
  - n = 1, 2, 4, 8, 16 (byte)
  - Biên lớn nhất của các thành phần trong struct
    - BC n mặc định là 1
    - VC++ n mặc định là 8
    - Project settings → Compile Option C/C++ → Code Generation → Structure Alignment
  - Canh biên cho 1 cấu trúc

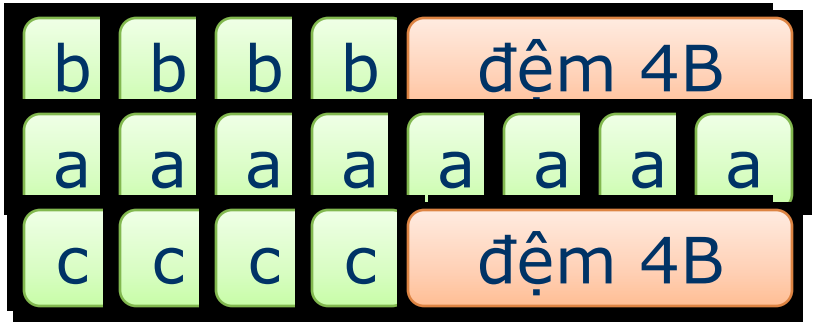
```
#pragma pack(push, 1)
struct MYSTRUCT { … };
#pragma pack(pop)
```

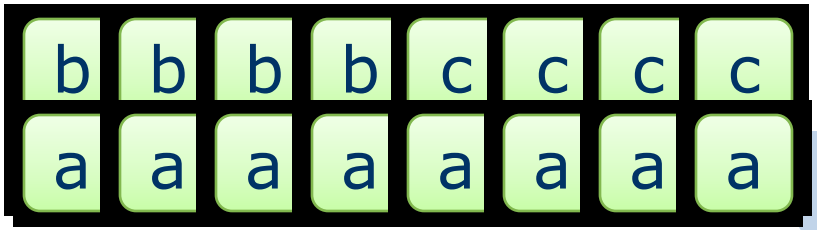
#pragma pack

- Ví dụ: không có #pragma pack (1)

```
struct A {
        double a;
        int b;
        int c;
};
struct B {
        int b;
        double a;
        int c;
};
struct C {
        int b;
        int c;
        double a;
};
```







Các lưu ý về cấu trúc

- Lưu ý
  - Kiểu cấu trúc được định nghĩa để làm khuôn dạng còn biến cấu trúc được khai báo để sử dụng khuôn dạng đã định nghĩa.
  - Trong C++, có thể bỏ từ khóa struct khi khai báo biến (hoặc sử dụng typedef)
  - Khi nhập các biến kiểu số thực trong cấu trúc phải nhập thông qua một biến trung gian.

```
struct DIEM { float x, y;} d1;
float temp; scanf("%f", &temp); d1.x = temp;
```

Mảng cấu trúc

Mảng cấu trúc

- Tương tự như mảng với kiểu dữ liệu cơ sở (char, int, float, ...)

```
struct DIEM
{
        int x;
        int y;
};

DIEM mang1[20];
DIEM mang2[10] = {{3, 2}, {4, 4}, {2, 7}};
```

Truyền cấu trúc cho hàm

- Truyền cấu trúc cho hàm
  - Giống như truyền kiểu dữ liệu cơ sở
    - Tham trị (không thay đổi sau khi kết thúc hàm)
    - Tham chiếu
    - Con trỏ
  - Ví dụ

```
struct DIEM { int x, y; };

void xuat1(int x, int y) { … };
void xuat2(DIEM diem) { … };
void xuat3(DIEM &diem) { … };
void xuat4(DIEM *diem) { … };
```

Hợp nhất – union

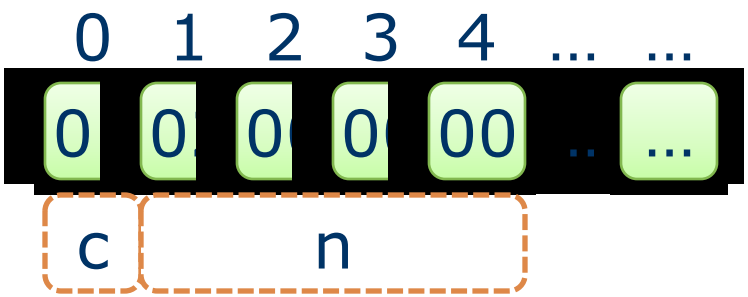
- Khái niệm
  - Được khai báo và sử dụng như cấu trúc
  - Các thành phần của union có chung địa chỉ đầu (nằm chồng lên nhau trong bộ nhớ)
- Khai báo

```
union <tên kiểu union>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần 2>;
};
```

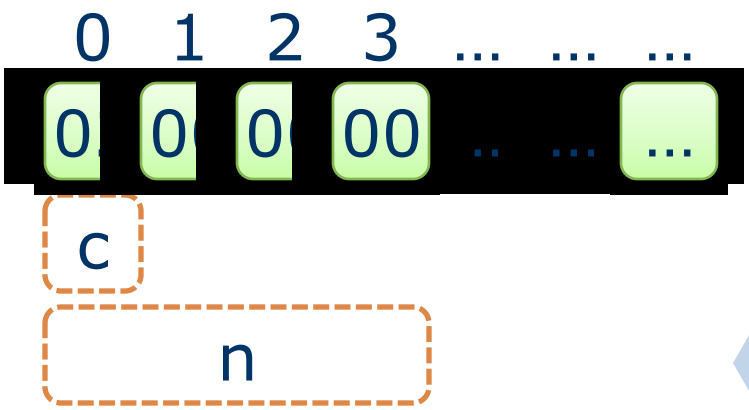
So sánh struct và union

Ví dụ

```
struct MYSTRUCT
{
        char c;
        int n;
} s;

s.c = 1; s.n = 2;
```




```
union MYUNION
{
        char c;
        int n;
} u;

u.c = 1; u.n = 2;
```



Ví dụ

- struct trong union

```
union date_tag
{
        char full_date[9];
        struct
        {
                char month[2];
                char break_value1;
                char day[2];
                char break_value2;
                char year[2];
        } part_date_tag;
} date = {“29/12/82”};
```

Ví dụ

- union trong struct

```
struct generic_tag
{
        char type;
        union
        {
                char c;
                int i;
                float f;
        } share_tag;
};
```

Bài tập

- Phân số
  - Khai báo kiểu dữ liệu phân số (PHANSO)
  - Nhập/Xuất phân số
  - Rút gọn phân số
  - Tính tổng, hiệu, tích, thương hai phân số
  - Kiểm tra phân số tối giản
  - Quy đồng hai phân số
  - Kiểm tra phân số âm hay dương
  - So sánh hai phân số

Bài tập

- Đơn thức
  - Khai báo kiểu dữ liệu đơn thức (DONTHUC)
  - Nhập/Xuất đơn thức
  - Tính tích, thương hai đơn thức
  - Tính đạo hàm cấp 1 của đơn thức
  - Tính đạo hàm cấp k của đơn thức
  - Tính giá trị đơn thức tại $x = x_0$

Bài tập

- Đa thức
  - Khai báo kiểu dữ liệu đa thức (DATHUC)
  - Nhập/Xuất đa thức
  - Tính tổng, hiệu, tích hai đa thức
  - Tính đạo hàm cấp 1 của đa thức
  - Tính đạo hàm cấp k của đa thức
  - Tính giá trị đơn thức tại $x = x_0$

Bài tập

- Điểm trong mặt phẳng Oxy
  - Khai báo kiểu dữ liệu điểm (DIEM)
  - Nhập/Xuất tọa độ điểm
  - Tính khoảng cách giữa hai điểm
  - Tìm điểm đối xứng qua gốc toạ độ/trục Ox/Oy
  - Kiểm tra điểm thuộc phần tư nào?
- Tam giác
  - Khai báo kiểu dữ liệu tam giác (TAMGIAC)
  - Nhập/Xuất tam giác
  - Tính chu vi, diện tích tam giác

Bài tập

- Ngày
  - Khai báo kiểu dữ liệu ngày (NGAY)
  - Nhập/Xuất ngày (ngày, tháng, năm)
  - Kiểm tra năm nhuận
  - Tính số thứ tự ngày trong năm
  - Tính số thứ tự ngày kể từ ngày 1/1/1
  - Tìm ngày trước đó, sau đó k ngày
  - Tính khoảng cách giữa hai ngày
  - So sánh hai ngày

Bài tập

- Mảng phân số
  - Nhập/Xuất n phân số
  - Rút gọn mọi phân số
  - Đếm số lượng phân số âm/dương trong mảng
  - Tìm phân số dương đầu tiên trong mảng
  - Tìm phân số nhỏ nhất/lớn nhất trong mảng
  - Sắp xếp mảng tăng dần/giảm dần

Bài tập

- Mảng điểm
  - Nhập/Xuất n điểm
  - Đếm số lượng điểm có hoành độ dương
  - Đếm số lượng điểm không trùng với các điểm khác trong mảng
  - Tìm điểm có hoành độ lớn nhất/nhỏ nhất
  - Tìm điểm gần gốc tọa độ nhất

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KỸ THUẬT THAO TÁC TRÊN BIT

Nội dung

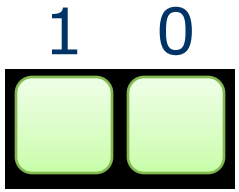
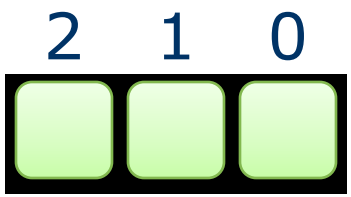
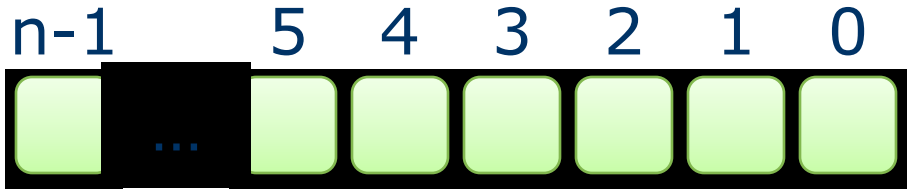
1. Các toán tử logic
2. Các toán tử dịch bit
3. Các ứng dụng
4. Bài tập

Đơn vị đo thông tin

- Hai trạng thái tắt-0 và mở-1 (nhị phân).
- Ký số nhị phân (Binary Digit) – bit
- bit - Đơn vị chứa thông tin nhỏ nhất.
- Các đơn vị đo thông tin lớn hơn:

Tên gọi	Ký hiệu	Giá trị
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Byte
MegaByte	MB	2^{10} KB = 2^{20} Byte
GigaByte	GB	2^{10} MB = 2^{30} Byte
TeraByte	TB	2^{10} GB = 2^{40} Byte
PentaByte	PB	2^{10} TB = 2^{50} Byte

Đơn vị đo thông tin

1 bit		2
2 bit		2^2
3 bit		2^3
n bit		2^n

0…000 → 1…111 = $2^n - 1$

Biểu diễn thông tin trong MTĐT

- Đặc điểm
  - Được lưu trong các thanh ghi hoặc trong các ô nhớ. Thanh ghi hoặc ô nhớ có kích thước 1 byte (8 bit) hoặc 1 word (16 bit).
  - Biểu diễn số nguyên không dấu, số nguyên có dấu, số thực và ký tự.
- Hai loại bit đặc biệt
  - msb (most significant bit): bit nặng nhất (bit n)
  - lsb (least significant bit): bit nhẹ nhất (bit 0)

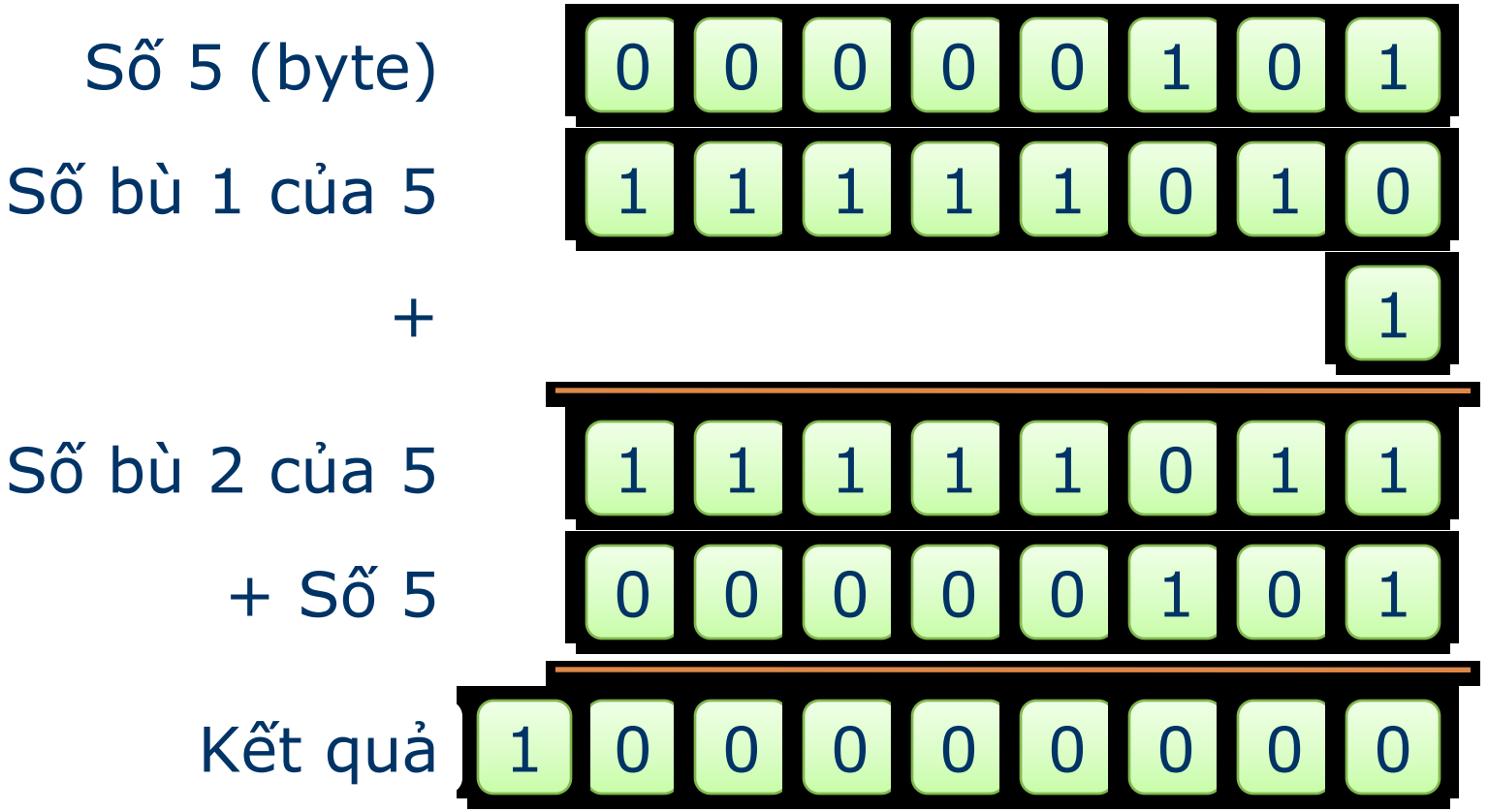
Biểu diễn số nguyên không dấu

- Đặc điểm
  - Biểu diễn các đại lương luôn dương.
  - Ví dụ: chiều cao, cân nặng, mã ASCII…
  - Tất cả bit được sử dụng để biểu diễn giá trị.
  - Số nguyên không dấu 1 byte lớn nhất là $1111\ 1111_2 = 2^8 - 1 = 255_{10}$.
  - Số nguyên không dấu 1 word lớn nhất là $1111\ 1111\ 1111\ 1111_2 = 2^{16} - 1 = 65535_{10}$.
  - Tùy nhu cầu có thể sử dụng số 2, 3… word.
  - lsb = 1 thì số đó là số đó là số lẻ.

Biểu diễn số nguyên có dấu

- Đặc điểm
  - Lưu các số dương hoặc âm.
  - Bit msb dùng để biểu diễn dấu
    - msb = 0 biểu diễn số dương. VD: 0101 0011
    - msb = 1 biểu diễn số âm. VD: 1101 0011
  - Trong máy tính, số âm được biểu diễn ở dạng số bù 2.

Số bù 1 và số bù 2



Biểu diễn số nguyên có dấu

- Nhận xét
  - Số bù 2 của x cộng với x là một dãy toàn bit 0 (không tính bit 1 cao nhất do vượt quá phạm vi lưu trữ). Do đó số bù 2 của x chính là giá trị âm của x hay – x.
  - Đổi số thập phân âm –5 sang nhị phân?
    - ➔ Đổi 5 sang nhị phân rồi lấy số bù 2 của nó.
  - Thực hiện phép toán a – b?
    - ➔ a – b = a + (–b) => Cộng với số bù 2 của b.

Tính giá trị có dấu và không dấu

- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Số nguyên không dấu ?
    - Tất cả 16 bit lưu giá trị.

      => giá trị là 52464.
  - Số nguyên có dấu ?
    - Bit msb = 1 do đó số này là số âm.

      => độ lớn là giá trị của số bù 2.
    - Số bù 2 = 0011 0011 0001 0000 = 13072.

      => giá trị là –13072.

Tính giá trị có dấu và không dấu

- Bảng giá trị số không dấu/có dấu (byte & word)

msb = 0

msb = 1

HEX	Không dấu	Có dấu
00	0	0
01	1	1
02	2	2
...	...	...
...	...	...
7E	126	126
7F	127	127
80	128	–128
81	129	–127
...	...	...
...	...	...
FE	254	–2
FF	255	–1

HEX	Không dấu	Có dấu
0000	0	0
0001	1	1
0002	2	2
...	...	...
...	...	...
7FFE	32766	32766
7FFF	32767	32767
8000	32768	–32768
8001	32769	–32767
...	...	...
...	...	...
FFFE	65534	–2
FFFF	65535	–1

Tính giá trị có dấu và không dấu

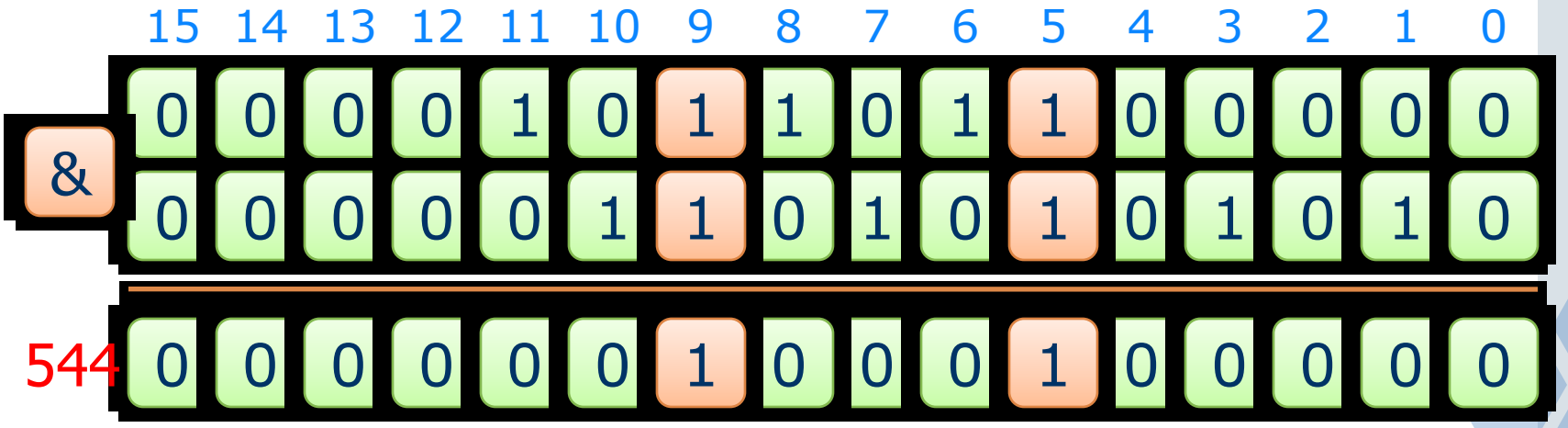
- Nhận xét
  - msb=0 ➔ giá trị có dấu bằng giá trị không dấu.
  - msb=1 ➔ thì giá trị có dấu bằng giá trị không dấu trừ 2^8=256 (byte) hay 2^{16}=65536 (word).
- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Giá trị không dấu là 52464.
  - Giá trị có dấu: vì bit msb = 1 nên giá trị có dấu bằng 52464 – 65536 = –13072.

Các toán tử trên bit

- Toán tử & (and)

&	0	1
0	0	0
1	0	**1**

- Ví dụ
  - int x = 2912, y = 1706, z = x & y;

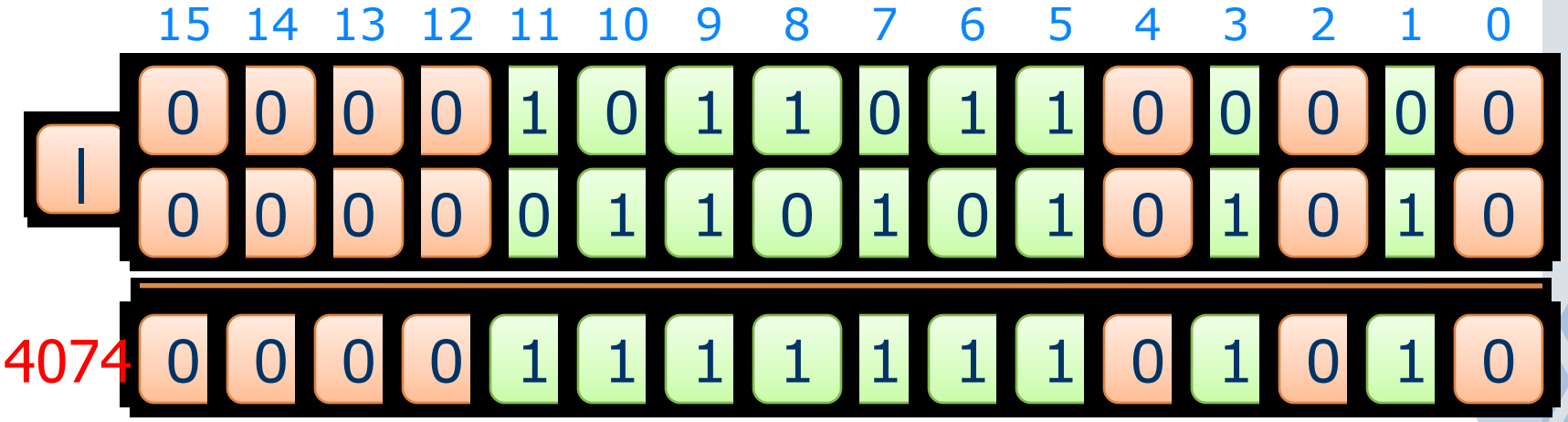


Các toán tử trên bit

- Toán tử | (or)

|	0	1
0	**0**	1
1	1	1

- Ví dụ
  - int x = 2912, y = 1706, z = x | y;

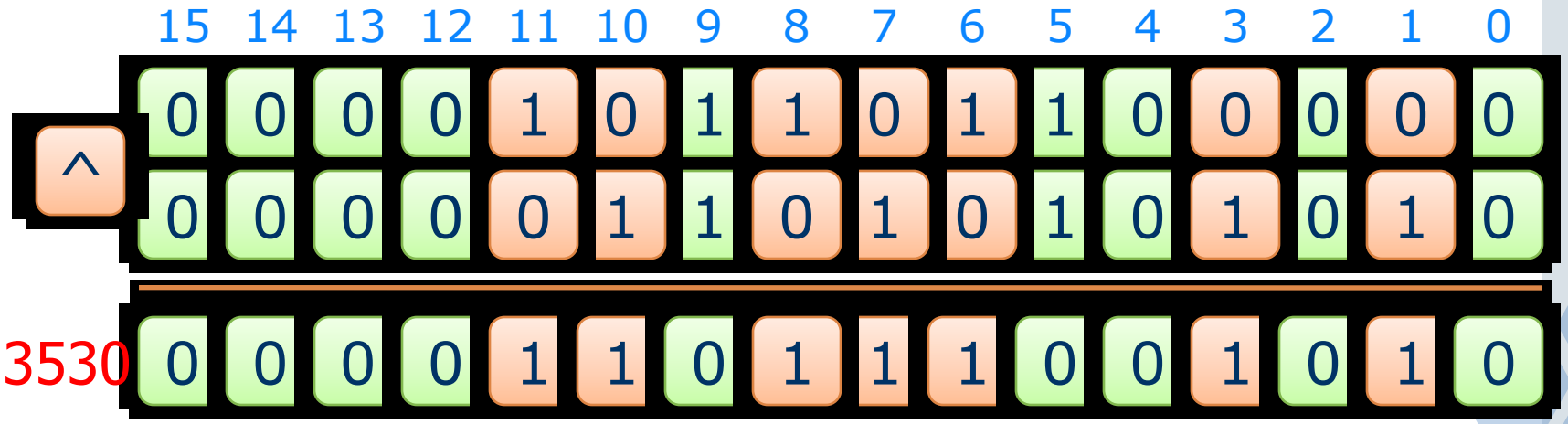


Các toán tử trên bit

- Toán tử ^ (xor)

^	0	1
0	**0**	1
1	1	**0**

- Ví dụ
  - int x = 2912, y = 1706, z = x ^ y;

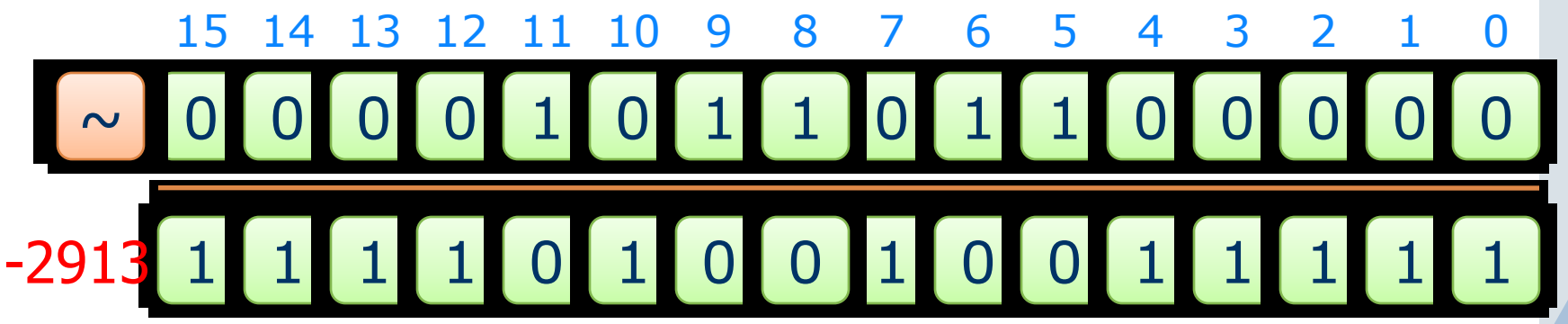


Các toán tử trên bit

- Toán tử ~ (not)

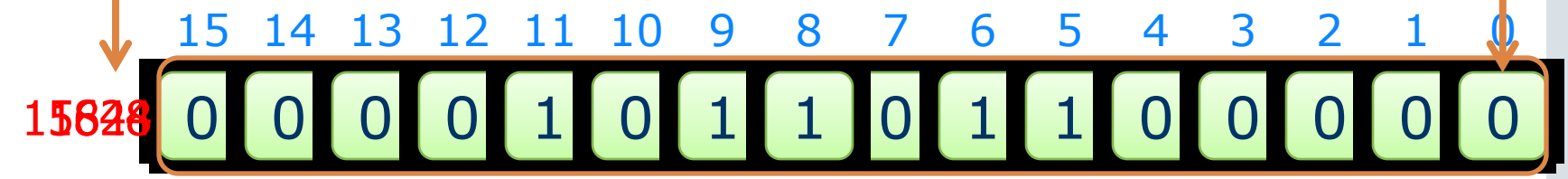
~	0	1
	1	0

- Ví dụ
  - int x = 2912, z = ~x;



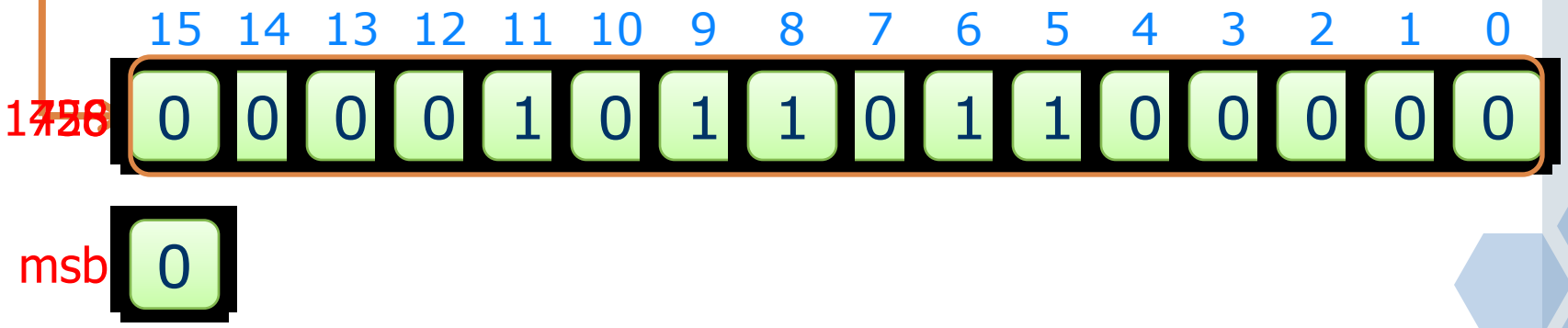
Các toán tử trên bit

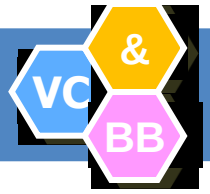
- Toán tử << n (shift left)
  - Dịch các bit sang trái n vị trí.
  - Các bit vượt quá phạm vi lưu trữ sẽ mất.
  - Tự động thêm bit 0 vào cuối dãy bit.
- Ví dụ
  - int x = 2912, z = x << 2;



Các toán tử trên bit

- Toán tử >> n (shift right)
  - Dịch các bit sang phải n vị trí.
  - Các bit vượt quá phạm vi lưu trữ sẽ mất.
  - Giữ lại bit nặng nhất (msb) ⇔ dấu của số
- Ví dụ
  - int x = 2912, z = x >> 2;





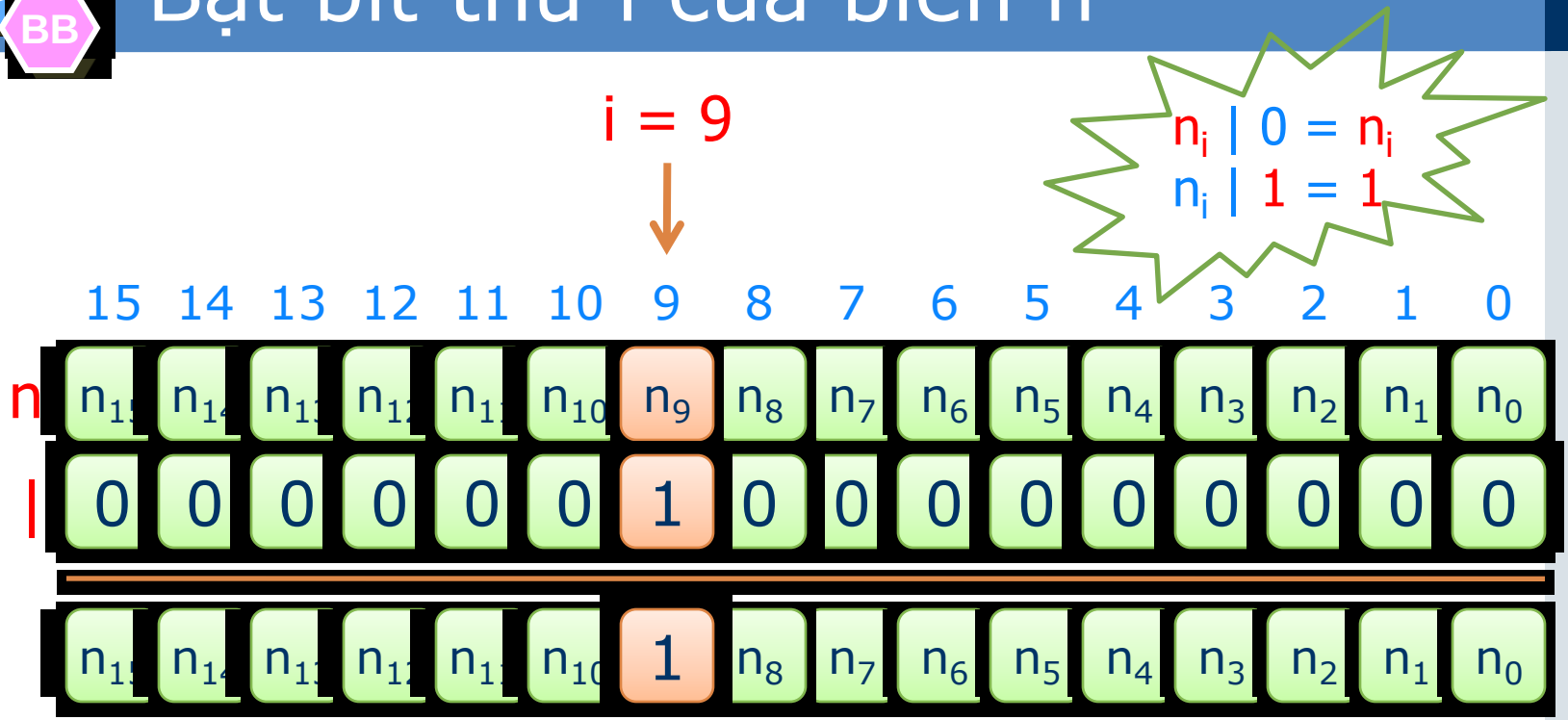
Các toán tử trên bit

- Lưu ý
  - Không được nhầm lẫn các các toán tử trên bit (&, |, ~) với các toán tử kết hợp (&&, || , !)
  - Các toán tử gộp: &= |= ^= <<= >>=
  - Máy tính làm việc trên bit nên các thao tác trên hệ nhị phân sẽ nhanh hơn rất nhiều so với hệ khác.
  - Phải luôn nhớ độ dài của dãy bit đang làm việc (8bit, 16bit, 32bit, 64bit, …)

Ứng dụng trên số nguyên

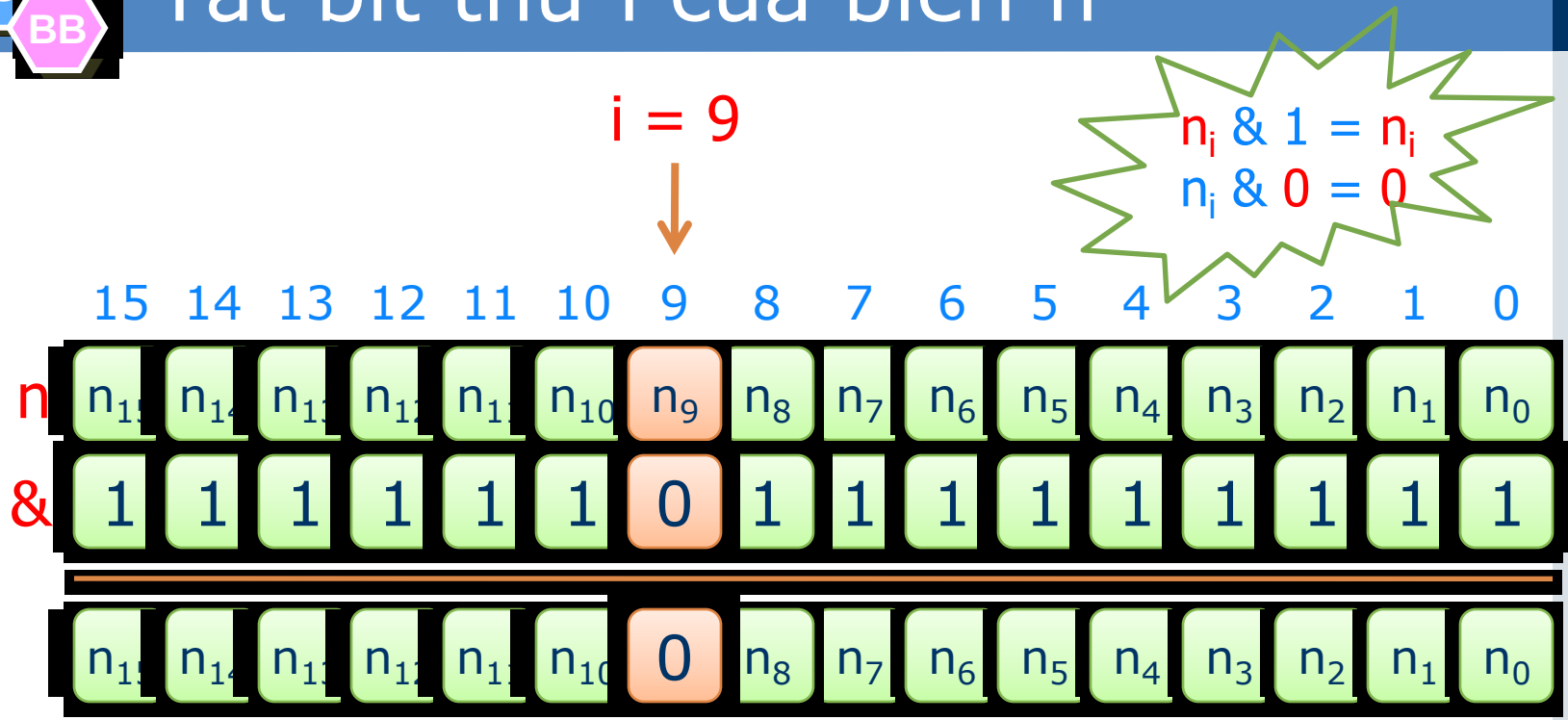
- Ứng dụng của các toán tử &, |, ^, ~
  a. Bật bit thứ i của biến n (onbit)
  b. Tắt bit thứ i của biến n (offbit)
  c. Lấy giá trị của bit thứ i của biến n (getbit)
  d. Gán giá trị 0 cho biến n (setzero)
- Ứng dụng của các toán tử dịch bit << và >>
  e. Nhân n với 2^i (mul2pow)
  f. Chia n với 2^i (div2pow)

Bật bit thứ i của biến n



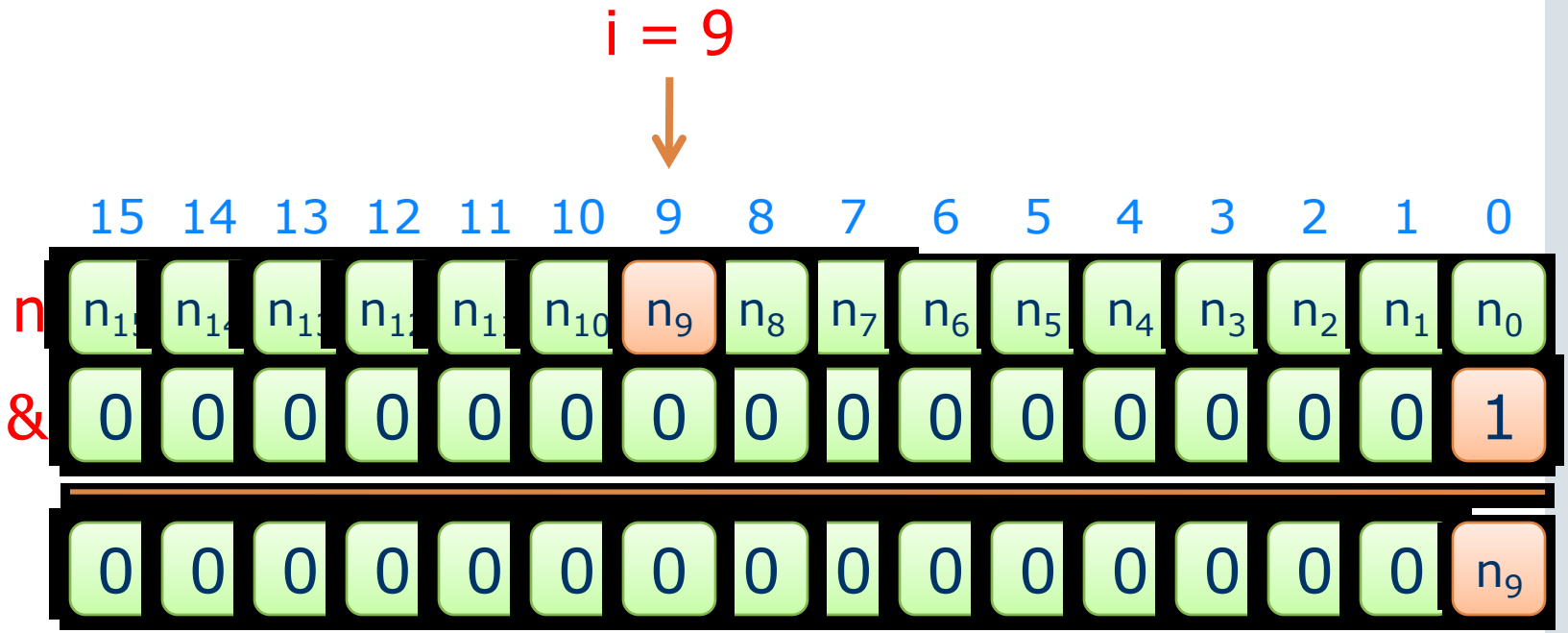

```
void onbit(int &n, int i)
{
        n = n | (0x1 << i);
}
```

Tắt bit thứ i của biến n




```
void offbit(int &n, int i)
{
        n =  n & (~(0x1 << i));
}
```

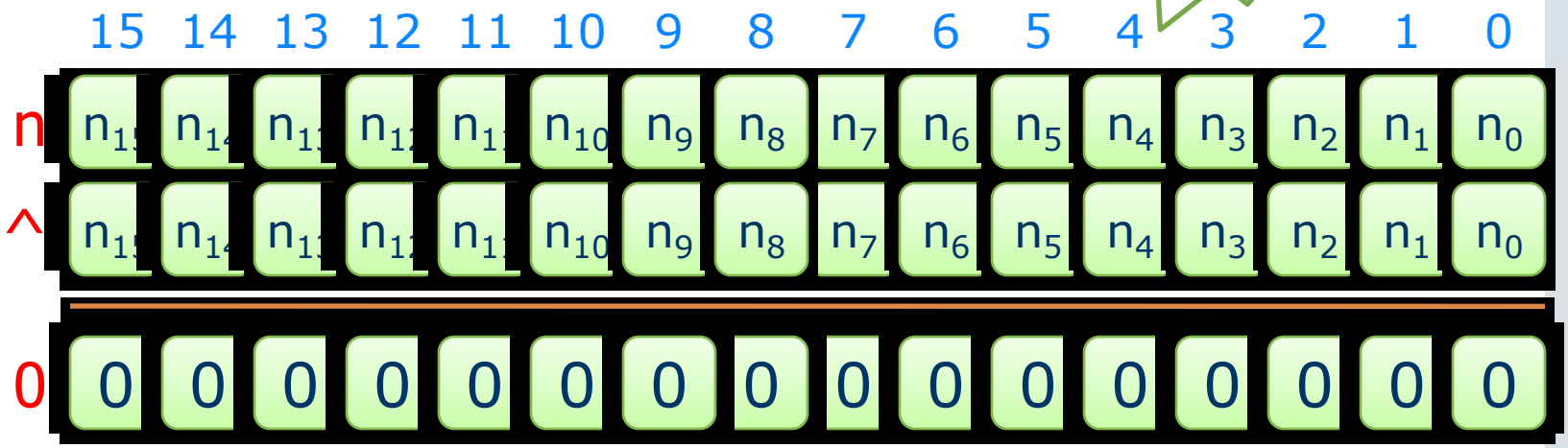
Lấy giá trị bit thứ i của biến n




```
int getbit(int n, int i)
{
        return (n >> i) & 0x1;
}
```

Gán giá trị 0 cho biến n

$n_i \wedge n_i = 0$




```
void setzero(int &n)
{
        n = n ^ n;
}
```

Nhân n với 2^i

- Đặc điểm toán tử <<
  - $n = \Sigma(n_j 2^j)$ với $j \in [0, k]$ (k là chỉ số bit msb)
  - Dịch trái i bit ➔ số mũ mỗi ký số tăng thêm i
  - ➔ $n << i = \Sigma(n_j 2^{j+i}) = 2^i \Sigma(n_j 2^j) = 2^i n$
  - Vậy, dịch trái i bit ⇔ nhân với 2^i

```
int mul2powi(int n, int i)
{
        return n << i;
}
```

Chia n với 2^i

- Đặc điểm toán tử >>
  - $n = \Sigma(n_j 2^j)$ với $j \in [0, k]$ (k là chỉ số bit msb)
  - Dịch phải i bit ➔ số mũ mỗi ký số giảm đi i
  - ➔ $n << i = \Sigma(n_j 2^{j-i}) = 2^{-i}\Sigma(n_j 2^j) = 2^{-i}n = n/2^i$
  - Vậy, dịch phải i bit ⇔ chia cho 2^i

```
int div2powi(int n, int i)
{
        return n >> i;
}
```

Bài tập

- Bài 1: Viết hàm thực hiện các thao tác trên bit.
- Bài 2: Viết bitcount đếm số lượng bit 1 của một số nguyên dương n.
- Bài 3: Cho mảng a gồm n số nguyên khác nhau. Viết hàm liệt kê các tổ hợp 1, 2, …, n phần tử của số nguyên đó (không cần theo thứ tự)

  Ví dụ, n = 3, mảng a = {1, 2, 3}

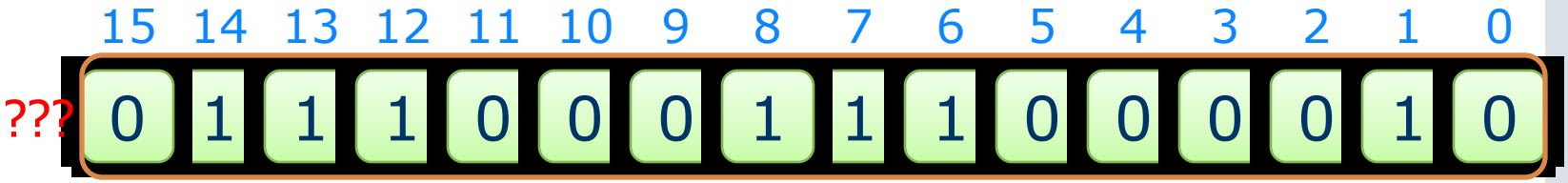
  ➔{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}
- Bài 4: Giống bài 3 nhưng chỉ liệt kê các tổ hợp k phần tử ($1 \le k \le n$)

Bài tập

- Bài 5: Viết hàm RotateLeft(n, i) thực hiện thao tác "xoay" các bit của n (kô dấu) sang trái i vị trí và các bit bị mất sẽ được đưa vào cuối dãy bit.

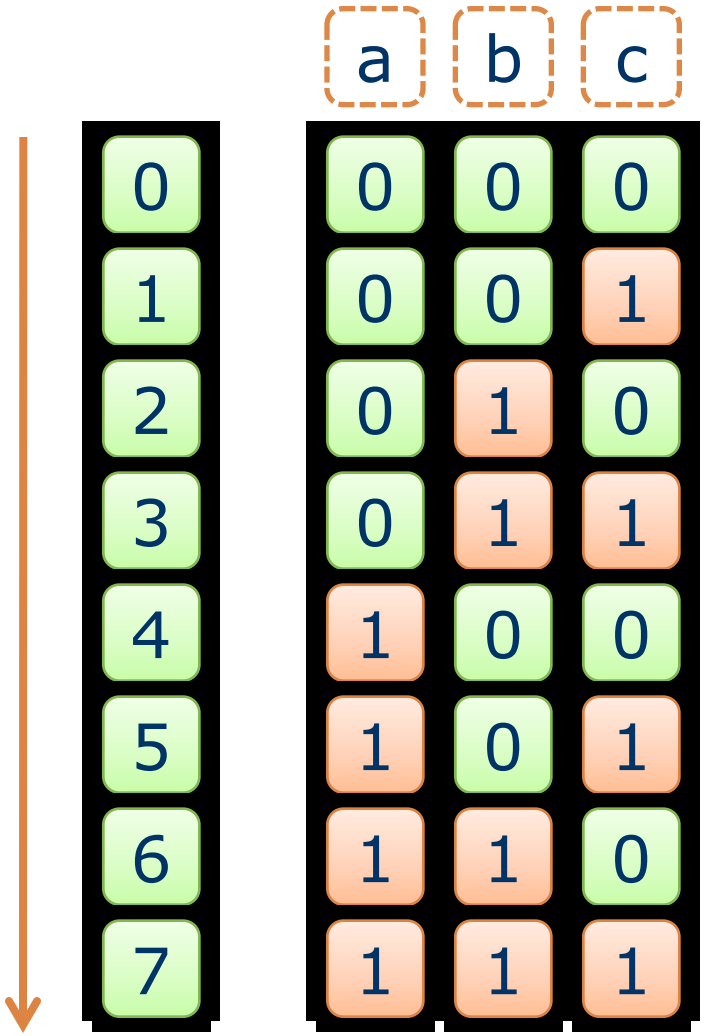
  Ví dụ:

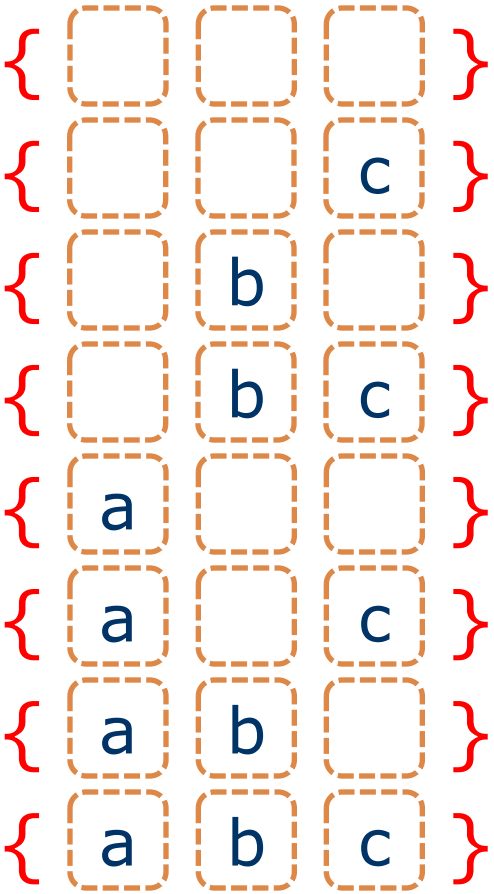
  - int n = 291282; n = RotateLeft(n, 2);



- Bài 6: Tương tự bài 2 nhưng viết hàm RotateRight(n, i) để xoay bit sang phải.

Bài 3





Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

HÀM NÂNG CAO (PHẦN 1)

Nội dung

1. Các tham số của hàm main
2. Hàm có đối số mặc định
3. Hàm trả về tham chiếu
4. Hàm nội tuyến (inline)

Các đối số của chương trình

Các đối số của chương trình

- Hàm main là hàm nên cũng có tham số.
- Chương trình tự động thực hiện hàm main mà không cần lời gọi hàm.

➔ Làm sao truyền đối số?

➔ Khi thực thi tập tin chương trình (.exe), ta truyền kèm đối số. Tất nhiên, hàm main cũng phải định nghĩa các tham số để có thể nhận các đối số này.

Các tham số của hàm main

Các tham số của hàm main

```
void main(int argc, char *argv[])
{
        ...
}
```

- Trong đó
  - argc là số lượng đối số (tính luôn tên tập tin chương trình)
  - argv là mảng chứa các đối số (dạng chuỗi)

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
argv = {"Cong.EXE", "2912", "1706"};
```

```
argc = 3
```

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
#include <stdio.h>
#include <stdlib.h>      // atoi
void main(int argc, char *argv[]) {
      if (argc == 3) {
            int x = atoi(argv[1]);
            int y = atoi(argv[2]);
            printf("%d + %d = %d", x, y, x+y);
      }
      else
            printf("Sai! VD: Cong 2912 1706");
}
```

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
argv = {"test", "input.txt", "output.txt"};
```

```
argc = 3
```

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
#include <stdio.h>
void main(int argc, char *argv[]) {
	if (argc == 3) {
		// Nhập dữ liệu từ tập tin argv[1]
		// Xử lý
		// Xuất kết quả ra tập tin argv[2]
	}
	else
		printf("Sai! VD: test in.txt out.txt");
}
```

Hàm có đối số mặc định

- Ví dụ
  - Viết hàm Tong để tính tổng 4 số x, y, z, t

```
int Tong(int x, int y, int z, int t)
{
        return x + y + z + t;
}
```

  - Tính tổng 4 số 2912, 1706, 1506, 1904

```
Tong(2912, 1706, 1506, 1904);
```

  - Nếu chỉ muốn tính tổng 2 số 2912, 1706

```
Tong(2912, 1706, 0, 0);     // z = 0, t = 0
```

Hàm có đối số mặc định

- Khái niệm
  - Hàm có đối số mặc định là hàm có một hay nhiều tham số hình thức được gán giá trị.
  - Tham số này nhận giá trị mặc định đó nếu không có đối số truyền vào cho tham số đó.
  - Phải được dồn về tận cùng bên phải.
- Ví dụ

```
int Tong(int x, int y, int z = 0, int t = 0)
{
        return x + y + z + t;
}
```

Hàm có đối số mặc định

- Lưu ý
  - Muốn truyền đối số khác thay cho đối số mặc định, phải truyền đối số thay cho các đối số mặc định trước nó.

```
int Tong(int x, int y = 0, int z = 0);

    Tong(1, 5);

    Tong(1, 0, 5);
```

Hàm có đối số mặc định

- Ví dụ
  - In thông tin SV trong lớp gồm: họ tên, phái, lớp, năm sinh

```
void XuatThongTin(char *hoten, char phai = 0,
char *lop = "TH07", int namsinh = 1989)
{
        puts(hoten);
        printf(phai == 0? "Nam\n" : "Nu\n");
        puts(lop);
        printf("%d", namsinh);
}
```

Hàm có đối số mặc định

- Ví dụ
  - In thông tin SV trong lớp gồm: họ tên, phái, lớp, năm sinh

```
void main()
{
        XuatThongTin("Nguyen Van A");
        XuatThongTin("Tran Thi B", 1);
        XuatThongTin("Hoang Van C", 0, "TH00");
        XuatThongTin("Le D", 1, "TH07", 1988);
}
```

Hàm có đối số mặc định

- Nhận xét
  - x = a thường xuyên xảy ra thì nên chuyển x thành tham số có đối số mặc định là a.

    Ví dụ, hầu hết phai = 0 (nam), lop = "TH07" và namsinh = 1989.
  - x = a và y = b thường xuyên xảy ra nhưng y = b thường xuyên hơn thì nên đặt tham số mặc định x trước y.

    Ví dụ, lop = "TH07" xảy ra nhiều hơn phai = 0 nên đặt lop sau phai.

Chỉ thị tiền xử lý #define

- Định nghĩa hằng ký hiệu
  - Chỉ thị #define <name> <value>
  - Mọi chỗ xuất hiện <name> trong chương trình nguồn được thay thế bằng <value> để tạo ra chương trình tiền xử lý.
  - Ví dụ
    - #define MAX 1000
    - #define PI 3.14
    - #define message "Hello World\n"

Chỉ thị tiền xử lý #define

- Định nghĩa các macro (lệnh gộp - lệnh tắt)
  - #define <name>(<param-list>) <expression>
  - Mọi chỗ xuất hiện của <name> với lượng tham số đưa vào phù hợp sẽ được thay thế bởi <expression> (tham số được thay thế tương ứng)
  - Ví dụ
    - #define showmsg(msg) printf(msg)
    - ➔showmsg("Hello"); ⇔ printf("Hello");

Hàm nội tuyến (inline)

- Ví dụ
  - Xét 2 cách sau

```
#define PI 3.14159
float addPi(float s)
{
	return s + PI;
}

void main()
{
	float s = 0;
	for (int i = 1; i<=100000; i++)
		s = s + PI;		// Cách 1 (0.7s)
		s = addPi(s);		// Cách 2 (1.4s)
}
```

Hàm nội tuyến (inline)

- Nhận xét
  - Sử dụng hàm giúp chương trình dễ hiểu nhưng lại tốn chi phí cho lời gọi hàm.
- Khắc phục
  - Sử dụng hàm nội tuyến (inline) bằng cách thêm từ khóa inline trước prototype của hàm.

    ➔inline float addPi(float s) {return s + PI;}
- Khái niệm
  - Sao chép thân hàm đến bất cứ nào nào hàm được gọi ➔ kết quả giống hệt cách 1.

Hàm nội tuyến (inline)

- Lưu ý
  - Giảm thời gian thực hiện hàm (gọi và kết thúc).
  - Giảm không gian bộ nhớ do các hàm con chiếm dụng khi hàm được gọi.
  - Không cho phép các hàm nội tuyến đệ quy.
  - Phần lớn không cho phép thực hiện nội tuyến các hàm sử dụng vòng lặp while.
  - Chỉ inline các hàm nhỏ, inline các hàm lớn sẽ gây phản tác dụng (bộ nhớ cho hàm inline chiếm giữ sẽ lâu giải phóng hơn).

Hàm trả về tham chiếu

- Ví dụ
  - Hàm chỉ trả về giá trị. Ví dụ, x = f();
  - Vậy, g() = x hợp lệ hay không?
  - ➔ Hợp lệ khi g(x) trả về tham chiếu đến một biến (C++)
- Cú pháp

```
<kiểu trả về> &<tên hàm>([<ds tham số>])
{
        return <biến>;
}
```

Hàm trả về tham chiếu

- Ví dụ

```
#include <stdio.h>

int x;

int &getx()
{
        return x;
}

void main()
{
        getx() = 5; // ⇔ x = 5
}
```

Hàm trả về tham chiếu

- Ứng dụng
  - Chỉ số của mảng trong C/C++ bắt từ 0

    ➔Không quen thuộc lắm.
  - Viết hàm để khi muốn truy cập đến phần tử thứ i của mảng a ta sử dụng V(i) thay vì a[i-1]

```
int a[100];
int &V(int i)
{
      return a[i-1];
}
…
V(1) = 2912; // ⇔ a[0] = 2912;
```

Hàm trả về tham chiếu

- Chú ý
  - Trong trường hợp sau, biến x phải là biến toàn cục ➔ không nên sử dụng!

```
int x; // biến toàn cục

int &getx()
{
	return x;
}

void main()
{
	getx() = 2912;
}
```

Hàm trả về tham chiếu

- Chú ý
  - Nếu không muốn sử dụng biến toàn cục, phải truyền x ở dạng tham chiếu.

```
int &getx(int x) {  // SAI! x là tham trị ➔ bản sao
      return x;
}

int &getx() {
      int x;        // SAI! x là biến cục bộ
      return x;
}

int &getx(int &x) { // ĐÚNG! x là tham chiếu
      return x;
}
```

Hàm trả về tham chiếu

- Ví dụ

```
#include <stdio.h>

int &V(int a[], int i)
{
	return a[i-1];
}

void main()
{
	int a[100];
	for (int i = 1; i <= 100; i++)
		V(a, i) = 0;
}
```

Bài tập

- Bài 1: Viết chương trình có tên TinhToan sao cho khi gõ: TinhToan 2912 – 1706 sẽ xuất ra màn hình 1206 (có thể thay bằng +, *, /)
- Bài 2: Viết chương trình quản lý thông tin sinh viên (sử dụng hàm có đối số mặc định), bao gồm nhập, sắp xếp tăng dần theo tên và xuất danh sách sinh viên.
- Bài 3: Chuyển các hàm nhỏ hàm nội tuyến.
- Bài 4: Nhập mảng, sắp xếp mảng tăng dần và xuất mảng sử dụng hàm trả về tham chiếu.

Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

HÀM NÂNG CAO (PHẦN 2)

Nội dung

1. Tham số …
2. Khuôn mẫu hàm
3. Nạp chồng hàm
4. Nạp chồng toán tử

Tham số ...

Khai báo

```
<kiểu trả về> <tên hàm>(<dsts biết trước>, …)
{
        …
}
```

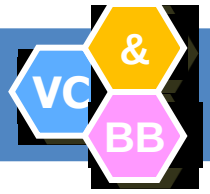
Ý nghĩa

- Hàm có số lượng tham số không biết trước và thường cùng kiểu (không được là char, unsigned char, float).
- Phải có ít nhất 1 tham số biết trước.
- Tham số … đặt ở cuối cùng.

Tham số ...

- Ví dụ

```
void XuatTong1(char *msg, int n, …)
{
      // Các lệnh ở đây
}

void XuatTong2(char *msg, …)
{
      // Các lệnh ở đây
}

int Tong(int a, …)
{
      // Các lệnh ở đây
}
```



Truy xuất danh sách tham số …

- Sử dụng kiểu và các macro sau (stdarg.h)
  - va_list : kiểu dữ liệu chứa các tham số có trong …
  - va_start(va_list ap, lastfix) : macro thiết lập ap chỉ đến tham số đầu tiên trong … với lastfix là tên tham số cố định cuối cùng.
  - type va_arg(va_list ap, type) : macro trả về tham số có kiểu type tiếp theo.
  - va_end(va_list ap) : macro giúp cho hàm trả về giá trị một cách "bình thường".

Tham số ...

Ví dụ

```
#include <stdarg.h>
void XuatTong1(char *msg, int n, …)
{
        va_list ap;
        va_start(ap, n); // ts cố định cuối cùng
        int value, s = 0;
        for (int i=0; i<n; i++)
        {
                value = va_arg(ap, int);
                s = s + value;
        }
        va_end(ap);
        printf("%s %d", msg, s);
}
```

Tham số ...

- Ví dụ

```
#include <stdarg.h>
void XuatTong2(char *msg, …)
{
        va_list ap;
        va_start(ap, msg); // ts cố định cuối
        int value, s = 0;
        while ((value = va_arg(ap, int)) != 0)
        {
                s = s + value;
        }
        va_end(ap);
        printf("%s %d", msg, s);
}
```

Tham số ...

- Ví dụ

```
#include <stdarg.h>
int Tong(int a, …)
{
	va_list ap;
	va_start(ap, n); // ts cố định cuối cùng
	int value, s = a;
	while ((value = va_arg(ap, int)) != 0)
	{
		s = s + value;
	}
	va_end(ap);
	return s;
}
```

Khuôn mẫu hàm

- Viết hàm tìm số nhỏ nhất trong 2 số
  - Viết các hàm khác nhau để tìm min 2 số int, 2 số long, 2 số float, 2 số double, 2 phân số…
- Nhược điểm
  - Hàm bản chất giống nhau nhưng khác kiểu dữ liệu nên phải viết nhiều hàm giống nhau.
  - Sửa 1 hàm phải sửa những hàm còn lại.
  - Không thể viết đủ các hàm cho mọi trường hợp do còn nhiều kiểu dữ liệu khác.

Khuôn mẫu hàm

- Khái niệm
  - Viết một hàm duy nhất nhưng có thể sử dụng cho nhiều kiểu dữ liệu khác nhau.
- Cú pháp
  - tempate <ds mẫu tham số> <khai báo hàm>
- Ví dụ

```
template <class T> <khai báo hàm>

hoặc

template <class T1, class T2> <khai báo hàm>
```

Khuôn mẫu hàm

- Ví dụ

```
template <class T>
T min(T a, T b)
{
	if (a < b)
		return a;
	return b;
}

void main()
{
	int a = 2912, b = 1706;
	int m = min<int>(a, b);
	printf("So nho nhat la %d", m);
}
```

Khuôn mẫu hàm

- Lợi ích của việc sử dụng khuôn mẫu hàm
  - Dễ viết, do chỉ cần viết hàm tổng quá nhất.
  - Dễ hiểu, do chỉ quan tâm đến kiểu tổng quát nhất.
  - Có kiểu an toàn do trình biên dịch kiểm tra kiểu lúc biên dịch chương trình.
  - Khi phối hợp với sự quá tải hàm, quá tải toán tử hoặc con trỏ hàm ta có thể viết được các chương trình rất hay, ngắn gọn, linh động và có tính tiến hóa cao.

Nạp chồng hàm

- Nhu cầu
  - Thực hiện một công việc với nhiều cách khác nhau. Nếu các hàm khác tên sẽ khó quản lý.
- Khái niệm nạp chồng/quá tải (overload) hàm
  - Hàm cùng tên nhưng có tham số đầu vào hoặc đầu ra khác nhau.
  - Nguyên mẫu hàm (prototype) khi bỏ tên tham số phải khác nhau.
  - Cho phép người dùng chọn phương pháp thuận lợi nhất để thực hiện công việc.

Nạp chồng hàm

- Ví dụ
  - Nhập mảng theo nhiều cách

```
void Nhap(int a[], int &n)
{
	// Nhập n rồi nhập mảng a
}
void Nhap(int a[], int n)
{
	// Nhập mảng a theo n truyền vào
}
int Nhap(int a[])
{
	// Nhập n, nhập mảng a rồi trả n về
}
```

Nạp chồng hàm

- Ví dụ
  - Gán giá trị cho PHANSO

```
void Gan(PHANSO &ps, int tu, int mau)
{
	ps.tu = tu;
	ps.mau = mau;
}

void Gan(PHANSO &ps, int tu)
{
	ps.tu = tu;
	ps.mau = 1;
}
```

Nạp chồng hàm

- Chú ý
  - Các hàm sau đây là như nhau

```
int Tong(int a, int b)   // int Tong(int, int)
{
      return a + b;
}
int Tong(int b, int a)   // int Tong(int, int)
{
      return a + b;
}
int Tong(int x, int y)   // int Tong(int, int)
{
      return x + y;
}
```

Nạp chồng hàm

- Sự nhập nhằng, mơ hồ (ambiguity)
  - Do sự tự chuyển đổi kiểu

```
float f(float x) { return x / 2; }
double f(double x) { return x / 2; }

void main()
{
        float x = 29.12;
        double y = 17.06;
        printf("%.2f\n", f(x)); // float
        printf("%.2lf\n", f(y));// double
        printf("%.2f", f(10));  // ???
}
```

Nạp chồng hàm

- Sự nhập nhằng, mơ hồ (ambiguity)
  - Do sự tự chuyển đổi kiểu

```
void f(unsigned char c)
{
      printf("%d", c);
}
void f(char c)
{
      printf("%c", c);
}
void main()
{
      f('A');      // char
      f(65);       // ???
}
```

Nạp chồng hàm

- Sự nhập nhằng, mơ hồ (ambiguity)
  - Do việc sử dụng tham chiếu

```
int f(int a, int b)
{
      return a + b;
}
int f(int a, int &b)
{
      return a + b;
}
void main()
{
      int x = 1, y = 2;
      printf("%d", f(x, y));  // ???
}
```

Nạp chồng hàm

- Sự nhập nhằng, mơ hồ (ambiguity)
  - Do việc sử dụng tham số mặc định

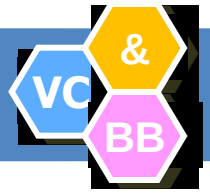
```
int f(int a)
{
	return a*a;
}
int f(int a, int b = 0)
{
	return a*b;
}
void main()
{
	printf("%d\n", f(2912, 1706));
	printf("%d\n", f(2912));	//???
}
```

Nạp chồng toán tử

- Khái niệm
  - Giống như quá tải hàm.
  - Có một số quy định khác về tham số.
  - Tạo thêm hàm toán tử (operator function) để nạp chồng toán tử.

```
<kiểu trả về> operator#(<ds tham số>)
{
        // Các thao tác cần thực hiện
}
```

  - # là toán tử (trừ . :: .* ?), <ds tham số> phụ thuộc vào toán tử được nạp chồng.



Nạp chồng toán tử

- Toán tử hai ngôi
  - Toán tử gán (=), số học (+, –, *, /, %), quan hệ (<, <=, >, >=, !=, ==), luận lý (&&, ||, !)
  - Gồm có hai toán hạng
  - Có thể thay đổi toán hạng vế trái
  - Có thể trả kết quả về cho phép toán tiếp theo
- Toán tử một ngôi
  - Toán tử tăng giảm (++, – –), toán tử đảo dấu (–)
  - Chỉ có một toán hạng

Nạp chồng toán tử

- Toán tử hai ngôi
  - Toán tử + (cho hai PHANSO)

```
typedef struct {int tu, mau;} PHANSO;

PHANSO operator+(PHANSO ps1, PHANSO ps2)
{
        PHANSO ps;
        ps.tu = ps1.tu*ps2.mau + ps2.tu*ps1.mau;
        ps.mau = ps1.mau*ps2.mau;
        return ps;
}
…
PHANSO a = {1, 2}, b = {3, 4}, c = {5, 6};
PHANSO d = a + b + c;    // ⇔ d = a + (b + c)
```

Nạp chồng toán tử

- Toán tử hai ngôi
  - Toán tử + (cho hai PHANSO)

```
typedef struct {int tu, mau;} PHANSO;

void operator+(PHANSO &ps1, PHANSO ps2)
{
        PHANSO ps;
        ps.tu = ps1.tu*ps2.mau + ps2.tu*ps1.mau;
        ps.mau = ps1.mau*ps2.mau;
        ps1 = ps;
}
…
PHANSO a = {1, 2}, b = {3, 4}, c = {5, 6};
PHANSO d = a + b + c;    // Lỗi
```

Nạp chồng toán tử

- Toán tử hai ngôi
  - Toán tử + (cho hai PHANSO)

```
typedef struct {int tu, mau;} PHANSO;

PHANSO operator+(PHANSO ps1, int n)
{
        PHANSO ps;
        ps.tu = ps1.tu + ps1.mau*n;
        ps.mau = ps1.mau;
        return ps;
}
…
PHANSO a = {1, 2};
PHANSO b = a + 2; // OK
PHANSO c = 2 + a; // Lỗi sai thứ tự toán hạng
```

Nạp chồng toán tử

- Toán tử hai ngôi
  - Toán tử == (cho hai PHANSO)

```
typedef struct {int tu, mau;} PHANSO;

int operator==(PHANSO ps1, PHANSO ps2)
{
      if (ps1.tu*ps2.mau == ps2.tu*ps1.mau)
            return 1;
      return 0;
}
…
PHANSO a = {1, 2}, b = {2, 4};
if (a == b)
      printf("a bằng b");
```

Nạp chồng toán tử

- Toán tử một ngôi
  - Toán tử tăng ++ (trước)

```
typedef struct {int tu, mau;} PHANSO;

PHANSO operator++(PHANSO &ps)
{
      ps.tu = ps.tu + ps.mau;
      return ps;
}
…
PHANSO a1 = {1, 2}, a2 = {1, 2};
PHANSO c1 = ++a1;
PHANSO c2 = a2++; // OK nhưng warning ++ sau
```

Nạp chồng toán tử

- Toán tử một ngôi
  - Toán tử tăng ++ (sau)

```
typedef struct {int tu, mau;} PHANSO;

PHANSO operator++(PHANSO &ps, int notused)
{
      ps.tu = ps.tu + ps.mau;
      return ps;
}
…
PHANSO a = {1, 2}, b = {3, 4};
PHANSO c1 = ++a;  // operator++(ps)
PHANSO c2 = a++;  // operator++(ps, 0)
```

Nạp chồng toán tử

- Toán tử một ngôi
  - Toán tử đảo dấu –

```
typedef struct {int tu, mau;} PHANSO;

PHANSO operator–(PHANSO ps)
{
        ps.tu = –ps.tu;
        return ps;
}
…
PHANSO a = {1, 2};
PHANSO b = –a;
```

Bài tập

- Bài 1: Viết chương trình tính tổng các số nguyên truyền vào hàm (có truyền thêm số lượng).
- Bài 2: Sửa lại bài 1 để cho phép người dùng tính tổng các số có kiểu bất kỳ được truyền vào hàm (có truyền thêm số lượng)
- Bài 3: Viết chương trình sắp xếp mảng tăng dần. Các phần tử của mảng có kiểu bất kỳ (char, int, long, float, double, phân số, sinh viên …)
- Bài 4: Sửa lại bài 3 để cho phép người dùng thay đổi quy luật sắp xếp (tăng, giảm, âm tăng dương giảm, …)