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

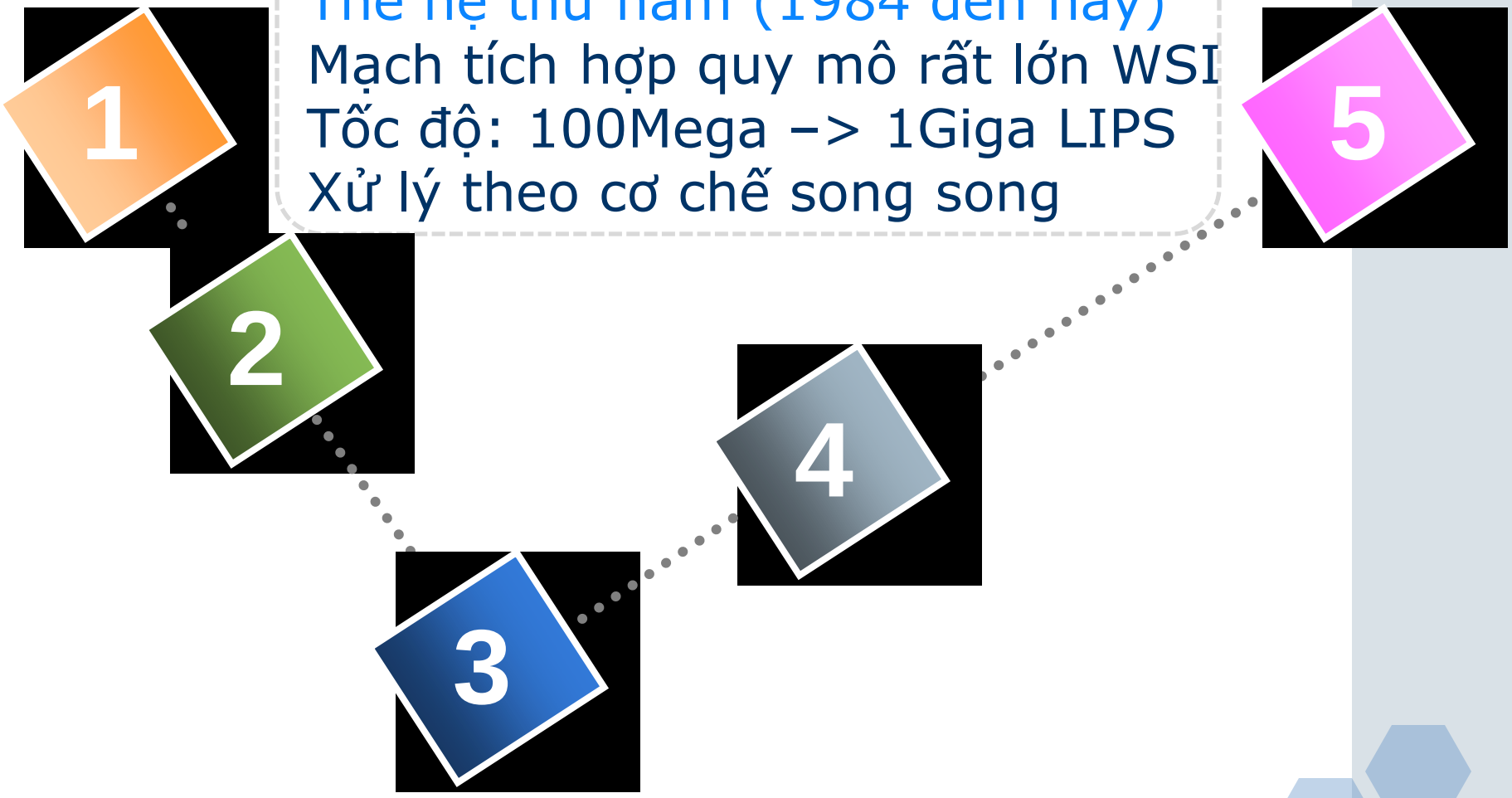
Tốc độ cao: 10^{12} phép tính/s

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

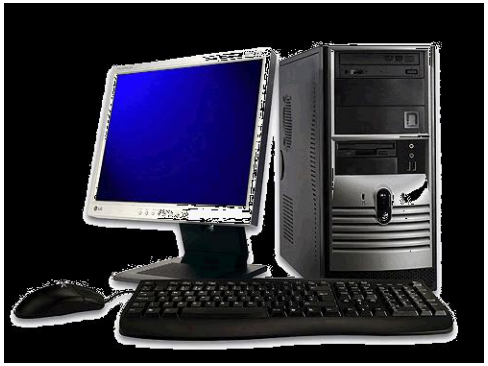
Máy tính lớn (Mainframe)

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.

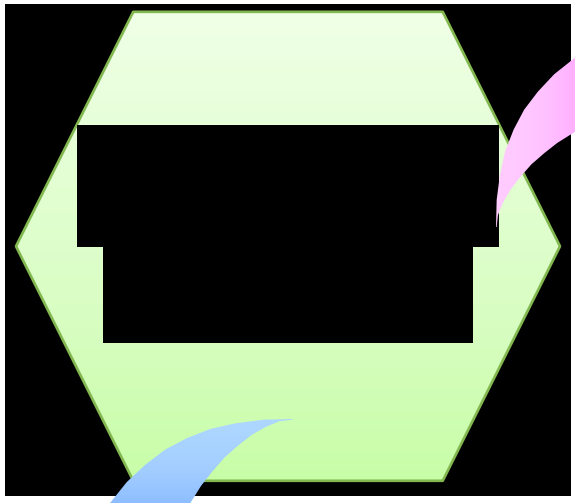
Máy tính xách tay (Laptop)
Còn gọi là “Notebook”.
Là loại máy tính nhỏ, có thể mang theo người.
Chạy bằng pin.

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.

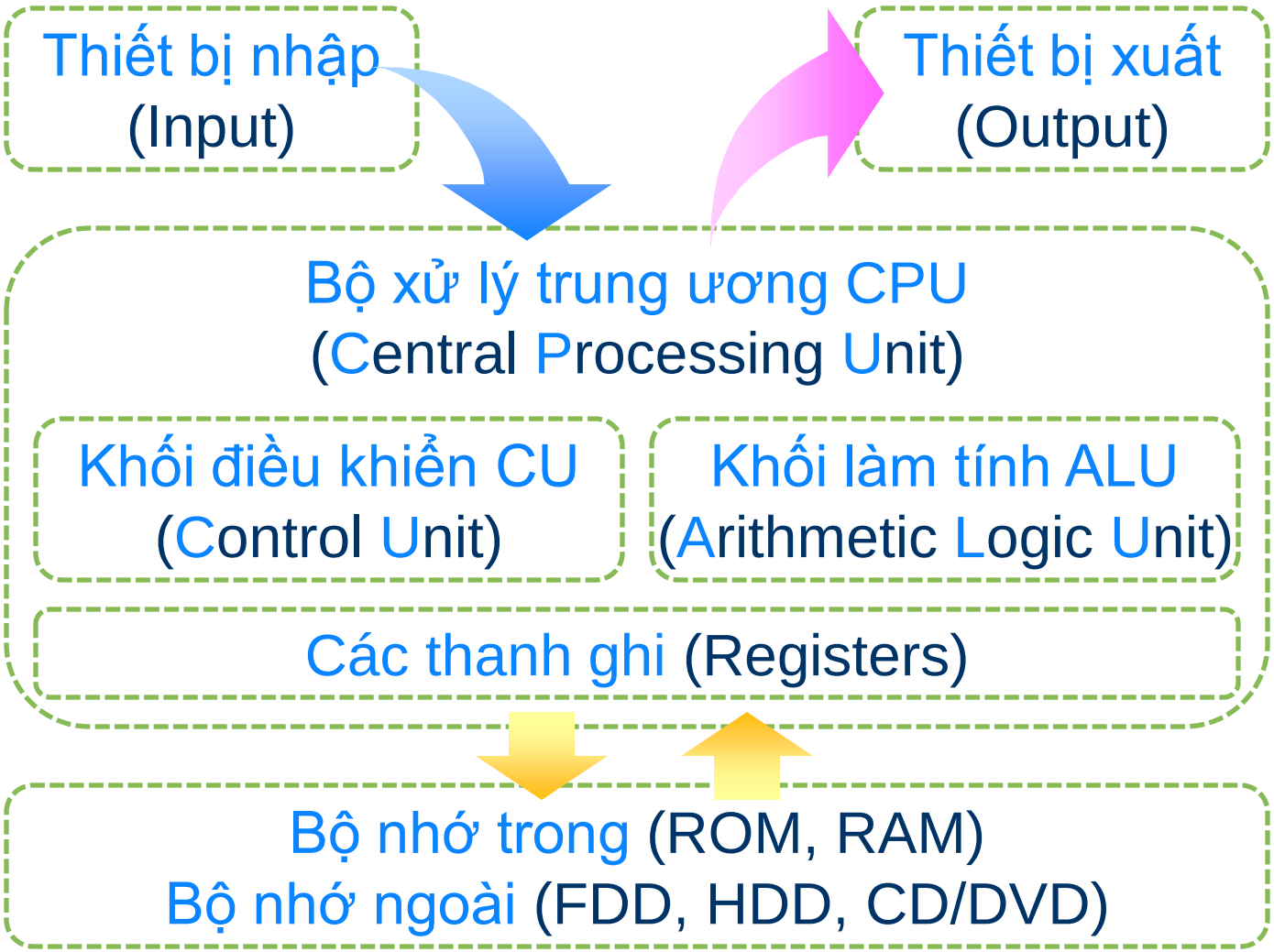
Các thành phần cơ bản



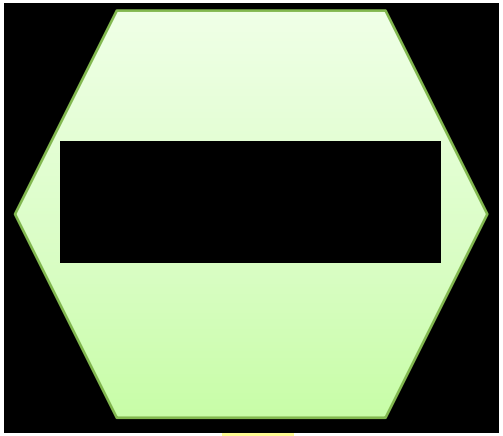
Phần mềm (Software)
- Phần mềm hệ thống
- Phần mềm ứng dụng

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

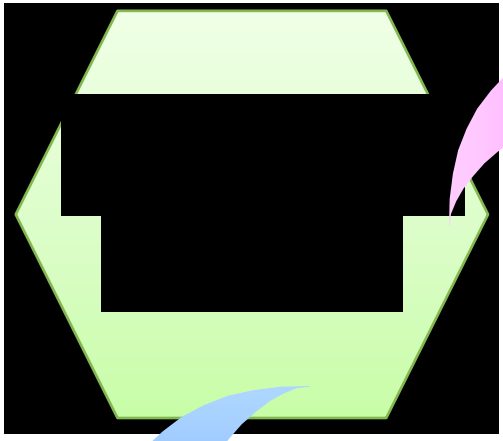


Phần cứng - Bộ nhớ



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

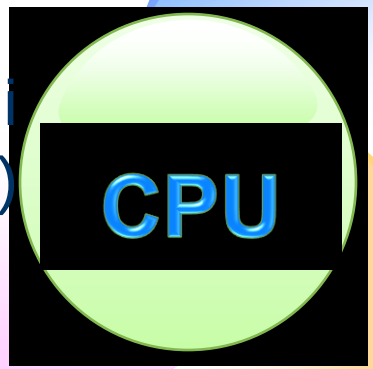
Thẻ nhớ (Memory Stick hay Compact Flash Card)
Dung lượng khoảng
32 MB, 64 MB, 128 MB…

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”…

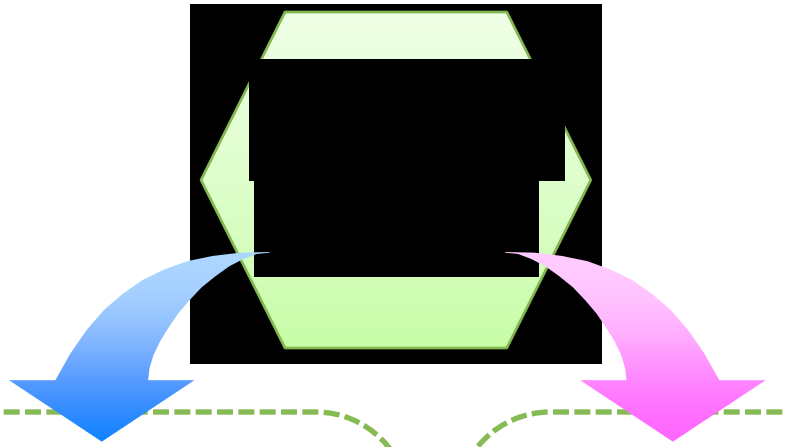
Phần cứng - Thiết bị xuất

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

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.

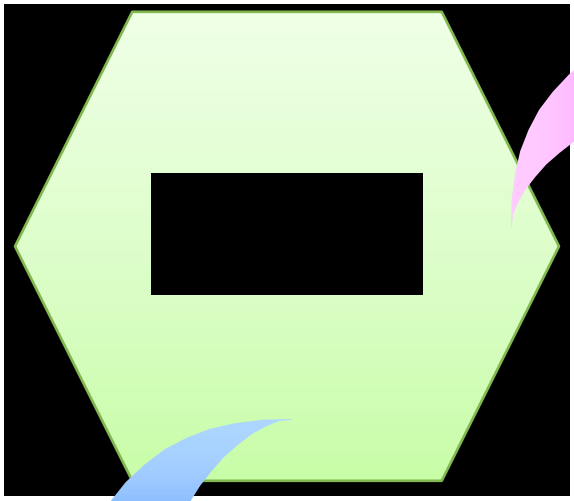
Kiểu luận lý

- 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



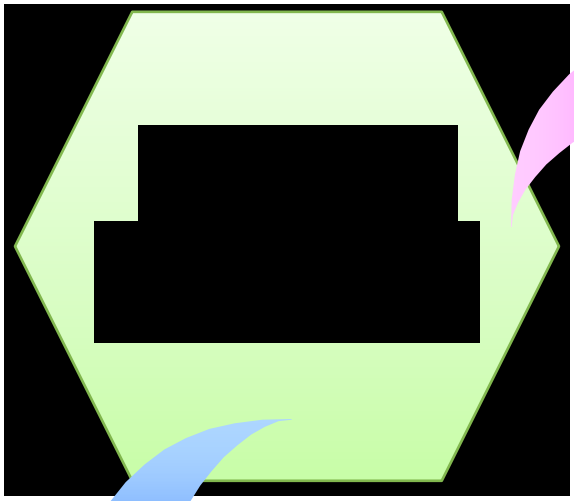
Ví dụ

```
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



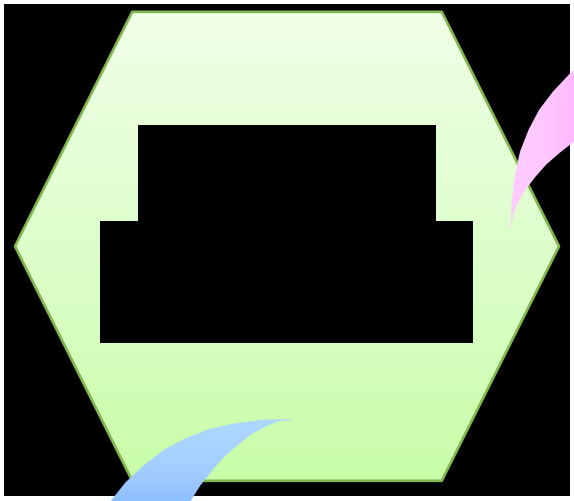
Cú pháp

<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

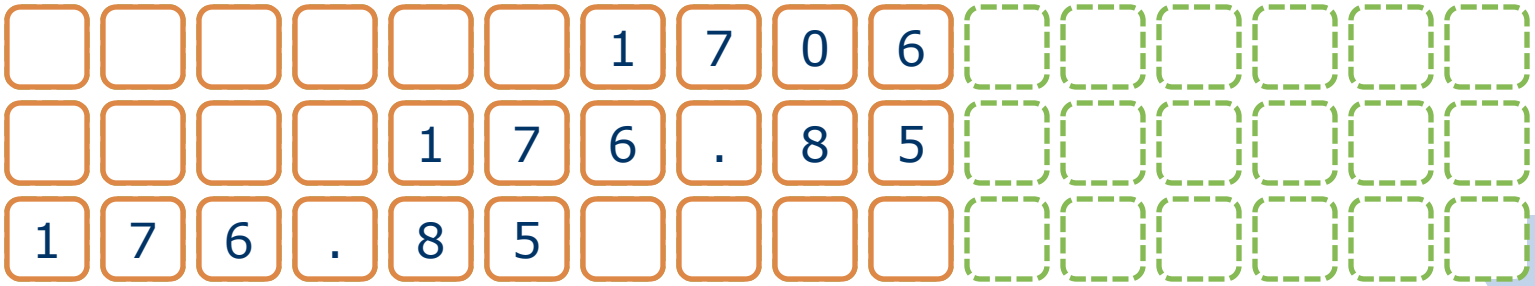
- Ví dụ
  - int a = 10, b = 20;
  - printf(“%d”, a); ➔ Xuất ra 10
  - printf(“%d”, b); ➔ Xuất ra 20
  - printf(“%d %d”, a, b); ➔ Xuất ra 10 20

  - 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.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

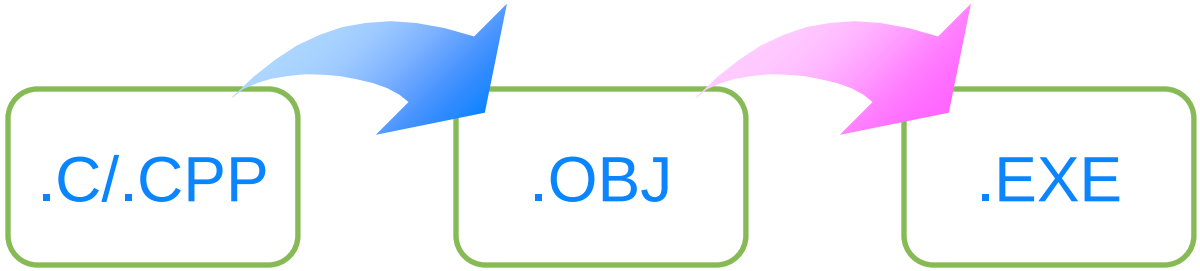
- Dennis Ritchie tại Bell Telephone năm 1972.
- Tiền thân của ngôn ngữ B, KenThompson, cũng tại Bell Telephone.
- Là ngôn ngữ lập trình có cấu trúc và phân biệt chữ Hoa - thường (case sensitive)
- ANSI C.

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.

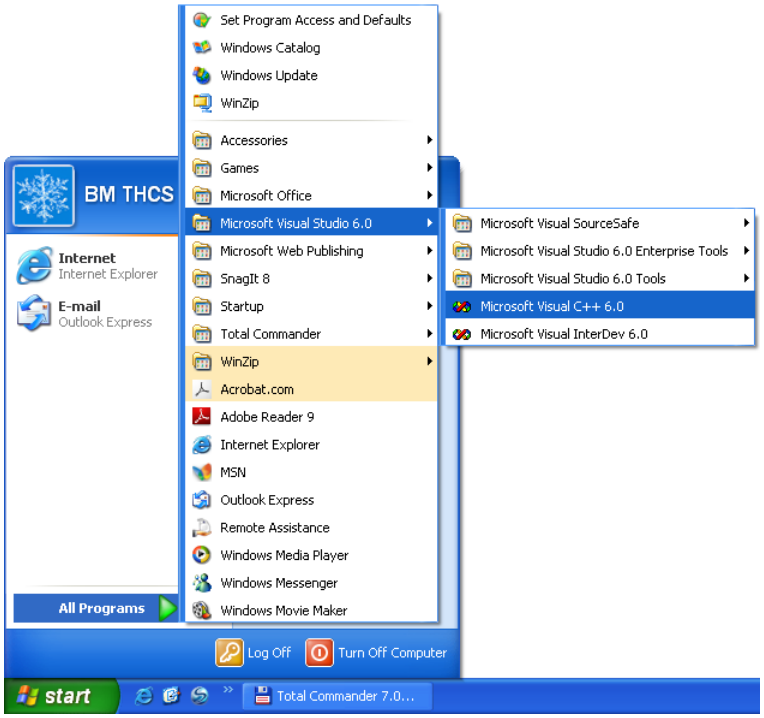
Giới thiệu

- 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

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 cấu 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.hcmuns.edu.vn

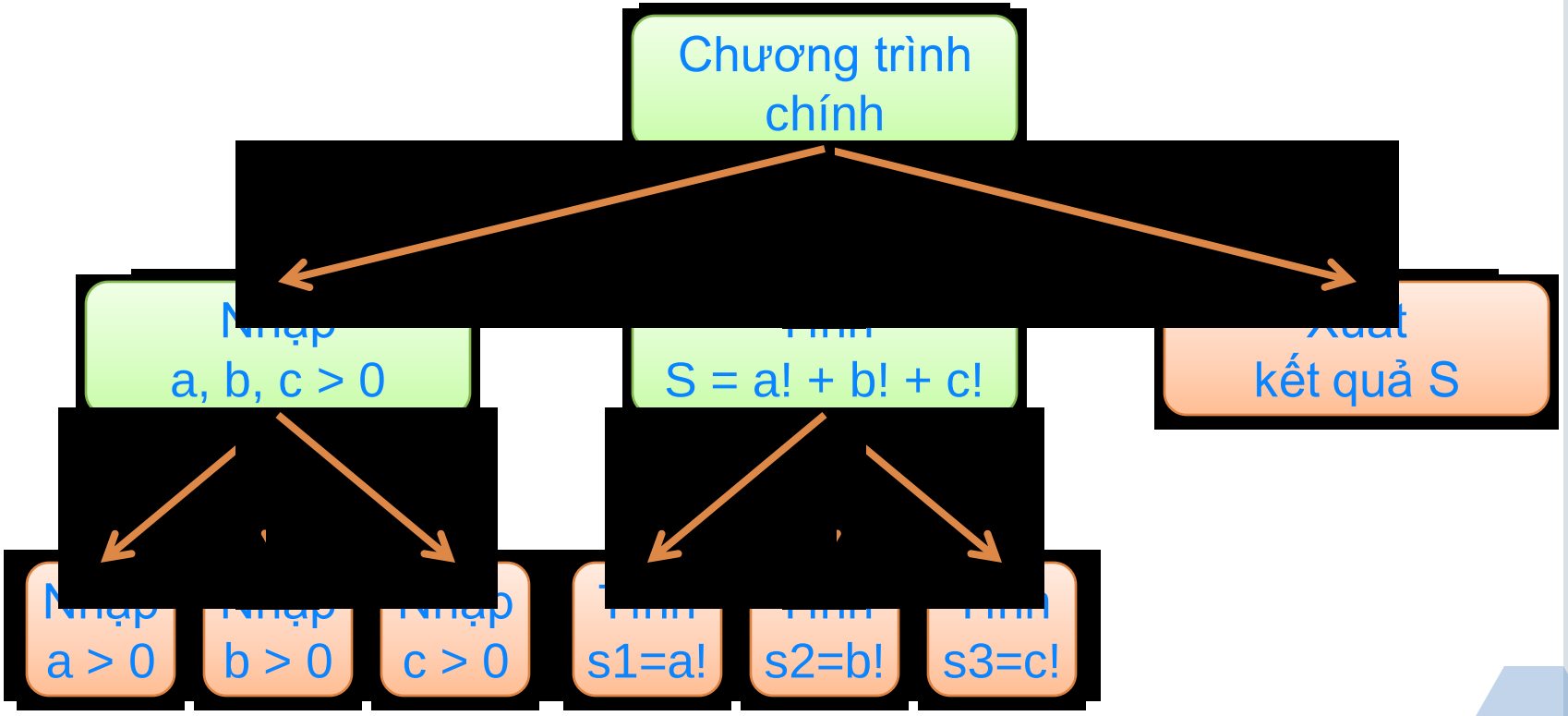
HÀM (FUNCTION)

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 thực hành

5. Bài 4, 5, 6, 7, 8 trang 140-141 chương 8 (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 n


e. Viết thủ tục hoán vị hai số nguyên.


f. Viết thủ tục sắp xếp 4 số nguyên tăng dầ

Bài tập thực hành

6. Bài tập 3 trang 155 chương 9 (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 thực hành

7. Bài tập 4 trang 156 chương 9 (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. $S = 1! + 2! + \ldots + n!$
8. Hàm trả về USCLN của 2 số nguyên.
9. 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

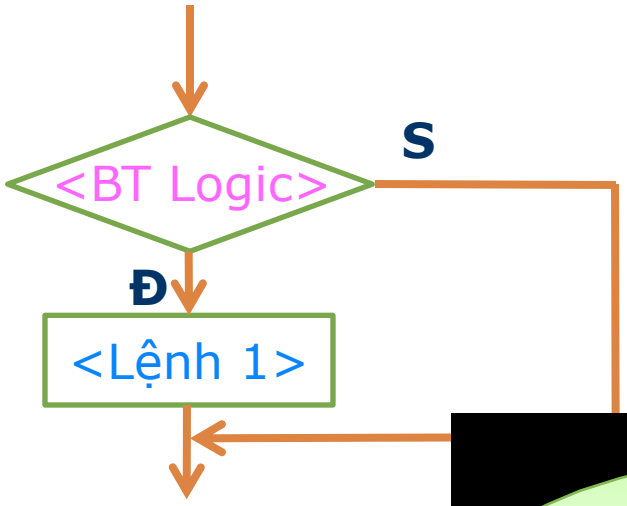
Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÂU LỆNH ĐIỀU KIỆN & CÂU LỆNH RẼ NHÁNH

Nội dung

1. Câu lệnh điều kiện if
2. Câu lệnh rẽ nhánh switch
3. Một số kinh nghiệm lập trình
4. Một số ví dụ minh họa

Câu lệnh if (thiếu)



if (<BT Logic>**)**

<Lệnh 1>

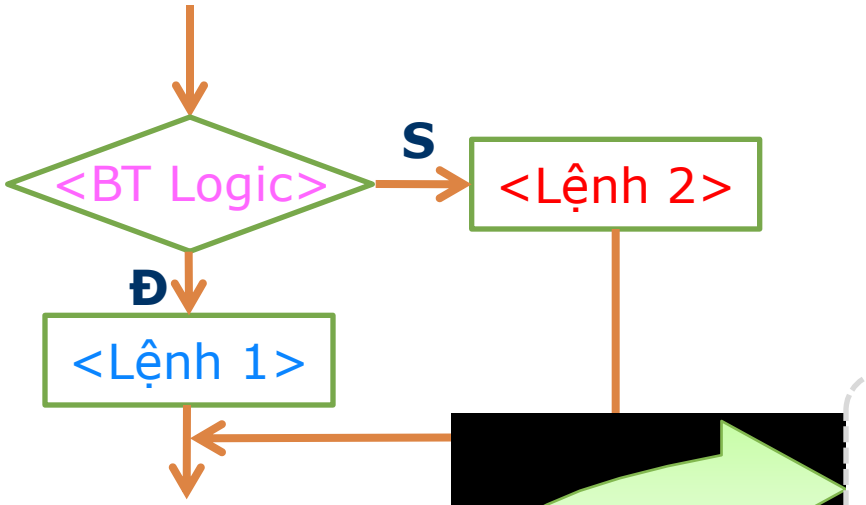
Trong ( ), cho kết quả (sai = 0, đúng ≠ 0)

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

Câu lệnh if (thiếu)

```
void main()
{
        if (a == 0)
                printf("a bang 0");

        if (a == 0)
        {
                printf("a bang 0");
                a = 2912;
        }
}
```

Câu lệnh if (đủ)




```
if (<BT Logic>)
        <Lệnh 1>
else
        <Lệnh 2>;
```

Trong ( ), cho kết quả (sai = 0, đúng ≠ 0)

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

Câu lệnh if (đủ)

```
void main()
{
        if (a == 0)
                printf("a bang 0");
        else
                printf("a khac 0");

        if (a == 0)
        {
                printf("a bang 0");
                a = 2912;
        }
        else
                printf("a khac 0");
}
```

Câu lệnh if - Một số lưu ý

- Câu lệnh if và câu lệnh if… else là một câu lệnh đơn.

```
{
        if (a == 0)
                printf("a bang 0");
}
{
        if (a == 0)
        {
                printf("a bang 0");
                a = 2912;
        }
        else
                printf("a khac 0");
}
```

Câu lệnh if - Một số lưu ý

- Câu lệnh if có thể lồng vào nhau và else sẽ tương ứng với if gần nó nhất.

```
if (a != 0)
        if (b > 0)
                printf("a != 0 va b > 0");
else
        printf("a != 0 va b <= 0");

if (a !=0)
{
        if (b > 0)
                printf("a != 0 va b > 0");
        else
                printf("a != 0 va b <= 0");
}
```

Câu lệnh if - Một số lưu ý

- Nên dùng else để loại trừ trường hợp.

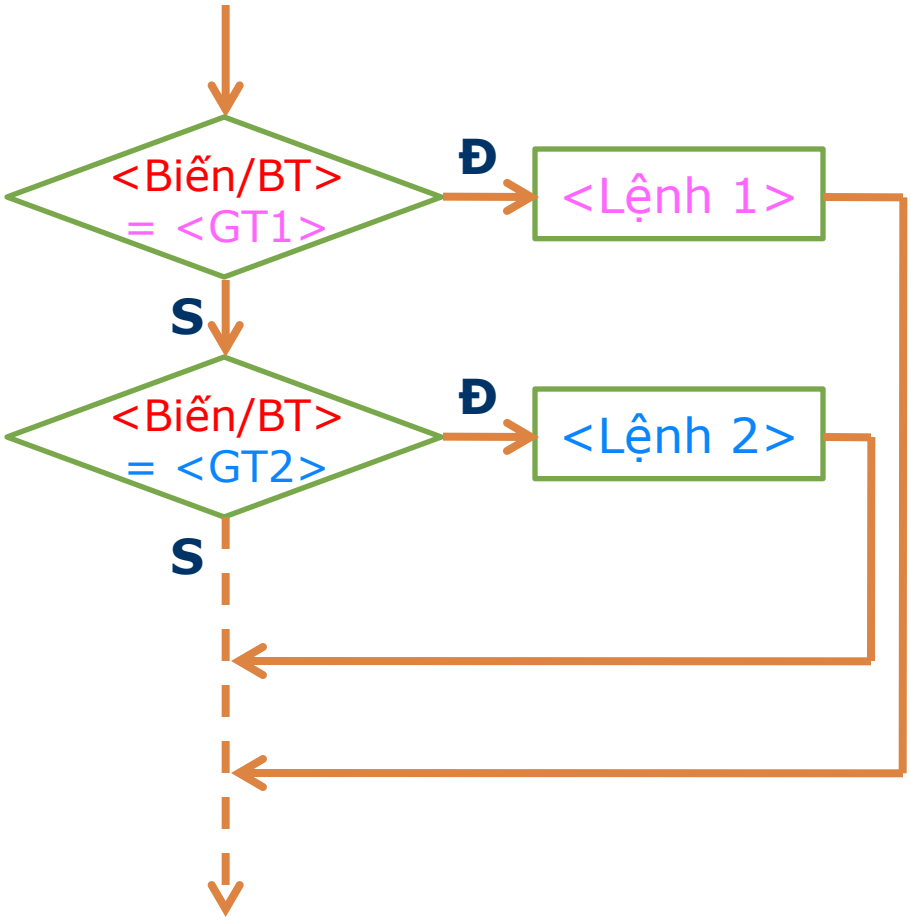
```
if (delta < 0)
      printf("PT vo nghiem");
if (delta == 0)
      printf("PT co nghiem kep");
if (delta > 0)
      printf("PT co 2 nghiem");

if (delta < 0)
      printf("PT vo nghiem");
else  // delta >= 0
      if (delta == 0)
            printf("PT co nghiem kep");
      else
            printf("PT co 2 nghiem");
```

Câu lệnh if - Một số lưu ý

- Không được thêm ; sau điều kiện của if.

```
void main()
{
        int a = 0;
        if (a != 0)
                printf("a khac 0.");

        if (a != 0);
                printf("a khac 0.");

        if (a != 0)
        {
        };
        printf("a khac 0.");
}
```

Câu lệnh switch (thiếu)



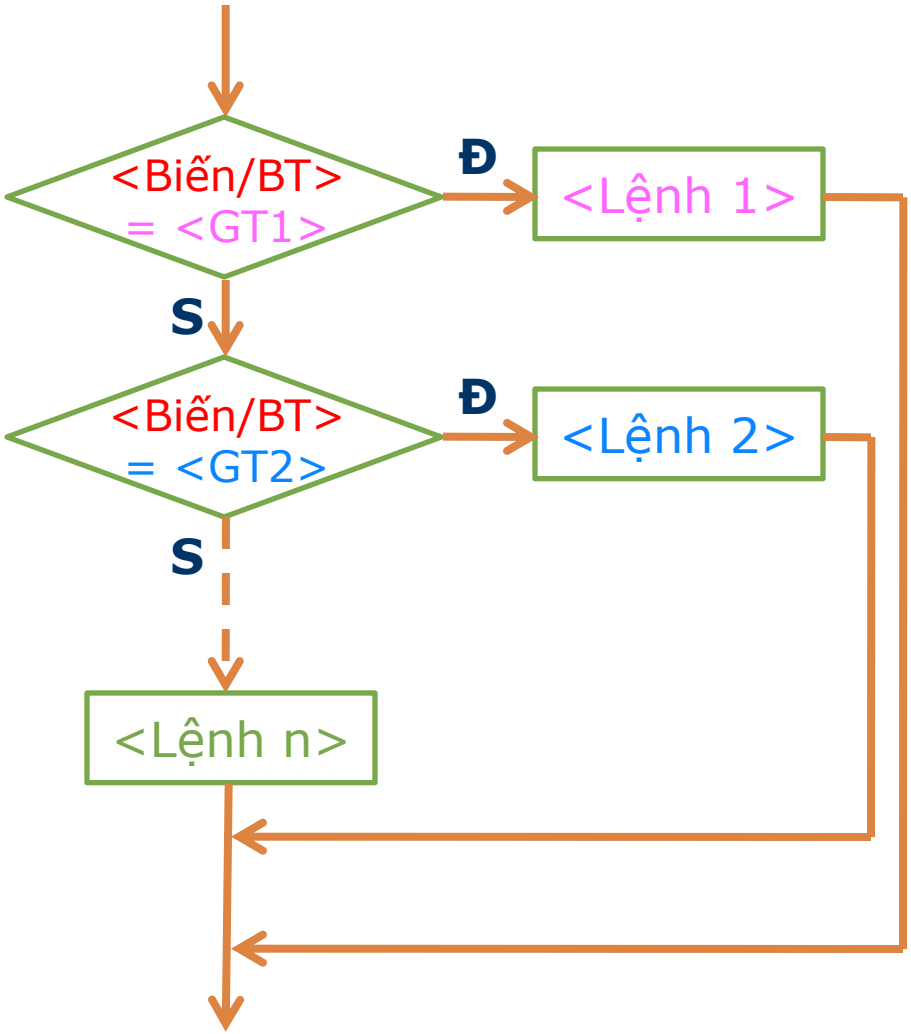

```
switch (<Biến/BT>)
{
   case <GT1>:<L1>;break;
   case <GT2>:<L2>;break;
   ...
}
```

- <Biến/BT> là biến/biểu thức cho giá trị rời rạc.
- <Lệnh> : đơn hoặc khối lệnh {}.

Câu lệnh switch (thiếu)

```
void main()
{
        int a;
        printf("Nhap a: ");
        scanf("%d", &a);

        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : printf("Hai"); break;
                case 3 : printf("Ba"); break;
        }
}
```

Câu lệnh switch (đủ)




```
switch (<Biến/BT>)
{
    case <GT1>:<L1>;break;
    case <GT2>:<L2>;break;
    ...
    default:
        <Lệnh n>;
}
```

Câu lệnh switch (đủ)

```
void main()
{
        int a;
        printf("Nhap a: ");
        scanf("%d", &a);

        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : printf("Hai"); break;
                case 3 : printf("Ba"); break;
                default : printf("Ko biet doc");
        }
}
```

Câu lệnh switch - Một số lưu ý

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

```
{
        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : switch (b)
                         {
                            case 1 : printf("A"); break;
                            case 2 : printf("B"); break;
                         } break;
                case 3 : printf("Ba"); break;
                default : printf("Khong biet doc");
        }
}
```

Câu lệnh switch - Một số lưu ý

- Các giá trị trong mỗi trường hợp phải khác nhau.

```
switch (a)
{
        case 1 : printf("Mot"); break;
        case 1 : printf("MOT"); break;
        case 2 : printf("Hai"); break;
        case 3 : printf("Ba"); break;
        case 1 : printf("1"); break;
        case 1 : printf("mot"); break;
        default : printf("Khong biet doc");
}
```

Câu lệnh switch - Một số lưu ý

- switch sẽ nhảy đến case tương ứng và thực hiện đến khi nào gặp break hoặc cuối switch sẽ kết thúc.

```
switch (a)
{
        case 1 : printf("Mot"); break;
        case 2 : printf("Hai"); break;
        case 3 : printf("Ba"); break;
}
```

Câu lệnh switch - Một số lưu ý

- switch nhảy đến case tương ứng và thực hiện đến khi nào gặp break hoặc cuối switch sẽ kết thúc.

```
switch (a)
{
        case 1 : printf("Mot"); break;
        case 2 : printf("Hai"); break;
        case 3 : printf("Ba"); break;
}
switch (a)
{
        case 1 : printf("Mot"); break;
        case 2 : printf("Hai"); break;
        case 3 : printf("Ba"); break;
}
```

Câu lệnh switch - Một số lưu ý

- Tận dụng tính chất khi bỏ break;

```
switch (a)
{
        case 1 : printf("So le"); break;
        case 2 : printf("So chan"); break;
        case 3 : printf("So le"); break;
        case 4 : printf("So chan"); break;
}
switch (a)
{
        case 1 :
        case 3 : printf("So le"); break;
        case 2 :
        case 4 : printf("So chan"); break;
}
```

Kinh nghiệm lập trình

Câu lệnh if

```
if (a == 1)
   printf("Mot");
if (a == 2)
   printf("Hai");
if (a == 3)
   printf("Ba");
if (a == 4)
   printf("Bon");
if (a == 5)
   printf("Nam");
```

Câu lệnh switch

```
switch (a)
{
   case 1:  printf("Mot");
            break;
   case 2:  printf("Hai");
            break;
   case 3:  printf("Ba");
            break;
   case 4:  printf("Bon");
            break;
   case 5:  printf("Nam");
}
```

Kinh nghiệm lập trình

Câu lệnh switch

```
switch (a)
{
case 3.14:
case <10:
case 1: printf("OK");
        break;
case 2:
case 3: printf("OK");
        break;
}
```

Câu lệnh if

```
if (a == 3.14)
    printf("OK");
if (a < 10)
    printf("OK");
if (a == 1)
    printf("OK");
if (a == 2 || a == 3)
    printf("OK");
```

Bài tập

1. Nhập một số bất kỳ. Hãy đọc giá trị của số nguyên đó nếu nó có giá trị từ 1 đến 9, ngược lại thông báo không đọc được.
2. Nhập một chữ cái. Nếu là chữ thường thì đổi sang chữ hoa, ngược lại đổi sang chữ thường.
3. Giải phương trình bậc nhất $ax + b = 0$.
4. Giải phương trình bậc hai $ax^2 + bx + c = 0$.

Bài tập

5. Nhập 4 số nguyên a, b, c và d. Tìm số có giá trị nhỏ nhất (min).
6. Nhập 4 số nguyên a, b, c và d. Hãy sắp xếp giá trị của 4 số nguyên này theo thứ tự tăng dần.
7. Tính tiền đi taxi từ số km nhập vào. Biết:
   a. 1 km đầu giá 15000đ
   b. Từ km thứ 2 đến km thứ 5 giá 13500đ
   c. Từ km thứ 6 trở đi giá 11000đ
   d. Nếu trên 120km được giảm 10% tổng tiề

Bài tập

8. Nhập vào tháng và năm. Cho biết tháng đó có bao nhiêu ngày.
9. Nhập độ dài 3 cạnh 1 tam giác. Kiểm tra đó có phải là tam giác không và là tam giác gì?

Bài tập 1 (if)

```
#include <stdio.h>

void main()
{
        int n;
        printf("Nhap mot so nguyen: ");
        scanf("%d", &n);
        if (n == 1)
                printf("Mot");
        else
                if (n == 2)
                        printf("Hai");
                …
                else
                        printf("Khong biet doc");
}
```

Bài tập 1 (switch)

```
#include <stdio.h>

void main()
{
        int n;
        printf("Nhap mot so nguyen: ");
        scanf("%d", &n);
        switch (n)
        {
                case 1: printf("Mot"); break;
                case 2: printf("Hai"); break;
                case 3: printf("Ba"); break;
                …
                default: printf("Ko biet doc");
        }
}
```

Bài tập 2

```
#include <stdio.h>

void main()
{
	char ch;
	printf("Nhap mot ky tu: ");
	scanf("%c", &ch);

	if (ch >= 'a' && ch <= 'z')
		ch = ch - 32;
	else
		if (ch >= 'A' && ch <= 'Z')
			ch = ch + 32;

	printf("Ky tu sau khi doi: %c", ch);
}
```

Bài tập 3

```
#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("Phuong trinh VSN");
		else
			printf("Phuong trinh VN");
	else
		printf("Nghiem = %f", float(-b)/a);
}
```

Bài tập 4

```
#include <stdio.h>

void main()
{
	int a, b, c;
	printf("Nhap a, b, c: ");
	scanf("%d%d%d", &a, &b, &c);
	if (a == 0)
	{
		// Giai PT Bac 1 o day
	}
	else
	{
		// Giai PT Bac 2 o day
	}
}
```

Bài tập 5

```
#include <stdio.h>

void main()
{
        int a, b, c, d, min;
        printf("Nhap a, b, c, d: ");
        scanf("%d%d%d%d", &a, &b, &c, &d);

        min = a;
        if (b < min) min = b;
        if (c < min) min = c;
        if (d < min) min = d;

        printf("So nho nhat la %d", min);
}
```

Bài tập 6

```
#include <stdio.h>

void main()
{
        int a, b, c, d, tam;

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

        if (a > b)
        { tam = a; a = b; b = tam; }
        …
        printf("Cac so theo thu tu tang dan: ");
        printf("%d %d %d %d", a, b, c, d);
}
```

Bài tập 7

- Nên khai báo hằng số lưu giá tiền và km
  - #define G1 15000
  - #define G2 13500
  - #define G3 11000
- Cách tính tiền dựa trên số km n
  - n = 1 ➔ T = G1
  - 2 ≤ n ≤ 5 ➔ T = G1 + (n – 1)*G2;
  - n > 5 ➔ T = G1 + 4*G2 + (n – 1 – 4)*G3;
- n > 120 ➔ T = T*0.9;

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

NHẬP MÔN LẬP TRÌNH

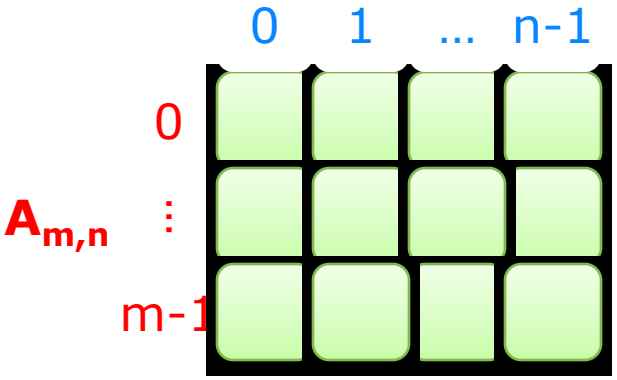
Đặng Bình Phương
dbphuong@fit.hcmuns.edu.vn

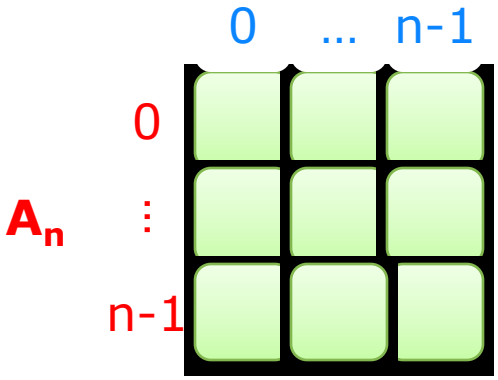
MẢNG HAI CHIỀU

Nội dung

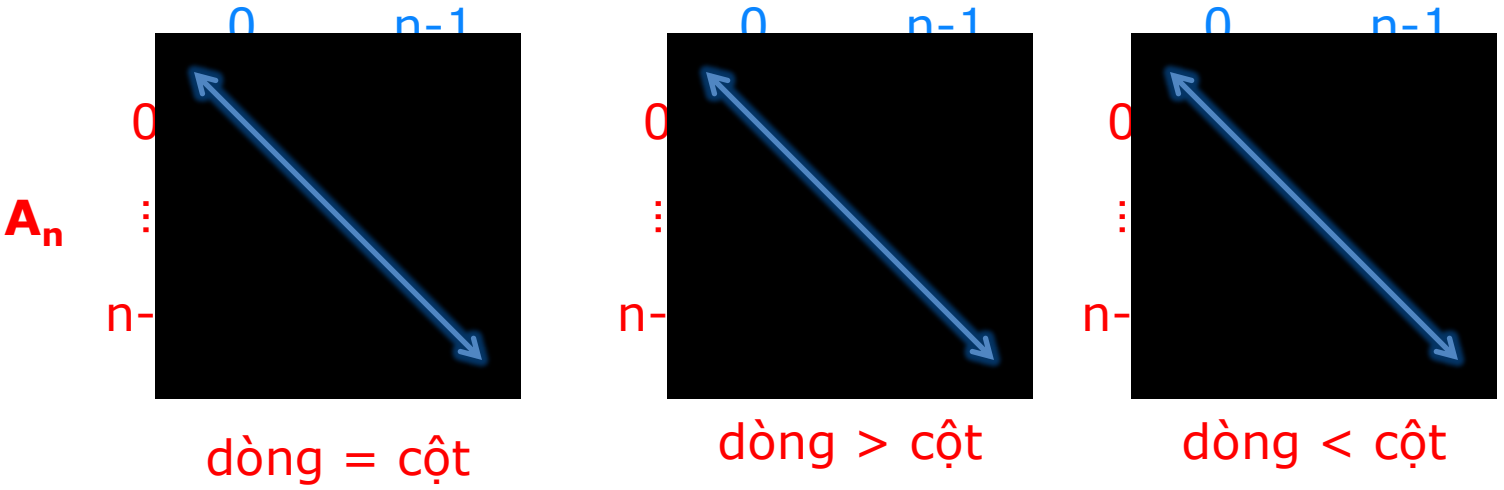
1. Khái niệm
2. Khai báo
3. Truy xuất dữ liệu kiểu mảng
4. Một số bài toán trên mảng 2 chiều

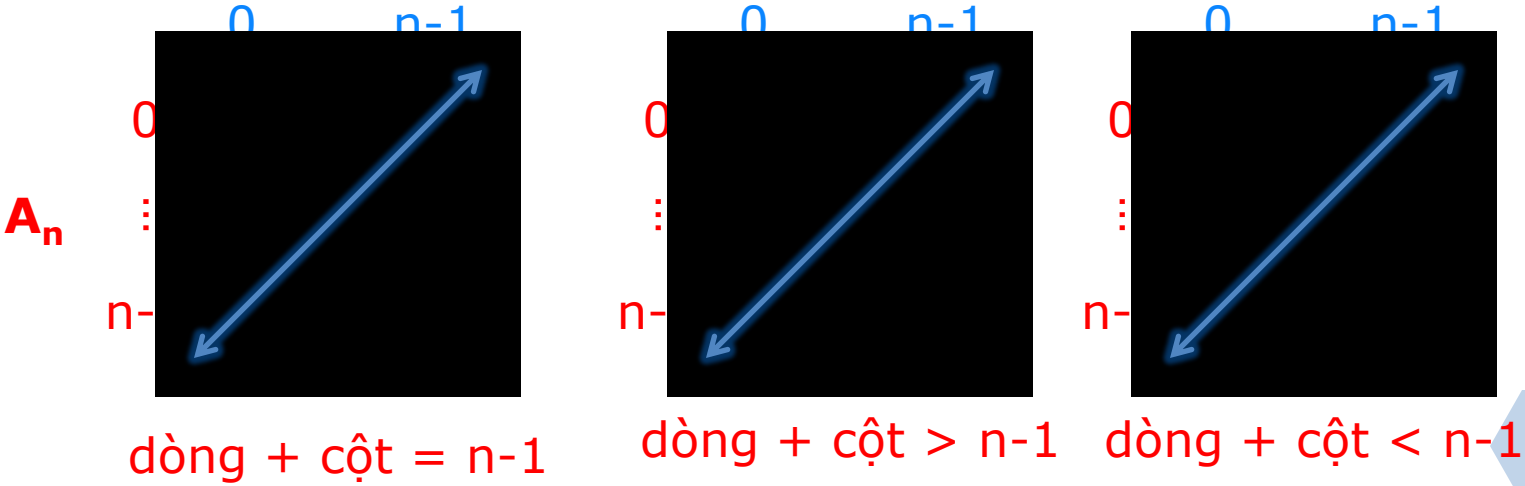
Ma Trận





Ma Trận


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


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

Khai báo kiểu mảng 2 chiều

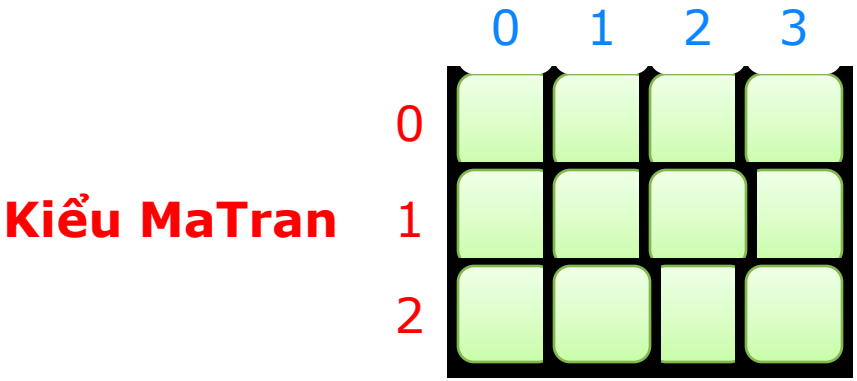
- Cú pháp

```
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;
```

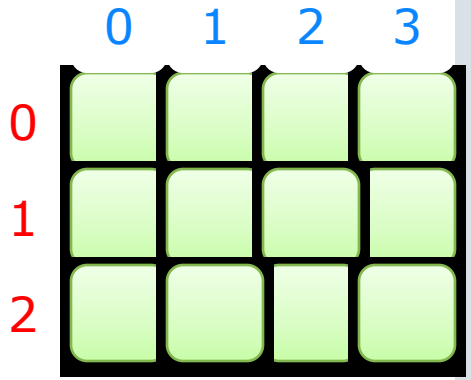
Truy xuất đến một phần tử

- 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

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

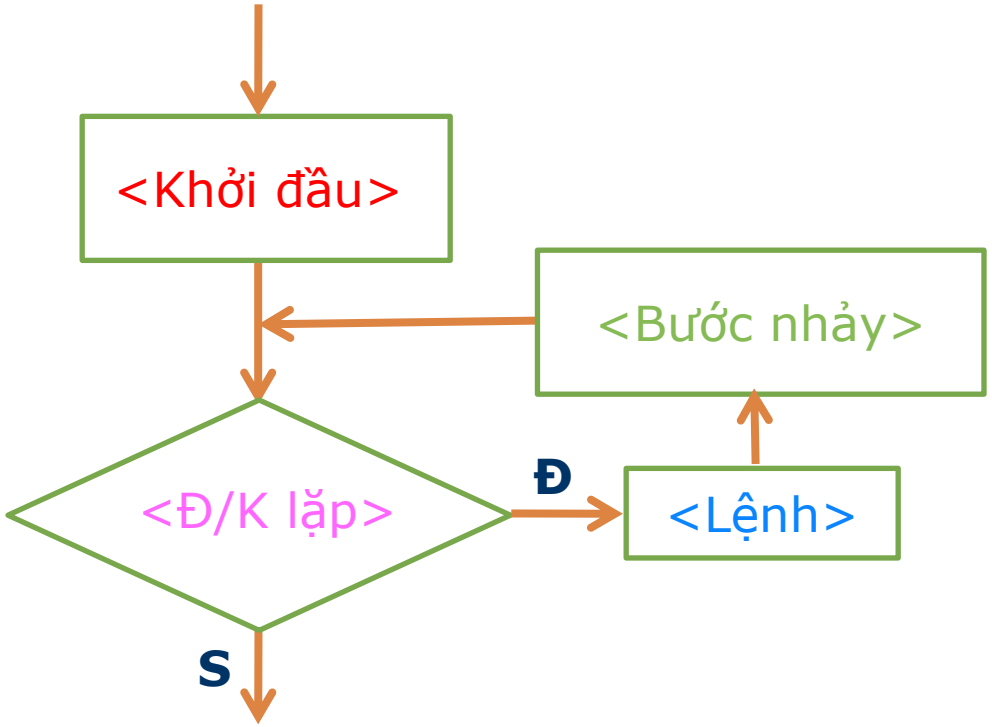
Đặt vấn đề

- 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

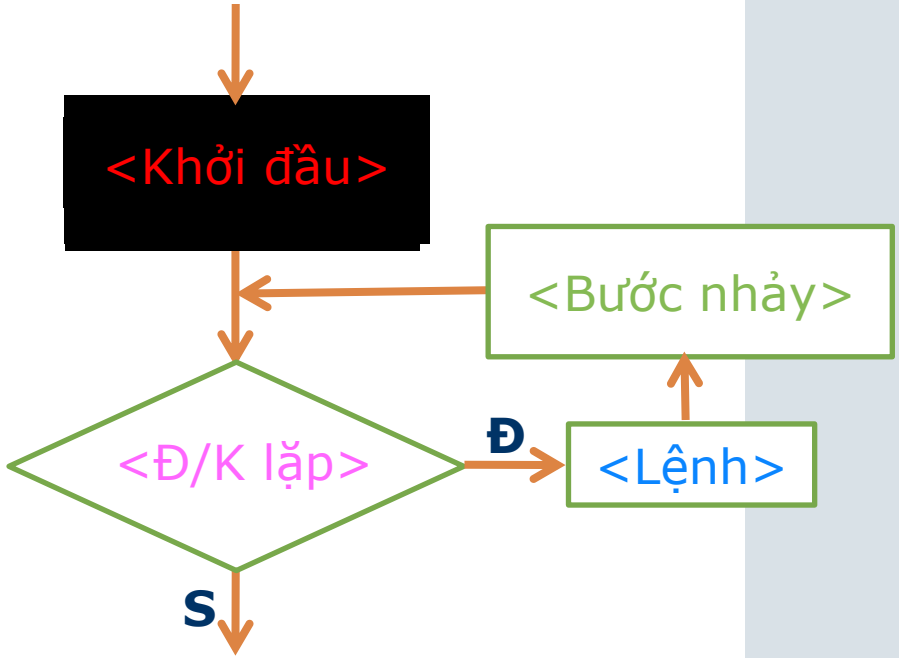
- Câu lệnh for là một câu lệnh đơn và có thể lồng nhau.

```
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>

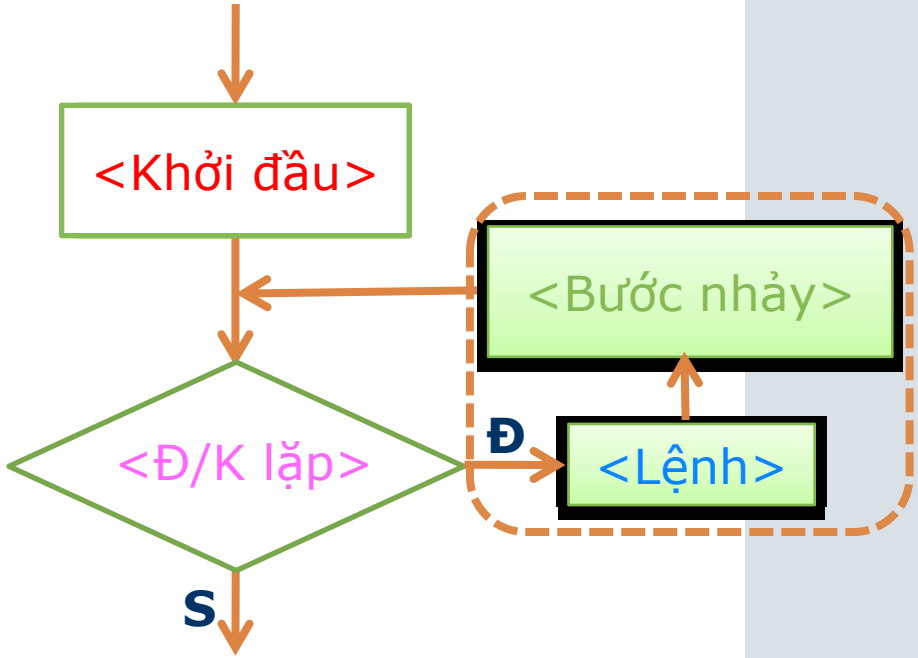
```
int i;
for (i = 0; i < 10; i++)
      printf("%d\n", i);

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.

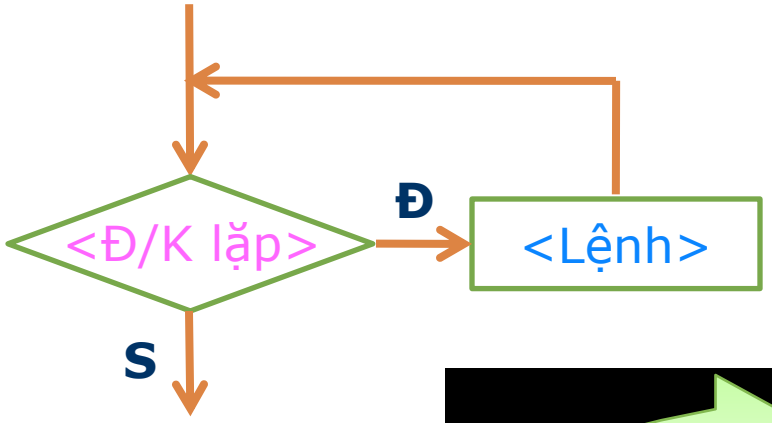
```
for (i = 0; i < 10; i++);
{
      printf("%d", i);
      printf("\n");
}

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.

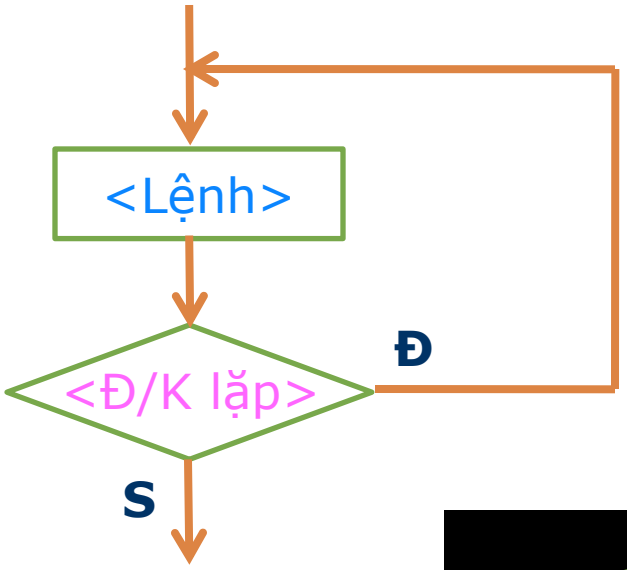
```
int n = 0;
while (n < 10);
{
        printf("%d\n", n);
        n++;
}

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

Câu lệnh while - Một số lưu ý

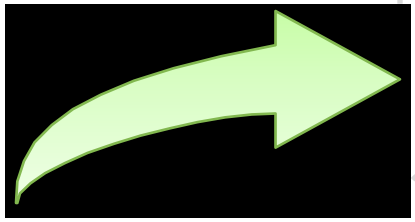
- Câu lệnh while có thể bị lặp vô tận (loop)

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

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

Câu lệnh do… while

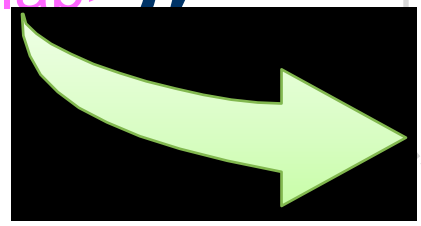


do



<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.hcmuns.edu.vn

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

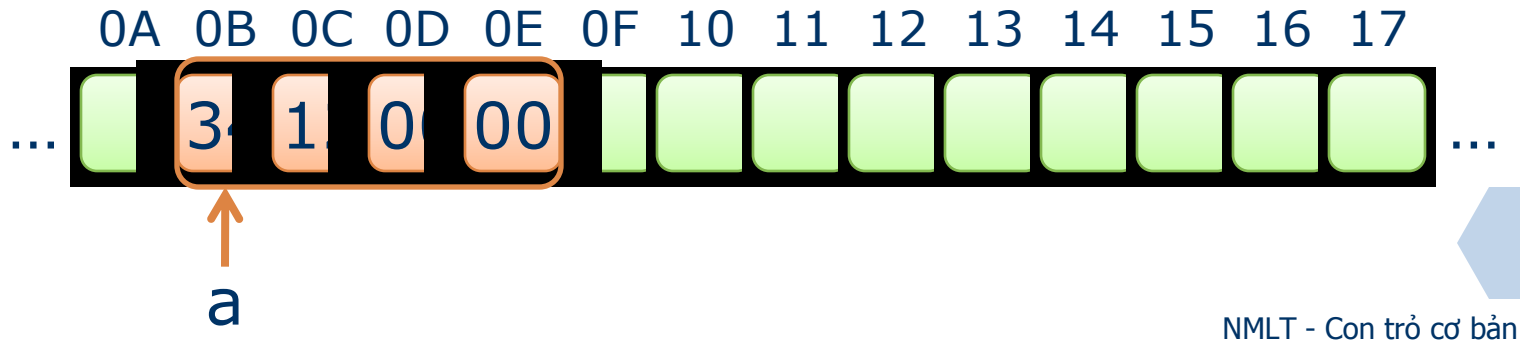
Kiến trúc máy tính

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 $2^{31} - 1$

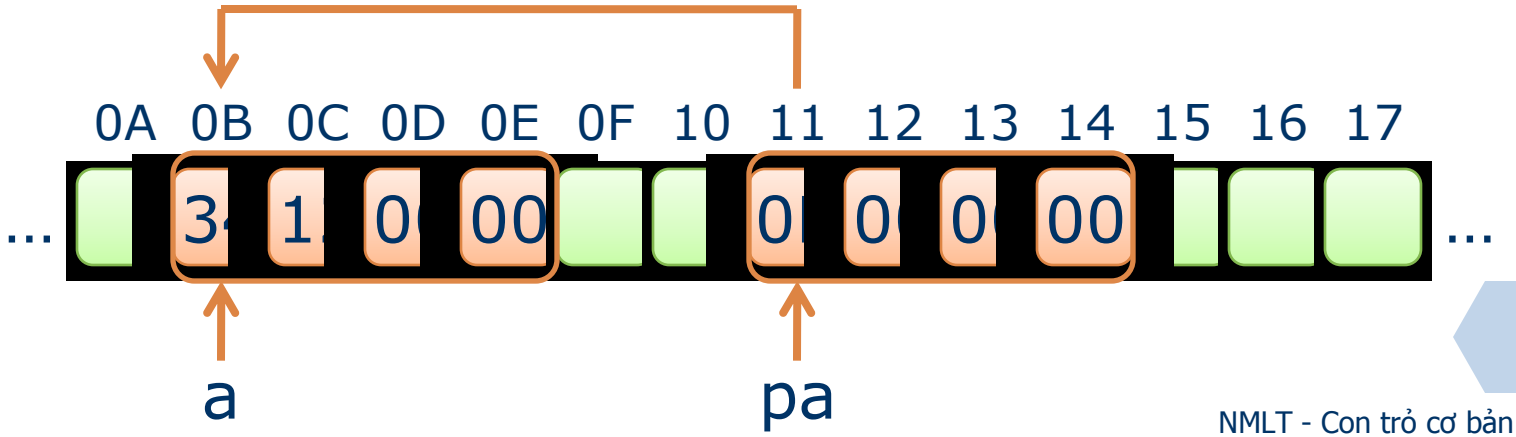
Khai báo biến trong C

- 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;
```

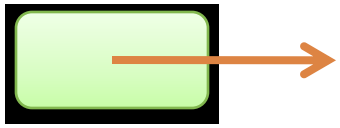
- Lưu ý khi khai báo kiểu dữ liệu mới
  - Giảm bối rối khi mới tiếp xúc với con trỏ.
  - Nhưng dễ nhầm lẫn với biến thường.

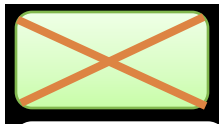
Con trỏ NULL

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 varialbe
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>;
```

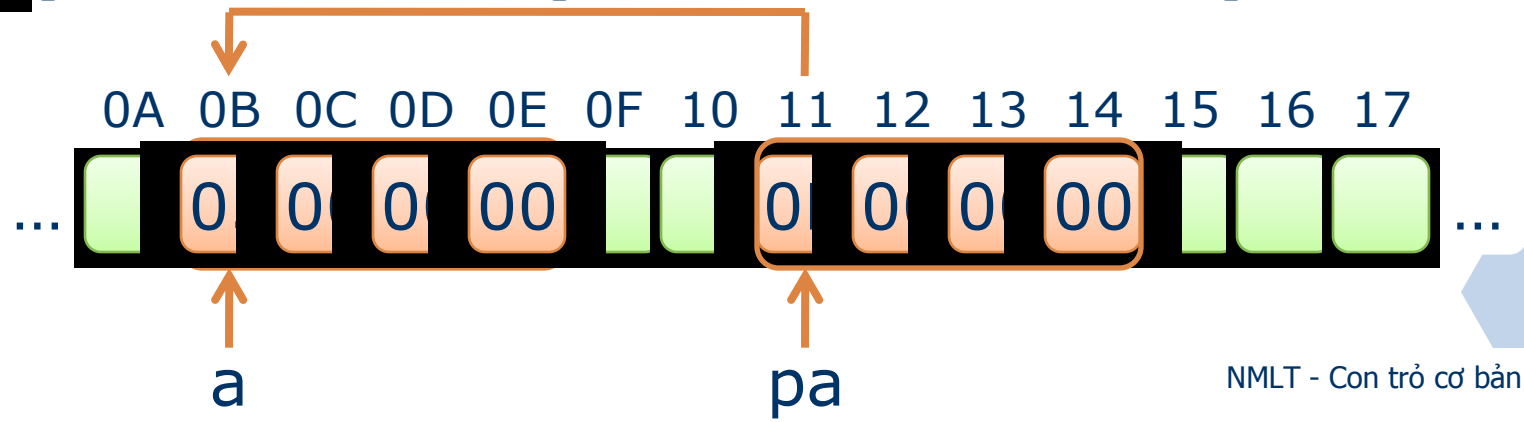
- Ví dụ

```
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;
…
```

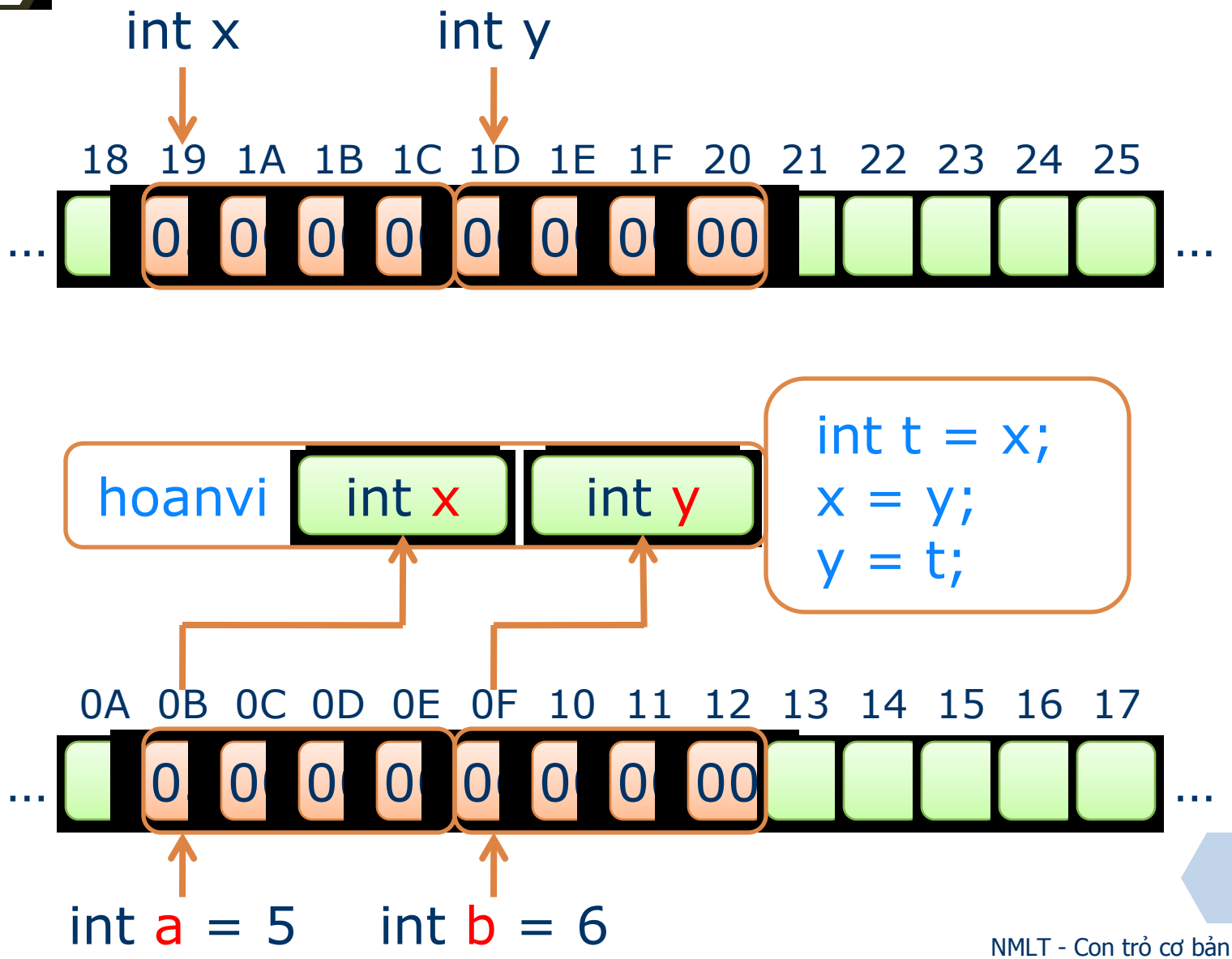
- Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau:
  - Môi trường MD-DOS (16 bit): 2 bytes
  - Môi trường Windows (32 bit): 4 bytes

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)

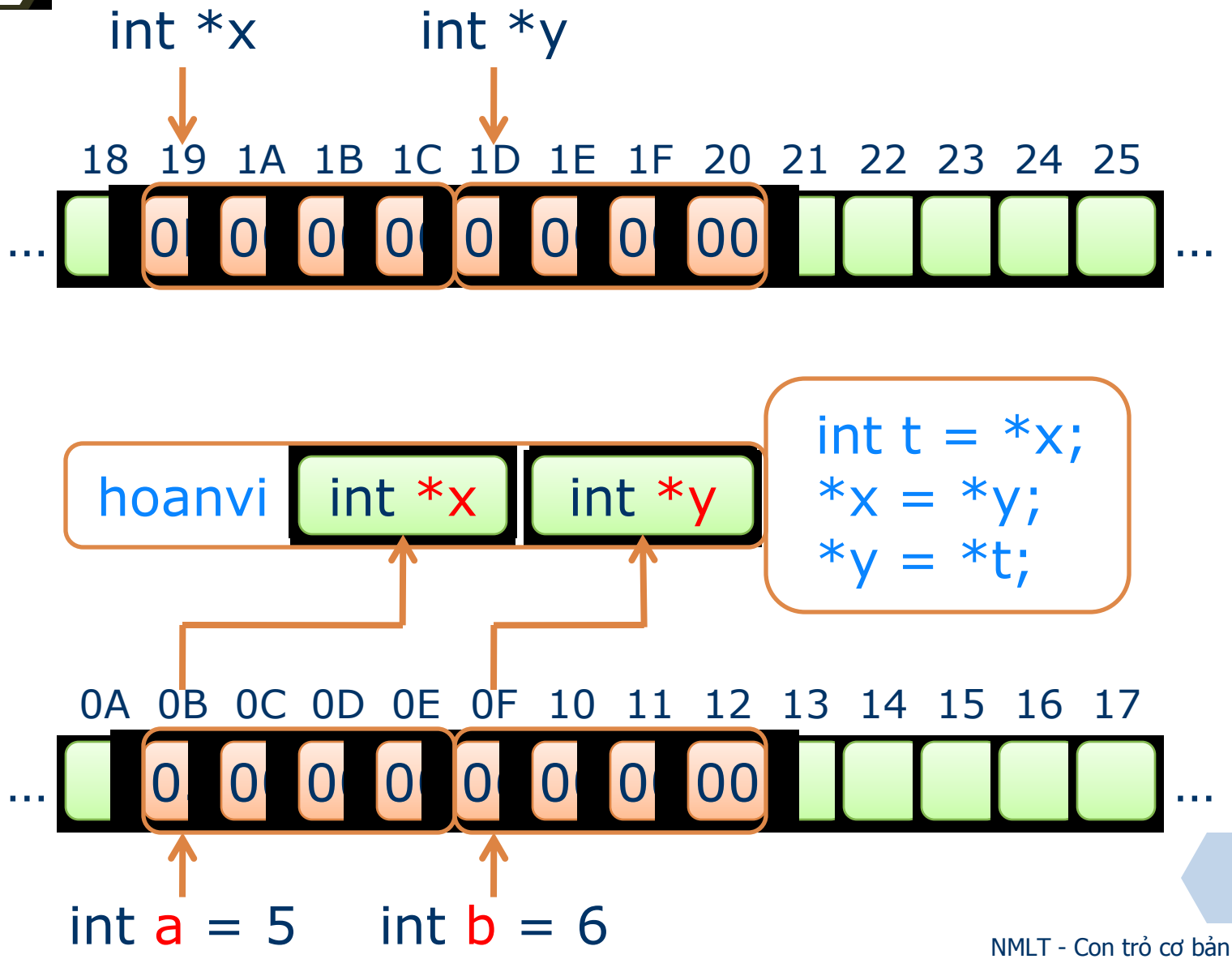


Các cách truyền đối số

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)

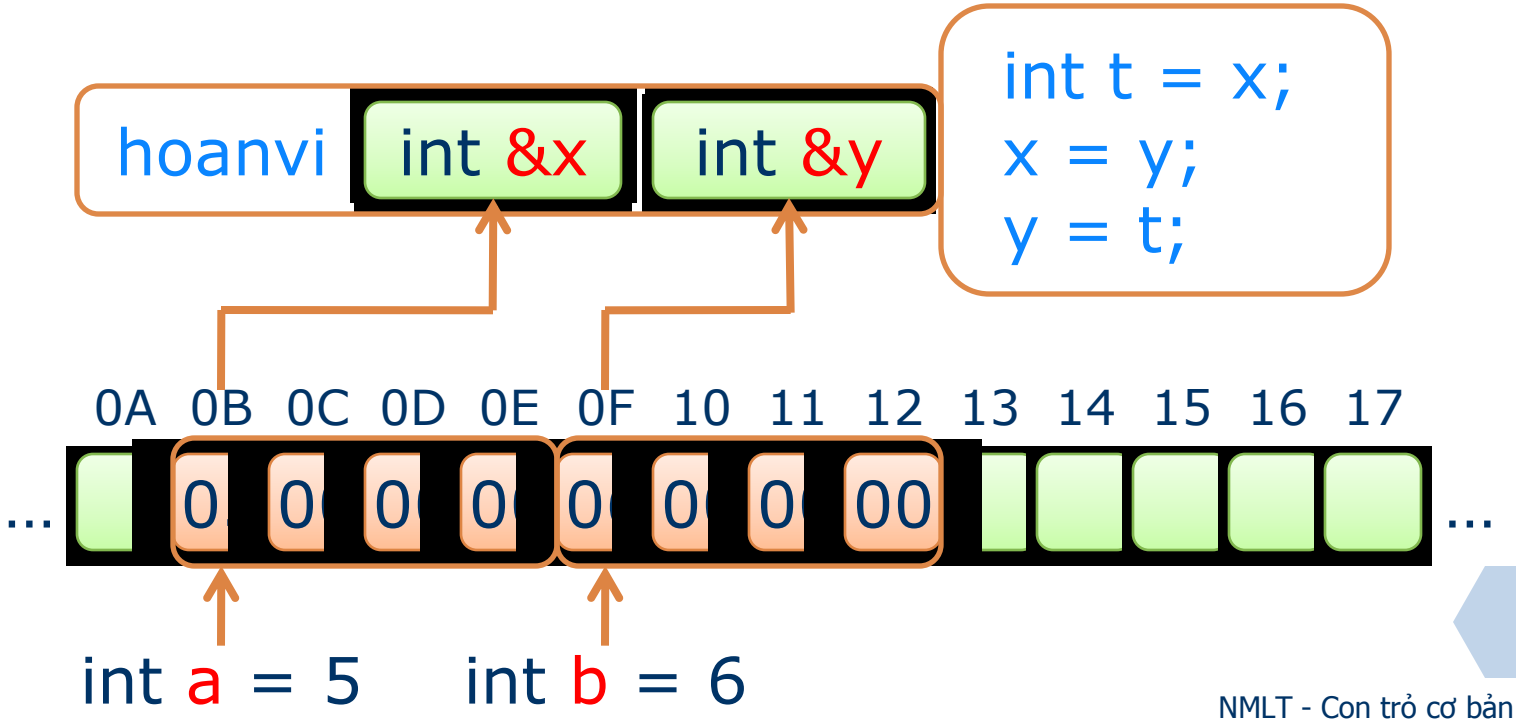


Các cách truyền đối số

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;
}
```

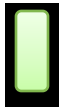
Truyền tham chiếu (C++)



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

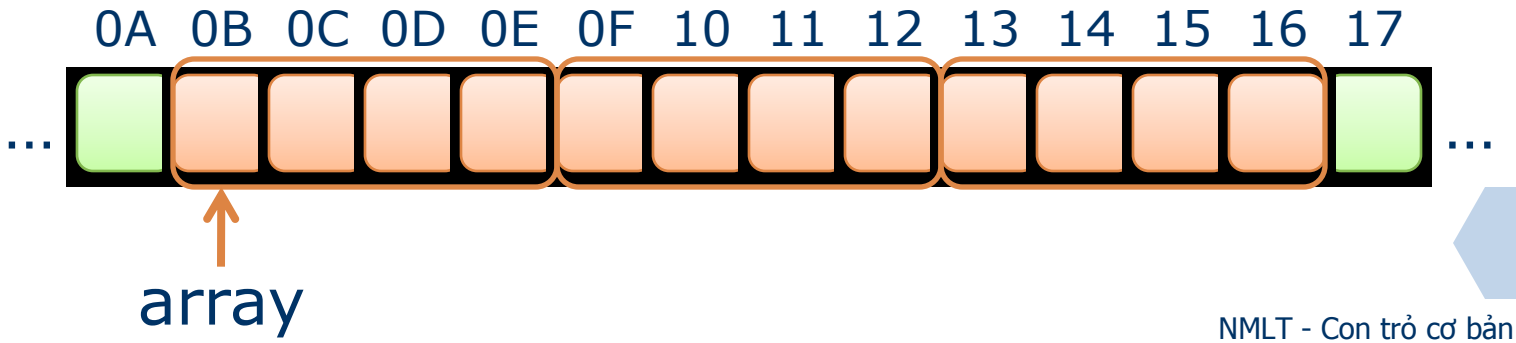
```
int array[3];
```

- Tên mảng array là một hằng con trỏ

  ➔ không thể thay đổi giá trị của hằng này.

- array là địa chỉ đầu tiên của mảng

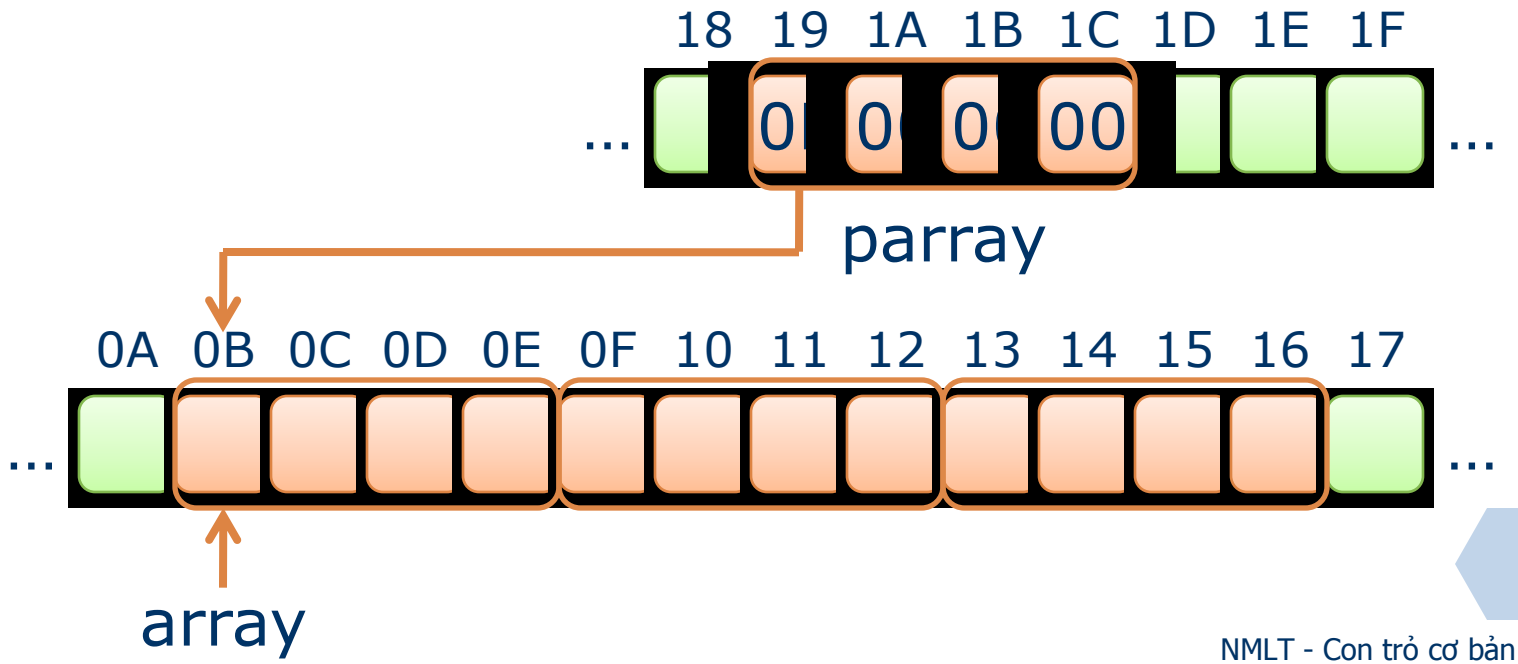
  ➔array == &array[0]



Con trỏ và mảng một chiều

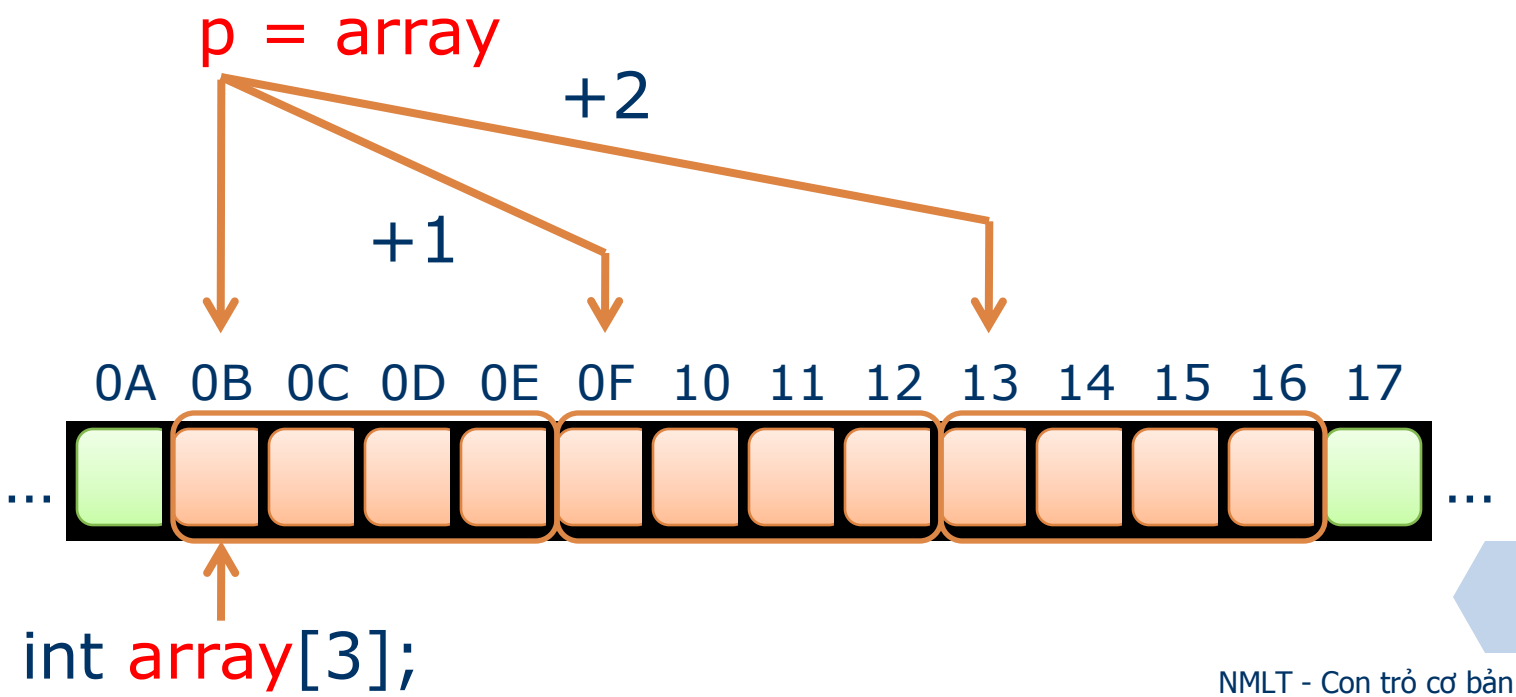
Con trỏ đến mảng một chiều

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

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



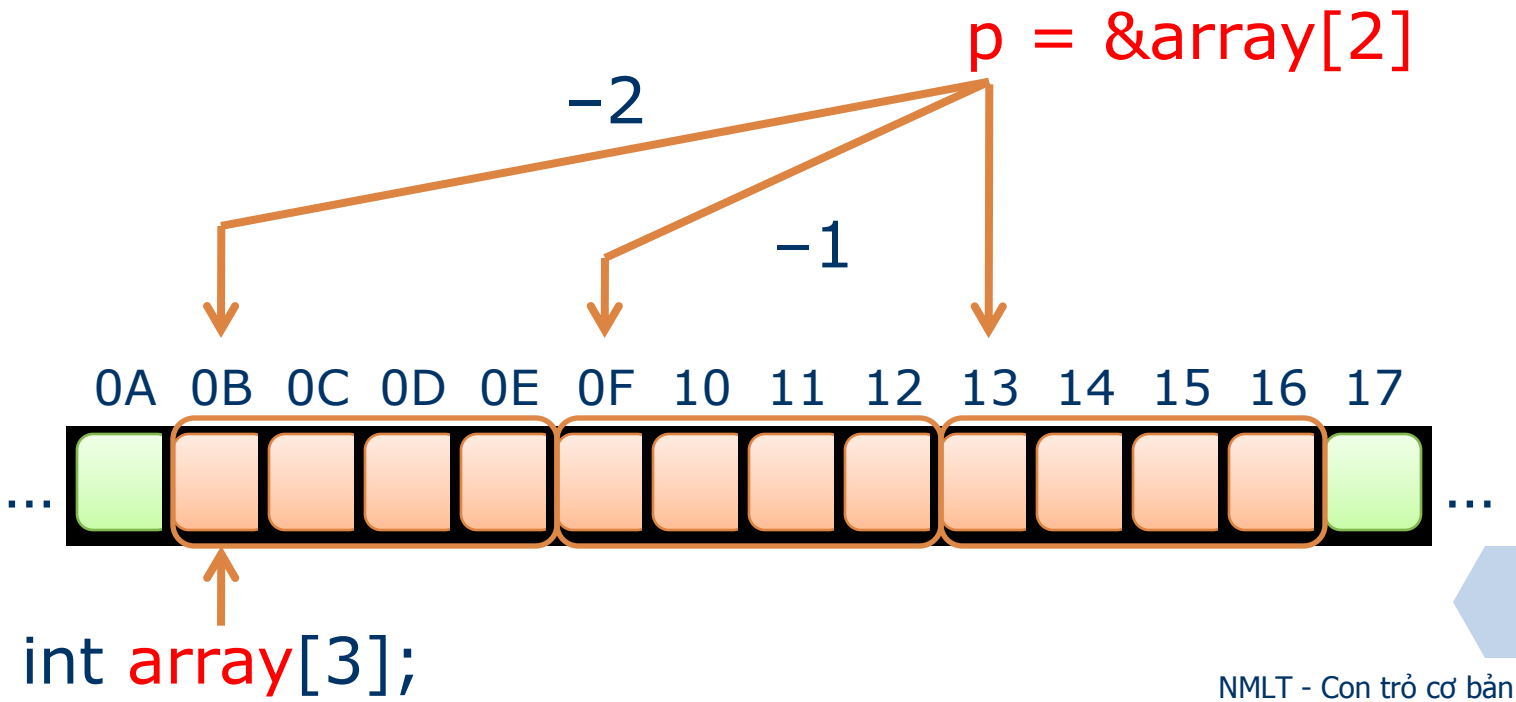
Phép toán số học trên con trỏ

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



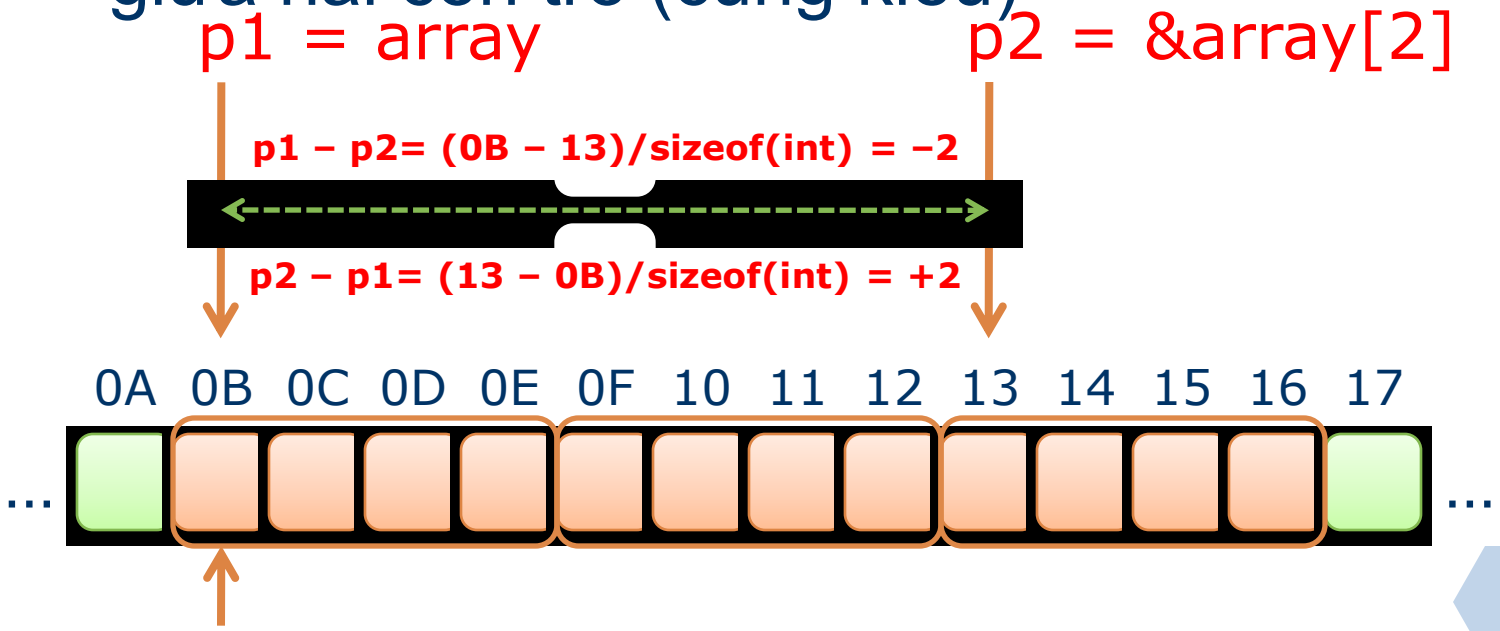
Phép toán số học trên con trỏ

- 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)

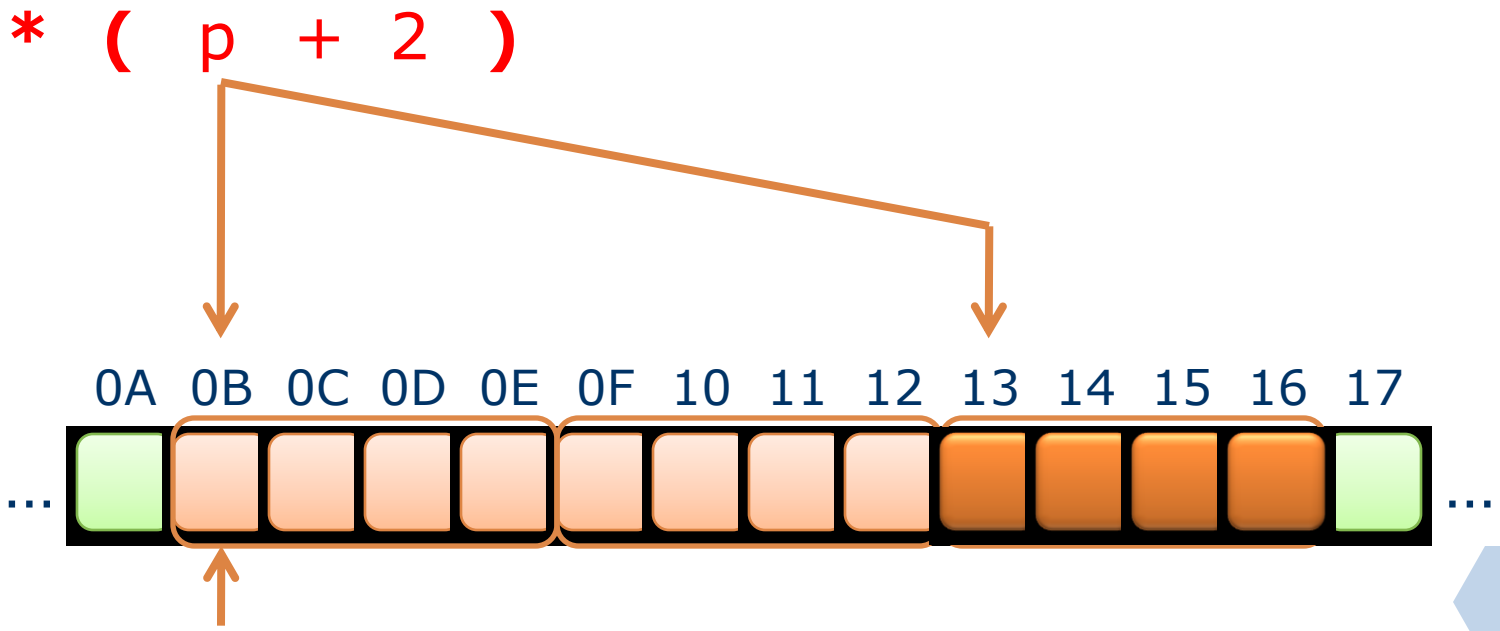


Phép toán số học trên con trỏ

- 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 (không sử dụng biến mảng)
  - 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", &p[i]);
                scanf("%d", a + i);
                scanf("%d", p + i);
                scanf("%d", a++);
                scanf("%d", p++);
}
```

➔ &a[i] ⇔ (a + i) ⇔ (p + i) ⇔ &p[i]

Con trỏ và mảng một chiều

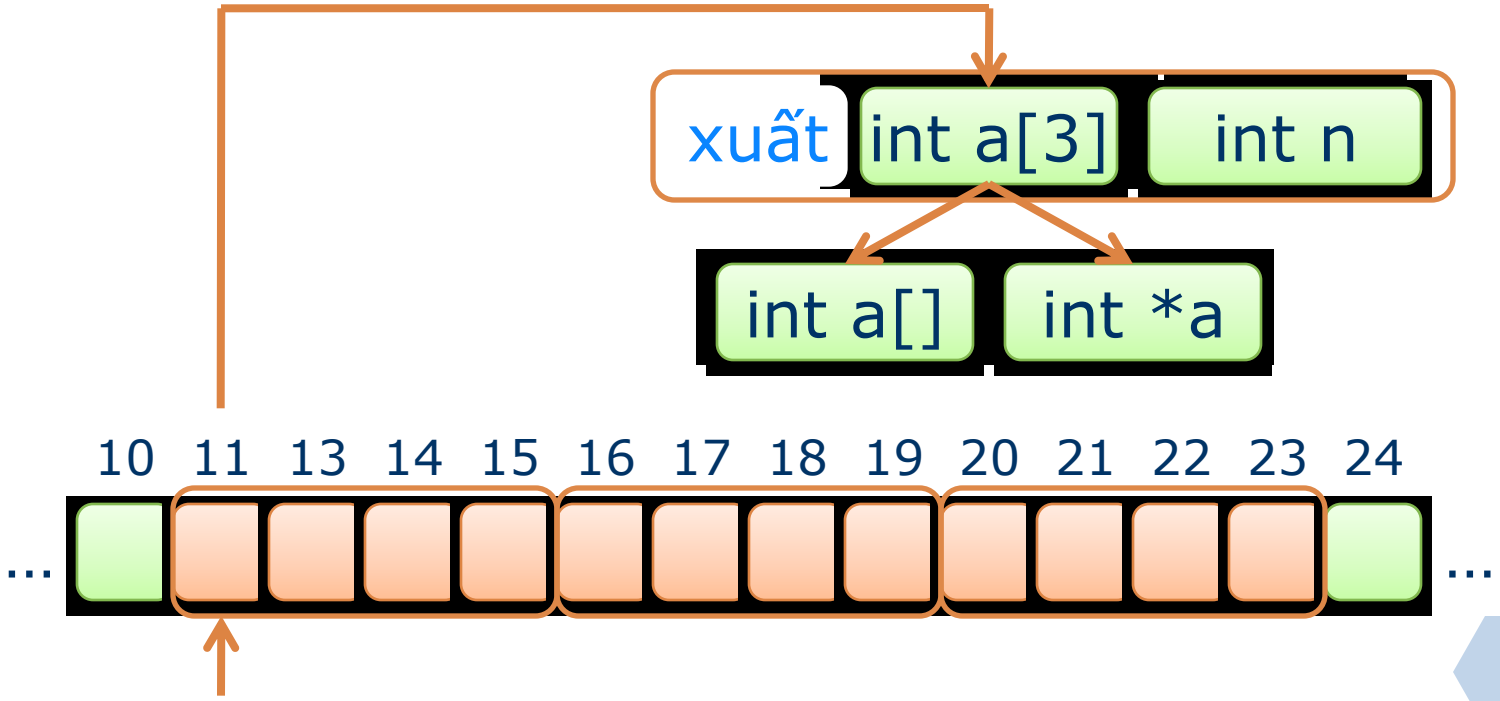
Ví dụ xuất mảng

```
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", p[i]);
              printf("%d", *(a + i));
              printf("%d", *(p + i));
              printf("%d", *(a++));
              printf("%d", *(p++));
}
```

➔ a[i] ⇔ *(a + i) ⇔ *(p + i) ⇔ p[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.



Con trỏ và mảng một chiều

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 nhân, chia, lấy phần dư.
  - 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>)
  - Không thể tăng/giảm biến mảng. Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm nó.
  - Đối số mảng một chiều truyền cho hàm là địa chỉ phần tử đầu tiên của mảng.

Con trỏ 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

```
struct PHANSO
{
        int tu, mau;
};
PHANSO ps1, *ps2 = &p1; // ps2 là con trỏ

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

Con trỏ cấu trúc

Gán hai cấu trúc

```
struct PHANSO
{
        int tu, mau;
};
PHANSO ps1, *ps2;

ps1.tu = 1; ps1.mau = 2;        // ps1 = 1/2

ps2 = &ps1;
ps2->tu = 3; ps2->mau = 4;      // ps1 = 3/4
```

Bài tập lý thuyết

- 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 lý thuyết

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 lý thuyết

- Bài 1: Toán tử nào dùng để xác định địa chỉ của một biến?
- Bài 2: Toán tử nào dùng để xác định giá trị của biến do con trỏ trỏ đến?
- Bài 3: Phép lấy giá trị gián tiếp là gì?
- Bài 4: Các phần tử trong mảng được sắp xếp trong bộ nhớ như thế nào?
- Bài 5: Cho mảng một chiều data. Trình bày 2 cách lấy địa chỉ phần tử đầu tiên của mảng này.

Bài tập lý thuyết

- Bài 6: Nếu ta truyền cho hàm đối số là mảng một chiều. Trình bày hai cách nhận biết phần tử cuối của mảng?
- Bài 7: Trình bày 6 phép toán có thể thực hiện trên con trỏ?
- Bài 8: Cho con trỏ p1 trỏ đến phần tử thứ 3 còn con trỏ p2 trỏ đến phần tử thứ 4 của mảng int. p2 – p1 = ?
- Bài 9: Giống như câu trên nhưng đối với mảng float?

Bài tập

- Bài 10: Trình bày khai báo con trỏ pchar trỏ đến kiểu char.
- Bài 11: Cho biến cost kiểu int. Khai báo và khởi tạo con trỏ pcost trỏ đến biến này.
- Bài 12: Gán giá trị 100 cho biến cost sử dụng hai cách trực tiếp và gián tiếp.
- Bài 13: In giá trị của con trỏ và giá trị của biến mà nó trỏ tới.
- Bài 14: Sử dụng con trỏ để làm lại các bài tập về mảng một chiều.

Bài tập lý thuyết

❖ Bài 15: Cho đoạn chương trình sau:

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

Hãy chọn phát biểu sai cú pháp:

a. a = *pint;

b. c = *pd;

c. *pint = *pd;

d. pd = a;

Bài tập thực hành

- Bài 16: Viết chương trình nhập số nguyên dương n gồm k chữ số (0 < k ≤ 5) , sắp xếp các chữ số của n theo thứ tự tăng dần.

  Ví dụ:

  - Nhập n = 1536
  - Kết quả sau khi sắp xếp: 1356.

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.

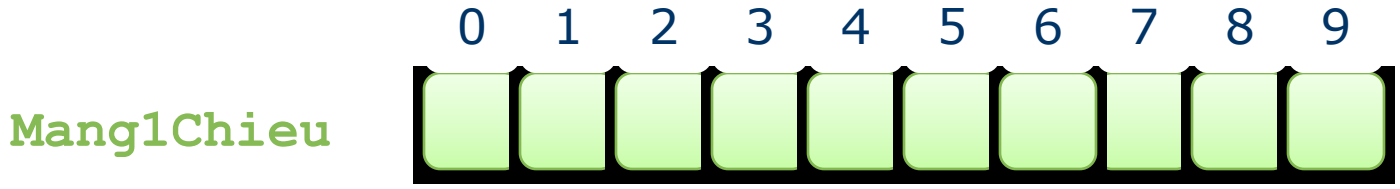
Lưu ý

- 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

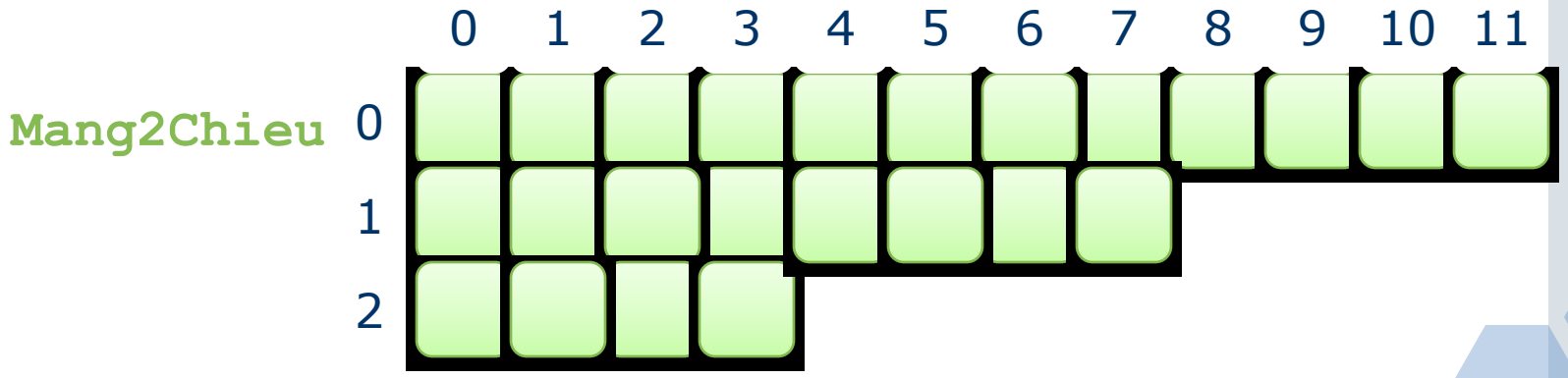
Khai báo biến mảng (tường minh)

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];
```

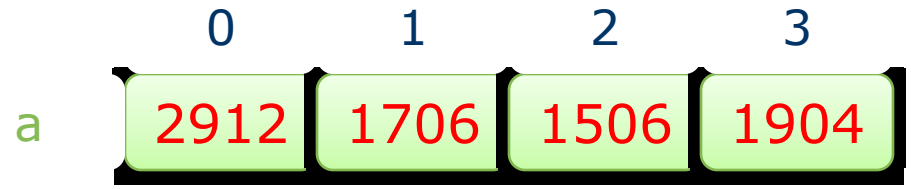
- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#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

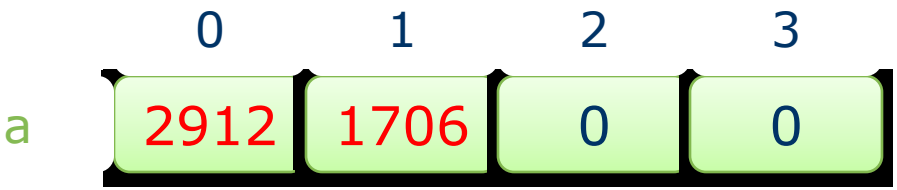
- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

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



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

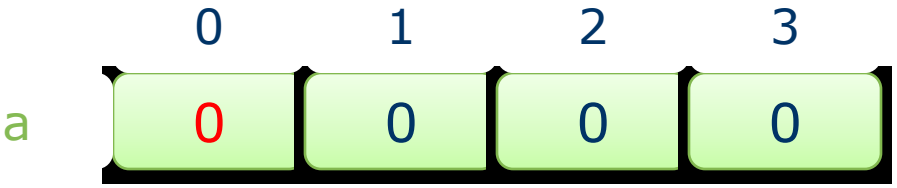
```
int a[4] = {2912, 1706};
```



Khởi tạo giá trị cho mảng lúc khai báo

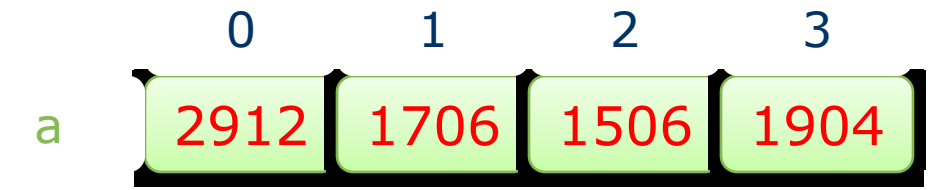
- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

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



  - Tự động xác định số lượng phần tử

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



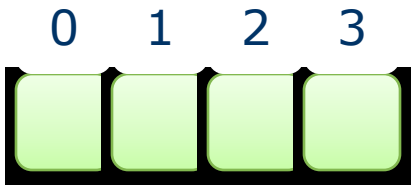
Truy xuất đến một phần tử

- 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
```

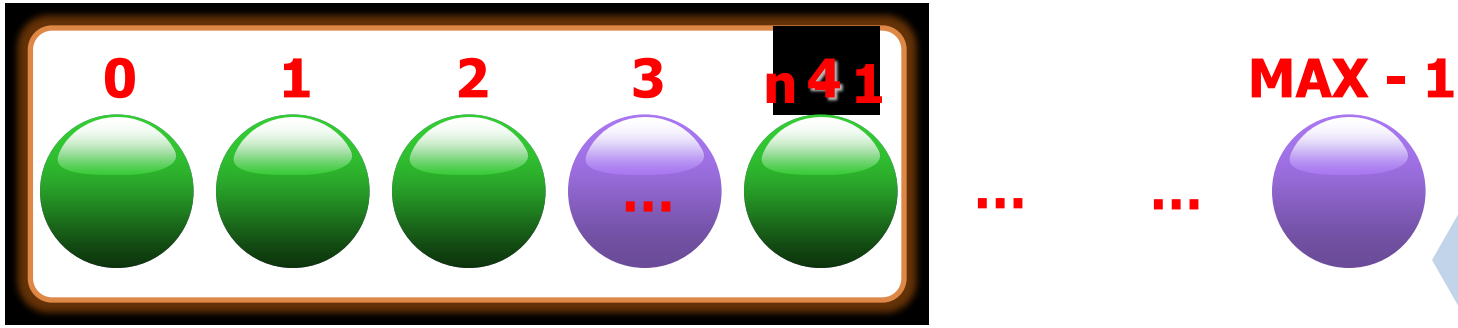
- Các hàm
  - Hàm void HoanVi(int &x, int &y): hoán vị giá trị của hai số nguyên.
  - Hàm int LaSNT(int n): kiểm tra một số có phải là số nguyên tố. Trả về 1 nếu n là số nguyên tố, ngược lại trả về 0.

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.

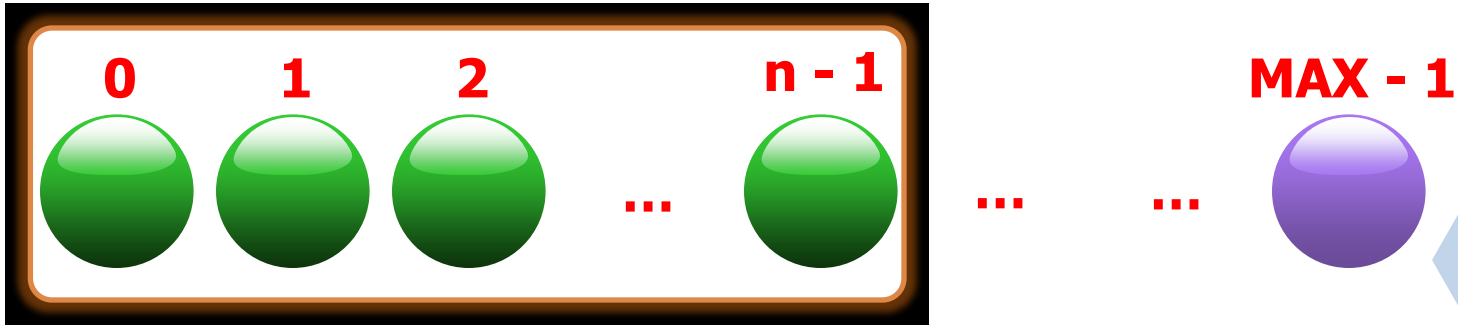


Hàm Nhập Mảng

```
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.

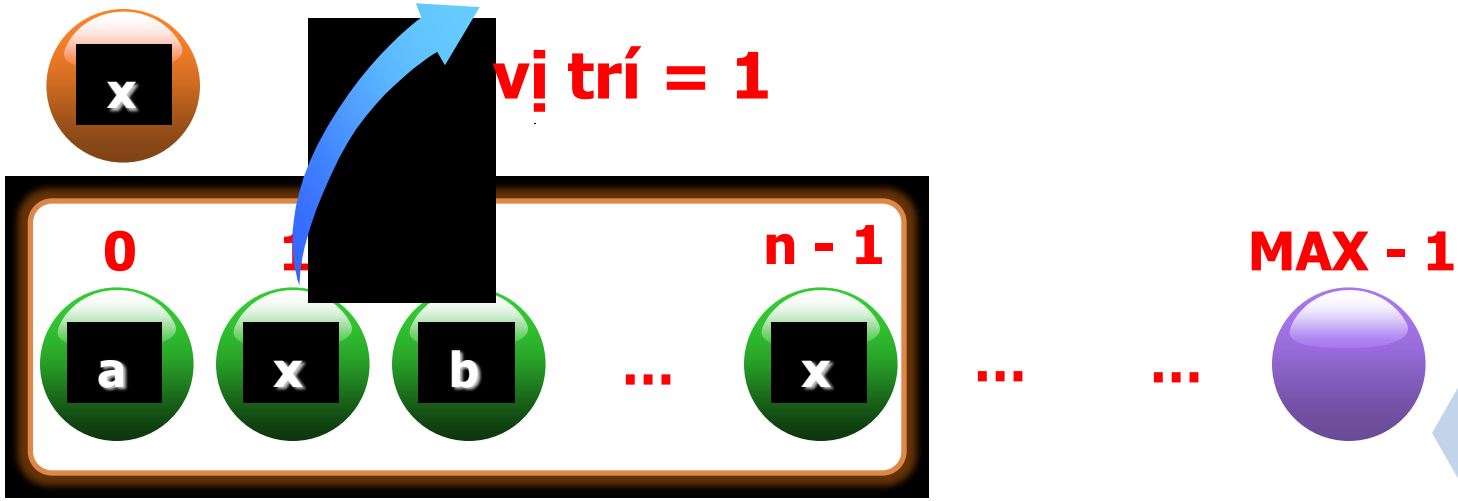


Hàm Xuất Mảng

```
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

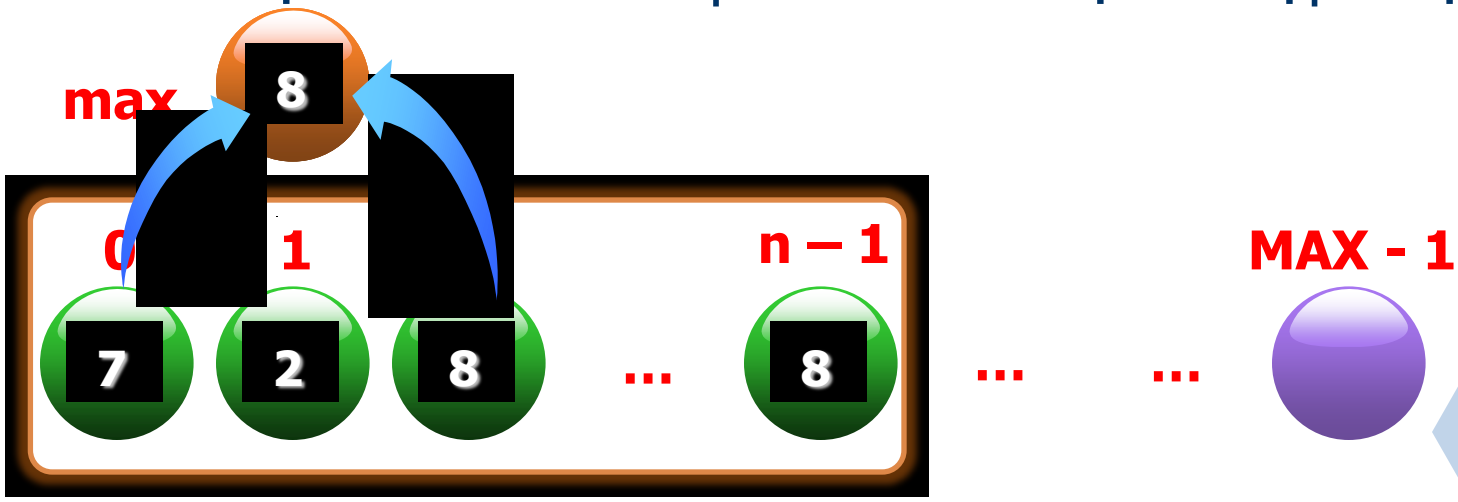
  => nc = nc + nb

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;
}
```

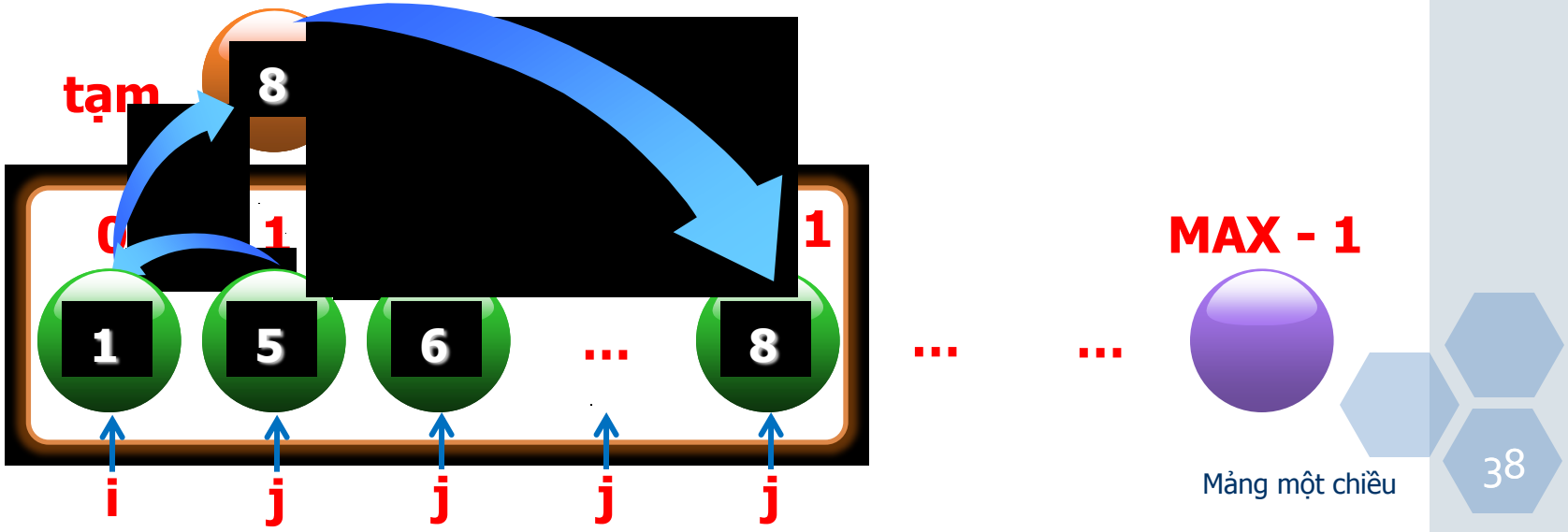
Sắp xếp mảng thành tăng dần

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).

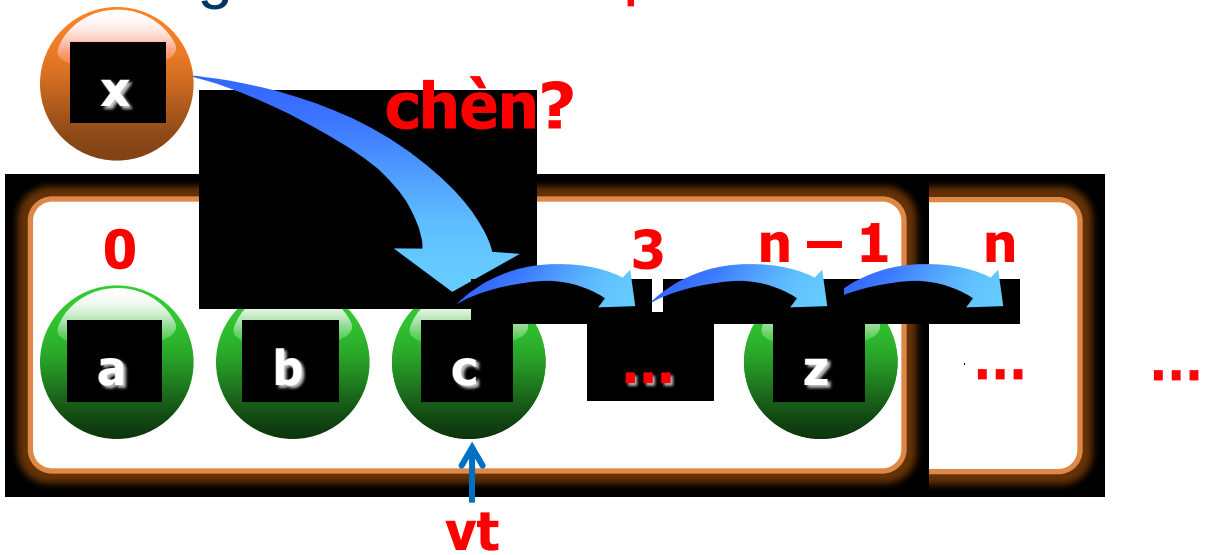


Hàm Sắp Xếp Tăng

```
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.

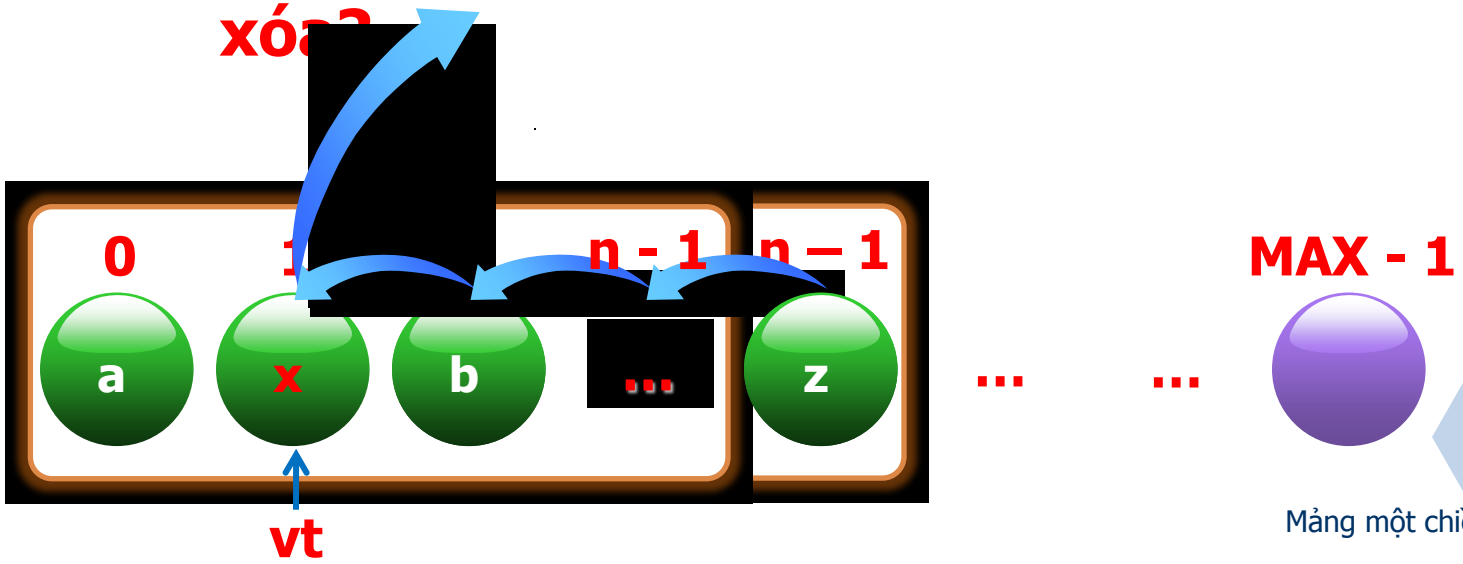


Hàm Thêm

```
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

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

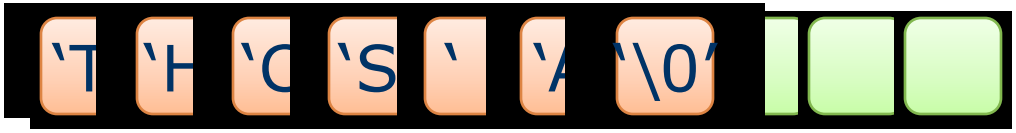
Ví dụ

```
char hoten[30];   // Dài 29 ký tự
char ngaysinh[9]; // Dài 8 ký tự
```

Khởi tạo

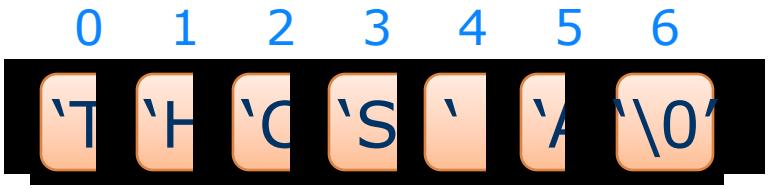
- Khởi tạo như mảng thông thường
  - Độ dài cụ thể

```
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);
```

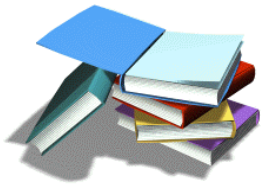
```
Nhap mot chuoi: Tin hoc co so A
Chuoi nhan duoc la: Tin hoc co so A_
```

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)



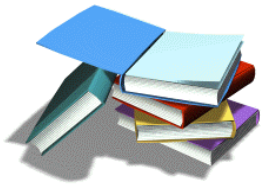
Tính độ dài chuỗi s.
size_t thay cho unsigned (trong <stddef.h>)
dùng để đo các đại lượng không dấu.

- 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)



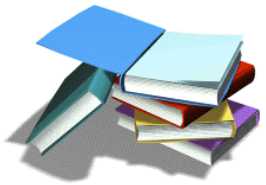
Sao chép chuỗi src sang chuỗi dest, dừng khi ký tự kết thúc chuỗi '\0' vừa được chép.
! dest phải đủ lớn để chứa src

- 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)



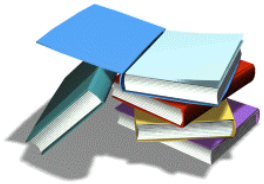
Tạo bản sao của một chuỗi s cho trước. Hàm sẽ tự tạo vùng nhớ dài strlen(s) + 1 (bytes) để chứa chuỗi s. Phải tự hủy vùng nhớ này khi không sử dụng nữa.

- 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)



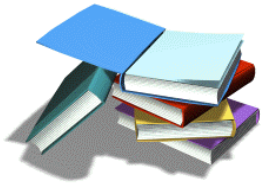
Chuyển chuỗi s thành chuỗi thường ('A' thành 'a', 'B' thành 'b', ..., 'Z' thành 'z')

- 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)



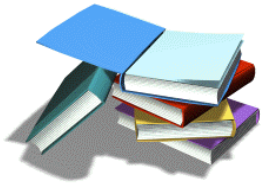
Chuyển chuỗi s thành chuỗi IN ('a' thành 'A', 'b' thành 'B', ..., 'z' thành 'Z')

- 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)



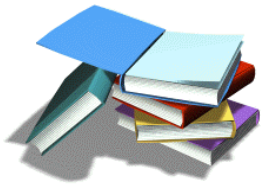
Đảo ngược thứ tự các ký tự trong chuỗi s (trừ ký tự kết thúc chuỗi).

- 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)



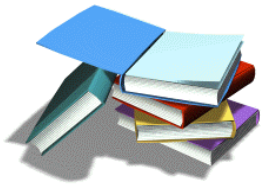
So sánh hai chuỗi s1 và s2 (phân biệt hoa thường).

- < 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)



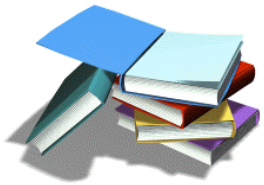
So sánh hai chuỗi s1 và s2 (không phân biệt hoa thường).

- < 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)



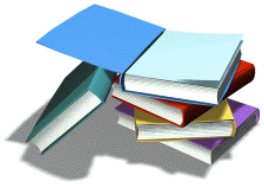
Nối chuỗi src vào sau chuỗi dest.
! Chuỗi dest phải đủ chứa kết quả

- 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

TẬP TIN

Nội dung

1. Khái niệm dòng (stream)
2. Khái niệm và phân loại tập tin
3. Các thao tác xử lý căn bản
4. Một số hàm quản lý tập tin

Nhập xuất

Khái niệm

- C lưu dữ liệu (biến, mảng, cấu trúc, …) trong bộ nhớ RAM.
- Dữ liệu được nạp vào RAM và gửi ra ngoài chương trình thông qua các thiết bị (device)
  - Thiết bị nhập (input device): bàn phím, con chuột
  - Thiết bị xuất (output device): màn hình, máy in
  - Thiết bị vừa nhập vừa xuất: tập tin
- Các thiết bị đều thực hiện mọi xử lý thông qua các dòng (stream).

Stream (dòng)

Khái niệm

- Là môi trường trung gian để giao tiếp (nhận/gửi thông tin) giữa chương trình và thiết bị.

  ➔ Muốn nhận/gửi thông tin cho một thiết bị ta sẽ gửi thông tin cho stream nối với thiết bị đó (độc lập thiết bị).
- Stream là dãy byte dữ liệu
  - "Chảy" vào chương trình gọi là stream nhập.
  - "Chảy" ra chương trình gọi là stream xuất.

Stream (dòng)

- Phân loại
  - Stream văn bản (text)
    - Chỉ chứa các ký tự.
    - Tổ chức thành từng dòng, mỗi dòng tối đa 255 ký tự, kết thúc bởi ký tự cuối dòng '\0' hoặc ký tự sang dòng mới '\n'.
  - Stream nhị phân (binary)
    - Chứa các byte.
    - Được đọc và ghi chính xác từng byte.
    - Xử lý dữ liệu bất kỳ, kể cả dữ liệu văn bản.
    - Được sử dụng chủ yếu với các tập tin trên đĩa.

Stream (dòng)

- Các stream chuẩn định nghĩa sẵn

Tên	Stream	Thiết bị tương ứng
stdin	Nhập chuẩn	Bài phím
stdout	Xuất chuẩn	Màn hình
stderr	Lỗi chuẩn	Màn hình
stdprn (MS-DOS)	In chuẩn	Máy in (LPT1:)
stdaux (MS-DOS)	Phụ chuẩn	Cổng nối tiếp COM 1:

- Ví dụ (hàm fprintf xuất ra stream xác định)
  - Xuất ra màn hình: fprintf(stdout, "Hello");
  - Xuất ra máy in: fprintf(stdprn, "Hello");
  - Xuất ra thiết bị báo lỗi: fprintf(stderr, "Hello");
  - Xuất ra tập tin (stream fp): fprintf(fp, "Hello");

Tập tin

Nhu cầu

- Dữ liệu giới hạn và được lưu trữ tạm thời
  - Nhập: gõ từ bàn phím.
  - Xuất: hiển thị trên màn hình.
  - Lưu trữ dữ liệu: trong bộ nhớ RAM.

  ➔ Mất thời gian, không giải quyết được bài toán với số dữ liệu lớn.
- Cần một thiết bị lưu trữ sao cho dữ liệu vẫn còn khi kết thúc chương trình, có thể sử dụng nhiều lần và kích thước không hạn chế.

Tập tin

Khái niệm

- Tập hợp thông tin (dữ liệu) được tổ chức theo một dạng nào đó với một tên xác định.
- Một dãy byte liên tục (ở góc độ lưu trữ).
- Được lưu trữ trong các thiết bị lưu trữ ngoài như đĩa mềm, đĩa cứng, USB…
  - Vẫn tồn tại khi chương trình kết thúc.
  - Kích thước không hạn chế (tùy vào thiết bị lưu trữ)
- Cho phép đọc dữ liệu (thiết bị nhập) và ghi dữ liệu (thiết bị xuất).

Tập tin

- Phân loại
  - Theo người sử dụng: quan tâm đến nội dung tập tin nên sẽ phân loại theo phần mở rộng

    ➔ .EXE, .COM, .CPP, .DOC, .PPT, …
  - Theo người lập trình: tự tạo các stream tường minh để kết nối với tập tin xác định nên sẽ phân loại theo cách sử dụng stream trong C

    ➔ tập tin kiểu văn bản (ứng với stream văn bản) và tập tin kiểu nhị phân (ứng với stream nhị phân).

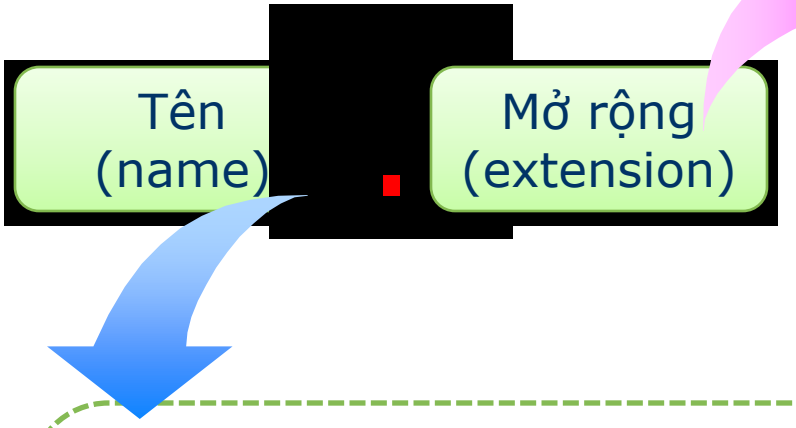
Phân loại tập tin

- Tập tin kiểu văn bản (stream văn bản)
  - Dãy các dòng kế tiếp nhau.
  - Mỗi dòng dài tối đa 255 ký tự và kết thúc bằng ký hiệu cuối dòng (end_of_line).
  - Dòng không phải là một chuỗi vì không được kết thúc bởi ký tự '\0'.
  - Khi ghi '\n' được chuyển thành cặp ký tự CR (về đầu dòng, mã ASCII 13) và LF (qua dòng, mã ASCII 10).
  - Khi đọc thì cặp CR-LF được chuyển thành '\n'.

Phân loại tập tin

- Tập tin kiểu nhị phân (stream nhị phân)
  - Dữ liệu được đọc và ghi một cách chính xác, không có sự chuyển đổi nào cả.
  - Ký tự kết thúc chuỗi '\0' và end_of_line không có ý nghĩa là cuối chuỗi và cuối dòng mà được xử lý như mọi ký tự khác.

Quy tắc đặt tên 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.

Định vị tập tin

Đường dẫn

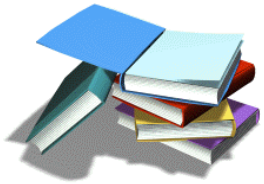
- Chỉ đến một tập tin không nằm trong thư mục hiện hành. Ví dụ: c:\data\list.txt chỉ tập tin list.txt nằm trong thư mục data của ổ đĩa C.
- Trong chương trình, đường dẫn này được ghi trong chuỗi như sau: "c:\\data\\list.txt"
- Dấu '\' biểu thị ký tự điều khiển nên để thể hiện nó ta phải thêm một dấu '\' ở trước. Nhưng nếu chương trình yêu cầu nhập đường dẫn từ bàn phím thì chỉ nhập một dấu '\'.

Quy trình thao tác với tập tin

- 1. Mở tập tin: tạo một stream nối kết với tập tin cần mở, stream được quản lý bởi biến con trỏ đến cấu trúc FILE
  - Cấu trúc được định sẵn trong STDIO.H
  - Các thành phần của cấu trúc này được dùng trong các thao tác xử lý tập tin.
- 2. Sử dụng tập tin (sau khi đã mở được tập tin)
  - Đọc dữ liệu từ tập tin đưa vào chương trình.
  - Ghi dữ liệu từ chương trình lên tập tin.
- 3. Đóng tập tin (sau khi sử dụng xong).

Hàm mở tập tin

FILE *fopen(const char *filename, const char *mode)



Mở tập tin có tên (đường dẫn) là chứa trong filename với kiểu mở mode (xem bảng).

- Thành công: con trỏ kiểu cấu trúc FILE
- Thất bại: NULL (sai quy tắc đặt tên tập tin, không tìm thấy ổ đĩa, không tìm thấy thư mục, mở tập tin chưa có để đọc, …)

```
FILE* fp = fopen("taptin.txt", "rt");
if (fp == NULL)
        printf("Khong mo duoc tap tin!");
```

Đối số mở tập tin (mode)

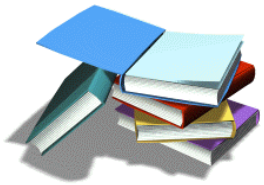
Đối số	Ý nghĩa
b	Mở tập tin kiểu nhị phân (binary)
t	Mở tập tin kiểu văn bản (text) (mặc định)
r	Mở tập tin chỉ để đọc dữ liệu từ tập tin. Trả về NULL nếu không tìm thấy tập tin.
w	Mở tập tin chỉ để ghi dữ liệu vào tập tin. Tập tin sẽ được tạo nếu chưa có, ngược lại dữ liệu trước đó sẽ bị xóa hết.
a	Mở tập tin chỉ để thêm (append) dữ liệu vào cuối tập tin. Tập tin sẽ được tạo nếu chưa có.
r+	Giống mode r và bổ sung thêm tính năng ghi dữ liệu và tập tin sẽ được tạo nếu chưa có.
w+	Giống mode w và bổ sung thêm tính năng đọc.
a+	Giống mode a và bổ sung thêm tính năng đọc.

Đọc và ghi dữ liệu (stdio.h)

- Thực hiện đọc/ghi dữ liệu theo các cách sau:
  - Nhập/xuất theo định dạng
    - Hàm: fscanf, fprintf
    - Chỉ dùng với tập tin kiểu văn bản.
  - Nhập/xuất từng ký tự hay dòng lên tập tin
    - Hàm: getc, fgetc, fgets, putc, fputs
    - Chỉ nên dùng với kiểu văn bản.
  - Đọc/ghi trực tiếp dữ liệu từ bộ nhớ lên tập tin
    - Hàm: fread, fwrite
    - Chỉ dùng với tập tin kiểu nhị phân.

Hàm xuất theo định dạng

int fprintf(FILE *fp, char *fnt, ...)



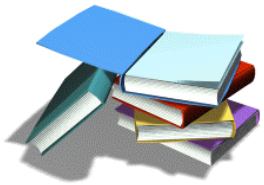
Ghi dữ liệu có chuỗi định dạng fnt (giống hàm printf) vào stream fp.
Nếu fp là stdout thì hàm giống printf.

- Thành công: trả về số byte ghi được.
- Thất bại: trả về EOF (có giá trị là -1, được định nghĩa trong STDIO.H, sử dụng trong tập tin có kiểu văn bản)

```
int i = 2912; int c = 'P'; float f = 17.06;
FILE* fp = fopen("taptin.txt", "wt");
if (fp != NULL)
        fprintf(fp, "%d %c %.2f\n", i, c, f);
```

Hàm nhập theo định dạng

int fscanf(FILE *fp, char *fnt, …)



Đọc dữ liệu có chuỗi định dạng fnt (giống hàm scanf) từ stream fp.
Nếu fp là stdin thì hàm giống printf.

- Thành công: trả về số thành phần đọc và lưu trữ được.
- Thất bại: trả về EOF.

```
int i;
FILE* fp = fopen("taptin.txt", "rt");
if (fp != NULL)
        fscanf(fp, "%d", &i);
```

Hàm nhập theo định dạng

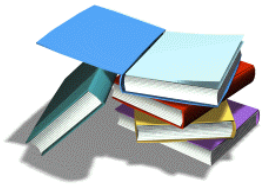
- Ví dụ
  - Một tập tin chứa nhiều dòng, mỗi dòng là thông tin mỗi sinh viên theo định dạng sau:
    - <MSSV>-<Tên>(<Phái>)tab<NTNS>tab<ĐTB>
    - Ví dụ: 0312078-H. P. Trang(Nu) 17/06/85 8.5
- Đọc chuỗi thông tin phức hợp
  - %[chuỗi]: đọc cho đến khi không gặp ký tự nào trong chuỗi thì dừng.
  - %[^chuỗi]: đọc cho đến khi gặp một trong những ký tự trong chuỗi thì dừng.

Hàm Tách 2 Mảng

```
struct SINHVIEN {
        char MSSV[8];       // 0312078
        char HoTen[30];     // H. P. Trang
        char GioiTinh[4]; // Nu
        char NTNS[9];       // 17/06/85
        float DiemTB;       // 8.5
};
void main() {
        SINHVIEN sv;
        FILE *fp = fopen("dssv.txt", "rt");
        if (fp != NULL) {
                fscanf(fp, "%[^-]-%[^(](%[^)])\t%[^\t]
                                \t%f", &sv.MSSV, &sv.HoTen,
                        &sv.GioiTinh, &sv.NTNS, &sv.DiemTB);
                fclose(fp);
        }
}
```

Hàm nhập ký tự

int getc(FILE *fp) và int fgetc(FILE *fp)



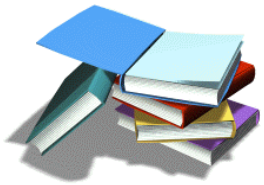
Đọc một ký tự từ stream fp.
getc là macro còn fgetc là phiên bản hàm của macro getc.

- Thành công: trả về ký tự đọc được sau khi chuyển sang số nguyên không dấu.
- Thất bại: trả về EOF khi kết thúc stream fp hoặc gặp lỗi.

```
char ch;
FILE* fp = fopen("taptin.txt", "rt");
if (fp != NULL)
        ch = getc(fp); // ⇔ ch = fgetc(fp);
```

Hàm nhập chuỗi

Int fgets(char *str, int n, FILE *fp)



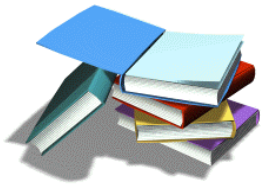
Đọc một dãy ký tự từ stream fp vào vùng nhớ str, kết thúc khi đủ n-1 ký tự hoặc gặp ký tự xuống dòng.

- Thành công: trả về str.
- Thất bại: trả về NULL khi gặp lỗi hoặc gặp ký tự EOF.

```
char s[20];
FILE* fp = fopen("taptin.txt", "rt");
if (fp != NULL)
        fgets(s, 20, fp);
```

Hàm xuất ký tự

int putc(int ch, FILE *fp) và int fputc(in ch, FILE *fp)



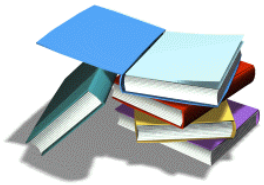
Ghi ký tự ch vào stream fp.
putc là macro còn fputc là phiên bản hàm của macro putc.

- Thành công: trả về ký tự ch.
- Thất bại: trả về EOF.

```
FILE* fp = fopen("taptin.txt", "rt");
if (fp != NULL)
        putc('a', fp); // hoặc fputc('a', fp);
```

Hàm xuất chuỗi

int fputs(const char *str, FILE *fp)



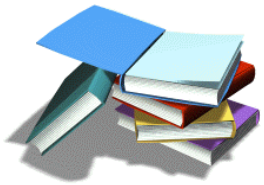
Ghi chuỗi ký tự str vào stream fp. Nếu fp là stdout thì fputs giống hàm puts, nhưng puts ghi ký tự xuống dòng.

- Thành công: trả về ký tự cuối cùng đã ghi.
- Thất bại: trả về EOF.

```
char s[] = "Ky thuat lap trinh";
FILE* fp = fopen("taptin.txt", "wt");
if (fp != NULL)
        fputs(s, fp);
```

Hàm xuất trực tiếp

int fwrite(void *buf, int size, int count, FILE *fp)



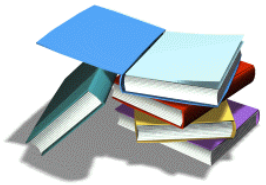
Ghi count mẫu tin có kích thước mỗi mẫu tin là size (byte) từ vùng nhớ buf vào stream fp (theo kiểu nhị phân).

- Thành công: trả về số lượng mẫu tin (không phải số lượng byte) đã ghi.
- Thất bại: số lượng nhỏ hơn count.

```
int a[] = {1, 2, 3};
FILE* fp = fopen("taptin.dat", "wb");
if (fp != NULL)
        fwrite(a, sizeof(int), 3, fp);
```

Hàm nhập trực tiếp

int fread(void *buf, int size, int count, FILE *fp)



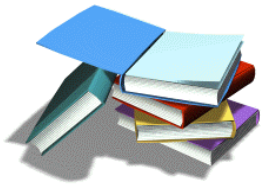
Đọc count mẫu tin có kích thước mỗi mẫu tin là size (byte) vào vùng nhớ buf từ stream fp (theo kiểu nhị phân).

- Thành công: trả về số lượng mẫu tin (không phải số lượng byte) thật sự đã đọc.
- Thất bại: số lượng nhỏ hơn count khi kết thúc stream fp hoặc gặp lỗi.

```
int a[5];
FILE* fp = fopen("taptin.dat", "wb");
if (fp != NULL)
        fread(a, sizeof(int), 3, fp);
```

Hàm đóng tập tin xác định

int fclose(FILE *fp)



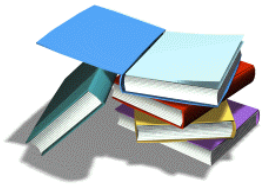
Đóng stream fp.
Dữ liệu trong stream fp sẽ được "vét" (ghi hết lên đĩa) trước khi đóng.

- Thành công: trả về 0.
- Thất bại: trả về EOF.

```
FILE* fp = fopen("taptin.txt", "rt");
...
fclose(fp);
```

Hàm đóng tất cả stream

int fcloseall()



Đóng tất cả stream đang được mở ngoại trừ các stream chuẩn stdin, stdout, stdprn, stderr, stdaux.
Nên đóng từng stream thay vì đóng tất cả.

- Thành công: trả về số lượng stream được đóng.
- Thất bại: trả về EOF.

```
FILE* fp1 = fopen("taptin1.txt", "rt");
FILE* fp2 = fopen("taptin2.txt", "wt");
...
fcloseall();
```

"Vét" dữ liệu trong stream

- Khi chương trình kết thúc, các stream đang mở sẽ được "vét" (flush) và đóng lại. Tuy nhiên, ta nên đóng một các tường minh các stream sau khi sử dụng xong (nhất là các stream tập tin) để tránh các sự cố xảy ra trước khi chương trình kết thúc bình thường.
- Ta có thể "vét" dữ liệu trong stream mà không cần đóng stream đó bằng một trong hai hàm:
  - Vét stream fp xác định: `int fflush(FILE *fp);`
  - Vét tất cả stream đang mở: `int flushall();`

Con trỏ chỉ vị (position indicator)

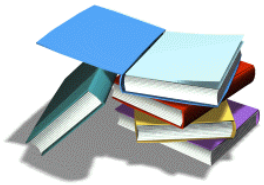
- Khái niệm
  - Được tạo tự động khi mở tập tin.
  - Xác định nơi diễn ra việc đọc/ghi trong tập tin
- Vị trí con trỏ chỉ vị
  - Khi tập tin chưa mở: ở đầu tập tin (giá trị 0).
  - Khi mở tập tin:
    - Ở cuối tập tin khi mở để chèn (mode a hay a+)
    - Ở đầu tập tin (hay giá trị 0) khi mở với các mode khác (w, w+, r, r+).

Truy xuất tuần tự & ngẫu nhiên

- Truy xuất tuần tự (sequentially access)
  - Phải đọc/ghi dữ liệu từ vị trí con trỏ chỉ vị đến vị trí n-1 trước khi đọc dữ liệu tại vị trí n.
  - Không cần quan tâm đến con trỏ chỉ vị do con trỏ chỉ vị tự động chuyển sang vị trí kế tiếp sau thao tác đọc/ghi dữ liệu.
- Truy xuất ngẫu nhiên (random access)
  - Có thể đọc/ghi tại vị trí bất kỳ trong tập tin mà không cần phải đọc/ghi toàn bộ dữ liệu trước đó ➔ quan tâm đến con trỏ chỉ vị.

Hàm đặt lại vị trí con trỏ chỉ vị

void rewind(FILE *fp)



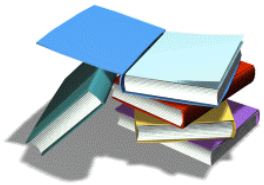
Đặt lại vị trí con trỏ chỉ vị về đầu (byte 0) tập tin fp.

- Không

```
FILE* fp = fopen("taptin.txt", "w+");
fprintf(fp, "0123456789");
rewind(fp);
fprintf(fp, "*****");
```

Hàm tái định vị con trỏ chỉ vị

int fseek(FILE *fp, long offset, ing origin)



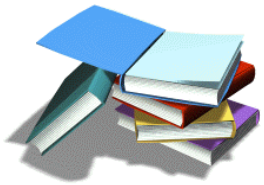
Đặt vị trí con trỏ chỉ vị trong stream fp với vị trí offset so với cột mốc origin (SEEK_SET hay 0: đầu tập tin; SEEK_CUR hay 1: vị trí hiện tại; SEEK_END hay 2: cuối tập tin)

- Thành công: trả về 0.
- Thất bại: trả về giá trị khác 0.

```
FILE* fp = fopen("taptin.txt", "w+");
fseek(fp, 0L, SEEK_SET);  // ⇔ rewind(fp);
fseek(fp, 0L, SEEK_END); // cuối tập tin
fseek(fp, -2L, SEEK_CUR);// lùi lại 2 vị trí
```

Hàm xác định vị trí con trỏ chỉ vị

long ftell(FILE *fp)



Hàm trả về vị trí hiện tại của con trỏ chị vị (tính từ vị trí đầu tiên của tập tin, tức là 0) của stream fp.

- Thành công: trả về vị trí hiện tại của con trỏ chỉ vị.
- Thất bại: trả về -1L.

```
FILE* fp = fopen("taptin.txt", "rb");
fseek(fp, 0L, SEEK_END);
long size = ftell(fp);
printf("Kich thuoc tap tin la %ld\n", size);
```

Dấu hiệu kết thúc tập tin

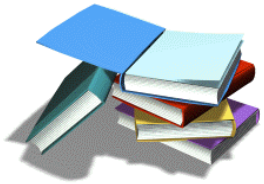
- Khi đã biết kích thước tập tin
  - Sử dụng fwrite để lưu n mẫu tin
    ➔ kích thước = n * sizeof(1 mẫu tin);
  - Sử dụng hàm fseek kết hợp hàm ftell
- Khi chưa biết kích thước tập tin
  - Hằng số EOF (=-1) (chỉ cho tập tin văn bản)
    ➔ while ((c = fgetc(fp)) != EOF) …
  - Hàm int feof(FILE *fp) (cho cả 2 kiểu tập tin)
    ➔ trả về số 0 nếu chưa đến cuối tập tin
    ➔ trả về số khác 0 nếu đã đến cuối tập tin.

Các hàm quản lý tập tin

- Hàm nhập xuất tập tin (File I/O function) là các đã đề cập phần trước
  - Mở và đóng tập tin: fopen, fclose
  - Nhập/Xuất tập tin:
    - Theo định dạng: fprintf, fscanf
    - Từng ký tự hay chuỗi: fputc, fputs, fgetc, fgets
    - Trực tiếp từ bộ nhớ: fwrite, fread
- Hàm quản lý tập tin (File-Management function)
  - Xóa tập tin: remove
  - Đổi tên tập tin: rename

Hàm xóa tập tin

int remove(const char *filename)



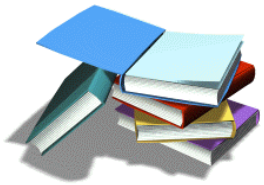
Xóa tập tin xác định bởi filename.

- Thành công: trả về 0.
- Thất bại: trả về -1.

```
if (remove("c:\\vc.txt") == 0)
        printf("Tap tin vc.txt da bi xoa!");
else
        printf("Ko xoa duoc tap tin vc.txt!");
```

Hàm đổi tên tập tin

int rename(const char *oldname, const char *newname)



Đổi tên tập tin oldname thành newname.
Hai tập tin phải cùng ổ đĩa nhưng không cần thiết phải cùng thư mục (có thể sử dụng để di chuyển hay sao chép tập tin).

- Thành công: trả về 0.
- Thất bại: trả về -1.

```
if (rename("c:\\a.txt", "c:\\BT\b.cpp") == 0)
        printf("Doi ten tap tin thanh cong");
else
        printf("Doi ten tap tin that bai");
```

Bài tập lý thuyết

- Bài 1: Sự khác nhau giữa stream kiểu văn bản và stream kiểu nhị phân?

➔

- Bài 2: Cần phải làm gì trước khi muốn truy xuất tập tin?

➔

Bài tập lý thuyết

- Bài 3: Khi mở tập tin bằng fopen, ta cần phải xác định thông tin nào và hàm sẽ trả về cái gì?
  - ➔
- Bài 4: Ba phương pháp để truy xuất tập tin?
  - ➔ 1.
  - ➔ 2.
  - ➔ 3.

Bài tập lý thuyết

- Bài 5: Hai phương pháp để đọc thông tin từ tập tin là gì?
  - ➔ 1.
  - ➔ 2.
- Bài 6: Giá trị của EOF?
  - ➔
- Bài 7: Ta dùng hằng ký hiệu EOF để làm gì?
  - ➔

Bài tập lý thuyết

- Bài 8: Cách xác định cuối tập tin trong kiểu văn bản và kiểu nhị phân?

➔

- Bài 9: Con trỏ chỉ vị là gì và cách thay đổi nó?

➔

➔

Bài tập lý thuyết

- Bài 10: Nếu mở một tập tin chưa có (bằng mode w), cho biết giá trị của con trỏ chị vị lúc đầu?

  ➔

- Bài 11: Viết lệnh đóng tất cả các stream tập tin.

  ➔

- Bài 12: Trình bày hai cách khác nhau để chuyển con trỏ chỉ vị về đầu tập tin fp.

  ➔ 1.

  ➔ 2.

Bài tập lý thuyết

- Bài 13: Đoạn chương trình sau có sai không?

```
void main()
{
	FILE *fp;
	int c;
	if ((fp = fopen("abc.xyz", "rb")) == NULL)
		printf("Khong mo duoc tap abc.xyz\n");
	else
	{
		while ((c = fgetc(fp)) != EOF)
			fprintf(stdout, "%c", c);
		fclose(fp);
	}
}
```

Bài tập thực hành

- Bài 14: Viết chương trình ghi 3 số nguyên a, b, c được nhập từ bàn phím vào một tập tin.
- Bài 15: Viết chương trình đọc 3 số nguyên a, b, c từ một tập tin, sau đó giải phương trình $ax^2 + bx + c = 0$ rồi ghi kết quả vào một tập tin khác.
- Bài 16: Viết chương trình đọc n số nguyên từ một tập tin cho trước, sau đó sắp xếp tăng dần rồi ghi kết quả vào 1 tập tin khác. Ví dụ:

4 → 4

2 5 1 4 → 1 2 4 5

Bài tập thực hành

- Bài 17: Viết chương trình ghi các dòng văn bản được nhập từ bàn phím lên tập tin.
- Bài 18: Viết chương trình in nội dung một tập tin lên màn hình.
- Bài 19: Viết chương trình đếm số ký tự chữ cái của tập tin và xuất kết quả ra một tập tin khác.
- Bài 20: Viết chương trình đếm số từ của tập tin và xuất kết quả ra một tập tin khác.
- Bài 21: Viết chương trình đếm số lần lặp lại của một từ trong một tập tin.

Bài tập thực hành

- Bài 22: Viết chương trình mở tập tin văn bản đã có trên đĩa, sao chép nó thành một tập tin văn bản mới với điều kiện là các chữ thường đổi thành chữ hoa, tất cả các ký tự khác không đổi.
- Bài 23: Viết chương trình ghép 2 tập tin văn bản, nội dung tập tin thứ hai được ghép sau tập tin thứ nhất.
- Bài 24: Viết sao sao chép một tập tin cho trước.
- Bài 25: Viết chương trình ghi một danh sách cấu trúc xuống tập tin sau đó đọc lên kiểm tra lại.

NMLT
MỞ ĐẦU

Trần Phước Tuấn
tranphuoctuan.khoatoan.dhsp@gmail.com
http://baigiang.tranphuoctuan.com

Đề cương bài giảng

- **Mục tiêu môn học:** Cung cấp cho sinh viên các kỹ năng cơ bản để lập trình giải quyết các vấn đề, bài toán. Các chương trình được thể hiện bằng NNLT C. Riêng về ngôn ngữ lập trình C, các sinh viên được cung cấp các kỹ năng:
  - *Đọc và viết được chương trình đơn giản*
  - *Hiểu cấu trúc ngôn ngữ*
  - *Sử dụng thành thạo các thư viện chuẩn*
  - *Nhận biết và sửa chữa các lỗi thường gặp khi lập trình*
- **Các môn học tiên quyết:** không.
- **Nội dung bài giảng**:

Nội dung môn học

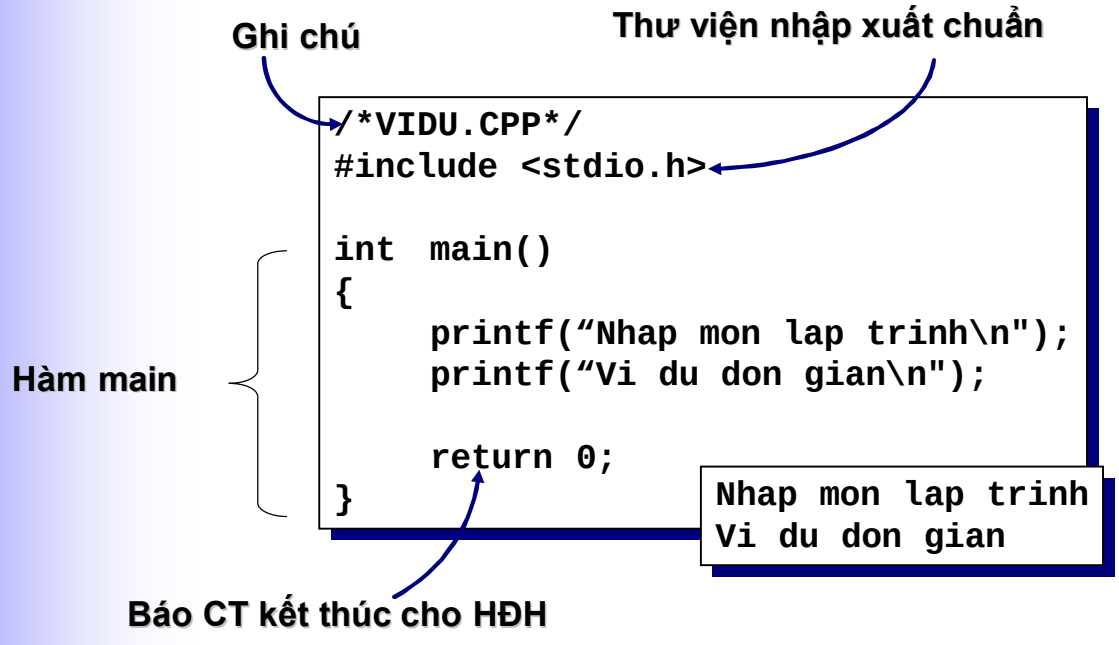
- Tổng quan
- Các kiểu dữ liệu cơ bản
- Lệnh nhập, xuất dữ liệu
- Các cấu trúc điều khiển
- Hàm
- Struct
- Mảng
- Con trỏ
- Chuỗi ký tự

Tổng quan

- Khái niệm chương trình – lập trình
- Cấu trúc một chương trình đơn giản
- Khái niệm Thuật toán – biểu diễn thuật toán
- Khái niệm NNLT, sơ lược lịch sử phát triển NNLT
- Ngôn ngữ lập trình C

Các thành phần của chương trình C

Ví dụ chương trình C



Một số lưu ý từ ví dụ

- Phần ghi chú được trình biên dịch bỏ qua
- Phân biệt chữ in hoa và chữ in thường
- Câu lệnh luôn được kết thúc bằng dấu ;
- Chuỗi ký tự phải ghi giữa cặp nháy kép “
- In xuống dòng dùng ký tự \n
- Chương trình nên thông báo kết quả thực hiện với hệ thống: Tốt – 0, có lỗi – 1, 2, 3 …
- Chương trình có một hàm main

Ví dụ 2

Khai báo 2 biến số nguyên, “a” và “b”

Nhập 2 số nguyên vào a và b

Viết các biểu thức “a”, “b” và “a-b” theo định dạng %i

```
#include <stdio.h>

int  main(void)
{
     int   a, b;

     printf(“Nhap 2 so ngguyen: ");
     scanf("%i %i", &a, &b);

     printf("%i - %i = %i\n", a, b, a - b);

     return 0;
}
```

```
Nhap 2 so nguyen: 21 17
21 - 17 = 4
```

Biến – Variable

```
int a, b;
```

- Chứa dữ liệu có thể thay đổi được trong chương trình.
- Muốn sử dụng phải khai báo.
- Tên: gồm chữ cái, ký số, dấu nối (_), không được bắt đầu bằng ký số.
- Biến khai báo trong khối được gọi là biến cục bộ, không thuộc khối nào được gọi là biến toàn cục
- Có tác dụng trong toàn khối kể từ lúc được khai báo.

Lệnh xuất - printf

Xuất dữ liệu ra màn hình:

```
printf("%i - %i = %i\n", a, b, a - b);
```

- Các ký tự hằng được in nguyên văn
- Các ký tự định dạng được thay bằng giá trị của biểu thức tương ứng:
- %i: ký tự định dạng số nguyên kiểu int
- Các ký tự điều khiển: \n – xuống dòng; \t – dấu tab; \\ – dấu \; \"– dấu " …
- Thư viện: **stdio.h**

Lệnh nhập - scanf

Nhập dữ liệu từ bàn phím

```
scanf("%i %i", &a, &b);
```

- Trong chuỗi định dạng chỉ có ký tự định dạng và khoảng trắng.
- Dữ liệu phải được nhập vào các biến.
- Trước tên biến phải ghi dấu & - toán tử địa chỉ. Nếu không có toán tử địa chỉ, giá trị của biến sẽ không được cập nhật
- Thư viện: **stdio.h**

Kiểu dữ liệu cơ sở trong C

Các kiểu số nguyên của C

- C hỗ trợ khá nhiều kiểu số nguyên
- Các giá trị lớn nhất và nhỏ nhất được định nghĩa trong thư viện "`limits.h`"

Kiểu	định dạng	kích thước	nhỏ nhất	lớn nhất
`char`	%c	1	CHAR_MIN	CHAR_MAX
`unsigned char`	%c	1	0	UCHAR_MAX
`short` [int]	%hi	2	SHRT_MIN	SHRT_MAX
`unsigned short`	%hu	2	0	USHRT_MAX
`int`	%i	**2 or 4**	INT_MIN	INT_MAX
`unsigned int`	%u	**2 or 4**	0	UINT_MAX
`long` [int]	%li	4	LONG_MIN	LONG_MAX
`unsigned long`	%lu	4	0	ULONG_MAX

Ví dụ về số nguyên

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

int  main()
{
     unsigned long  big = ULONG_MAX;

     printf("minimum int = %i, ", INT_MIN);
     printf("maximum int = %i\n", INT_MAX);
     printf("maximum unsigned = %u\n", UINT_MAX);
     printf("maximum long int = %li\n", LONG_MAX);
     printf("maximum unsigned long = %lu\n", big);

     return 0;
}
```

```
minimum int = -32768, maximum int = 32767
maximum unsigned = 65535
maximum long int = 2147483647
maximum unsigned long = 4294967295
```

Ví dụ kiểu ký tự

In ra mã ASCII của ký tự

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

int main()
{
    char  lower_a = 'a';
    char  lower_m = 'm';

    printf("minimum char = %i, ", CHAR_MIN);
    printf("maximum char = %i\n", CHAR_MAX);

    printf("Sau '%c' la '%c'\n", lower_a, lower_a + 1);
    printf("Ky tu in hoa '%c'\n", lower_m - 'a' + 'A');

    return 0;
}
```

Trong NNLT C, ký tự chính là số nguyên

```
minimum char = -128, maximum char = 127
Sau 'a' la 'b'
Ky tu in hoa 'M'
```


Số nguyên trong các cơ số khác

- Các hệ cơ số có thể thực hiện được: cơ số 8 (octal), cơ số 10 (decimal), cơ số 16 (hexadecimal)

Số 0: số octal

0x: số hexadecimal

```
#include <stdio.h>

int main(void)
{
    int   dec = 20, oct = 020, hex = 0x20;

    printf("dec=%d, oct=%d, hex=%d\n", dec, oct, hex);
    printf("dec=%d, oct=%o, hex=%x\n", dec, oct, hex);

    return 0;
}
```

```
dec=20, oct=16, hex=32
dec=20, oct=20, hex=20
```

Số thực

- C hỗ trợ nhiều kiểu số thực lưu trữ dấu chấm động.
- Các giá trị lớn nhất và nhỏ nhất được định nghĩa trong thư viện "`float.h`"

Kiểu	định dạng	kích thước	nhỏ nhất	lớn nhất
`float`	`%f %e %g`	4	`FLT_MIN`	`FLT_MAX`
`double`	`%lf %le %lg`	8	`DBL_MIN`	`DBL_MAX`
`long double`	`%Lf %Le %Lg`	10	`LDBL_MIN`	`LDBL_MAX`

Ví dụ số thực:

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

int  main(void)
{
     double f = 3.1416, g = 1.2e-5, h = 5000000000.0;

     printf("f=%lf\tg=%lf\th=%lf\n", f, g, h);
     printf("f=%le\tg=%le\th=%le\n", f, g, h);
     printf("f=%lg\tg=%lg\th=%lg\n", f, g, h);

     printf("f=%7.2lf\tg=%.2le\th=%.4lg\n", f, g, h);

     return 0;
}
```

```
f=3.141600          g=0.000012          h=5000000000.000000
f=3.141600e+00      g=1.200000e-05      h=5.000000e+09
f=3.1416            g=1.2e-05           h=5e+09
f=   3.14           g=1.20e-05          h=5e+09
```

Hằng – Constant

```
const int days_in_week = 7;
```

Chứa dữ liệu không thể thay đổi được trong chương trình.

- Muốn sử dụng phải khai báo.
- Phải có kiểu (tương tự như biến)
- Hằng số có chứa "." hoặc "e" có kiểu `double` (3.5, 1e-7, -1.29e15)
- Hằng số kiểu `float` kết thúc bởi "F" (3.5F, 1e-7F)
- Hằng số kiểu `long double` kết thúc bởi "L" (-1.29e15L, 1e-7L)
- Hằng số không có ".", "e" hoặc "F" có kiểu `int`. Ví dụ: 10000, -35. (Một vài trình biên dịch tự động chuyển thành `long int` nếu giá trị hằng tràn kiểu `int`)
- Khai báo hằng `long int` phải thêm vào cuối "L" (9000000L)

Ví dụ về hằng

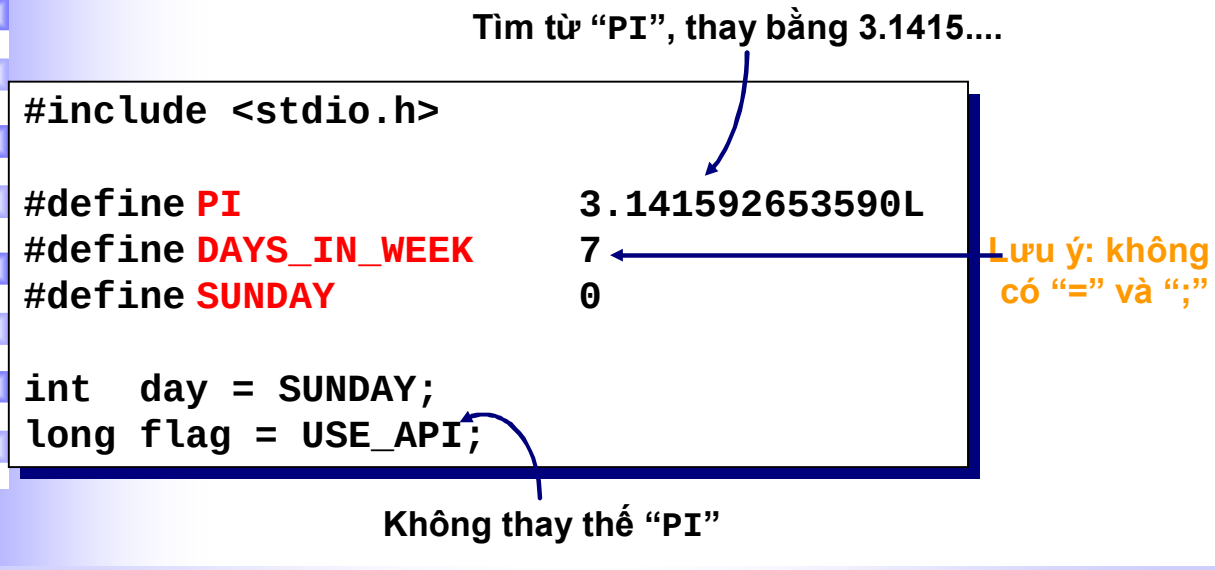
Các hằng pi, days_in_week, sunday được tạo với từ khóa const

```
#include <stdio.h>

int main(void)
{
    const long double pi = 3.141592653590L;
    const int days_in_week = 7;
    const sunday = 0;

    days_in_week = 5;

    return 0;
}
```

Lỗi

Hằng xử lý trước biên dịch

- Các hằng có thể được xác lập trước khi biên dịch
  - Bản chất là tìm kiếm và thay thế
  - Thường được đặt tên với các chữ cái in hoa



Toán tử trong C

Toán tử trong C

- Phép toán số học
- Ép kiểu
- Các toán tử trên bit
- Các toán tử so sánh
- Phép gán
- Toán tử **`sizeof`**
- Biểu thức điều kiện

Toán tử số học

- NNLT C hỗ trợ các phép toán số học:

+	cộng
-	trừ
*	nhân
/	chia
%	chia lấy dư

Lưu ý:

- *"/" cho kết quả phụ thuộc vào kiểu của các toán hạng*
- *"%" không thực hiện được với các số thực*

Ví dụ về toán tử chia "/"

- Trình biên dịch dựa vào kiểu của các toán hạng để quyết định phép chia tương ứng

"i", "j" kiểu int, "/" là phép chia lấy nguyên → k nhận giá trị 1

"f", "g" kiểu double, "/" là phép chia số thực → h nhận giá trị 1.25

Phép chia nguyên, bất kể "h" có kiểu double. Kết quả là 1.00000

```
int  main(void)
{
    int     i = 5,   j = 4,   k;
    double  f = 5.0, g = 4.0, h;

    k = i / j;
    h = f / g;
    h = i / j;

    return 0;
}
```


Ép kiểu

- Ép kiểu làm thay đổi *tạm thời* kiểu của một biến trong một biểu thức.

Phép chia số nguyên được thực hiện, kết quả, 1, được đổi sang kiểu double, 1.00000

```
int  main(void)
{
    int i = 5, j = 4;
    double f;

    f = (double)i / j;
    f = i / (double)j;
    f = (double)i / (double)j;
    f = (double)(i / j);

    return 0;
}
```

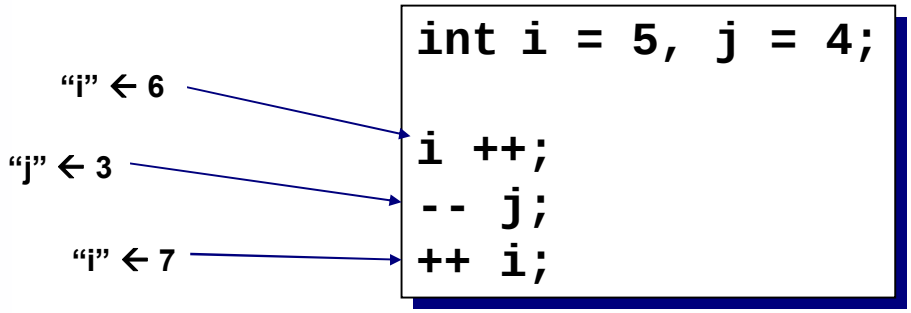
Phép tăng (giảm) 1

- NNLT C có 2 toán tử đặc biệt hỗ trợ việc tăng (giảm) giá trị của một biến thay đổi 1 đơn vị

  ++ tăng 1

  -- giảm 1

- Các toán tử này có thể đặt ở trước hoặc sau biến.



Trước hay sau ?

Thứ tự thực hiện các toán tử ++ và -- phụ thuộc vào vị trí của chúng (trước hay sau) so với biến:

```
#include <stdio.h>

int main(void)
{
    int   i, j = 5;
    i = ++j;
    printf("i=%d, j=%d\n", i, j);
    j = 5;
    i = j++;
    printf("i=%d, j=%d\n", i, j);

    return 0;
}
```

Tương đương:
1. j++;
2. i = j;

Tương đương:
1. i = j;
2. j++;

```
i=6, j=6
i=5, j=6
```

Kiểu luận lý trong C

- Trong C không có kiểu dữ liệu luận lý (thể hiện các giá trị ĐÚNG – SAI), thay vào đó các biểu thức so sánh sẽ cho kết quả là **SỐ**
- Giá trị 0 (0.0) ứng với kết quả SAI (FALSE)
- Các giá trị khác như 1, -3.5, -7, 10.4, … (khác không) đều được xem là ĐÚNG (TRUE)

Các toán tử so sánh

- NNLT C hỗ trợ các phép so sánh:

<	bé hơn
<=	bé hơn hay bằng
>	lớn hơn
>=	lớn hơn hay bằng
==	bằng
!=	không bằng

- Tất cả đều cho kết quả 1 khi so sánh đúng và 0 trong trường hợp ngược lại.

Toán tử luận lý

- NNLT C hỗ trợ các toán tử luận lý:

&&	và (and)
||	hoặc (or)
!	phủ định (not)

- Tất cả đều cho kết quả 1 hoặc 0 tương ứng các trường hợp ĐÚNG hoặc SAI

```
int i, j = 10, k = 28;

i = ((j > 5) && (k < 100)) || (k > 24);
```


Toán tử luận lý

- Lưu ý khi sử dụng các toán tử luận lý:

Nếu không có các dấu (), các phép toán được thực hiện từ trái sang phải

```
if((i < 10) && (a[i] > 0))
    printf("%i\n", a[i]);
```

Nên viết
"i < 10" được kiểm tra trước, nếu không đúng giá trị của biểu thức sẽ là 0 và "a[i] > 0" sẽ không được tính

Không nên: `(a < b < c)`

Nên: `((a < b) && (b< c)`

Toán tử trên bit

- Các toán tử trên bit chỉ có tác dụng trên các kiểu số nguyên:

&	And
|	Or
^	XOr
>>	Đẩy phải
<<	Đẩy trái

Ví dụ về các toán tử trên bit

```
#include <stdio.h>

int main(void)
{
    short a = 0x6eb9;
    short b = 0x5d27;
    unsigned short c = 7097;

    printf("0x%x, ", a & b);
    printf("0x%x, ", a | b);
    printf("0x%x\n", a ^ b);

    printf("%u, ", c << 2);
    printf("%u\n", c >> 1);

    return 0;
}
```

```
0x4c21, 0x7fbf, 0x339e
28388, 3548
```

```
0x6eb9   0110 1110 1011 1001
0x5d27   0101 1101 0010 0111
0x4c21   0100 1100 0010 0001
```

```
0x6eb9   0110 1110 1011 1001
0x5d27   0101 1101 0010 0111
0x7fbf   0111 1111 1011 1111
```

```
0x6eb9   0110 1110 1011 1001
0x5d27   0101 1101 0010 0111
0x339e   0011 0011 1001 1110
```

```
 7097   0001 1011 1011 1001
28388   0110 1110 1110 0100
```

```
 7097   0001 1011 1011 1001
 3548   0000 1101 1101 1100
```

Phép gán

- Có thể sử dụng liên tiếp nhiều phép gán
- Giá trị được gán sẽ sẵn sàng cho lệnh kế tiếp

```
int  i, j, k, l, m, n;

i = j = k = l = m = n = 22;

printf("%i\n", j = 93);
```

"n = 22" gán trước, lại gán "n" cho "m", "m" cho "l", … → i, j, k, l, m, n đều nhận giá trị 22.

"j" được gán 93, giá trị 93 sẽ được in ra màn hình

Phép gán

Lưu ý:

- Vế trái phép gán luôn phải là biến
- Không được nhầm lẫn giữa so sánh bằng "==" và gán "="

```
#include <stdio.h>

int main(void)
{
    int  i = 0;

    if(i = 0)
        printf("i nhan gia tri khong\n");
    else
        printf("i khac khong\n");

    return 0;
}
```

```
i khac khong
```

Một số phép gán đặc biệt

- Các phép gán kết hợp toán tử khác:

  `+=` `-=` `*=` `/=` `%=`

  `&=` `|=` `^=`

  `<<=` `>>=`

- Tổng quát:

  biến ***op*=** **biểu thức**

  tương đương:

  biến = biến *op* (biểu thức)

```
a += 27;
```
```
a = a + 27;
```

```
f /= 9.2;
```
```
f = f / 9.2;
```

```
i *= j + 2;
```
```
i = i * (j + 2);
```


Toán tử `sizeof`

`sizeof(Obj)`

- Cho biết kích thước của đối tượng theo đơn vị byte

```
#include <stdio.h>

int main(void)
{
    long  big;

    printf("\"big\" is %u bytes\n", sizeof(big));
    printf("a short is %u bytes\n", sizeof(short));
    printf("a double is %u bytes\n", sizeof (double));

    return 0;
}
```

```
"big" is 4 bytes
a short is 2 bytes
a double is 8 bytes
```

Biểu thức chọn theo điều kiện

(điều kiện) ? BT1 : BT2

- Biểu thức nhận giá trị BT1 nếu điều kiện khác 0 (ĐÚNG), các trường hợp khác nhận giá trị BT2

```
int i, j = 100, k = -1;

i = (j > k) ? j : k;
```

```
Nếu (j > k)
    i = j;
Ngược lại
    i = k;
```

```
int i, j = 100, k = -1;

i = (j < k) ? j : k;
```

```
Nếu (j < k)
    i = j;
Ngược lại
    i = k;
```

- Có thể định nghĩa sẵn một macro để tìm số lớn:

```
#define max(x, y) ((x>y) ? x : y)
```


Độ ưu tiên của toán tử

- Thứ tự thực hiện các toán tử trong một biểu thức phụ thuộc vào *độ ưu tiên* của chúng.
- Có 15 mức ưu tiên.
- Thông thường, toán tử một ngôi có độ ưu tiên cao hơn toán tử hai ngôi.
- Các cặp dấu ngoặc đơn () thường được dùng để chỉ rõ thứ tự các toán tử.

```
#include <stdio.h>
int main(void)
{
    int  j = 3 * 4 + 48 / 7;
    printf("j = %i\n", j);

    return 0;
}
```

```
j = 18
```

Bảng thứ tự thực hiện các toán tử

Toán tử	Thứ tự (nếu cùng ĐƯT)
() [] -> .	→
! ++ -- - + (cast) * & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= >= >	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %= …	←

Luyện tập

```
#include <stdio.h>

int main(void)
{
    int  i = 0, j, k = 7, m = 5, n;
    j = m += 2;
    printf("j = %d\n", j);
    j = k++ > 7;
    printf("j = %d\n", j);
    j = i == 0 & k;
    printf("j = %d\n", j);
    n = !i > k >> 2;
    printf("n = %d\n", n);
    return 0;
}
```

Tóm lược

- Một số thành phần của chương trình trong C
- Hàm main
- Lệnh xuất / nhập – printf / scanf
- Biến
- Các kiểu số nguyên và số thực
- Hằng – 2 cách khai báo
- Sử dụng toán tử trong C