

# Kỹ thuật lập trình



# Bài 1

## CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

# Nội dung

- ▶ Các khái niệm cơ bản.
- ▶ Các bước xây dựng chương trình
- ▶ Biểu diễn thuật toán
- ▶ Cài đặt thuật toán bằng ngôn ngữ lập trình

# Các khái niệm cơ bản

- ▶ **Chương trình**
- ▶ Program: a set of instructions.
- ▶ Chương trình: Tập các lệnh máy mà CPU phải thực thi nhằm giải một bài toán.
- ▶ CPU chạy 1 chương trình theo cách tuần tự từng lệnh.

1001001000101...

1100110011000...

0101001100110...

1010110100001

.....

# Các khái niệm cơ bản

## ▶ Lập trình máy tính

- Gọi tắt là **lập trình** (programming).
- Nghệ thuật **cài đặt** một hoặc nhiều **thuật toán** trừu tượng có liên quan với nhau bằng một **ngôn ngữ lập trình** để tạo ra một **chương trình máy tính**.



pótay

1001001000101...  
1100110011000...  
0101001100110...  
1010110100001  
.....

# Thuật toán (Giải thuật) Algorithm

## ▶ Thuật toán

- Là **tập hợp** (dãy) **hữu hạn** các **chỉ thị** (hành động) được **định nghĩa rõ ràng** nhằm **giải quyết** một bài toán cụ thể nào đó.
- Cách giải một bài toán
- Cách giải 1 bài toán cụ thể là 1 giải thuật cụ thể.
- Mô tả 1 giải thuật là diễn đạt các bước thực thi của giải thuật đó.
  - Dùng ngôn ngữ tự nhiên.
  - Dùng lưu đồ (flowchart)
  - Mã giả

# Thuật toán

## ▶ Ví dụ

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

Đầu vào: a, b thuộc  $\mathbb{R}$

Đầu ra: nghiệm phương trình  $ax + b = 0$

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

# Tính chất của thuật toán

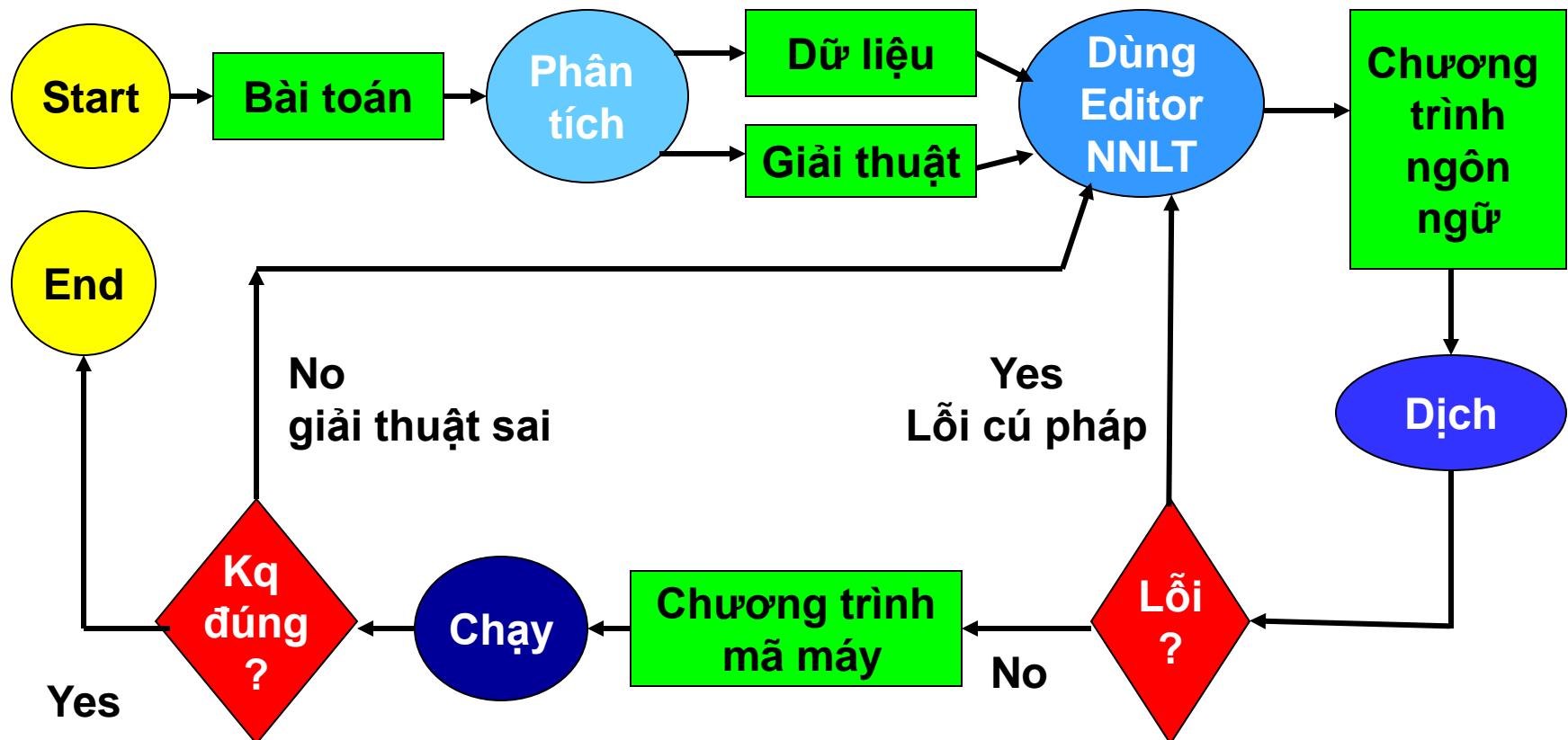
## ► Bao gồm 5 tính chất sau:

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

# Các bước xây dựng chương trình



# Các bước xây dựng chương trình



# Biểu diễn bằng ngôn ngữ tự nhiên

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

Đầu vào: a, b thuộc R

Đầu ra: nghiệm phương trình  $ax + b = 0$

1. Bắt đầu
2. Nhập 2 số thực a và b.
3. Nếu  $a = 0$  thì
  - 3.1. Nếu  $b = 0$  thì
    - 3.1.1. Phương trình vô số nghiệm
    - 3.1.2. Kết thúc thuật toán.
  - 3.2. Ngược lại
    - 3.2.1. Phương trình vô nghiệm.
    - 3.2.2. Kết thúc thuật toán.
4. Ngược lại
  - 4.1. Phương trình có nghiệm.
  - 4.2. Giá trị của nghiệm đó là  $x = -b/a$
  - 4.3. Kết thúc thuật toán.

# Biểu diễn bằng ngôn ngữ tự nhiên

Nhập vào 2 số a, b. Cho biết tổng của chúng

Đầu vào: a, b thuộc R

Đầu ra: tổng của a và b

1. Nhập 2 số a và b.
2. Tính tổng:  $c = a + b$
3. Xuất tổng: c
4. Kết thúc thuật toán

# Biểu diễn bằng lưu đồ

- ▶ Flowchart: Dùng hình vẽ để mô tả một giải thuật.
- ▶ Trong flowchart chỉ rõ tiến trình thực thi giải thuật.
- ▶ Không thể hiểu lầm vì có quy tắc để vẽ.
- ▶ Đây là cách diễn đạt hình thức cho giải thuật.

# Quy tắc vẽ lưu đồ



**Khối giới hạn**

Chỉ thị bắt đầu và kết thúc.

**Khối vào ra**

Nhập dữ liệu.

**Khối lựa chọn**

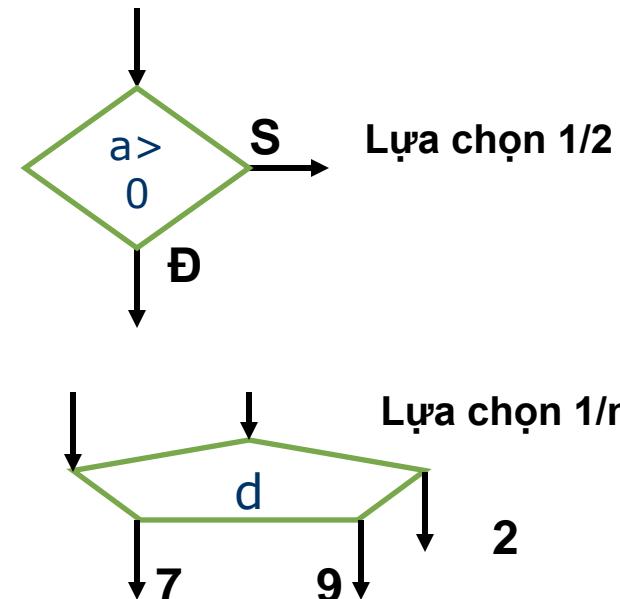
Tùy điều kiện sẽ rẽ nhánh.

**Khối thao tác**

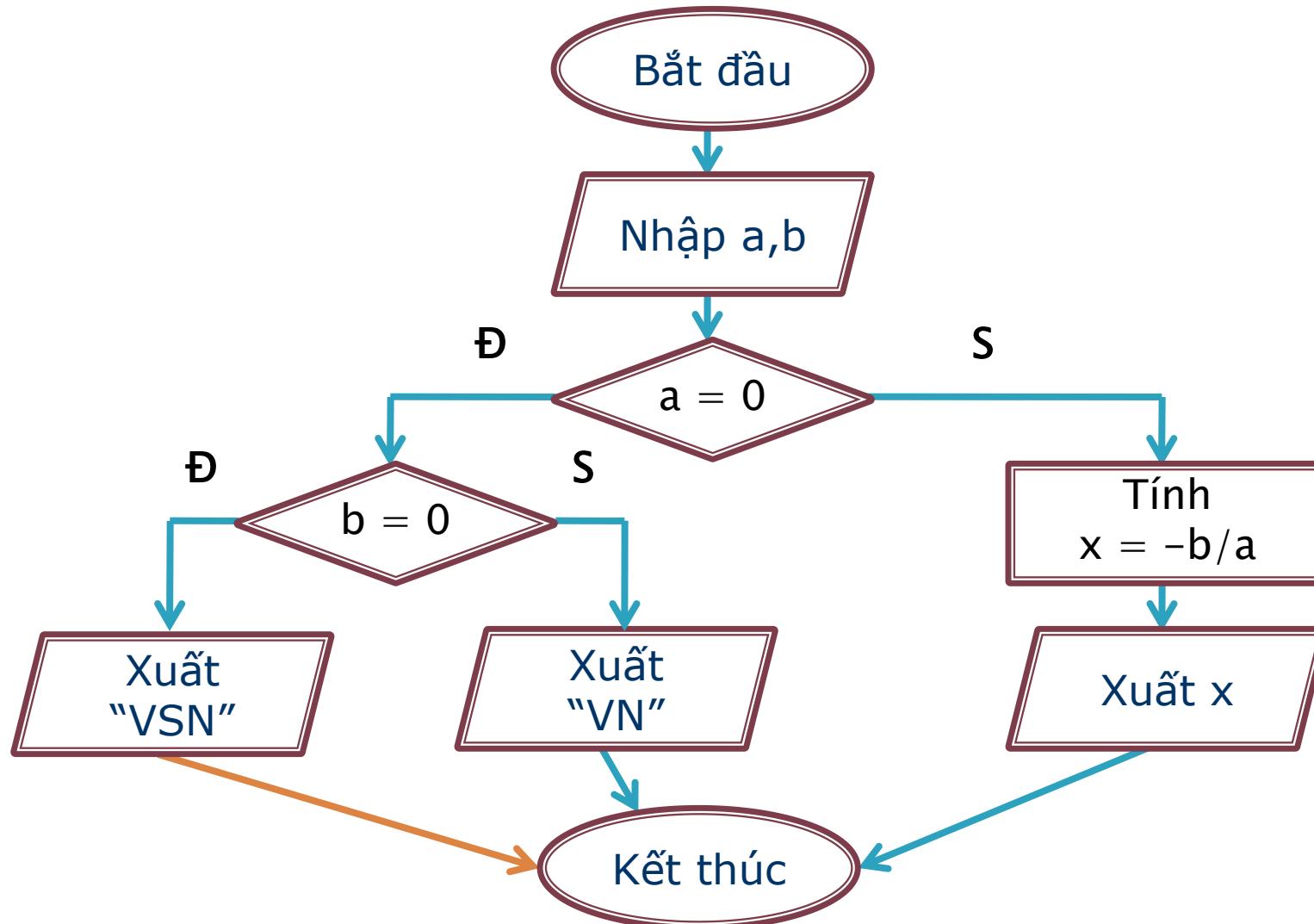
Ghi thao tác cần thực hiện.

**Đường đi**

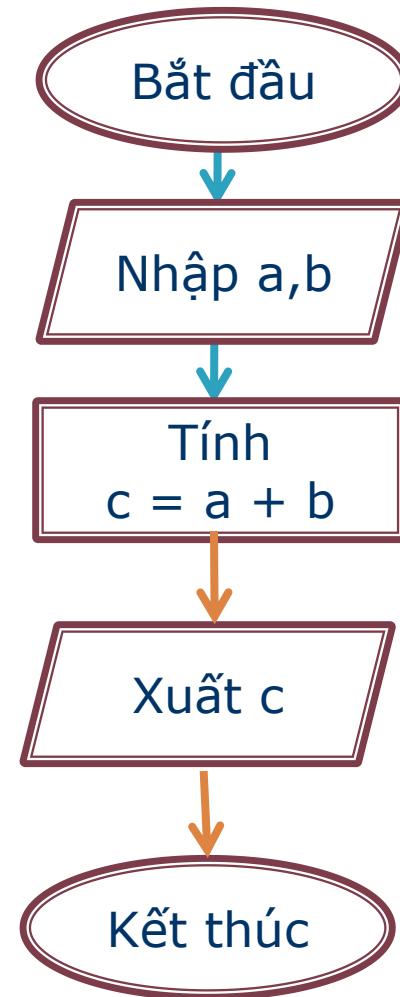
Chỉ hướng thao tác tiếp theo.



# Giải thuật giải pt $ax + b = 0$



# Giải thuật tính tổng a và b



# Biểu diễn bằng mã giả

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

Đầu vào: a, b thuộc R

Đầu ra: nghiệm phương trình  $ax + b = 0$

Nhập a, b

If a = 0 Then

Begin

    If b = 0 Then

        Xuất “Phương trình vô số nghiệm”

    Else

        Xuất “Phương trình vô nghiệm”

End

Else

    Xuất “Phương trình có nghiệm  $x = -b/a$ ”

# Cài đặt thuật toán bằng C/C++

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

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

# Bài tập

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

# Bài tập

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

# Bài tập

7. Nhập điểm thi 3 môn Toán, Lý, Hóa của một học sinh và nhập điểm chuẩn. Tính điểm trung bình theo hệ số: Toán hệ số 2, Lý và Hóa hệ số 1. Xuất điểm trung bình và kết quả (“Đậu” nếu điểm trung bình lớn hơn hoặc bằng điểm chuẩn, ngược lại “Không đậu”)
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. Cho biết số xe của bạn được mấy nút?
10. Nhập vào 2 số nguyên.  
Tính min và max của hai số đó.

# Bài 2

# GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH C

# Nội dung

- ▶ Giới thiệu
- ▶ Bộ từ vựng của C
- ▶ Cấu trúc chương trình C
- ▶ Ví dụ minh họa

# Giới thiệu

## ▶ Giới thiệu C

- Dennis Ritchie tại Bell Telephone năm 1972.
- Tiền thân của ngôn ngữ **B**, Ken Thompson, 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++ 5.2, Dev C.
- Visual C++, 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**
- Bắt đầu bằng chữ hoặc ký tự gạch dưới, theo sau là chữ cái, chữ số hoặc ký tự gạch dưới (\_).
- 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**.
- Thông thường :Đặt chữ hoa cho các hằng, chữ thường cho các đại lượng còn lại (biến,hàm..).
- Nên đặt 1 cách gợi nhớ ( 8 ký tự đầu là có nghĩa và tuỳ thuộc chương trình ).

# 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, bAltaP, ...
  - Các tên sau, ghạch dưới tên nào hợp lệ

1xyz	_abc	Delta_1	A#B
Del ta	3MN	f(x)	do
X-1	BETA	a_1	x1

# 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, thư viện

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

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

```
// circle.c: Calculate and print the areas of circles

#include <stdio.h> // Preprocessor directive

double circularArea( double r ); // Function declaration (prototype form)

int main( ) // Definition of main( ) begins
{
    double radius = 1.0, area = 0.0;

    printf( "    Areas of Circles\n\n" );
    printf( "    Radius           Area\n"
            "    -----'\n" );

    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );

    radius = 5.0;
    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );

    return 0;
}

// The function circularArea( ) calculates the area of a circle
// Parameter:    The radius of the circle
// Return value: The area of the circle

double circularArea( double r ) // Definition of circularArea( ) begins
{
    const double pi = 3.1415926536; // Pi is a constant
    return pi * r * r;
}
```

## Cấu trúc một chương trình C

```
// circle.c: Calculate and print the areas of circles
#include <stdio.h> // Preprocessor directive
double circularArea( double r ); // Function declaration (prototype form)

int main( ) // Definition of main( ) begins
{
    double radius = 1.0, area = 0.0;

    printf( "    Areas of Circles\n\n" );
    printf( "    Radius          Area\n"
            "    ----- \n" );

    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );

    radius = 5.0;
    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );

    return 0;
}

// The function circularArea( ) calculates the area of a circle
// Parameter:    The radius of the circle
// Return value: The area of the circle

double circularArea( double r ) // Definition of circularArea( ) begins
{
    const double pi = 3.1415926536; // Pi is a constant
    return pi * r * r;
}
```

# Thành phần trong một chương trình C

→ Bộ ký tự

→ Các từ khóa

→ Lời chú thích

```
// circle.c: Calculate and print the areas of circles
#include <stdio.h> // Preprocessor directive
double circularArea( double r ); // Function declaration (prototype form)
int main() // Definition of main( ) begins
{
    double radius = 1.0, area = 0.0;
    printf( "    Areas of Circles\n\n" );
    printf( "    Radius          Area\n"
            "-----\n" );
    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );
    radius = 5.0;
    area = circularArea( radius );
    printf( "%10.1f    %10.2f\n", radius, area );
    return 0;
}

// The function circularArea( ) calculates the area of a circle
// Parameter:    The radius of the circle
// Return value: The area of the circle
double circularArea( double r ) // Definition of circularArea( ) begins
{
    const double pi = 3.1415926536; // Pi is a constant
    return pi * r * r;
}
```

# Ví dụ

Chương trình C đơn giản:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    printf("Chào bạn!");
    printf("Mời bạn làm quen với ngôn ngữ lập trình C!");
    getch();
}
```

# Ví dụ

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

void main()
{
    int x, y, tong;
    printf("Nhập hai số nguyên: ");
    scanf("%d%d", &x, &y);
    tong = x + y;
    printf("Tổng hai số là %d", tong);
    getch();
}
```

# Bài tập

1. Tên (định danh) nào sau đây đặt không hợp lệ, tại sao?
  - Ky thuat lap trinh, 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 C. Giải thích ý nghĩa của từng phần trong cấu trúc.
4. Viết chương trình C xuất ra thông tin của cá nhân gồm:  
Ho ten: Nguyen Van A  
Ngay sinh: 1/1/1998  
Lop: C10TH  
Mon hoc: Ky thuat lap trinh

# Bài tập

5. Viết chương trình C xuất ra màn hình như sau:

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

# Bài 3

## CÁC KIỂU DỮ LIỆU CƠ SỞ

# Nội dung

- ▶ Các kiểu dữ liệu cơ sở
- ▶ Biến, Hằng, Câu lệnh & Biểu thức
- ▶ Các lệnh nhập xuất
- ▶ 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:  $-2^{n-1} \dots +2^{n-1} - 1$

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:  $0 \dots 2^n - 1$

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

- $17.06 = 1.706 \cdot 10 = 1.706 \cdot 10^1$

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

- (\*) 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= 1;  
unsigned char dem;  
float ketqua = 1.1, delta;
```

**Định nghĩa:** Là đại lượng thay đổi, mỗi biến có tên và  
địa chỉ vùng nhớ riêng

**Cú pháp:**

<kiểu> <tên biến>;

<kiểu> <tên biến 1>, <tên biến 2>;

# Biến

## Khởi đầu cho biến :

Nếu trong khai báo ngay sau tên biến ta đặt dấu = và một giá trị nào đó thì đây chính là cách vừa khai báo vừa khởi đầu cho biến.

## Ví dụ :

```
int a,b=20,c,d=40;
```

```
float e=-55.2,x=27.23,y,z,t=18.98;
```

Việc khởi đầu và việc khai báo biến rồi gán giá trị cho nó sau này là hoàn toàn tương đương.

# Biến

## In ra giá trị của biến :

Sử dụng từ khóa “printf” để in ra các giá trị cho biến. Tuy nhiên, tùy theo kiểu dữ liệu của biến mà xác định đúng để in ra.

### Ví dụ :

```
int a=20; // a là số nguyên, dùng %d để xác nhận in ra
printf("Giá trị biến a là: %d",a);
```

int: %d

float: %f

long: %lf

double: %ld

chuỗi: %s, %c

# Hằng số

## Hằng thường

Cú pháp  
<kiểu> <tênhằng> = <giá trị>;

Ví dụ

```
int a = 1506; // 150610
```

```
int b = 01506; // 15068
```

```
int c = 0x1506; // 150616 (0x hay 0X)
```

```
float d = 15.06e-3; // 15.06*10-3 (e hay E)
```

# Hằng số

## Hằng ký hiệu

**Định nghĩa:** Là величина không  
thay đổi trong quá trình tính toán

**Cú pháp**

#define <tên hằng> <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;
```

# Các loại hằng

Hằng nguyên ( int ) : -32768 đến 32767

**Ví dụ :** #define number1 -50

Định nghĩa hằng int number1 có giá trị là -50

#define sodem 2732

Định nghĩa hằng int sodem có giá trị là 2732

Hằng thực (float và double):

214.35 , - 234.34 hoặc 1.543e7 (15430000)

Hoặc 123.456E-4 (0.123456)

▶ **Chú ý :**

Cần phân biệt hai hằng 5056 và 5056.0 : ở đây 5056 là số nguyên còn 5056.0 là hằng thực.

# Chương trình C sử dụng biến, hằng

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float PI=3.141593;
    printf("số PI là: %lf ",PI);
    printf("số 2*PI là: %lf ", 2*PI);
    getch();
}

#include <stdio.h>
#include <conio.h>
#define PI 3.141593
void main()
{
    printf("số PI là: %lf ",PI);
    printf("số 2*PI là: %lf ", 2*PI);
    getch();
}
```

# 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

- $2 + 3$ ,  $a / 5$ ,  $(a + b) * 5$ , ...
- $y/3 + \sin(x) * \cos(b+c)$
- $n1 > n2 ? n1 : n2$

# Toán tử

- Toán tử (operator): Ký hiệu mô tả một phép toán, cho một kết quả duy nhất
- Toán hạng (operand): Dữ liệu để phép toán tác động
- Kết quả (result): Dữ liệu mới được sinh ra sau khi thực hiện xong phép toán

## CÁC LOẠI TOÁN TỬ:

- **Toán tử số học – Arithmetic ops**
- **Toán tử so sánh – Relational ops**
- **Toán tử luận lý – Logical ops**
- **Toán tử gán – Assignment ops**
- Toán tử chọn thành phần – structure access ops
- Toán tử trên bit – Bitwise ops
- Các toán tử thêm – miscellaneous ops

# Toán tử số học

Toán tử	Ký hiệu	Cú pháp	Kết quả
Cộng (addition)	+	$x+y$	Cộng x với y
Trừ ( subtraction)	-	$x-y$	Trừ x cho y
Nhân ( multiplication)	*	$x*y$	Nhân x với y
Chia ( division)	/	$x/y$	Chia x cho y (thực, nguyên)
Lấy dư (remainder)	%	$x \% y$	Lấy dư nguyên của phép chia x/y
Tăng trước ( preIncrement)	++	$++x$	Tăng x 1 đơn vị trước khi dùng
Tăng sau ( postIncrement)	++	$x++$	Tăng x 1 đơn vị sau khi dùng
Giảm trước ( preDecrement)	--	$--x$	Giảm x 1 đơn vị trước khi dùng
Giảm sau ( postDecrement)	--	$x--$	Giảm x 1 đơn vị sau khi dùng
Đổi dấu (minus)	-	$-x$	Đổi dấu của x
Giữ trị ( plus)	+	$+x$	Giữ trị x không đổi

# Toán tử số học

Ví dụ:

$$11/3 = ?$$

$$11\%3 = ?$$

$$-(2+6) = ?$$

$$n=5$$

$$x = n++ \text{ Cho } x = ? \text{ và } n = ?$$

$$x = ++n \text{ Cho } x = ? \text{ và } n = ?$$

Lưu ý:  $n++$  tương ứng với  $n = n+1$

$n--$  tương ứng với  $n = n - 1$

# Thí dụ:

Toán tử % chỉ  
được thực hiện  
trên số nguyên

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int m=6; double n=4.5;
    int a=m++;
    printf("a=%d, m=%d", a, m);
    double b=++n;
    printf("\nb=%lf, n=%lf", b, n);
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    double x=12.4%3;
    printf("%lf",x);
    getch();
}
```

Lỗi

a=6, m=7  
b=5.500000, n=5.500000

# Toán tử so sánh và luận lý

Cho kết quả đúng ( 1 ) , sai ( 0 ) .

Kiểm tra có lớn hơn ? (greater than)	$x > y$
Kiểm tra có lớn hơn hay bằng ? (greater than or equal )	$x \geq y$
Kiểm tra có nhỏ hơn ? (less than )	$x < y$
Kiểm tra có nhỏ hơn hay bằng ? (greater than or equal )	$x \leq y$
Kiểm tra có bằng ? (equal- lưu ý : 2 dấu bằng )	$x == y$
Kiểm tra có khác nhau ? ( not equal to)	$x != y$
Not (đảo) 1 điều kiện (logical not)	$!x$
And (và) 2 điều kiện ( logical and)	$x \& \& y$
Or (hay) 2 điều kiện ( logical or)	$x    y$

# Toán tử so sánh và luận lý

Ví dụ:

$a = 0, b = 3$

$!a \rightarrow ?$

$a > b \rightarrow ?$

$a \&& b \rightarrow ?$

$a \&& !b \rightarrow ?$

**Chú ý:** Các phép quan hệ (so sánh) có số ưu tiên nhỏ hơn so với !  
nhưng lớn hơn so với  $\&&$  và  $\|$ , vì vậy biểu thức như :  $(a < b) \&& (c > d)$   
có thể viết lại thành :  $a < b \&& c > d$

# Thí dụ

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int m=6; double n=4;
    printf("gia tri cua m<n la %d", m<n);
    printf("\ngia tri cua m&&n la %d", m&&n);
    getch();
}
```

```
gia tri cua m<n la 0
gia tri cua m&&n la 1
```

```
0
0
```

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

void main()
{
    clrscr();
    int m=6, n=4,k=9 ;
    printf("%d", m<n||k<n);
    printf("\n%d", m&&n>k);
    getch();
}
```

# Toán tử gán

- + Toán tử gán :
- Cú pháp : < biến> = < biểu thức>

gán bằng	$x=y$	gán y cho x , sao chép trị của y sang cho x
gán hợp (compound assignment)	$x O=y$	$x= x O y$ O là các toán tử $+, -, *, /, \%, <<, >>, \&, ^,  $

\* Ví dụ :  $c = a + b$  ;  $d = t + 3$  ;

$i = i + 2$  (Viết gọn  $i += 2$  ; )

$i = i * 2$  ( $i *= 2$  ; )

$x = 2, y = 3;$

$x = y \rightarrow x = ?$  và  $y = ?$

Chú ý : Các phép toán trong C có độ ưu tiên khác nhau và quy tắc kết hợp khác nhau

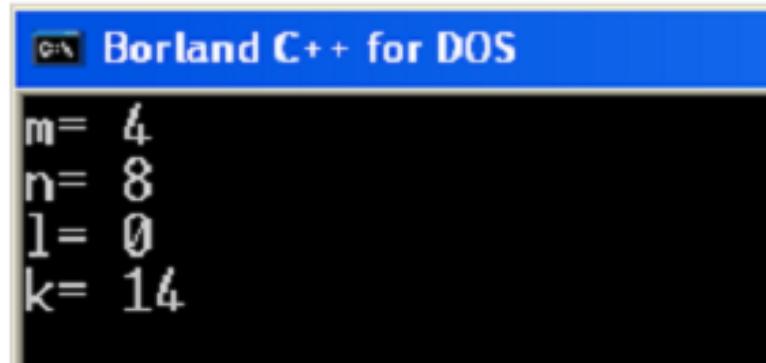
Gán giá trị mới cho biến, giá trị cũ bị đè.

# Thí dụ:

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

void main()
{
    clrscr();
    int m, n, l=5, k=2;
    m=3, n=9;

    l=k + m++;
    k+=-n + m;
    printf("m= %d",m);
    printf("\nn= %d",n);
    printf("\nl= %d",l);
    printf("\nk= %d",k);
    getch();
}
```



# Toán tử gán

- + Toán tử gán có phép toán dấu phẩy:
  - Cú pháp : < biến> = < biểu thức1, biểu thức2>
- Ví dụ :
- $$m = (t=2, t * t + 3) \rightarrow t = ? \quad \text{và } m = ?$$

# Phép toán điều kiện

+ Phép toán điều kiện – Biểu thức điều kiện :

- Cú pháp : <điều kiện> ? <biểu thức1> : <biểu thức2>

Trong biểu thức <điều kiện>, nếu đúng (1) sẽ tính và lấy kết quả của <biểu thức1>, ngược lại tính lấy kết quả của <biểu thức 2>

• Ví dụ :

$i=6$

$m = (i >= 5) ? 1 : -1 \rightarrow m = ?$

# Thứ tự ưu tiên các phép toán

Thứ tự ưu tiên các phép toán :

- **Cặp ngoặc từ trong ra ngoài**
- Toán tử số học, nhân chia trước, cộng trừ sau
- Toán tử so sánh
- Toán tử luận lý
- Toán tử gán

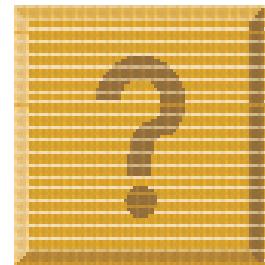
# Thứ tự ưu tiên các phép toán

## Thứ tự ưu tiên các phép toán :

TT	Phép toán	Trình tự kết hợp
1	() [] ->	Trái qua phải
2	! ~ & * - ++ -- (type ) sizeof	Phải qua trái
3	* ( phép nhân ) / %	Trái qua phải
4	+ -	Trái qua phải
5	<< >>	Trái qua phải
6	< <= > >=	Trái qua phải
7	== !=	Trái qua phải
8	&	Trái qua phải
9	^	Trái qua phải
10		Trái qua phải
11	&&	Trái qua phải
12		Trái qua phải
13	?:	Phải qua trái
14	= += -= *= /= %= <<= >>= &= ^=  =	Phải qua trái
15	,	Trái qua phải

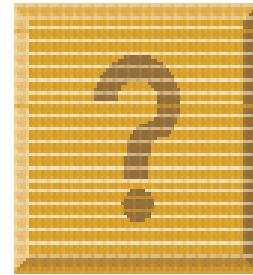
# Thí dụ:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    clrscr();
    printf("%d", 2*6 + 24/(3+7*2>5)-9&&4-7+10/3);
    getch();
}
```



# Thí dụ:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    clrscr();
    printf("%d", 2*6 + 24/(3+7*2>5)-9&&4-7+10/3);
    getch();
}
```



# Ví dụ

```
# include < stdio.h>
# Include < conio.h>
void main (void)
{
    int n1,n2, n3, nmax ;
    do
    {
        /* đọc 3 số từ bàn phím*/
        printf(" nhập số thứ nhất : "); scanf( " %d", &n1);
        printf (" nhập số thứ hai : "); scanf( " %d", &n2);
        printf(" nhập số thứ ba : "); scanf( " %d", &n3);
        /* tìm số lớn nhất */
        nmax = n1>n2 ? n1:n2;
        nmax = nmax > n3 ? nmax : n3;
        /* In ra kết quả */
        printf ( " số lớn nhất trong 3 số %d%d%d là : %d \ n",
", n1,n2,n3 ,nmax);
    }
    printf ( " ấn ESC để kết thúc );
    while ( getch ()! = 27 );
```

# Đổi kiểu giá trị

## Chuyển đổi kiểu giá trị

## ( type ) biểu thức

## ► Ví dụ :

(float) (a+b)

# Chuyển đổi kiểu thông qua phép gán :

Giá trị của vé phải được chuyển sang kiểu vé trái  
đó là kiểu của kết quả

## ▶ Ví dụ :

```
int n;
```

# Ví dụ:

Đổi giá trị thực r sang nguyên:

$$(\text{int})(r+0.5)$$

Chú ý thứ tự ưu tiên :

$$(\text{int})1.4*10=1*10=10$$

$$(\text{int})(1.4*10)=(\text{int})14.0=14$$

# Bài tập

1) Xác định các thứ tự ưu tiên để tính giá trị biểu thức

$$56*22/7 + 13/(3+9\%5*3) + (21-7*2) + 65 < 100$$

$$31-30\%(5+2-3*4+8) + (12+33 < 8 + 23\%4) + 5$$

2) Xác định giá trị cho các biến:

int m, a, b, c;

$m = (a > b \&\& b \leq c) ? (a/b + a \% c - (a < c)) : (c \% a + b/c + (++a))$

Xác định giá trị biến m, nếu các giá trị a, b, c ban đầu được gán lần lượt là:

a	b	c	m
55	3	5	?
45	12	2	?
15	22	13	?

# 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

{
    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	
<b>%c</b>	Ký tự	<b>char</b>
<b>%d, %ld</b>	Số nguyên có dấu	<b>char, int, short, long</b>
<b>%f, %lf</b>	Số thực	<b>float, double</b>
<b>%s</b>	Chuỗi ký tự	<b>char[], char*</b>
<b>%u</b>	Số nguyên không dấu	<b>unsigned int/short/long</b>

# Chuỗi định dạng

**Mã định dạng để in ra giá trị các biến:**

Mã định dạng	Ý nghĩa
%3d	Số nguyên có kích thước tối đa 3 ký số
%6f	Số thực có kích thước tối đa 6 ký số
%.2f	Số thực có 2 số lẻ (phần nguyên không quy định)
%f	Số thực kích thước căn cứ vào giá trị thực tế
%0	Số nguyên hệ 8
%x	Số nguyên hệ 16
%c	Ký tự
%s	Chuỗi ký tự
%lf	Số thực kiểu double
%ld	Số nguyên kiểu long

# Chuỗi định dạng

## ▶ Ví dụ

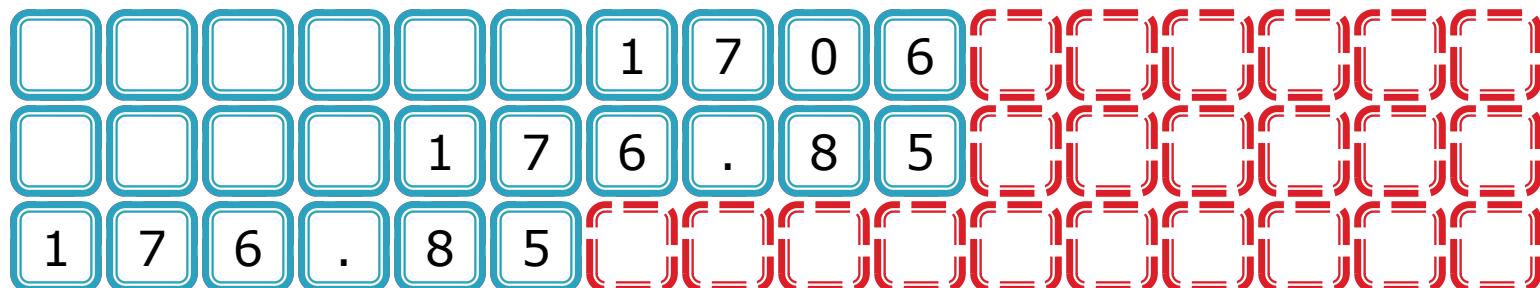
- int a = 10, b = 20;
- printf("%d", a);                      → Xuất ra 10
- printf("%d", b);                      → Xuất ra 20
- printf("%d %d", a, b);      → Xuất ra 10 20
  
- float x = 15.06;
- printf("%f", x);      → Xuất ra 15.060000
- printf("%f", 1.0/3);      → Xuất ra 0.333333

# 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");`

# Câu lệnh nhập

## Ví dụ:

- Nhập số nguyên:  
Tương đương:  
`scanf("%d%d", &i,&k);`  
`scanf("%d", &i);`  
`scanf("%d", &k);`
- Nhập số thực:  
`scanf("%f%f", &x,&y);`
- Nhập số nguyên tối đa 3 ký số:  
`scanf("%3d", &n);`
- Nhập số thực tối đa 2 số lẻ:  
`scanf("%.2f", &m);`

# Một số hàm hữu ích khác

## ▶ Các hàm trong thư viện 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

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

6. Nhập năm sinh của một người và tính tuổi của người đó.
7. Nhập 2 số a và b. Tính tổng, hiệu, tích và thương của hai số đó.
8. 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
9. Nhập điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.

# Bài tập

10. Nhập bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.
11. Nhập vào số xe (gồm 4 chữ số) của bạn. Cho biết số xe của bạn được mấy nút?
12. Nhập từ bàn phím số nguyên  $n$  là 1 số tiền (ngàn đồng), đổi tiền với các mệnh giá tương ứng: 200, 100, 50, 10, 1.

Ví dụ:  $n = 956 \rightarrow$  4 tờ 200

1 tờ 100

1 tờ 50

0 tờ 10

6 tờ 1

# Bài 4

## CÂU LỆNH ĐIỀU KIỆN & CÂU LỆNH RẼ NHÁNH

# Nội dung

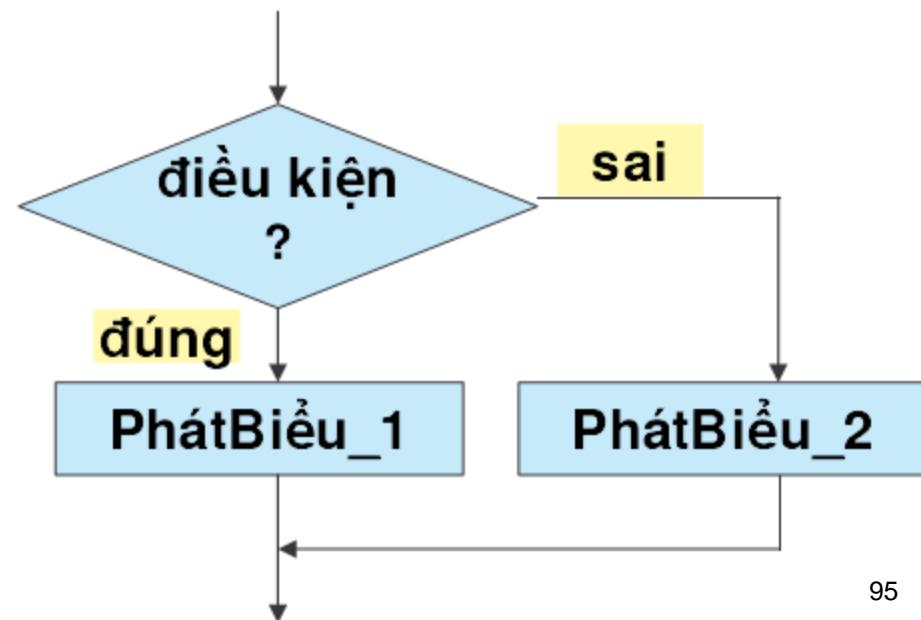
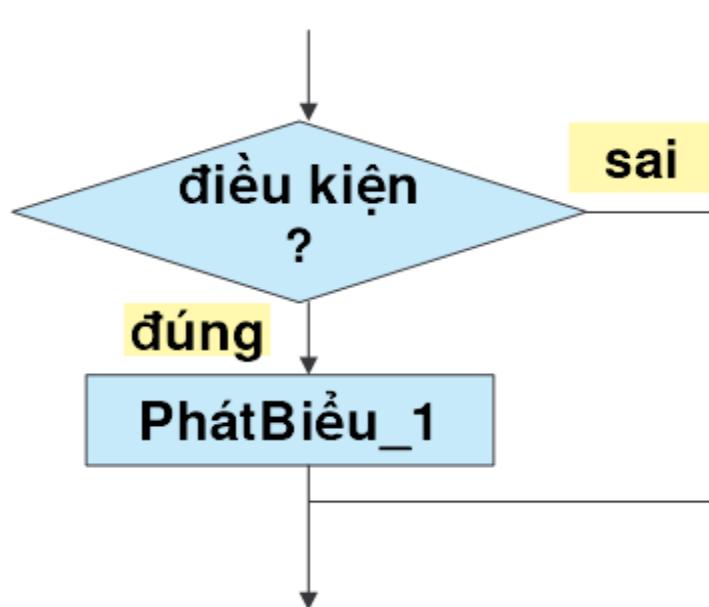
- ▶ Câu lệnh điều kiện if – Chọn 1 trong 2
- ▶ Câu lệnh rẽ nhánh switch – Chọn 1 trong n
- ▶ Ví dụ

# Lệnh if – Chọn 1/2

Lệnh if :

- Cú pháp : **if** ( điều kiện) Phát\_biểu1 ;  
Biểu thức điều kiện được tính toán (trả trị 0: sai; 1: đúng)

- Hoặc : **if** ( biểu thức) Phát\_biểu1;  
**else** Phát\_biểu2;



# Lệnh if

Thí dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int dtb=(8+7+9)/3;
    if(dtb>7)
        printf("duoc thuong");
}
```

duoc thuong

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int dtb=(3+7+9)/3;
    if(dtb>7)
        printf("duoc thuong");
    else
        printf("bi phat");
}
```

bi phat

# Lệnh if

Thí dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int dtb=(8+7+9)/3;
    if(dtb>7)
    {
        printf("\nDuoc thuong 500.000");
        printf("\nDuoc di choi");
    }
}
```

Duoc thuong 500.000  
Duoc di choi\_

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int dtb=(3+7+9)/3;
    if(dtb>7)
        printf("duoc thuong");
        printf("duoc di choi")
    else
    {
        printf("bi phat");
        printf("o nha hoc bai");
    }
}
```



# Lệnh if

## Bài tập:

Viết chương trình in ra số lớn hơn trong 2 số a,b nhập từ bàn phím

\*Cách 1:

```
if (a>b)
    max=a;
else
    max=b;
```

\*Cách 2 : max = (a>b)? a:b;

( Viết lại hoàn chỉnh chương trình: nhập, xử lý & xuất kết quả)

# Lệnh if lồng nhau

```
if (biểu thức 1)
    Phát_biểu1;
else
    if (biểu thức 2)
        if (biểu thức 3)
            Phát_biểu2;
        else
            Phát_biểu3
    else
        Phát_biểu4;
```

# Lệnh if lồng nhau

Thí dụ:

Tính điểm TB của 3 môn Toán, Lý, Hóa và xếp  
loại học lực:

ĐTB<5: yếu

5<=ĐTB<7: TB

7<=ĐTB<8: Khá

8<=ĐTB<9: Giỏi

ĐTB>=9: Xuất sắc

# Lệnh if lồng nhau

Thí dụ:

```
if (DTB<5)
    printf("\nDat loai Yeu");
else
    if (DTB<7)
        printf("\nDat loai TB");
    else
        if (DTB<8)
            printf("\nDat loai Kha");
        else
            if (DTB<9)
                printf("\nDat loai Gioi");
            else
                printf("\nDat loai Xuat sac");
```

```
if (DTB<5)
    printf("\nDat loai Yeu");
    if (DTB>=5 && DTB<7)
        printf("\nDat loai TB");
    if (DTB>=7 && DTB<8)
        printf("\nDat loai Kha");
    if (DTB>=8 && DTB<9)
        printf("\nDat loai Gioi");
    if (DTB>9)
        printf("\nDat loai Xuat sac");
```

# Lệnh if lồng nhau

Thí dụ:

Vẽ lưu đồ thuật toán và viết chương trình cho bài toán giải phương trình bậc 1:  $ax + b = 0$ .

# Lệnh switch – Chọn 1/n

## Lệnh switch

- Cú pháp : **switch** (biểu thức).

{

**case** N1 : lệnh 1; break;

**case** N2 : lệnh 2; break;

.....

[ **default** : lệnh;]

}

# Lệnh switch

- Biểu thức là giá trị nguyên : Ni là các số nguyên ( $i=1,2\dots$ ) hoặc các ký tự ('a', 'b'...)
  - Với biểu thức khác với mọi Ni => thực hiện lệnh sau default.
  - Chú ý : nếu nhóm câu lệnh sau nhãn case Ni không có câu lệnh break thì máy sẽ chuyển sang nhóm câu lệnh sau nhãn case Ni+1
- \* Ví dụ : đổi 1 số nguyên sang chuỗi

# Lệnh switch

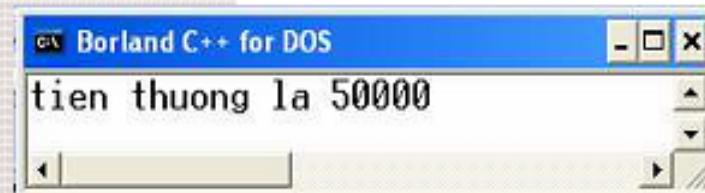
ký tự là tên các môn học

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    int ma ;
    printf(" \n cho mã cần chuyển");    scanf(" %d ", &ma);
    switch(ma)
    {
        case 0 : printf(" \n lớp tin học a "); break;
        case 1 : printf( " \n lớp tin học b"); break;
        case 2 : printf(" \n lớp trung cấp "); break;
        case 3 : printf (" \n lớp chuyên viên "); break;
        default : printf( " \n lớp thiếu tiền học phí");
    }
}
```

# Lệnh switch

Thí dụ:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int toan=7, ly=8, hoa=9;
    int tb = (toan+ly+hoa)/3;
    long tienthuong;
    switch(tb)
    {
        case 10: tienthuong=200000; break;
        case 9: tienthuong=150000; break;
        case 8: tienthuong=100000;
        default: tienthuong=50000;
    }
    printf("tien thuong la %ld", tienthuong);
}
```



# Bài tập

- 1) Viết chương trình kiểm tra 1 số nguyên n nhập từ bàn phím là số chẵn hay số lẻ.
- 2) Viết chương trình tính tiền điện (n nhập vào là số kw sử dụng) với các mức giá:

Số kw	<100	100-150	150-200	>200
ĐGiá	500	550	650	700

# Bài tập

3) Tìm những lỗi cú pháp các đoạn chương trình sau:

A/ `scanf ( "d", value);`

B/ `printf ("tích các %d và %d là %d " \n, x,y);`

C/ `printf (" phần dư của %d chia cho %d là \n ", x , y , x%y );`

D/ `if(x=y);`

`printf (" %d bằng %d \n ", x,y);`

E/ `if ( age>=65);`

`printf (" Gia ! ");`

`else`

`printf(' Tre ! ');`

# Bài tập

- 4) 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.
- 5) Nhập một chữ cái. Nếu là chữ thường thì đổi sang chữ hoa, ngược lại đổi sang chữ thường
- 6) Giải phương trình bậc nhất  $ax + b = 0$ .
- 7) Giải phương trình bậc hai  $ax^2 + bx + c = 0$

# Bài 5

# CÂU LỆNH LẬP

# Nội dung

- ▶ Câu lệnh for
- ▶ Câu lệnh while
- ▶ Câu lệnh do...while

# 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 !

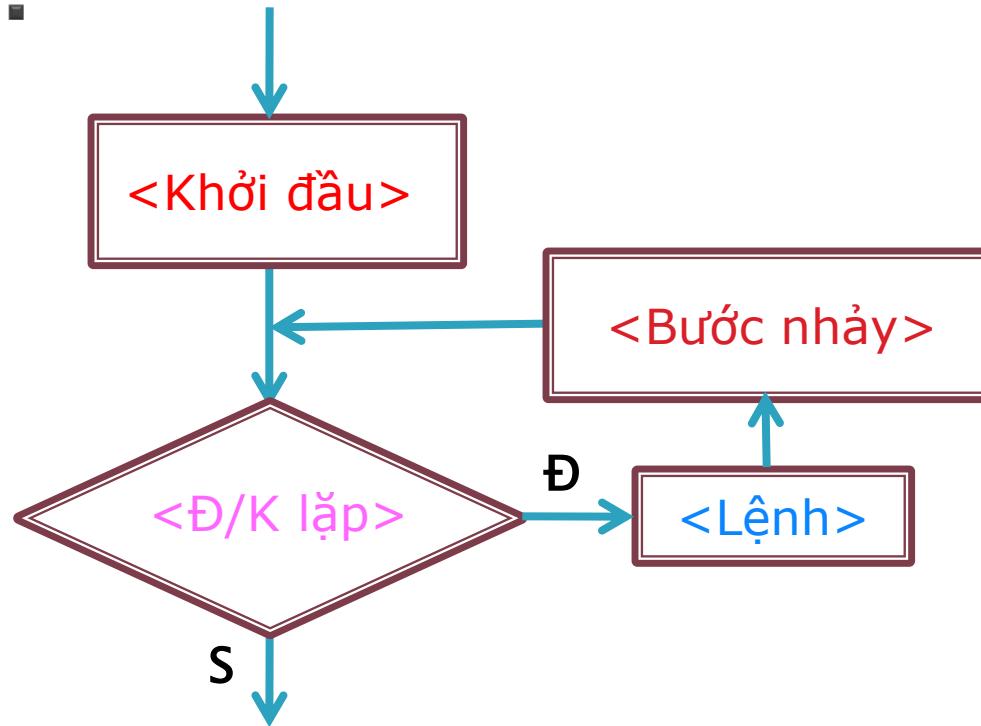
## ▶ Giải pháp

- Sử dụng cấu trúc lặp lại một hành động trong khi còn thỏa một điều kiện nào đó.

# Phát biểu lặp - Loops

- ▶ Phát biểu được lặp đi lặp lại nhiều lần
- ▶ 3 yếu tố diễn đạt vòng lặp:
  - Xác định các dữ liệu khởi tạo
  - Xác định các điều kiện còn thực thi
  - Xác định nội dung của một lần thực thi
- ▶ Các 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);
```

<Khởi đầu>

<Bước nhảy>

<Đ/K lặp>

<Lệnh>

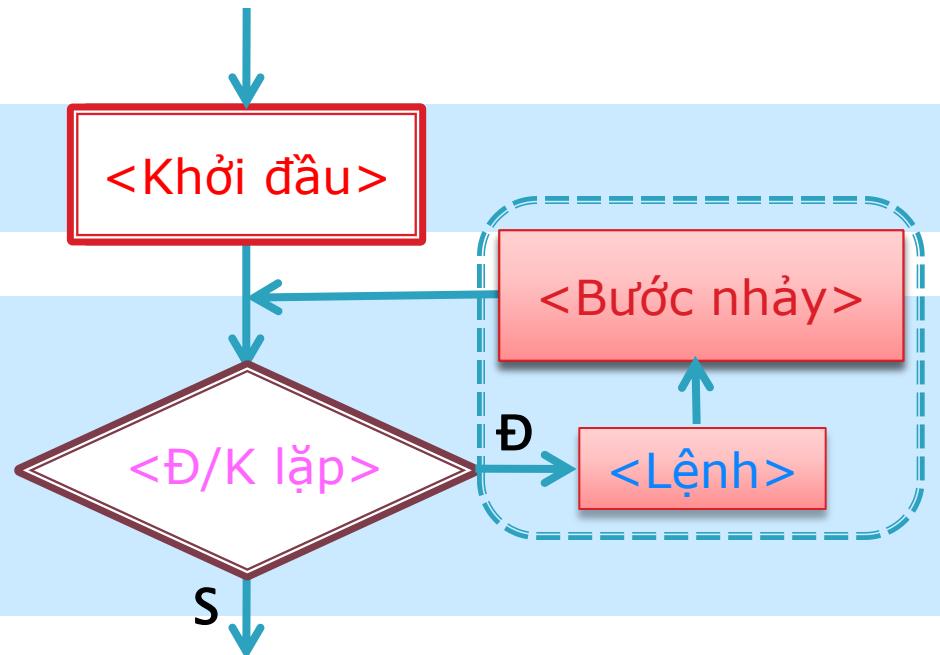
S

Đ

# 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);
```

# For

Ví dụ:

```
int i;  
for (i=1; i<=5; i++)  
{  
    printf ("Chào bạn!");  
    printf("Bạn có khỏe không?");  
}
```

Ví dụ:

```
int i;  
for (i=1; i<=5; i++)  
{  
    printf ("Chào bạn!");  
    printf("Bạn có khỏe không?");  
}
```

# For

\*Ví dụ : Tính Tổng  $S=1+2+3+..+n$

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

\* Cơ chế hoạt động :

a/Khởi gán giá trị cho  $i=1, s=0$  .

b/Kiểm tra  $i \leq n$  không

c/ + Nếu giá trị của  $i \leq n (==0)$  là sai máy sẽ ra khỏi lệnh for.

+ Nếu giá trị của  $i \leq n (!=0)$  là đúng thì máy sẽ thực hiện lệnh.

d/ Tính giá trị của  $s+=i$ , tăng  $i$  và quay lại bước kiểm tra (b)

# For

Tương đương:

int i,s;

s = 0;

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

    s+=i;

(viết lại chương trình đầy đủ)

# For

Bài tập:

1) Viết chương trình xuất ra màn hình

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

# For

Bài tập:

2) Viết chương trình xuất ra màn hình

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

3) Viết chương trình tính tổng S:

$$S = 1 + 1/2 + 1/3 + \dots + 1/n$$

$$S = n!$$

$$S = 1! + 2! + 3! + \dots + n!$$

# 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 while - 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++;
}
```

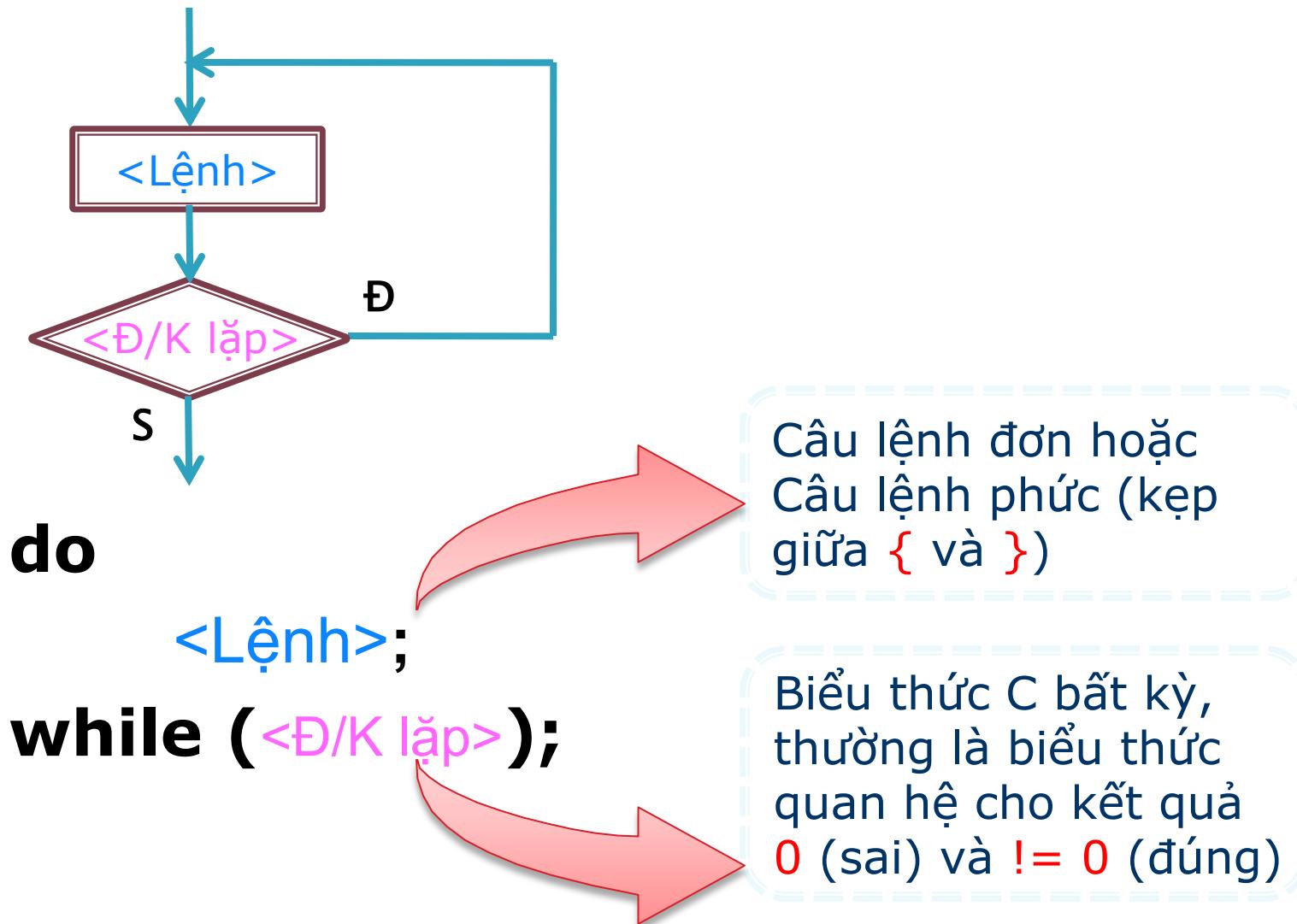
# Câu lệnh while - Một số lưu ý

- ▶ 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



# 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("Nhập 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("Nhập n: ");
    scanf("%d", &n);
}
while (n > 10);
```

# Bài tập thực hành

## 3. 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 không?

# Bài tập thực hành

4. Nhập một số nguyên dương n. Tính:
  - a.  $S = 1 + 2 + \dots + n$
  - b.  $S = 1^2 + 2^2 + \dots + n^2$
  - c.  $S = 1 + 1/2 + \dots + 1/n$
  - d.  $S = 1*2* \dots *n = n!$
  - e.  $S = 1! + 2! + \dots + n! -$
5. Nhập 3 số nguyên a, b và n với a, b < n. Tính tổng các số nguyên dương nhỏ hơn n chia hết cho a nhưng không chia hết cho b.
6. Tính tổng các số nguyên tố nhỏ hơn n ( $0 < n < 50$ )

# Bài tập thực hành

7. Nhập một số nguyên dương n. Xuất ra số ngược lại. Ví dụ: Nhập 1706 → Xuất 6071.
8. 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.
9. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b nhập từ bàn phím.
10. 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

- ▶  $S = 1/2 + 1/4 + \dots + 1/2n$
- ▶  $S = 1 + 1/3 + 1/5 + \dots + 1/(2n+1)$
- ▶  $S = 1/(1x^2) + 1/(2x^3) + \dots + 1/(nx^{n+1})$
- ▶  $S = 1/2 + 2/3 + \dots + n/(n+1)$
- ▶  $S = 1 + 1/(1 + 2) + \dots + 1/(1 + 2 + \dots + n)$
- ▶ **Liệt kê tất cả ước số của số nguyên dương n**
- ▶ **Tính tổng các ước số của số nguyên dương n**
- ▶ **Đếm số lượng ước số của số nguyên dương n**
- ▶ **Tính tổng các ước số chẵn của số nguyên dương n**

# Nhắc lại: Ngắt điều khiển

- ▶ Lệnh break
- ▶ Lệnh continue
- ▶ goto

# Lệnh break

Câu lệnh Break :

- Cú pháp : Dùng để thoát khỏi vòng lặp. Khi gặp câu lệnh này trong vòng lặp, máy ra khỏi và chỉ đến câu lệnh sau các lệnh trên. Nếu nhiều vòng lặp --- -> break sẽ thoát ra khỏi vòng lặp gần nhất.

# Lệnh continue

Lệnh continue :

- Cú pháp continue; : khi gặp lệnh này trong các vòng lặp, máy sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện vòng lặp tiếp theo.
- Đối với lệnh For máy sẽ tính lại biểu thức 3 (bt3) và quay lại bước 2.
- Đối với lệnh while, do while máy sẽ tính lại giá trị của biểu thức 1 và quay lại bước 1.

# Goto

Toán tử goto và nhãn ( label );

- Ví dụ : tiếp tục :  $st = a[i]$ ; => tiếp tục là nhãn của lệnh  $st = a[i]$ ;
- Lệnh goto nhãn => nhảy đến câu lệnh đứng sau nhãn.
- CHÚ Ý : PHẠM VI NHÃN TRONG CÙNG 1 HÀM.

# Bài tập

1/ Kiểm tra tìm lỗi :

while ( x<.= 10 )

    Total t=x;

    ++x ;

2/ Giải phương trình bậc 2 :  $ax^2 + bx + c = 0$  với a, b, c là số thực nhập từ bàn phím.

3/ Tìm các số nằm trong khoảng từ 150 đến 140 thoả tính chất số bằng tổng lập phương các chữ số của chúng :

Ví dụ :  $153 = 1^3 + 5^3 + 3^3$  hoặc  $370 = 3^3 + 7^3 + 0^3$

4/ Số tuyêt hảo là số bằng tổng các ước số thực sự của nó. Ví dụ :  $6 = 1 + 2 + 3$ . Tìm các số tuyêt hảo trong khoảng từ 1 đến 3000.

# Bài tập

- 6/ Tìm tất cả các số nguyên tố từ 2 đến 100 bằng lệnh For.
- 7/ Tìm các số nguyên có 3 chữ số sao cho tổng 3 chữ bằng tích 3 chữ. Ví dụ : 123.
- 8/ a/ Dùng lệnh while để viết chương trình tính :  
 $S1 = 1 \times 3 \times 5 \times 7 \times 9 \dots \times (2n - 1)$ .  
 $S2 = 2 \times 4 \times 6 \times 8 \times \dots \times (2n)$ .
- b/ làm lại bài trên bằng cách dùng do...while.

# Bài 6

# Hàm - Functions

# Định nghĩa

- ❑ Hàm là đoạn chương trình thực hiện trọn vẹn một công việc nhất định, có tên, đầu vào và đầu ra.
- ❑ Hàm chia cắt việc lớn bằng nhiều việc nhỏ. Nó giúp cho chương trình sáng sửa, dễ sửa, nhất là đối với các chương trình lớn.
- ❑ 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.

# Định nghĩa

Ví dụ:

Xét bài toán tính tổng  $S=1+2+3+\dots+n$

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n, s=0;
    printf("Nhập n="); scanf("%d", &n);
    for (i=1; i<=n; i++)
        s=s+i;
    printf("Tổng s=%d", s);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
void tong()
{
    int n, s=0;
    printf("Nhập n=");
    scanf("%d", &n);
    for (i=1; i<=n; i++)
        s=s+i;
    printf("Tổng s=%d", s);
}
// kết thúc hàm tính tong
void main()//chương trình chính
{
    tong();
    getch();
}
```

# 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;
}
```

# Hàm - 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("Nhập 2 số nguyên: ");
    scanf("%d%d", &x, &y);
    printf("%d cộng %d bằng %d", x, y, x + y);
}
```

# Khai báo

Hoàn tất tác vụ  
thì có kết quả  
là gì.---> kiểu trả về

Việc này là việc gì

Để làm việc này thì cần dữ liệu gì  
Tham số, parameters  
(data để hàm tác động)

ReturnType

FunctionName

( Type param1, Type & param2,...)

Hàm không trả trị:  
void

```
{ <code>  
.....  
.....  
}
```

Muốn làm thì làm thế nào?  
- thân hàm  
- function body  
- Trình tự tiến hành công việc

Hàm trả trị phải  
có return Exp;

# Khai báo

Ví dụ:

Trở lại bài toán  $S=1+2+3+\dots+n$

ReturnType

FunctionName

( Type param1, Type & param2,...)

```
{ <code>
    .....
    .....
}
```

int

tinh tong

( int n)

```
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
```

# Khai báo

Ví dụ:

Hàm tìm giá trị lớn nhất trong 2 số nguyên x, y:

ReturnType

FunctionName

( Type param1, Type & param2,...)

```
{ <code>
  .....
  .....
}
```

int

maxxy

( int x, int y)

```
{
  if (x>y) return x;
  else return y;
}
```

# Gọi hàm

Gọi hàm: Yêu cầu thực thi 1 tác vụ (hàm) với dữ liệu cụ thể

Cú pháp gọi hàm:

Tên hàm (các dữ liệu cho từng tham số)

Các trị cho từng tham số cần thiết:

- Đủ tham số
- Đúng kiểu tham số đã khai báo trong hiện thực hàm

Có thể gọi hàm lồng nhau

# Gọi hàm

Ví dụ: In ra số lớn nhất trong 3 số

```
#include <stdio.h>
#include <conio.h>
int maxxy(int x, int y)
{
    if (x>y)
        return x;
    return y;
}
void main()
{
    int a, b,c;
    printf("Nhập a, b, c"); scanf("%d %d %d", &a, &b, &c);
    printf("Số lớn nhất: %d", maxxy(maxxy(a,b),c));
    getch();
}
```

# Khai báo

Cách 1 :

```
#include <stdio.h>
#include <conio.h>
float giaithua ( int n )
{
    int i ;float KQ ;
    for ( KQ=1,i =1 ; i<=n ; i ++ )
        KQ = KQ * i ;
    return KQ ;
}
```

# Khai báo

```
void main ( ) /* khai báo biến toàn cục nếu có */  
{  
    int n ;  
    printf ( " Nhập n = " ); scanf ( " %d ", &n );  
    printf ( " %d giai thừa là % f ", n, giaithua (n) );  
    getch ();  
}
```

# Khai báo

Cách 2 :

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

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

/\*Khai báo prototype\*/ mục đích hàm đặt ở đâu  
cũng được không cần trước hàm gọi

# Khai báo

```
float giaithua ( int n );
```

```
void main ()
```

```
{
```

```
}
```

```
/* Chi tiết hàm giải thừa */
```

```
float giaithua ( int n )
```

```
{ ... return KQ };
```

Chú ý : - Kiểu của hàm cùng kiểu giá trị cần trả về.

# Khai báo

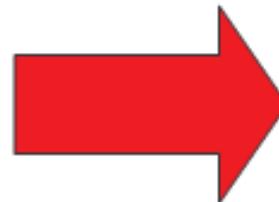
- Các hàm độc lập, không được lồng nhau.
- Kiểu void tên hàm () : không cần trả về giá trị nào, hiểu ngầm là trả về int.
- Ở cách 1 : hàm ở trên không được gọi hàm dưới.
- Ở cách 2 : các hàm gọi được lẫn nhau.

# Bài tập

- 1) Viết lại chương trình in ra số lớn nhất trong 3 số nguyên a, b, c (viết hàm số lớn nhất của 2 số riêng)
- 2) Viết hàm tính diện tích của hình chữ nhật (nhập vào dài và rộng). Viết chương trình chính nhập vào dài, rộng của 2 hcn. So sánh và in ra diện tích hcn lớn hơn.

# Hàm trả trị và không trả trị

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n=10, S=0;
    clrscr();
    for(int i=1; i<=n; i++) S+=i;
    printf("Tong S=%d", S);
    getch();
}
```



```
#include <stdio.h>
#include <conio.h>
void TinhTong()
{
    int n=10, S=0;
    for(int i=1; i<=n; i++) S+=i;
    printf("Tong S=%d", S);
}
void main()
{
    clrscr();
    TinhTong();
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
int TinhTong(int n)
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
void main()
{
    int n=10;
    clrscr();
    printf("Tong S=%d", TinhTong(n));
    getch();
}
```



# Hàm không trả trị không thể gọi trong hàm printf

```
#include <stdio.h>
#include <conio.h>
void TinhTong()
{
    int n=10, S=0;
    for(int i=i; i<=n; i++) S+=n;
    printf("Tong S=%d", S);
}
void main()
{
    clrscr();
    printf("Tong %d", TinhTong());
    getch();
}
```



```
#include <stdio.h>
#include <conio.h>
void TinhTong()
{
    int n=10, S=0;
    for(int i=i; i<=n; i++) S+=n;
    printf("Tong S=%d", S);
}
void main()
{
    clrscr();
    TinhTong();
    getch();
}
```

Lỗi

# Hàm trả trị có thể gọi trong hàm printf hoặc trong phát biểu đơn

```
#include <stdio.h>
#include <conio.h>
int TinhTong(int n)
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
void main()
{
    int n=10;
    clrscr();
    printf("Tong S=%d",TinhTong(n));
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
int TinhTong(int n)
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
void main()
{
    int n=10, S;
    clrscr();
    S= TinhTong(n);
    printf("Tong S=%d",S);
    getch();
}
```

# Tham số hình thức

- Tham số của hàm là dữ liệu mang tính hình thức → Tham số hình thức (formal parameters) mang tính tổng quát hóa.

```
int TinhTong(int n)
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
```

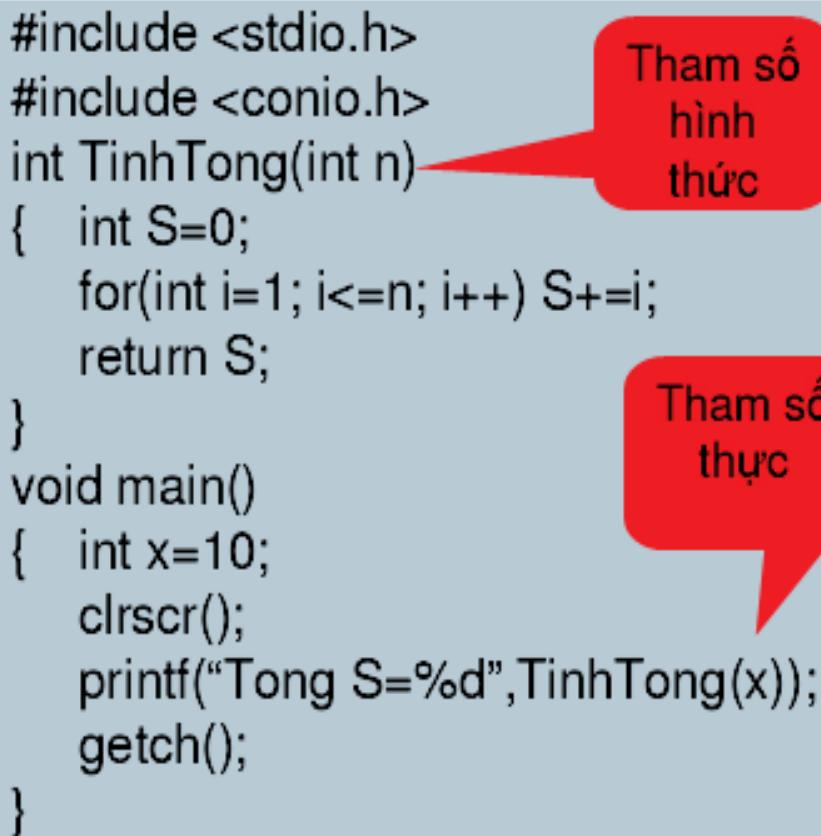
- n là tham số hình thức. Biến n không nhất thiết phải giống với biến khai báo trong hàm main

```
void main()
{
    int x=10;
    clrscr();
    printf("Tong S=%d",TinhTong(x));
    getch();
}
```

# Tham số thực (đối số) của hàm

- Còn gọi tham số thực – Real parameters
- Là dữ liệu thực sự mà hàm sẽ tác động.
- Được đưa vào cho hàm tại nơi sử dụng hàm (nơi gọi hàm).

```
#include <stdio.h>
#include <conio.h>
int TinhTong(int n)
{
    int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
void main()
{
    int x=10;
    clrscr();
    printf("Tong S=%d",TinhTong(x));
    getch();
}
```



Tham số hình thức

Tham số thực

# Tham trị và tham biến

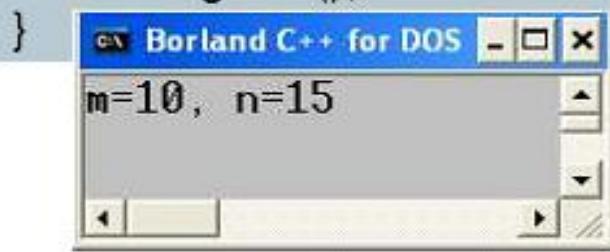
- Tham trị (value parameter): Tham số khi nhận dữ liệu cụ thể để hàm thực thi là bản **copy** của dữ liệu gốc
- Tham biến (reference parameter): Tham số khi nhận dữ liệu cụ thể để hàm thực thi chính là dữ liệu gốc
- ReturnType FuncName (Type vParam, Type &rParam)

# Tham trị và tham biến

## Ví dụ: Chương trình hoán vị 2 số

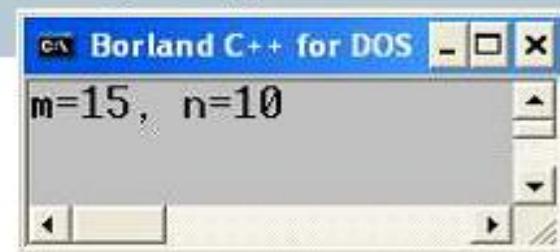
### Tham trị

```
#include <stdio.h>
#include <conio.h>
void swap(int x, int y)
{
    int t=x;
    x=y;
    y=t;
}
void main()
{
    int m=10, n=15;
    clrscr();
    swap(m,n);
    printf("m=%d, n=%d",m,n);
    getch();
}
```



### Tham biến

```
#include <stdio.h>
#include <conio.h>
void swap(int &x, int &y)
{
    int t=x;
    x=y;
    y=t;
}
void main()
{
    int m=10, n=15;
    clrscr();
    swap(m,n);
    printf("m=%d, n=%d",m,n);
    getch();
}
```



# Tham trị và tham biến

## Có thể truyền hằng cho tham trị

```
#include <stdio.h>
#include <conio.h>
int TinhTong(int n)
{   int S=0;
    for(int i=1; i<=n; i++) S+=i;
    return S;
}
void main()
{   clrscr();
    printf("Tong S=%d",TinhTong(10));
    getch();
}
```

# Tham trị và tham biến

## Không thể truyền hằng cho tham biến

```
#include <stdio.h>
#include <conio.h>
void swap(int &x, int &y)
{
    int t=x;
    x=y;
    y=t;
}
void main()
{
    clrscr();
    swap(10,15);
    getch();
}
```



Lỗi

# Nguyên tắc hoạt động của hàm

- Khi gấp một lời gọi hàm thì hàm bắt đầu được thực hiện theo 4 bước sau
  - Cấp phát bộ nhớ cho các đối số và các biến cục bộ
  - Gán giá trị của các tham số thực cho các đối số tương ứng
  - Thực hiện các câu lệnh trong thân hàm
  - Khi gấp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ xóa các đối, các biến cục bộ và thoát khỏi hàm

# Cấu trúc tổng quát của 1 chương trình

- ❑ Chương trình gồm nhiều hàm viết theo trình tự
  - Các #include
  - Các #define
  - Khai báo các đối tượng dữ liệu ngoài
  - Khai báo các hàm
  - Hàm main
  - Định nghĩa các hàm
- ❑ Chú ý: hàm main có thể đặt sau hoặc xen vào giữa các hàm khác

# Bài tập

1) Viết chương trình tính:

$S = 1! + 2! + 3! + \dots + n!$  trong đó viết hàm tính  $x!$  riêng  
và sử dụng lại để tính  $S$

2) Viết chương trình xuất ra màn hình các số nguyên tố  
từ 1 đến 100 (viết hàm kiểm tra 1 số  $x$  có phải là số  
nguyên tố không)

# Bài tập

- 3) Viết chương trình tính nhập vào phân số a/b, rút gọn, quy đồng mẫu và xuất ra phân số kết quả (viết hàm USCLN, BSCNN riêng )
- 4) Viết chương trình tính:

$$s(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$$

# Nhắc lại tham biến và tham trị

```
void USC_BSC(int x, int y, int &usc, int &bsc)
{
    int max, min, i;
    if (x>y)
    {
        max = x;
        min = y;
    }
    else
    {
        max=y;
        min=x;
    }
    //USC
    for (i=min; i>0; i--)
        if (x%i==0 && y%i==0)
            break;
    usc=i;
    //BSC
    for (i=max; i<=x*y; i++)
        if (i%x==0 && i%y==0)
            break;
    bsc=i;
}
```

```
void main()
{
    int a, b, boi, uoc;
    printf("a="); scanf("%d", &a);
    printf("\nb="); scanf("%d", &b);
    //Gọi ham
    USC_BSC(a,b,uoc,boi);
    printf("Boi so: %d, Uoc so: %d",boi,uoc);
    getch();
}
```

# Hàm 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  $S(n-2), \dots, S(2), S(1), S(0) = 1$

# 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;
}
```

# Hàm đệ quy

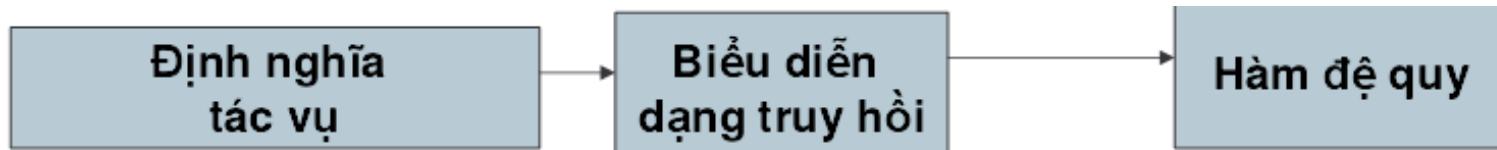
- Định nghĩa: Khi một hàm gọi tới chính nó, đó là hàm đệ quy.
- (Giai thừa của n) =  $1, n \leq 1$   
 $n * (\text{Giai thừa của } n-1)$

$$\begin{aligned}n! &= 1 * 2 * 3 * 4 * \dots * (n-1) * n \\&= (n-1)! * n \\&= n * (n-1)!\end{aligned}$$

# Đặc điểm của đệ quy

- Ta hiểu được định nghĩa “Người = con của hai người khác” với tiên đề: có 2 người đầu tiên
- Ta hiểu được định nghĩa “Thư mục = các Thư mục con + các tập tin” với tiên đề: có thư mục ban đầu là thư mục ổ đĩa.
- Ta làm được  $n! = n^*(n-1)!$  vì tiên đề  $1! = 1$
- Đệ quy là khả thi nếu **đệ quy có chận (còn gọi là điều kiện biên)**

# Cách viết hàm đệ quy



$n! = 1 * 2 * 3 * \dots * n$

$n! = 1, n \leq 1$   
 $= n * (n-1)!$

Tính giai thừa của  
số nguyên n  
Tên hàm: **Gt(int n)**

Kết quả là 1,  $n \leq 1$   
Kết quả là  $n * Gt(n-1)$

```
double Gt(int n)  
{ if (n<2) return 1;  
    return n * Gt(n-1);  
}
```

Nhận xét:  
Định nghĩa sao viết vậy.

# Ví dụ:

Viết hàm đệ quy tính  $n!$ :

```
int Giaithua(int n)
```

```
{
```

```
    if (n<=1)
```

```
        return 1;
```

```
    else
```

```
        return n*Giaithua(n-1);
```

```
}
```

# Bài 7

# Mảng - Arrays

# Giới thiệu

Bài toán:

Nhập 20 số nguyên, bài toán có nhu cầu lưu lại 20 số nguyên này để xử lý: sắp xếp, tìm kiếm,...

→ Không thể khai báo 20 biến số nguyên và xử lý tính toán trên từng biến được → ???

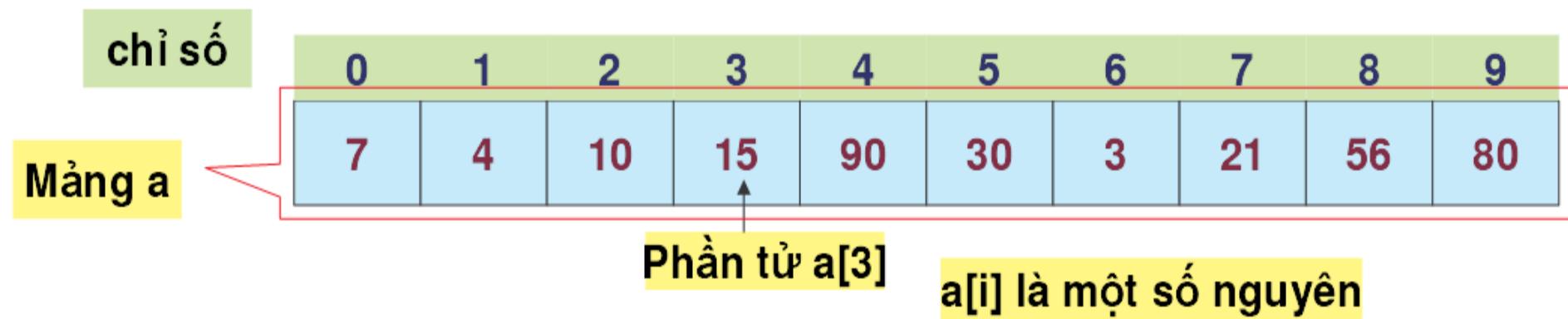
# Nội dung

- ❑ Những khái niệm về mảng
- ❑ Khai báo mảng
- ❑ Quản lý các phần tử
- ❑ Lưu trữ mảng
- ❑ Mảng 1 chiều
- ❑ Một số thuật toán trên mảng
- ❑ Mảng nhiều chiều

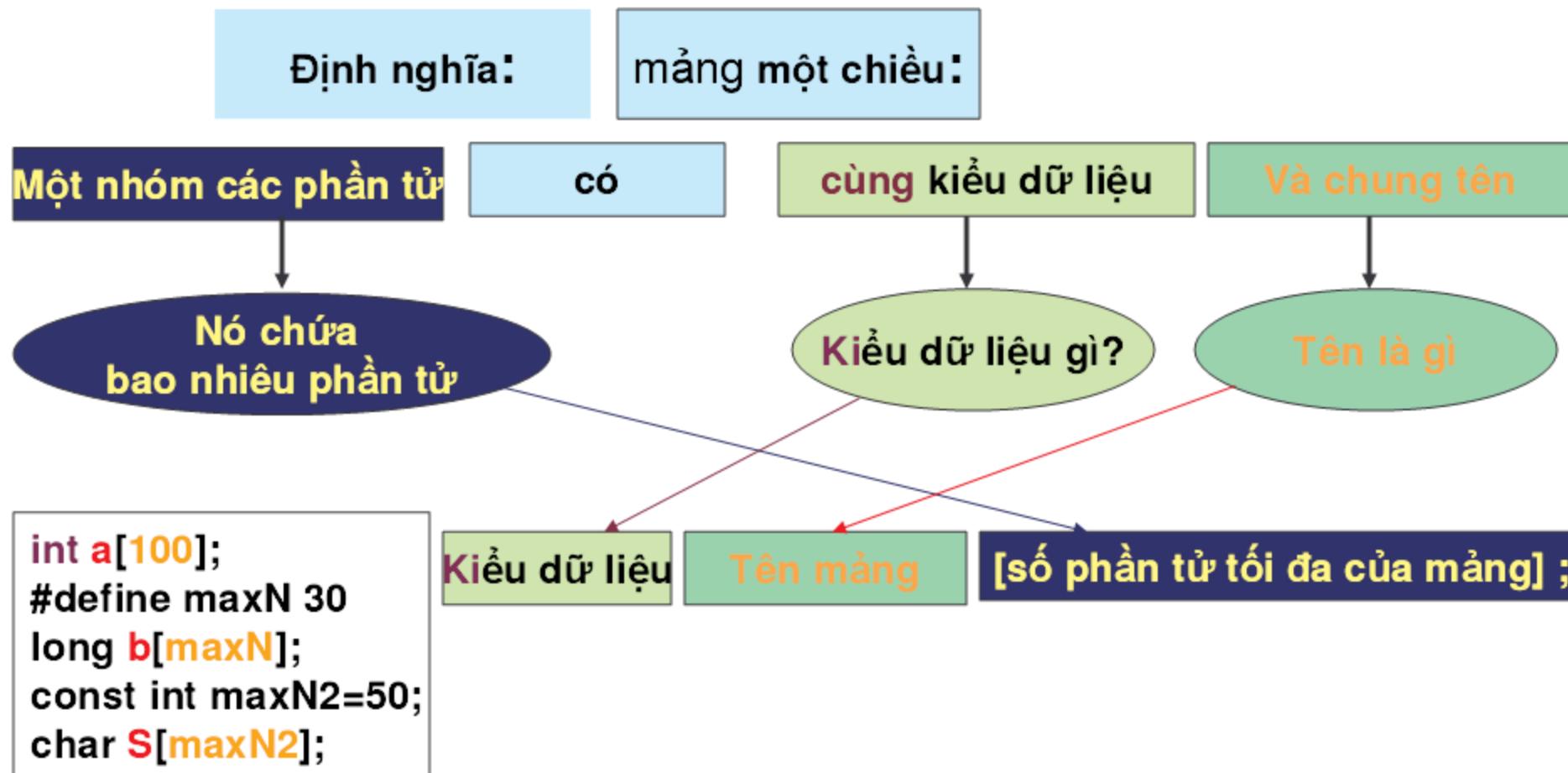
# Những khái niệm về mảng

- ❑ **Array**: là một nhóm các phần tử có cùng kiểu và chung tên
- ❑ **Element**: phần tử của mảng. Có thể là những kiểu dữ liệu cơ bản như int, long, char,... hoặc các kiểu do người dùng định nghĩa
- ❑ **Index**: số nguyên thể hiện vị trí của phần tử xác định. Trong C chỉ số (index) bắt đầu bằng 0
- ❑ **Dimension**: chiều của mảng, cũng chính là số lượng các chỉ số giúp ta xác định một phần tử của mảng. Mảng 1 chiều cần 1 chỉ số, 2 chiều cần 2 chỉ số...

# Ví dụ mảng 1 chiều



# Khai báo mảng 1 chiều



# Quản lý các phần tử

- ❑ Một mảng có thể chứa số lượng phần tử tối đa xác định → gọi là kích thước mảng
- ❑ Tại một thời điểm, nó có thể chứa một số phần tử xác định → nên khai báo như sau:

```
int a[100] ;  
int n ; // số phần tử hiện thời của mảng
```

Một phần tử của mảng được xác định bởi chỉ số của nó trong mảng.  
Vì thế, vì thế phần tử thứ i trong mảng được định nghĩa bởi: a[i]

# Khởi tạo cho mảng 1 chiều

- Khởi tạo = Khai báo + gán giá trị cho mảng

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int a[5]={4,6,3,8,9};
    for(int i=0; i<5; i++)
        printf("\na[%d]=%d",i, a[i]);
    getch();
}
```

Cú pháp khởi tạo mảng

```
a[0]=4
a[1]=6
a[2]=3
a[3]=8
a[4]=9
```

# Duyệt mảng 1 chiều

- ❑ Xử lý mảng = Xử lý nhóm các phần tử = Xử lý mỗi phần tử trong mảng
- ❑ Một phần tử được xác định bởi chỉ số
  - Duyệt mảng xuôi

Duyệt mảng  
không điều kiện

```
for (i=0; i<n; i++) Xử lý a[i];
```

Duyệt mảng  
có điều kiện

```
for (i=0; i<n; i++)  
if (Điều kiện) xử lý a[i];
```

- Duyệt mảng ngược

```
for (i=n-1; i>=0; i--) xử lý a[i];
```

```
for (i=n-1; i>=0; i--)  
if (điều kiện) xử lý a[i];
```

# Duyệt mảng 1 chiều

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int a[5]={4,6,3,8,9},S1=0, S2=0, S3=0, i;
    printf("nội dung mảng:");
    for(i=0; i<5; i++)
        printf("%3d", a[i]);
    for(i=0; i<5; i++)
        S1 += a[i];
    printf("\nTổng mảng: %d", S1);
    for(i=0; i<5; i++)
        if(a[i]%2==0) S2+=a[i];
    printf("\nTổng mảng chẵn: %d", S1);
    for(i=4; i>=0; i--)
        if(a[i]%2) S3+=a[i];
    printf("\nTổng mảng lẻ: %d", S1);
    getch();
}
```

```
nội dung mảng: 4 6 3 8 9
Tổng mảng: 30
Tổng mảng chẵn: 30
Tổng mảng lẻ: 30
```

Duyệt mảng không có điều kiện

Duyệt mảng có điều kiện

Duyệt mảng ngược

# Nhập / xuất mảng

- Viết chương trình nhập vào mảng n phần tử số nguyên. Xuất ra màn hình.

```
void main()
{
    int a[100];
    int i, n;
    printf("Nhập n="); scanf("%d", &n);
    for (i =0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
    }
    //xuat mang
    for (i=0; i<n; i++)
        printf("%d ", a[i]);
    getch();
}
```

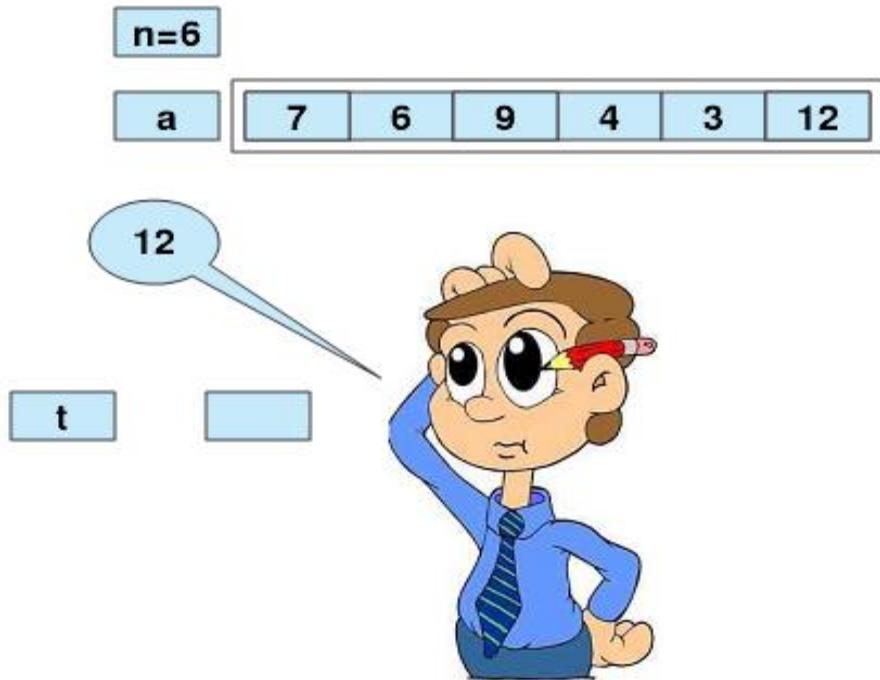
# Nhập / xuất mảng

- Viết chương trình nhập vào mảng n phần tử số nguyên. Xuất ra màn hình các số lẻ trong mảng.

```
void main()
{
    int a[100];
    int i, n;
    printf("Nhập n="); scanf("%d", &n);
    for (i =0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
    }
    //xuat mang
    for (i=0; i<n; i++)
        if (a[i]%2==1)
            printf("%d ", a[i]);
    getch();
}
```

# Nhập / xuất mảng

- Viết chương trình nhập vào mảng n phần tử số nguyên. Xuất ra màn hình giá trị lớn nhất trong mảng.



```
// Thuật toán lấy giá trị lớn nhất
max=a[0];
i=1..n-1
  if (max<a[i]) max=a[i];
return max;
```

Bài tập: tìm giá trị nhỏ nhất

# Nhập / xuất mảng

- Viết chương trình nhập vào mảng n phần tử số nguyên. In ra màn hình:
  - 1) Tổng các phần tử trong mảng
  - 2) Trung bình của các phần tử trong mảng
  - 3) Tổng các phần tử lẻ trong mảng
  - 4) Đếm các phần tử không âm
  - 5) Kiểm tra xem mảng có đối xứng hay không
  - 6) Nhập vào giá trị x. Tìm xem x có trong mảng hay không
  - 7) Nhập vào giá trị x. Đếm số lần xuất hiện của x trong mảng
  - 8) Kiểm tra mảng có phải là mảng tăng hay không
  - 9) Kiểm tra mảng có chứa toàn số dương/ âm không
  - 10) Tách mảng

# Nhập/xuất mảng – Sử dụng hàm

```
#include <conio.h>
#include <stdio.h>
void NhapMang(int a[], int n)
{
    for (int i=0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
    }
}
void XuatMang(int a[], int n)
{
    for (i=0; i<n; i++)
        printf("%d", a[i]);
}

void main ()
{
    int a[100], n;
    printf("Nhập n: ");
    scanf("%d", &n);
    NhapMang(a, n);
    XuatMang(a, n);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
void NhapMang(int a[], int &n)
{
    int i;
    printf("Nhập n: ");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
    }
}
void XuatMang(int a[], int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d ", a[i]);
}
void main()
{
    int a[100], n;
    NhapMang(a, n);
    XuatMang(a, n);
    getch();
}
```

# Nhập / xuất mảng – Sử dụng hàm

- ❑ Hàm tìm giá trị lớn nhất của mảng

```
int MaxMang(int a[], int n)
{
    int i, max = a[0];
    for (i = 0; i < n; i++)
        if (a[i] > max)
            max = a[i];
    return max;
}
```

- ❑ Tìm x có trong mảng không

```
int Timx(int a[], int n, int x)
{
    int i;
    int vitri = -1;
    for (i = 0; i < n; i++)
        if (a[i] == x)
    {
        vitri = i;
    }
    return vitri;
}
```

# Bài tập

- Làm lại các bài tập trên, viết các hàm cho từng chức năng
  - 1) Nhập mảng
  - 2) Xuất mảng
  - 3) Tổng các phần tử trong mảng
  - 4) Trung bình của các phần tử trong mảng
  - 5) Tổng các phần tử lẻ trong mảng
  - 6) Đếm các phần tử không âm
  - 7) Kiểm tra xem mảng có đối xứng hay không
  - 8) Nhập vào giá trị x. Tìm xem x có trong mảng hay không
  - 9) Nhập vào giá trị x. Đếm số lần xuất hiện của x trong mảng
  - 10) Kiểm tra mảng có phải là mảng tăng hay không

# Sắp xếp mảng

- Mảng a có 5 phần tử:

4 6 2 7 9

- Mảng a sau khi sắp xếp tăng dần:

2 4 6 7 9

→ Thuật toán???

# Bài tập

- Sắp xếp mảng tăng dần, giảm dần

# Mảng nhiều chiều

## □ Định nghĩa:

- Mảng nhiều chiều là 1 mảng mà các phần tử của nó được xác định bằng nhiều chỉ mục (bao nhiêu chiều có bấy nhiêu chỉ mục để xác định)
- Mảng 2 chiều là mảng nhiều chiều đơn giản và được sử dụng nhiều nhất
- Mảng 2 chiều giống như 1 bảng, gồm nhiều dòng và nhiều cột

# Mảng 2 chiều – Ma trận

Ma trận, mảng 2 chiều:

Một nhóm các phần tử có cùng kiểu, chung tên.

Các phần tử được xác định bằng số dòng và số cột.

chỉ số dòng

chỉ số cột

	0	1	2	3	4
0	5	7	9	2	3
1	12	5	7	4	6
2	1	8	9	0	3
3	2	5	6	3	4

Ma trận m

$m[2][3]$

$m[i][j]$  is an integer

# Khai báo ma trận

- **Ma trận:** một nhóm các phần tử mà chúng có **cùng kiểu**, **chung tên**. Các phần tử được nhóm theo **dòng** và **cột**

```
DataType mt[ maxRow ] [maxColumn] ;  
int r, c ; // Số dòng và cột hiện thời
```

```
#define maxRow1 20  
#define maxCol1 50  
int m1 [maxRow1] [maxCol1];  
int r1, c1;  
const int maxRow2=30;  
const int maxCol2=60;  
long m2 [maxRow2] [maxCol2];  
int r2, c2;  
char m3 [20] [25];  
int r3, c3;
```

# Khởi tạo ma trận

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int mt1[2][3] = {1,2,3,4,5,6};
    int mt2[2][3] = {{2,1,4},{4,7,6}};
    int i,j;    clrscr();
    printf("ma tran 1\n");
    for (i=0;i<2;i++)
    {
        for(j=0; j<3; j++) printf("%4d",mt1[i][j]);
        printf("\n");
    }
    printf("ma tran 2\n");
    for (i=0;i<2;i++)
    {
        for(j=0; j<3; j++) printf("%4d",mt2[i][j]);
        printf("\n");
    }
    getch();
}
```

ma	tran	1
1	2	3
4	5	6
ma	tran	2
2	1	4
4	7	6

# Duyệt ma trận

```
for (int i=0;i<row, i++)  
{ for (int j=0;j<col;j++) Xử lý m[i][j];  
   Những thao tác khác;  
}  
  
// Tính tổng ma trận  
S=0;  
for (int i=0;i<row, i++)  
{ for (int j=0;j<col;j++) S+= m[i][j];  
}  
  
// Xuất ma trận  
for (int i=0;i<row, i++)  
{ for (int j=0;j<col;j++) printf("%3d",m[i][j]);  
    printf("\n");  
}
```

row=4

col=5

m	0	1	2	3	4
0	5	7	9	2	3
1	2	5	7	4	6
2	1	8	9	0	3
3	2	5	6	3	4

# Quản lý ma trận

- Quản lý ma trận cũng như quản lý các mảng 1 chiều
- Một số thao tác trên ma trận:
  - tìm phần tử lớn nhất, nhỏ nhất.
  - Tổng ma trận, trung bình ma trận
  - Xuất đường chéo chính, phụ

# Bài 8

# Chuỗi - String

# Khái niệm chuỗi

- ❑ Chuỗi là 1 mảng các ký tự kết thúc bằng ký tự NULL ('\0')
- ❑ Ký tự '\0' được tự động thêm vào trong chuỗi
- ❑ Mỗi ký tự trong chuỗi được lưu trữ theo định dạng mã ASCII

# Nhập xuất chuỗi

- char S[80];
- Nhập chuỗi  
scanf("%s", S);  
gets(S);
- Xuất chuỗi  
printf(S);  
printf("%s", S);  
puts(S);

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    char S[20];
    printf("Nhập chuỗi bằng hàm scanf ");
    scanf("%s", &S);
    printf("\nChuỗi vừa nhập: %s", S);
    printf("\nNhập chuỗi bằng hàm gets");
    gets(S);
    printf("\nChuỗi vừa nhập:");
    puts(S);
    getch();
}
```

```
Nhap chuoi bang ham scanf Hello World
Chuoi vua nhap: Hello
Nhap chuoi bang ham gets
Chuoi vua nhap: World
```

- Hàm **scanf** không nhập chuỗi có khoảng trắng được
- Để nhập chuỗi có khoảng trắng ta dùng hàm **gets**
- Nếu vùng đệm keyboard có dữ liệu thì dữ liệu sẽ chuyển vào biến.
- Để nhập một chuỗi mới chúng ta phải xoá bộ đệm keyboard bằng hàm **fflush(stdin)**

# Nhập xuất chuỗi

puts(S)  
tương đương với  
printf("%s\n",S)

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    char S[20];
    printf("Nhập chuỗi bằng hàm scanf ");
    scanf("%s", &S);
    printf("\nChuỗi vừa nhập: %s", S);
    printf("\nNhập chuỗi bằng hàm gets");
    → fflush(stdin);
    gets(S);
    printf("Chuỗi vừa nhập:");
    puts(S);
    getch();
}

Nhập chuỗi bằng hàm scanf Hello world
Chuỗi vừa nhập: Hello
Nhập chuỗi bằng hàm getsHi
Chuỗi vừa nhập:Hi
```

# Truy xuất một ký tự của chuỗi

- Chuỗi là một mảng các ký tự. Vì thế chúng ta có thể truy xuất vào phần tử ký tự của nó bằng chỉ số.

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
{
    clrscr();
    char S[20];
    printf("Nhập chuỗi: ");
    gets(S);
    int i=0;
    while(S[i]!=NULL)
    {
        S[i] = toupper(S[i]);
        i++;
    }
    printf("Chuỗi vừa nhập được viết hoa:");
    puts(S);
    getch();
}
```

```
Nhập chuỗi: Hello world
Chuỗi vừa nhập được viết hoa:HELLO WORLD
```

# Thư viện string.h

## STRING.H

### Functions

\_fmemccpy  
\_fmemset  
\_fstrcsn  
\_fstrncat  
\_fstrpbrk  
\_fstrrstr  
memcmp  
move data  
strchr  
strdup  
strlwr  
strnicmp  
strset  
strupr

\_fmemchr  
\_fstreat  
\_fstrdup  
\_fstrncmp  
\_fstrrchr  
\_fstrtok  
memcpy  
movmem  
strncpy  
\_strerror  
strncat  
strnset  
strspn

\_fmemcmp  
\_fstrchr  
\_fstricmp  
\_fstrnicmp  
\_fstrrev  
\_fstrupr  
memicmp  
setmem  
strncpy  
\_strerror  
strncmp  
strpbrk  
strrchr  
strtok

\_fmemcpy  
\_fstrcmp  
\_fstrlen  
\_fstrncpy  
\_fstrset  
memcpy  
memmove  
stpcpy  
strcpy  
stricmp  
strncmp  
strrchr  
strtok

\_fmemicmp  
\_fstrcpy  
\_fstrlwr  
\_fstrnset  
\_fstrspn  
memchr  
memset  
strcat  
strcsn  
strlen  
strncpy  
strrev  
strxfrm

# Thư viện string.h

- `size_t strlen (const char*S)`
- `char *strcpy(char *dest, const char *src);`
- `int strcmp(const char *s1, const char*s2);`
- `char *strcat(char *dest, const char *src);`
- `char *strlwr(char *s);`
- `char *strupr(char *s);`
- `char *strrev(char *s);`
- `char *strstr(const char *s1, const char *s2);`
- `char *strchr(const char *s, int c);`

# Một số bài tập về chuỗi

- Nhập một chuỗi từ bàn phím, xuất ra chuỗi đảo ngược. Vd: đảo của “abcd egh” là “hge dcba”
- Nhập họ tên của một người, in hoa các ký tự đầu của từng từ và xuất ra màn hình.
- Nhập vào một chuỗi rồi sau đó xuất ra màn hình dưới dạng mỗi từ một dòng  
Vd: Nguyễn Văn Minh  
xuất ra: Nguyễn

Văn

Minh

# Bài 9

# Cấu trúc

# Khái niệm cấu trúc

- ❑ Cấu trúc bao gồm một số các phần tử dữ liệu mà chúng không cùng kiểu, được nhóm lại với nhau
- ❑ Cấu trúc có thể chứa nhiều phần tử mong muốn
- ❑ Một item được gọi là một phần tử, thành phần hay thuộc tính

# Khai báo cấu trúc

```
struct StructureName  
{ Type1 Field1;  
  Type2 Field2;  
  .....  
};
```

```
struct SINHVIEN  
{ char Hoten[15];  
  char NgaySinh[20];  
  char DiaChi[40];  
  int diem1, diem2, diem3;  
};
```

# Khai báo và khởi tạo cấu trúc

```
#include <conio.h>
#include <stdio.h>
struct SINHVIEN
{ char Hoten[15], NgaySinh[20], DiaChi[40];
  int diem1, diem2, diem3;
};
void main()
{ clrscr();
  SINHVIEN sv = {"Nguyen Van A", "01/01/2001", "100 Hung
                  Vuong, Q10, TPHCM", 8, 9, 10};
  printf("%s %s %s %d %d %d", sv.Hoten, sv.NgaySinh, sv.DiaChi,
         sv.diem1, sv.diem2, sv.diem3);
  getch();
}
```

Nguyen Van A 01/01/2001 100 Hung Vuong, Q10, TPHCM 8 9 10

# Nhập và truy xuất các phần tử cấu trúc

```
#include <conio.h>
#include <stdio.h>
struct SINHVIEN
{ char Hoten[15], NgaySinh[20], DiaChi[40];
  int diem1, diem2, diem3;
};
void main()
{ clrscr(); SINHVIEN sv;
  fflush(stdin);
  printf("Nhập họ tên:"); gets(sv.Hoten);
  printf("Nhập Ngày sinh(dd/mm/yyyy):"); gets(sv.NgaySinh);
  printf("Nhập Địa chỉ: "); gets(sv.DiaChi);
  printf("Nhập Điểm 1: "); scanf("%d", &sv.diem1);
  printf("Nhập Điểm 2: "); scanf("%d", &sv.diem2);
  printf("Nhập Điểm 3: "); scanf("%d", &sv.diem3);
  printf("%s %s %s %d %d %d", sv.Hoten, sv.NgaySinh, sv.DiaChi, sv.diem1,
  sv.diem2, sv.diem3);
  getch();
}
```

# Nhập và truy xuất các phần tử cấu trúc

- 1) Khai báo cấu trúc sinh viên bao gồm mssv, hoten, diemkt,diemck

Nhập thông tin cho 1 sinh viên. Xuất thông tin sinh viên, điểm các cột và điểm lần 1 (30%KT + 70%CK)

- 2) Khai báo cấu trúc nhân viên gồm mã nhân viên, họ tên, bậc lương, ngày công.

Nhập thông tin cho 1 nhân viên, xuất thông tin nhân viên bao gồm cả lương và phụ cấp.

- lương = bậc lương \* ngày công
- phụ cấp = 10%lương nếu ngày công  $\geq 25$
- phụ cấp = 0 nếu ngày công  $< 25$

# Mảng cấu trúc

- 1) Khai báo cấu trúc sinh viên bao gồm mssv, hoten, diemkt,diemck

Nhập thông tin cho n sinh viên. Xuất thông tin sinh viên, điểm các cột và điểm lần 1 (30%KT + 70%CK)

- 2) Khai báo cấu trúc nhân viên gồm mã nhân viên, họ tên, bậc lương, ngày công.

Nhập thông tin cho n nhân viên, xuất thông tin nhân viên bao gồm cả lương và phụ cấp.

- lương = bậc lương \* ngày công
- phụ cấp = 10%lương nếu ngày công  $\geq 25$
- phụ cấp = 0 nếu ngày công  $< 25$

# Cấu trúc lồng nhau

- Trong C không có kiểu dữ liệu ngày tháng nên nếu ta muốn sử dụng ta phải định nghĩa một cấu trúc:

```
struct Ngay
```

```
{
```

```
    int d,m,y;
```

```
};
```

- Sử dụng cấu trúc Ngay:

```
struct SINHVIEN
```

```
{
```

```
    char mssv[10];
```

```
    char hoten[30];
```

```
    Ngay ng;
```

```
};
```

# Bài tập

1. Khai báo cấu trúc giáo viên gồm mã gv, họ tên, số tiết dạy.
  - Nhập thông tin n giáo viên. Xuất thông tin giáo viên, phụ cấp. Phụ cấp là 380000 nếu số tiết dạy  $\geq 40$ , phụ cấp = 0 nếu số tiết dạy  $< 40$
2. Khai báo cấu trúc điểm trên tọa độ Oxy. Nhập vào 3 điểm A, B, C. Tính và xuất ra chu vi, diện tích tam giác

# Bài 10

## Con trỏ

# Khái niệm

Khi ta khai báo 1 biến trong chương trình, trình biên dịch dành ra một vùng nhớ với địa chỉ duy nhất để lưu trữ biến đó.

Tương ứng với 1 biến, ta sẽ có các thông tin: tên biến, kiểu biến, địa chỉ của biến và giá trị của biến

Vd: float x=3.4

Tên biến: x

Kiểu biến: float

Địa chỉ của biến: &x

Giá trị của biến: 3.4

→ Con trỏ là 1 biến chứa địa chỉ của 1 biến. Biến có nhiều kiểu khác nhau, do đó cũng có nhiều kiểu con trỏ tương ứng

# Khai báo

<kiểu dữ liệu> \*<tên biến con trỏ>

Vd:

int \*px; // biến con trỏ kiểu int, dùng chứa địa chỉ của biến int

float \*pn; //chứa địa chỉ biến float

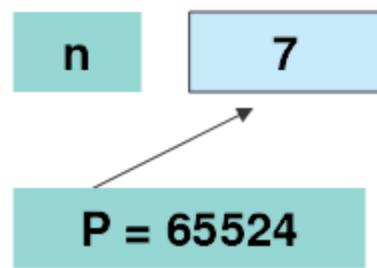
int x;

float n;

px = &x; // px chứa địa chỉ biến x

pn = &n; // pn chứa địa chỉ biến n

# Sử dụng



```
int n=7, m;  
int* p;  
p= &n;  
m=*p
```

```
1.1  
NONAME01.C  
#include <conio.h>  
#include <stdio.h>  
void main()  
{  
    clrscr();  
    int n=7;  
    int *p = &n;  
    int m = *p;  
    printf("Dia chi cua %d la %u",n, p );  
    printf("\nGia tri cua m la %d", m );  
    getch();  
}
```

```
Dia chi cua 7 la 65524  
Gia tri cua m la 7
```

# Thay đổi dữ liệu bằng con trỏ

```
1.1
#include <conio.h>
#include <stdio.h>
void main()
{
    clrscr();
    int n=7;
    int *p = &n;
    *p = 15;
    printf("n = %d", n);
    getch();
}
```

n = 15

# Thay đổi dữ liệu bằng con trỏ

Kết quả của đoạn code sau:

```
int n=7, m=8
```

```
int *p1 = &n;
```

```
int *p2=&m;
```

```
*p1 += 12-m+(*p2);
```

```
*p2 = m + n – 2*(*p1);
```

```
printf("%d", m+n);
```

# Con trả là tham số của 1 hàm

```
#include <conio.h>
#include <stdio.h>
void Swap1(int x, int y)
{
    printf("swap1: dia chi cua x, y: %u, %u\n", &x, &y);
    int t=x; x= y; y=t;
}
void Swap2(int &x, int &y)
{
    printf("swap2: dia chi cua x, y: %u, %u\n", &x, &y);
    int t=x; x= y; y=t;
}
void Swap3(int* px, int* py)
{
    printf("swap3: dia chi cua x, y: %u, %u\n", px, py);
    int t=*px; *px= *py; *py=t;
}
void main()
{
    int a=5, b=7; clrscr();
    printf("a=%d, b=%d\n", a, b);
    printf("dia chi cua a: %u, b: %u\n", &a, &b);
    Swap1(a,b);
    printf("a=%d, b=%d\n", a, b);
    Swap2(a,b);
    printf("a=%d, b=%d\n", a, b);
    Swap3(&a,&b);
    printf("a=%d, b=%d\n", a, b); getch();
}
```

```
a=5, b=7
dia chi cua a: 65524, b: 65522
swap1: dia chi cua x, y: 65518, 65520
a=5, b=7
swap2: dia chi cua x, y: 65524, 65522
a=7, b=5
swap3: dia chi cua x, y: 65524, 65522
a=5, b=7
```

# Con trỏ và mảng

Vd: ta khai báo

float a[10];

→ Tên mảng a là 1 con trỏ hằng

→ &a[0] lấy địa chỉ của phần tử đầu tiên của mảng nên a tương đương với &a[0]

→ (a+i) tương đương với &a[i]

→ \*(a+i) tương đương với a[i]

# Con trỏ cấu trúc

Ta có thể truy xuất đến cấu trúc bằng cách sử dụng con trỏ chỉ đến cấu trúc đó

Truy xuất đến phần tử của con trỏ cấu trúc bằng toán

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
struct SINHVIEN
{ char Hoten[15];
  int diem1, diem2, diem3;
};
double TrungBinh(SINHVIEN *sv)
{      return (sv->diem1+sv->diem2+sv->diem3)/3.;
}
void Xuat(SINHVIEN *sv)
{      printf("%s %d %d %d %.1lf", sv->Hoten, sv->diem1, sv->diem2, sv->diem3,
TrungBinh(sv));
}
void main()
{      clrscr();  SINHVIEN *sv = new SINHVIEN;
      strcpy(sv->Hoten, "Nguyen Van A"); sv->diem1=5; sv->diem2=7; sv->diem3=9;
      Xuat(sv);  getch();
}
```

Nguyen Van A 5 7 9 7.0

# Bài 11

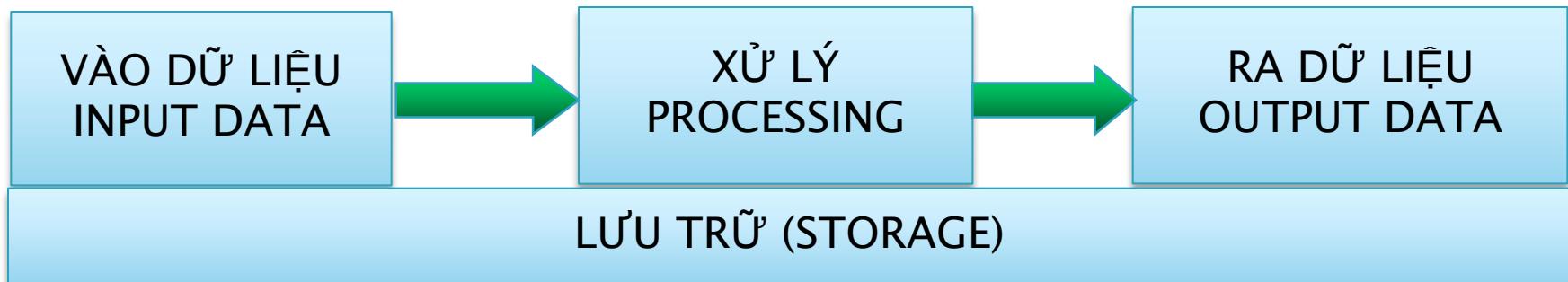
## Tập tin

# Đặt vấn đề

Chu trình xử lý:

Dữ liệu vào → Xử lý → Ra dữ liệu

Ngoài ra còn có nhu cầu lưu trữ dữ liệu.



# Khái niệm

Có 02 loại tập tin:

- File nhị phân (binary file): Dữ liệu ghi trên tập tin theo các byte nhị phân như trong bộ nhớ
- File văn bản (text file): Là file có cấu trúc như các file văn bản chuẩn của Dos, hay file văn bản trong chương trình soạn thảo notepad của Windows.

# Khai báo biến con trả file

FILE \*tênbiếncontrở;  
(tùy khóa FILE phải viết hoa)

Ví dụ:

```
FILE *F1;  
FILE *taptin1, *taptin2;
```

# Một số hàm về FILE

1/ Hàm mở tập tin: fopen

FILE \*fopen(const char \*filename, const char \*type)

filename: Tên file: Có thể chỉ rõ đường dẫn tuyệt đối của file hoặc chỉ dùng tên file khi file cùng thư mục chứa file chương trình (\*.cpp)

type: kiểu mở file

Kiểu	Ý nghĩa
“r”, “rt”	Mở file text để đọc, nếu không có sẽ thông báo có lỗi
“w”, “wt”	Mở file text để ghi, nếu file đã có nó sẽ bị xóa
“a”, “at”	Mở file text để ghi bổ sung, nếu file không có sẽ tạo file mới
“rb”	Mở file nhị phân để đọc, nếu file không có sẽ báo lỗi
“wb”	Mở file nhị phân để ghi, nếu file đã có nó sẽ bị xóa
“ab”	Mở file nhị phân để ghi bổ sung, nếu file không có sẽ tạo file mới

# Một số hàm về FILE

Ví dụ mở file:

```
FILE* f; //khai báo một con trỏ kiểu FILE
f=fopen("tên file", "kiểu mở file");
if (f==NULL) //kiểm tra xem có mở được file hay chưa
{
    printf("Khong mo duoc file");
    exit(1);
}
```

# Một số hàm về FILE

2/ Hàm đóng tập tin: fclose

fclose(FILE \*fp)

fp: Tên con trỏ tương ứng với file cần đóng

fcloseall(): đóng tất cả các file đang mở

Ví dụ:

```
int n;
f=fopen("test.inp","wt");
if (f==null)
{
    printf("khong mo duoc file");
    exit(1);
}
fprintf(f,"%d",n);
fclose(f);
```

# Một số hàm về FILE

## 3/ Đọc dữ liệu:

Dùng hàm fscanf để đọc các thành phần dữ liệu  
fscanf(con trỏ file, “format kiểu dữ liệu cần đọc”, địa chỉ biến lưu);

fgets(địa chỉ vùng nhớ chứa chuỗi, chiều dài cực đại của chuỗi, con trỏ file): đọc một chuỗi từ file

getw(con trỏ file): đọc số nguyên từ file

## 4/ Ghi dữ liệu:

Dùng hàm fprintf để ghi dữ liệu vào file

fprintf(con trỏ file, “format kiểu dữ liệu ghi”, biến lưu);

fputs(chuỗi cần ghi, con trỏ file): ghi một chuỗi vào file

putw(số nguyên cần ghi, con trỏ file): ghi một số nguyên vào file

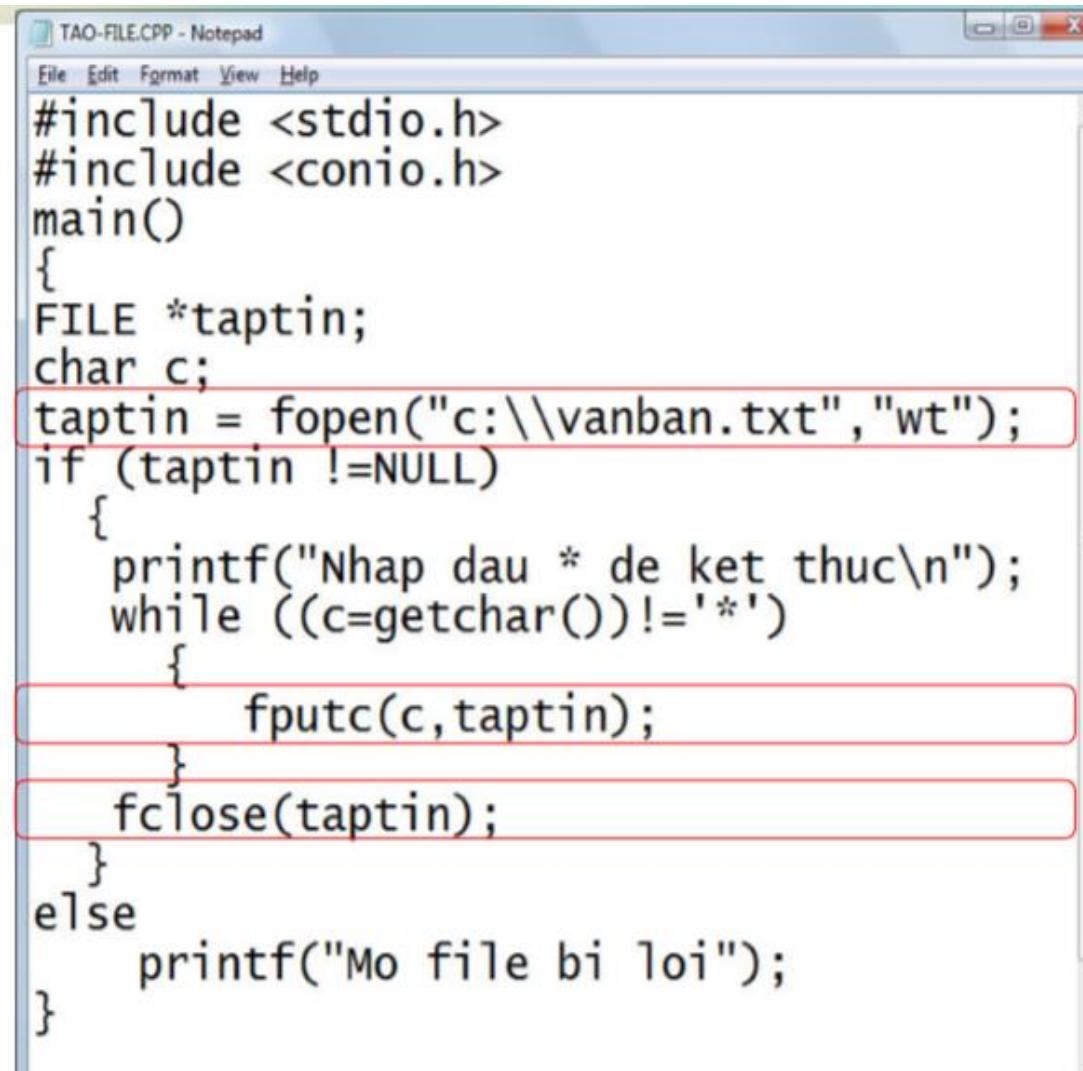
# Một số hàm về FILE

## 6/ Một số hàm khác:

- Kiểm tra cuối file: feof(con trỏ file) → giá trị khác 0: ở cuối file
- Kiểm tra lỗi thao tác trên file: ferror(con trỏ file)
- Làm sạch vùng đệm của 1 file: fflush(con trỏ file)
- Làm sạch vùng đệm của tất cả các file đang mở:  
fflush();

# Ví dụ

Viết chương trình tạo một file có tên vanban.txt trong đĩa c, cho phép nhập nội dung vào file, khi nào muốn kết thúc nhập ký tự \*.

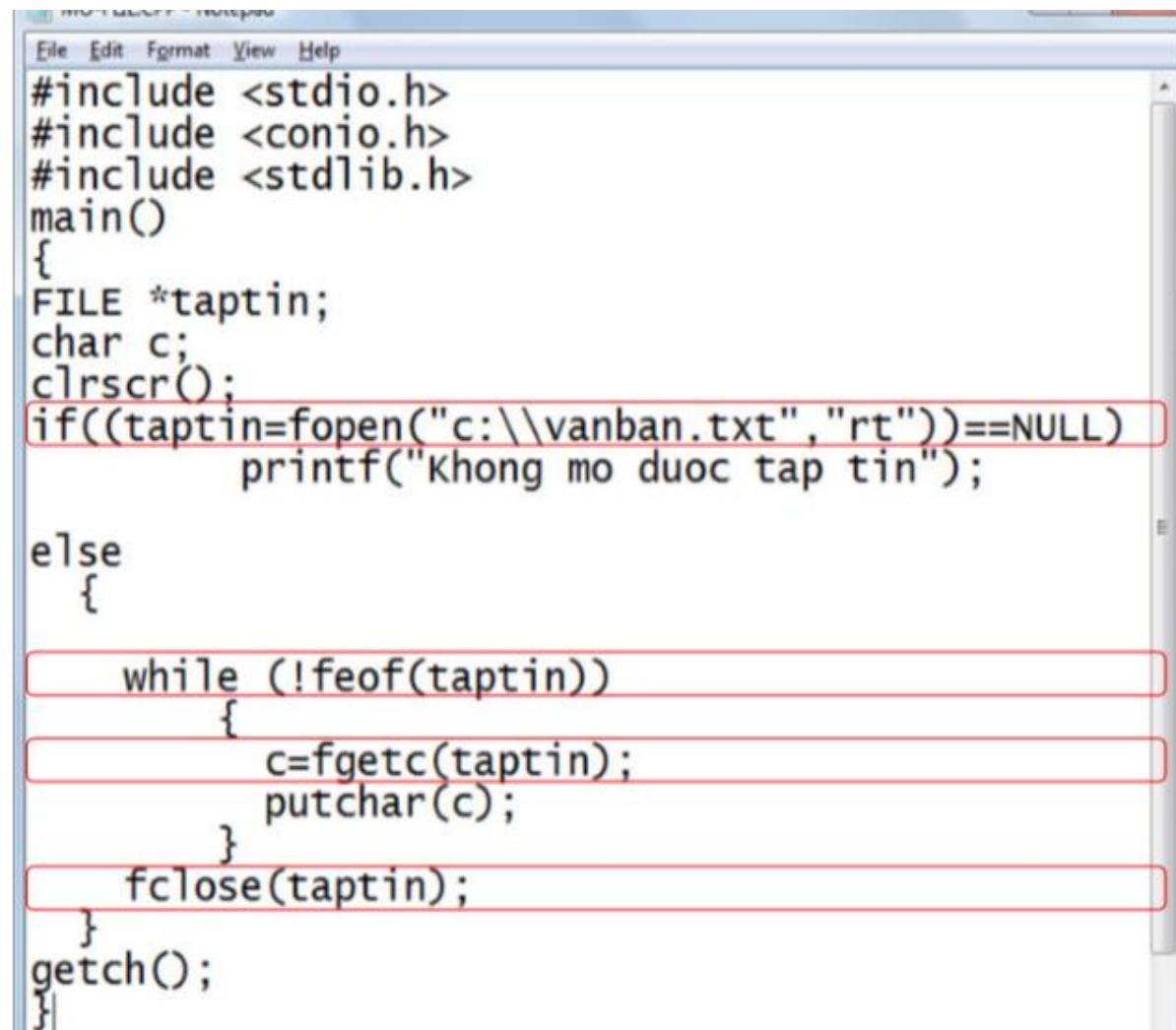


The screenshot shows a Notepad window titled "TAO-FILE.CPP - Notepad". The code is a C++ program that creates a file named "vanban.txt" in the "c:\\" directory. The program uses the `fopen` function to open the file in write mode ("wt"). It then enters a loop where it reads characters from the standard input using `getchar` and writes them to the file using `fputc`. The loop continues until the user enters the character '\*' to indicate they want to stop. Finally, the program closes the file using `fclose`. If the file cannot be opened, it prints an error message.

```
#include <stdio.h>
#include <conio.h>
main()
{
FILE *taptin;
char c;
taptin = fopen("c:\\vanban.txt","wt");
if (taptin !=NULL)
{
    printf("Nhập dấu * để kết thúc\n");
    while ((c=getchar())!='*')
    {
        fputc(c,taptin);
    }
    fclose(taptin);
}
else
    printf("Mở file bị lỗi");
}
```

# Ví dụ

Viết  
chương trình  
mở một file có  
tên vanban.txt  
trong đĩa c



```
File Edit Format View Help
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
main()
{
FILE *taptin;
char c;
clrscr();
if((taptin=fopen("c:\\vanban.txt", "rt"))==NULL)
    printf("Khong mo duoc tap tin");
else
{
    while (!feof(taptin))
    {
        c=fgetc(taptin);
        putchar(c);
    }
    fclose(taptin);
}
getch();
}
```

# Bài tập

1/ Viết chương trình ghi vào file tên là syll.txt với nội dung:

Ho ten:

MSSV:

Dia chi:

2/ Viết chương trình đọc file syll.txt ra màn hình

3/ Viết chương trình tạo một file nhị phân tên songuyen.dat chứa các số nguyên từ 100 đến 200 lưu trong ổ đĩa C

4/ Viết chương trình mở file nhị phân tên songuyen.dat lưu trong ổ đĩa C, xuất nội dung file lên màn hình.

5/ Viết chương trình nhập một danh sách sinh viên gồm 2 thông tin họ tên và tuổi. Lưu danh sách sinh viên vào file nhị phân data.dat sau đó đọc nội dung file ra màn hình

# Bài tập

6/ Viết chương trình tính lương cho cán bộ công nhân viên.

Thông tin bảng lương:

Mã NV	Họ tên	LCB	Ngày công	Tiền lương	BHXH	BHYT	Thực lãnh
01	Tran A	100	24	?	?	?	?
02	Tran B	100	20				
03	Nguyen C	200	28				
...							

Cách tính:

- Tiền lương =  $(LCB/22) * \text{ngày công}$
- BHXH = Tiền lương \* 5%
- BHYT = Tiền lương \* 1%
- Thực lãnh = Tiền lương – BHXH - BHYT

# Bài tập

Yêu cầu:

Viết chương trình thông qua việc xây dựng các hàm sau:

- Hàm nhập danh sách lương nhân viên, lưu thông tin này vào file LUONG.DAT
- Hàm đọc nội dung file LUONG.DAT và in ra màn hình theo bảng đã cho
- Viết hàm tính tổng ngày công mà toàn bộ nhân viên đã làm (đọc từ file LUONG.DAT)
- Viết hàm tính lương trung bình phải trả cho 1 nhân viên (đọc từ file LUONG.DAT)
- Viết hàm đọc nội dung file LUONG.DAT và in ra màn hình như bảng đã cho nhưng sắp xếp theo tiền thực lãnh giảm dần.

# Ôn tập

