



# Khoa Công nghệ Thông tin (FIT HUFI)

---

## NHẬP MÔN LẬP TRÌNH

### I. GIỚI THIỆU MÔN HỌC

- ❖ Tên học phần: Nhập môn lập trình
- ❖ Số tín chỉ - số tiết: 3 (3,0,6) - 45 tiết
- ❖ Loại học phần: bắt buộc
- ❖ Đối tượng: SV ĐH chính quy ngành CNTT/ATTT

*BM Công nghệ phần mềm – FIT HUFI*

7/15/2021

# Nội dung môn Nhập môn lập trình

- Chương 1. Tổng quan về lập trình
- Chương 2. Các thành phần cơ bản của NNLT C/C++
- Chương 3. Các cấu trúc điều khiển chương trình
- Chương 4. Hàm
- Chương 5. Dữ liệu mảng, chuỗi
- Chương 6. Dữ liệu cấu trúc

## II. MÔ TẢ HỌC PHẦN

Học phần này bao gồm các nội dung sau:

- ❖ Các khái niệm cơ bản về lập trình.
- ❖ Các thành phần của ngôn ngữ lập trình C/C++.
- ❖ Các cấu trúc điều khiển chương trình.
- ❖ Cách xây dựng và sử dụng hàm.
- ❖ Các kiến thức về mảng 1 chiều, mảng 2 chiều và những thao tác cơ bản trên kiểu dữ liệu mảng.
- ❖ Các kiến thức cơ bản về dữ liệu có cấu trúc.

### **III. NHIỆM VỤ CỦA SINH VIÊN**

- ❖ Tham dự giờ học lý thuyết trên lớp.
- ❖ Làm các bài tập, tiểu luận theo yêu cầu của giảng viên.
- ❖ Dự kiểm tra tại lớp và thi cuối học phần.
- ❖ **Đánh giá học phần:**
  - Đánh giá quá trình: 30%
    - Điểm thái độ học tập: 10%
    - Điểm tiểu luận (bài tập, kiểm tra tại lớp): 20%
  - Điểm thi kết thúc học phần: 70% (tự luận)

## V. TÀI LIỆU THAM KHẢO

- ❖ Bộ slide bài giảng NMLT – FIT HUFI
- ❖ Giáo trình Nhập môn lập trình – FIT HUFI
- ❖ Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết, Nhập môn lập trình, NXB Khoa học và Kỹ thuật, 2011 (tìm ở thư viện Trường)
- ❖ Programming Language C – B.W. Kernighan & D.M Ritchie

## **VI. THÔNG TIN GIẢNG VIÊN GIẢNG DẠY**

- ❖ Giảng viên giảng dạy: ThS Nguyễn Thị Bích Ngân
- ❖ Email: [nganntb@hufi.edu.vn](mailto:nganntb@hufi.edu.vn)
- ❖ Mã lớp trên kênh Google Classroom:



## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 1. TỔNG QUAN

*BM Công nghệ phần mềm –  
FIT HUFI*

7/15/2021

# Mục tiêu

- Tổng quan về lập trình
- Làm quen với NNLT C.
- Biết cấu trúc 1 chương trình viết bằng NNLT C.
- Biết các thư viện chuẩn
- Hiểu và vận dụng các kiểu dữ liệu cơ bản, hằng, biến của NNLT C.

# Nội dung

- 1. Tổng quan về lập trình**
- 2. Giới thiệu về ngôn ngữ lập trình C**
- 3. Đặc điểm của ngôn ngữ lập trình C**
- 4. Cấu trúc chương trình C**
- 5. Thư viện hàm chuẩn C**
- 6. Ưu và nhược**
- 7. Bài tập**

# 1. Tổng quan về lập trình

- ❑ Một chương trình (program) là một dãy các chỉ thị (instruction) điều khiển sự hoạt động của máy tính nhằm giải quyết một công việc nào đó.
- ❑ Người viết chương trình (còn gọi là lập trình viên) là những người tạo lập ra các chương trình điều khiển máy tính.

# 1. Tổng quan về lập trình

- Ngôn ngữ lập trình là ngôn ngữ được lập trình viên sử dụng để viết chương trình cho máy tính.
- Khi một chương trình được viết bằng một NNLT nào đó thì các chỉ thị, câu lệnh trong chương trình phải tuân theo các quy tắc, các luật do NNLT đó quy định.
- Người lập trình thường viết chương trình bằng các NNLT cấp cao vì tính dễ dùng, có thể diễn đạt được các ý tưởng trừu tượng và có tính tương thích cao.
- Một số NNLT thông dụng: C/C++, C#, Java, Python, ...

## 2. Giới thiệu về ngôn ngữ lập trình C

- ❑ C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.
- ❑ Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và rất “mềm dẻo”, có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn.
- ❑ Hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp.
- ❑ Cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trừu tượng mới.

### 3. Đặc điểm của ngôn ngữ lập trình C

- Tính cô đọng (compact)
- Tính cấu trúc (structured)
- Biên dịch (compile)
- Tính linh động (flexible)
- Tính tương thích (compatible)

# 4. Cấu trúc chương trình C

Một chương trình C bao gồm các phần như: Các chỉ thị tiền xử lý, định nghĩa kiểu dữ liệu mới, khai báo biến ngoài, các hàm tự tạo, hàm main.

## Cấu trúc:

1. Các chỉ thị tiền xử lý (*khai báo thư viện, khai báo hằng số*)
2. Định nghĩa kiểu dữ liệu (*khai báo cấu trúc*)
3. Khai báo các biến ngoài (*khai báo biến toàn cục*)
4. Khai báo các prototype của hàm tự tạo (*khai báo định nghĩa hàm*)
5. Hàm main
6. Định nghĩa các hàm tự tạo (*thân của các hàm đã khai báo ở phần 4*)

# 4. Cấu trúc chương trình C

## Các chỉ thị tiền xử lý:

#include <Tên tập tin thư viện>

#define ....

Chỉ thị #define được sử dụng trong việc định nghĩa các ký hiệu

### Ví dụ:

#include <stdio.h>

#define MAX 100

#define SIZE 25

# 4. Cấu trúc chương trình C

**Định nghĩa kiểu dữ liệu** (không bắt buộc): dùng để đặt tên lại cho một kiểu dữ liệu nào đó để gọi nhở hay đặt 1 kiểu dữ liệu cho riêng mình dựa trên các kiểu dữ liệu đã có.

**Cú pháp:**      `typedef <Tên kiểu cũ> <Tên kiểu mới>`

**Ví dụ:**

`typedef int SoNguyen; // Kiểu SoNguyen là kiểu int`

# 4. Cấu trúc chương trình C

## Khai báo các biến ngoài:

Được sử dụng để khai báo các biến toàn cục khi cần thiết.  
Phần này không bắt buộc khai báo trong chương trình.

Cú pháp: <Kiểu dữ liệu> <Tên biến>;

Ví dụ: int a; //khai báo biến số nguyên a

## Khai báo các Prototype của hàm tự tạo:

Cú pháp: <Kiểu trả về của hàm> <Tên hàm>([<các đối số>]);

Ví dụ: boolean isPrime(int a); // prototype của hàm isPrime

# 4. Cấu trúc chương trình C

## **Hàm main:**

Khi chương trình thực thi thì hàm Main được gọi trước tiên. Đây là phần bắt buộc khai báo trong chương trình.

## **Cú pháp:**

```
<Kiểu dữ liệu trả về> main()
{
    //các khai báo cục bộ trong hàm ]
    //Các câu lệnh dùng để định nghĩa hàm main]
    [return <kết quả trả về>; ]
}
```

## **Ví dụ:** void main()

```
{
    printf("Hello");
    getch();
}
```

# 4. Cấu trúc chương trình C

## **Định nghĩa các hàm tự tạo:**

Đây là phần không bắt buộc trong chương trình

**Cú pháp:**      <Kiểu dữ liệu trả về> <Tên hàm>([<Các đối số>])

{

[//các khai báo cục bộ trong hàm ]

[//Các câu lệnh dùng để định nghĩa hàm]

[return <kết quả trả về>; ]

}

**Ví dụ:** Viết hàm trả về giá trị tổng hai số nguyên a và b.

*int Tinh tong(int a, int b)*

{

*int c = a + b;*

*return c;*

}

## 4. Thư viện hàm chuẩn C

Tất cả trình biên dịch C đều chứa thư viện hàm chuẩn C.

Một số thư viện chuẩn trong C:

- **stdio.h:** Tập tin chứa các hàm vào/ra chuẩn.
- **conio.h:** Tập tin định nghĩa các hàm vào ra trong chế độ DOS (DOS console).
- **math.h:** Tập tin định nghĩa các hàm tính toán.
- **alloc.h:** Tập tin định nghĩa các hàm liên quan đến việc quản lý bộ nhớ.
- **io.h:** Tập tin định nghĩa các hàm vào ra cấp thấp.
- **graphics.h:** Tập tin định nghĩa các hàm liên quan đến đồ họa.

# Bài tập

1. Viết chương trình xuất ra câu thông báo: “Chao ban den voi ngon ngu C”.
2. Viết chương trình xuất ra đoạn thông báo:

“Chao ban!  
Day la chuong trinh C dau tien.  
Vui long nhan phim Enter de ket thuc.”
3. Viết chương trình nhập vào 1 số nguyên, xuất ra màn hình số nguyên vừa nhập.
4. Viết chương trình nhập vào 2 số nguyên, tính và xuất kết quả tổng, hiệu, tích và thương của 2 số nguyên vừa nhập.
5. Viết chương trình xuất ra các số từ 1 đến 10.





## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 2. CÁC THÀNH PHẦN CƠ BẢN CỦA NNLT C/C++

*BM Công nghệ phần mềm –  
FIT HUST*

7/15/2021

# Mục tiêu

- ❑ Hiểu và vận dụng các kiểu dữ liệu cơ bản, hằng, biến của NNLT C.
- ❑ Hiểu và vận dụng các phép toán cơ bản của NNLT C.
- ❑ Biết viết các dạng biểu thức toán học, biểu thức quan hệ, biểu thức logic dưới dạng NNLT C.
- ❑ Hiểu ý nghĩa và vận dụng đúng cách các toán tử tăng, giảm, biểu thức điều kiện, độ ưu tiên các toán tử.

# Nội dung

- 1. Danh hiệu**
- 2. Biến**
- 3. Các kiểu dữ liệu**
- 4. Hằng số**
- 5. Biểu thức**
- 6. Các phép toán**
- 7. Bài tập**

# 1. Danh hiệu

Có 4 loại danh hiệu khác nhau: ký hiệu, từ khóa, tên và chú thích.

**Ký hiệu:** là tập kí tự hợp lệ trong ngôn ngữ C bao gồm:

- 52 kí tự chữ: A, B, ...., Z và a, b, ..., z.
- 10 kí tự số, các kí hiệu toán học: +, -, \*, /, =, <, >, (, )
- Các kí tự đặc biệt như: ., ; : [ ] { } ? ! \ & \ | # \$ " ' @ ^...
- Kí tự gạch nối \_
- Dấu cách (khoảng trắng) dùng để phân cách giữa các từ

# 1. Định danh

**Tùy khóa:** là các từ sử dụng để dành riêng trong ngôn ngữ lập trình C

Các từ khóa không được sử dụng làm các biến, hằng.

Không được định nghĩa lại các từ khoá.

*Bảng liệt kê các từ khoá:*

auto	break	case	char	continue	default	do	double
else	extern	float	for	goto	if	int	long
short	sizeof	static	struct	switch	typedef	union	register
void		while					return
_cs	_ds	_es	_ss	_AH	_AL	_AX	_BH
_BX	_CH	_CL	_CX	_DH	_DL	_DX	_BP
_SI	_SP						_DI

# 1. Danh hiệu

**Tên:** là một dãy các ký tự liền nhau bắt đầu bằng chữ cái hoặc ký tự gạch dưới.

## *Quy tắc đặt tên:*

- ❑ Tên là một dãy các ký tự liền nhau bắt đầu bằng chữ cái hoặc ký tự gạch dưới theo sau là chữ cái, dấu gạch dưới, chữ số.
- ❑ Một tên không được chứa các kí tự đặc biệt như dấu cách, dấu chấm câu,...
- ❑ Theo tiêu chuẩn C các kí tự chữ thường và hoa thì xem như khác nhau.

*Ví dụ:* biến ADD, add và Add là khác nhau.

- ❑ Tên một biến nên có ý nghĩa, gợi tả và mô tả rõ kiểu dữ liệu của nó.
- Ví dụ:* nếu tìm tổng của 2 số thì tên biến lưu trữ tổng nên đặt là sum (tổng).

# 1. Danh hiệu

**Chú thích:** là những dòng mô tả ý nghĩa câu lệnh đang dùng, giải thích ý nghĩa của một hàm nào đó.

Phần nội dung ghi chú này khi biên dịch sẽ được bỏ qua. Trong ngôn ngữ lập trình C, nội dung chú thích phải được viết trong cặp dấu /\* và \*/ hoặc sau dấu “//” nếu chú thích nằm trên một dòng.

## Ví dụ:

```
#include <stdio.h>
#include <conio.h>
int main ()
{
    int a, b; //Khai báo 2 biến số nguyên
    /* Nhập vào 2 số nguyên a và b */
    printf("\n Nhap vao hai so nguyen a, b: \t");
    scanf("%d %d", &a, &b);
    return (a+b);
}
```

## 2. Biến

- **Biến** là một tên đại diện cho một giá trị dữ liệu cần lưu trữ.
- Khi biến được tạo sẽ xuất hiện một vùng nhớ để lưu trữ giá trị của biến.
- Kiểu dữ liệu quyết định tổng số bộ nhớ được chỉ định. Những tên được gán cho biến giúp chúng ta sử dụng lại dữ liệu khi cần đến.

### Cú pháp: khai báo biến

<Kiểu dữ liệu> <Tên biến> [= <Giá trị>];

### Ví dụ:

*void main()*

```
{   int sum;           //khai báo biến số nguyên sum  
    sum = 10 + 25 + 50;  
    printf("\n 10 + 25 + 50 = %d ", sum);  
}
```

### 3. Các kiểu dữ liệu

Các loại dữ liệu khác nhau được lưu trữ trong biến là :

#### ➤ Số (Numbers)

- Các số nguyên.

*Ví dụ:* 10 hay 178993455.

- Các số thực.

*Ví dụ:* 15.22 hay 15463452.25.

- Các số dương.

- Các số âm.

#### ➤ Tên

*Ví dụ:* John.

#### ➤ Giá trị luận lý

*Ví dụ:* đúng hay sai (true – false)

### 3. Các kiểu dữ liệu

C có 5 kiểu dữ liệu cơ bản. Tất cả những kiểu dữ liệu khác dựa vào một trong số những kiểu này. 5 kiểu dữ liệu đó là:

- **int**: là một số nguyên, về cơ bản nó biểu thị kích cỡ tự nhiên của các số nguyên (**integers**).
- **float** và **double**: dùng cho các số có dấu chấm động. Kiểu **float** (số thực) chiếm 4 byte và có thể biểu diễn 6 số phần thập phân, trong khi **double** chiếm 8 byte và có thể biểu diễn 10 số phần thập phân.
- **char**: chiếm 1 byte và có khả năng lưu một ký tự đơn.
- **void**: được dùng điển hình để khai báo một hàm không trả về giá trị.

### 3. Các kiểu dữ liệu

#### a. Kiểu char:

Kiểu dữ liệu **char** được dùng để lưu trữ một ký tự đơn.

Một kiểu dữ liệu **char** có thể lưu một ký tự đơn được đặt trong hai dấu nháy đơn (''). Khi truy xuất giá trị của một biến kiểu **char** ta dùng kí hiệu đại diện %c.

**Cú pháp:**      **char <tên biến> [= ‘kí tự’];**

**Ví dụ:**

```
#include <stdio.h>
void main()
{
    char c = 'a';
    printf("\nXuat du lieu bien c = %c", c);
    printf("\nNhap vao ki tu c = "); scanf("%c", &c);
    printf("\nXuat ky tu c vua nhap: %c");
```

### 3. Các kiểu dữ liệu

#### b. Kiểu int:

Kiểu dữ liệu **int** được dùng để lưu trữ một số nguyên có miền giá trị từ  $-2^{15}$  đến  $2^{15} - 1$ .

Khi truy xuất giá trị của một biến kiểu **int** ta dùng kí hiệu đại diện `%d`.

**Cú pháp:** `int <Tên biến>;`

**Ví dụ:**

```
#include <stdio.h>
void main()
{
    int a = 12345;
    printf("\nXuat gia tri bien a = %d", a);
    printf("\nNhap vao so nguyen a = ");    scanf("%d", &a);
    printf("\nXuat gia tri a vua nhap: %d", a);
}
```

### 3. Các kiểu dữ liệu

#### c. Kiểu float và double:

- ❑ Một số thực kiểu ***float*** có giá trị trong khoảng 1.2E-38 ÷ 3.4E+38 với độ chính xác khoảng 6 số phần thập phân.
- ❑ Một số thực kiểu ***double*** có giá trị trong khoảng 2.2E – 308 ÷ 1.8E + 308 với độ chính xác khoảng 10 số phần thập phân.
- ❑ Khi truy xuất giá trị của một biến kiểu số thực ta dùng kí hiệu đại diện ***%f***.
- ❑ Nếu cách hiển thị một số thực là ***%.nf*** khi đó giá trị số hiển thị **n** kí tự cho phần thập phân. Ví dụ như ***%.5f*** thì 5 kí tự cho phần thập phân của số hiển thị.

**Cú pháp:**      ***float <Tên biến>;***  
                        ***double <Tên biến>;***

### 3. Các kiểu dữ liệu

#### c. Kiểu float và double:

Ví dụ:

```
#include<stdio.h>
void main()
{
    float a;
    printf("nhap gia tri cua a: ");
    scanf("%f", &a);
    printf("Ket qua: %.1f\n",a);
    printf("Ket qua: %.3f\n",a);
    printf("Ket qua: %.6f\n",a);
}
```

### 3. Các kiểu dữ liệu

#### d. Kiểu void:

C có một kiểu dữ liệu đặc biệt gọi là **void**. Trong C, các hàm số thường trả về dữ liệu thuộc một kiểu nào đó. Tuy nhiên, khi một hàm không trả về kết quả thì kiểu trả về của hàm sẽ là kiểu **void**.

#### Ví dụ:

```
void Giai_PT(int a, int b);
```

### 3. Các kiểu dữ liệu

#### e. Các kiểu dữ liệu bổ sung

Các kiểu dữ liệu bổ sung bao gồm **unsigned**, **signed**, **short**, **long**.

Các kiểu dữ liệu bổ sung kết hợp với các dữ liệu cơ sở làm thay đổi miền giá trị của chúng.

*Thứ tự kết hợp là: kiểu dữ liệu bổ sung → kiểu dữ liệu cơ sở.*

*Các kiểu có dấu (**signed**) và không dấu (**unsigned**)*

Kiểu **signed** chỉ sử dụng với kiểu dữ liệu **char** vì **char** là kiểu mặc định không dấu.

Kiểu **unsigned** chỉ rõ rằng một biến chỉ nhận giá trị dương.

**Ví dụ:** `unsigned int varNum = 50000;`

**Ý nghĩa:** biến `varNum` vẫn được cấp phát vùng nhớ 2 byte. Tuy nhiên, giá trị mà `varNum` nhận được nằm trong khoảng từ 0 đến 65535, thay vì từ -32768 tới 32767 như kiểu `int` quy định.

### 3. Các kiểu dữ liệu

*Các kiểu có dấu (signed) và không dấu (unsigned)*

Kiểu **signed** chỉ sử dụng với kiểu dữ liệu **char** vì **char** là kiểu mặt định không dấu.

Kiểu **unsigned** chỉ rõ rằng một biến chỉ nhận giá trị dương.

Ví dụ:

***unsigned int varNum = 50000;***

Ý nghĩa: biến *varNum* vẫn được cấp phát vùng nhớ 2 byte. Tuy nhiên, giá trị mà *varNum* nhận được nằm trong khoảng từ 0 đến 65535, thay vì từ -32768 tới 32767 như kiểu **int** quy định.

### 3. Các kiểu dữ liệu

#### *Các long và short:*

Kiểu **short** được kết hợp với kiểu dữ liệu cơ sở khi chiều dài yêu cầu ngắn hơn chiều dài bình thường của kiểu dữ liệu cơ sở.

**Ví dụ:** kiểu **int** chiếm 2 byte vùng nhớ trong khi đó kiểu **short int** chỉ chiếm 1 byte.

```
int a; // giá trị a nằm trong khoảng -32768 ... 32767
```

```
short int b; // giá trị b nằm trong khoảng -128 ... 127
```

Kiểu **long** được dùng khi chiều dài yêu cầu dài hơn chiều dài bình thường của kiểu dữ liệu cơ sở.

**Ví dụ:** kiểu **int** chiếm 2 byte trong khi kiểu **long int** chiếm 4 byte.  
kiểu **double** chiếm 8 byte thì **long double** chiếm 16 byte.

# 4. Hằng số

Hằng số là đại lượng cố định trong chương trình.

Muốn sử dụng hằng ta phải khai báo trước với từ khóa **const** hoặc **define**.

## Cú pháp:

**const <kiểu dữ liệu><Tên hằng> = <Giá trị hằng>;**

**Hoặc:**

**#define <Tên hằng> <Giá trị hằng>**

## Ví dụ:

```
const int a= 32767 ;
```

```
const float b = 3.14;
```

```
const int a= 3, b = 4, C = 5;
```

```
#define a 32767
```

```
#define b 3.14
```

## 5. Biểu thức

Biểu thức là một công thức tính toán, bao gồm 2 phần: toán tử và toán hạng. Toán tử là các phép toán, toán hạng có thể là hằng, biến hay hàm nào đó.

Có 2 loại toán tử: một ngôi (lấy đối số, tăng giảm một đơn vị) và 2 ngôi (cộng, trừ, nhân, chia, lấy dư và lũy thừa).

**Ví dụ:**  $2 + 3*(++a) + \text{sum}(a, b) - 2^3$

Có 3 loại biểu thức:

**Biểu thức số học:** trả về giá trị số

**Biểu thức logic:** trả về kết quả đúng, sai

**Biểu thức quan hệ:** liên quan đến các phép toán so sánh ( $>$ ,  $<$ ,  $!=$ ,  $==$ , ...).

## 5. Biểu thức

Ví dụ: toán tử 1 ngôi

```
#include <stdio.h>

void main()
{
    int a = 5, b = 10;
    int c = -a;
    printf("\nGia tri c = %d", c); //kết quả c = -5
    printf("\nXuat a++ = %d", a++); //kết quả 5
    printf("\nXuat ++b = %d", ++b); //kết quả 11
}
```

Ghi chú: biểu thức  $x = x + 1$  tương đương với  $++x$  hay  $x++$ ;

## 5. Biểu thức

Ví dụ: toán tử 2 ngôi

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

```
void main()
```

```
{
```

```
    int a = 5, b = 10;
```

```
    int c = (a*b/2)%4;
```

```
    int d = a^3;
```

```
    printf("nGia tri c = %d", c); //kết quả c = 1
```

```
    printf("nGia tri d = %d", d); //kết quả 125 = 5*5*5
```

```
}
```

# 5. Biểu thức

## ❑ Biểu thức phẩy:

Mỗi câu lệnh trong ngôn ngữ lập trình C được kết thúc bằng dấu chấm phẩy, tuy nhiên trong một biểu thức của ngôn ngữ lập trình C có thể có nhiều câu lệnh được cách nhau bởi dấu phẩy.

### Ví dụ:

$x = a * b, q = x + y, k = q / z;$

int n, m;

char ho[50], ten[50];

## 5. Biểu thức

❑ Biểu thức điều kiện:

Cú pháp:

<Tên biến> = <Biểu thức điều kiện> ? <Biểu thức 1> : <Biểu thức 2>

Ví dụ:

$m = a > b ? a : b$  /\*  $m = \max(a,b)$  \*/

# 6. Các phép toán

**a. Toán tử số học:** cộng, trừ, nhân, chia, lấy dư và lũy thừa lần lượt tương ứng với các ký hiệu +, -, \*, /, %, ^.

## Ghi chú:

- Phép toán % không dùng cho kiểu dữ liệu **float** hay **double**.
- Phép chia(/) thực hiện theo kiểu của các toán hạng dù là phép chia số nguyên hay số thực.

## Ví dụ:

```
#include <stdio.h>
void main()
{
    int x = 10, y = 3;
    float z1, z2;
    z1 = x/y; z2 = (float)x/y;
    printf("\n%d chia %d dư: %d", x, y, x%y); //kết quả du 1
    printf("\n z1 = %.2f", z1); // kết quả z1 = 3.00
    printf("\n z2 = %.3f", z2); // kết quả z2 = 3.333
```

# 6. Các phép toán

b. **Toán tử quan hệ:** bao gồm

$==$ ,  $!=$ ,  $>=$ ,  $>$ ,  $<=$ ,  $<$

c. **Toán tử logic:** bao gồm  $\&\&$  (phép AND),  $\|$  (phép OR),  $!$  (phép NOT).

$A \&\& B$ : đúng  $\Leftrightarrow A$ : đúng và  $B$ : đúng

$A \| B$ : sai  $\Leftrightarrow A$ : sai và  $B$ : sai

Nếu  $A$ : đúng thì  $!A$ : sai

## 6. Các phép toán

d. **Toán tử trên bit:** bao gồm phép AND (&), OR(|), XOR(^), phép dịch trái (<<), phép dịch phải (>>) và phép đảo bit (~).

*Cách thực hiện các phép toán trên bit:*

$$1 \& 1 = 1 \quad 1 | 1 = 1 \quad 1 ^ 1 = 0$$

$$1 \& 0 = 0 \quad 1 | 0 = 1 \quad 1 ^ 0 = 1$$

$$0 \& 1 = 0 \quad 0 | 1 = 1 \quad 0 ^ 1 = 1$$

$$0 \& 0 = 0 \quad 0 | 0 = 0 \quad 0 ^ 0 = 0$$

- $x << M$  nghĩa là dịch sang trái số nguyên  $x$  đi  $M$  bit, tương đương với  $x * 2^M$
- $x >> M$  nghĩa là dịch sang phải số nguyên  $x$  đi  $M$  bit, tương đương với phép chia  $x / 2^M$  (chia lấy phần nguyên).

**Ví dụ:** Ta có thể thay phép tính  $x * 80$  bằng cách:

$$x << 6 + x << 4 \text{ vì } 80 = 2^6 + 2^4$$

## 6. Các phép toán

### e. Toán tử tăng giảm 1:

- $i = i + 1$  có thể được viết thành:  $i++$ (tăng sau) hoặc  $++i$ (tăng trước).
- $i = i - 1$  có thể được viết thành:  $i--$  (giảm sau) hoặc  $--i$  (giảm trước).

**Ví dụ:** với  $i = 5 ; j = 10;$

a/  $i = ++j ; i = j$    kết quả  $i = 11, j = 11$

b/  $i = j++$    kết quả  $i = 10, j = 11$

c/  $j = --i + 2$    kết quả  $i = 4, j = 6$

d/  $j = i-- + 2$    kết quả  $i = 4, j = 7$

# 6. Các phép toán

## f. Toán tử gán: <Tên biến> = <biểu thức>

Có 3 loại phép gán khác nhau: phép gán đơn, phép gán kép và phép gán mở rộng.

### Ví dụ:

#### *Gán đơn:*

$i = 3;$

$i = i + 4;$

#### *Gán kép:*

$a = b = c = 5;$

$a = b + (c = 5);$

#### *Gán mở rộng:*

$x += y \Leftrightarrow x = x + y \quad x -= y \Leftrightarrow x = x - y \quad x *= y \Leftrightarrow x = x * y$

$x /= y \Leftrightarrow x = x / y \quad x \% = y \Leftrightarrow x = x \% y \quad x >>= y \Leftrightarrow x = x >> y$

$x <<= y \Leftrightarrow x = x << y \quad x \&= y \Leftrightarrow x = x \& y \quad x |= y \Leftrightarrow x = x | y$

$x ^= y \Leftrightarrow x = x ^ y$

# 7. Độ ưu tiên của các phép toán

**Thứ tự ưu tiên của các toán tử số học:**

Đầu tiên ++, -- sau đó là \*, /, % rồi mới đến +, -

**Thứ tự ưu tiên của các toán tử quan hệ và logic:**

Cao nhất: !

> >= <<=

== !=

&&

Thấp nhất: ||

# 7. Độ ưu tiên của các phép toán

Tổng kết về độ ưu tiên của các toán tử, với độ ưu tiên từ trên xuống dưới và từ trái qua phải.

<b>Cao nhất</b>	() [] ->
	sizeof() ! ~ ++ -- (Kiểu)* &
	* / %
	+ -
	<< >>
	< <= > >=
	== !=
	&
	^

# 7. Độ ưu tiên của các phép toán

	<code>&amp;&amp;</code>
	<code>  </code>
	<code>? :</code>
	<code>= += -= *= /= %= ^=  = &lt;&lt;= &gt;&gt;</code>
<b>Thấp nhất</b>	<code>,</code>

## 8. Câu lệnh

a. **Câu lệnh đơn:** là một câu lệnh không chứa các câu lệnh khác bên trong nó và kết thúc bằng một dấu chấm phẩy (;)

**Ví dụ:**

```
int x=5, y = 7; //một câu lệnh đơn  
x++; //một câu lệnh đơn  
x = x > y ? x : y ; //không là lệnh đơn
```

b. **Câu lệnh phức:** là một câu lệnh chứa câu lệnh khác bên trong nó hoặc một khối lệnh gồm nhiều câu lệnh như lệnh điều kiện (lệnh if), lệnh rẽ nhánh (lệnh switch), lệnh lặp (lệnh for, while, do ... while).

**Ví dụ:**

```
x = x > y ? x : y ; //lệnh phức
```

# Bài tập

1. Viết chương trình nhập vào 2 số thực. Tính và xuất kết quả tổng, hiệu, tích, thương của 2 số thực vừa nhập, kết quả lấy 2 số lẻ.
2. Viết chương trình đổi nhiệt độ từ đơn vị F (Ferarit) ra độ C (Celsius) theo công thức:  $C = 5/9 (F-32)$
3. Viết chương trình tính giá trị  $F(x)$  và  $G(x)$ , trong đó  $x$  là số nguyên nhập từ phím:  $F(x) = 5x^2 + 6x + 1$  và  $G(x) = 2x^4 - 5x^2 + 4x + 1$
4. Viết chương trình tính giá trị của biểu thức, trong đó  $x$  là số nguyên nhập từ phím:

$$F(x) = \frac{1+x}{1-x}$$

$$g(x) = \frac{3x^5 + 2x + \sqrt{x+1}}{5x^2 - 3}$$

# Bài tập

5. Viết chương trình nhập vào chiều dài, chiều rộng của 1 hình chữ nhật. Tính và xuất kết quả chu vi, diện tích của hình chữ nhật trên.
6. Viết chương trình nhập vào bán kính của 1 hình tròn. Tính và xuất kết quả chu vi, diện tích của hình tròn trên.
7. Viết chương trình nhập vào chiều dài cạnh 1 hình vuông. Tính và xuất kết quả chu vi, diện tích, đường chéo của hình vuông trên.
8. Nhập vào 2 số nguyên a,b. Tìm số lớn nhất trong 2 số.
9. Nhập 3 số nguyên a, b, c. Xuất số lớn nhất và số nhỏ nhất của 3 số đó.

Gợi ý: Sinh viên dùng biểu thức điều kiện.

10. Nhập vào 5 số nguyên. Tính trung bình cộng 5 số đó.





## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN CHƯƠNG TRÌNH

*BM Công nghệ phần mềm –  
FIT HUST*

7/15/2021

# I. Cấu trúc lệnh rẽ nhánh

# Nội dung

## 1. Lệnh rẽ nhánh

1. Lệnh if
2. Lệnh switch-case
3. Bài tập

## 2. Lệnh lặp

1. Lệnh for
2. Lệnh while
3. Lệnh do – while
4. Bài tập

# 1. Lệnh if

Câu lệnh if cho phép thay đổi luồng thực thi của câu lệnh dựa vào điều kiện, một câu lệnh hoặc một khối các câu lệnh sẽ được quyết định thực thi hay không được thực thi.

Ví dụ: Xét tính chẵn lẻ của một số nguyên a.

⇒ Chọn 1 trong 2 kết quả:

- Số a là số chẵn nếu a chia hết cho 2
- Số a là số lẻ nếu a không chia hết cho 2

# 1. Lệnh if (tt)

(2)

- Lệnh if đơn giản
- Lệnh if-else đơn giản
- Lệnh if lồng nhau

# Lệnh if đơn giản

## ☐ Cú pháp

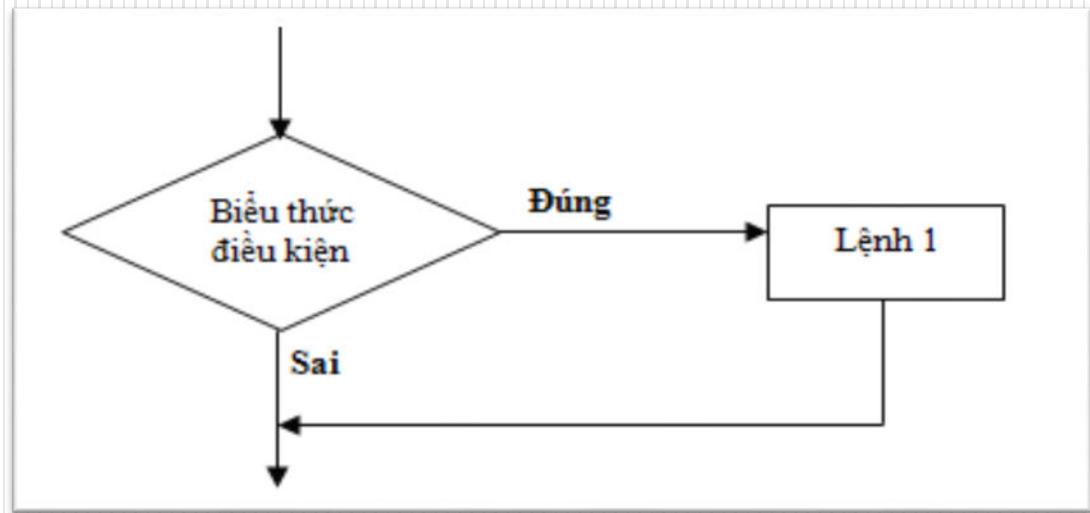
```
if (<Biểu thức điều kiện>)
    <Lệnh 1>
```

# Lệnh if đơn giản (tt)

(2)

## ☐ Lưu đồ

```
if (<BTĐK>)
    <Lệnh 1>
```



Lệnh 1: một câu lệnh đơn / một khối lệnh / một câu lệnh phức.

Các bước thực hiện:

B1: tính giá trị biểu thức điều kiện .

B2: kiểm tra

- Nếu điều kiện đúng (bằng 1) thì thực hiện Lệnh 1
- Ngược lại, nếu điều kiện sai (bằng 0) thì bỏ qua Lệnh 1

# Lệnh if đơn giản (tt)

(3)

- ❑ Ví dụ: Thực hiện chương trình nhập vào một số thực a. In ra màn hình kết quả nghịch đảo của a khi  $a \neq 0$

```
float a;
printf("Nhập a = "); scanf("%f",&a);
if (a !=0 )
    printf("Nghịch đảo của %f là %f",a,1/a);
printf("\n Kết thúc chương trình");
getch();
return 0;
```

- ❑ Chú ý: lệnh *printf(" \n Ket thuc chuong trinh")*; được thực hiện bình thường cho dù giá trị biểu thức điều kiện trong if đúng hay sai vì nó không phụ thuộc vào lệnh if

# Lệnh if-else đơn giản

## □ Cú pháp

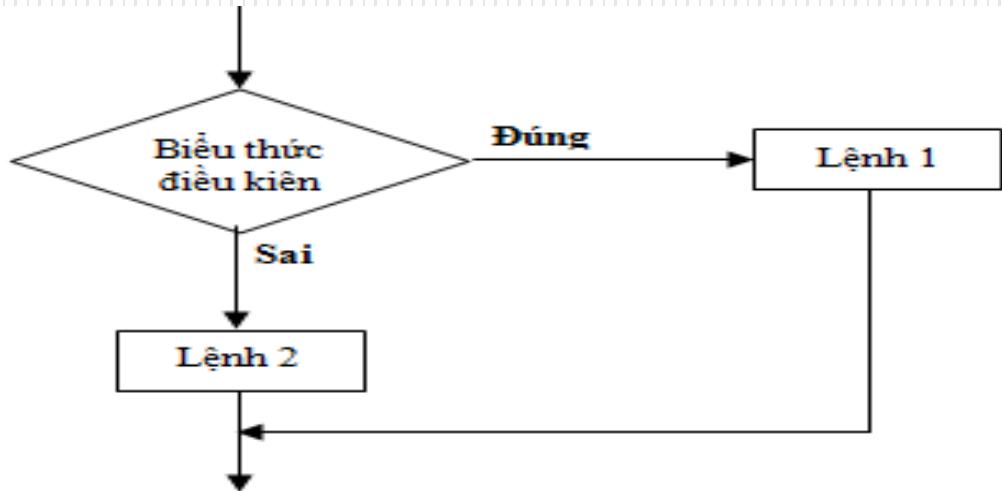
```
if (<Biểu thức điều kiện>)
    <Lệnh 1>
else
    <Lệnh 2>
```

# Lệnh if-else đơn giản (tt)

(2)

## ☐ Lưu đồ

```
if (<BTĐK>
    <Lệnh 1>
else
    <Lệnh 2>
```



Lệnh 1, Lệnh 2: một câu lệnh đơn / một khối lệnh / một câu lệnh phức.

Các bước thực hiện:

B1: tính giá trị biểu thức điều kiện .

B2: kiểm tra

- Nếu điều kiện đúng (bằng 1) thì thực hiện Lệnh 1

- Ngược lại, nếu điều kiện sai (bằng 0) thì thực hiện Lệnh 2

# Lệnh if-else đơn giản (tt)

(3)

- ❑ Ví dụ: Viết chương trình nhập vào một số thực a. In ra màn hình kết quả nghịch đảo của a khi  $a \neq 0$ , khi  $a=0$  in ra thông báo “Khong the tim duoc nghich dao cua a”

```
float a;  
printf("Nhap a = "); scanf("%f",&a);  
if (a !=0 )  
    printf("Nghich dao cua %f la %f",a,1/a);  
else  
    printf("Khong the tim nghich dao cua a");  
printf("\n Ket thuc chuong trinh");
```

- ❑ Chú ý: lệnh *printf(" \n Ket thuc chuong trinh")*; được thực hiện bình thường cho dù giá trị biểu thức điều kiện trong if đúng hay sai vì nó không phụ thuộc vào lệnh if-else

# Lệnh if lồng nhau

- ❑ Câu lệnh if đơn giản và if-else có thể lồng vào nhau
- ❑ Ví dụ: Nhập vào giá trị 2 số nguyên a, b. So sánh số a với số b vừa nhập

```
int a, b;  
printf("Nhập a = "); scanf("%d",&a);  
printf("Nhập b = "); scanf("%d",&b);  
if (a > b )  
    printf("a lớn hơn b");  
else  
    if(a==b)  
        printf("hai số bằng nhau");  
    else  
        printf("a nhỏ hơn b");
```

## 2. Lệnh switch-case

Câu lệnh `switch` nhánh `case` cho phép lựa chọn một trong các lựa chọn đã đưa ra. Nếu biểu thức có giá trị bằng với một giá trị sau từ khóa `case` nào đó, thì câu lệnh tương ứng sẽ được thực thi.

**Ví dụ:** Viết chương trình nhập vào giá trị tháng của năm, xuất số ngày trong tháng vừa nhập.

## 2. Lệnh switch-case (tt)

(2)

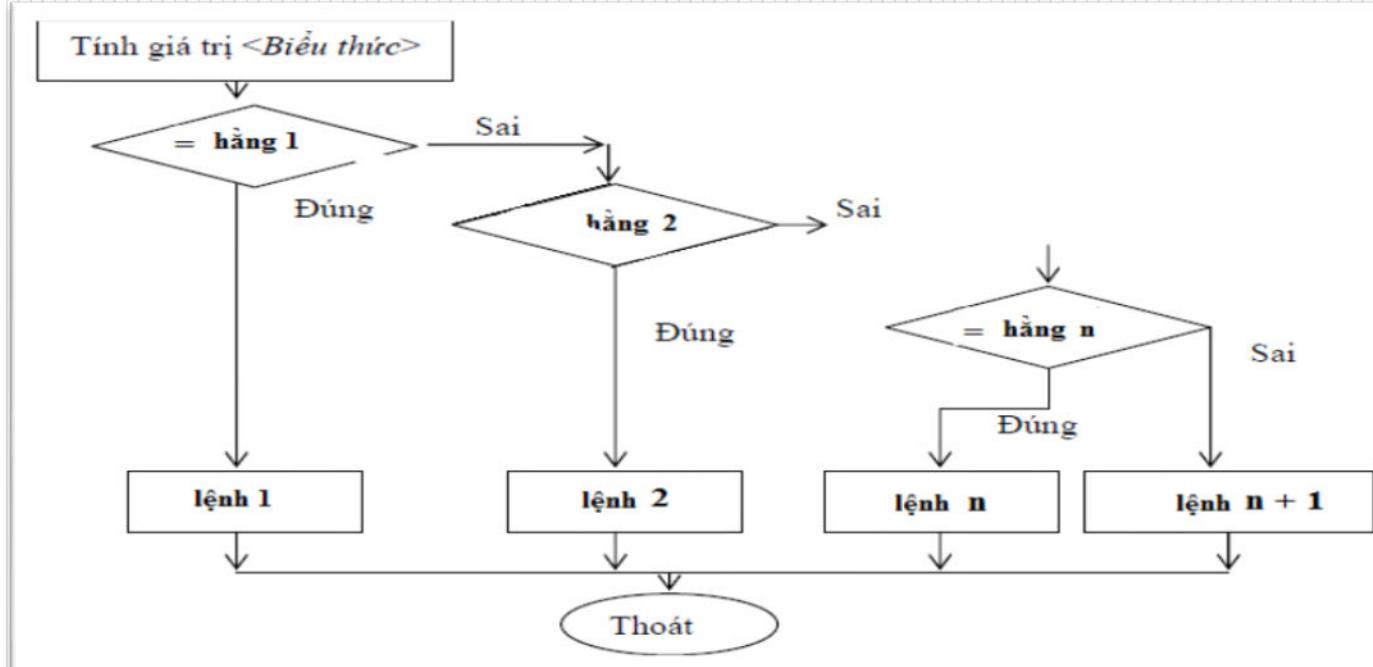
❑ Cú pháp:

```
switch (<biểu thức>)
{
    case <hàng 1> : lệnh 1 ;
    break;
    case <hàng 2> : lệnh 2 ;
    break;
    . . . . . .
    case <hàng n> : lệnh n ;
    break ;
    default      :      lệnh n+1;
}
```

## 2. Lệnh switch-case (tt)

(3)

☐ Lưu ý:



B1: Tính giá trị của biểu thức trước.

- B2: - Nếu giá trị của biểu thức bằng hằng 1 thì thực hiện lệnh 1 rồi thoát.  
- Nếu giá trị của biểu thức khác hằng 1 thì so sánh với hằng 2, nếu bằng hằng 2 thì thực hiện lệnh 2 rồi thoát.  
- Cứ như thế, so sánh tới hằng n.  
- Nếu tất cả các phép so sánh trên đều sai thì thực hiện lệnh n+1 mặc định của trường hợp *default*.

## 2. Lệnh switch-case (tt)

(4)

- ❑ Ví dụ: Viết chương trình nhập vào giá trị tháng của năm, xuất số ngày trong tháng vừa nhập.

# Bài tập (tt)

1. Viết chương trình nhập vào hai số a, b. Kiểm tra a có là bội số của a không.
2. Viết chương trình nhập vào đơn giá một mặt hàng, và số lượng bán của mặt hàng. Tính tiền khách phải trả, với thông tin như sau:

Thành tiền: đơn giá \* số lượng.  
Giảm giá : Nếu thành tiền > 100, thì giảm 3% thành tiền, ngược lại không giảm.  
Tổng tiền phải trả: thành tiền – giảm giá.
3. Viết chương trình hỗ trợ cách giải phương trình bậc 1 ( $ax + b = 0$ )
4. Nhập vào 1 số bất kỳ (0->9), cho biết cách đọc số vừa nhập.
5. Tính chu vi, diện tích của một trong các hình bên dưới (người dùng chọn hình): Hình vuông, Hình chữ nhật, Hình tròn.
6. Viết chương trình nhập vào hai số nguyên và lựa chọn phép toán (cộng, trừ, nhân, chia) để tính kết quả.

## II. Cấu trúc lệnh lặp

# 1. Khái niệm

Lệnh lặp là một câu lệnh, một đoạn lệnh trong chương trình được thực hiện lặp đi lặp lại nhiều lần cho đến khi một điều kiện xác định được thỏa mãn.

Một lệnh lặp cho phép lặp lại các câu lệnh nhiều lần.

**Ví dụ:** Viết chương trình xuất câu Xin chào 100 lần.

## 2. Lệnh for

- ❑ Lệnh for thực thi việc lặp lại một câu lệnh, một khối lệnh nhiều lần với số lần lặp xác định trước.
- ❑ Cú pháp

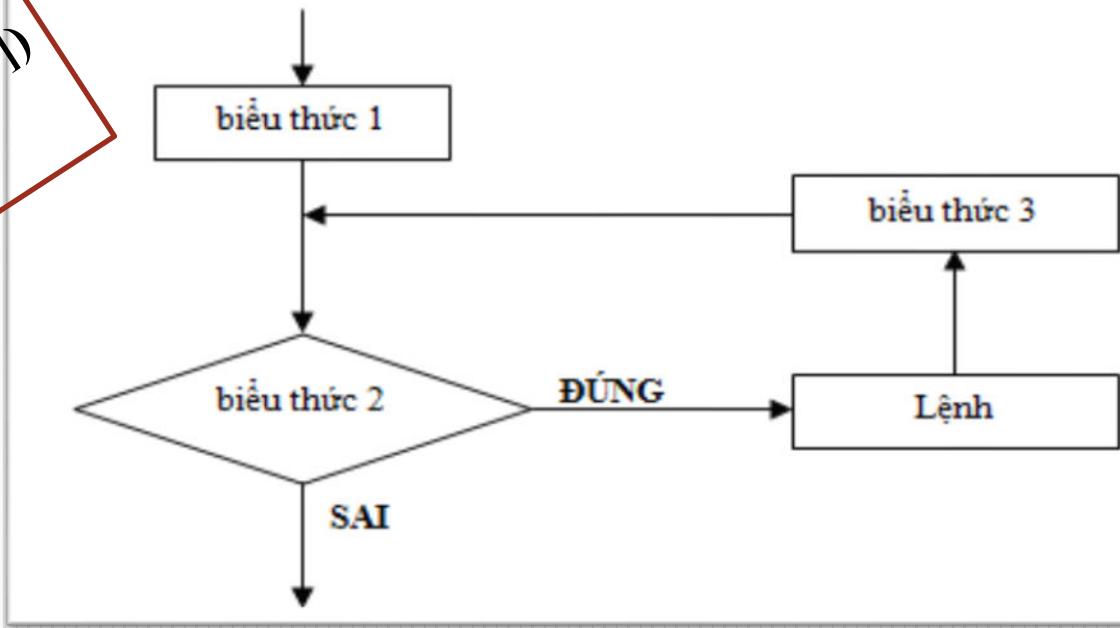
```
for ( [ <biểu thức 1>] ; [ <biểu thức 2> ] ; [ <biểu thức 3>])  
    <câu lệnh>;
```

## 2.Lệnh for (tt)

(2)

☐ Lưu đồ

for ([<BT1>]; [<BT2>]; [<BT3>])  
    <câu lệnh>;



Bước 1: Xác định giá trị của biểu thức 1.

Bước 2: Xác định giá trị của biểu thức 2.

Bước 3: Nếu biểu thức 2 sai thì sẽ thoát vòng lặp for;  
        Nếu biểu thức 2 đúng thì máy sẽ thực hiện Lệnh.

Bước 4: Tính giá trị của biểu thức 3 và quay lại Bước 2.

## 2.Lệnh for (tt)

(3)

- ❑ Ví dụ: Viết chương trình xuất câu Xin chao lên màn hình 20 lần.

```
printf("\n Xin chao!!! ");
```

```
int i;
for (i=1; i<=20; i++)
    printf("\n Xin chao!!! ");
```



## 2.Lệnh for (tt)

(4)

- ❑ Ví dụ: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

```
int n,i,tong;
printf("\n Nhap vao so nguyen duong n:");
scanf("%d",&n);
tong=0;
for (i=1; i<=n; i++)
    tong+=i;
printf("\n Tong tu 1 den %d =%d ",n,tong);
```

## 2.Lệnh for (tt)

(5)

### ❑ Chú ý:

- ❑ Biểu thức 1 là biểu thức gán trị khởi động cho biến lặp.
- ❑ Biểu thức 2 là biểu thức điều kiện. Nếu biểu thức 2 vắng mặt, điều kiện luôn đúng.
- ❑ Biểu thức 3 thông thường là biểu thức thay đổi điều kiện.
- ❑ Biểu thức 1, 3 có thể gồm nhiều biểu thức cách nhau bởi dấu phẩy.
- ❑ Biểu thức thứ 2 có thể bao gồm nhiều biểu thức, nhưng tính đúng sai của nó được xem là tính đúng sai của biểu thức cuối cùng.
- ❑ Các vòng for có thể lồng với nhau để thực hiện một câu lệnh nào đó.

## 2.Lệnh for (tt)

(6)

Ví dụ: Viết chương trình xuất bảng cửu chương từ 1- 9

```
for( i = 1 ; i <= 9 ; i++)
{
    printf("\n bang cuu chuong thu %d ", i);
    for(j = 1 ; j <= 9 ; j++)
        printf("\n %d x %d = %d ", i, j, i * j);
}
```

### 3. Lệnh while

- ❑ Lệnh while thực thi việc lặp lại một khối lệnh khi điều kiện kiểm tra là đúng. Điều kiện sẽ được kiểm tra trước khi vào thân vòng lặp do đó nếu có thay đổi giá trị kiểm tra ở trong thân vòng lặp thì khối lệnh vẫn được thực thi cho đến khi kết thúc khối lệnh.
- ❑ Cú pháp

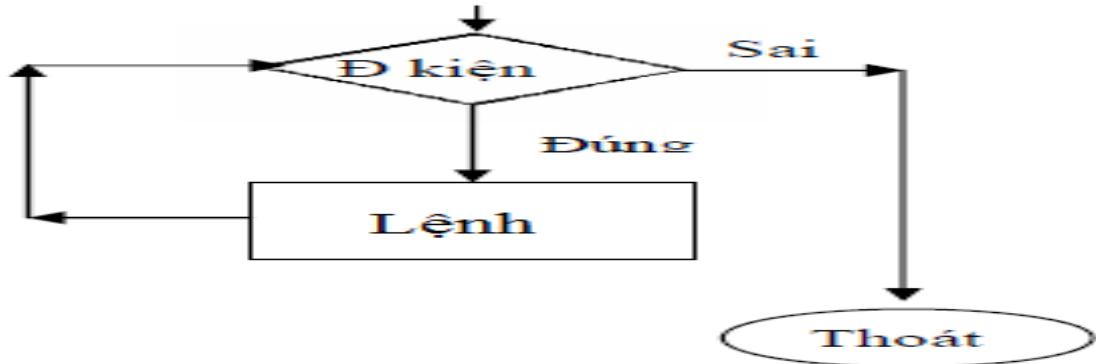
```
while(<biểu thức điều kiện>
      < Lệnh >)
```

### 3. Lệnh while (tt)

(2)

- ☐ Lưu đồ

**while(<BTĐK>)  
    < Lệnh >**



Bước 1: Tính giá trị của biểu thức điều kiện.

Bước 2: Nếu biểu thức điều kiện là sai (0), thì máy sẽ thoát ra khỏi vòng lặp.

Nếu biểu thức điều kiện là đúng (1) thì máy sẽ thực hiện câu lệnh và quay lại bước 1.

- ☐ Chú ý: Trong biểu thức điều kiện có thể gồm nhiều biểu thức cách nhau bởi dấu phẩy “,” nhưng tính đúng sai của nó là tính đúng sai của biểu thức sau cùng.

### 3.Lệnh while (tt)

(3)

- ❑ Ví dụ: Viết chương trình in các số nguyên từ 1 đến n, trong đó n nhập từ phím

```
int i,n;
printf("\n Nhập n:"); scanf("%d", &n);
printf("\n Day số từ 1 đến %d :",n);
i=1;
while (i<=n)
    printf("%d ",i++);
```

### 3.Lệnh while (tt)

(4)

#### □ Chú ý:

- Giá trị của biến được sử dụng trong biểu thức phải được thiết lập trước khi vòng lặp **while** thực hiện. Đây gọi là bước khởi tạo giá trị. Lệnh này chỉ thực hiện một lần trước khi thực hiện vòng lặp.
- Thân vòng lặp phải làm thay đổi giá trị của biến trong biểu thức kiểm tra. Biến này được gọi là biến tăng (**incremented**) nếu giá trị trong thân vòng lặp tăng, và được gọi là biến giảm (**decremented**) nếu giá trị giảm.

## 4. Lệnh do-while

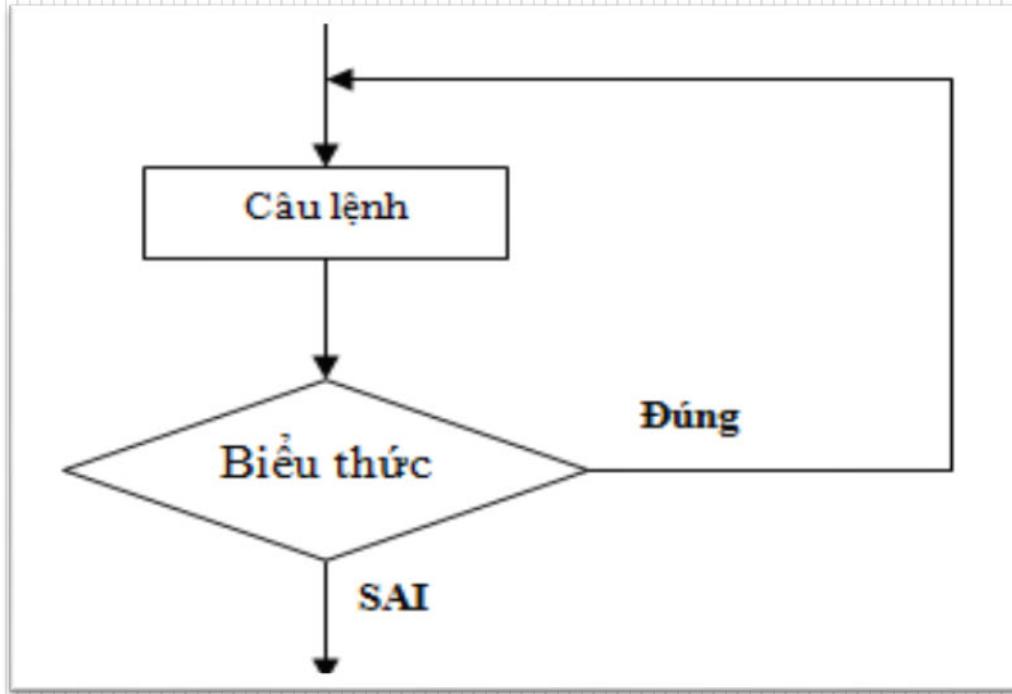
- ❑ Lệnh do...while thực thi lặp lại một khối lệnh nhiều lần.
- ❑ Khối lệnh được thực hiện ít nhất một lần. Sau đó sẽ kiểm tra điều kiện nếu điều kiện là đúng thì tiếp tục thực thi khối lệnh. Nếu điều kiện là sai thì kết thúc vòng lặp.
- ❑ Cú pháp

```
do  
    <câu lệnh>  
    while(<biểu thức>)
```

## 4. Lệnh do-while (tt)

(2)

❑ Lưu đồ:



Bước 1: Câu lệnh được thực hiện trước tiên.

Bước 2: Tính giá trị của biểu thức, nếu biểu thức đúng thì quay lại bước 1, nếu giá trị biểu thức sai thì ngừng vòng lặp.

## 4. Lệnh do-while (tt)

(3)

- ❑ Ví dụ: Viết chương trình nhập vào một số dương, nếu nhập số âm yêu cầu nhập lại.

```
int value;  
do  
{  
    printf( "Nhập vào một số dương");  
    scanf("%d",&value);  
} while(value <=0);
```

## 5. Lệnh break

- ❑ Câu lệnh break được dùng để thoát khỏi vòng lặp (bất chấp biểu thức điều kiện trong câu lệnh lặp đúng)

```
int i, j ;  
printf(" nhập hai số nguyên dương i và j :");  
scanf("%d%d",&i,&j);  
for( ; i > 0 &&j > 0; i- -,j- -)  
{  
    if( j == 5 ) break;  
    printf(" i = %d, j = %d ", i, j);  
}
```

## 6. Lệnh continue

- Câu lệnh continue được dùng để bỏ qua 1 bước lặp hiện tại của vòng lặp.

```
int i, tong ;  
for( i=1; i < 20; i++)  
{  
    if( i % 2 ==0 ) continue;  
    tong = tong + i;  
}  
printf(" ket qua: tong= %d ", tong);
```

# Bài tập

1. Viết chương trình in trên màn hình các số từ 1->100, các số ngăn cách nhau bởi 1 đoạn khoảng trắng.
2. Viết chương trình vẽ hình chữ nhật có kích thước d x r, trong đó d là chiều dài, và r là chiều rộng được nhập từ phím.

*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

3. Viết chương trình hiển thị tất cả các số lẻ nhỏ hơn n, trong đó n nhập từ phím.

# Bài tập (tt)

6. Viết chương trình in ra các số là bội số của 5 nhỏ hơn n, trong đó n nhập từ phím.
7. Viết chương trình nhập vào 2 số nguyên, tìm USCLN, BSCNN.
8. Viết chương trình đếm số lượng số chẵn trong [n, m], trong đó n, m nhập từ phím.
9. Viết chương trình tính tổng các số tự nhiên nhỏ hơn n.
10. Viết chương trình tìm tổng các số tự nhiên lớn nhất nhỏ hơn 100.





## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 4. HÀM (FUNCTION)

***BM Công nghệ phần mềm –  
FIT HUFI***

7/15/2021

# Nội dung

1. Khái niệm hàm (Function)
2. Định nghĩa hàm
3. Thực thi hàm
4. Truyền tham số
5. Kết quả trả về
6. Prototype của hàm

# Nội dung

**7. Các hàm chuẩn**

**8. Thư viện hàm**

**9. Sự đê quy**

**10. Bài tập**

# Mục tiêu bài học

- Trình bày được khái niệm hàm (function)
- Nêu được ý nghĩa của hàm con trong giải toán lập trình.
- Trình bày được cú pháp của hàm.
- Xác định được các thành phần của hàm trong bài toán thực tế
- Khai báo được tham số cho hàm.
- Khai báo prototype cho hàm.
- Phân loại được các dạng hàm.
- Nêu được một số hàm chuẩn trong NNLT C.
- Trình bày được khái niệm đệ qui và hàm đệ qui
- Vận dụng hàm con trong 1 số bài toán thông dụng

# Dân nhập

## □ Bài toán

**Viết chương trình tính  $C_n^k = \frac{n!}{k!(n-k)!}$  với n và k là các số nguyên dương  $k \leq n$ ;**

## □ Các bước giải quyết bài toán

- B1: Nhập k, n
- B2: Tính n!
- B3: Tính n! và (n-k)!
- B4: Tính  $C_n^k$

# Dân nhập

```
int n, k, int kq;
```

```
//Nhập n và k
```

```
//Tính n giai thừa
```

```
int n_giaithua=1;
```

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

```
    n_giaithua=n_giaithua*i;
```

```
//tính k giai thừa
```

```
int k_giaithua=1;
```

```
for(int i=1;i<=k;i++)
```

```
    k_giaithua=k_giaithua*i;
```

```
//Tính h=n-k giai thừa
```

```
int h=n-k, h_giaithua=1;
```

```
for(int i=1;i<=h;i++)
```

```
    h_giaithua=h_giaithua*i;
```

```
//tính kết quả
```

```
int m;
```

```
m=(k_gi
```

```
kq=n_gia
```

```
printf("kết
```

```
%5d \n",
```



# Dân nhập

## ☐ Nhận xét

- **Mục đích:** Cả ba đoạn chương trình đều mục đích là tính giai thừa của một số nguyên:  $n$ ,  $k$ ,  $h=n-k$
- **Câu lệnh:** Cấu trúc câu lệnh giống nhau

→ **Khó cải tiến, sửa chữa**

Cách khắc phục: Sử dụng hàm

## 4.1 Khái niệm hàm

```
int TinhGT(int x)
{
    int gt=1;
    for(int i=1;i<=x;i++)
        gt=gt*i;
    return gt;
}
```

TinhGT(k)

TinhGT(n)

TinhGT(h)

```
int n, k, int kq;
//Nhập n và k
//tính ket qua
int m, h=n-k;
m=( k_giaithua * h_giaithua );
kq=n_giaithua / m;
printf("ket qua can tinh la %5d
\n", kq);
```

## 4.1 Khái niệm hàm

❑ **Khái niệm:** Hàm là một phần của chương trình giải quyết một công việc nào đó.

❑ **Ưu điểm**

- Chương trình ngắn gọn, dễ hiểu, dễ quản lý
- Dễ bảo trì, sửa chữa
- Dễ kiểm tra lỗi

❑ **Ưu điểm**

- Hàm chuẩn
- Hàm tự định nghĩa

## 4.2 Định nghĩa hàm

```
<Tên kiểu kết quả><Tên hàm> ([<kiểu t số><tham số>][...])  
{  
    [<Khai báo biến cục bộ và các câu lệnh thực hiện hàm>]  
    [return <Biểu thức>;]  
}
```

- **Tên kiểu trả về:** là kiểu dữ liệu của kết quả trả về
- **Tên hàm:** là định danh trong C và không trùng với tên biến và hàm khác
- **Tham số:** là các dữ liệu đầu vào của hàm
- **Kiểu t số:** là kiểu dữ liệu của tham số

## 4.2 Định nghĩa hàm

### ❑ Ví dụ:

- Viết hàm tính tổng hai số nguyên.
- Viết hàm tính chu vi hình tròn khi biết bán kính  $r$ .
- Viết hàm tính tổng  $n$  số tự nhiên đầu tiên.
- Viết hàm cho biết số tự nhiên  $n$  có phải là số nguyên tố không?
- Viết hàm tìm ước chung lớn nhất của hai số nguyên dương
- Viết hàm tính tổng các chữ số trong số tự nhiên  $n$

## 4.2 Định nghĩa hàm

### ❑ Cách xác định hàm

- **Hàm sẽ thực hiện công việc gì?**
- **Tên hàm:** do người lập trình tự đặt, nên đặt tên hàm sao cho dễ nhớ và liên quan tới nội dung của hàm:

VD: KtraSNT, chuviHT, tong2songuyen,...

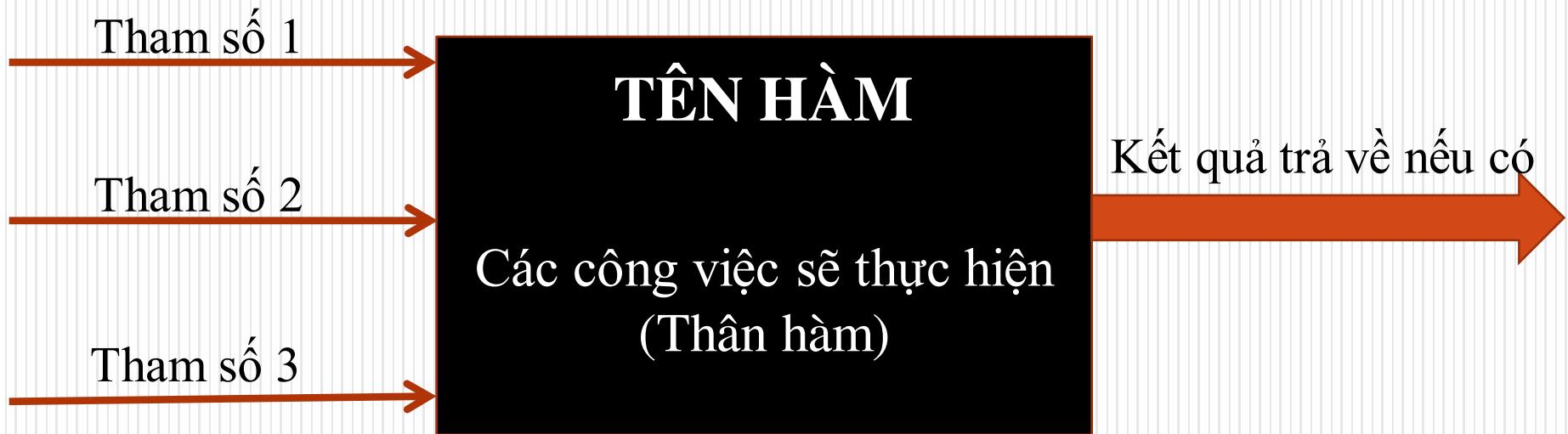
- **Kiểu trả về:** là kiểu của kết quả trả về, nếu không có kết quả trả về kiểu trả về là void

VD: T<sup>ổ</sup>ng 2 sô nguyên: int; Chu vi hình tròn: float

## 4.2 Định nghĩa hàm

- **Tham số** là các dữ liệu đầu vào của hàm
  - Tính tổng 2 số nguyên: 2 số nguyên a, b
  - Chu vi hình tròn: bán kính r
  - Kiểm số nguyên tố n: số tự nhiên n
- **Kiểu tham số**: là kiểu dữ liệu của dữ liệu đầu vào
  - Tổng 2 số nguyên: int a, int b
  - Chu vi hình tròn: float r
- **Thân hàm**: là thuật toán, các câu lệnh để giải quyết công việc của hàm.

## 4.2 Định nghĩa hàm



## 4.2 Định nghĩa hàm

Ví dụ: Viết hàm tính tổng 2 số nguyên

- Tên hàm
- Công thức: int Tong2songuyen(int a, int b)
- Tham số:
- Kiểu trả về:
- Kiểu tham số:
- Số n
- Tham số:

```
int tong=0;  
tong=a +b;  
return tong;
```

Tên hàm

DS tham số

Số nguyên là

Kiểu trả về

KQ trả về

## 4.2 Định nghĩa hàm

Ví dụ: Viết hàm tính chu vi hình tròn khi biết bán kính

➤ Tên hàm: chuviHT

➤ float chuviHT (float r)

➤ {

➤     return 2\*r\*3.14;

➤ }

➤

bán kính r

t

## 4.2 Định nghĩa hàm

**Ví dụ: In ra màn hình n từ xin chào!**

```
void Xuatloichao(int n)
{
    for(int i=0;i<n;i++)
        printf("xin chao!\n");
```

h n t

## 4.3 Thực thi hàm

- ❑ Thực thi làm là sử dụng hàm đã viết. Hàm chỉ được thực thi khi ta có một lời gọi tới hàm đó.

**<Tên hàm>([Danh sách các tham số])**

- ❑ Lưu ý khi sử dụng hàm

- Phải viết đúng tên hàm (giống phần khai báo)
- Phải truyền tham số chính xác về số lượng và kiểu tham số (tham số thực)
- Giá trị của hàm có thể được gán vào biến, hay sử dụng trong biểu thức tính toán, trong câu lệnh.

## 4.3 Thực thi hàm

❑ VD: Ta có định nghĩa hàm tính hiệu 2 số nguyên như sau:

```
int hieu2so(int a, int b)
{
    return a - b;
}
```

Tham số hình thức

Khi có nhu cầu tính hiệu 2 số nguyên x và y ( $x-y$ ) ta có thể gọi hàm như sau:

Tham số thực

```
int hieu=hieu2so(x,y);
```

## 4.3 Thực thi hàm

❑ VD: Gọi hàm sai

```
int hieu=hieu2so();
```

```
int hieu=hieu2so(x);
```

```
int hieu=hieu2so x,y ;
```

```
int hieu=hieu2so(x, y, z);
```

int hieu=hieu2so(x, y); với y là một mảng

## 4.3 Sử dụng hàm

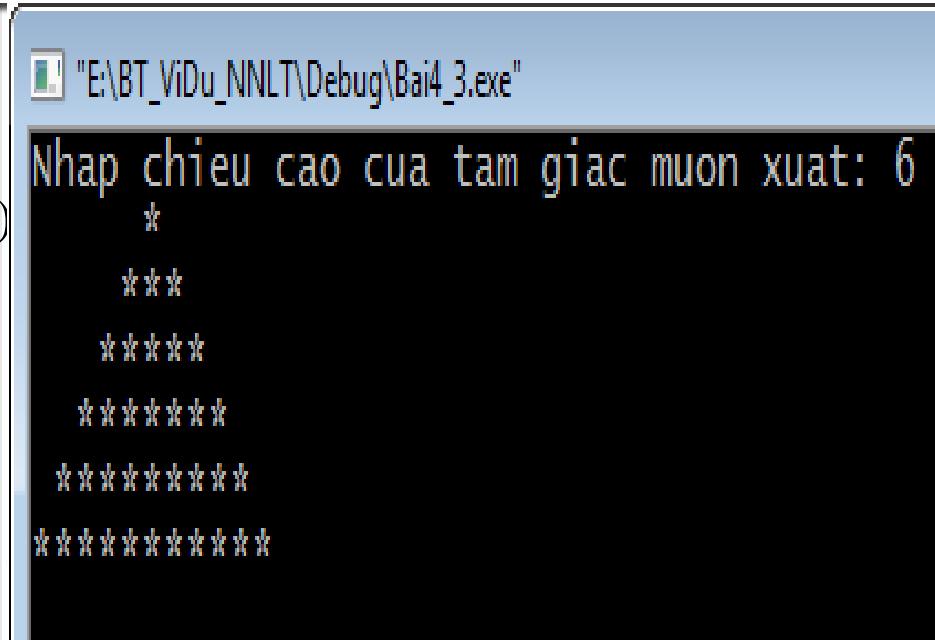
### ❑ Nguyên lý hoạt động khi gọi hàm

- Gán giá trị của tham số thực vào tham số hình thức
- Thực thi lần lượt từng câu lệnh trong hàm
- Kết thúc khi gặp câu lệnh return hoặc dấu } đóng hàm
- Tiếp tục trở về thực hiện chương trình đã gọi hàm

## 4.3 Sử dụng hàm

```
void XuatC(char x, int n)
{
    int i;
    for(i = 1; i <= n; i++) putchar(x)
}

void main(){
    int h, i;
    printf("Nhap chieu cao cua tam giac muon xuat: ");
    for(i = 1; i <= h; i++){
        XuatC(' ', h-i); //gọi hàm thực thi
        XuatC('*',2*i-1); //gọi hàm thực thi
        XuatC('\n',1); //gọi hàm thực thi
    }
    getch();
}
```



## 4.4 Truyền tham số

❑ Có hai loại tham số:

- Tham số dạng tham trị: không thay đổi giá trị của tham số khi truyền vào hàm
- Tham số dạng tham biến: cho phép thay đổi giá trị của tham số trong quá trình sử dụng hàm. Để khai báo một tham số dạng tham biến ta thêm dấu & vào trước tên biến trong khai báo hàm

## 4.4 Truyềñ tham sô

Tham trị

```
void swap(int x, int y)
```

```
{  
    int t=x; x=y; y=t;  
}
```

Tham biến

```
void swap(int&x, int &y)
```

```
{  
    int t=x; x=y; y=t;  
}
```

$m'=9$

$n'=5$

```
int m=5, n=9;  
swap(m,n);  
printf("m=%d, n=%d", m,n);
```

$m=5, n=9$

$m=9, n=5$

## 4.5 Kết quả trả về

### Có hai trường hợp

- ❑ Hàm có kết quả trả về: *trong thân hàm bắt buộc phải có lệnh return. Giá trị hàm muốn trả về là giá trị sau từ return*

```
double mean(int num1,int num2)
{
    return ( num1 + num2)/2;
}
```

## 4.5 Kết quả trả về

- ☐ Hàm không có kết quả trả về: *không có return hoặc có return nhưng không có giá trị phía sau*

```
void inHCNsao(int a,int b)
{
    for(int i=0;i<a;i++)
    {
        for(int j=0;j<b;j++)
            printf("* ");
        scanf("\n");
    }
}
```

## 4.6 Prototype của hàm

- ❑ Là khai báo nguyên mẫu của hàm: giống như định nghĩa hàm nhưng không có phần thân hàm.

**<Tên Kiểu><Tên Hàm> ([Danh sách các đối số]);**

- ❑ VD:

```
double atof(char s[]);  
void func(int i,int j);  
double luythua(double n, int so_mu);
```

## 4.7 Các hàm chuẩn

❑ Làm các hàm có sẵn trong ngôn ngữ lập trình C, có sẵn trong các thư viện

❑ VD:

Hàm scanf()

Hàm printf()

Hàm avg()

Hàm sqrt()

## 4.8 Thư viện hàm

- ❑ Thư viện hàm là tập hợp các hàm đã được xây dựng trước.
- ❑ Mỗi thư viện hàm chứa các hàm theo một công dụng riêng.
- ❑ Ví dụ:

math.h.

string.h

ctype.h

## 4.9 Sự đê qui

- ❑ Đê qui là việc định nghĩa một vấn đề dựa vào chính vấn đề đó nhưng ở mức độ khác
- ❑ VD:

$$n! = n^*(n-1)!$$

Tổng của mảng n phần tử = tổng mảng có n-1 phần tử cộng với giá trị của phần tử ở vị trí n-1

## 4.9 Sự đệ qui

❑ Đệ qui là việc định nghĩa một vấn đề dựa vào chính vấn đề đó nhưng ở mức độ khác

❑ VD:  $5! = 5 * 4!$

$$n!$$

$$4! = 4 * 3!$$

T

$$3! = 3 * 2!$$

có n-1

phân tử

$$2! = 2 * 1!$$

1

$$\Rightarrow 5! = 120$$

## 4.9 Sụ đệ qui

- ❑ Hàm đệ qui là hàm được định nghĩa dựa vào chính nó.
- ❑ Hàm đệ qui gồm 2 phần
  - Phần cơ sở: là phần không cần không cần đệ qui, dễ dàng tính được giá trị
  - Phần đệ qui: là phần thực hiện đệ qui
- ❑ VD: trong công thức tính n!
  - Phần cơ sở:  $n=1$  thì  $n!=1$
  - Phần đệ qui:  $n!=n*(n-1)!$

## 4.9 Sự đê qui

❑ Xác định các phần cơ sở và phần đê qui của bài toán sau

➤ Tính các số tự nhiên từ 1 tới n

➤ Tính  $\sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}}}$  (n dấu căn)

## 4.9 Sự đệ qui

- ❑ Viết hàm đệ qui: ta cũng viết thành 2 phần
- ❑ VD: Viết hàm tính n!

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

```
{
```

```
    if(n==1) return 1;
```

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

```
}
```

Phần cơ sở

Phần đệ qui

# Bài tập

Làm bài tập 1, 2, ..., 15 trang 62





## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 5. MẢNG – CON TRỎ - CHUỖI

*BM Công nghệ phần mềm –  
FIT HUFI*

7/15/2021

# NỘI DUNG

I. MẢNG 1 CHIỀU

II. MẢNG 2 CHIỀU

III. CHUỖI KÝ TỰ

# I. MẢNG 1 CHIỀU

## 5.1 Mảng 1 chiều

5.1.1 Khái niệm và khai báo mảng 1 chiều

5.1.2 Truy xuất các phần tử của mảng

5.1.3 Các phần tử của mảng trong bộ nhớ

5.1.4 Khởi tạo mảng

5.1.5 Tham số mảng của một hàm

5.1.6 Những thao tác trên mảng

Bài tập

## 5.1.1 Khái niệm và khai báo mảng 1 chiều

### Khái niệm:

một nhóm các phần tử  
được lưu liên tiếp với nhau  
trong bộ nhớ máy tính

Mảng một chiều là

có cùng

kiểu dữ liệu

và

chung tên

Nó chứa tối đa  
bao nhiêu phần tử?

Kiểu dữ liệu gì?

Tên là gì?

### Khai báo:

Kiểu dữ liệu

Tên mảng

[số phần tử tối đa của mảng] ;

## 5.1.1 Khái niệm và khai báo mảng 1 chiều

Ví dụ:

```
int a[100];
```

Mảng **a** chứa tối đa **100** số nguyên

```
#define maxN 30  
float b[maxN];
```

Mảng **b** chứa tối đa **30** số thực

```
const int maxC=50;  
char c[maxC];
```

Mảng **c** chứa tối đa **50** ký tự

## 5.1.1 Khái niệm và khai báo mảng 1 chiều

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

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

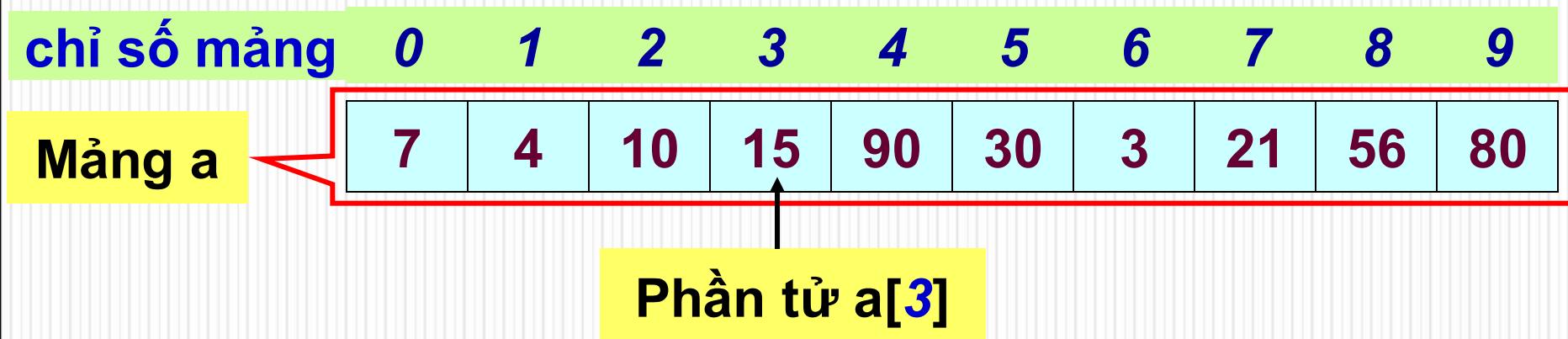
Một phần tử của mảng được xác định bởi chỉ số của nó trong mảng. Vì thế, phần tử thứ **i** trong mảng được định nghĩa bởi: **a[i]**

## 5.1.1 Khái niệm và khai báo mảng 1 chiều

- ❑ Mỗi phần tử của mảng một chiều được truy xuất giá trị thông qua chỉ số **i** (index) của nó. Chỉ số để xác định phần tử nằm ở vị trí nào trong mảng. Phần tử đầu tiên của mảng có chỉ số là **0**, phần tử thứ hai có chỉ số là **1**,...và tương tự tăng dần cho đến khi hết mảng.
- ❑ Nếu mảng có **n** phần tử thì chỉ số mảng  **$i \in [0, n-1]$**

## 5.1.1 Khái niệm và khai báo mảng 1 chiều

Ví dụ: Cho mảng 1 chiều tên a chứa 10 số nguyên như sau:



Gọi  $i$  là chỉ số của mảng  $\rightarrow i \in [0, 9]$ .

$a[i]$  là giá trị của phần tử thứ  $i$ .

## 5.1.2 Truy xuất các phần tử của mảng

❑ **Mỗi phần tử** của mảng một chiều được **truy xuất** thông qua **chỉ số** của phần tử đó trong mảng.

❑ **Cú pháp:**

**<Tên mảng>[<chỉ số phần tử>]**

- ❑ Chẳng hạn muốn truy xuất một phần tử thứ i của mảng a, ta ghi là a[i]. Giá trị i được tính từ 0.
- ❑ **Ví dụ:** Truy xuất phần tử thứ 2 của mảng *a* như sau: *a[1]*

# Ví dụ

Cho mảng một chiều a có tối đa 5 phần tử: *int a[5];*  
Hãy truy xuất các phần tử sau của a với các chỉ số phần tử tính từ 0.

- Phần tử đầu tiên của mảng
- Phần tử thứ 3
- Phần tử có chỉ số 5
- phần tử cuối cùng của mảng

a[0]

a[2]

Chỉ số 5 không tồn tại vì  
chỉ số của mảng a từ 0→4

a[4]

## 5.1.2 Truy xuất các phần tử của mảng

❑ Khởi tạo giá trị cho mảng một chiều là gán giá trị cho từng phần tử trong mảng.

❑ **Có nhiều cách:**

- **Cách 1:** Nhập trực tiếp từ bàn phím

`scanf("%d", &a[i]);`

Hay

`cin>>a[i];`

## 5.1.2 Truy xuất các phần tử của mảng

- **Cách 2:** Gán giá trị cho từng phần tử.

`a[2]=5; //gán giá trị cho phần tử thứ 3 của mảng a là 5`

- **Cách 3:** Ngoài ra, có thể khởi tạo giá trị cụ thể cho các phần tử của mảng một chiều trong khi khai báo mảng.
- **Ví dụ:** Khởi tạo mảng một chiều a chứa số nguyên có 10 phần tử với các giá trị 5, 15, 20, 25, 30 như sau:  
`int a[10]={5, 15, 20, 25, 30};`

## 5.1.2 Truy xuất các phần tử của mảng

☐ Khi muốn lấy giá trị một phần tử trong mảng tại vị trí chỉ số phần tử, ta có thể:

- Truy xuất trực tiếp.
- Khai báo một biến để nhận giá trị của phần tử mảng thông qua phép gán (*biến này phải có cùng kiểu dữ liệu với phần tử của mảng*).

## 5.1.2 Truy xuất các phần tử của mảng

- Giả sử có mảng một chiều a có 5 phần tử là số nguyên như sau:

<i>Chỉ số mảng</i>	0	1	2	3	4
<i>Giá trị phần tử</i>	20	30	10	4	6

- Với đoạn lệnh sau:

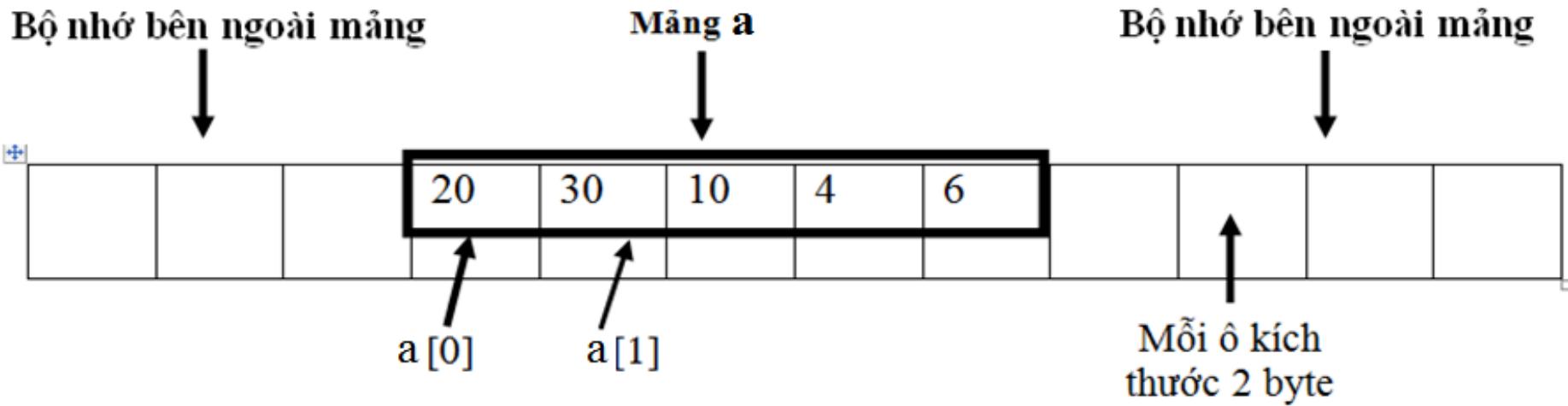
```
int value;
```

```
value = a[2]; // value nhận giá trị là 10
```

→ Lệnh trên sẽ lấy giá trị 10 lưu tại phần tử thứ 3 của mảng, và giá trị này sẽ được gán vào biến value.

## 5.1.3 Các phần tử của mảng trong bộ nhớ

- ❑ Những phần tử của mảng được chứa trong vùng nhớ một cách liên tục.
- ❑ Ví dụ: Xét mảng a có 5 phần tử ở ví dụ trong phần 5.1.2, ta có mô phỏng cách lưu trữ dữ liệu như hình sau:



## 5.1.4 Khởi tạo mảng

- ❑ Trong ngôn ngữ lập trình C, có thể khởi tạo giá trị các phần tử mảng ngay khi mảng được khai báo.
- ❑ Việc khởi tạo cho mảng được thực hiện lúc khai báo mảng bằng một loạt giá trị hằng khởi động cho các phần tử của mảng.
- ❑ Các giá trị hằng được đặt giữa một cặp ngoặc nhọn ({ }), các phần tử cách nhau bằng dấu phẩy (,).

## 5.1.4 Khởi tạo mảng

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
```

## 5.1.4 Khởi tạo mảng

- ❑ Trong việc khởi tạo mảng, kích thước của mảng không cần xác định, chương trình C sẽ đếm số phần tử được khởi động và lấy đó làm kích thước. Nếu có xác định kích thước, thì số giá trị được khởi tạo liệt kê phải không được lớn hơn kích thước đã khai báo
- ❑ *Ví dụ:* Ta khai báo: **double b[ ] = {4.5, 7.5, 8.2, 4.32};** thì mảng b sẽ được hiểu là có 4 phần tử kiểu double.

## 5.1.4 Khởi tạo mảng

- Nếu khai báo **int a[10] = {2, 5, 6, 1, 0, 6, 7};**  
→ thì một mảng a gồm 10 phần tử được tạo ra. Trong đó có 7 phần tử đầu tiên được khởi tạo giá trị, các phần tử còn lại chưa được khởi tạo giá trị.
- Lưu ý: Trình biên dịch C không báo lỗi khi có sự vượt quá giới hạn của mảng.

## 5.1.5 Tham số mảng của một hàm

- ❑ Tham số mảng luôn luôn là một tham biến.
- ❑ Một tham số mảng xác định bởi:

**param\_a[SIZE]**

Cho biết số phần tử tối đa

hoặc

**param\_a[ ]**

Không cho biết số phần tử tối đa

hoặc

**\*param\_a**

Khai báo dạng con trả

## 5.1.5 Tham số mảng của một hàm

❑ Ví dụ:

- Nhập một mảng các số nguyên với n phần tử

**void Input (int a[SIZE ], int &n)**

**void Input (int a[ ], int &n)**

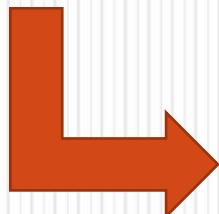
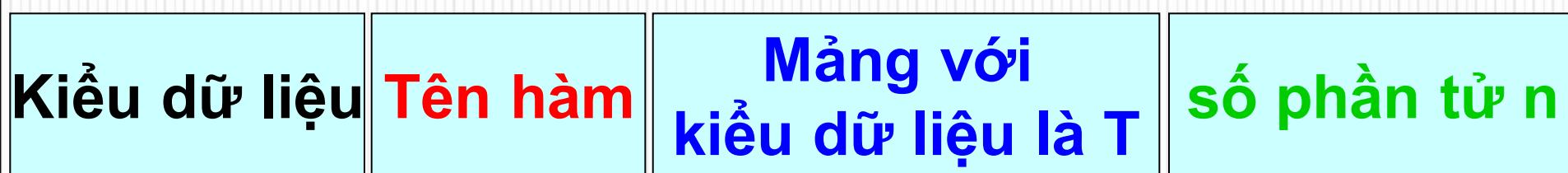
**void Input (int \*a, int &n)**

- Tìm giá trị lớn nhất của mảng các số nguyên với n phần tử

**int getMax(int a[ ], int n)**

## 5.1.5 Tham số mảng của một hàm

❑ Công thức chung:



**Kiểu\_dữ\_liệu Tên\_hàm ( T a[ ] , int n )**

{

**<Khởi tạo các biến >**

**<Duyệt mảng để tính toán>**

**< trả về giá trị >**

}

## 5.1.5 Tham số mảng của một hàm

❑ Ví dụ:

**void Input(int a[ ], int &n)**

```
{  
    printf("\n nhap so phan tu cua mang:");  
    scanf("%d", &n);  
    for (int i=0; i<n; i++)  
    {  
        printf("nhap a[%d]= ", i);  
        scanf("%d", &a[i]);  
    }  
}
```

## 5.1.6 Những thao tác trên mảng

- Xử lý mảng = Xử lý nhóm các phần tử = Xử lý mỗi phần tử trong mảng.

### ➤ Duyệt mảng theo chiều 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 (Thỏa điều kiện)  
 Xử lý a[i];`

### ➤ Duyệt mảng theo chiều ngược

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

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

## 5.1.6 Những thao tác trên mảng

```
#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("noi dung mang:");
    for (i=0; i<5; i++)
        printf("%3d", a[i]);
    for (i=0; i<5; i++)
        S1 += a[i];
    printf("\nTong mang: %d", S1);
    for (i=0; i<5; i++)
        if(a[i]%2 == 0) S2+=a[i];
    printf("\nTong mang chan: %d", S2);
    for (i=4; i>=0; i--)
        if(a[i]%2 != 0) S3+=a[i];
    printf("\nTong mang le: %d", S3);
    getch();
}
```

noi dung mang: 4 6 3 8 9  
Tong mang: 30  
Tong mang chan: 18  
Tong mang le: 12

Duyệt mảng không có điều kiện

Duyệt mảng có điều kiện

Duyệt mảng ngược

## 5.1.6 Những thao tác trên mảng

- Nhập mảng
- Xuất mảng
- Lấy giá trị lớn nhất/ nhỏ nhất trong mảng
- Tính tổng mảng
- Tính giá trị trung bình của mảng
- Đảo mảng
- Kiểm tra tính đối xứng của mảng
- Tìm kiếm các phần tử trong mảng
- Sắp xếp mảng
- Thêm/xóa phần tử của mảng

## 5.1.6 Những thao tác trên mảng

```
#include <conio.h>
#include <stdio.h>
void NhapMang(int a[ ], int &n)
{
    printf("\n nhap so phan tu cua mang:");
    scanf("%d", &n);
    for (int i=0; i<n; i++)
    {
        printf("nhap a[%d]= ", i); scanf("%d", &a[i]); }
}
void XuatMang(int a[ ], int n)
{
    printf("Noi dung mang:");
    for(int i=0; i<n; i++)
        printf("%3d", a[i]);
}
void main()
{
    int a[100], n;
    NhapMang(a, n);
    XuatMang(a, n);
    getch();
}
```

Viết một chương trình C  
cho phép nhập mảng và  
sau đó xuất mảng này ra  
màn hình.

### Kết quả

```
nhap so phan tu cua mang:3
nhap a[0]: 1
nhap a[1]: 3
nhap a[2]: 6
Noi dung mang: 1 3 6
```

## 5.1.6 Những thao tác trên mảng

```
#include <conio.h>
#include <stdio.h>
void NhapMang(int a[ ], int &n)
{
    n=0; int x;
    do{
        printf("nhap mot so nguyen. Nhap so 0 de dung: ");
        scanf("%d", &x);
        if(x!=0) a[n++]=x;
    }while(x!=0);
}
void XuatMang(int a[ ], int n)
{
    printf("Noi dung mang:");
    for(int i=0; i<n; i++) printf("%3d", a[i]);
}
void main()
{
    int a[100], n; clrscr();
    NhapMang(a, n);
    XuatMang(a, n);
    getch();
}
```

Viết một chương trình C cho phép nhập một mảng các số nguyên, sau đó xuất mảng này ra màn hình. Thao tác nhập sẽ kết thúc khi người dùng nhập số 0. Giá trị 0 không đưa vào mảng.

### Kết quả

```
nhap mot so nguyen. Nhap so 0 de dung: 4
nhap mot so nguyen. Nhap so 0 de dung: 5
nhap mot so nguyen. Nhap so 0 de dung: 6
nhap mot so nguyen. Nhap so 0 de dung: 3
nhap mot so nguyen. Nhap so 0 de dung: 0
Noi dung mang: 4 5 6 3
```

## 5.1.6 Những thao tác trên mảng

Tìm giá trị lớn nhất/nhỏ nhất của mảng

n=6

a	7	6	9	4	3	12
---	---	---	---	---	---	----

YEAH!  
Số lớn nhất  
là 12

max



// Thuật toán lấy giá trị lớn nhất  
**max=a[0];**  
Duyệt với  $i=1..n-1$   
    Nếu (**max < a[i]**)  
        **max=a[i];**  
Trả về **max;**

Bài tập: tìm giá trị nhỏ nhất

## 5.1.6 Những thao tác trên mảng

Tính tổng/trung bình cộng các phần tử của mảng

Tính tổng

$$S = a_0 + a_1 + a_2 + \dots + a_{n-1}$$

$$\text{TBC} = \frac{a_0 + a_1 + a_2 + \dots + a_{n-1}}{n}$$

Tính tổng:

$$S=0$$

Duyệt  $i=0 .. n-1$

$$S=S+a_i$$

Trả về  $S$

Tính trung bình cộng:

$$S=0$$

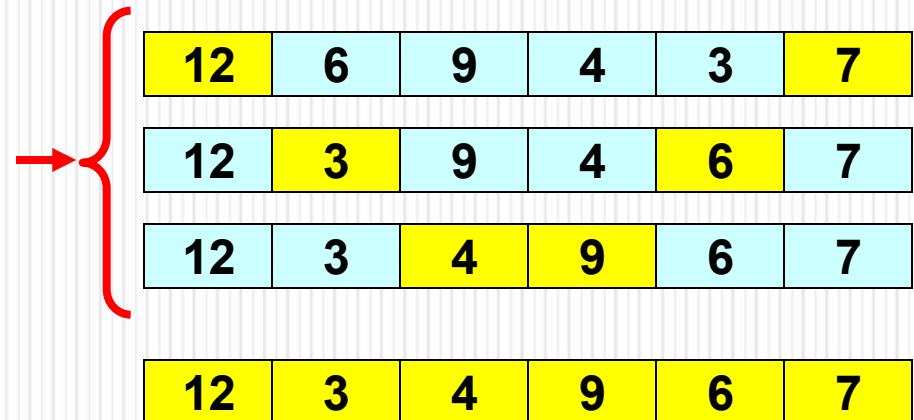
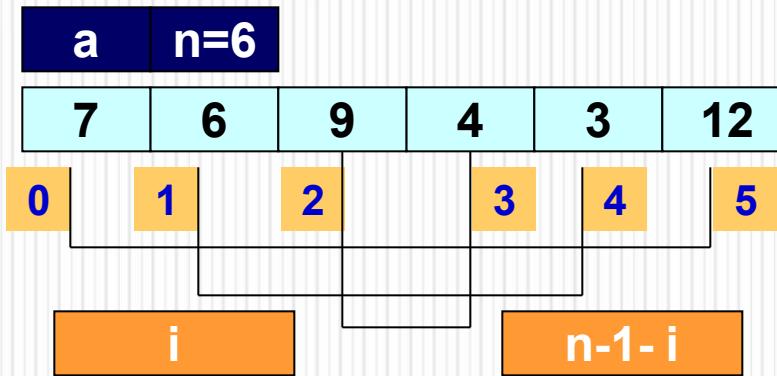
Duyệt  $i=0 .. n-1$

$$S=S+a_i$$

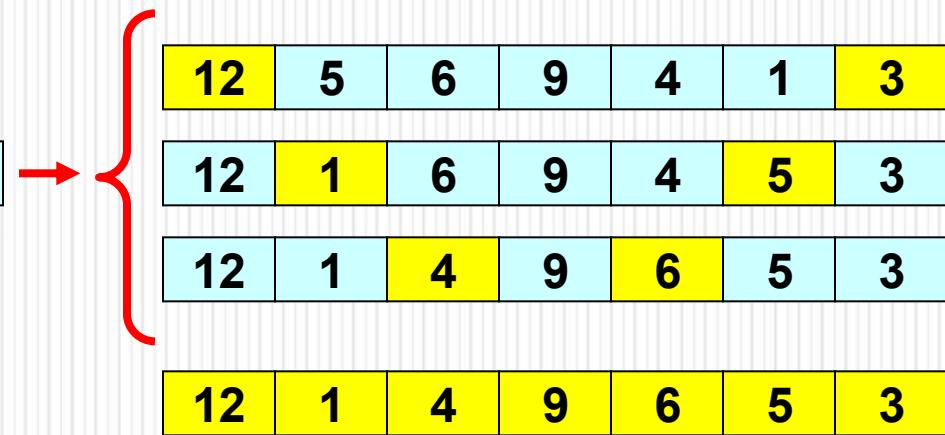
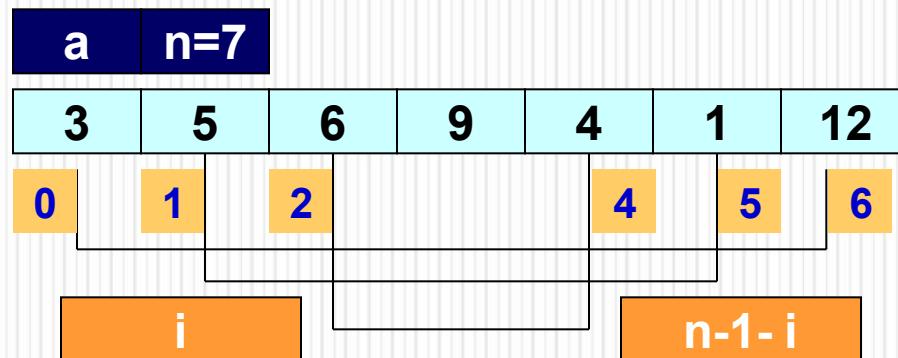
Trả về  $S/n$

## 5.1.6 Những thao tác trên mảng

### Đảo ngược mảng



Hoán vị  $n/2$  lần



## 5.1.6 Những thao tác trên mảng

### Kiểm tra tính đối xứng của mảng

- ❑ Một mảng là đối xứng nếu tất cả các phần tử đều đối xứng.
- ❑ Ví dụ:
  - 1 2 3 3 2 1 là mảng đối xứng
  - 1 2 3 4 3 2 1 là mảng đối xứng
  - 1 2 [3] 4 4 [5] 2 1 là mảng không đối xứng

## 5.1.6 Những thao tác trên mảng

**Viết một chương trình nhập vào mảng các số nguyên và kiểm tra xem chúng có đối xứng không?**

```
int KiemTraDoiXung (int a[ ], int n)
{
    for (int i=0; i<n/2; i++)
        if (a[i] != a[n-1-i])
            return 0;
    return 1;
}
```

```
void main()
{
    int a[100], n;
    NhapMang (a, n);
    if (KiemTraDoiXung(a, n)==1)
        printf("Đối xứng!");
    else
        printf("Không đối xứng!");
    getch();
}
```

## 5.1.6 Những thao tác trên mảng

### Tìm một giá trị trong mảng

- Trả về vị trí của giá trị tồn tại trong mảng.
- Tìm kiếm tuần tự

#### Tìm kiếm vị trí đầu tiên mà x tồn tại trong mảng

```
int TimX(int a[ ], int n, int x)
{
    for(int i=0; i<n; i++)
        if(a[i]==x)
            return i; //Vị trí tìm thấy x
    return -1; //Không tìm thấy x
}
```

Bài tập: Tìm kiếm vị trí cuối cùng mà x tồn tại trong mảng

## 5.1.6 Những thao tác trên mảng

### Đếm số lần xuất hiện x trong mảng

```
dem=0;  
Duyệt i=0 .. n-1  
    Nếu (a[i]==x)  
        dem=dem+1;  
Trả về dem;
```

**Viết một chương trình thực hiện**

- Nhập mảng các số nguyên
- In ra tất cả các phần tử trong mảng
- Nhập một số nguyên x
- In ra số lần xuất hiện giá trị x trong mảng

```
int DemSoLanXuatHien(int a[ ], int n, int x)  
{//Sinh viên tự viết xem như bài tập  
.....  
}  
  
void main()  
{  
    int a[100], n, x;  
    NhapMang(a, n);  
    XuatMang(a, n);  
  
    printf("Nhập giá trị x=");  
    scanf("%d", &x);  
    int k= DemSoLanXuatHien (a, n, x);  
    printf("Số lần xuất hiện của %d là %d", x, k);  
}
```

## 5.1.6 Những thao tác trên mảng

Sắp xếp

Sắp xếp  
Bubble  
tăng

Di chuyển  
giá trị nhỏ  
nhất lên

$i=0$		$j=1$	$j=2$	$j=3$	$j=4$	for ( $j=n-1; j>i; j-$ ) if ( $a[j]<a[j-1]$ ) Swap ( $a[j], a[j-1]$ );
$i=1$		$j=2$	$j=3$	$j=4$		for ( $j=n-1; j>i; j-$ ) if ( $a[j]<a[j-1]$ ) Swap ( $a[j], a[j-1]$ );
$i=2$		$j=3$	$j=4$			for ( $j=n-1; j>i; j-$ ) if ( $a[j]<a[j-1]$ ) Swap ( $a[j], a[j-1]$ );
$i=3$		$j=4$				for ( $j=n-1; j>i; j-$ ) if ( $a[j]<a[j-1]$ ) Swap ( $a[j], a[j-1]$ );

## 5.1.6 Những thao tác trên mảng

- Sắp xếp tăng dần mảng a có n phần tử sử dụng Bubble sort

```
void BubbleAscendingSort (int a[ ], int n )
```

```
{
```

```
    int i, j, t ;  
    for (i=0; i<n-1; i++)  
        for (j=n-1; j>i ; j--)  
            if (a[j] < a[j-1])  
                {  
                    //Hoán vị  
                    t = a[j];  
                    a[j]= a[j-1];  
                    a[j-1] = t;  
                }  
}
```

**Bài tập:** Viết một hàm sắp xếp các số thực theo chiều giảm dần.

## 5.1.6 Những thao tác trên mảng

Thêm một phần tử x vào vị trí thứ k của mảng

❑ Kiểm tra nếu  $k \in [0, n]$  thì:

- Dời các phần tử từ vị trí  $n-1$  đến  $k$  lùi lại 1 vị trí.
- Thêm  $x$  vào vị trí thứ  $k$  của mảng, tăng  $n$  thêm 1.

```
void ThemX(int a[], int &n, int x, int k)
{
    if(k<0 || k>n) return;
    for(int i=n; i>k; i--)
        a[i]=a[i-1];//Dời phần tử lùi 1 v.trí
    a[k]=x;//Thêm x vào vị trí thứ k
    n++;//Tăng n thêm 1
}
```

## 5.1.6 Những thao tác trên mảng

### Xóa một phần tử x khỏi mảng

❑ Kiểm tra nếu x có tồn tại trong mảng thì:

- Dời các phần tử sau x tới 1 vị trí.
- Giảm n bớt 1.

```
void XoaX(int a[], int &n, int x)
{
    int k=TimX(a, n, x);
    if(k== -1) return;
    for(int i=k; i<n-1; i++)
        a[i]=a[i+1];//Dời phần tử tới 1 v.trí
    n--;//Giảm n bớt 1
}
```

# Bài tập

1. Viết hàm nhập mảng 1 chiều các số nguyên.
2. Viết hàm xuất mảng 1 chiều các số nguyên.
3. Viết hàm liệt kê các giá trị chẵn trong mảng 1 chiều các số nguyên.
4. Viết hàm liệt kê các vị trí mà giá trị tại đó là giá trị âm trong mảng 1 chiều các số nguyên.
5. Viết hàm tìm “*giá trị nhỏ nhất*” trong mảng 1 chiều các số nguyên.
6. Viết hàm tìm “*số chính phương đầu tiên*” trong mảng 1 chiều các số nguyên, nếu mảng không có số chính phương nào thì trả về -1.
7. Viết hàm tìm đoạn  $[a,b]$  sao cho đoạn này chứa tất cả các giá trị của mảng 1 chiều các số nguyên.
8. Viết hàm liệt kê các giá trị âm trong mảng 1 chiều các số nguyên.
9. Viết hàm liệt kê các giá trị trong mảng 1 chiều các số thực thuộc đoạn  $[x,y]$  cho trước.
10. Viết hàm tính tổng các giá trị lẻ trong mảng 1 chiều các số nguyên.

# Bài tập

11. Viết hàm tính tổng các giá trị có chữ số hàng chục là chữ số 5 trong mảng 1 chiều các số nguyên.
12. Viết hàm tính tổng trung bình cộng các giá trị là số nguyên tố trong mảng 1 chiều các số nguyên.
13. Viết hàm đếm số lượng các giá trị chẵn trong mảng 1 chiều các số nguyên.
14. Viết hàm đếm số lượng các giá trị dương chia hết cho 3 trong mảng 1 chiều các số nguyên.
15. Viết hàm kiểm tra mảng số nguyên có tồn tại giá trị 0 hay không? Nếu có thì trả về 1, ngược lại trả về 0.
16. Viết hàm kiểm tra mảng số thực có đối xứng hay không? Nếu có thì trả về 1, ngược lại trả về 0.
17. Viết hàm sắp xếp các phần tử của mảng số nguyên tăng dần.
18. Viết hàm sắp xếp các phần tử của mảng số nguyên sao cho các phần tử chẵn tăng dần, các phần tử lẻ thì giảm dần.

# Bài tập

19. Viết hàm thêm một phần tử có giá trị x vào đầu mảng các số nguyên.
20. Viết hàm thêm một phần tử có giá trị x vào vị trí k của mảng các số nguyên.
21. Viết hàm xóa một phần tử ở đầu mảng các số nguyên.
22. Viết hàm xóa tất cả các phần tử âm của mảng các số nguyên.
23. Viết hàm để đưa tất cả các phần tử có giá trị 0 về đầu mảng.
24. Viết hàm để đưa tất cả các phần tử có dạng  $2^k$  về đầu mảng.
25. Viết hàm liệt kê các dãy con tăng trong mảng 1 chiều các số nguyên.
26. Cho 2 mảng 1 chiều các số nguyên a và b. Hãy viết hàm kiểm tra xem mảng a có phải là mảng con trong mảng b không?
27. Cho mảng 1 chiều các số nguyên a. Hãy tạo 2 mảng b và c sao cho mảng b chỉ chứa giá trị dương và mảng c chỉ chứa giá trị âm từ mảng a.
28. Cho 2 mảng 1 chiều các số nguyên b, c đã sắp xếp tăng dần. Hãy tạo mảng a từ b và c sao cho a cũng chứa các giá trị tăng dần.

## II. MẢNG 2 CHIỀU

- 1. Khái niệm**
- 2. Khai báo**
- 3. Chỉ số mảng**
- 4. Truy xuất phần tử mảng**
- 5. Khởi tạo mảng 2 chiều**
- 6. Ma trận vuông**

### Bài tập

# Đặt vấn đề

Bảng thông tin nhiệt độ trung bình sáng/trưa/chiều và tối cho các ngày trong tuần tại TPHCM từ thứ 2 đến chủ nhật như sau:

Thứ <del>Buổi</del>	Thứ 2	Thứ 3	Thứ 4	Thứ 5	Thứ 6	Thứ 7	Chủ nhật
Sáng	20	21	19	19	20.5	20	21
Trưa	38	36	35.5	37	38	34	33
Chiều	32	30	30	31	31	30	29
Tối	22	20	19	19	18	21	20

**Vấn đề:** lưu trữ bảng dữ liệu thế nào để việc truy xuất các thông tin được nhanh chóng và chính xác?

- ❖ **Nếu lưu thành từng biến riêng biệt.** → Ưu/ khuyết điểm của cách này?
- ❖ **Nếu lưu thành mảng 1 chiều.** → Ưu/ khuyết điểm của cách này?

# 1. Khái niệm

Mảng hai chiều *m dòng n cột* được xem như là một bảng hình chữ nhật chứa  $m \times n$  phần tử cùng kiểu dữ liệu. Mảng 2 chiều còn gọi là *ma trận cấp m×n*

Ví dụ: Cho mảng 2 chiều a có 4 dòng 3 cột chứa số nguyên như sau:

Cột →

Dòng ↓

7	8	10
4	6	8
9	10	10
3	5	7

## 2. Khai báo

### ❑ Cú pháp:

**<định danh><định nghĩa> [<số dòng>] [<số cột>];**

### ❑ Ví dụ:

❑ Khai báo mảng 2 chiều a có 3 dòng 2 cột chứa số nguyên:

**int a[3][2];**

❑ Khai báo mảng 2 chiều ch có 50 dòng 100 cột chứa ký tự:

**char ch[50][100];**

# Ví dụ

Khai báo các mảng 2 chiều sau:

- a. mảng b có 4 dòng 5 cột chứa số thực.

**float b[4][5];**

- b. mảng abc có 20 dòng 30 cột chứa số nguyên.

**int abc[20][30];**

- c. mảng DSDiem có 100 dòng 20 cột chứa điểm trung bình  
của học sinh.

**float DSDiem[100][20];**

### 3. Chỉ số mảng

- ❑ Trong mảng 2 chiều, chỉ số dòng và cột bắt đầu từ 0.
- ❑ Giả sử ma trận 2 chiều  $a$  có  $m$  dòng  $n$  cột thì:
  - ❑ chỉ số dòng từ 0 đến  $m-1$
  - ❑ chỉ số cột từ 0 đến  $n-1$

Ví dụ:

- ❑ mảng  $a$  có 4 dòng 3 cột, thì các phần tử xét theo chỉ số mảng như sau:

$a[0][0]$	$a[0][1]$	$a[0][2]$
$a[1][0]$	$a[1][1]$	$a[1][2]$
$a[2][0]$	$a[2][1]$	$a[2][2]$
$a[3][0]$	$a[3][1]$	$a[3][2]$

### 3. Chỉ số mảng (tt)

Mảng a có m dòng và n cột có các phần tử xét theo chỉ số mảng thế nào?

Cột  $0 \rightarrow n-1$

Dòng:  
 $0 \rightarrow m-1$

$a[0][0]$	$a[0][1]$	...	$a[0][n-1]$
$a[1][0]$	$a[1][1]$	...	$a[1][n-1]$
...	...	...	...
$a[m-1][0]$	$a[m-1][1]$	....	$a[m-1][n-1]$

## 4. Truy xuất phần tử mảng

- ❑ Mỗi phần tử mảng hai chiều được truy xuất thông qua chỉ số dòng và chỉ số cột của phần tử đó trong mảng hai chiều.

### ❑ Cú pháp:

```
<tênmảng>[<chỉ số dòng phần tử>][<chỉ số cột phần tử>]
```

- ❑ Chẳng hạn muốn truy xuất một phần tử tại dòng thứ i, cột thứ j của mảng a, ta ghi là a[i][j]. Giá trị i, j được tính từ 0.
- ❑ **Ví dụ:** Truy xuất phần tử dòng 2 cột 1 của mảng *diem* như sau:  
*diem[1][0]*

# Ví dụ:

Cho mảng 2 chiều a có 4 dòng 5 cột: *int a[4][5];*

Hãy truy xuất các phần tử sau của a với các chỉ số dòng và cột tính từ 0.

1. phần tử đầu tiên của mảng

a[0][0]

2. phần tử ở dòng 1 cột 2

a[1][2]

3. phần tử ở dòng 4 cột 3

Dòng 4 không tồn tại vì dòng của  
mảng a từ 0→3

4. phần tử ở dòng 2 cột 5

Cột 5 không tồn tại vì cột của mảng a  
từ 0→4

5. phần tử cuối cùng của mang

a[3][4]

## 5. Khởi tạo mảng 2 chiều

- Khởi tạo giá trị cho mảng hai chiều là gán giá trị cho từng phần tử trong mảng hai chiều. Có nhiều cách: nhập hoặc gán giá trị cho từng phần tử.
- Ngoài ra, có thể khởi tạo giá trị cụ thể cho các phần tử của mảng hai chiều trong khi khai báo mảng hai chiều.

### □ Ví dụ

```
int matrix[5][4] = { {11, 12, 13, 14},  
                     {21, 15, 41, 16},  
                     {43, 58, 24, 91},  
                     {32, 15, 25, 16},  
                     {56, 23, 45, 47} };
```

## 5. Khởi tạo mảng 2 chiều (tt)

### □ Ví dụ

```
int quy_thang [4][3]={ 1,2,3, 4,5,6 };
```

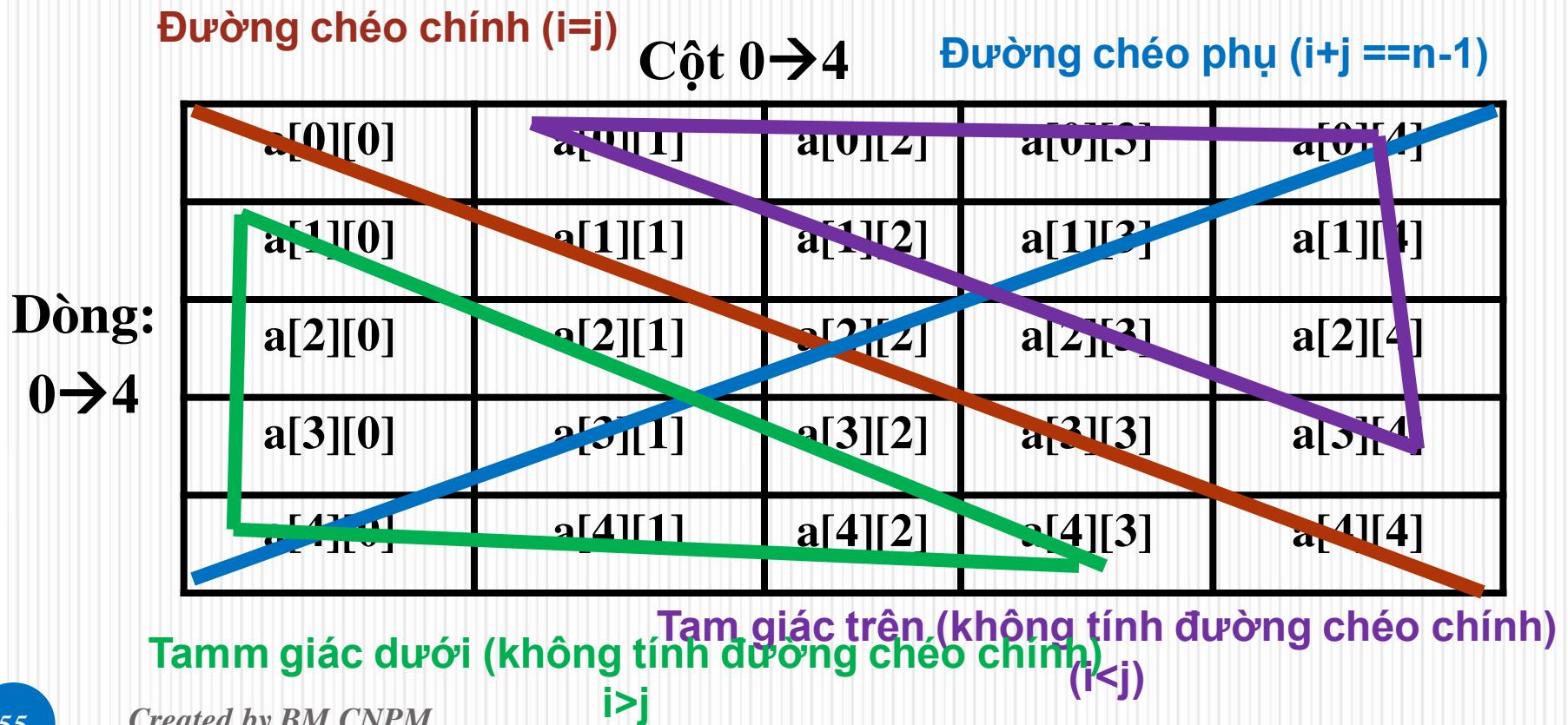
Trong mảng này có  $4*3$  phần tử, nhưng chỉ có 6 phần tử đầu của mảng có giá trị, các phần tử còn lại có giá trị không xác định.

## 6. Ma trận vuông

- **Ma trận vuông** là mảng 2 chiều a có **số dòng bằng số cột**.
- **Ma trận vuông** có **số dòng và số cột bằng n** gọi là **ma trận vuông cấp n**.
- Một số khái niệm trong ma trận vuông:  
Cho ma trận vuông a cấp n.
  - **đường chéo chính:** gồm các phần tử  $a[i][j]$  với  $i=j$ ,  $0 \leq i, j \leq n$
  - **đường chéo phụ:** gồm các phần tử  $a[i][j]$  với  $i+j = n-1$ ,  $0 \leq i, j \leq n$
  - **tam giác trên:** gồm các phần tử  $a[i][j]$  với  $i < j$ ,  $0 \leq i, j \leq n$
  - **tam giác dưới:** gồm các phần tử  $a[i][j]$  với  $i > j$ ,  $0 \leq i, j \leq n$

## 6. Ma trận vuông (tt)

- Ví dụ : Xét ma trận vuông a cấp 5



## 7. Các thao tác nhập/xuất ma trận

Hàm nhập/xuất ma trận a có m dòng, n cột

```
void nhapM2C (int a[][50], int m, int n)
{
    for(int i=0; i<m; i++)
        for(int j=0; j<n; j++)
    {
        cout<<“ Nhập phần tử thu dòng “<<i<<“ cột “<<j;
        cin>> a[i][j];
    }
}
```

```
void xuatM2C (int a[][50], int m, int n)
{
    for(int i=0; i<m; i++)
    {
        for(int j=0; j<n; j++)
            cout<< a[i][j]<< “ “;
        cout<<endl;
    }
}
```

# Bài tập

## Phần I: Cho mảng 2 chiều a có m dòng n cột chứa số nguyên:

1. Viết hàm xuất mảng a đảo cột thành dòng (xuất theo từng cột, mỗi cột phải xuống hàng).
2. Viết hàm tạo giá trị ngẫu nhiên cho các phần tử của mảng a. Gợi ý: dùng hàm rand() trong thư viện <stdlib.h> để sinh số ngẫu nhiên
3. Viết hàm nhập a sao cho các phần tử toàn là số chẵn, nếu nhập sai thì nhập lại.
4. Viết hàm xuất các số lẻ/ số chẵn trong a.
5. Xuất các số chẵn ở dòng k/số lẻ ở cột k của a.
6. Tính tổng các phần tử mảng.
7. Tính tổng các số chẵn thuộc dòng lẻ và cột chẵn của a.
8. Xuất tích các phần tử trên từng dòng.
9. Tìm max/min trên ma trận.
10. Tìm max của dòng k trong mảng a.

# Bài tập (tt)

## Phần II: Cho ma trận vuông a cấp n chứa số nguyên:

1. Kiểm tra ma trận có a vuông không?
2. Kiểm tra ma trận a có toàn là số chẵn không?
3. Xuất các phần tử trên đường chéo chính.
4. Xuất các phần tử trên đường chéo phụ
5. Xuất các phần tử thuộc tam giác trên.
6. Xuất các phần tử thuộc tam giác dưới.
7. Tính tổng các phần tử chẵn thuộc tam giác trên và tam giác dưới.
8. Đếm số phần tử cực đại trong ma trận a. biết phần tử cực đại là phần tử lớn hơn các phần tử xung quanh nó.
9. Xuất các cột có tổng lớn nhất trong a.
10. Xuất các dòng có tổng nhỏ nhất trong a.

# III. CHUỖI KÝ TỰ

## 1. Chuỗi ký tự

### 1.1 Khái niệm

### 1.2 Khai báo

### 1.3 Nhập/xuất chuỗi

## 2. Các hàm thao tác trên chuỗi

## 3. Bài tập

## 1.1 Khái niệm

- ❑ Chuỗi là mảng ký tự kết thúc bởi ký tự **null** ('\0').
- ❑ Có thể gán các hằng chuỗi cho các biến chuỗi.
- ❑ Hằng chuỗi là một chuỗi các ký tự nằm trong dấu nháy kép.
- ❑ Ký tự null '\0' được tự động thêm vào biểu diễn bên trong của chuỗi.
- ❑ Khi khai báo một biến chuỗi, hãy dành thêm một phần tử trống cho ký tự kết thúc.

## 1.2 Khai báo

- ❑ Khai báo một biến chuỗi tiêu biểu:  
**char str[10];**
- ❑ **str** là một biến mảng ký tự có thể lưu giữ tối đa 10 ký tự bao gồm cả ký tự kết thúc.

## 1.3 Nhập xuất chuỗi

- ❑ Sử dụng các hàm trong thư viện nhập/xuất chuẩn **stdio.h** để thực hiện các thao tác nhập/xuất chuỗi.
- ❑ Hàm gets() là cách đơn giản nhất để nhập vào một chuỗi thông qua thiết bị nhập chuẩn.
- ❑ Các ký tự được nhập vào cho đến khi ấn phím Enter
- ❑ Hàm gets() thay thế ký tự sang dòng mới ‘\n’ bằng ký tự ‘\0’
- ❑ Cú pháp: **gets(str);**

## 1.3 Nhập xuất chuỗi

- ❑ Hàm puts() được dùng để hiển thị một chuỗi trên thiết bị xuất chuẩn.
- ❑ Cú pháp : **puts(str);**
- ❑ Các hàm scanf() và printf() được sử dụng để nhập và hiển thị các kiểu dữ liệu hỗn hợp trong cùng một câu lệnh.
- ❑ Cú pháp để nhập chuỗi:  
**scanf(“%s”, str);**
- ❑ Cú pháp để hiển thị chuỗi:  
**printf(“%s”, str);**

## 2. Các hàm thao tác trên chuỗi

Các hàm xử lý chuỗi nằm trong tập tin **string.h**. Một số thao tác được thực hiện bởi các hàm này là:

- Ghép chuỗi
- So sánh chuỗi
- Xác định vị trí một ký tự trong chuỗi
- Sao chép một chuỗi sang chuỗi khác
- Tính chiều dài chuỗi

## 2.1 Hàm strcat()

- ❑ Nối hai giá trị chuỗi vào một chuỗi.
- ❑ Cú pháp:  
**strcat(str1, str2);**
- ❑ Nối str2 vào cuối chuỗi str1
- ❑ Trả về str1
- ❑ Ví dụ: str1 = “hello ”, str2 = “Peter” thì strcat (str1, str2) trả về kết quả str1 là “hello Peter”

## 2.2 Hàm strcmp()

- ❑ So sánh hai chuỗi và trả về một giá trị số nguyên dựa trên kết quả của sự so sánh.
- ❑ Cú pháp:

**strcmp(str1, str2);**

- ❑ Hàm trả về một giá trị:
  - ✓ Nhỏ hơn 0, nếu str1 < str2
  - ✓ 0, nếu str1 giống str2
  - ✓ Lớn hơn 0, nếu str1 > str2
- ❑ Ví dụ: str1 = “Anh”, str2 = “Ban” thì strcmp (str1, str2) cho kết quả là -1 (<) vì “Anh” < “Ban” theo thứ tự alpha.

## 2.3 Hàm strchr()

- ❑ Xác định vị trí xuất hiện của một ký tự trong một chuỗi.
- ❑ Cú pháp: **strchr(str, chr);**
- ❑ Hàm trả về :
  - ✓ Con trả trở đến vị trí tìm được đầu tiên của ký tự (trở bởi **chr**) trong chuỗi **str**.
  - ✓ NULL nếu **chr** không có trong chuỗi

## 2.4 Hàm strcpy()

- ❑ Sao chép giá trị trong một chuỗi vào một chuỗi khác.
- ❑ Cú pháp:

**strcpy(str1, str2);**

- ❑ Giá trị của str2 được chép vào str1
- ❑ Hàm trả về str1
- ❑ Ví dụ: str1 = “”, str2 = “Peter” thì strcpy (str1, str2) cho kết quả str2 = str2 = “Peter”.

## 2.5 Hàm strlen()

- ❑ Xác định chiều dài của chuỗi.

- ❑ Cú pháp:

**strlen(str);**

- ❑ Hàm trả về một giá trị nguyên là độ dài của str.
- ❑ Ví dụ: str = “Peter” thì strlen(str) cho kết quả là 5.

## 2.5 Hàm strrev()

- ❑ Đảo chuỗi
- ❑ Cú pháp:

**strrev(str);**

- ❑ Hàm trả về chuỗi mới là chuỗi đảo của chuỗi str
- ❑ Ví dụ: str = “peter” thì strrev (str) cho kết quả str là “retep”

# Bài tập

Cho s là một chuỗi ký tự

1. Kiểm tra s có phải là chuỗi đối xứng không?
2. Xóa khoảng trắng thừa trong chuỗi s
3. Liệt kê mỗi từ trong chuỗi s trên 1 dòng.
4. Có bao nhiêu từ trong chuỗi s
5. Giả sử s là đường dẫn tuyệt đối. Viết hàm lấy tên file từ đường dẫn tuyệt đối đó.
6. Nhập chuỗi nhị phân -> In ra chuỗi thập phân
7. In ra từ dài nhất của chuỗi.
8. Có bao nhiêu ký tự a trong chuỗi. In ra các vị trí của ký tự a trong chuỗi

# Bài tập

Giả sử s là chuỗi họ tên đã chuẩn hóa. Viết các hàm sau:

1. Lấy ra họ
2. Lấy ra tên
3. Lấy ra họ lót (nếu có)
4. Đổi ký tự đầu từ thành chữ hoa
5. Đổi ký tự thường thành chuỗi hoa và ngược lại
6. Có bao nhiêu nguyên âm, phụ âm, khoảng trắng trong chuỗi s.

# Bài tập

1. Nhập chuỗi st, đếm xem trong st:
  - Có bao nhiêu chữ cái: A, B, C,...Z
  - Có bao nhiêu chữ số 0, 1, 2, ..... 9
  - Có bao nhiêu khoảng trắng
  - Có bao nhiêu kí tự khác
  - Trong bốn loại trên, loại nào nhiều nhất
2. Có hai chuỗi s và s1. s1 xuất hiện bao nhiêu lần trong s, tại vị trí nào?
3. Có 3 chuỗi s, s1, s2. Tìm xem trong s có s1 hay không? Nếu có thì thay bằng s2

# ÔN TẬP CHƯƠNG

# Nội dung

## 1. Những kiến thức quan trọng

1.1. Mảng 1 chiều

1.2. Mảng 2 chiều

1.3. Chuỗi

## 2. Bài tập

2.1. Bài tập mảng 1 chiều

2.2. Bài tập mảng 2 chiều

2.3. Bài tập xử lý chuỗi

# 1. Những kiến thức quan trọng

## 1.1. Mảng 1 chiều (M1C)

□ Cú pháp khai báo M1C:

<Tên kiểu dữ liệu> <Tên mảng> [ <số phần tử>];

□ Thứ tự chỉ số của mảng 1 chiều a có n phần tử:

$a[0], a[1], \dots a[n-1]$

□ Lấy giá trị các phần tử trong mảng:

<tên mảng>[<chỉ số phần tử>]

### Ví dụ:

- Khai báo M1C a chứa 10 phần tử là số nguyên: *int a[10];*
- Các phần tử của a gồm:  $a[0], a[1], \dots, a[9]$ .
- Truy xuất phần tử thứ i của mảng a:  $a[i]$

# 1. Những kiến thức quan trọng (tt)

## 1.2. Mảng 2 chiều (M2C)

- Cú pháp khai báo M2C:

<đại lượng><tên mảng> [<số dòng>] [<số cột>];

- Thứ tự chỉ số của m2C a có m dòng n cột chứa  $m \times n$  phần tử gồm:

A[0][0]	A[0][1]	A[0][2]	...	A[0][m-2]	A[0][m-1]
A[1][0]	A[1][1]	A[1][2]	...	A[1][m-2]	A[1][m-1]
...	...	...	...	...	...
A[n][0]	A[n][1]	A[n][2]	...	A[n][m-2]	A[n][m-1]

- Lấy giá trị các phân tử trong mảng:

<tên mảng> [<chỉ số dòng phân tử>] [<chỉ số cột phân tử>]

# 1. Những kiến thức quan trọng (tt)

## 1.2. Mảng 2 chiều (M2C) (tt)

### Ví dụ:

- Khai báo M2C a có 2 dòng 3 cột chứa số nguyên: `int a[2][3];`
- Các phần tử của a gồm:

<b>a[0][0]</b>	<b>a[0][1]</b>	<b>a[0][2]</b>
<b>a[1][0]</b>	<b>a[1][1]</b>	<b>a[1][2]</b>

- Truy xuất phần tử ở dòng i cột j của mảng a: `a[i][j]`

# 1. Những kiến thức quan trọng (tt)

## 1.3. Con trỏ

- ❑ Khai báo biến con trỏ:

**<định danh> \*<biến con trỏ>**

- ❑ Toán tử địa chỉ (&) và toán tử nội dung (\*)

❑ Biến được khai báo là x thì &x là địa chỉ của x.

❑ Toán tử lấy nội dung của một địa chỉ được kí hiệu là dấu \* trước một pointer, dùng để lấy giá trị của biến mà con trỏ đang trỏ đến.

- ❑ Cấp phát và giải phóng vùng nhớ cho biến con trỏ

❑ **void \*malloc(size\_t size):** Cấp phát vùng nhớ có kích thước size.

❑ **void \*calloc(size\_t nitems, size\_t size):** Cấp phát vùng nhớ có kích thước là nitems\*size.

❑ **void free(void \*block):** Giải phóng vùng nhớ được quản lý bởi con trỏ block

# 1. Những kiến thức quan trọng (tt)

## 1.3. Con trỏ (tt)

### Ví dụ

*int a = 10; //khai báo biến a kiểu số nguyên*

*int \*p; //Khai báo biến con trỏ p kiểu số nguyên*

*p = &a;// giá trị p chứa địa chỉ của biến a*

*int num1 = 2, num2, \*pnt;*

*pnt = &num1 //pnt chứa địa chỉ của num1*

*num2 = \*pnt;// num2 nhận giá trị của num1 thông qua phép  
//tính toán lấy nội dung của biến con trỏ pnt*

# 1. Những kiến thức quan trọng (tt)

## 1.4. Chuỗi

- ❑ Chuỗi trong C là một dãy các kí tự kiểu char.
- ❑ Một chuỗi trong C được đánh dấu kết thúc là ‘\0’ (còn gọi là NULL trong bảng mã ASCII) và có độ dài tùy ý.
- ❑ Ví dụ: Khai báo chuỗi

*char str[20] = “\nHappy New Year”;*

*char \*str;*

*str = “Happy new year \n”;*

# 1. Những kiến thức quan trọng (tt)

## 1.4. Chuỗi

❑ Một số hàm thông dụng trong xử lý chuỗi thuộc thư viện

<string.h> : strcpy, strcmp, strlen, strcat, ...

# Bài tập ôn tập chương 5 (GIÁO TRÌNH)





## NHẬP MÔN LẬP TRÌNH

# CHƯƠNG 7. KIỂU DỮ LIỆU

## CẤU TRÚC -STRUCT

*BM Công nghệ phần mềm –  
FIT HUFI*

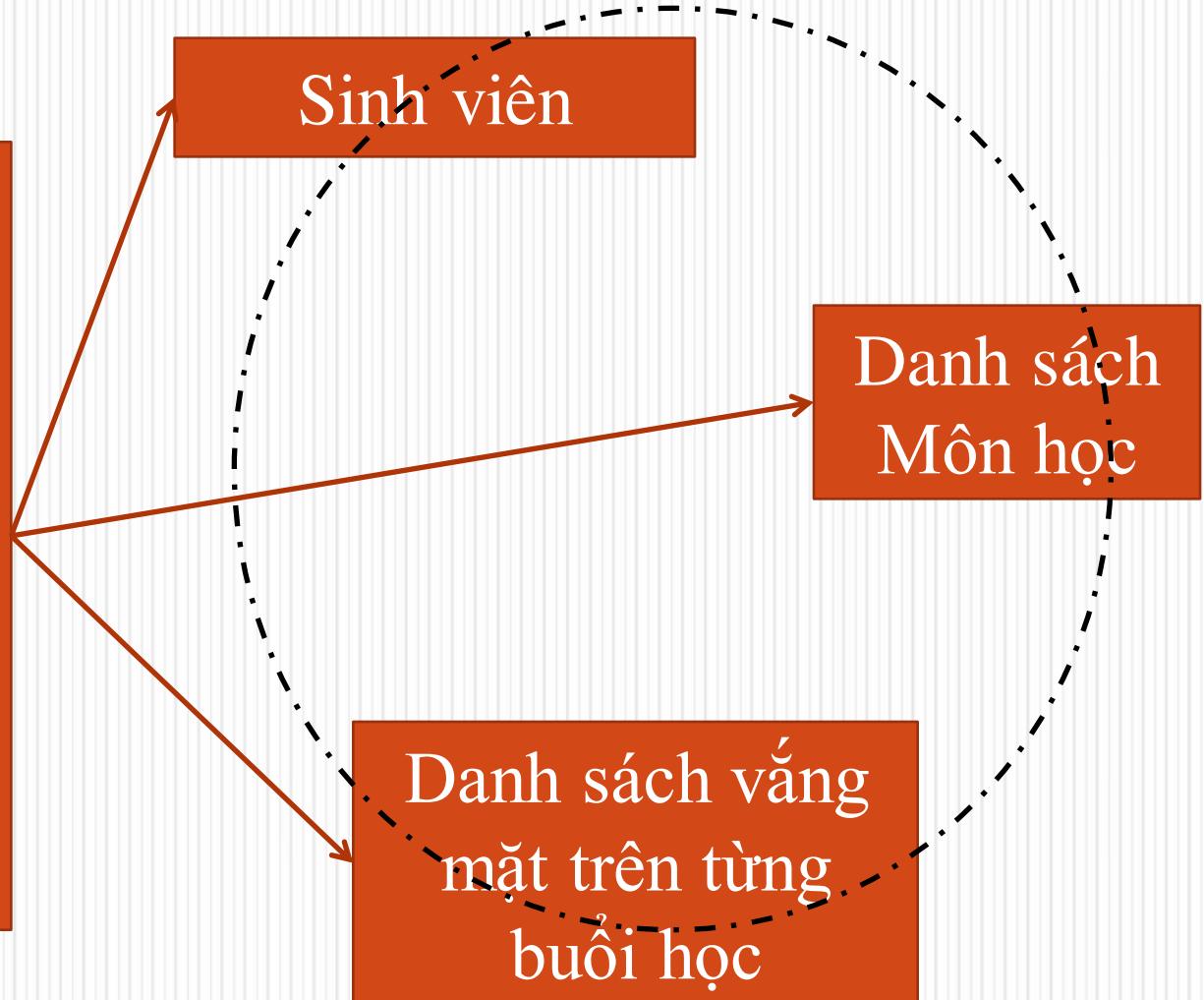
7/15/2021

# Nội dung

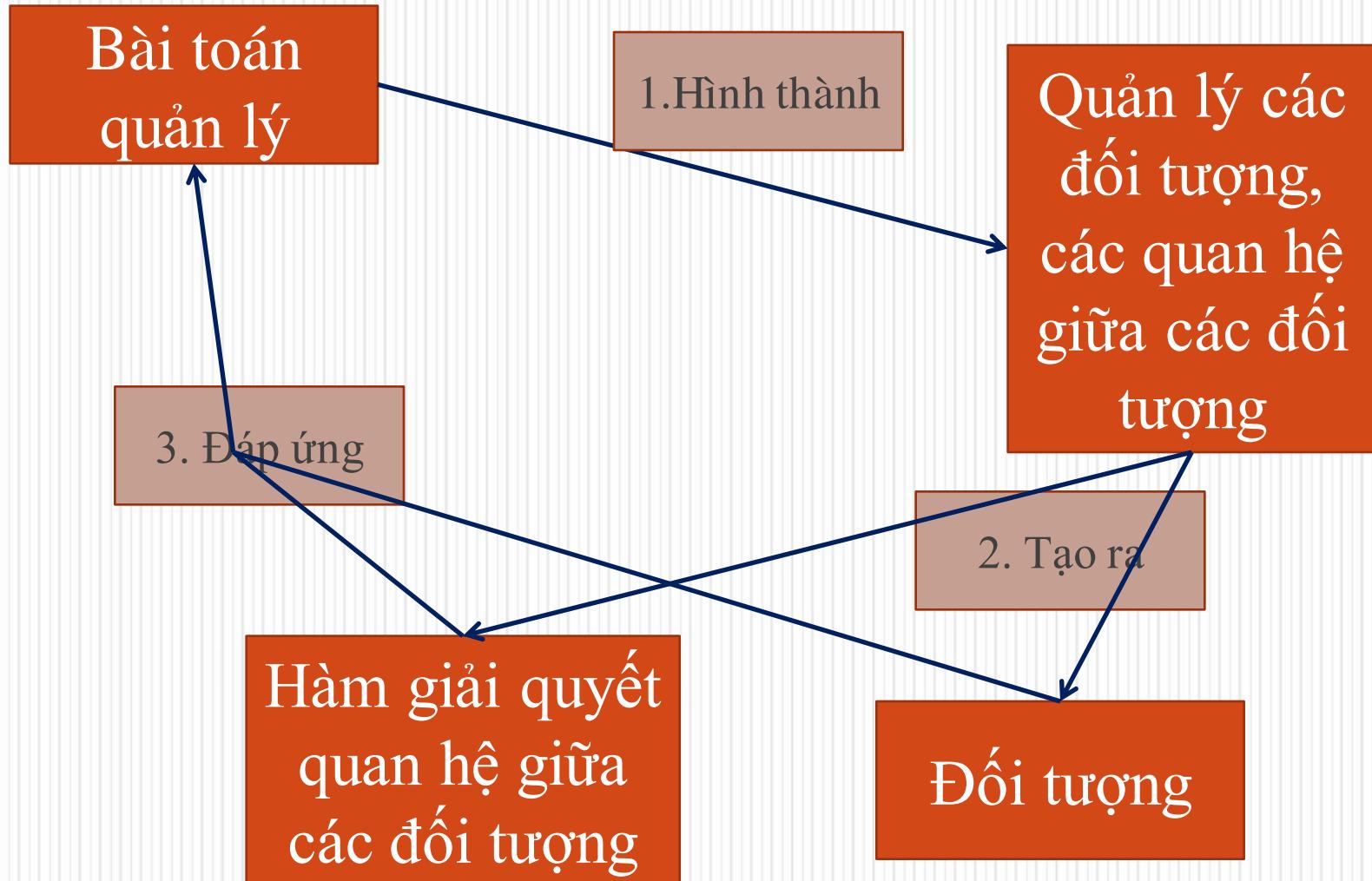
- 1. Giới thiệu**
- 2. Định nghĩa**
- 3. Cú pháp**
- 4. Định nghĩa kiểu mới**
- 5. Nguyên tắc truy cập thành phần của cấu trúc**
- 6. Mảng cấu trúc**
- 7. Một số ví dụ**
- 8. Cấu trúc lồng**
- 9. Bài tập**

# 1. Giới thiệu

Quản lý điểm danh sv trong lớp: Kiểm tra vắng mặt của sv trong 1 môn học bất kì, xếp hạng môn học dựa trên độ vắng mặt sv,...



# 1. Giới thiệu (tt)



## 2. Định nghĩa

Cấu trúc là một kiểu dữ liệu bao gồm nhiều thành phần có thể thuộc nhiều kiểu dữ liệu khác nhau. Các thành phần được truy nhập thông qua một tên.

### 3. Cú pháp

□ Cú pháp:

**struct [tên\_cấu\_trúc]**

{

**//khai báo các thành phần**

# Ví dụ

Khai báo kiểu sinh viên sau:

```
struct SinhVien {  
    char mssv[50];  
    char hoten[50];  
    int solanvang;  
};  
  
SinhVien x;
```

## 4. Định nghĩa kiểu mới

- ❑ Cú pháp 1:

**typedef <tên kiểu dữ liệu cũ> <tên kiểu dữ liệu mới>;**

- ❑ Ví dụ:

```
typedef SinhVien student;
```

## 4. Định nghĩa kiểu mới (tt)

- Cú pháp 2:

```
typedef struct <tên_cấu_trúc_1> {  
    //khai báo các thành phần  
} <tên_cấu_trúc_2>;
```

- Ví dụ:

```
typedef struct SinhVien{  
    char mssv[50];  
  
    char hoten[50];  
  
    int solanvang;  
} SV;
```

## 5. Nguyên tắc truy cập thành phần của cấu trúc

- ❑ Các thành phần của cấu trúc được truy nhập thông qua tên biến cấu trúc và tên thành phần.

*tên\_bien\_cau\_truc.tên\_thành\_phần*

- ❑ Ví dụ:

```
SinhVien x;  
x.hoten;  
x.solanvang;  
puts("Nhập họ tên:");  
gets(x.hoten);  
printf("Nhập số lan vang mat:");  
scanf("%d", x.solanvang);
```

## 6. Mảng cấu trúc

- ❑ Cú pháp:

**struct <tên cấu trúc> <tên mảng> [ <kích thước>];**

- ❑ Ví dụ 1:

```
SinhVien a[10];  
puts("Nhập mã số sinh viên:");  
gets(a[2].mssv);  
puts("Nhập họ tên:");  
gets(a[2].hoten);  
printf("Nhập số lần vang mat:");  
scanf("%d", a[2].solanvang);
```

## 7. Một số ví dụ

- ❑ Ví dụ 1: Cho 1 mảng sinh viên. Viết hàm tìm tất cả sinh viên có số lần vắng mặt trên lớp nhỏ hơn 3.

Phân tích bài toán:

- Tạo kiểu dữ liệu sinh viên.
- Viết hàm nhập dữ liệu mảng sinh viên.
- Viết hàm xuất mảng sinh viên đó.
- Viết hàm tìm tất cả sinh viên có số lần vắng mặt trên lớp nhỏ hơn 3.

## 7. Một số ví dụ (tt)

### Tạo kiểu dữ liệu

```
struct SinhVien {  
    char mssv[50];  
    char hoten[50];  
    int solanvang;  
};
```

## 7. Một số ví dụ (tt)

### □ Viết hàm nhập mảng

```
void NhapMangSV(SinhVien a[], int n) {  
    for(int i=0;i<n;i++) {  
        puts("Nhap sinh vien thu %d:\n", i);  
        puts("Nhap ma so sinh vien:");  
        gets(a[i].mssv);  
        puts("Nhap ho ten:");  
        gets(a[i].hoten);  
        printf("Nhap so lan vang mat:");  
        scanf("%d", &a[i].solanvang);  
    }  
}
```

## 7. Một số ví dụ (tt)

### □ Viết hàm xuất sinh viên

```
void XuatMangSV(SinhVien a[], int n) {  
    for(int i=0;i<n;i++) {  
        puts("Xuat sinh vien thu %d:\n", i);  
        puts("Xuat ma so sinh vien:");  
        puts(a[i].mssv);  
        puts("Xuat ho ten:");  
        puts(a[i].hoten);  
        printf("Xuat so lan vang mat:%d",  
            a[i].solanvang);  
    }  
}
```

## 7. Một số ví dụ (tt)

- Viết hàm tìm tất cả sinh viên có số lần vắng mặt trên lớp nhỏ hơn 3.

```
void TimSVvang(SinhVien a[], int n) {  
    for(int i=0;i<n;i++)  
        if(a[i].solanvang<3) {  
            puts("Xuat sinh vien thu %d:\n",i);  
            puts(a[i].mssv);  
            puts(a[i].hoten);  
            printf("Xuat so lan vang mat:%d",  
                a[i].solanvang);  
        }  
}
```

## 7. Một số ví dụ (tt)

### □ Hàm chính thực hiện bài toán

```
void main() {  
    SinhVien a[100];  
    int n;  
    //nhap n  
    NhapMangSV(a, n);  
    XuatMangSV(a, n);  
    TimSVvang(a, n);  
}
```

## 7. Một số ví dụ (tt)

- ❑ Ví dụ 2: Viết hàm sắp xếp mảng sinh viên tăng dần theo Họ tên, hay theo số lần vắng học trên lớp. ???

## 8. Cấu trúc lồng

- ❑ Cấu trúc lồng nhau là một cấu trúc có thành phần lại là một cấu trúc.
- ❑ Ví dụ:

```
struct Date{ int d,m,y; };
```

```
struct SinhVien {
```

```
    char mssv[50];
```

```
    char hoten[50];
```

```
    Date ns;
```

## 8. Cấu trúc lồng (tt)

### Ví dụ:

```
void main()
{
    SinhVien x;
    printf("Nhập ngày sinh của sinh viên
x:");
    scanf("%d%d%d", &x.ns.d, &x.ns.m,
&x.ns.y);
}
```

## 9. Bài tập

1. Hãy khai báo một cấu trúc mô tả phân số có các thông tin tử số và mẫu số. Sau đó viết chương trình thực hiện các chức năng sau:

- Nhập/ xuất phân số.
- Kiểm tra phân số mẫu phải khác 0.
- Tính cộng/trừ/nhân/chia 2 phân số.
- Tối giản phân số.

2. Hãy khai báo một cấu trúc mô tả hình tròn có các thông tin: điểm O làm tâm và bán kính r. Sau đó viết chương trình thực hiện các chức năng sau:

- Nhập, xuất thông tin một hình tròn
- Tính chu vi, diện tích hình tròn.

## 9. Bài tập(tt)

3. Viết chương trình quản lý nhân sự cho một công ty, mỗi nhân viên trong công ty gồm các thông tin sau: mã số( không có hai người trùng mã số), họ, tên, ngày sinh, nơi sinh, địa chỉ, ngày công tác, lương. Viết chương trình quản lý nhân viên với các thao tác sau:

- Thêm vào một nhân viên.
- Xem danh sách nhân viên.
- Tìm nhân viên theo mã số.
- Tìm một nhân viên theo tên.
- In ra bảng lương của các nhân viên trong công ty theo thứ tự giảm dần.
- Xóa một nhân viên.

## 9. Bài tập(tt)

4. Xây dựng cấu trúc đa thức gồm nhiều đơn thức. Viết hàm hiện thực các phép toán cộng trừ nhân chia đa thức. Và viết hàm hiện thực việc rút gọn đa thức.

