

Chương 1

Lập trình C++ cơ bản

Giáo viên

: ThS. Trần Văn Thọ

Đơn vị

: Bộ môn KTHT & MMT

Nội dung

- 1.1 Chương trình C++
- 1.2 Kiểu dữ liệu và các lệnh đơn giản
- 1.3 Lệnh điều khiển
- 1.4 Kiểu dữ liệu mảng
- 1.5 Kiểu dữ liệu cấu trúc
- 1.6 Hàm

1.1 Chương trình C++

A) Giới thiệu ngôn ngữ C++

- C++ là ngôn ngữ lập trình máy tính bậc cao, phát minh năm 1979 dựa trên ngôn ngữ lập trình C
- Nhờ ưu điểm đa năng, mạnh mẽ và hỗ trợ lập trình hướng đối tượng, C++ trở nên phổ biến và thành tiêu chuẩn quốc tế ISO 14882
 - Luôn trong top 5 ngôn ngữ phổ biến nhất thế giới

- C++ được 1 loạt ngôn ngữ bậc cao hiện đại tham khảo và kế thừa như Java, Javascript, PHP, C#, Objective-C
- C++ thường được giảng dạy trong các ngành kỹ thuật nói chung và ngành máy tính nói riêng

B. Các thành phần cơ bản của ngôn ngữ C++

- **Tập ký tự**

- Ký tự là phần tử cơ bản nhất của một ngôn ngữ, bao gồm:
 - Các ký tự (viết thường và viết hoa) của bảng chữ cái tiếng Anh.
 - Các chữ số.
 - Các dấu và ký tự đặc biệt: + - * / % = ~ ! # % ^ & () { } [] : . , ? \ ' " _ ; |

Từ khóa

- Từ khóa là từ được quy định sẵn trong ngôn ngữ và dành riêng cho các mục đích nhất định.
- Một số ví dụ từ khóa của C++
 - **char, int, float, double, struct, signed, unsigned, short, long**
 - **if, else, for, while, do, switch, case, break, continue, return**
 - **using, namespace**

Kiểu dữ liệu

- **Dữ liệu:** là đại lượng biểu diễn **thông tin** trong chương trình
 - Dữ liệu cơ bản: kí tự, số thực, số nguyên
 - Dữ liệu phức hợp: dãy số, vector, ma trận, văn bản ...
- **Kiểu dữ liệu:** là tập hợp các đại lượng dữ liệu cùng dạng biểu diễn
 - Kiểu dữ liệu **int** là tập hợp các dữ liệu dạng số nguyên
 - Kiểu dữ liệu **float** là tập hợp các dữ liệu dạng số thực

Phép toán trên kiểu dữ liệu

- Trên một kiểu dữ liệu, ngôn ngữ C++ quy định một số phép toán tương ứng
- Ví dụ: trên kiểu dữ liệu **int** có các phép toán cơ bản như phép số học, phép so sánh
 - Cộng trừ nhân chia: +, -, *, /
 - Đảo dấu: -
 - Chia lấy phần dư: %
 - So sánh lớn hơn, nhỏ hơn, bằng: >, <, ==

Hằng (const)

- **Hằng** : là đại lượng lưu dữ liệu có giá trị không đổi trong chương trình
- Ngôn ngữ C/C++ quy định có các loại hằng cơ bản sau:
 - Hằng nguyên có giá trị là một số nguyên
 - Hằng thực có giá trị là một số thực
 - Hằng ký tự: giá trị là một ký tự trong bảng mã ASCII
 - Hằng chuỗi ký tự: giá trị là một dãy ký tự

Biến (variable)

- **Biến** : là đại lượng lưu trữ dữ liệu mà giá trị có thể thay đổi được trong chương trình.
- Để tạo và sử dụng biến dữ liệu thì cần phải có **khai báo biến**.

Câu lệnh

- **Câu lệnh:** là chỉ thị biểu đạt thao tác xử lý mà máy tính cần thực hiện.
- **Chú ý:** các lệnh được ngăn cách nhau bởi dấu ; đặt ở cuối mỗi câu lệnh

Hàm (function)

- Hàm: là một đại lượng thực thi công việc của chương trình
- Bên trong hàm chứa các câu lệnh
- Khi máy tính chạy chương trình, các câu lệnh bên trong hàm sẽ được thực thi

Một số hàm toán học chuẩn của ngôn ngữ C++

Hàm	Ý nghĩa	Ký hiệu toán học
sqrt(x)	Căn bậc 2 của x	\sqrt{x}
pow(x,y)	x mũ y	x^y
exp(x)	e mũ x	e^x
log(x)	Logarithm tự nhiên (cơ số e) của x	$\ln x$
sin(x)	sin của x	$\sin x$
cos(x)	cos của x	$\cos x$
tan(x)	tang của x	$\tan x$
abs(x)	Trị tuyệt đối của x	$ x $

Định danh

- Định danh là dãy ký tự dùng để đặt tên cho một phần tử trong một chương trình như kiểu dữ liệu, biến, hằng, hàm.
- Các phần tử được đặt tên gồm có
 - Kiểu dữ liệu: có tên là các từ khóa do ngôn ngữ quy định như **int**, **char**, **float**,...
 - Biến dữ liệu
 - Hằng dữ liệu
 - Hàm

Quy tắc đặt tên

- Tên dài không quá 255 kí tự
- Chỉ bao gồm chữ cái, chữ số và dấu _
- Phải bắt đầu bằng chữ cái hoặc dấu _
- Tên không được trùng với từ khóa
- Trong cùng một phạm vi của chương trình, không được phép có đại lượng trùng tên.
- **Chú ý:** C++ phân biệt chữ hoa và chữ thường.

Cách thức đặt định danh

- Hằng số: chữ hoa
- Các biến, hàm, kiểu dữ liệu : Bằng chữ thường.
- Nếu tên gồm nhiều từ thì ta nên phân cách các từ bằng dấu gạch dưới.

Lời chú thích (comment)

- Là những giải thích ngắn gọn về chương trình hoặc 1 phần của chương trình
- Biểu diễn của chú thích
 - Chú thích trên một dòng: *// Dòng chú thích*
 - Chú thích bằng cả một đoạn

```
/* Dòng chú thích 1  
    Dòng chú thích 2  
    Dòng chú thích 3 */
```

C. Cấu trúc của 1 chương trình C++ cơ bản

- Chương trình đầu tiên Hello World

```
#include <iostream> // khai báo các hàm xuất nhập dữ liệu chuẩn  
using namespace std; // sử dụng tên hàm chuẩn
```

```
int main() // hàm main – hàm thực thi chương trình chính  
{  
    cout << "Hello World!"; // lệnh cout để xuất dữ liệu ra màn hình  
    return 0; // kết thúc hàm main và kết quả trả lại là zero  
}
```

C. Cấu trúc của 1 chương trình C++ cơ bản

Khai báo tập tiêu đề của thư viện với chỉ thị #include
Khai báo kiểu dữ liệu (người lập trình tự tạo)
Khai báo hàm nguyên mẫu
Khai báo các hằng và biến dữ liệu toàn cục
Chương trình chính – hàm main() <pre>int main() { [dãy câu lệnh] return 0; }</pre>
Nội dung câu lệnh của các hàm đã khai báo ở trên

C. Cấu trúc của 1 chương trình C++ cơ bản

- **Phần 1: Khai báo tập tiêu đề**
 - Thông báo cho chương trình dịch biết là chương trình có sử dụng những thư viện nào.
 - VD: `#include <iostream> // thao tác vào ra để xuất nhập dữ liệu`
`#include <math.h> // hàm toán học`
- **Phần 2: Định nghĩa các kiểu dữ liệu mới**
 - Định nghĩa các kiểu dữ liệu mới (nếu cần) dùng cho cả chương trình.
- **Phần 3: Khai báo các hàm nguyên mẫu**
 - Giúp cho chương trình dịch biết được những thông tin cơ bản của các hàm sử dụng trong chương trình.

C. Cấu trúc của 1 chương trình C++ cơ bản

- **Phần 4: Khai báo các biến và hằng dữ liệu toàn cục**

Ví dụ:

```
int a, b; // biến
```

```
int tong, hieu, tich; // biến
```

```
const float PI = 3.14; // hằng
```

- **Phần 5: Hàm main()**

- Khi thực hiện, chương trình sẽ bắt đầu bằng việc thực hiện các lệnh trong hàm **main()**.
- Trong hàm **main()** có thể có lệnh gọi tới các hàm khác.

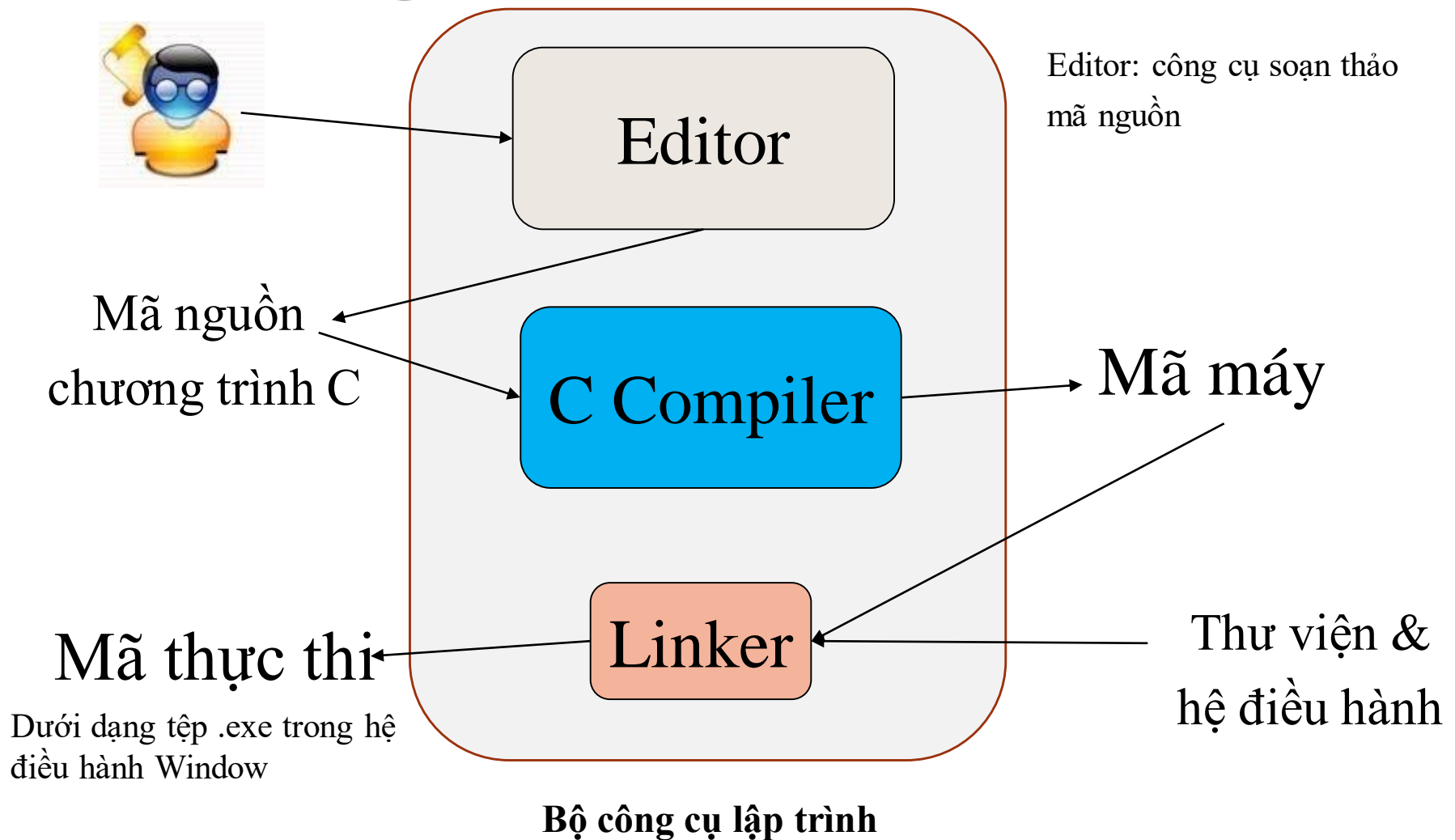
- **Phần 6: Nội dung của các hàm đã khai báo**

- Nội dung các câu lệnh thực thi bên trong từng hàm đã khai báo nguyên mẫu ở phần 3.

D) Biên dịch và liên kết chương trình

- Chương trình Hello World được gọi là mã nguồn hay **source code**
- Để máy tính “hiểu” được chương trình thì cần dịch mã nguồn sang mã máy (**machine code**) bằng trình biên dịch (**compiler**)
- Để tạo ra chương trình “chạy” được, cần liên kết mã máy và tạo ra mã thực thi (**executable code**) bằng trình liên kết (linker)

E) Bộ công cụ lập trình (IDE)



Giới thiệu bộ công cụ CodeBlock

- Bộ công cụ lập trình còn gọi là môi trường phát triển tích hợp (Integrated Development Environment – IDE)
- Bao gồm 3 công cụ chính: editor, compiler, linker và nhiều tool khác hỗ trợ công việc lập trình, phát triển ứng dụng

1.2 Kiểu dữ liệu, biểu thức và phép toán

- Kiểu dữ liệu cơ bản được thiết kế sẵn trong ngôn ngữ lập trình, gồm có:
 - Kí tự : **char, wchar_t**
 - Số nguyên: **int**
 - Số thực: **float, double**
 - logic: **bool**
- Kiểu dữ liệu dẫn xuất là kiểu mở rộng từ kiểu cơ bản
 - **short int, long int, long double**
 - **unsigned char, unsigned int**

Bảng kiểu dữ liệu cơ bản của C++

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền giá trị
unsigned char	Kí tự ASCII	1 byte	$0 \div 255$
char	Kí tự có dấu	1 byte	$-128 \div 127$
bool	Logic	1 byte	true, false
wchar_t	Kí tự Unicode	2 byte	$-32,768 \div 32,767$
unsigned int	Số nguyên không dấu	4 bytes	$0 \div 4,294,967,295$
short int	Số nguyên có dấu	2 bytes	$-32,768 \div 32,767$
int	Số nguyên có dấu	4 bytes	$-2,147,483,648 \div 2,147,483,647$
unsigned long	Số nguyên không dấu	4 bytes	$0 \div 4,294,967,295$
long	Số nguyên có dấu	4 bytes	$-2,147,483,648 \div 2,147,483,647$
float	Số thực dấu phẩy động, độ chính xác đơn	4 bytes	$\pm 3.4\text{E}-38 \div \pm 3.4\text{E}+38$
double	Số thực dấu phẩy động độ chính xác kép	8 bytes	$\pm 1.7\text{E}-308 \div \pm 1.7\text{E}+308$
long double	Số thực dấu phẩy động	10 bytes	$\pm 3.4\text{E}-4932 \div \pm 1.1\text{E}+4932$

Cách khai báo biến kiểu dữ liệu chuẩn

- Khai báo từng biến: **<tên kiểu> <tên biến>;**

```
float x;      // biến kiểu thực float
float y;      //
double z;     // biến kiểu thực double
int i;        // biến kiểu nguyên int
```

- Khai báo một loạt biến: **<tên kiểu> <danh sách tên biến>;**

```
float x,y,z;  // danh sách 3 biến kiểu float
int i,j,k;    // danh sách 3 biến kiểu int
```

- Khởi tạo giá trị ban đầu: **<tên kiểu> <tên biến>=<giá trị>;**

```
int a = 3;
float x = 5.0, y = 2.6;
```

Cách khai báo hằng dữ liệu

- Khai báo bằng chỉ thị: **`#define <tên hằng> <giá trị>`**

```
#define MAX_SINH_VIEN 60           // hằng kiểu số nguyên
#define CNTT "Cong nghe thong tin" // hằng kiểu xâu kí tự
#define DIEM_CHUAN 23.5          // hằng kiểu số thực
```

- Khai báo với từ khóa: **`const <tên kiểu> <tên hằng> = <giá trị>;`**

```
const int MAX_SINH_VIEN = 60;           // hằng kiểu số nguyên
const string CNTT = "Cong nghe thong tin"; // hằng kiểu xâu kí tự
const float DIEM_CHUAN = 23.5;          // hằng kiểu số thực
```

Các lệnh đơn giản

- **Lệnh khối**

- *Dạng lệnh:*

- {

- // danh sách câu lệnh

- }

- **Lệnh gán**

- *Dạng lệnh:* Biến = Biểu thức;

Biểu thức số học

- Toán hạng phải có kiểu số (char, int, long, float, double)
- Phép toán số học: + - * / %
- Chú ý phép chia số thực khác phép chia số nguyên

```
int i = 3, j;
```

```
float t;
```

```
j = i / 2; // j = 1
```

```
t = i / 2.0; // t = 1.5
```

Biểu thức logic

- Giá trị chỉ có **true** hoặc **false**
- Phép toán so sánh: `>` `<` `>=` `<=` `==` `!=`
- Phép toán logic: `&&` `||` `!`
- Chú ý giá trị 0 được coi tương đương giá trị **false**
 - 0 `&&` `a > 1` nhận giá trị **false**
 - 1 `||` `b < 2` nhận giá trị **true**
 - 2 `&&` `4 > 2` nhận giá trị **true**
- Được sử dụng trong các cấu trúc lệnh có điều kiện như `if`, `while`, `for`

Các phép toán viết tắt

- Phép tăng 1 đơn vị: ++

`int i = 0, k;`

`i++; // i = 1`

`k = i++; // k = 1 và i = 2`

`k = ++ i; // k = 3 và i = 3`

- Phép giảm 1 đơn vị: --

- Phép tính kết hợp phép gán

`i += 2 ; // i = i + 2`

`i *= 2 ; // i = i * 2`

Các lệnh đơn giản

- **Lệnh nhập dữ liệu từ bàn phím**
 - **cin** : là đối tượng chuẩn của C++ chuyên dùng để nhập dữ liệu cho biến từ bàn phím
 - Khi dùng cần khai báo `#include <iostream>`
 - Sử dụng không gian tên chuẩn: **using namespace std;**
 - Cú pháp cơ bản: `cin >> tên_biến;`
 - Ví dụ:
`int n;`
`cin >> n; // nhập giá trị cho biến n`
`float a, b, c;`
`cin >> a >> b >> c ; // nhập giá trị cho a, b và c`

Các lệnh đơn giản

- **Nhập xâu ký tự với hàm `getline`.**
 - Cú pháp: **`getline(cin, btx);`**
 - Sử dụng hàm xóa bộ nhớ đệm **`fflush(stdin)`** trước khi sử dụng hàm **`getline()`**
 - Ví dụ:
 `string s;`
 `fflush(stdin);`
 `getline(cin,s);`

Các lệnh đơn giản

- **Lệnh xuất dữ liệu ra màn hình**

- **cout** : là đối tượng chuẩn của C++ chuyên dùng để xuất dữ liệu ra màn hình

- Khi dùng cần khai báo: `#include <iostream>`
- Sử dụng không gian tên chuẩn: **using namespace std;**
- Cú pháp cơ bản: `cout<< biểu_thức;`
- Ví dụ:

```
int i;
```

```
float x=1.0;
```

```
cin >> i; // nhập giá trị cho biến i
```

```
cout << setw(3) << i << setw(8) << i+2 << setw(8);
```

```
cout << fixed << setprecision(4) << i * 2.0 << setw(15);
```

```
cout << scientific << left << x * 100 << endl;
```

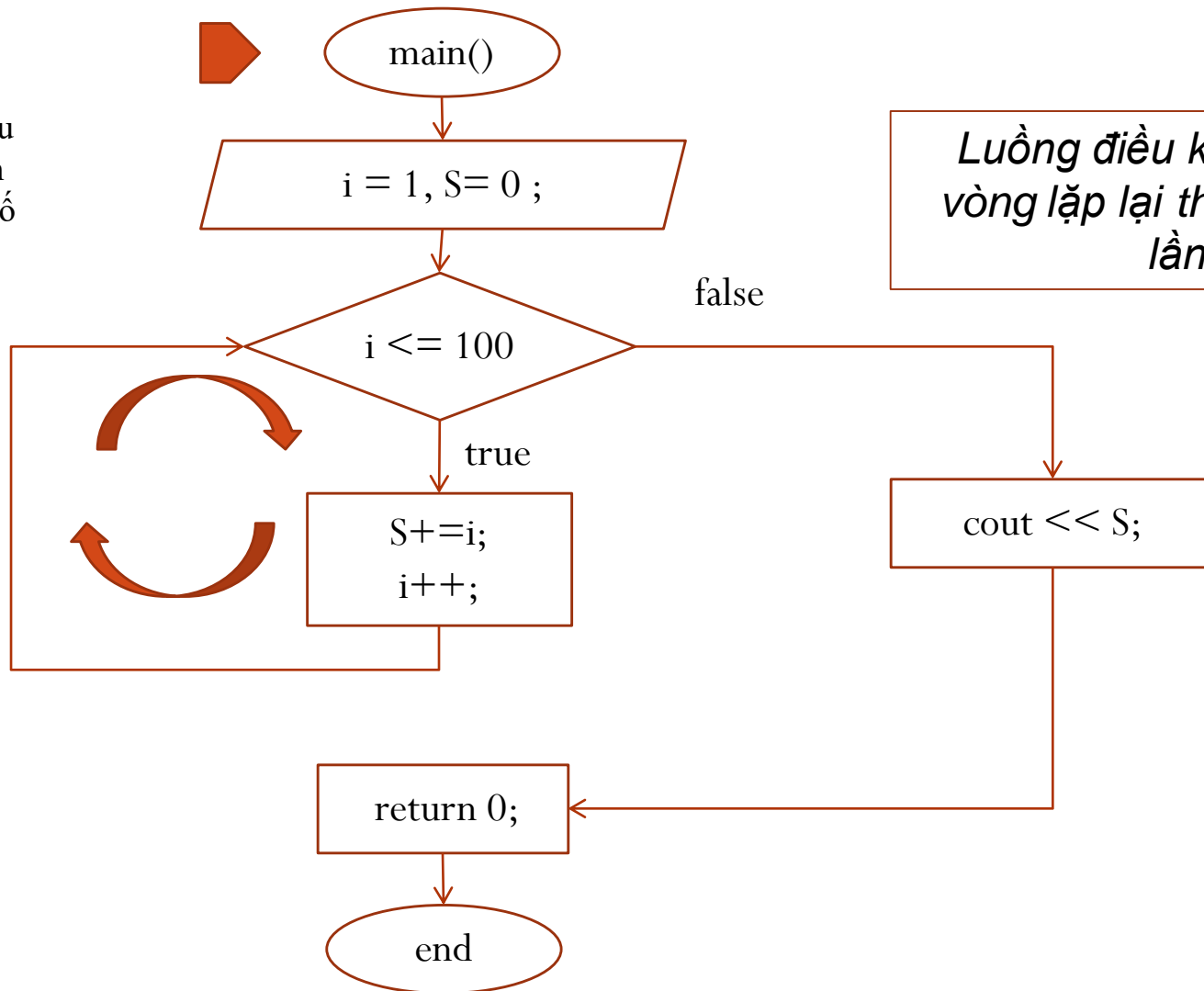
Định dạng xuất dữ liệu

- **setw(n)** đặt độ rộng dành cho dữ liệu
- **setprecision(n)** đặt độ chính xác bao nhiêu chữ số thập phân cho số thực, sử dụng cùng với
 - **fixed:** định dạng dấu chấm tĩnh
 - **scientific:** dấu chấm động và đi kèm số mũ
- **left** căn lề trái
- **right** căn lề phải (mặc định)

Bài tập minh họa

1.3) Lệnh điều khiển

Luồng điều
khiển tính
tổng 100 số
tự nhiên
đầu tiên



*Luồng điều khiển quay
vòng lặp lại thao tác 100
lần*

A) Cấu trúc lệnh chu trình lặp

- **Cấu trúc lệnh for()**
for(biểu thức 1; biểu thức 2; biểu thức 3)
lệnh;
- **Cấu trúc lệnh while()**
while(biểu thức)
lệnh;
- **Cấu trúc lệnh do...while()**
do
lệnh ;
while(biểu thức);

Ví dụ

- Cấu trúc lệnh `for()`

```
for(i=1;S=0;i<=100;i++)
```

```
    S+=i;
```

- Cấu trúc lệnh `while()`

```
i=1; S =0;
```

```
while (i<=100) {
```

```
    S+=i;
```

```
    i++;
```

```
}
```

`do while()`

```
i=1; S=0;
```

```
do
```

```
{
```

```
    S+=i;
```

```
    i++
```

```
} while(i<=100)
```


Cấu trúc chu trình lồng nhau

- Trường hợp với 3 cấu trúc lệnh for

```
for(i=0; i<3; i++) // chu trình ngoài cùng
{
    for(j=0; j<3; j++) // chu trình giữa
    {
        for(k=0; k<3; k++) // chu trình trong cùng
        {
            cout << i << "," << j << "," << k << endl;
        }
    }
}
```

Lệnh **continue** và lệnh **break**

- Lệnh **continue**;
 - Đặt bên trong chu trình
 - Tác dụng: ngay lập tức nhảy đến lần lặp tiếp theo của chu trình, bỏ qua các câu lệnh đứng sau **continue**
- Lệnh **break**;
 - Đặt bên trong chu trình
 - Tác dụng: nhảy ra khỏi chu trình, kết thúc vòng lặp vô điều kiện.

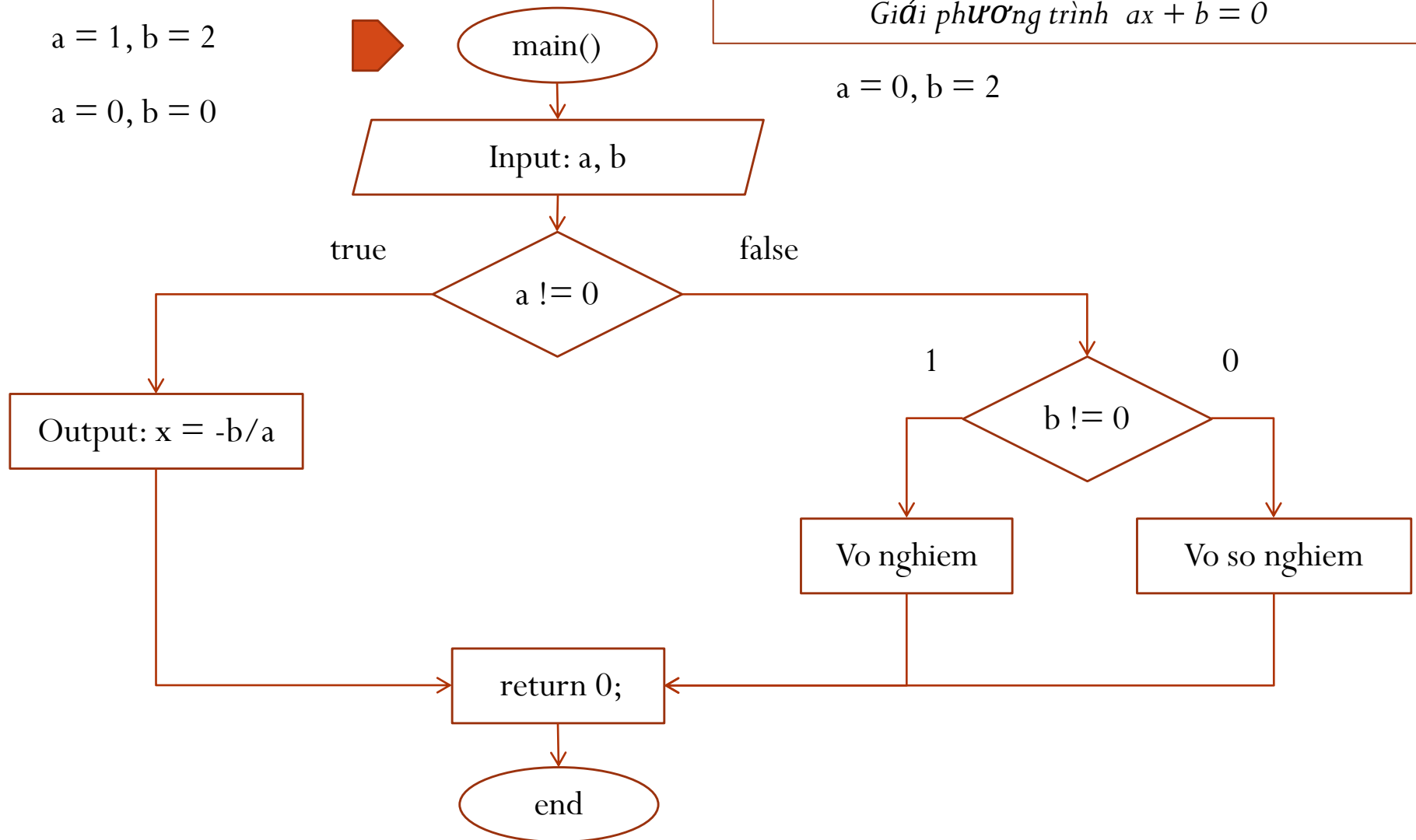
B) Cấu trúc lệnh rẽ nhánh

$a = 1, b = 2$

$a = 0, b = 0$

Giải phương trình $ax + b = 0$

$a = 0, b = 2$



B.1) Cấu trúc lệnh if else

- Cú pháp 1: **if** (biểu_thức) lệnh;
- Cú pháp 2: **if** (biểu_thức) lệnh_1;
else lệnh_2;
- Chú ý: biểu_thức phải có kiểu số nguyên hoặc logic và đặt trong ngoặc ()

Nhiều cấu trúc if else lồng nhau

- Giải phương trình bậc nhất

```
if (a!=0) {  
    cout<<"Nghiem duy nhat x ="<< -b/a<<endl;  
}  
else {  
    if (b!=0)  
        cout<<"Phuong trinh vo nghiem\n";  
    else  
        cout<<"Phuong trinh vo so nghiem\n";  
}
```

B.2) Cấu trúc lệnh switch

```
switch(biểu_thức)
{
    case giá_trị_1:    lệnh_1; break;
    case giá_trị_2:    lệnh_2; break;
    ...
    case giá_trị_n:    lệnh_n; break;

    [default: lệnh_n+1; break;]
}
```

Mệnh đề **default** có thể vắng mặt trong lệnh

Trình tự thực hiện lệnh switch case

- 1) Tính giá trị của biểu_thức (kiểu số nguyên)
- 2) Thực hiện rẽ nhánh
 - Nếu biểu_thức == giá_trị_1: chuyển đến lệnh 1
 - ...
 - Nếu biểu_thức == giá_trị_n: chuyển đến lệnh n
 - Nếu biểu_thức có giá trị khác: chuyển đến default

Ví dụ

```
diem_HP = 'B'; // Quy đổi điểm từ chữ sang số
```

```
➡ switch(diem_HP)
{
    case 'A': cout<<"Diem so: 4.0"; break;
    case 'B': cout<<"Diem so: 3.0"; break;
    case 'C': cout<<"Diem so: 2.0"; break;
    case 'D': cout<<"Diem so: 1.0"; break;
    case 'F': cout<<"Khong dat"; break;
}
<lệnh tiếp>;
```

Kết quả: Diem so: 3.0


Chú ý với lệnh break;

- Lệnh break dùng để ngắt luồng điều khiển và thoát ra ngoài cấu trúc lệnh switch
- Nếu thiếu break, thì luồng điều khiển sẽ thực hiện các lệnh tiếp bên trong cấu trúc

Ví dụ

- Thiếu lệnh break

• Diem_HP = 'B';

```
 switch(diem_HP)
{
    case 'A': cout<<"Diem so: 4.0"; break;
    case 'B': cout<<"Diem so: 3.0"; //Thiếu break
    case 'C': cout<<"Diem so: 2.0"; break;
    case 'D': cout<<"Diem so: 1.0"; break;
    case 'F': cout<<"Khong dat"; break;
}
<lệnh tiếp>;
```

Kết quả: Diem so: 3.0Diem so 2.0

Bài tập

BT1: Tìm số nguyên lẻ nhỏ nhất có 3 chữ số abc sao cho thỏa mãn điều kiện

$$abc = a^3 + b^3 + c^3$$

Cách làm

- . a, b, c là các chữ số với a : 1..9, b: 0..9 và c: 0..9
- . Dùng 3 vòng lặp for lồng nhau để duyệt qua tất cả các số **abc**
- . Kiểm tra điều kiện :
 - . Nếu c là số lẻ → bỏ qua
 - . Nếu $abc = a^3 + b^3 + c^3 \rightarrow$ tìm thấy và in ra màn hình
- . Nếu tìm thấy thì thoát khỏi các vòng lặp, kết thúc duyệt

Câu hỏi



1.4 Kiểu dữ liệu mảng

- **Mảng:** là dãy gồm nhiều phần tử nối tiếp nhau, mỗi phần tử là một **biến dữ liệu cùng kiểu**
- Mỗi phần tử của mảng được định danh qua **tên mảng** và ***chỉ số phần tử***
- Cấu trúc của mảng xác định qua số chiều

A) Mảng 1 chiều

- Khai báo gồm: kiểu dữ liệu của phần tử, tên mảng và số lượng phần tử

```
int A[10];    // A là tên mảng
```

```
char s[100];
```

```
float dayso[10];
```

- Chú ý: số lượng phần tử phải là hằng hoặc biểu thức hằng

```
const size_t SIZE = 10;
```

```
int B[SIZE];
```

```
int C[SIZE+10];
```

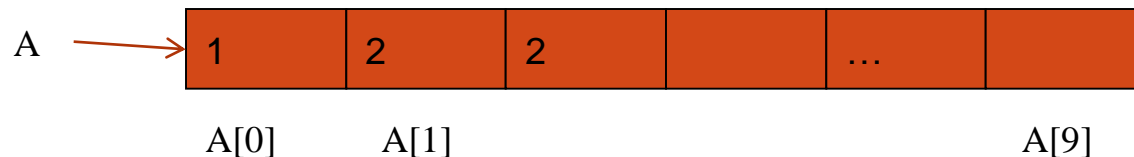
Thao tác với phần tử mảng

- Truy xuất đến phần tử qua chỉ số

$i = A[0] = 1;$

$A[i] = A[i+1] = 2$

- Chú ý: chỉ số mảng bắt đầu từ 0
- Chỉ số phải là số nguyên không âm và nhỏ hơn số lượng phần tử: $0 \leq i < n$ (n là số phần tử)



B) Mảng 2 chiều

- Khai báo gồm: kiểu dữ liệu, tên mảng, kích thước mảng theo từng chiều

```
int A[10][10]; // số phần tử = 10 x 10
```

- Mảng 2 chiều được sử dụng để lưu dữ liệu
 - Dạng bảng 2 chiều (hàng và cột)
 - Dạng ma trận
 - Nhiều mảng 1 chiều cùng kích thước
 - Dạng lưới chia ô trên mặt phẳng 2 chiều

1 2 4
 5 6 8
 2 3 4

Ma trận

	Thắng	Hòa	Thua	Điểm
AC Milan	1	1	1	4
MU	3	0	0	9
Ajax	1	1	1	4
Auxerre	0	0	3	0

Bảng

↑	↓	→
→	↑	↗
↓	→	?

Lưới 2 chiều

Cách nhập và in toàn bộ phần tử

- Sử dụng 2 cấu trúc for lồng nhau

- Nhập dữ liệu phần tử

```
for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    {
        cout << "A[" << i << "," << j << "]=";
        cin >> A[i][j];
    }
```

- In giá trị phần tử

```
for(i=0;i<n;i++)
{
    for(j=0;j<m;j++)
        cout << setw(5) << A[i][j];
    cout << endl;; // xuống dòng
}
```

Kiểu chuỗi ký tự cơ bản

- Chuỗi ký tự cơ bản là mảng 1 chiều với phần tử là ký tự

```
char ten[100] = "Tran Van Tuan";
```

```
char lop[40] = "63IT1";
```

- In chuỗi ký tự

```
cout << ten << " lop" << lop << endl;
```

- Nhập chuỗi ký tự

```
cin.getline(ten, 100); // độ dài tối đa 100 ký tự
```

- Truy nhập ký tự

```
cout << ten[0] ;// ký tự T
```

Bài tập minh họa

1.5) Kiểu dữ liệu cấu trúc

- Kiểu struct: biểu diễn dữ liệu **phức hợp** gồm nhiều trường dữ liệu, do người lập trình tự chọn.
- Kiểu struct dùng để mô tả các dạng dữ liệu mà mỗi đại lượng dữ liệu gồm nhiều thành phần thông tin

Ví dụ: mỗi 1 hồ sơ sinh viên gồm các thành phần:

- Họ tên, giới tính, ngày sinh, quê quán, địa chỉ, số CMTND, mã số SV, lớp, năm nhập trường ...

1.5) Kiểu dữ liệu cấu trúc

- Struct giúp tổ chức, liên kết các thành phần dữ liệu liên quan vào 1 đại lượng dữ liệu chung
- Không dùng struct, vẫn có thể biểu diễn được dữ liệu nhưng sẽ rời rạc và không có tính cấu trúc

VD: Ma trận thường được biểu diễn với các dữ liệu sau

```
unsigned int  n,m; // số hàng và cột  
int  A[10][10]; // các phần tử
```

Tuy nhiên, các dữ liệu này đơn thuần là 2 số nguyên và 1 mảng 2 chiều độc lập, không có mối liên kết trong cùng 1 đại lượng dữ liệu chung là ma trận

Khai báo (tạo) kiểu struct

- Sử dụng từ khóa struct, tên kiểu tự chọn và khai báo các thành phần dữ liệu
- Cú pháp định nghĩa một kiểu cấu trúc:

```
struct <tên cấu trúc>
{
    <khai báo trường 1>;
    ...
    <khai báo trường n>;
};
```

Ví dụ

- Định nghĩa một “mục” trong danh bạ điện thoại

```
struct  phone_entry
{
    string name;    // tên
    string address; // địa chỉ
    long  home;    // số dt cố định
    long  mobile;  // số dt di động
};
```


Khai báo biến kiểu cấu trúc

- Cùng một lúc với định nghĩa cấu trúc

```
struct <tên cấu trúc>
{
    ...
} <danh sách tên biến> ;
```

- Ví dụ

```
struct phone_entry // định nghĩa kiểu cấu trúc
{
    string name; // tên
    string address; // địa chỉ
    long home; // số dt cố định
    long mobile; // số dt di động
} entry1, entry2, entry3 ; // khai báo biến
```

Khai báo biến cấu trúc

- Khai báo sau khi đã định nghĩa kiểu cấu trúc

```
struct <tên cấu trúc>
{
    ...
};
...
```

```
struct <tên cấu trúc> <danh sách biến>;
```

- Ví dụ

```
struct phone_entry // định nghĩa
{
    string name; // tên
    string address; // địa chỉ
    long home; // số dt cố định
    long mobile; // số dt di động
};
```

```
struct phone_entry entry1, entry2, entry3; // khai báo
```

Truy nhập trường dữ liệu của biến cấu trúc

- Cú pháp truy nhập: <tên biến>.<tên trường>
- Ví dụ nhập dữ liệu cho biến cấu trúc

```
struct phone_entry  entry1;  
getline(cin, entry1.name); // nhập tên  
getline(cin, entry1.address); // nhập địa chỉ  
entry1.home = 36280158; // gán số đt  
entry1.mobile = 913514588; // gán số đt
```

Định nghĩa tên kiểu dữ liệu với từ khóa typedef

- Người lập trình có thể đặt tên mới cho một kiểu dữ liệu tùy ý
- Cú pháp
typedef <tên kiểu đã có> <tên kiểu mới>;

```
typedef struct MaTran  
{  
    unsigned int n;  
    unsigned int m;  
    int E[10][10];  
}MaTranSoNguyen;
```

```
// khai báo với tên kiểu mới  
MaTranSoNguyen mA,mB,mC;
```

Bài tập minh họa

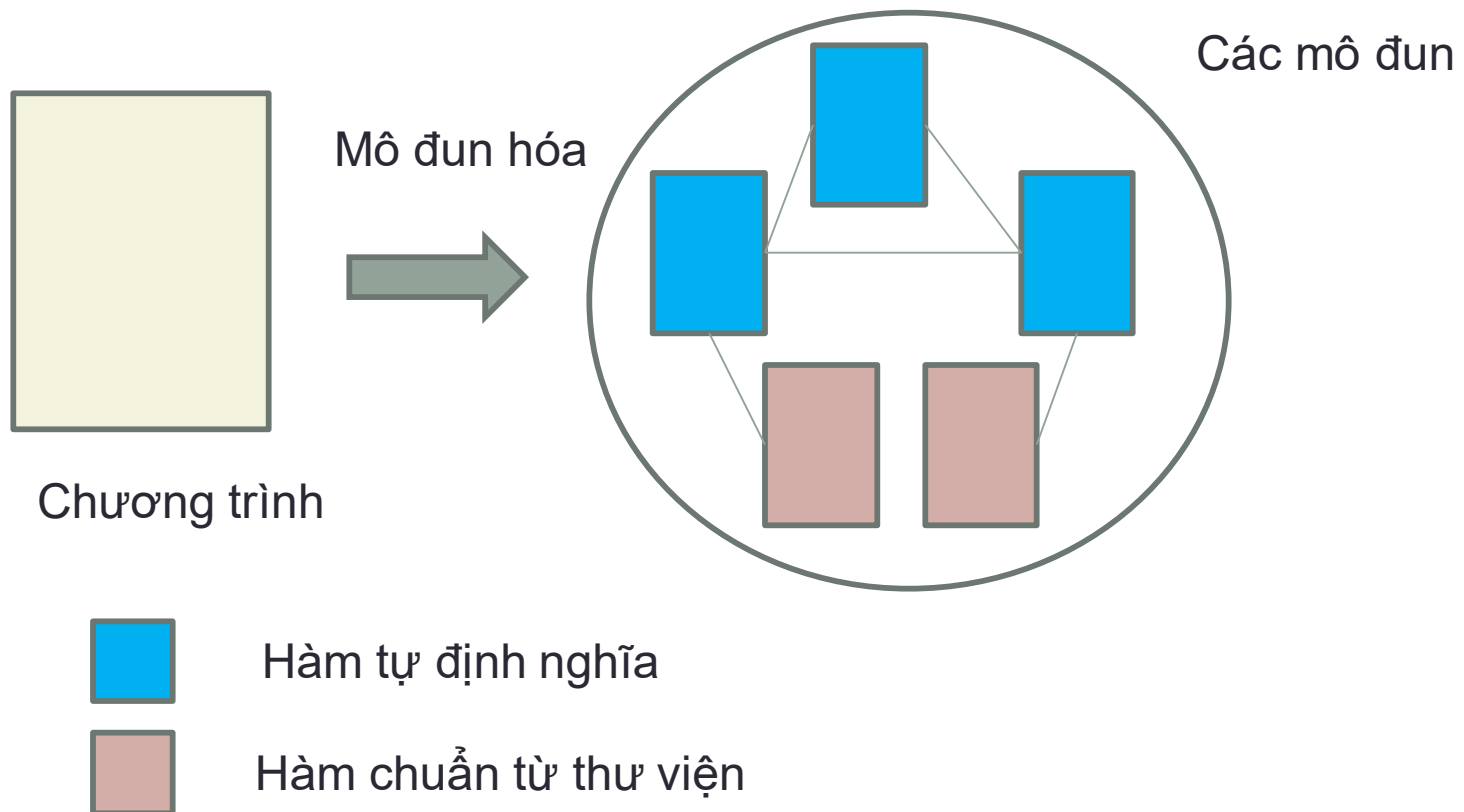
- Kết quả thi học kỳ của một lớp sinh viên được cho trong bảng sau:

STT	Họ và tên	Môn 1	Môn 2	Môn m
1					
2					
.....					
n					

- Viết chương trình để:
 - Nhập từ bàn phím số sinh viên (n), số môn thi (m), họ tên SV và kết quả thi của lớp.
 - Tính điểm trung bình của từng SV

1.6) Hàm

- Chương trình C++ nói chung là kết hợp của
 - Hàm lập trình viên tự viết (user-defined function)
 - Hàm chuẩn trong thư viện của C++



1.6.1 Cách tạo HÀM C++

- Khai báo **nguyên mẫu hàm** (function prototype)
- Lập trình phần xử lý bên trong hàm (function definition)

A) Khai báo nguyên mẫu

- Cú pháp:

kiểu_dl_trả_về tên_hàm(danh_sách_tham_số);

Ví dụ: nguyên mẫu hàm giá trị max của 2 số nguyên x, y

Cách khai báo 1: int max(int, int);

Cách khai báo 2: int max(int x, int y);

Tiếp

- Chú ý:
 - Danh sách tham số có thể trống (không tham số)
 - Kiểu trả về của hàm có thể là **void** (hàm không trả lại giá trị nào)

```
void printHelloWorld();
```

B) Lập trình xử lý bên trong hàm (định nghĩa hàm)

- Cú pháp

```
<kiểu_dl> <tên_hàm>(<ds_tham_số_hình_thức>)
```

```
{ // phần thân hàm (function body)
```

```
    Khai_báo_biến_dl_cục_bộ;
```

```
    Dãy_câu_lệnh_xử_lý;
```

```
}
```


- Biến cục bộ là biến dữ liệu được sử dụng bên trong hàm
- Dãy câu lệnh thực hiện thao tác xử lý, tính toán của hàm

Chương trình minh họa


- Hàm tính tổng bình phương của 2 số thực

```
float tong_binh_phuong(float a, float b)
{
    float tong; // biến cục bộ
    tong = a*a + b*b;
    return tong;
}
```

Danh sách
tham số



Trả về kết quả



Tiếp

- *Chú ý:* lệnh return
 - Khi thực hiện lệnh return, hàm sẽ kết thúc và trả lại giá trị biểu thức đặt sau return
 - Nếu kiểu của hàm là void, sau return để trống:
return ;

Chương trình ví dụ

- Lập hàm tính giá trị lớn nhất của một dãy số nguyên đầu vào A[]
 - Xử lý: Duyệt dãy và tìm giá trị lớn nhất
 - Kiểu trả về: int
 - Tham số đầu vào
 - Mảng A phần tử nguyên
 - Số phần tử n
 - Giá trị trả về: phần tử lớn nhất trong mảng A

1.6.3 Lời gọi hàm

- Sau khi tạo hàm, để sử dụng cần phải **gọi hàm (call function)**
- Lời gọi hàm (function call) bao gồm
 - Tên hàm
 - Các tham số **truyền cho hàm**
- Ví dụ

```
int x, y, tong;
```

```
x = y = 5; // x và y là 2 tham số truyền cho hàm
```

```
tong = tong_binh_phuong(x, y);
```



Tên hàm

Tham số thực sự

Tham số hình thức và tham số thực sự

Tham số hình thức

- Đặt trong định nghĩa hàm

```
// a và b là tham số hình thức
float max(float a, float b)
{
    return a > b ? a : b ;
}
```

Tham số thực sự

- Đặt trong lời gọi hàm

```
// x và 5.5 là tham số thực
sự
int m, x = 7.7;
m = max(x, 5.5);
```

Chú ý

- Số lượng tham số thực sự = số lượng tham số hình thức
- Kiểu dữ liệu của tham số thực sự phải **tương thích** với kiểu của tham số hình thức

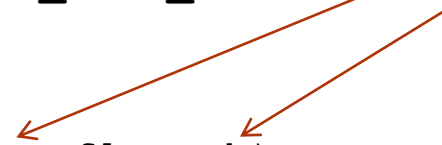
`m = max(&x, 5.5) // sai vì sử dụng tham số thực sự là địa chỉ`

B) Truyền tham số

- Truyền tham số là truyền tham số thực sự vào bên trong phần thân hàm thông qua tham số hình thức

```
int x = 3;  
tong = tong_binh_phuong(2, x);
```

```
float tong_binh_phuong(float a, float b)  
{  
    float tong; // biến cục bộ  
    tong = a*a + b*b;  
    return tong;  
}
```



- Có 2 kiểu truyền tham số thường sử dụng

Các kiểu truyền tham số

Truyền giá trị

- **Tham số hình thức là bản sao giá trị của tham số thực sự**
- Kết quả: thay đổi tham số hình thức không làm ảnh hưởng đến tham số thực sự

Truyền tham chiếu

- Tham số hình thức là một định danh khác (bí danh) của chính tham số thực sự
- Kết quả: thay đổi giá trị của tham số hình thức đồng nghĩa với thay đổi giá trị của tham số thực sự

Chương trình minh họa

- Cho số nguyên a và b, hãy viết một hàm hoán đổi giá trị giữa a và b (hàm swap)
- Áp dụng 2 kiểu truyền tham số cho hàm nói trên

Minh họa truyền tham trị

```
void swap(int a, int b)
{
    int temp;    // biến cục bộ
    temp = a;
    a = b;
    b = temp;
}
int main()
{
    int n,m;
    m = 2; n = 3;
    swap(m,n);
    cout<<"m="<<m<<" n="<<n; //m=2, n=3
    return 0;
}
```

CHÚ Ý: Trong C, C++, tất cả tham số được truyền theo tham trị

Sử dụng tham chiếu thay thế cho truyền tham biến

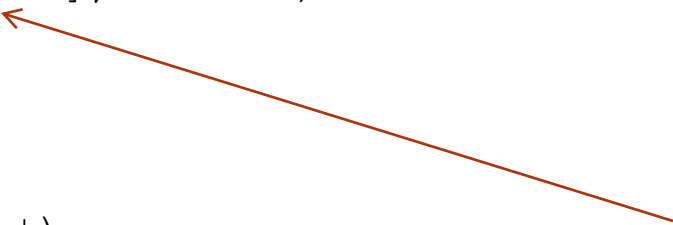
```
void swap(int &a, int &b)
{
    int n=0;
    int &temp = n;    //bien temp la tham chieu den bien n
    temp = a;
    a = b;
    b = temp;
}
int main()
{
    int n,m;
    m = 2; n = 3;
    swap(m,n);
    cout<<"m="<<m<<" n="<<n; //m=3, n=2
    return 0;
}
```

C) Tham số kiểu mảng

- Có thể đặt mảng 1 chiều, mảng 2 chiều làm tham số hình thức của hàm

```
void    max(int  A[100],  int n)
{
    int max = A[0];
    int i;
    for(i=1; i<n; i++)
        if(max<A[i])    max = A[i];
    return max;
}
```

tham số
hình thức
là mảng 1
chiều



Chú ý

- Tham số thực sự là **địa chỉ của mảng có số chiều và kích thước tương ứng** với tham số hình thức

```
int B[100];  
// Nhập dữ liệu cho mảng B  
  
...  
// Tìm phần tử lớn nhất trong m phần tử của B  
int max_value = max(B, m);
```

Chương trình minh họa

- Hàm void sapxep(int A[], int n) thực hiện việc sắp xếp một dãy số (mảng 1 chiều) theo thứ tự tăng dần.
- Tham số:
 - A là mảng 1 chiều cần sắp xếp
 - n là số phần tử trong mảng
- Cũng có thể truyền tham số dạng con trỏ như sau
 - void sapxep(int* A, int n)
 - A là con trỏ, trỏ đến mảng 1 chiều cần sắp xếp

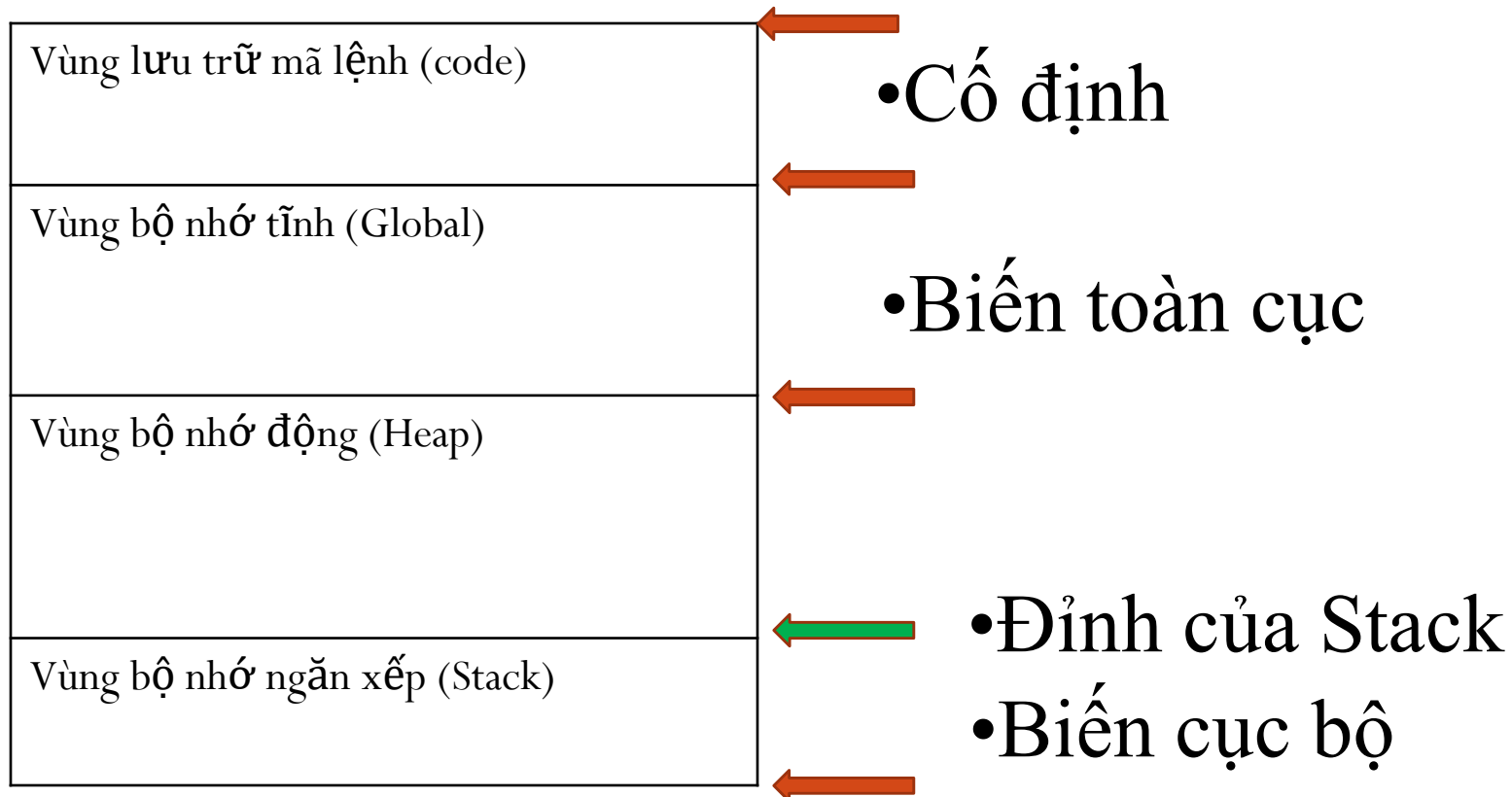
1.6.4 Phạm vi sử dụng của biến

- **Phạm vi** (scope) của một biến dữ liệu: là phần chương trình trong đó biến có tồn tại và có thể sử dụng để lưu trữ giá trị
- Tùy theo phạm vi, một biến sẽ có **thời gian tồn tại và cấp lưu trữ**
- **Thời gian tồn tại** là khoảng từ lúc biến được tạo ra trong bộ nhớ cho đến khi được giải phóng khỏi bộ nhớ (không còn tồn tại)
- **Cấp lưu trữ** là : cấp lưu trữ động hay tĩnh

Biến cục bộ và toàn cục

- Biến cục bộ là biến khai báo trong thân một hàm
- Phạm vi là bên trong hàm
- Tồn tại trong khoảng thời gian thực thi hàm
- Lưu trữ động mỗi khi hàm được gọi
- Biến toàn cục là biến khai báo bên ngoài, không thuộc bất cứ hàm nào
- Phạm vi là trong toàn bộ chương trình
- Tồn tại trong suốt thời gian thực hiện chương trình
- Lưu trữ tĩnh ngay từ khi khởi động chương trình

Các vùng trong bộ nhớ của 1 chương trình máy tính



Biến cục bộ

- Nếu khai báo biến cục bộ nằm trong một khối lệnh thì biến chỉ có ý nghĩa trong phạm vi khối lệnh đó
- Khi thực thi xong khối lệnh thì biến được giải phóng và không còn tồn tại

Khối
lệnh

```
int i; // biến i ngoài khối
for(i=0;i<10;i++)
{ // khối lệnh
    int x // biến x nằm trong khối, cấp lưu trữ động
    x = i*2;
    cout<<setw(4)<<x;
}
x = 3; // Sai, báo lỗi
// ra ngoài khối, biến x không còn tác dụng
```

Ví dụ biến toàn cục và biến cục bộ

```
#include <iostream>
using namespace std;
int ketqua; // biến toàn cục
float tong_binh_phuong(float a, float b)
{
    float tong; // biến cục bộ của hàm
    tong = a*a + b*b;
    return tong;
}

int main()
{
    float x = 5.5, y = 6.6; // biến cục bộ hàm main
    ketqua = tong_binh_phuong(x, y);
    cout<<"Ket qua:"<<setw(6)<<ketqua<<endl;
    return 0;
}
```

Bài tập

VD1: Cho ma trận $A_{n \times m}$. Viết chương trình để tìm cột k nhỏ nhất chứa phần tử lớn nhất của ma trận. Sau đó xóa cột k của ma trận đã cho.

•Yêu cầu:

- Viết hàm nhập ma trận.
- Viết hàm xuất ma trận.
- Viết hàm xóa cột k khỏi ma trận
- Viết hàm main để hoàn thiện yêu cầu của bài tập.