

# TỔNG QUAN VỀ C++

Khoa Công nghệ phần mềm

C++



Microsoft®

Visual Studio®

# Nội dung

- 1 Ôn tập lập trình C
- 2 Phong cách lập trình
- 3 Lịch sử ngôn ngữ lập trình
- 4 C và C++
- 5 Ngôn ngữ C++

# 1. Bài tập C

Nhập bốn số nguyên và xuất các giá trị vừa nhập.

→ Có bao nhiêu cách để thực hiện?





# 1. Bài tập C – Giải

1. Dùng 4 biến: cách dài nhất, cơ bản nhất.
2. Dùng mảng: khai báo biến gọn hơn, 1 lần thay cho nhiều lần.
3. Dùng mảng và 2 vòng lặp do while: viết code nhập gọn hơn, viết 1 lần thay cho nhiều lần.
4. Dùng mảng và 2 vòng lặp for: viết code gọn hơn, vòng for viết gọn hơn vòng while.

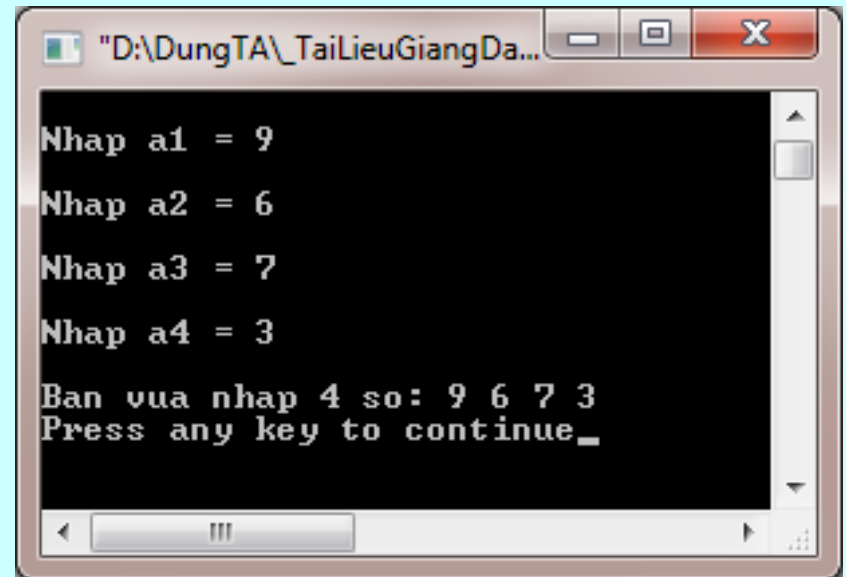
# 1. Bài tập C – Giải

5. Dùng mảng, 1 vòng lặp for gộp: viết code gọn hơn nhưng không tách riêng được 2 phần nhập xuất.
6. Dùng hàm để tách riêng phần nhập xuất: code có thể tái sử dụng nhiều lần.
7. Dùng file để nhập xuất từ file thay cho việc nhập bằng bàn phím và xuất ra màn hình.

# 1. Bài tập C – Giải

## ❖ Cách 1: Dùng 4 biến

```
void main(){
    int a1, a2, a3, a4;
    printf("\nNhap a1 = ");
    scanf("%d", &a1);
    printf("\nNhap a2 = ");
    scanf("%d", &a2);
    printf("\nNhap a3 = ");
    scanf("%d", &a3);
    printf("\nNhap a4 = ");
    scanf("%d", &a4);
    printf("\nBan vua nhap 4 so: %d %d %d %d\n", a1, a2, a3, a4);
}
```



# 1. Bài tập C – Giải

## ❖ Cách 2: Dùng mảng

```
void main(){
    int a[4];
    printf("\nNhap a1 = ");
    scanf("%d", &a[0]);
    printf("\nNhap a2 = ");
    scanf("%d", &a[1]);
    printf("\nNhap a3 = ");
    scanf("%d", &a[2]);
    printf("\nNhap a4 = ");
    scanf("%d", &a[3]);
    printf("\nBan nhap 4 so:%d %d %d %d\n", a[0], a[1], a[2], a[3]);
}
```



# 1. Bài tập C – Giải

❖ Cách 3: Dùng mảng và vòng lặp while

```
void main(){
    int a[4], i;
    i = 0;
    do{
        printf("\nNhap a%d = ", i);
        scanf("%d", &a[i]);
        i++;
    }while(i<4);
    i = 0;
    printf("\nBan vua nhap 4 so:");
    do{
        printf("%d ", a[i]);
        i++;
    }while(i<4);
}
```



# 1. Bài tập C – Giải

## ❖ Cách 4: Dùng mảng và vòng lặp for

```
void main()
{
    int a[4], i;
    for (i=0; i<4; i++){
        printf("\nNhap a%d = ", i);
        scanf("%d", &a[i]);
    }
    printf("\nBan vua nhap 4 so:");
    for (i=0; i<4; i++){
        printf("%d ", a[i]);
    }
}
```

# 1. Bài tập C – Giải

## ❖ Cách 5: Dùng mảng và vòng lặp for gộp

```
void main()
{
    int a[4], i;
    for (i=0; i<4; i++)
    {
        printf("\nNhap a%d = ", i);
        scanf("%d", &a[i]);
        printf("%d ", a[i]);
    }
}
```

# 1. Bài tập C – Giải

## ❖ Cách 6: Dùng hàm

```
void Nhap(int []);  
void Xuat(int []);  
  
void main()  
{  
    int a[4];  
    Nhap(a);  
    Xuat(a);  
}
```

```
void Nhap(int a[])  
{  
    int i;  
    for(i=0; i<4; i++)  
    {  
        printf("Nhap a[%d] = ", i);  
        scanf("%d", &a[i]);  
    }  
}
```

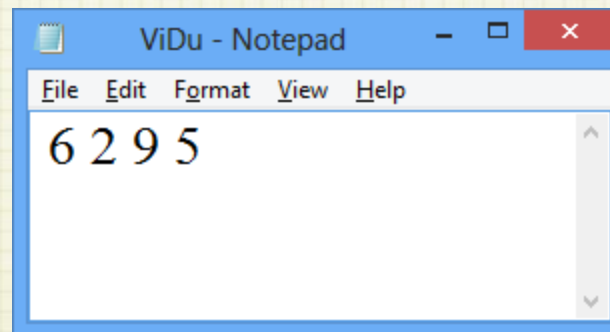
```
void Xuat(int a[])  
{  
    int i;  
    printf("Cac phan tu cua mang: ");  
    for(i=0; i<4; i++)  
    {  
        printf("%3d", a[i]);  
    }  
}
```

# 1. Bài tập C – Giải

## ❖ Cách 7: Dùng file

```
void Nhap(int a[], char *f)
{
    int i;
    FILE *fp;
    fp = fopen(f, "r");
    for(i=0; i<4; i++)
    {
        fscanf(fp, "%d", &a[i]);
    }
}
```

```
void Xuat(int a[], char *f)
{
    int i;
    FILE *fp;
    fp = fopen(f, "w");
    for(i=0; i<4; i++)
    {
        fprintf(fp, "%d", a[i]);
    }
}
```



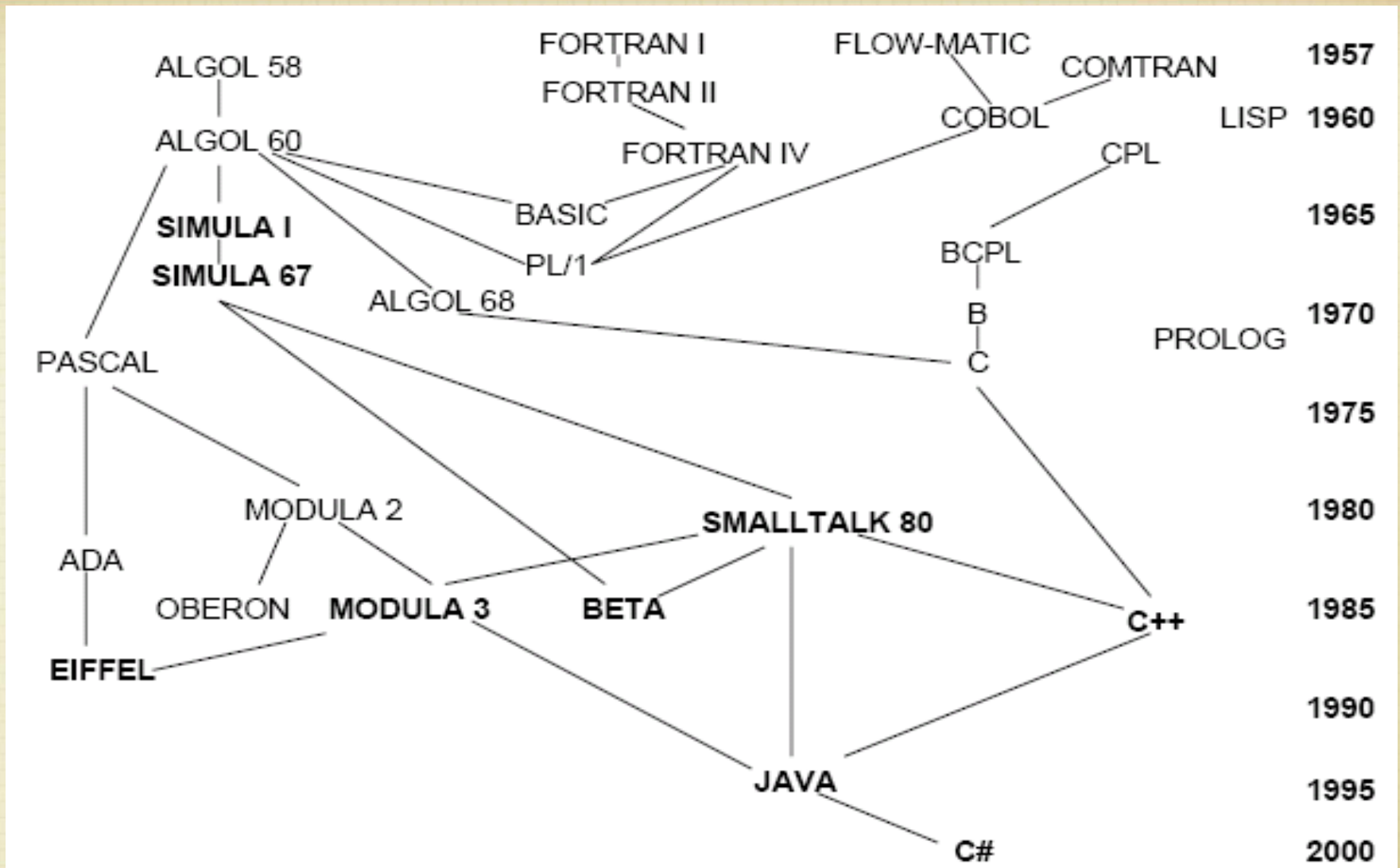


## 2. Phong cách lập trình

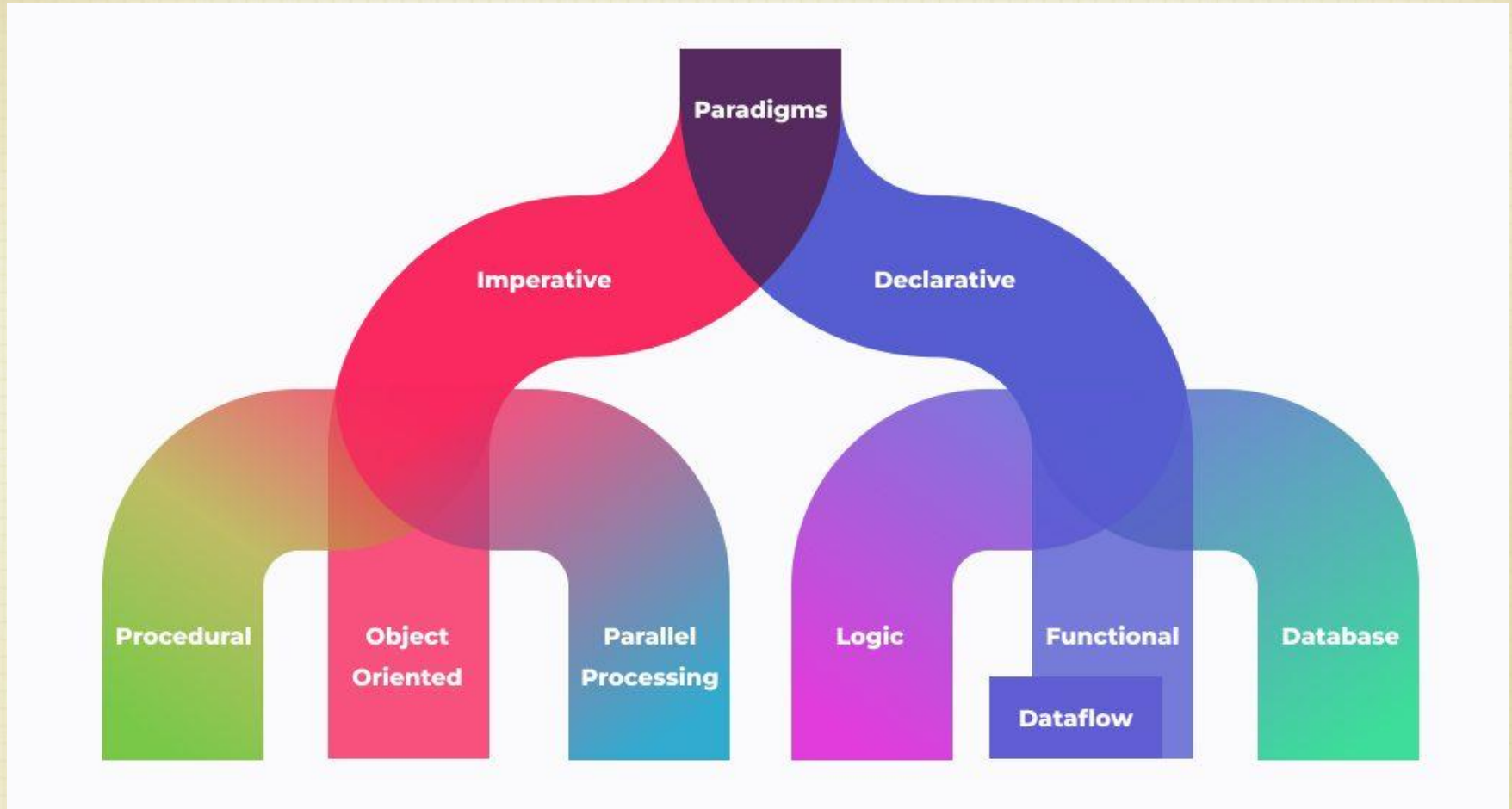
- Quy tắc đặt tên biến, tên hàm,...
- Quy tắc xuống hàng
- Sử dụng Tab và { }
- Khai báo nguyên mẫu của hàm (prototype):

Nếu hàm được xây dựng sau lời gọi hàm thì bắt buộc phải khai báo nguyên mẫu hàm trước lời gọi hàm.

# 3. Lịch sử ngôn ngữ lập trình



# 3. Lịch sử ngôn ngữ lập trình



# 4. C và C++

- ❖ Mở rộng của **C**: mọi khả năng, mọi khái niệm trong **C** đều dùng được trong **C++**
- ❖ C: ngôn ngữ lập trình cấu trúc.
- ❖ C++: ngôn ngữ lập trình hướng đối tượng.
- ❖ C++ là ngôn ngữ lai: cho phép tổ chức chương trình theo các lớp và các hàm.

Các ngôn ngữ thuần túy hướng đối tượng (như SMALLTALK) chỉ hỗ trợ các khái niệm về lớp.



## Lập trình cấu trúc

Tổ chức phân chia chương trình thành các hàm/thủ tục.

-> Chương trình dùng để xử lý dữ liệu nhưng lại tách rời dữ liệu.

Việc trao đổi dữ liệu giữa các hàm thực hiện thông qua các đối số của hàm và các biến toàn cục.

## Lập trình hướng đối tượng

- Tổ chức chương trình thành các lớp.
- Lớp bao gồm cả dữ liệu và các phương thức xử lý dữ liệu.
- Có thể xem Lớp là sự mở rộng của Struct trong C bằng cách thêm vào các phương thức.
- Dùng tên Lớp để khai báo biến kiểu Lớp hay còn gọi là đối tượng.

- Một chương trình hướng đối tượng sẽ bao gồm các Lớp có quan hệ với nhau.
- Phân tích thiết kế chương trình theo phương pháp hướng đối tượng nhằm thiết kế xây dựng các Lớp/xác định các Lớp để mô tả các thực thể của bài toán.

# 4. C và C++ (tt)

	C++
Chú thích	<pre>/* Ghi chú trên nhiều dòng*/ // Ghi chú trên một dòng</pre>
Ép kiểu	<pre>(Kiểu)Biểu_thức -&gt; (float)i Kiểu(Biểu_thức) -&gt; float(i+1)</pre>
Khai báo	Các lệnh khai báo biến có thể đặt bất kỳ chỗ nào trong chương trình <u>trước</u> khi biến được sử dụng.
Hằng	<ul style="list-style-type: none"><li>Dùng từ khóa <b>const</b> đặt trước một khai báo có khởi gán giá trị và <b>không được phép thay đổi giá trị của hằng sau khi đã được khai báo</b>. VD: <b>const</b> int MAXSIZE = 1000 -&gt; int a[MAXSIZE];</li><li>Có thể dùng hàm để khởi gán giá trị cho hằng khi khai báo hằng. VD: <b>const</b> DIEM gmh = {getmaxx()/2,getmaxy()/2,WHITE};</li></ul>

# 5. Ngôn ngữ C++

1

**Nhập xuất với C++**

2

**Toán tử phạm vi**

3

**Các kiểu dữ liệu của C++**

4

**Cấp phát bộ nhớ**

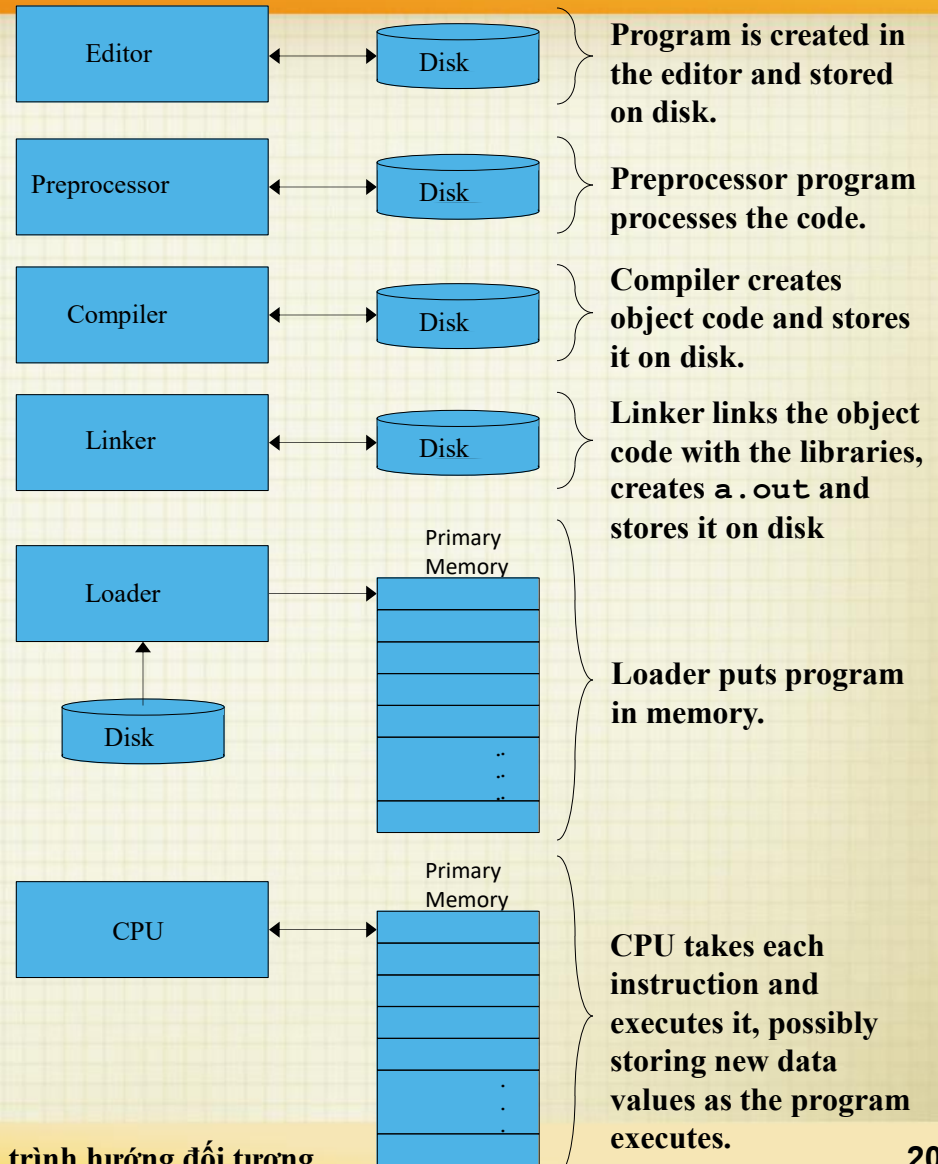
5

**Hàm trong C++**

# Môi trường của C++

## ❖ Biên dịch và thực thi chương trình C++:

- Edit
- Preprocess
- Compile
- Link
- Load
- Execute





# 5.1 Nhập xuất với C++

❖ `cin >> biến ... >> biến`

-> Nhập các giá trị từ bàn phím và gán cho các biến.

❖ `cout << biểu thức ... << biểu thức`

-> Đưa giá trị các biểu thức ra màn hình.

```
#include <iostream>
```

```
using namespace std;
```

# 5.1 Nhập xuất với C++ (tt)

## Lưu ý:

1) Để nhập một chuỗi không quá n ký tự:

```
char ht[50];
```

```
cin.get(ht,50,'\n');
```

```
cin.getline(ht,50);
```

Hoặc:

```
string s;
```

```
getline(cin,s); //thêm #include <string>
```

2) `cout << endl; //end line`

-> Xuống dòng và di chuyển con trỏ đến đầu dòng tiếp theo.

# 5.1 Nhập xuất với C++ (tt)

## Lưu ý: (tt)

3) `#include <iomanip>`

```
cout << setiosflags(ios::showpoint) << setprecision(p)
```

//bật cờ hiệu showpoint

//p là số chữ số sau dấu chấm thập phân

=> Có hiệu lực cho tất cả câu lệnh cout sau đó.

4) `cout << setw(w) //độ rộng tối thiểu là w vị trí`

=> Chỉ có hiệu lực cho 1 giá trị được xuất gần nhất.



# Ví dụ 1

```
1 // Fig. 1.2: fig01_02.cpp
2 // A first program in C++.
3 #include <iostream>
4 using namespace std;
5 // function main begins program execution
6 int main()
7 {
8     cout << "Welcome to C++!\n";
9
10    return 0; // indicate that p
11
12 } // end function main
```

Function **main** returns an integer value.

Function **main** appears once in every C++ program.

Left brace { begins function body.

Statements end with a semicolon ;.

Corresponding right brace ends function body.

Stream insertion operator.

Name **cout** belongs to namespace **std**.

Keyword **return** is one of several means to exit function; value **0** indicates program terminated successfully.

Welcome to C++!



# Ví dụ 2

```
1 #include <iostream>
2 using namespace std;
3 // function main begins program execution
4 int main(){
5     int integer1; // first number to be added
6     int integer2; // second number to be added
7     int sum;      // variable in which sum will be stored
8     cout << "Enter first integer: ";
9     cin >> integer1;
10    cout << "Enter second integer\n"; // prompt
11    cin >> integer2;                // read an integer
12    sum = integer1 + integer2; // assign result to sum
13    cout << "Sum is " << sum << endl; // print sum
14    return 0; // indicate that program ended
15 } // end function main
```

Declare integer variables.

Use stream extraction operator with standard input stream to obtain user input.

Calculations can be performed in output statements: alternative for lines 12 and 13:  
`std::cout << "Sum is " << integer1 + integer2 << std::endl;`

Stream manipulator `std::endl` outputs a newline, then “flushes output buffer.”

Concatenating, chaining or cascading stream insertion operations.

# Ví dụ 3

```
#include <iostream>
using namespace std;
void main() {
    int n;
    double d;
    char s[100];

    cout << "Input an int, a double and a
    string.";
    cin >> n >> d >> s;
    cout << "n = " << n << "\n";
    cout << "d = " << d << "\n";
    cout << "s = " << s << "\n";
}
```

## 5.2 Toán tử phạm vi

**Ký hiệu: ::**

- 1) Dùng để truy cập đến biến toàn cục trong trường hợp có biến cục bộ trùng tên.

VD:  $y = ::x + 3$

- 2) Dùng để chỉ rõ phương thức thuộc lớp nào khi nó được viết bên ngoài định nghĩa của lớp mà nó thuộc về.

Kiểu\_dữ\_liệu\_của\_hàm **Tên\_lớp::Tên\_hàm**

- 3) using **std::cout**;

## 5.2 Toán tử phạm vi (tt)

```
1 // Using the unary scope resolution operator.
```

```
2 #include <iostream>
```

```
3 #include <iomanip>
```

```
4 using namespace std;
```

```
5
```

```
6 // define global constant PI
```

```
7 const double PI = 3.141592653
```

```
8 int main()
```

```
9 {
```

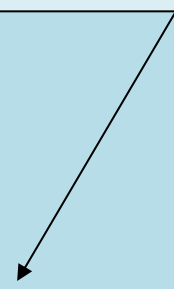
```
10 //define local constant PI
```

```
11 const float PI = static_cast< float >( ::PI );
```

Access the global **PI** with  
**::PI**.

Cast the global **PI** to a  
**float** for the local **PI**.

This example will show  
the difference between  
**float** and **double**.





## 5.2 Toán tử phạm vi (tt)

```
12 // display values of local and global PI constants
13 cout << setprecision( 20 )
14     << " Local float value of PI = " << PI
15     << "\nGlobal double value of PI = " << ::PI<< endl;
16 return 0; // indicates successful termination
17 } // end main
```

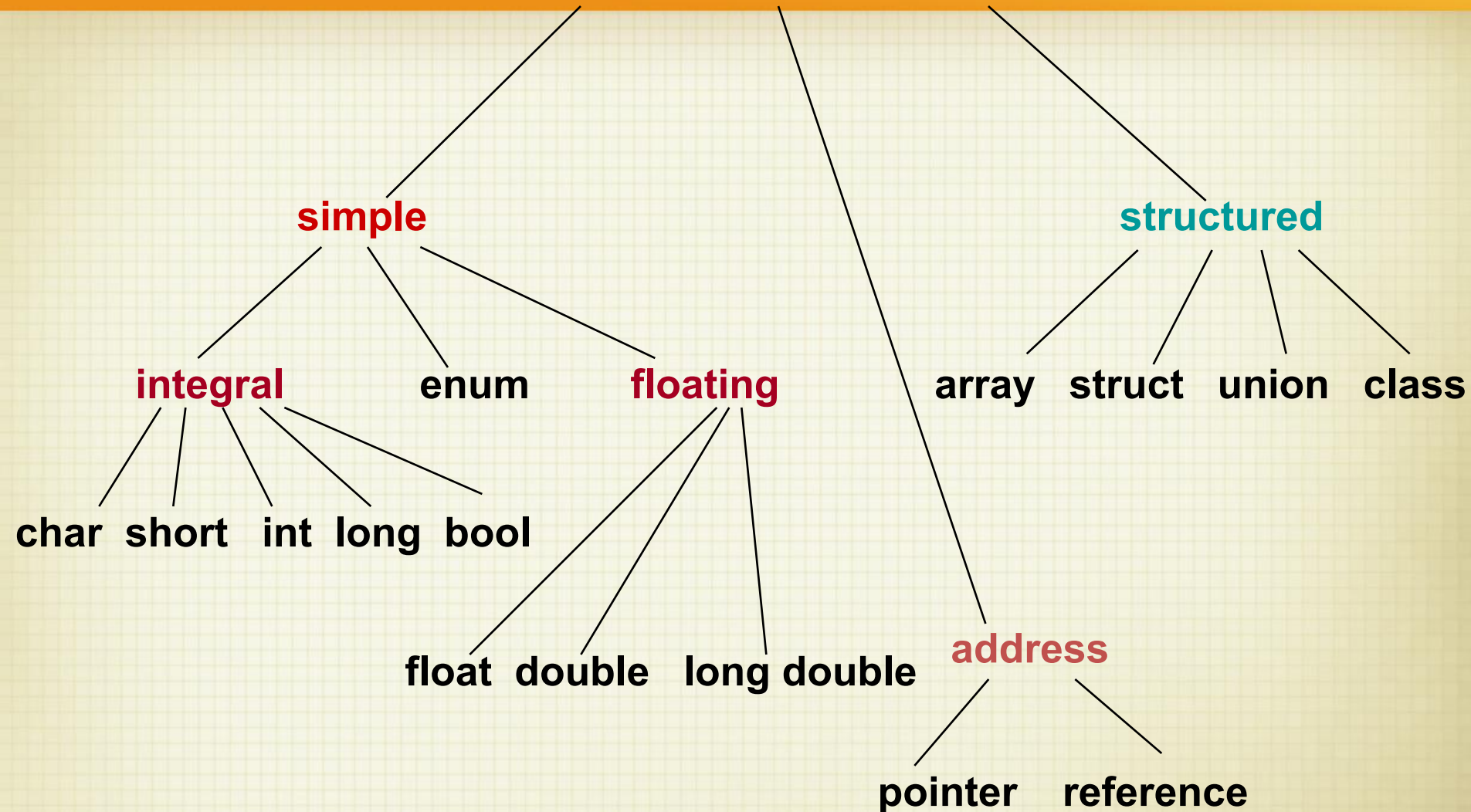
*Borland C++ command-line compiler output:*

```
Local float value of PI = 3.141592741012573242
Global double value of PI = 3.141592653589790007
```

*Microsoft Visual C++ compiler output:*

```
Local float value of PI = 3.1415927410125732
Global double value of PI = 3.14159265358979
```

# 5.3 Các kiểu dữ liệu của C++



## 5.3 Các kiểu dữ liệu của C++ (tt)

- 1) **sizeof(int) = 2; sizeof('A') = 1** //hằng ký tự
- 2) **Kiểu cấu trúc:** tên viết sau từ khóa **struct** là tên kiểu cấu trúc và có thể dùng nó để khai báo biến/mảng cấu trúc.

**Ví dụ:** **struct** TS{ char ht[25]; long sbd;};

-> TS a; //khai báo biến cấu trúc

-> TS ts[1000]; //khai báo mảng cấu trúc

- 3) **Kiểu liệt kê:** **enum** MAU{xanh, do, tim, vang};
- 4) **Kiểm hợp union:** **union** Có\_tên/Không\_tên{...};



## 5.3 Các kiểu dữ liệu của C++ (tt)

5) Định nghĩa Kiểu dữ liệu: dùng từ khóa **typedef**

```
typedef struct {  
    int tu, mau;  
}PS;
```

-> PS p; //khai báo biến p kiểu phân số

```
typedef int MT[20][20];
```

-> MT m; //khai báo biến m là m.trận 20 dòng 20 cột



# 5.3 Các kiểu dữ liệu của C++ (tt)

## 6) Định nghĩa Kiểu phân số, Ma trận, Vector:

- `struct PS{  
 int tu, mau;  
};  
-> PS p; //khai báo biến p kiểu phân số`
- `struct MT{  
 double a[20][20]; //mảng chứa các phần tử kiểu thực của ma trận  
 int m; //số dòng của ma trận  
 int n; //số cột của ma trận  
};  
-> MT mt1; //khai báo biến mt1 kiểu ma trận`
- `struct VT{  
 double b[20]; //mảng chứa các phần tử kiểu thực của vector  
 int n; //số phần tử của vector  
};  
-> VT v; //khai báo biến v kiểu vector`

## 5.3 Các kiểu dữ liệu của C++ (tt)

### 7) Truy xuất đến từng trường của biến cấu trúc:

```
struct {  
    int tu, mau;  
}PS;  
-> PS p;  
cout << "Nhap tu va mau:";  
cin >> p.tu >> p.mau;  
-> PS *p;  
cin >> p->tu >> p->mau;
```

# 5.4 Cấp phát bộ nhớ

## ❖ Cấp phát bộ nhớ: **new**

**Kiểu** \*p; //khai báo 1 biến con trỏ để chứa địa chỉ vùng nhớ sẽ được cấp phát

p = **new Kiểu**; //cấp phát bộ nhớ cho 1 phần tử

p = **new Kiểu**[n]; //cấp phát bộ nhớ cho n phần tử

**Ví dụ:**

float \*p = **new** float;

int \*pn = **new** int[100];



## 5.4 Cấp phát bộ nhớ

### ❖ Giải phóng bộ nhớ: delete

delete p;

delete [] pn;

### ❖ Kiểm tra kết quả cấp phát bộ nhớ:

+ Nếu thành công: p sẽ chứa địa chỉ đầu vùng nhớ được cấp phát.

+ Nếu thất bại: p = NULL



# 5.5 Hàm trong C++

1

Đôi có giá trị mặc định

2

Biến tham chiếu

3

Sử dụng biến tham chiếu

4

Con trỏ hàm

5

Inline Function

6

Định nghĩa chồng các hàm

7

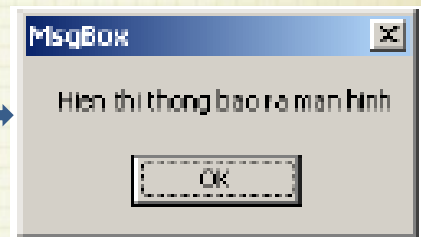
Định nghĩa chồng các toán tử

# 5.5.1 Đối có giá trị mặc định

❖ Ví dụ 1:

```
MessageBox( LPCTSTR lpszText,  
            LPCTSTR lpszCaption = NULL,  
            UINT     nType = MB_OK )
```

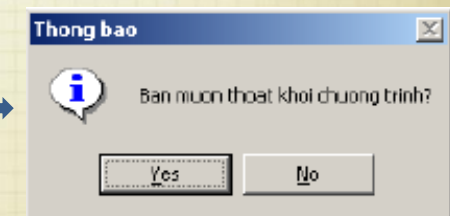
```
MessageBox("Hien thi thong bao ra man hinh");
```



```
MessageBox( "Chuc nang khong su dung duoc",  
            "Bao loi" );
```



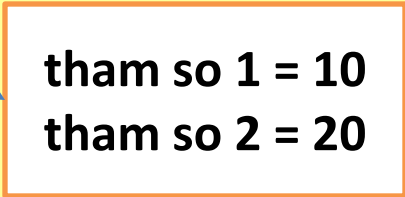
```
MessageBox( "Ban muon thoat khoi chuong trinh?",  
            "Thong bao",  
            MB_YESNO | MB_ICONASTERISK );
```



# 5.5.1 Đối có giá trị mặc định (tt)

## ❖ Ví dụ 2:

```
void Ham1 (int a=0, int b=1) {  
    cout<<"tham so 1 = "<<a<<endl;  
    cout<<"tham so 2 = "<<b<<endl;  
}  
void main() {  
    int x=10, y=20;  
    cout << "Goi Ham1 4 lan, ta duoc : "<<endl;  
    Ham1(x,y);  
    Ham1(x);  
    Ham1(y);  
    Ham1();  
}
```



The diagram shows a call to `Ham1(x,y)` in the code. An arrow points from this call to a box containing the output of that function call:

```
tham so 1 = 10  
tham so 2 = 20
```

# 5.5.1 Đối có giá trị mặc định (tt)

## ❖ Mục đích:

- Gán giá trị mặc định cho các đối số của hàm.
- Giá trị này sẽ được sử dụng khi không truyền giá trị cho các đối số của hàm trong lời gọi hàm.

Nói cách khác, khi không có tham số tương ứng, đối được khởi gán bởi giá trị mặc định.

## ❖ Khai báo hàm có đối mặc định:

- Nếu có khai báo nguyên mẫu hàm thì các đối mặc định phải được khởi gán trong nguyên mẫu, trong định nghĩa không cần khởi gán lại.

```
void delay(int n = 1000);
```

```
void delay(int n){...}
```

- Nếu không khai báo nguyên mẫu hàm thì các đối mặc định được khởi gán trong dòng đầu của định nghĩa hàm.

```
void delay(int n = 1000){...}
```



# 5.5.1 Đối có giá trị mặc định (tt)

## ❖ Gọi hàm có đối mặc định:

- Các tham số thiếu trong lời gọi hàm phải tương ứng với các đối mặc định (tính từ trái sang phải).

VD: void f(int d1, float d2, char \*d3 = "Ha Noi",  
int d4 = 100, double d5 = 3.14)

-> f(3): sai, vì thiếu tham số cho đối không mặc định d2

-> f(3,3.4); f(3,3.4,"ABC"); f(3,3.4,"ABC",10); f(3,3.4,"ABC",10,1.0): đúng

- Như vậy, nếu trong lời gọi hàm cung cấp đủ tham số thì dùng giá trị truyền vào; nếu không cung cấp đủ tham số thì dùng giá trị mặc định.
- Có thể dùng hằng, biến toàn cục, hàm để khởi gán giá trị cho đối mặc định.

VD: void f(int n, int m = MAX, int xmax = getmaxx())

## 5.5.2 Biến tham chiếu

### ❖ Ba loại biến:

- **Biến giá trị:** dùng để chứa dữ liệu.
- **Biến con trỏ:** dùng để chứa địa chỉ.
- **Biến tham chiếu:** dùng làm bí danh (alias) cho một biến giá trị.

### ❖ Ví dụ:

- `int x = 10, *px = &x, &y = x;`
- `y = 30;`

## 5.5.2 Biến tham chiếu (tt)

### ❖ Đặc điểm của biến tham chiếu:

- Biến tham chiếu không được cấp phát bộ nhớ, vì vậy không có địa chỉ riêng.
- Biến tham chiếu sẽ sử dụng chung vùng nhớ của biến mà nó tham chiếu.
- Vì biến tham chiếu không có địa chỉ riêng nên khi khai báo phải chỉ rõ nó tham chiếu đến biến nào.
- Biến tham chiếu có thể tham chiếu đến một phần tử mảng. VD: `int a[5], &r = a[0];`
- Không cho phép khai báo mảng tham chiếu.



## 5.5.2 Biến tham chiếu (tt)

```
1 // Comparing pass-by-value and pass-by-reference
2 // with references.
3 #include <iostream>
4 using namespace std;
5
6 int squareByValue( int ); // function prototype
7 void squareByReference( int & ); // function prototype
8 int main(){
9     int x = 2, z = 4;
10    // demonstrate squareByValue
11    cout << "x = " << x << " before squareByValue\n";
12    cout << "Value returned by squareByValue: "
13        << squareByValue( x ) << endl;
14    cout << "x = " << x << " after squareByValue\n" << endl;
```

Notice the **&** operator, indicating pass-by-reference.



## 5.5.2 Biến tham chiếu (tt)

```
15 // demonstrate squareByReference
16 cout << "z = " << z << " before squareByReference" << endl;
17 squareByReference( z );
18 cout << "z = " << z << " after squareByReference" << endl;
19 return 0; // indicates successful termination
20 } // end main
21 // squareByValue multiplies number by itself, stores result in number
22 // result in number and returns the new value of number
23 int squareByValue( int number ) {
24     return number *= number; // caller's argument modified
25 } // end function squareByValue
26 void squareByReference( int &numberRef ) {
27     numberRef *= numberRef; // caller's argument modified
28 } // end function squareByReference
```

Changes **number**, but original parameter (**x**) is not modified.

Changes **numberRef** an alias for the original parameter. Thus, **z** is changed.

## 5.5.2 Biến tham chiếu (tt)

```
1 // References must be initialized.
2 #include <iostream>
3 using std::cout;
4 using std::endl;
5 int main(){
6     int x = 3;
7
8     int &y = x;
9     cout << "x = " << x << endl << "y = " << y << endl;
10    y = 7;
11    cout << "x = " << x << endl << "y = " << y << endl;
12    return 0; // indicates successful termination
13 }
```

**y** declared as a reference to **x**.

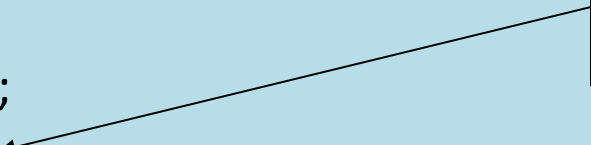


<b>x</b>	=	3
<b>y</b>	=	3
<b>x</b>	=	7
<b>y</b>	=	7

## 5.5.2 Biến tham chiếu (tt)

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int x = 3;
5      int &y;
6      cout << "x = " << x << endl << "y = " << y << endl;
7      y = 7;
8      cout << "x = " << x << endl << "y = " << y << endl;
9      return 0; // indicates successful termination
10 }
```

Uninitialized reference  
– compiler error.



*Borland C++ command-line compiler error message:*

```
Error E2304 Fig03_22.cpp 11: Reference variable 'y' must be
    initialized in function main()
```

*Microsoft Visual C++ compiler error message:*

```
D:\cpphttp4_examples\ch03\Fig03_22.cpp(11) : error C2530: 'y' :
    references must be initialized
```

# 5.5.3 Sử dụng Biến tham chiếu

## ❖ Truyền tham trị:

- Gán giá trị các tham số trong lời gọi hàm cho các đối số của hàm và làm việc trên vùng nhớ của các đối số này.
- Không thao tác trực tiếp trên các tham số, vì vậy không thay đổi được giá trị của các tham số.

## ❖ Truyền tham chiếu:

- Không tạo ra bản sao của các tham số mà thao tác trực tiếp trên vùng nhớ của các tham số, vì vậy dễ dàng thay đổi giá trị của các tham số khi cần.
- Nếu đối là biến tham chiếu kiểu K thì tham số tương ứng trong lời gọi hàm phải là biến/phần tử mảng kiểu K.



## 5.5.3 Sử dụng Biến tham chiếu (tt)

### ❖ Ví dụ:

- `void swap1(int x, int y) { int t = x; x = y; y = t; }`
- `void swap2(int *x, int *y) { int *t = x; x = y; y = t; }`
- `void swap3(int &x, int &y) { int t = x; x = y; y = t; }`

# 5.5.4 Con trỏ hàm

## ❖ Biến con trỏ:

- `float a[20][20];`  
`float *p = (float*)a;`  
`p = ((float*)a) + (i * 20);` //truy xuất đến từng dòng của mảng hai chiều, i bắt đầu từ 0

## ❖ Con trỏ hàm:

```
double nhandoi(double x) {return x * 2;}
double nhanba(double x) {return x * 3;}
void tinh(double (*f)(double), double x) {cout << f(x);}
int main() {
    tinh(nhandoi, 2); cout << endl;
    tinh(nhanba, 3); cout << endl;
    system("pause");
    return 0;
}
```

## 5.5.4 Con trỏ hàm (tt)

### ❖ Con trỏ hàm (tt):

- Khai báo một Con trỏ hàm có Kiểu và Bộ đối như hàm cần lấy địa chỉ.
- Gán Tên hàm cho Con trỏ hàm.

### ❖ Ví dụ:

```
double tinh_max(double *a,int n){...}
```

```
double (*f)(double *,int);
```

```
f = tinh_max;    //lấy địa chỉ của hàm gán cho con trỏ hàm
```



# 5.5.5 Inline Function

## ❖ Ưu điểm:

Tốc độ chương trình tăng lên do không phải thực hiện các thao tác có tính thủ tục khi gọi hàm (cấp phát vùng nhớ cho các đối và biến cục bộ, truyền dữ liệu của các tham số cho các đối, giải phóng vùng nhớ trước khi thoát khỏi hàm, ...).

## ❖ Lưu ý:

- Chỉ nên dùng inline function cho các hàm có kích thước nhỏ vì nó làm tăng khối lượng bộ nhớ chương trình.
- Các hàm chứa biến static, hàm đệ quy, hàm có chứa các lệnh chu trình/lệnh goto/lệnh switch sẽ không được xử lý theo cách inline.



## 5.5.5 Inline Function (tt)

❖ Ví dụ:

```
inline float sqr(float x) {  
    return (x*x);  
}  
  
inline int Max(int a, int b) {  
    return ((a>b) ? a : b);  
}
```

## 5.5.6 Định nghĩa chồng các hàm

- ❖ Định nghĩa chồng các hàm là dùng cùng một tên để định nghĩa các hàm khác nhau.
- ❖ Trình biên dịch C++ sẽ dựa vào sự khác nhau về tập đối của các hàm này để **đổi tên thành các hàm khác nhau**.

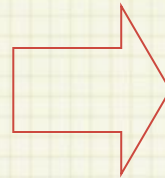
Vì vậy, các hàm trùng tên phải có tập đối khác nhau:

- Về số lượng các đối
  - Về thứ tự các đối
  - Về kiểu dữ liệu của các đối
- ❖ Không chấp nhận các hàm trùng tên chỉ khác nhau ở Kiểu trả về của hàm.

# 5.5.6 Định nghĩa chồng các hàm (tt)

## ❖ Ví dụ:

```
int abs(int i);  
long labs(long l);  
double fabs(double d);
```



```
int abs(int i);  
long abs(long l);  
double abs(double d);
```

## ❖ Gọi hàm:

abs(123); -> gọi hàm abs(int i)

abs(123L); -> gọi hàm abs(long l)

abs(3.14); -> gọi hàm abs(double d)

abs('A'); -> gọi hàm abs(int i)

abs(3.14F) -> gọi hàm abs(double d)

} **Chọn hàm có bộ đối gần kiểu nhất**



## 5.5.6 Định nghĩa chồng các hàm (tt)

### ❖ Lưu ý:

- Chỉ nên định nghĩa chồng các hàm khi muốn thực hiện các công việc như nhau nhưng trên các đối tượng có kiểu khác nhau.
- Dùng phép chuyển kiểu (nếu cần) để bộ tham số trong lời gọi hàm **hoàn toàn trùng kiểu** với bộ đối số của hàm được định nghĩa chồng.



## 5.5.6 Định nghĩa chồng các hàm (tt)

### ❖ Ví dụ 1:

```
int    Max (int a, int b)
{ return (a>b) ? a : b; }
float  Max (float a, float b)
{ return (a>b) ? a : b; }
SinhVien  Max (SinhVien a, SinhVien b)
{ return (a.diemtb > b.diemtb) ? a : b; }
void main() {
    int    x1=1, y1=2;
    float  x2=3, y2=4;
    long   x3=5, y3=6;
    cout << Max(x1,y1) << "\t" << Max(x2,y2) << endl;
    cout << Max(x3,y1) << endl;
    cout << Max(x3,y2) << endl;
    cout << Max(x3,y3) << endl;
}
```

## 5.5.6 Định nghĩa chồng các hàm (tt)

### ❖ Ví dụ 2:

```
int      F (int a=0, int b=1)
    { ... }
float    F (float a=5, float b=9)
    { ... }
void main() {
    int      x1=1, y1=2;
    float    x2=3, y2=4;
    long     x3=5, y3=6;
    cout << F(x1) << "\t" << F(y2) << endl;
    cout << F(x3) << F() << endl;
}
```

## 5.5.7 Định nghĩa chồng các toán tử

- ❖ C++ cung cấp sẵn các phép toán dùng để thao tác trên các kiểu dữ liệu chuẩn (int, float,...).
- ❖ Để thực hiện các phép toán trên các kiểu dữ liệu không chuẩn (kiểu dữ liệu tự định nghĩa như mảng, cấu trúc,...) một cách tự nhiên ta dùng cách **định nghĩa chồng các toán tử**.

## 5.5.7 Định nghĩa chồng các toán tử (tt)

### ❖ Khai báo hàm toán tử:

Kiểu\_trả\_về **operator** Tên\_phép\_toán(danh sách đối số){...}

### ❖ Ví dụ:

```
struct PS {  
    int tu;  
    int mau;  
};  
PS operator+(PS p1, PS p2);  
PS operator-(PS p1, PS p2);  
PS operator*(PS p1, PS p2);  
PS operator/(PS p1, PS p2);
```



## 5.5.7 Định nghĩa chồng các toán tử (tt)

### ❖ Lưu ý:

Số đối của hàm toán tử tùy thuộc vào số toán hạng tham gia vào phép toán được định nghĩa chồng.

**Ví dụ:** Phép toán có 2 toán hạng  $\Rightarrow$  Hàm toán tử cần có ít nhất là 2 đối.

### ❖ Sử dụng hàm toán tử:

- Dùng như phép toán được định nghĩa chồng (phép toán của C++).
- Dùng dấu () để qui định thứ tự thực hiện các phép tính.

# Bài tập 1

- ❖ Tìm lỗi sai cho các khai báo prototype hàm dưới đây (các hàm này trong cùng một chương trình):

`int func1 (int);`

`float func1 (int);`

`int func1 (float);`

`void func1 (int = 0, int);`

`void func2 (int, int = 0);`

`void func2 (int);`

`void func2 (float);`

# Bài tập 2

❖ Cho biết kết xuất của chương trình sau:

```
void func (int i, int j = 0 ){  
    cout << "Số nguyên: " << i << " " << j << endl;  
}  
void func (float i = 0, float j = 0){  
    cout << "Số thực:" << i << " " << j << endl;  
}  
void main(){  
    int i = 1, j = 2;  
    float f = 1.5, g = 2.5;  
    func();           func(i);  
    func(f);          func(i, j);  
    func(f, g);  
}
```



# Bài tập 3

- a. Viết chương trình nhập vào một phân số, rút gọn phân số và xuất kết quả.
- b. Viết chương trình nhập vào hai phân số, tìm phân số lớn nhất và xuất kết quả.
- c. Viết chương trình nhập vào hai phân số. Tính tổng, hiệu, tích, thương giữa chúng và xuất kết quả.
- d. Viết chương trình nhập vào một ngày. Tìm ngày kế tiếp và xuất kết quả.



# Bài tập 4

Cho một danh sách lưu thông tin của các nhân viên trong một công ty, thông tin gồm:

- Mã nhân viên (chuỗi, tối đa là 8 ký tự)
- Họ và tên (chuỗi, tối đa là 20 ký tự)
- Phòng ban (chuỗi, tối đa 10 ký tự)
- Lương cơ bản (số nguyên)
- Thưởng (số nguyên)
- Thực lãnh (số nguyên, trong đó thực lãnh = lương cơ bản + thưởng )

Hãy thực hiện các công việc sau:

- Tính tổng thực lãnh tháng của tất cả nhân viên trong công ty.
- In danh sách những nhân viên có mức lương cơ bản thấp nhất.
- Đếm số lượng nhân viên có mức thưởng  $\geq 1200000$ .
- In danh sách các nhân viên tăng dần theo phòng ban, nếu phòng ban trùng nhau thì giảm dần theo mã nhân viên.

# Kiểm tra quá trình

Viết chương trình nhập vào thông tin của một học sinh, thông tin mỗi học sinh gồm: mã học sinh, họ tên, điểm toán, điểm văn, điểm lý, điểm hóa. Tính điểm trung bình và xuất thông tin của học sinh ra màn hình.

# Q & A

