



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CÁC ĐẶC ĐIỂM CỦA C++

C++



Microsoft®
Visual Studio

Khoa Công nghệ phần mềm

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, KHU PHỐ 6, PHƯỜNG LINH TRUNG, QUẬN THỦ ĐỨC, TP. HỒ CHÍ MINH

[T] 08 3725 2002 101 | [F] 08 3725 2148 | [W] www.uit.edu.vn | [E] info@uit.edu.vn



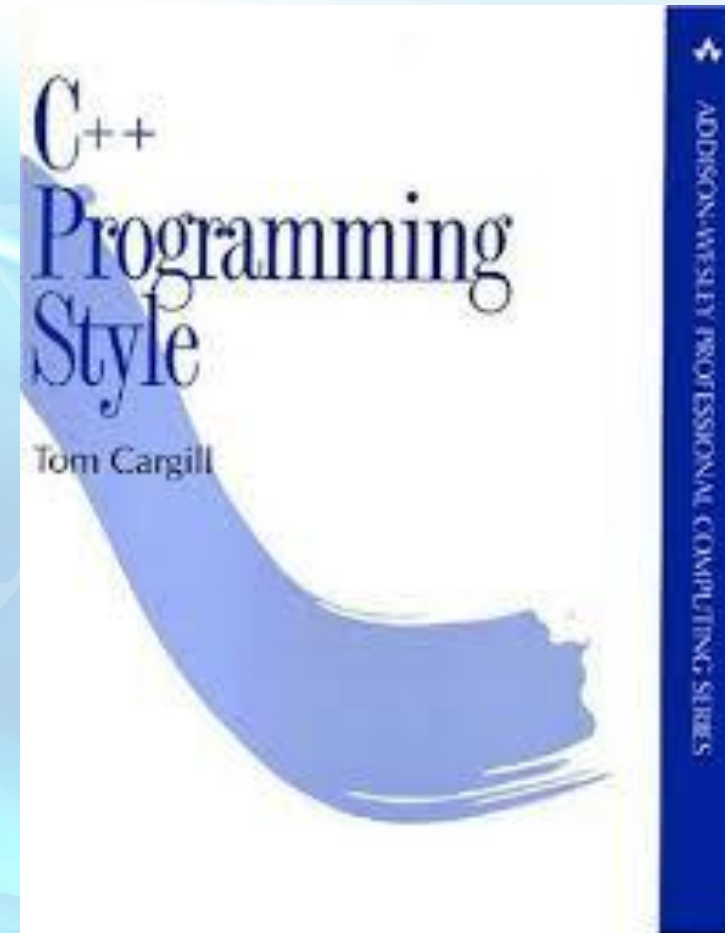
Nội dung

- 1 Một số lưu ý
- 2 Ngôn ngữ C++
- 3 Tham số mặc nhiên
- 4 Tái định nghĩa hàm
- 5 Truyền tham số
- 6 Inline Functions



Phong cách lập trình

- Đặt tên biến, hàm, ...
- Canh lề (tab)
- Khai báo prototype
- { }
- Chú thích





Quy ước đặt tên hằng

- “#define” hoặc “const”
- Tên hằng phải thể hiện được ý nghĩa

#define	N 100	// Không rõ nghĩa.
#define	NUMBER_OF_ELEMENTS 100	// Rõ nghĩa.

- Tên hằng được viết hoa toàn bộ và các từ trong tên cách nhau bằng ký tự “_”.

#define	NumberOfElements 100	// Sai quy ước.
#define	NUMBEROFELEMENTS 100	// Sai quy ước.
#define	NUMBER_OF_ELEMENTS 100	// Đúng quy ước quy ước.



Quy ước đặt tên biến

- Tên biến phải thể hiện được ý nghĩa

int	t, m;	// Không rõ nghĩa.
int	iTuSo, iMauSo;	// Rõ nghĩa.

- Tên biến được viết hoa các ký tự đầu mỗi từ trong tên, các ký tự còn lại viết thường.

int	ituso, imauso;	// Sai quy ước.
int	iTuso, iMauso;	// Sai quy ước.
int	iTuSo, iMauSo;	// Đúng quy ước



Quy ước đặt tên biến

- Tên biến có phần tiếp đầu ngữ (prefix) thể hiện kiểu dữ liệu của biến (phong cách Hungarian)

Kiểu dữ liệu số

char – c	char	cKyTu;
short – s	short	sSoNguyenNgan;
int – i	int	iSoNguyen;
long – l	long	lSoNguyenDai;
float – f	float	fSoThuc;
double – d	double	dSoThucDai;
	int	nSo;

Kiểu dữ liệu luận lý

bool - b	bool	bLuanLy;
----------	------	----------

Kiểu dữ liệu mảng

[] – arr	int	arrSoNguyen[50];
	HocSinh	arrDanhSach[50];

Kiểu dữ liệu chuỗi

char *, char [] – str	char	*strChuoi;
	char	strChuoi[50];

Kiểu dữ liệu con trỏ

* - p	int	*pConTro;
	HocSinh	*pDanhSach;



Quy ước đặt tên kiểu dữ liệu tự định nghĩa

- Tên kiểu dữ liệu tự định nghĩa (struct, class) thường là danh từ và phải thể hiện được ý nghĩa của kiểu dữ liệu:

struct TinhPhanSo

// Sai quy ước.

struct PhanSo

// Đúng quy ước.

struct TinhDiemHocSinh

// Sai quy ước.

class HocSinh

// Đúng quy ước.



Quy ước đặt tên kiểu dữ liệu tự định nghĩa

- Tên kiểu dữ liệu tự định nghĩa được viết hoa các ký tự đầu mỗi từ trong tên, các ký tự còn lại viết thường.

struct phanso *// Sai quy ước.*

struct PHANSO *// Sai quy ước.*

struct Phanso *// Sai quy ước.*

struct PhanSo *// Đúng quy ước.*



Quy ước đặt tên hàm

- Tên hàm thường là động từ và phải thể hiện hành động cần thực hiện:

int DataFile(char *strFileName) *// Sai quy ước.*

int LoadDataFile(char *strFileName) *// Đúng quy ước.*

int BadValue(long IValue) *// Sai quy ước.*

int CheckForBadValue(long IValue) *// Đúng quy ước.*



Quy ước đặt tên hàm

- Tên hàm được viết hoa các ký tự đầu mỗi từ trong tên, các ký tự còn lại viết thường:

<code>int checkforbadvalue(long lValue)</code>	<i>// Sai quy ước.</i>
<code>int CheckforBadvalue(long lValue)</code>	<i>// Sai quy ước.</i>
<code>int CheckForBadValue(long lValue)</code>	<i>// Đúng quy ước.</i>



Quy ước viết câu lệnh

- Viết mỗi câu lệnh riêng trên một dòng:

<pre>// Sai quy ước. x = 3; y = 5;</pre>	<pre>// Đúng quy ước. x = 3; y = 5;</pre>
--	---

<pre>// Sai quy ước. if (a > b) cout << "a lon hon b"; else cout << "a nho hon b";</pre>	<pre>// Đúng quy ước. if (a > b) cout << "a lon hon b"; else cout << "a nho hon b";</pre>
---	--



Quy ước viết câu lệnh

- Viết các câu lệnh if, while, for riêng trên một đoạn:

Sai quy ước

```
if (a > b)
    cout << "a lon hon b";
for (int i = 0; i < n; i++)
    x = x + 5;
k = k * x;
```

Đúng quy ước

```
if (a > b)
    cout << "a lon hon b";

for (int i = 0; i < n; i++)
    x = x + 5;

k = k * x;
```




Quy ước viết câu lệnh

- Viết các câu lệnh cùng thực hiện một công việc riêng trên một đoạn:

```
int c = a;  
a = b;  
b = c;  
k = k * a;  
x = b + c;
```

```
int c = a;  
a = b;  
b = c;  
  
k = k * a;  
x = b + c;
```



Quy ước cách khoảng

- Viết cách vào một khoảng tab đối với các câu lệnh nằm giữa dấu “{” “}”.

```
void Swap(int &a, int &b)
{
    int c = a;
    a = b;
    b = c;
}
```

```
void Swap(int &a, int &b)
{
    int c = a;
    a = b;
    b = c;
}
```

- Viết cách vào một khoảng tab đối với câu lệnh ngay sau if, else, while, for.

```
if (a > b)
    cout << "a lon hon b";
else
    cout << "a nho hon b";

for (int i = 0; i < n; i++)
    x = x + 5;
```

```
if (a > b)
    cout << "a lon hon b";
else
    cout << "a nho hon b";

for (int i = 0; i < n; i++)
    x = x + 5;
```



Quy ước cách khoảng

- Viết cách một khoảng trắng xung quanh các toán tử 2 ngôi

`x=x+5*a-c;` *// Sai quy ước.*

`x = x + 5 * a - c;` *// Đúng quy ước.*

`if(a>=b)` *// Sai quy ước.*

`if (a >= b)` *// Đúng quy ước.*

- Viết cách một khoảng trắng sau các dấu “,” “;”.

`void CalculateValues(int a,int b,int c);` *// Sai quy ước.*

`void CalculateValues(int a, int b, int c);` *// Đúng quy ước.*

`for(int i = 0;i < n;i++)` *// Sai quy ước.*

`for(int i = 0; i < n; i++)` *// Đúng quy ước.*



Bài tập C

- ❖ Nhập bốn số nguyên và xuất các giá trị vừa nhập
 - Có bao nhiêu cách để giải quyết?





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 vòng lặp do while → viết code nhập gọn hơn, viết 1 lần thay cho nhiều lần
4. Dùng mảng và vòng lặp for → viết code gọn hơn, for viết gọn hơn vòng while



Bài tập C – Giải

5. Dùng mảng, 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



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);  
}
```

```
Nhap a1 = 9  
Nhap a2 = 6  
Nhap a3 = 7  
Nhap a4 = 3  
Ban vua nhap 4 so: 9 6 7 3  
Press any key to continue_
```



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]);
}
```




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);
}
```



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]);
    }
}
```



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]);
    }
}
```



Bài tập C – Giải

❖ Cách 6: Dùng hàm

```
void nhap(int []);  
void xuat(int []);
```

```
void main() {  
    int a[4];  
    nhap(a);  
    xuat(a);  
}
```

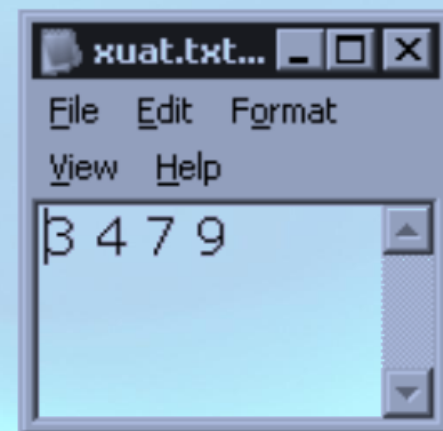
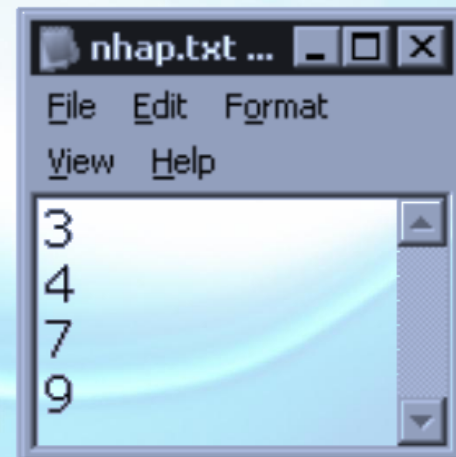
```
void nhap(int b[]) {  
    int i;  
    for (i=0; i<4; i++) {  
        printf("\n a%d = ", i);  
        scanf("%d", &b[i]);  
        printf("%d ", b[i]);  
    }  
}  
  
void xuat(int c[]) {  
    printf("\n 4 so: ");  
    for (i=0; i<4; i++) {  
        printf("%d ", c[i]);  
    }  
}
```




Bài tập C – Giải

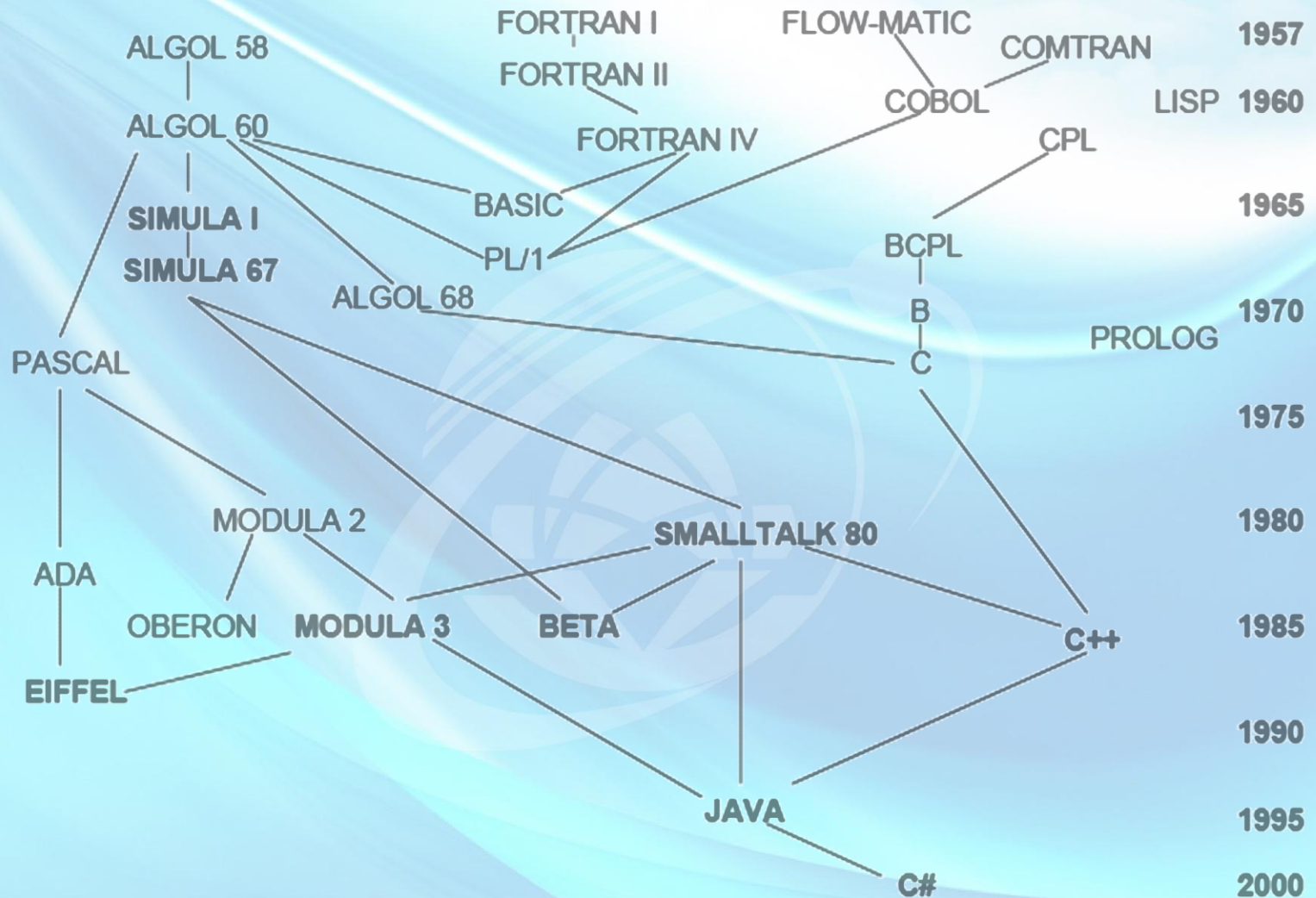
❖ Cách 7: Dùng file

```
void nhap(int b[], char *f) {  
    int i;  
    FILE *fp;  
    fp = fopen(f, "r");  
    for (i=0; i<4; i++) {  
        fscanf(fp, "%d", &b[i]);  
    }  
    fclose(fp);  
}  
void xuat(int c[], char *f) {  
    int i;  
    FILE *fp;  
    fp = fopen(f, "w");  
    for (i=0; i<4; i++) {  
        fprintf(fp, "%d ", c[i]);  
    }  
    fclose(fp);  
}
```





Lịch sử ngôn ngữ lập trình





Lịch sử của C++

- ❖ Mở rộng của **C**
- ❖ Đầu thập niên **1980**: Bjarne Stroustrup (Bell Laboratories)
- ❖ Cung cấp khả năng lập trình hướng đối tượng
- ❖ **Ngôn ngữ lai**





Lịch sử của C++

C++ được xây dựng trên nền của C

- C được phát minh bởi Dennis Ritchie năm 1972
- C dùng để viết hệ điều hành UNIX
- Lịch sử của C và Unix gắn liền với nhau
- UNIX được hoàn thành với C



Lịch sử của C++

C++ được đưa ra bởi Bjarne Stroustrup

- Phiên bản đầu tiên ra mắt năm 1980, với tên "C with class"
- Phiên bản thương mại đầu tiên vào năm 1985
- ANSI và ISO đưa ra phiên bản C++ chuẩn

❖ Ngôn ngữ lai

→ C++ hỗ trợ lập trình hướng đối tượng



Ưu điểm của C++

Ưu điểm:

- Được sử dụng rộng rãi
- Là sự mở rộng của C
- Hỗ trợ lập trình hướng đối tượng
- Có nhiều thư viện mẫu chuẩn STL



Mở rộng của C++

Một số mở rộng của C++ so với C:

- ✓ Lời chú thích
- ✓ Từ khóa mới
- ✓ Dữ liệu, khai báo biến
- ✓ Chuyển kiểu
- ✓ Nhập xuất
- ✓ Cấp phát bộ nhớ
- ✓ Biến, Hằng tham chiếu
- ✓ Hàm đa năng
- ✓ Toán tử đa năng
- ✓ Hàm nội tuyến
- ✓ Toán tử phạm vi
- ✓ Con trỏ this



Lời chú thích

Có hai cách chú thích:

✓ Cách 1: `/* ..*/`

Ví dụ: `/* chú thích trên
nhiều dòng*/`

✓ Cách 2: `//` (*chú thích của C*)

Ví dụ: `// Chú thích trên một dòng`



Từ khóa mới

Một số từ khóa mới:

delete	catch	class
new	friend	inline
protected	operator	private
this	public	template
virtual	throw	try

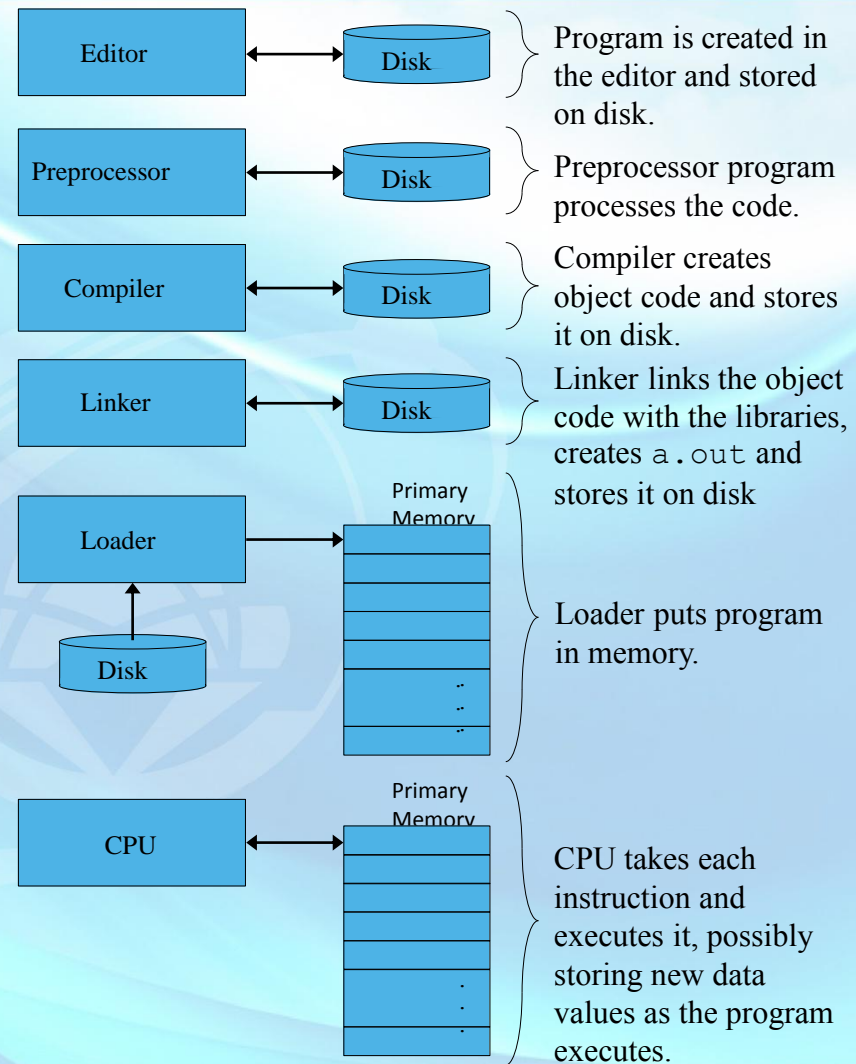
- Nếu trong chương trình viết bằng C có tên trùng → thay đổi lại



Môi trường của C++

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

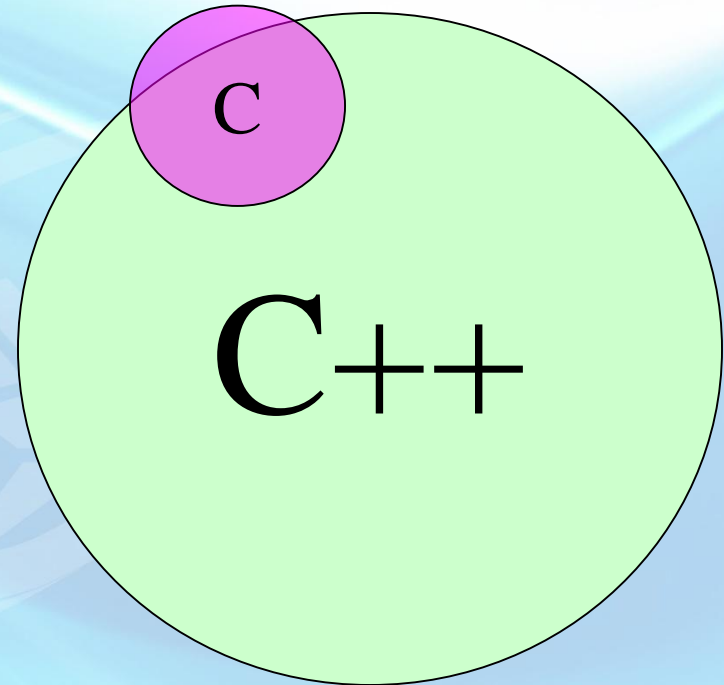
- Edit
- Preprocess
- Compile
- Link
- Load
- Execute





Khác biệt đối với C

- ❖ Chú thích
- ❖ Các kiểu dữ liệu
- ❖ Kiểm tra kiểu, đổi kiểu
- ❖ Phạm vi và khai báo
- ❖ Không gian tên
- ❖ Hằng
- ❖ Quản lý bộ nhớ
- ❖ Tham chiếu





Khác biệt đối với C

❖ Phạm vi và khai báo:

- Không giống như C, chúng ta có thể **khai báo một biến tại một vị trí bất kỳ** trong chương trình.
- Một biến chỉ có tầm tác dụng trong khối lệnh nó được khai báo.
- Do đó, **C++ cung cấp toán tử định phạm vi (::)** để xác định rõ biến nào được sử dụng khi xảy ra tình trạng định nghĩa chồng một tên biến trong một khối lệnh con.



Toán tử phạm vi

❖ Toán tử phạm vi (::)

- Thường được dùng để truy cập các biến toàn cục trong trường hợp có biến cục bộ trùng tên
- Ví dụ:

$y = ::x + 3;$





Toán tử phạm vi

```
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 = 3.14159265358979;
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.



Toán tử phạm vi

```
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
```



Nhập xuất với C++

❖ cin

❖ Luồng nhập chuẩn

❖ cout

❖ Luồng xuất chuẩn

❖ cerr

❖ Luồng thông báo lỗi chuẩn



Nhập xuất với C++

❖ cin and cout (and #include <iostream>):

```
cout << "hey";
```

```
char name[10];
```

```
cin >> name;
```

```
cout<<"Hey "<<name<<"", nice name." << endl;
```

```
cout << endl;
```



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 program ended successfully
```

```
11
```

```
12 } // end function main
```

Function main returns an integer value.

...ive to include
... header file

Function main appears every C++ program..

Left brace { begins function body.

Statements end with a semicolon ;.

Corresponding right brace } 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
6     int integer2; // second number to
7     int sum;      // variable in which sum will be stored
8     cout << "Enter first integer: ";
9     cin >> integer1;
10    cout << "Enter second integer: ";
11    cin >> integer2;
12    sum = integer1 + integer2; // assign result to
13    cout << "Sum is " << sum << endl; // print sum
14    return 0; // indicate that program ended successfully
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.



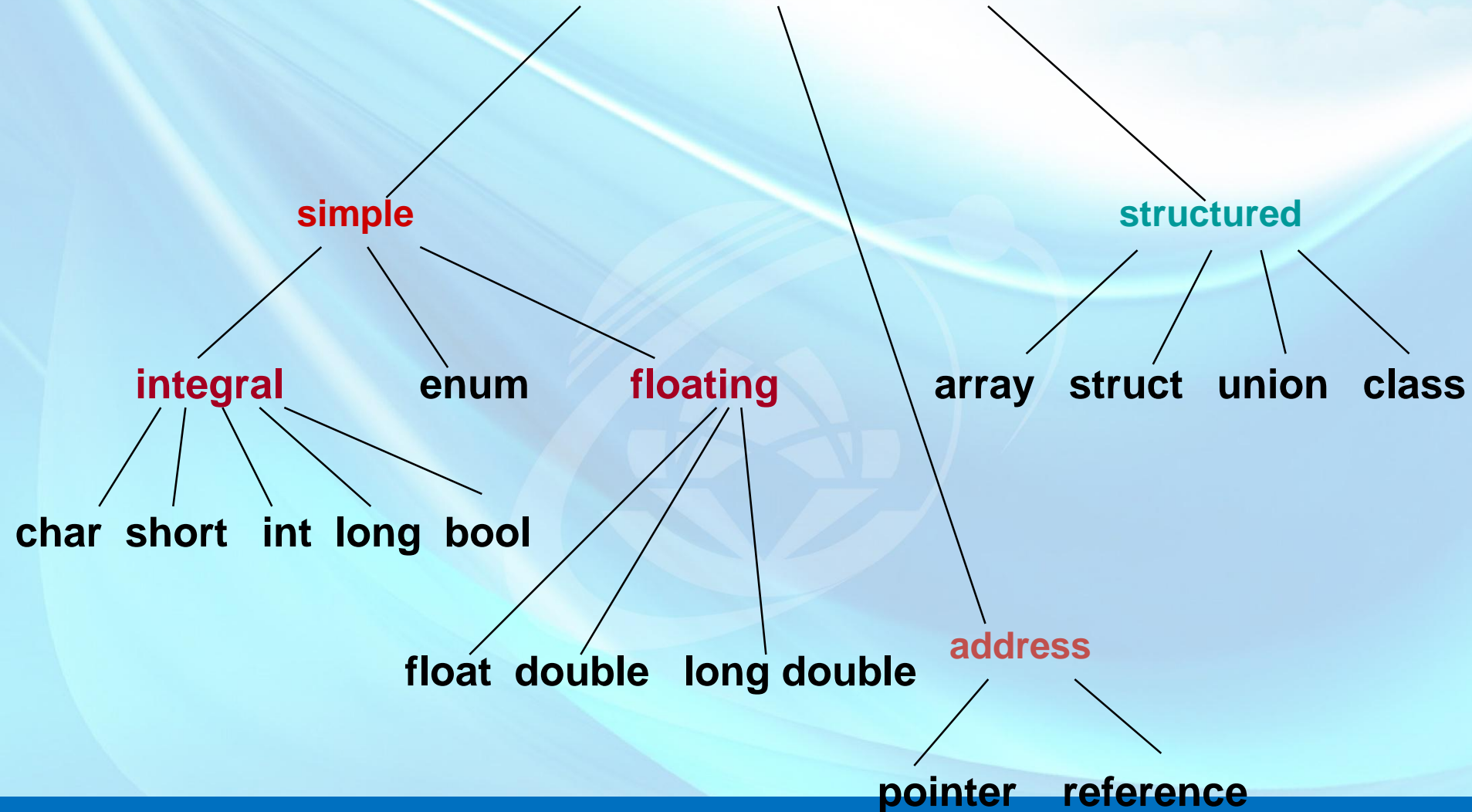
Ví dụ 3

```
#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";
}
```




Các kiểu dữ liệu của C++





Tham số mặc nhiên

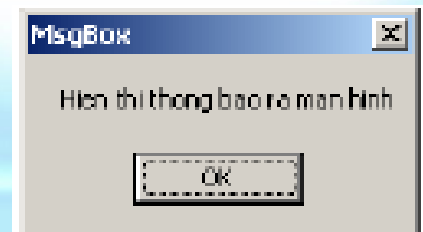
❖ Ví dụ 1: Hàm thể hiện một cửa sổ thông báo trong Visual C++

```
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 );
```





Tham số mặc nhiên

❖ 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();  
}
```



Tham số mặc nhiên

❖ Mục đích:

- Gán các giá trị mặc nhiên cho các tham số của hàm.

❖ Khai báo tham số mặc nhiên:

- Tất cả các tham số mặc nhiên đều phải để ở cuối hàm.
- Chỉ cần đưa vào khai báo, không cần trong định nghĩa.

❖ Gọi hàm có tham số mặc nhiên:

- Nếu cung cấp đủ tham số → dùng tham số truyền vào.
- Nếu không đủ tham số → dùng tham số mặc nhiên.



Tái định nghĩa hàm

❖ Functions overloading

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



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

❖ C++ cho phép **định nghĩa các hàm trùng tên.**



Tái định nghĩa hàm

❖ Qui tắc tái định nghĩa:

- Các hàm **trùng tên** phải **khác** nhau về **tham số**: Số lượng, thứ tự, kiểu

❖ Qui tắc gọi hàm?

- Tìm hàm có kiểu tham số phù hợp
- Dùng phép ép kiểu tự động
- Tìm hàm gần đúng (phù hợp) nhất



Tái định nghĩa hàm

❖ 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;
}
```



Tái định nghĩa hàm

❖ 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;
}
```




Toán tử quản lý bộ nhớ động

❖ Toán tử cấp phát bộ nhớ động **new**

```
int *x;
```

```
x = new int;           //x = (int*)malloc(sizeof(int));
```

```
char *y;
```

```
y = new char[100];     //y = (char*)malloc(100);
```

❖ Toán tử giải phóng vùng nhớ động **delete**

```
delete x;              // free(x);
```

```
delete y;              // free(y);
```



Truyền tham số

❖ Truyền theo giá trị (tham trị)

- Giá trị tham số khi ra khỏi hàm sẽ không thay đổi.

❖ Truyền theo địa chỉ (tham chiếu)

- Giá trị tham số khi ra khỏi hàm có thể thay đổi.



Tham chiếu

- ❖ Tham chiếu là địa chỉ vùng nhớ được cấp phát cho một biến.
- ❖ Ký hiệu **&** đặt trước biến hoặc hàm để xác định tham chiếu của chúng
- ❖ Ví dụ 1:
 - `int x = 10, *px = &x, y = x;`
 - `*px = 20;`
 - `y = 30;`



Tham chiếu

❖ Ví dụ 2:

- `int arrget(int *a, int i) { return a[i]; }`
- `arrget(a, 1) = 1; // a[1] = 1;`
- `cin >> arrget(a,1); // cin >> a[1];`

❖ Ví dụ 3:

- `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; }`



Tham chiếu

```
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.



Tham chiếu

```
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
22 // result in number and returns the new value of number
23 int squareByValue( int number ) {
24     return number *= number; // caller's argument not 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.



Tham chiếu

```
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.

x	=	3
y	=	3
x	=	7
y	=	7



Tham chiếu

```
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



Hàm Inline

- ❖ Hàm inline hay còn gọi là hàm nội tuyến.
- ❖ Sử dụng từ khóa **inline**
- ❖ Yêu cầu trình biên dịch **copy code vào trong chương trình** thay vì thực hiện lời gọi hàm:
 - Giảm thời gian thực thi chương trình
 - Tăng kích thước của mã lệnh thực thi
- ❖ **Chỉ nên định nghĩa inline khi hàm có kích thước nhỏ**



Hàm Inline

❖ Ví dụ:

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



Function Templates

- ❖ Compact way to make overloaded functions
 - Generate separate function for different data types
- ❖ Format
 - Begin with keyword **template**
 - Formal **type parameters in brackets** `<>`
 - Every type parameter preceded by typename or class
 - Placeholders for built-in types (i.e., int) or user-defined types
 - Specify arguments types, return types, declare variables
 - Function definition like normal, except formal types used



Function Templates

❖ Example

```
template < class T >
//or template< typename T >
T square(T value1)
{
    return value1 * value1;
}
```

- T is a formal type, used as parameter type
 - Above function returns variable of same type as parameter
- In function call, T replaced by real type



Function Templates

```
1  // Using a function template.
2  #include <iostream>
3  using std::cout;
4  using std::cin;
5  using std::endl;
6  // definition of function template maximum
7  template < class T > // or template < typename T >
8  T maximum( T value1, T value2, T value3 )
9  {
10     T max = value1;
11     if ( value2 > max )
12         max = value2;
13     if ( value3 > max )
14         max = value3;
15
16     return max;
17 } // end function template maximum
```

Formal type parameter `T`
placeholder for type of data to
be tested by `maximum`.

`maximum` expects all
parameters to be of the
same type.



Function Templates

```
18  int main()
19  {
20      // demonstrate maximum with int values
21      int int1, int2, int3;
22      cout << "Input three integer values: ";
23      cin >> int1 >> int2 >> int3;
24      // invoke int version of maximum
25      cout << "The maximum integer value is: "
26           << maximum( int1, int2, int3 );
27      // demonstrate maximum with double values
28      double double1, double2, double3;
29      cout << "\n\nInput three double values: ";
30      cin >> double1 >> double2 >> double3;
31      // invoke double version of maximum
32      cout << "The maximum double value is: "
33           << maximum( double1, double2, double3 );
```

maximum called with various data types.



Function Templates

```
34  // demonstrate maximum with char values
35  char char1, char2, char3;
36  cout << "\n\nInput three characters: ";
37  cin >> char1 >> char2 >> char3;
38  // invoke char version of maximum
39  cout << "The maximum character value is: "
40       << maximum( char1, char2, char3 ) << endl;
41  return 0; // indicates successful termination
42  } // end main
```

```
Input three integer values: 1 2 3
The maximum integer value is: 3
Input three double values: 3.3 2.2 1.1
The maximum double value is: 3.3
Input three characters: A C B
The maximum character value is: C
```



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 << "So nguyen: " << i << " " << j << endl;  
}  
void func (float i = 0, float j = 0){  
    cout << "So thuc:" << 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); func(l, f);  
}
```




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.



Bài tập 4

- a. Viết chương trình nhập vào một ngày. Tìm ngày kế tiếp và xuất kết quả.
- b. Viết chương trình nhập họ tên, điểm toán, điểm văn của một học sinh. Tính điểm trung bình và xuất kết quả.



Bài tập 5

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 ≥ 1200000 .
- In danh sách các nhân viên tăng dần theo phòng ban, nếu phòng ban trùng nhau thì giảm dần theo mã nhân viên.



Q & A



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, KHU PHỐ 6, PHƯỜNG LINH TRUNG, QUẬN THỦ ĐỨC, TP. HỒ CHÍ MINH

[T] 08 3725 2002 101 | [F] 08 3725 2148 | [W] www.uit.edu.vn | [E] info@uit.edu.vn