Chương 7:

Hàm - Lập trình hướng cấu trúc

7.1. **Giới thiệu**:

Ví dụ ta có chương trình sau: Nhập một mảng 1 chiều n phần tử, và xuất mảng đó ra màn hình sau khi tăng giá trị từng phần tử lên 1 đơn vị

```
1   #include  <iostream>
2   #include <conio>
3   using namesapce std;
4   int a[100];
5   int n;
6   int main()
7   {
8         cout<<"Nhập dữ liệu của mảng 1 chiều: "<<endl;
9         cout<<"nhập vào số lượng phần tử: ";
10        cin>>n;
11       for (int i = 0;  i < n; i++)
12       {
13              coutt<<"Nhập vào phần tử thứ "<<i<<": ";
14              cin>>a[i];
14        }
16
17      cout<<"Mảng vừa nhập"<<endl;
18      for (i = 0;  i < n; i++)
19            cout<<a[i]<<"   ";
20      cout<<endl;
21
22
23        //Tăng các phần tử của mảng lên 1 đơn vị
24        for (i = 0; i < n; i++)
```

```
35              ++a[i];
26
27      cout<<"Mảng sau khi xử lý: "<<endl;
28      for (i = 0;  i < n; i++)
29            cout<<a[i]<<"   ";
30      cout<<endl;
31
32
33      getch();
        return 0;
    }
```

Phân tích:

- Ta nhận thấy đoạn chương trình từ **18 -> 20** và **28->30** rất giống nhau và cùng thể hiện là xuất nội dung của mảng ra màn hình.
- Làm sao để không lặp lại những đoạn chương trình giống nhau như vậy?

7.2. **Định nghĩa hàm (hay còn gọi là đơn thể)**:

Mục đích của việc xây dựng hàm:

- **Giải quyết vấn đề phải lặp lại nhiều đoạn chương trình giống nhau**
- **Chia bài toán lớn thành nhiều bài toán nhỏ hơn. Khi đó để giải quyết bài toán lớn ta sẽ đi giải quyết các bài toán nhỏ đó**

a. **Giải quyết bài toán ví dụ**:

```
1   #include  <iostream>
2   #include <conio>
3   using namesapce std;
4   int a[100];
5   int n;
6
7
```

```
8   //Khai báo hàm (prototype)
9   void    xuat_mang_1chieu();
10
11  int main()
12  {
13        cout<<"Nhập dữ liệu của mảng 1 chiều: "<<endl;
14        cout<<"nhập vào số lượng phần tử: ";
14        cin>>n;
16       for (int i = 0;  i < n; i++)
17       {
18              coutt<<"Nhập vào phần tử thứ "<<i<<": ";
19              cin>>a[i];
20        }
21
22       cout<<"Mảng vừa nhập"<<endl;
23       xuat_mang_1chieu();    //Gọi hàm xuất mảng 1 chiều
24
35        //Tăng các phần tử của mảng lên 1 đơn vị
26        for (i = 0; i < n; i++)
27              ++a[i];
28
29       cout<<"Mảng sau khi xử lý: "<<endl;
30       xuat_mang_1chieu();     //Gọi hàm xuất mảng 1 chiều
31
32       getch();
33       return 0;
34  }
35
36  void    xuat_mang_1chieu();
37  {
38        for (i = 0;  i < n; i++)
39              cout<<a[i]<<"   ";
40        cout<<endl;
41
42  }
```

b. **Hàm không có giá trị trả về**:

Cú pháp:

i. Hàm không có truyền tham số

Cú pháp	Ví dụ
void tên_hàm() { *//Khối lệnh;* }	void Hàm_Chào() { cout<<"hello world"; cout<<endl; }

ii. Hàm có truyền tham số

Cú pháp	Ví dụ
void tên_hàm(kiểu thamsô1 **,** kiểu **&** thamsố2 ,..) { *//Khối lệnh;* }	void Xuất_Tổng(int a, int b) { int s = a + b; cout<<"Tổng "<<a<<"và"<<b<<"là "; cout<<s; }

c. **Hàm có giá trị trả về**:

Cú pháp:

i. Hàm không có truyền tham số

Cú pháp	Ví dụ
kiểu_trả_về tên_hàm() { *//Khối lệnh;* **return** giá_trị_trả_về; }	**int** Tổng_1_đến_10() { int s = 0; for (int i = 1; i<=10 ; i++) s + = i; **return s;** *// Giá trị trả về* }

ii. Hàm có truyền tham số

Cú pháp	Ví dụ
kiểu_trả_về tên_hàm(kiểu thamsô1**,** kiểu **&** thamsố2 ,..) { *//Khối lệnh;* **return** giá_trị_trả_về; }	**int** Tính_Tổng(int a, int b) { int s = a + b; **return s;** *// Giá trị trả về* }

7.3. **Khai báo hàm – khai báo prototype**:

Cú pháp:

void Tên_Hàm(danh sách **kiểu** các tham số)**;**
kiểu_trả_về Tên_Hàm(danh sách **kiểu** các tham số)**;**

Chú ý:

- *Danh sách kiểu tham số*: chỉ ra các kiểu của tham số (có thể có hoặc không có tên biến)

- Vị trí đặt prototype thường là phía **trên hàm main** - Có dấu **chấm phẩy (;)** ở cuối prototype

Ví dụ:

```
//Khai báo prototype
void   Xuat_Tong(int , int ) ;

void main()
{
   //Khối lệnh trong hàm main
}

//Định nghĩa hàm
void  Xuat_Tong(int a, int b)
{
   int s = a + b;
   cout<<"Tổng của "<<a<<" và "<<b<<" là "<<s;
   cout<<endl;
}
```

7.4. **Lời gọi hàm**:

a. **Ví dụ với hàm không có giá trị trả về**

```
//Khai báo prototype
void   Xuat_Tong(int , int ) ;

void main()
{
        Xuat_Tong( 2 , 3 );   //Gọi hàm Xuat_Tong
}

//Định nghĩa hàm
void  Xuat_Tong(int a, int b)
{
    int s = a + b;
    cout<<"Tổng của "<<a<<" và "<<b<<" là "<<s;
    cout<<endl;
}
```

Khi đó có sự so khớp như sau:

Hàm void Xuat_Tong(int a , int b)

Lời gọi hàm Xuat_Tong(2 , 3);

Hay tại thời điểm gọi hàm Xuat_Tong *a sẽ có giá trị 2*, *b sẽ có giá trị là 3*
Vậy hàm Xuat_Tong sẽ xuất ra màn hình:

Tổng của 3 và 5 là 8

b. **Ví dụ với hàm có giá trị trả về**

```
//Khai báo prototype
int  Tinh_Tong(int , int ) ;

void main()
{
        int i;
        i = Xuat_Tong( 2 , 3 );   //Gọi hàm Tinh_Tong
       cout<<i;
}

//Định nghĩa hàm
int  Tinh_Tong(int a, int b)
{
    int s = a + b;
    return s;
}
```

Tương tự ta có sự so khớp như sau:

Hàm int Tinh_Tong(int a , int b)

Lời gọi hàm i = Tinh_Tong(2 , 3);

Hay tại thời điểm gọi hàm Xuat_Tong *a sẽ có giá trị 2*, *b sẽ có giá trị là 3*
Và giá trị 5 sẽ được **trả về** cho biến i, khi đó i sẽ có giá trị là 5

Chú ý:
Lệnh **return** ngoài ý nghĩa là trả về giá trị của hàm, nó còn có ý nghĩa là thoát khỏi chương trình con chứa nó

7.5. **Tham trị và tham chiếu (quan trọng)**:

Tham trị	Tham chiếu
- khi khai báo **không có dấu &** giữa kiểu và tham số Ví dụ: void ham(int a) { *//Khối lệnh* }	- khi khai báo **có** dấu & giữa kiểu và tham số void ham(int **&** a) { *//Khối lệnh* }
- **Không làm thay đổi** biến được gọi	- **Làm thay đổi** biến được gọi
void binh_phuong(int); void main() { int i = 5; binh_phuong(i); cout<<i; ***// i = 5*** ***// Không làm thay đổi i*** } void binh_phuong(int a) { a *= a; cout<<a; *// a = 25* }	void binh_phuong(**int &**); void main() { int i = 5; binh_phuong(i); cout<<i; ***// i = 25*** ***// Làm thay đổi i*** } void binh_phuong(**int &** a) { a *= a; cout<<a; *// a = 25* }

7.6. **Giá trị mặc định của tham số**:

a. **Ý nghĩa:**

Cho phép gọi hàm một cách linh hoạt:

- Nếu có truyền tham số thì hàm sẽ lấy tham số được truyền
- Nếu không có truyền tham số thì hàm sẽ lấy giá trị mặc định

b. **Các khai báo:**

Ta chỉ cần khai báo các giá trị mặc định của tham số lúc khai báo prototype (khai báo hàm)

```
kiểu_trả_về  tên_hàm(kiểu  thamsố1 = giátrị1, kiểu  thamsố2=giá_trị 2, ...);
```

Chú ý:

- Giá trị mặc định của tham số phải được khai báo từ **phải sang trái**

Ví dụ 1:

```
int Tinh_Tong(int , int b = 1);   //Giá trị mặc định của b là 1

int main()
{
        int i;
        i   = Tinh_Tong ( 2 , 3);
        cout<<i;          //i =5

        int j;
        j  = Tính_Tong( 2 );  //Chỉ truyền có 1 tham số
                              //Khi đó b sẽ lấy giá trị mặc định là 1
        cout<<j;     // j = 2 + 1 = 3
}

//Định nghĩa hàm vẫn như bình thường
int Tinh_Tong(int a, int b)
{
        int s = a + b;
        return s;
}
```

Ví dụ 2:

Khai báo giá trị mặc định như sau là **sai**

```
int Tinh_Tong(int a = 1, int b);
```

Do a được khai báo có giá trị mặc định trong khi b không có giá trị mặc định (*nhắc lại: Giá trị mặc định của tham số phải được khai báo từ **phải sang trái***)

7.7. **Quá tải hàm:**

a. **Các thành phần của hàm:**

Hai hàm khác nhau khi chúng có ít nhất 1 trong 3 thành phần khác nhau, gồm:

- Tên hàm
- Kiểu của tham số
- Số lượng tham số

b. **Quá tải hàm:**

Quá tải hàm là những hàm có cùng tên hàm, nếu rơi vào 1 trong 2 trường hợp sau:

- Số lượng tham số khác nhau
- Cùng số lượng tham số, khác nhau ở một kiểu dữ liệu nào đó của tham số

Ví dụ 1:

```
void   Xuat_Tong(int a, int b);
void   Xuat_Tong(int a, int b, int c);

int main()
{
        Xuat_Tong( 1 , 2);

        Xuat_Tong( 1 , 2,  3);
}

void   Xuat_Tong(int a, int b)
{
        cout<<a+b;
}

void  Xuat_Tong(int a, int b, int c)
{
        cout<<a + b + c;
}
```

Ví dụ 2:

```
void   Xuat_Tong(int a, int  b);
void   Xuat_Tong(int a, float b);
```

7.8. Quá tải toán tử:

a. Các toán tử trong C++:

Các toán tử quen thuộc trong C++

Toán tử	Ví dụ
+	a = b + c;
-	a = b – c;
*	a = b * c;
/	a = b / c;
%	a = b % c;
=	a = b;
==	if (a == b){}
[]	a[1]; *//với a là mảng*
<<	cout<<i;
>>	cin>>i;

Các toán trong C++ đều có chung một cách đặt **tên hàm** theo mô tả sau:

operator <ký_hiệu_phép_toán>

b. Quá tải toán tử:

Ứng dụng: **ta có thể sử dụng các toán tử của C++ đối với các biến cấu trúc mà ta tự nghĩa như những kiểu bình thường trong C++**

Ví dụ:

với kiểu cấu trúc PHÂNSỐ được khai báo như sau:

```
struct PHANSO
{
        int x;
        int mau;
};

int main()
{
        PHANSO ps1, ps2;

        cin>>ps1>>ps2;   //Nhập như kiểu dữ liệu bình thường

        PHANSO ps;
        ps = ps1  +   ps2;   //Thực hiện phép cộng như kiểu dữ liệu
                             //bình thường khác

        cout<<ps;            //Xuất như kiểu dữ liệu bình thường
}
```

Để làm áp dụng các toán tử vào kiểu PHANSO như ví dụ trên thì ta phải **định nghĩa các hàm quá tải toán tử**

Ví dụ:

1. **Quá tải toán tử + của 2 phân số**

```
PHANSO    operator +  (PHANSO  a , PHANSO b)
{
        PHANSO ps;
        ps.tu   = a.tu * b.mau  + a.mau * b.tu;
        ps.mau = a.mau * b.mau;
        return ps;
}
```

2. **Quá tải toán tử << cho phân số**

```
ostream& operator(ostream& os,  PHANSO ps)
{
        os<<ps.tu<<"  /  "<<ps.mau;

        return os;
}
```

3. **Quá tải toán tử << cho phân số**

```
istream & operator(istream & is,  PHANSO ps)
{
        is>>ps.tu;
        is>>ps.mau;

        return is;
}
```

Bài tập về hàm:

1. Sử dụng hàm và quá tải hàm, quá tải toán tử để viết lại bài tập phân số
    a. kiểm tra tính đúng đắn của phân số
    b. Nhập / Xuất phân số
    c. Rút gọn phân số
    d. Cộng trừ nhân chia phân số
    e. Quá tải toán tử nhập, xuất, = , ==, +, - , *, /

2. Sử dụng hàm và quá tải hàm, quá tải toán tử viết lại bài tập về mảng 1 chiều kiểu số nguyên
    a. Nhập / xuất mảng 1 chiều
    b. Tìm các số nguyên tố trong mảng
    c. Tìm các số chính phương trong mảng
    d. Thêm 1 phần tử vào đầu dãy
    e. Thêm 1 phần tử vào cuối dãy
    f. Thêm 1 phần tử vào vị trí k
    g. Xoá 1 phần tử ở vị trí k
    h. Tìm kiếm phần tử có giá trị x
    i. Tìm min
    j. Tìm max
    k. Tính tổng
    l. Tính trung bình các số hạng
    m. Đảo ngược dãy
    n. Sắp xếp theo thứ tự tăng/giảm
3. Sử dụng hàm và quá tải hàm, quá tải toán tử viết lại bài tập về mảng 2 chiều kiểu số nguyên
    a. Nhập / xuất mảng
    b. Thêm 1 dòng vào đầu ma trận
    c. Thêm 1 phần tử vào cuối ma trận
    d. Thêm 1 phần tử vào dòng thứ k
    e. Xoá dòng thứ k
    f. Tìm min
    g. Tìm max
    h. Tính tổng
    i. Tính trung bình các số hạng
    j. Tìm kiếm phần tử có gía trị x