KỸ THUẬT LẬP TRÌNH C/C++

Chương 7:
KỸ THUẬT LẬP TRÌNH VỚI HÀM

Nội dung

1. Khái niệm hàm trong C++
2. Các loại hàm trong C++
3. Biến toàn cục và biến cục bộ
4. Tham số và đối số trong hàm
5. Nạp chồng hàm

1. Khái niệm hàm trong C++

1. Khái niệm hàm trong C++

Một hàm (function) là một nhóm các câu lệnh cùng nhau thực hiện một nhiệm vụ nhất định.

Hàm giúp phân chia vấn đề phức tạp thành các thành phần nhỏ giúp chương trình dễ hiểu và dễ sử dụng.

Một chương trình C++ có thể bao gồm một hoặc nhiều hàm. Trong đó, hàm main() là hàm bắt buộc của chương trình.

Một hàm có thể được gọi nhiều lần với các tham số khác nhau. Nó cung cấp tính mô đun và khả năng sử dụng lại đoạn mã.

Hàm trong C++ còn được gọi là thủ tục hoặc chương trình con trong các ngôn ngữ lập trình khác.

Trong chương trình, để sử dụng hàm thì cần phải:

+ Xây dựng hàm: khai báo hàm và viết các câu lệnh thực hiện các chức năng của hàm.
+ Gọi hàm: có truyền tham số hoặc không truyền tham số.

1. Khái niệm hàm trong C++

Hàm trong C++ có ưu điểm như sau:

- Tái sử dụng lại code: Một hàm được tạo trong C++, có thể được gọi lại hàm nhiều lần mà không cần phải viết lại đoạn code trùng nhau nhiều lần.
- Tối ưu code: Nó làm cho chương trình C++ ngắn gọn và tối ưu hơn, vì chương trình không có nhiều đoạn code trùng lặp nhau.
- Dể đọc và dể bảo trì: Nó làm cho chương trình chúng ta rõ ràng và dễ hiểu, rất tiện lợi cho việc bảo trì chương trình sau này.

1. Khái niệm hàm trong C++

❖ **Khai báo hàm:**

Cú pháp:

```
<kiểu trả về> <tên hàm>([danh sách tham số]) {
        <các câu lệnh trong hàm>
        return <giá trị>;
}
```

Trong đó:
<kiểu trả về>: kiểu dữ liệu bất kỳ trong C++ (char, int, float,…). Nếu không trả về thì là void.
<tên hàm>: theo quy tắc đặt tên định danh.
[danh sách tham số]: tham số hình thức đầu vào giống khai báo biến, cách nhau bằng dấu “,”
<giá trị>: trả về cho hàm qua lệnh return.
Giá trị trả về của hàm:

- Được xác định dựa vào mục đích của hàm.
- Trong thân hàm, phải trả về đúng <kiểu trả về> đã định ban đầu.
- Nếu các hàm không trả về giá trị thì phải khai báo kiểu trả về của hàm là void.

1. Khái niệm hàm trong C++

❖ **Gọi hàm:**

Khi một hàm được gọi thì điều khiển chương trình sẽ chuyển đến hàm đó. Một hàm được gọi thực hiện một nhiệm vụ đã xây dựng và khi câu lệnh trả về của nó được thực hiện hoặc khi nó kết thúc bằng hàm đóng, nó sẽ trả về chương trình điều khiển quay trở lại chương trình chính.

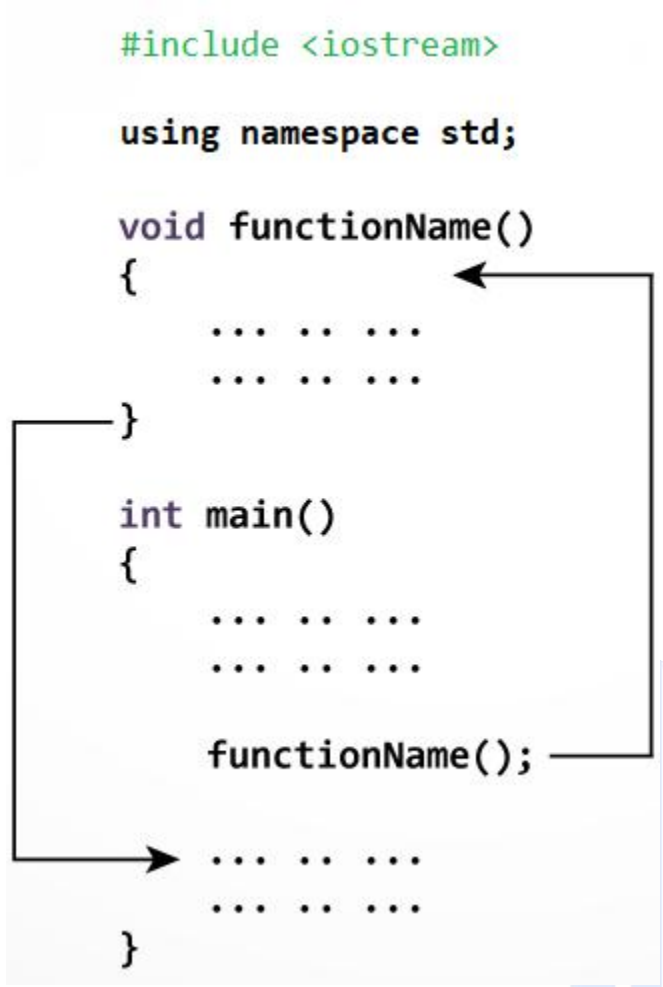
Cú pháp gọi hàm:

<tên hàm>([danh sách tham số]);

Trong đó:

<tên hàm>: tên hàm đã khai báo trước đó

[danh sách tham số]: tham số truyền lúc gọi hàm, tuỳ theo hàm đã xây dựng có tham số hoặc không có tham số.



Lưu ý: Trong C++, hàm cần được khai báo trước rồi mới gọi hàm.

1. Khái niệm hàm trong C++

Ví dụ 1: Viết chương trình hiển thị các số từ 1 tới 10, sử dụng hàm.

```
#include <iostream>
using namespace std;
void hien_thi() {
    for (int i = 1; i <= 10; i++) {
    cout << i << " ";
    }
}
int main() {
    hien_thi();
    return 0;
}
```

➢ Kết quả:

1 2 3 4 5 6 7 8 9 10

```
#include <iostream>
using namespace std;
int main() {
    hien_thi();
    return 0;
}
void hien_thi() {
    for (int i = 1; i <= 10; i++) {
    cout << i << " ";
    }
}
```

➢ Kết quả:

[Error] 'hien_thi' was not declared in this scope

2. Các loại hàm trong C++

2. Các loại hàm trong C++

Trong C++, các loại hàm mà người lập trình có thể tự định nghĩa:

- Hàm không có giá trị trả về:
  - + Không có tham số
  - + Có tham số
- Hàm có giá trị trả về:
  - + Không có tham số
  - + Có tham số

```
<kiểu trả về> <tên hàm>([danh sách tham số]) {
        <các câu lệnh trong hàm>
        return <giá trị>;
}
```

2. Các loại hàm trong C++

1. Hàm không có giá trị trả về

Đây là loại hàm không có sử dụng lệnh return và kiểu dữ liệu khai báo cho hàm là void, thường được dùng trong trường hợp xử lý dữ liệu ngay bên trong hàm đó.

Cú pháp:

```
void  functionName(parameter){
    // Các lệnh trong hàm
}
```

Trong đó:

functionName: tên hàm

parameter: tham số

- Có tham số
- Không có tham số

Ví dụ:

```
#include <iostream>
using namespace std;
void  hienthi() {
    cout << "Hello World!" << endl;
}
int main() {
    hienthi();
    return 0;
}
```

2. Các loại hàm trong C++

1. Hàm không có giá trị trả về

a) Không có tham số: Trong hàm không có giá trị trả về đã được định nghĩa và không có tham số nào trong trường parameter thì gọi là hàm không có giá trị trả về và không có tham số.

```
#include <iostream>
using namespace std;
void  hienthi() {
      cout << "Hello World!" << endl;
      cout << “C++ programming language” <<endl;
}
int main() {
      hienthi();
      return 0;
}
```

Kết quả: in ra màn hình

Hello World!

C++ programming language

2. Các loại hàm trong C++

1. Hàm không có giá trị trả về

b) Có tham số: Trong hàm không có giá trị trả về đã được định nghĩa và có tham số trong trường parameter thì gọi là hàm không có giá trị trả về và có tham số.

Ví dụ:

```
#include <iostream>
using namespace std;
void tong(int x, int y) {
    cout << "Tong hai so la: " << x + y << endl;
}
```

```
int main() {
        int a = 3;
        int b = 5;
        int c = 7;
        int d = 9;
        tong(a,b);
        tong(b,c);
        tong(c,d);
        tong(a,d);
    return 0;
}
```

Kết quả:

```
Tong hai so la: 8
Tong hai so la: 12
Tong hai so la: 16
Tong hai so la: 12
```

2. Các loại hàm trong C++

2. Hàm có giá trị trả về

Đây là loại hàm có kiểu dữ liệu khai báo cho hàm (int, float, ...) và có sử dụng lệnh return ở cuối hàm để trả về giá trị theo kiểu dữ liệu đã khai báo cho hàm. Thường được dùng trong trường hợp muốn hàm trả về một giá trị khi được gọi để xử lý tiếp trong chương trình gọi hàm.

Cú pháp:

```
datatype  functionName(parameter) {
    //Các lệnh trong hàm;
    return value;
}
```

Trong đó:

datatype: kiểu dữ liệu (int, float, ...)

functionName: tên hàm

parameter: tham số

- Có tham số
- Không có tham số

return value : trả về giá trị theo kiểu dữ liệu đã khai báo cho hàm

2. Các loại hàm trong C++

2. Hàm có giá trị trả về

a) Có tham số: Trong hàm có giá trị trả về đã được định nghĩa và có khai báo tham số trong trường parameter thì gọi là hàm có giá trị trả về và có tham số.

Ví dụ:

```
#include <iostream>
using namespace std;
int Tong(int a, int b) {
        return a + b;
}
int Hieu(int a, int b) {
        return a - b;
}
int Tich(int a, int b) {
        return a*b;
}
```

```
int main() {
    int x = 7;
    int y = 5;
    cout << "Tong hai so la: " << Tong(x, y) << endl;
    cout << "Hieu hai so la: "<< Hieu(x, y) << endl;
    cout << "Tich hai so la: "<< Tich(x, y) << endl;
    return 0;
}
```

Kết quả:

Tong hai so la: 12

Hieu hai so la: 2

Tich hai so la: 35

2. Các loại hàm trong C++

2. Hàm có giá trị trả về

b) Không có tham số: Trong hàm có giá trị trả về đã được định nghĩa và không có khai báo tham số trong trường parameter thì gọi là hàm có giá trị trả về và không có tham số.

```
#include <iostream>
using namespace std;
int Nhapso() {
	int n;
	cout <<"Nhap vao so nguyen n: ";
	cin >> n;
	return n;
}
int main() {
	int a;
	a = Nhapso();
	cout << "Tinh n binh phuong: "<< a*a << endl;
	return 0;
}
```

Kết quả:

Nhap vao so nguyen n: 5

Tinh n binh phuong: 25

3. Biến toàn cục và biến cục bộ

3. Biến toàn cục và biến cục bộ trong chương trình C++

1. Biến cục bộ (local variable)

Một biến được khai báo trong hàm (bên trong thân hàm giữa cặp dấu ngoặc nhọn { }) được gọi là biến cục bộ. Phạm vi của biến cục bộ chỉ giới hạn trong hàm mà biến được định nghĩa. Tức là biến cục bộ chỉ tồn tại và chỉ có thể được truy cập bên trong hàm. Biến cục bộ sẽ bị hủy khi hàm kết thúc.

```
#include <iostream>
using namespace std;
int Nhapso() {
	int n;  //biến cục bộ trong hàm Nhapso()
	cout <<"Nhap vao so nguyen n: ";
	cin >> n;
	return n;
}
int main() {
	int a; //biến cục bộ trong hàm main()
	a = Nhapso();
	cout << "Tinh n binh phuong: "<< a*a << endl;
	return 0;
}
```

3. Biến toàn cục và biến cục bộ trong chương trình C++

1. Biến cục bộ (local variable)

Một biến được khai báo trong một khối lệnh trong cặp dấu ngoặc nhọn { }. Biến này cũng chỉ được sử dụng trong khối lệnh đó cũng được xem là biến cục bộ.

Ví dụ:

```
for(int i=0;i<5;i++){
    cout<<i<<endl;
}
```

Biến i chỉ được sử dụng trong vòng lặp for, khi thực thi xong vòng lặp for thì biến i cũng sẽ bị hủy.

3. Biến toàn cục và biến cục bộ trong chương trình C++

2. Biến toàn cục (global variable)

Nếu một biến được định nghĩa ở bên ngoài của tất cả các hàm thì chúng được gọi là biến toàn cục. Phạm vi của biến toàn cục là trong toàn bộ chương trình, tất cả các hàm trong chương trình đều có thể sử dụng biến toàn cục. Biến toàn cục có thể được sử dụng và bị thay đổi giá trị trong bất cứ đâu trong chương trình sau khi được khai báo. Biến toàn cục chỉ bị hủy khi chương trình kết thúc.

```
#include <iostream>
using namespace std;
int n;  // biến toàn cục, các hàm trong chương trình đều sử dụng được biến này
int Nhapso() {
	cout <<"Nhap vao so nguyen n: ";
	cin >> n;
	return n;
}
int main() {
	Nhapso();
	cout << "Tinh n binh phuong: "<< n*n << endl;
	return 0;
}
```

3. Biến toàn cục và biến cục bộ trong chương trình C++

3. Phạm vi của biến (variable scope)

```
#include <iostream>
using namespace std;
int n;
int  Func1() {
        int  b;
        ...
}
int  Func2() {
        int  c;
        {
              int d;
        }
}

int main() {
        int e;
  }
```

3. Biến toàn cục và biến cục bộ trong chương trình C++

3. Phạm vi của biến (variable scope)

```
#include <iostream>
using namespace std;
int Nhapso() {
    int n;  //biến cục bộ trong hàm Nhapso()
    cout <<"Nhap vao so nguyen n: ";
    cin >> n;
    return n;
}
int main() {
    int a; //biến cục bộ trong hàm main()
    a = Nhapso();
    cout << "Tinh n binh phuong: "<< a*a << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int n;  // biến toàn cục
int Nhapso() {
    cout <<"Nhap vao so nguyen n: ";
    cin >> n;
    return n;
}
int main() {
    Nhapso();
    cout << "Tinh n binh phuong: "<< n*n << endl;
    return 0;
}
```

3. Tham số và đối số trong hàm

3. Tham số và đối số trong hàm

- **Tham số (Parameters)** của hàm là những biến được khai báo trong lúc khai báo hàm. Tham số đóng vai trò là giá trị đầu vào cho hàm.

  Ví dụ 2: Khai báo hàm func() có hai tham số

```
void  func(int param1, int param2) { //param1 và param2 là hai tham số của hàm func.
    // Các lệnh trong hàm
}
```

- **Đối số (Arguments)** là giá trị được truyền vào hàm mỗi khi thực hiện lời gọi hàm. Đối số phải có kiểu dữ liệu phù hợp với tham số của hàm.

  Ví dụ 3: Trong hàm main() thực hiện gọi hàm func và truyền đối số cho hàm func.

```
int main() {
    func(7, 9);     // Gọi hàm func() và truyền đối số. Giá trị 7 và 9 là hai đối số của hàm func.
    return 0;       // Khi đó, giá trị  7 và 9 sẽ được tiếp nhận và lưu trữ tạm thời trong hai tham
                    // số param1 và param2.
}
```

3. Tham số và đối số trong hàm

1. Truyền tham trị cho hàm

Truyền đối số cho hàm ở dạng giá trị được gọi là truyền tham trị cho hàm. Có thể truyền hằng, biến, biểu thức nhưng tham số của hàm sẽ chỉ nhận giá trị.

Để lấy giá trị đầu ra (output) của hàm thì cần:

- Khai báo hàm có kiểu dữ liệu trả về như: int, float, string, ...
- Sử dụng từ khóa return bên trong hàm.
- Sử dụng lệnh gọi hàm trong hàm main() như sau:

<tên biến> = <tên hàm>([danh sách đối số]);

```
void  func(int param1, int param2) {
        // Các lệnh trong hàm
    }
```

```
int main() {
        func(7, 9);
        return 0;
    }
```

3. Tham số và đối số trong hàm

1. Truyền tham trị cho hàm

Ví dụ 4(a): Viết chương trình cộng hai số nguyên được nhập vào từ bàn phím trong lúc chạy chương trình và in kết quả ra màn hình. Yêu cầu viết hàm con thực hiện chức năng cộng hai số nguyên và sử dụng phương pháp truyền tham trị cho hàm.

```
#include <iostream>
using namespace std;

int cong2so(int x, int y) {
	int tong = x + y;
	return tong;
}

int main() {
	int a, b, sum;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	sum = cong2so(a, b);
	cout << "Tong hai so la: " <<sum <<endl;
	return 0;
}
```

3. Tham số và đối số trong hàm

1. Truyền tham trị cho hàm

Ví dụ 4(b): Cho chương trình C++ sử dụng phương pháp truyền tham trị cho hàm. Cho biết kết quả chạy chương trình.

Kết quả chạy chương trình ?

```
#include <iostream>
using namespace std;

void  value(int y) {
	cout << "y = " << y << endl;
	y = 25;
	cout << "y = " << y << endl;
}

int main() {
	int x = 5;
	cout << "x = " << x << endl;

	value(x);

	cout << "x = " << x << endl;

	return 0;
}
```

3. Tham số và đối số trong hàm

1. Truyền tham trị cho hàm

❖ **Ưu điểm:**

- Đối số có thể là biến (vd: x, y), hằng (vd: 1, 2), biểu thức (vd: x + 1), arrays, structs, classes
- Đối số không bị thay đổi sau lời gọi hàm, hạn chế tác động không mong muốn của hàm lên đối số.

❖ **Nhược điểm:**

- Gây tốn thêm vùng nhớ do hàm phải tạo các tham số là bản sao của các đối số.
- Gây giảm hiệu suất trong trường hợp đối số là kiểu cấu trúc (structs) hoặc các lớp (classes), đặc biệt là nếu hàm đó được gọi nhiều lần. Vì mỗi lần gọi hàm đều phải sao chép giá trị của đối số vào tham số của hàm.
- Hàm chỉ có thể trả về một giá trị duy nhất bằng câu lệnh return.

3. Tham số và đối số trong hàm

1. Truyền tham trị cho hàm

❖ Khi nào nên sử dụng:

- Khi đối số là các kiểu dữ liệu cơ bản.
- Khi không có nhu cầu thay đổi giá trị của đối số sau khi thực hiện hàm.

❖ Khi nào không nên sử dụng:

- Khi đối số là các mảng (arrays), kiểu cấu trúc (structs), hoặc các lớp (classes).

Trong đa số trường hợp, truyền giá trị cho hàm (Passing arguments by value) là phương pháp thường được sử dụng nhất, vì tính linh hoạt (truyền đối số ở nhiều dạng) và an toàn (đối số không bị thay đổi bởi hàm) của nó.

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

Truyền đối số cho hàm ở dạng địa chỉ (con trỏ) được gọi là truyền tham chiếu cho hàm. Truyền tham chiếu được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm. Phương pháp truyền tham chiếu cho hàm có ưu điểm tiết kiệm bộ nhớ và hoạt động nhanh hơn.

Để lấy giá trị đầu ra (output) của hàm thì cần:

- Hàm không có kiểu trả về (void) và không có sử dụng từ khóa return bên trong hàm.
- Các đối số trong hàm được khai báo bắt đầu bằng toán tử & (gọi là biến tham chiếu)
- Khi gọi hàm, phải truyền vào tên biến, không được truyền giá trị cho tham số này.
- Sử dụng lệnh gọi hàm trong hàm main() như sau:

<tên hàm>([danh sách đối số]);

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

❖ Biến tham chiếu:

Trong C++, tham chiếu (reference) là một loại biến hoạt động như một bí danh của biến khác.

Khai báo bằng cách sử dụng ký hiệu “&” đặt giữa kiểu dữ liệu và tên biến (vd: int &a)

Mọi thay đổi trên biến tham chiếu cũng chính là thay đổi trên biến được tham chiếu đến.

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

Ví dụ 5 (a): Viết chương trình cộng hai số nguyên được nhập vào từ bàn phím trong lúc chạy chương trình và in kết quả ra màn hình. Yêu cầu viết hàm con thực hiện chức năng cộng hai số nguyên và sử dụng phương pháp truyền tham chiếu cho hàm.

```
#include <iostream>
using namespace std;

void cong2so(int x, int y, int &tong)
{
	tong = x + y;
}

int main() {
	int a,b,sum;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	cong2so(a,b,sum);
	cout << "Tong hai so la: " <<sum <<endl;
	return 0;
}
```

3. Tham số và đối số trong hàm

❖ Sử dụng phương pháp truyền tham trị cho hàm.

```
#include <iostream>
using namespace std;

int cong2so(int x, int y) {
	int tong = x + y;
	return tong;
}

int main() {
	int a,b,sum;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	sum = cong2so(a, b);
	cout << "Tong hai so la: " <<sum <<endl;
	return 0;
}
```

❖ Sử dụng phương pháp truyền tham chiếu cho hàm.

```
#include <iostream>
using namespace std;

void cong2so(int x, int y, int &tong)
{
	tong = x + y;
}

int main() {
	int a,b,sum;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	cong2so(a,b,sum);
	cout << "Tong hai so la: " <<sum <<endl;
	return 0;
}
```

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

Ví dụ 5 (b): Cho chương trình C++ sử dụng phương pháp truyền tham chiếu cho hàm. Cho biết kết quả chạy chương trình.

Kết quả chạy chương trình ?

```
#include <iostream>
using namespace std;

void  value(int &y) {
	cout << "y = " << y << endl;
	y = 25;
	cout << "y = " << y << endl;
}

int main() {
	int x = 5;
	cout << "x = " << x << endl;

	value(x);

	cout << "x = " << x << endl;

	return 0;
}
```

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

Ví dụ 6: Viết chương trình hoán đổi giá trị giữa 2 biến số nguyên được nhập vào từ bàn phím trong lúc chạy chương trình và in kết quả ra màn hình. Yêu cầu viết hàm con thực hiện chức năng hoán đổi hai số nguyên và sử dụng phương pháp truyền tham chiếu cho hàm.

```
#include <iostream>
using namespace std;

void hoan_doi(int &x, int &y) {
  int z = x;
  x = y;
  y = z;
}
int main() {
  int a,b;
  cout << "Nhap so nguyen thu nhat: ";
  cin >> a;
  cout << "Nhap so nguyen thu hai: ";
  cin >> b;
  hoan_doi(a, b);
  cout << "So nguyen thu nhat sau khi hoan doi: " << a <<endl;
  cout << "So nguyen thu hai sau khi hoan doi: " << b <<endl;
  return 0;
}
```

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

Đôi khi bạn cần một hàm trả về nhiều giá trị. Tuy nhiên, trong truyền tham trị thì hàm chỉ có một giá trị trả về. Một trong những cách để hàm trả về nhiều giá trị là sử dụng truyền tham chiếu cho hàm.

```
#include <iostream>
using namespace std;

void calculator(int x, int y, int &addOut, int &subOut)
{
	addOut = x + y;
	subOut = x - y;
}

int main()
{
	int a = 7, b = 9;
	int add, sub;

	calculator(a, b, add, sub);

	cout << " Tong la = " << add << endl;
	cout << " Hieu la = " << sub << endl;

	return 0;
}
```

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

❖ **Ưu điểm:**

- Hàm có thể thay đổi giá trị của các đối số.
- Không mất thời gian và bộ nhớ để sao chép giá trị của đối số vào tham số của hàm.
- Hàm có thể trả về nhiều giá trị thông qua tham chiếu.

❖ **Nhược điểm:**

- Truyền tham chiếu cho phép hàm thay đổi giá trị của các đối số (arguments), điều này sẽ là nguy hiểm tiềm ẩn nếu bạn chỉ muốn đọc các đối số đó (read only).

3. Tham số và đối số trong hàm

2. Truyền tham chiếu cho hàm

❖ **Khi nào nên sử dụng:**

- Khi đối số là kiểu cấu trúc (structs) hoặc các lớp (classes).
- Khi có nhu cầu thay đổi giá trị của đối số sau khi thực hiện hàm.

❖ **Khi nào không nên sử dụng:**

- Khi đối số có kiểu dữ liệu cơ bản (thì nên sử dụng truyền tham trị).

3. Tham số và đối số trong hàm

3. Đối số mặc định của hàm

Tham số mặc định là tham số có một giá trị khởi tạo tại thời điểm khai báo hàm.

Khi gọi hàm, nếu người dùng không cung cấp đối số cho tham số mặc định thì giá trị mặc định sẽ được sử dụng. Khi người dùng cung cấp đối số cho tham số mặc định, tham số sẽ được gán lại bằng giá trị của đối số được truyền vào.

Để khai báo tham số mặc định cho hàm, sử dụng toán tử gán "=".

Ví dụ 7:

```
int  cong2so(int x, int y = 7) {
        ...
}
```

Tham số thứ 2 của hàm cong2so có giá trị mặc định nên khi gọi hàm, có thể chỉ truyền vào 1 đối số cho tham số x. Hoặc truyền đối số cho cả tham số x và y.

3. Tham số và đối số trong hàm

3. Đối số mặc định của hàm

Ví dụ 7:

```
#include <iostream>
using namespace std;

int cong2so(int x, int y=7) {
	int tong = x + y;
	return tong;
}

int main() {
	int tong1, tong2;
	tong1 = cong2so(5);
	cout << "Tong hai so la: " << tong1 <<endl;

	tong2 = cong2so(5,9);
	cout << "Tong hai so la: " << tong2 <<endl;
	return 0;
}
```

Kết quả:

Tong hai so la: 12

Tong hai so la: 14

3. Tham số và đối số trong hàm

3. Đối số mặc định của hàm

Trường hợp khai báo tham số mặc định không hợp lệ:

```
void add(int x = 10, int y) { ... }
void add(int a, int b = 3, int c, int d) { ...}
void add(int a, int b = 3, int c, int d = 4) { ... }
```

Trường hợp khai báo tham số mặc định hợp lệ:

```
void add(int y, int x = 10) { ... }
void add(int a, int c, int d, int b = 3) { ...}
void add(int a, int c, int b = 3, int d = 4) { ... }
```

Lưu ý: Mọi tham số mặc định khi khai báo phải đặt phía sau tham số không có giá trị mặc định.

3. Tham số và đối số trong hàm

3. Đối số mặc định của hàm

Ví dụ:

```
#include <iostream>
using namespace std;
void tong(int a = 7, int b = 5, int c) {
    cout << "Tham so thu 1: " << a << endl;
    cout << "Tham so thu 2: " << b << endl;
    cout << "Tham so thu 3: " << c << endl;
    cout << "Tong: " << a+b+c << endl << endl;
}
int main() {
        int x = 1;
        int y = 2;
        int z = 3;
        tong(x, y, z);
        tong(x,y);
        tong(x);
        return 0;
}
```

Kết quả: ??

3. Tham số và đối số trong hàm

3. Đối số mặc định của hàm

Ví dụ:

```
#include <iostream>
using namespace std;
void tong(int c, int a = 7, int b = 5) {
    cout << "Tham so thu 1: " << a << endl;
    cout << "Tham so thu 2: " << b << endl;
    cout << "Tham so thu 3: " << c << endl;
    cout << "Tong: " << a+b+c << endl << endl;
}
```

```
int main() {
        int x = 1;
        int y = 2;
        int z = 3;
        tong(x, y, z);
        tong(x,y);
        tong(x);
        return 0;
}
```

Kết quả: ??

3. Tham số và đối số trong hàm

4. Truyền mảng cho hàm

Truyền mảng dưới dạng đối số của hàm.

Ví dụ 8: Viết chương trình in ra màn hình các phần tử trong mảng đã cho trước. Yêu cầu viết hàm con thực hiện chức năng in các phần tử ra màn hình.

Lưu ý: Khi bạn gọi hàm thì chỉ cần sử dụng tên của mảng, array_print(a);

```
#include <iostream>
using namespace std;
void array_print(int arr[5]) {
    for (int i = 0; i < 5; i++) {
        cout << arr[i] << endl;
    }
}
int main() {
    int a[5] = {10, 20, 30, 40, 50};
    array_print(a);
    return 0;
}
```

4. Nạp chồng hàm

4. Nạp chồng hàm

Nạp chồng hàm (Function Overloading):

Trong lập trình C++, hai hàm có thể trùng tên với nhau nếu danh sách tham số của hàm là khác nhau. Danh sách tham số của hàm khác nhau về số lượng tham số hoặc kiểu dữ liệu của tham số.

Những hàm có cùng tên nhưng khác danh sách tham số thì gọi là nạp chồng hàm.

Lưu ý: Kiểu trả về của các hàm nạp chồng có thể giống hoặc khác nhau. Nhưng danh sách tham số phải là khác nhau về số lượng tham số hoặc kiểu dữ liệu của tham số.

Ví dụ 9: Các hàm cùng tên nhưng khác danh sách tham số

```
int  test() { ... }
int  test(int a) { ... }
float  test(double a) { ... }
int  test(int a, double b) { ... }
```

Ví dụ 10: Khai báo hàm nạp chồng lỗi vì kiểu dữ liệu của tham số giống nhau (mặc dù hàm khác kiểu trả về).

```
int  test(int a) { ... }
double  test(int b) { ... }
```

4. Nạp chồng hàm

Ví dụ 11: Viết chương trình tính giá trị tuyệt đối của một số nguyên và số thực. Sử dụng phương pháp nạp chồng hàm với kiểu dữ liệu của tham số khác nhau.

```
#include <iostream>
using namespace std;
float absolute(float var){
        if (var < 0.0) {
                var = -var;
        }
        return var;
}
int absolute(int var) {
        if (var < 0) {
                var = -var;
        }
        return var;
}
int main() {
        cout << "Gia tri tuyet doi cua -7 = " << absolute(-7) << endl;
        cout << "Gia tri tuyet doi cua -9.5 = " << absolute(-9.5f) << endl;
        return 0;
}
```

Khi gọi một hàm nạp chồng, thì trình biên dịch quyết định chọn hàm thích hợp nhất để sử dụng bằng việc so sánh các kiểu tham số gọi hàm với các kiểu tham số đã được xác định trong các hàm.

Tiến trình lựa chọn hàm nạp chồng thích hợp nhất này được gọi là phân giải nạp chồng (overload resolution)

4. Nạp chồng hàm

Ví dụ 12: Sử dụng nạp chồng hàm với số lượng tham số và kiểu dữ liệu của tham số khác nhau.

```
#include <iostream>
using namespace std;
void display(int var1, double var2) {
        cout << "So nguyen la: " << var1;
        cout << " va so thuc la: " << var2 << endl;
}
void display(double var) {
        cout << "So thuc la: " << var << endl;
}
void display(int var) {
        cout << "So nguyen la: " << var << endl;
}
int main() {
        int a;
        double b;
        cout << "Nhap vao so nguyen: ";
        cin >> a;
        cout << "Nhap vao so thuc: ";
        cin >> b;
        display(a);
        display(b);
        display(a, b);
        return 0;
}
```

Bài tập

Câu 1: Viết chương trình C++ nhập vào 2 số nguyên từ bàn phím trong lúc chạy chương trình và cho phép chọn phép toán để tính toán trên hai số này như sau:

1: cộng/ 2: trừ/ 3: nhân/ 4: chia/ 5: exit. Viết các hàm cho từng phép toán.

Câu 2: Viết chương trình C++ nhập vào 2 số nguyên từ bàn phím trong lúc chạy chương trình, tìm giá trị lớn và nhỏ hơn giữa hai số này, sử dụng hàm (viết hàm tìm min và hàm tìm max). Sau đó in ra kết quả ra màn hình.

Câu 3: Viết chương trình C++ nhập và họ tên học sinh, điểm môn Toán, điểm môn Lý, điểm môn Anh của 1 học sinh trong lúc chạy chương trình. Tính điểm trung bình và xuất ra kết quả. Viết các hàm con gồm: Nhap(), Xuly(), Xuat() được gọi trong hàm chính.

Câu 4: Viết chương trình C++ nhập vào số nguyên n từ bàn phím trong lúc chạy chương trình, in ra màn hình các số chia hết cho 3 nằm trong khoảng từ 1 tới n (viết hàm con Xuly()).

Bài tập

Câu 5: Viết chương trình C++ nhập vào số nguyên dương n. Tính tổng dãy số từ 1 đến n;

s = 1 + 2 + 3 + … + n (viết hàm con tính tổng dãy số), sau đó in kết quả tổng ra màn hình.

Câu 6: Viết chương trình C++ sử dụng hàm tìm giá trị lớn nhất trong 3 số nguyên được nhập vào từ bàn phím trong lúc chạy chương trình.