Lập trình hướng đối tượng và C++

Bài 3: Hàm trong C++

TS. Nguyễn Hiếu Cường
Bộ môn CNPM, Khoa CNTT
Trường Đại học GTVT
Email: cuonggt@gmail.com

Hà Nội, 2019

Nội dung chính

1. Giới thiệu môn học
2. Các khái niệm cơ bản
3. **Hàm trong C++**
4. Lớp và đối tượng
5. Định nghĩa chồng toán tử
6. Hàm tạo và hàm huỷ
7. Dẫn xuất và thừa kế
8. Tương ứng bội
9. Khuôn hình

Hàm trong C++

- Các đặc điểm của hàm
- Tham số (đối) và truyền tham số
- Biến con trỏ và biến tham chiếu
- Đối có giá trị mặc định
- Định nghĩa chồng hàm

Các đặc điểm của hàm trong C

- Các hàm ngang cấp nhau và luôn có duy nhất một hàm main()
- Chương trình bắt đầu thực hiện từ hàm main()

```
int main()
{
  ...
  return 0;       // kết thúc thành công, mặc định (có
  thể bỏ)
}
```

Tham số của hàm

- Các loại đối (tham số) của hàm
  - Tác dụng của các đối trong hàm?
  - Đối là **biến thông thường**
  - Đối là **biến con trỏ**
- Truyền tham số
  - Đối là biến thông thường → Truyền theo giá trị
    - Khi gọi hàm: đối chứa bản sao của tham số thực sự tương ứng
  - Đối là biến con trỏ → Truyền theo địa chỉ
    - Khi gọi hàm: đối chứa địa chỉ của tham số thực sự tương ứng

Ví dụ (truyền tham số)

```
// Hoán vị hai số - Ví dụ sai
void hoan_vi(float a, float b) {
  float t;
  t=a; a=b; b=t;
}
int main() {
  float x=5, y=10;
  hoan_vi(x,y);
  printf("x = %f, y = %f", x, y);
}
```

```
// Hoán vị hai số - Ví dụ đúng: dùng con trỏ
void hoan_vi(float *a, float *b) {
  float t;
  t=*a; *a=*b; *b=t;
}
int main() {
  float x=5, y=10;
  hoan_vi(&x, &y);
  printf("x = %f, y = %f", x, y);          }
```

Ví dụ

- Xây dựng hàm tính n!

```
int gt(int n)
```

- Xây dựng hàm giải phương trình bậc 2

```
int ptb2(float a,float b,float c,float *x1,float *x2)
```

- Có nhận xét gì về các đối của các hàm trên?
  - Đối giá trị: tham số đầu vào
  - Đối con trỏ: tham số đầu ra

Ví dụ

- Kết quả của chương trình sau?

```
#include <stdio.h>
void doi(int *a, int *b)
{
    *a -= 1;    *a = ++(*b);   *b -= 5;
}
int main()
{
    int x = 1, y = 2;
    doi(&x, &y);
    printf("%d  %d", x, y);                    3  -2
}
```

Biến tham chiếu

- C có hai loại biến
  - **Biến thông thường**: chứa một giá trị
  - **Biến con trỏ**: chứa một địa chỉ
- C++ có thêm
  - **Biến tham chiếu**
    - Không tồn tại độc lập
    - Là bí danh (alias) của một biến khác

      int a = 10;

      int &r = a; → r chính là a, do đó r cũng có giá trị bằng 10
    - Biến tham chiếu thường dùng làm đối của hàm

      void hoan_vi(float &a, float &b);

Ví dụ (truyền tham số)

```
// Hoán vị hai số - Ví dụ sai
void hoan_vi(float a, float b) {
  float t;
  t=a; a=b; b=t;
}
int main() {
  float x=5, y=10;
  hoan_vi(x,y);
  cout<<"x= "<<x<<" y= "<<y;
}
```

```
// Hoán vị hai số - Ví dụ đúng: dùng con trỏ
void hoan_vi(float *a, float *b) {
  float t;
  t=*a; *a=*b; *b=t;
}
int main() {
  float x=5, y=10;
  hoan_vi(&x, &y)
  cout<<"x= "<<x<<" y= "<<y;
}
```

Biến con trỏ và biến tham chiếu

```
// Dùng biến con trỏ

void hoan_vi(float *a, float *b)
  {
    float t=*a;
    *a=*b; *b=t;
   }

int main()
   {
    float x=5, y=10;
    hoan_vi(&x, &y)
    cout<<"x= "<<x<<" y= "<<y;

   }
```

```
// Dùng biến tham chieu

void hoan_vi(float &a, float &b)
 {
    float t=a;
    a=b; b=t;
 }

int main()
 {
    float x=5, y=10;
    hoan_vi(x, y)
    cout<<"x= "<<x<<" y= "<<y;
 }
```

Ví dụ (cho biết kết quả)

```
void test(int& a, int b)
{
  a+=b;
  b=a;
}

int main()
{
  int x=1, y=2;
  test(x,y);
  cout<<x<<endl<<y;
}
```

```
int& f(int &a)
{
  a++;
  return a;
}

int main()
{
  int i=5;
  int &r= f(i);
  r++;
  cout<<i;
}
```

Tóm tắt về truyền tham số

Call Type	Description
Call by value	This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
Call by pointer	This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.
Call by reference	This method copies the reference of an argument into the formal parameter. Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

Chú ý

Trong C một số ký hiệu có thể có ý nghĩa khác nhau tùy theo tình huống mà nó xuất hiện.

```
int i = 42;
int &r = i;    //  & follows a type and is part of a declaration; r is a reference
int *p;        //  * follows a type and is part of a declaration; p is a pointer
p = &i;        //  & is used in an expression as the address-of operator
*p = i;        //  * is used in an expression as the dereference operator
int &r2 = *p; //  & is part of the declaration; * is the dereference operator
```

Đối có giá trị mặc định

- Số tham số trong lời gọi hàm = số đối của hàm
- Đối có giá trị mặc định

  Tham số tương ứng với đối có giá trị mặc định có thể vắng mặt trong lời gọi hàm

```
void delay(int n=1000) {
  for (int i=0 ; i<n ; ++i) ;
}

int main() {
  delay(5000);
  delay();    // tuong duong delay(1000)
}
```

Xây dựng hàm có đối mặc định

- Nếu hàm có nhiều đối, trong đó có đối có giá trị mặc định?
- Qui tắc xây dựng
  - Các đối mặc định cần phải là các đối cuối cùng tính từ trái sang phải
  - Ví dụ: một hàm có 5 đối theo thứ tự từ trái sang phải là d1, d2, d3, d4, d5 thì
    - nếu một đối mặc định thì phải là d5
    - nếu hai đối mặc định thì phải là d4, d5
    - nếu ba đối mặc định thì phải là d3, d4, d5
    - ...

Ví dụ (đối mặc định)

```
#include <iostream>
using namespace std;
void ht(char *dc="HA NOI",int n=10) ;
int main(){
  ht();
  ht("ABC",3);
  ht("DEF");
}
void ht(char *dc , int n ) {
  for (int i=0;i<n;++i)
    cout << "\n" << dc;
}
```

Định nghĩa chồng các hàm

- Định nghĩa chồng hàm hoặc tải bội (function overloading)
  - Dùng cùng một tên để định nghĩa các hàm khác nhau
- Các hàm phải có gì để phân biệt với nhau?
  - Các hàm định nghĩa chồng được phân biệt nhờ **có tập đối khác nhau**
  - Trình biên dịch sẽ căn cứ vào tập đối trong các lời gọi hàm
- Định nghĩa chồng toán tử

  Tương tự như với hàm, thay **tên hàm** bằng **operator <tên toán tử>**

  Sẽ trình bày kỹ hơn trong bài sau.

Tóm tắt

- Các đặc điểm của hàm
- Tham số (đối) và truyền tham số
- Biến con trỏ và biến tham chiếu
- Đối có giá trị mặc định
- Định nghĩa chồng hàm

Xét kết quả của chương trình

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     short number;
8     cout << "Enter a number: ";
9     cin >> number;
10
11    cout << "The factorial of " << number << " is ";
12    int accumulator = 1;
13    for(; number > 0; accumulator *= number--);
14    cout << accumulator << ".\n";
15
16    return 0;
17 }
```

Bài tập

- Tìm lỗi sai:

```
int n;
cin>>n;
for (int i=0; i<n; ++i)
{
  int a[100];
  cin>>a[i];
}
for (i=0; i<n; ++i)
  cout<< a[i];
...
```

Bài tập

- Xét kết quả của chương trình sau:

```
int& f(int &a, int b) {
  b += a;
  if (b>5) a++;
  return a;
}
int main() {
  int i=2, j=4;
  int k= f(i,j);
  k++;
  cout<<i<<"  "<<j<<"  "<<k<<endl;
  int &m= f(i,j);
  m++;
  cout<<i<<"  "<<j<<"  "<<m;
}
```

Bài tập

- Xét kết quả của chương trình sau:

```
int& f(int &a, int &b) {
  b += a;
  if (b>5) a++;
  return a;
}
int main() {
  int i=2, j=4;
  int &k= f(i,j);
  k++;
  cout<<i<<"  "<<j<<"  "<<k<<endl;
  int &m= f(i,j);
  m++;
  cout<<i<<"  "<<j<<"  "<<m;
}
```

Bài tập

1. Xây dựng hàm giải phương trình bậc 2, trong đó giá trị trả về thể hiện việc phương trình có 1 hoặc 2 nghiệm hay vô nghiệm

2. Xây dựng hàm xác định một số có phải số nguyên tố không. Áp dụng tìm tất cả các số nguyên tố <= N.

3. Xây dựng hàm tìm UCLN(a,b). Áp dụng tìm UCLN của một dãy N số nguyên.

4. Tìm 100 số Fibonacci đầu tiên (có thể xây dựng hàm xác định số Fibonacci thứ n).