Lập trình hướng đối tượng và C++

Bài 3: Hàm trong C++

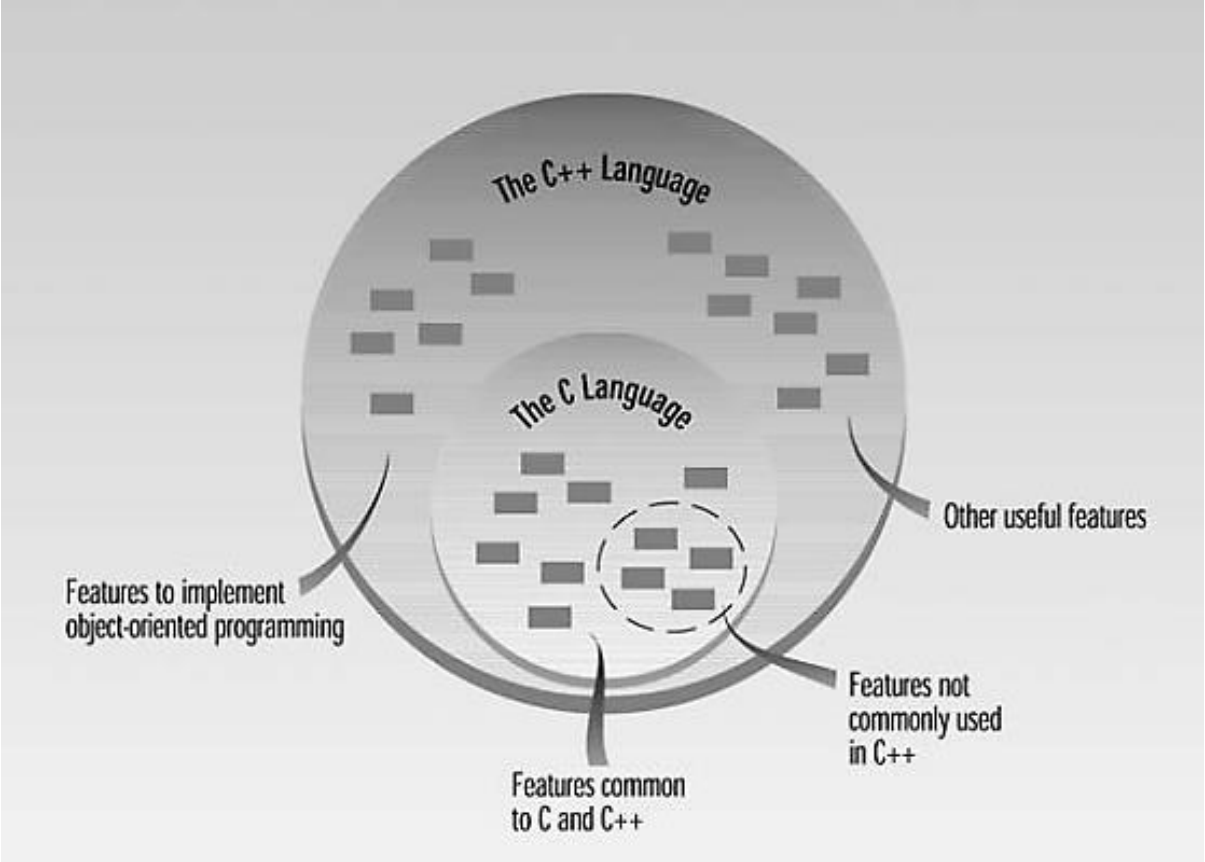
TS. Nguyễn Hiếu Cường
Bộ môn CNPM, Khoa CNTT, Trường Đại học GTVT
Email: cuonggt@gmail.com

Hà Nội, 2022

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

Các khái niệm cơ bản



Khái niệm hàm

- Hàm là một đoạn chương trình để thực hiện một việc nào đó
- Một chương trình C là một dãy các hàm, trong đó có một hàm main( )
  - Các hàm của C ngang cấp nhau, tức là không có hàm bên trong hàm
  - Các hàm có thể gọi nhau
- Chương trình được thực hiện từ hàm main( )

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

Tác dụng của hàm

- Một chương trình C có thể có nhiều hàm
  - Chương trình đơn giản thường chỉ có một hàm main
  - Chương trình lớn, ngoài hàm main có thể cần xây dựng các hàm khác
  - Trong các hàm có thể sử dụng các hàm chuẩn: sqrt(), pow(), sin()...

- Tác dụng của hàm?

Hàm chuẩn và hàm tự tạo

- Hàm chuẩn (built-in functions)
  - Đã có sẵn trong thư viện, khi cần dùng thì include thư viện chứa nó

    Ví dụ: Nhập n và x, tính $s = x + x^2 + x^3 + ... + x^n$

- Hàm tự tạo (user-defined functions)
  - Khi nhu cầu của người dùng không có hàm chuẩn đáp ứng được
  - Cần định nghĩa hàm trước khi sử dụng

    Ví dụ: Nhập n, tính s = 1! + 2! + 3! + ... + n!

Cách xây dựng hàm

1. Xác định xem hàm cần làm gì?
2. Cần đưa dữ liệu gì vào hàm và lấy kết quả gì ra khỏi hàm?
   - Xác định danh sách các tham số (các đối)
   - Cần phân biệt: *đối vào* và *đối ra*
3. Xác định kiểu trả về của hàm
   - Hàm có thể trả về một giá trị (kiểu int, float...)
   - Nếu hàm không định trả về cái gì thì để là kiểu void
4. Đặt tên hàm
   - Tuân thủ quy tắc đặt tên
   - Tên hàm nên thể hiện ý nghĩa mà hàm thực hiện

Cách sử dụng hàm

- Các loại hàm:
  - Hàm chuẩn: Cần khai báo thư viện chứa hàm
  - Hàm tự định nghĩa: Cần định nghĩa trước khi dùng
- Khi sử dụng hàm (gọi hàm) cần:
  - Ghi đúng tên hàm và cung cấp các đủ giá trị cho các đối
- Điều gì xảy ra khi một hàm được gọi?

Ví dụ (tính giai thừa)

```
#include<stdio.h>
int giai_thua(int k) {
   int i, t=1;
   for(i=1; i<=k; ++i)
     t *= i;
   return t;
}
int main() {
   int n, i, s=0;
   printf("n = ");
   scanf("%d", &n);
   for(i=1; i<=n; ++i)
     s += giai_thua(i);
   printf("Tong S = %d", s);
}
```

Ví dụ (tìm các số nguyên tố < N)

```
// In ra màn hình các số nguyên tố có giá trị < 1000.
#include<stdio.h>
int nguyento(int k) {
     int i;
     for(i=2; i*i< k; ++i)
       if(k%i==0)
         return 0;
     return 1;
}
int main() {
     int i;
     printf("Cac so nguyen to <1000 la:\n");
     for(i=2; i<1000; ++i)
       if(nguyento(i)==1)
         printf("%4d", i);
}
```

Tham số của hàm

- Các loại đối (tham số) của hàm
  - Biến thông thường
  - Biến con trỏ
- Truyền tham số?
  - Đối là biến thông thường → Truyền theo giá trị (call by value)
    - Đối chứa bản sao của tham số thực sự tương ứng
  - Đối là biến con trỏ → Truyền theo con trỏ (call by pointer)
    - 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);
}
```

Con trỏ

- Biến
  - Là vùng bộ nhớ (một dãy byte liên tiếp) dùng để lưu dữ liệu
  - Địa chỉ của biến: là địa chỉ của ô nhớ đầu tiên cấp cho biến
- Biến con trỏ (con trỏ)
  - Là loại biến đặc biệt, dùng để chứa địa chỉ của một biến
  - Con trỏ kiểu gì thì chỉ chứa được địa chỉ của biến kiểu đó

```
float *p, x = 10.5;
p = &x;
printf("x = %f", *p);
```

Ví dụ

- Xác định kết quả của đoạn chương trình sau:

```
float x = 5, y = 20, *px, *py;
px = &x;
py = &y;
printf("%d", 3*(*px)+*py);
```

Đối của hàm là con trỏ

- Khi nào thì đối của hàm là con trỏ?
  - Đối bắt buộc phải là con trỏ khi nó đóng vai trò "đầu ra"
  - Đối có thể là con trỏ khi nó đóng vai trò "đầu vào", tuy nhiên dữ liệu đưa vào hàm khi đó sẽ không được an toàn

- Ví dụ: xây dựng hàm giải phương trình bậc 2

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

Giải phương trình bậc 2

```
#include"stdio.h"
#include "math.h"
int ptb2(float a, float b, float c, float *x1, float *x2);
int main()
    {
        int s,ch;
        float a,b,c,x1,x2;
        printf("\n Vao a,b,c");
        scanf("%f%f%f",&a,&b,&c);
        s = ptb2(a,b,c,&x1,&x2);
        if (s == 0)
            printf("\n a = 0 ");
        else if (s == -1)
            printf("\n delta < 0 ");
        else
            printf("\n x1 = %0.2f x2 = %0.2f",x1,x2 );
    }
```

```
/* Hàm giải ph- ơng trình bậc hai */
int ptb2(float a, float b, float c, float *x1, float *x2)
        {
            float delta;
            if (a == 0) return 0;
            delta = b*b-4*a*c;
            if (delta<0) return -1;
            *x1 = (-b - sqrt(delta))/(2*a);
            *x2 = (-b + sqrt(delta))/(2*a);
            return (1);
}
```

Biến tham chiếu

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

```
int a = 10;
int &r = a;
```

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)

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

void getStudent (int &id, int &mark)
{
     cout << "Enter student's id and mark: ";
     cin >> id >> mark;
     return;
}

int main()
{
    int id1, mark1;
    getStudent (id1, mark1);
    cout << "Student " << id1 << " gets mark " << mark1 << endl;
    int id2, mark2;
    getStudent (id2, mark2);
    cout << "Student " << id2 << " gets mark " << mark2 << endl;
    cout << "Total mark: " << (mark1 + mark2);
    return 0;
}
```

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 nghĩa khác nhau trong những ngữ cảnh khác nhau!

```
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();          // In dòng chữ "HA NOI" trên 10 dòng
  ht("ABC",3);   // In dòng chữ "ABC" trên 3 dòng
  ht("DEF");     // In dòng chữ "DEF" trên 10 dòng
}
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 (số đối hoặc kiểu của đối)
  - 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 trong bài sau.

Tóm tắt

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

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 giá trị bằng 0, 1 hoặc 2 thể hiện việc phương trình vô nghiệm, có 1 nghiệm hoặc có 2 nghiệm.
2. Xây dựng hàm xác định một số có phải *số nguyên tố* không. Áp dụng hàm trên để tìm tất cả các số nguyên tố <= N.
3. Xây dựng hàm tìm *ước số chung lớn nhất* của hai số nguyên: UCLN(a,b). Áp dụng để tìm UCLN của một dãy số nguyên.
4. Nhập một dãy số, *sắp xếp dãy* theo thứ tự tăng dần. Yêu cầu: viết hàm sắp xếp, và hàm sắp xếp gọi hàm hoán vị hai số.
5. Tìm 50 *số Fibonacci* đầu tiên:
   - Cách 1: xây dựng hàm tính số Fibonacci thứ k (đệ quy)
   - Cách 2: xây dựng hàm tính số Fibonacci thứ k (lặp)
   - Cách 3: xây dựng hàm xác định một số tự nhiên bất kỳ có phải là số Fibonacci không...