TIN ĐẠI CƯƠNG

Bài 4: CHƯƠNG TRÌNH CON

Trần Thị Ngân

Bộ môn Công nghệ phần mềm, Khoa CNTT

Trường đại học Thủy Lợi

Nội dung chính

1. Phân rã bài toán
2. Hàm
3. Tham chiếu và tham trị
4. Phạm vi của biến (scope)
5. Bài tập

1. Phân rã bài toán

Ví dụ : In ra màn hình các thông số (chu vi, diện tích, diện tích hình tròn ngoại tiếp) của hình chữ nhật có hai cạnh cho trước.

→ có thể chia thành nhiều phần, mỗi phần có mục đích khác nhau :

- tính chu vi hình chữ nhật
- tính diện tích hình chữ nhật
- tính diện tích hình tròn ngoại tiếp
- in ra các thông số của hình chữ nhật

Ví dụ

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

int main()
{
    double x=1., y=3.;
    double chuvi, dientich, dientichHinhtron, r;
    const double pi = 3.14159;

    //tinh cac thong so
    chuvi = 2*(x+y);
    dientich= x*y;
    r = sqrt(x*x + y*y) / 2;
    dientichHinhtron = pi*r*r;

    //in ra thong so
    cout << "Thong so cua hinh chu nhat co hai canh la " << x << " va " << y << endl;
    cout << "\t Chu vi:                              " << chuvi << endl;
    cout << "\t Dien tich:                           " << dientich << endl;
    cout << "\t Dien tich hinh tron ngoai tiep: " << dientichHinhtron << endl;

    return 0;
}
```

Nếu muốn in ra thông số của rất nhiều hình chữ nhật khác nhau thì sao? Nếu cần sửa code thì phải sửa ở rất nhiều chỗ?

Phân rã bài toán

Ưu điểm :

- Một chương trình lớn có thể được chia ra/phân rã thành các chương trình con (hàm). Điều này giúp chương trình lớn dễ hiểu hơn
- Chương trình con được viết một lần, và có thể được sử dụng ở nhiều nơi
- Mỗi hàm con thường nhỏ → việc kiểm tra tính đúng đắn của hàm đó dễ dàng hơn
- Nếu xảy ra lỗi, chỉ cần sửa hàm con mà không phải thay đổi toàn bộ cấu trúc chương trình lớn

2. Hàm

- Hàm: là một đoạn chương trình máy tính thực hiện một nhiệm vụ nào đó và trả về kết quả
- Cú pháp :

<kiểu kết quả> <tên hàm> (<danh sách tham số>)

```
{
    //nội dung của hàm
}
```

<Kiểu kết quả> của hàm

- Có thể là ác kiểu dữ liệu cơ bản của C++ (int, float, bool,...)
- Cũng có thể là các kiểu dữ liệu do người lập trình định nghĩa
- Trả về kiểu "không là kiểu gì" (hàm void):

```
void InGiaTri(int a)
        {
         cout << "a = " << a << endl;
         return; //không bắt buộc
        }
```

<tên hàm>, <danh sách tham số> của hàm

<tên hàm>

- Nguyên tắc đặt tên giống như cách đặt tên biến
- Nên đặt tên hàm thể hiện được nội dung công việc mà hàm làm

<danh sách tham số>

- Là danh sách các biến cần trao đổi giữa hàm và bên ngoài
- Danh sách các tham số có thể rỗng

Ví dụ :

```
void hello()
{
    cout << "Hello world!";
}
```

Khai báo, triển khai, gọi hàm

Cách 1

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

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

int main()
{
   int tong = Cong(4, 9);
   cout << tong << endl;
   return 0;
}
```

Tên hàm: **Cong**

Giá trị trả về: int

Hàm cần 2 biến đầu vào đều là kiểu int

Vì hàm **Cong** khai báo có giá trị trả về là int

nên cần "return" ở cuối hàm

Gọi hàm **Cong**, truyền giá trị đầu vào 4 và 9

Theo thứ tự này ta có x = 4 và y = 9

Khai báo, triển khai, gọi hàm

Cách 2

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

int Cong(int x, int y);

int main()
{
  int tong = Cong(4, 9);
  cout << tong << endl;
  return 0;
}

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

Chỉ khai báo có hàm **Cong**, chưa triển khai nội dung hàm

Cần phải khai báo trước hàm Main vì trong hàm Main có gọi đến hàm Cong

Triển khai nội dung hàm **Cong**

3. Tham chiếu và tham trị

Truyền bằng tham trị

```cpp
#include <iostream>
using namespace std;
void InGiaTri(int a)
{
  a += 1;
  cout << "(Ben trong ham) a = " << a << endl;
  return;
}

int main()
{
  int a = 10;
  cout << "(Truoc khi goi ham) a = " << a << endl;
  InGiaTri(a);
  cout << "(Sau khi goi ham) a = " << a << endl;
  return 0;
}
```

```
(Truoc khi goi ham) a = 10
(Ben trong ham) a = 11
(Sau khi goi ham) a = 10
```

Truyền bằng tham chiếu

```cpp
#include <iostream>
using namespace std;
void InGiaTri(int& a)
{
  a += 1;
  cout << "(Ben trong ham) a = " << a << endl;
  return;
}

int main()
{
  int a = 10;
  cout << "(Truoc khi goi ham) a = " << a << endl;
  InGiaTri(a);
  cout << "(Sau khi goi ham) a = " << a << endl;
  return 0;
}
```

```
(Truoc khi goi ham) a = 10
(Ben trong ham) a = 11
(Sau khi goi ham) a = 11
```

3. Tham chiếu và tham trị

- Truyền tham trị

  Cú pháp:

  <kiểu kết quả> <tên hàm> (<kiểu dữ liệu> tên biến)

  Ví dụ: float TrungBinhCong(float x, float y);

  Truyền tham trị không làm thay đổi giá trị của các biến được truyền

- Truyền tham chiếu

  Cú pháp :

  <kiểu kết quả> <tên hàm> (<kiểu dữ liệu>& tên biến)

  Ví dụ: float TrungBinhCong(float& x, float& y);

  Khi truyền tham chiếu, những tác động lên biến tham chiếu bên trong hàm có ảnh hưởng đến biến được truyền

- Có thể khai báo cả tham trị và tham chiếu trong danh sách

  Ví dụ : float TrungBinhCong(float& x, float y);

TRUYỀN THAM TRỊ

- Truyền tham trị: là truyền giá trị của tham số
- Khi hàm được gọi, chương trình sẽ khởi tạo các ô nhớ tương ứng với danh sách tham số truyền vào; sao chép giá trị của tham số vào các ô nhớ mới này. Do đó, các thay đổi trong ô nhớ mới không ảnh hưởng đến các tham số truyền vào.
- Ví dụ 1: xét lời gọi hàm của các hàm tính tổng 2 số và đổi chỗ 2 số sau với giá trị truyền vào là a = 5, b = 8:

```
int tinhtong(int so1, int so2)
{
int tong;
tong = so1+so2;
return tong;
}
```

```
void doicho(int so1, int so2)
{
int temp = so1;
so1 = so2;
so2 = temp;
}
```

NHẬN XÉT

- Với lời gọi hàm ketqua = tinhtong(a, b); thì với a = 5, b = 8, kết quả sẽ trả về giá trị là 13
- Với a = 5, b = 8 và lời gọi hàm doicho(a, b); cùng với lệnh in ra màn hình a, b sau khi đổi chỗ thì kết quả trước và sau khi thực hiện không thay đổi
- Nguyên nhân: do khi truyền tham số, trình biên dịch sao chép a, b sang các tham số so1, so2 và thực hiện đổi chỗ trên 2 ô nhớ này, như vậy giá trị của a, b là không đổi (hay mọi thay đổi trong hàm sẽ không ảnh hưởng đến chương trình).
- Khắc phục: ???

TRUYỀN THAM CHIẾU

- Truyền tham chiếu sẽ làm thay đổi giá trị của biến truyền vào. Vì vậy các thao tác làm thay đổi biến được truyền vào thì kết thúc hàm vẫn được giữ nguyên.
- Để sử dụng các truyền tham chiếu, trong khai báo và định nghĩa của hàm, các tham số có dấu "&" trước tên của tham số đầu vào
- Ví dụ: Hàm đổi chỗ khi truyền tham chiếu sẽ thực hiện việc đổi chỗ 2 số một cách chính xác

```
void doicho (int &so1, int &so2);
```

VÍ DỤ SO SÁNH

- Hàm đổi chỗ 2 số dùng cách truyền tham trị:

```
void Swap1(int so1, int so2)
{
int temp = so1;
so1 = so2;
so2 = temp;
}
```

- Hàm đổi chỗ 2 số dùng cách tham chiếu:

```
void Swap2(int& so1, int& so2)
{
int temp = so1;
so1 = so2;
so2 = temp;
}
```

VÍ DỤ SO SÁNH

- Khi a = 1521, b = 3000, kết quả thực hiện chương trình:

```
a = 1521, b = 3000
Truyen theo tham tri (Swap1): a = 1521, b = 3000
Truyen theo tham chieu (Swap2): a = 3000, b = 1521
```

- Giải thích:

- Trong truyền tham trị (gọi Swap1(a,b)), các tham số *so1* và *so2* là bản sao của *a và b* nên các thay đổi của *so1* và *so2* trong hàm Swap1() **không ảnh hưởng tới giá trị của *a,b***.

- Trong truyền tham chiếu (gọi Swap2(a,b), các tham số *so1* và *so2* chính là *a* và *b* nên các thay đổi của *so1* và *so2* trong hàm Swap1() **làm thay đổi giá trị của *a,b***.

CÁC CÁCH TRUYỀN THAM SỐ- CHÚ Ý

- Trong trường hợp cần thay đổi giá trị của biến, nên sử dụng tham chiếu.
- Trong trường hợp không cần thay đổi giá trị của biến, nên sử dụng tham trị.
- Ngoài 2 cách truyền tham số trên, người lập trình cũng có thể sử dụng con trỏ

4. Phạm vi của biến (scope)

- Biến toàn cục (global) được khai báo bên ngoài tất cả các hàm
- Biến cục bộ (local) được khai báo bên trong một hàm hoặc một khối lệnh (giữa hai dấu { })
- Sau khi được khai báo, biến toàn cục có thể được sử dụng ở bất cứ chỗ nào trong chương trình
- Phạm vi của biến cục bộ chỉ giới hạn trong khối lệnh nơi nó được khai báo

Phạm vi của biến

- Đa số các biến có phạm vi cục bộ
- Các khai báo hàm thường có phạm vi toàn cục
- Các hằng thường được khai báo ở phạm vi toàn cục

```
//tinh dien tich hinh tron
#include <iostream> using
namespace std;
const double pi = 3.14159;

double dientich(double r)
{
    double dt;
        dt = pi * r * r;
    return dt;
}
int main()
{
    double bankinh = 2.;

    cout << "Dien tich " << dientich(bankinh); return
    0;
}
```

Biến toàn cục

Biến cục bộ

5. Bài tập

Bài 1

Có lỗi ở chương trình dưới đây. Hãy tìm và sửa nó.

```
#include <iostream>
using namespace std;
void TichHaiSo(int x, int y)
{
	return x * y;
}
int main()
{
	cout << TichHaiSo(4, 5) << endl;
	return 0;
}
```

Bài tập

Bài 2

Có lỗi ở chương trình dưới đây. Hãy tìm và sửa nó.

```
#include <iostream>
using namespace std;
int TichHaiSo(int x, int y)
{
    int ketqua = x * y;
}
int main()
{
    cout << TichHaiSo(4) << endl;
    return 0;
}
```

Bài tập

Bài 3

Viết chương trình con tính tổng của ba số thực và nhân tổng đó với 5.

Bài 4

Viết chương trình con có kiểu trả về void để chuyển đơn vị từ g sang kg.

Bài 5

Viết chương trình con hoán đổi giá trị hai số thực.