

TIN HỌC ĐẠI CƯƠNG

BÀI 3: HÀM

Phạm Xuân Cường
Khoa Công nghệ thông tin
cuongpx@tlu.edu.vn

- Phân rã vấn đề
- Định nghĩa hàm
- Truyền tham số
- Phạm vi của biến
- Hàm chung

Phân rã vấn đề

- Phân rã một vấn đề lớn, phức tạp thành các vấn đề nhỏ hơn, dễ giải quyết hơn
- VD: Phân rã vấn đề tính giá trị biểu thức $P = x^2 + \sqrt{x} + 1$ thành các vấn đề nhỏ hơn
 - VD nhỏ 1: Nhập giá trị cho x
 - VD nhỏ 2: Tính giá trị của P
 - VD nhỏ 3: In giá trị của P ra màn hình

Phân rã vấn đề trong C++ dùng hàm

- Phân rã vấn đề:

VĐ lớn → Chương trình C++

VĐ nhỏ 1 → Hàm 1

VĐ nhỏ 1.1 → Hàm 1.1

VĐ nhỏ 1.2 → Hàm 1.2

VĐ nhỏ 2 → Hàm 2

- Các công việc dùng thường xuyên cũng thường được viết thành hàm (như các hàm toán học trong thư viện C++ chuẩn)

Trước đây ta đã biết cách gọi hàm trong thư viện C++ chuẩn

```
#include <iostream>
using namespace std;
int main()
{
    double x = 1.44;
    double y = sqrt(1.44); // gọi hàm sqrt
    cout << y; // se in ra 1.2
    return 0;
}
```

Bây giờ ta sẽ tự viết hàm của mình

```
#include <iostream>
using namespace std;
// Viet ham tinh tong cua hai so
double tinh_tong(double x,double y)
{
    double z = x + y;
    return z;
}

int main()
{
    cout << tinh_tong(1.2, 1.3); // goi ham vua viet
    return 0;
}
```

Định nghĩa hàm

Định nghĩa hàm

```
<kiểu> <tên hàm> (danh sách tham số)
```

```
{
```

```
    các câu lệnh
```

```
}
```

- Danh sách tham số gồm không, một hoặc nhiều tham số (hình thức)
 - Mỗi tham số có dạng: <kiểu> <tên tham số>
 - Các tham số cách nhau bởi dấu phẩy
- Hàm phải trả về một giá trị có kiểu phù hợp với kiểu đã khai báo thông qua câu lệnh sau:
`return <biểu thức>;`

Ví dụ định nghĩa hàm

```
double tinh_tong(double x, double y)
{
    double z = x + y;
    return z;
}
```

Trong ví dụ này:

- Kiểu trả về: `double`
- Tên hàm: `tinh_tong`
- Danh sách tham số hình thức gồm `x` và `y` (đều có kiểu `double`)
- Thân hàm (đặt giữa hai dấu ngoặc móc) gồm hai câu lệnh, trong đó có câu lệnh `return` để trả về giá trị cho hàm

- Cú pháp lời gọi hàm: <tên hàm> (danh sách tham số)
- Các tham số trong lời gọi hàm được gọi là **tham số thực sự** (để phân biệt với **tham số hình thức** trong định nghĩa hàm)
- Vị trí của lời gọi hàm:
 - Trong phép gán:
`double tong = tinh_tong(1.2, 1.3);`
 - Trong biểu thức:
`double x = tinh_tong(1.2, 1.3) + 2;`

Hàm không có giá trị trả về

- Viết theo cú pháp sau:

```
void <tên hàm> (danh sách tham số)
```

```
{
```

```
    các câu lệnh
```

```
}
```

- Ở đây, **void** là kiểu dữ liệu đặc biệt, chỉ ra rằng hàm không trả về giá trị → thân của hàm không có câu lệnh **return** <bt>;
- Hàm không có giá trị trả về còn được gọi là **thủ tục**

Ví dụ hàm không có giá trị trả về

```
#include <iostream>
using namespace std;
// Viet ham in loi chao ra man hinh
void in_loi_chao()
{
    cout << "Xin chao cac ban";
}
int main()
{
    in_loi_chao(); // Ham kieu void dung doc lap
                  // thanh mot cau lenh
    return 0;
}
```

Cấu trúc chương trình với hàm

```
...  
định nghĩa hàm 1  
định nghĩa hàm 2  
int main()  
{  
    ...  
    gọi hàm 1  
    ...  
    gọi hàm 2  
    ...  
}
```

Làm việc với hàm

Viết hai hàm, một tính tổng và một tính hiệu của hai số thực, sau đó gọi chúng trong hàm main

```
#include <iostream>
using namespace std;
double tinh_tong(double x, double y)
{
    double t = x + y;
    return t;
}
double tinh_hieu(double x, double y)
{
    double h = x - y;
    return h;
}
```

Làm việc với hàm (tiếp)

```
int main()
{
    double a, b;
    cout << "Nhap so thu nhat: "; cin >> a;
    cout << "Nhap so thu hai: "; cin >> b;
    // Chu y: Trong cac dinh nghia ham luc truoc,
    // x va y la cac tham so hinh thuc, con o day
    // a va b la cac tham so thuc su.
    double tong = tinh_tong(a, b);
    double hieu = tinh_hieu(a, b);
    cout << "Tong la " << tong << endl;
    cout << "Hieu la " << hieu << endl;
    return 0;
}
```


Truyền tham số

Truyền tham số cho hàm

- Tham số hình thức là tham số trong định nghĩa hàm
`double tinh_tong(double x, double y)`
`{ ... }`
- Tham số thực sự là tham số trong lời gọi hàm
`double tong = tinh_tong(a, b);`
- Truyền tham số là quá trình truyền một tham số thực sự vào một tham số hình thức trong lời gọi hàm
 - Ví dụ: truyền a vào x và truyền b vào y

Hai kiểu tham số hình thức

- Tham trị
 - Sao chép tham số thực sự sang tham số hình thức. Ví dụ: Gán a cho x và gán b cho y .
 - Khai báo như thông thường
- Tham chiếu
 - Tham số hình thức và tham số thực sự đồng nhất với nhau
 - Ví dụ: x và a là một, y và b là một \rightarrow nếu thay đổi x và y trong hàm thì a và b cũng thay đổi theo
 - Kiểu tham số này có cách khai báo riêng (sẽ xem sau)

Ví dụ về tham số kiểu tham trị

```
#include <iostream>
using namespace std;
// n là tham số hình thức kiểu tham trị
void thay_doi(int n)
{
    n += 2; // Tăng n lên 2 đơn vị nhưng tham số thực sự (k)
           // truyền vào n sẽ không bị ảnh hưởng.
}
int main()
{
    int k = 3; // k là tham số thực sự
    cout << "Trước khi gọi hàm: k = " << k << endl; // k = 3
    thay_doi(k); // giá trị của k sẽ được gán cho n
    cout << "Sau khi gọi hàm: k = " << k << endl; // k = 3
    return 0;
}
```

Tham số kiểu tham chiếu

- Cách khai báo:
<kiểu tham số> & <tên tham số>
(chú ý dấu & giữa kiểu và tên tham số)
- Ví dụ:

```
// n là tham số kiểu tham chiếu  
void thay_doi(int & n) {  
    n = n + 2;  
}
```

Ví dụ về tham số kiểu tham chiếu

```
#include <iostream>
using namespace std;
// n là tham số hình thức kiểu tham chiếu (chú ý dấu &)
void thay_doi(int & n)
{
    n += 2; // Tăng n lên 2 đơn vị, do đó tham số thực sự (k)
           // truyền vào n cũng sẽ tăng lên 2 đơn vị
}
int main()
{
    int k = 3; // k là tham số thực sự
    cout << "Trước khi gọi hàm: k = " << k << endl; // k = 3
    thay_doi(k); // k và n đồng nhất với nhau
    cout << "Sau khi gọi hàm: k = " << k << endl; // k = 5
    return 0;
}
```

- Phân rã vấn đề **tính giá trị biểu thức $P = x^2 + \sqrt{x} + 1$** thành các vấn đề nhỏ hơn
 - VD nhỏ 1: Nhập giá trị cho x
 - VD nhỏ 2: Tính giá trị của P
 - VD nhỏ 3: In giá trị của P ra màn hình
- Mỗi vấn đề nhỏ sẽ được viết thành một hàm, sau đó tất cả được gọi đến trong hàm **main**

Ví dụ (tiếp)

```
#include <iostream>
#include <cmath>
using namespace std;
// Hàm nhập giá trị cho x
void nhap(double & x) // tham số kiểu tham chiếu
{
    cout << "Nhập giá trị cho x: ";
    cin >> x;
}
// Hàm tính giá trị biểu thức P
double tinh(double x) // tham số kiểu tham trị
{
    double P = x*x + sqrt(x) + 1;
    return P;
}
```


Ví dụ (tiếp)

```
// Hàm in giá trị biểu thức P
void xuất(double P) // tham số kiểu tham trị
{
    cout << "Giá trị của biểu thức P là ";
    cout << P << endl;
}

int main()
{
    double x, P;
    nhập(x);
    P = tính(x);
    xuất(P);
    return 0;
}
```

Phạm vi của biên

- Biến chỉ tồn tại và dùng được trong phạm vi của nó
- Biến có phạm vi toàn cục (biến toàn cục):
 - Có thể dùng ở bất cứ đâu trong chương trình
 - Khai báo trước hàm main và không nằm trong bất kỳ hàm nào
- Biến có phạm vi cục bộ (biến cục bộ):
 - Chỉ dùng được trong phạm vi cục bộ của nó
 - Khai báo trong một hàm hoặc khối lệnh, gồm các câu lệnh đặt giữa hai dấu ngoặc móc { }

Ví dụ về phạm vi của biến

```
#include <iostream>
using namespace std;
int n = 200; // n là biến toàn cục, dùng được ở mọi
             // nơi trong chương trình.
void f()
{
    int k = 10; // k là biến cục bộ, chỉ dùng được
               // trong hàm f.
}
int main()
{
    n = n * 2; // OK vì n là biến toàn cục
    k = k / 2; // Lỗi vì k là biến cục bộ trong hàm f
    return 0;
}
```

- Trong cùng phạm vi (toàn cục, hàm, khối lệnh):
 - Các biến không được phép trùng tên
- Khác phạm vi:
 - Các biến được phép trùng tên
- Giả sử có hai biến trùng tên: biến toàn cục x và biến cục bộ x trong phạm vi S :
 - Ngoài phạm vi S : x có nghĩa là x toàn cục
 - Trong phạm vi S : x có nghĩa là x cục bộ

Ví dụ về phân giải tên biến

```
#include <iostream>
using namespace std;
int n = 200; // n toan cuc
void f()
{
    int n = 10; // n cuc bo (khac n toan cuc ben tren)
}
int main()
{
    f(); // n cuc bo duoc gan gia tri 10.
    cout << n; // Day la n toan cuc nen se in ra 200.
    return 0;
}
```

Hàm chung

Viết hàm chung cho nhiều kiểu dữ liệu

- Xét hàm đổi giá trị của hai biến số thực:

```
void doi_cho(double & x, double & y)
{
    double tg = x; // tg la bien trung gian.
    x = y; // Gan y cho x, gia tri cu cua x duoc giu trong tg.
    y = tg; // Lay gia tri cu cua x trong tg gan cho y.
}
```

- Tuy nhiên, hàm `doi_cho` không làm việc được với hai biến có kiểu dữ liệu khác `double`
→ *giải pháp là viết lại hàm này cho tất cả các kiểu?*

Định nghĩa hàm chung

- C++ cho phép viết một hàm nhưng thao tác được trên nhiều kiểu dữ liệu khác nhau → hàm chung
- Ví dụ: `template < class T> // T la kiểu chung`
`void doi_cho(T & x, T & y) // doi_cho la ham chung`
{
 T tg = x;
 x = y;
 y = tg;
}
- Khi gọi hàm `doi_cho`, kiểu chung `T` được xác định tự động thông qua kiểu của các tham số thực sự truyền vào `x` và `y`

Ví dụ về hàm chung

```
#include <iostream>
using namespace std;
template < class T>
void doi_cho(T & x, T & y)
{
    T tg = x;
    x = y;
    y = tg;
}
int main()
{
    // xem slide tiếp theo...
}
```

Ví dụ về hàm chung (tiếp)

```
int main()
{
    double x = 1.2, y = 3.4;
    int a = 100, b = 200;
    cout << "Truoc khi doi cho:" << endl;
    cout << " x = " << x << ", y = " << y << endl;
    cout << " a = " << a << ", b = " << b << endl;
    doi_cho(x, y);
    doi_cho(a, b);
    cout << "Sau khi doi cho:" << endl;
    cout << " x = " << x << ", y = " << y << endl;
    cout << " a = " << a << ", b = " << b << endl;
    return 0;
}
```

Questions?