MÔN: LẬP TRÌNH NÂNG CAO

GIẢNG VIÊN: NGUYỄN QUỲNH DIỆP
EMAIL: diepnq@tlu.edu.vn

CHƯƠNG 1. HÀM

Giảng viên: Nguyễn Quỳnh Diệp – Khoa CNTT – ĐH Thủy Lợi

Email: diepnq@tlu.edu.vn

NỘI DUNG

- Cấu trúc chung của hàm
- Cách sử dụng hàm
- Các hàm được định nghĩa trước
- Biến toàn cục, biến cục bộ
- Truyền tham số
- Nạp chồng hàm
- Hàm đệ qui

KHÁI NIỆM VÀ
CẤU TRÚC CHUNG CỦA HÀM

VÍ DỤ:

- Nhập 3 số x,y,z
- Tính và đưa ra màn hình kết quả của:

$$\mathrm{f(x)} = x^3 + 2x^2 + 3$$
$$\mathrm{f(y)} = y^3 + 2y^2 + 3$$
$$\mathrm{f(z)} = z^3 + 2z^2 + 3$$

CÁCH 1: Không dùng hàm

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x, y, z, fx, fy, fz;
    cout << "nhap x = "; cin >> x;
    fx = x*x*x + 2*x*x + 3;
    cout<< "f" << x << " = "<< fx << endl;
    cout << "nhap y = "; cin >> y;
    fy = y*y*y + 2*y*y + 3;
    cout<< "f" << y << " = "<< fy << endl;
    cout << "nhap z = "; cin >> z;
    fz = z*z*z + 2*z*z + 3;
    cout<< "f" << z<< " = "<< fz << endl;
    return 0;
}
```

Kết quả

```
nhap x = 1
f1 = 6
nhap y = 2
f2 = 19
nhap z = 3
f3 = 48
```

CÁCH 2: Dùng hàm

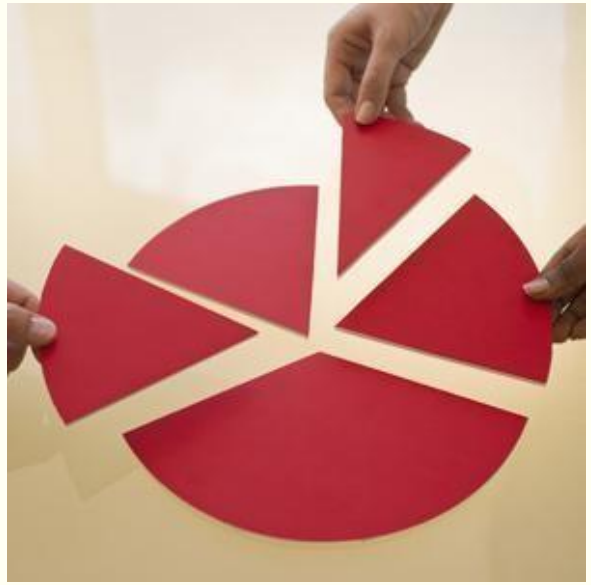
```
1  #include <iostream>
2  using namespace std;
3  int f(int x)
4  {  return x*x*x + 2*x*x +3;
5  }
6  int main()
7  { int x,y,z;
8    cout <<"Nhap x = ";
9    cin >>x;
10   cout << "f("<<x<<") = "<<f(x)<<endl;
11   cout <<"Nhap y = ";
12   cin >>y;
13   cout << "f("<<y<<") = "<<f(y)<<endl;
14   cout <<"Nhap z = ";
15   cin >>z;
16   cout << "f("<<z<<") = "<<f(z);
17 }
```

Kết quả

```
nhap x = 1
f1 = 6
nhap x = 2
f2 = 19
nhap x = 3
f3 = 48
```

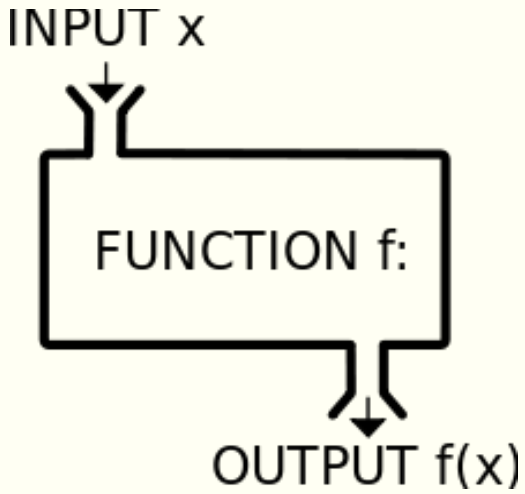
ĐẶT VẤN ĐỀ

- Khi bài toán có những công việc được lặp đi, lặp lại nhiều lần;
- Khi bài toán lớn, có những công việc độc lập nhau, có thể chia nhỏ -> chia thành các bài toán nhỏ hơn.
- Mỗi bài toán nhỏ được viết thành 1 hàm.



KHÁI NIỆM HÀM

- Hàm là một phần của chương trình, dùng để giải quyết một công việc được lặp đi, lặp lại nhiều lần trong chương trình; hoặc các công việc độc lập với nhau.
- Thông thường Hàm có đầu vào (input) và đầu ra (output). Khi sử dụng hàm là đưa những dữ liệu đầu vào và hàm sẽ thực hiện để cho kết quả tại đầu ra.



KHÁI NIỆM HÀM

- Hàm có thể có giá trị trả về hoặc không có giá trị trả về.
- Hàm bắt buộc phải có tên, quy tắc đặt tên giống với quy tắc đặt tên biến.
- Hàm có thể có 1 tham số, nhiều tham số hoặc không có tham số nào.
- Khối lệnh phía sau hàm chứa những dòng lệnh mà nó sẽ được thực hiện trong mỗi lần gọi hàm.

KHÁI NIỆM HÀM

- Mọi khái niệm như thủ tục, chương trình con trong các ngôn ngữ lập trình khác đều được hiểu là Hàm trong ngôn ngữ C++.

- Mọi chương trình C++ đều có 1 hàm chính là hàm main()

CÁC LOẠI HÀM

- *Hàm thư viện*: là những hàm đã được xây dựng sẵn. Muốn sử dụng các hàm thư viện phải khai báo thư viện chứa nó trong phần khai báo #include.
- *Hàm do người dùng định nghĩa*
    + Hàm trả về giá trị
    + Hàm không trả về giá trị

CẤU TRÚC CHUNG CỦA HÀM

Một hàm thường được tạo ra từ những yếu tố sau:

- Từ khóa void hoặc Kiểu trả về của hàm (data type of output).
- Tên hàm (function name).
- Danh sách tham số (function parameters).
- Khối lệnh (block of statements).

```
int f(int x)
{   int s;
    s = x*x*x + 2*x*x +3;
    return s;
}
```

MỘT SỐ HÀM ĐƯỢC ĐỊNH NGHĨA TRƯỚC

CÁC HÀM TOÁN HỌC

- **Hàm lượng giác:**

Hàm	Mô tả
cos (x)	Tính cosin của một góc x đo bằng radian
sin (x)	Tính sin của một góc x đo bằng radian
tan (x)	Tính tag của một góc x đo bằng radian
acos (x)	Tính acos trả ra giá trị radian
asin(x)	Tính asin trả ra giá trị radian
atan(x)	Tính actag trả ra giá trị radian

CÁC HÀM TOÁN HỌC

- **Hàm mũ và logarit:**

Hàm	Mô tả
exp(x)	Tính $\mathbf{e^x}$, x có kiểu **double**, **float**
log (x)	Tính logarit cơ số **e** của x, x có kiểu **double**, **float**
log10 (x)	Tính logarit cơ số **10** của x, x có kiểu **double**, **float**

CÁC HÀM TOÁN HỌC

- **Hàm lũy thừa, căn, lấy giá trị tuyệt đối:**

Hàm	Mô tả
pow(x, y)	Tính x^y , x,y có kiểu **double**, **float**
sqrt(x)	Tính căn bậc 2 của x, x có kiểu **double**, **float**
cbrt (x)	Tính căn bậc 3 của x, x có kiểu **double**, **float**
abs(x)	Lấy giá trị tuyệt đối,

CÁC HÀM TOÁN HỌC

- **Hàm làm tròn:**

Hàm	Mô tả
ceil(x)	Trả về số nguyên nhỏ nhất không nhỏ hơn x (làm tròn lên), x có kiểu **double**, **float**
floor (x)	Trả về số nguyên lớn nhất không lớn hơn x (làm tròn xuống), x có kiểu **double**, **float**

- **Ví dụ:**

```
ceil(2.2)          //  =  ?
ceil(3.7)          //  =  ?
ceil(-2.2)         //  =  ?
ceil(-3.7)         //  =  ?
```

CÁC HÀM TRÊN KIỂU KÍ TỰ

Hàm	Mô tả
tolower(ch)	Chuyển thành kí tự thường
toupper(ch)	Chuyển thành kí tự hoa
islower(ch)	Kiểm tra chữ thường
isupper(ch)	Kiểm tra chữ hoa
isdigit(ch)	Kiểm tra chữ số
isalpha(ch)	Kiểm tra xem kí tự có là chữ cái không
isspace(ch)	Kiểm tra kí tự dấu cách
iscntrl(ch)	Kiểm tra kí tự điều hiển

http://www.cplusplus.com/reference/cctype/

CÁC HÀM TRÊN KIỂU XÂU

Phương thức	Mô tả
.length() , .size()	Trả về độ dài của xâu
.clear()	Xóa nội dung của xâu
.append(str)	Thêm các kí tự/xâu str vào cuối xâu hiện tại
.insert(pos, str)	Chèn các kí tự/xâu con vào xâu tại vị trí bất kì
.replace(pos, len, str)	Thay thế xâu con trong xâu hiện tại bằng 1 xâu con mới
.substr(pos, len)	Trích xâu con từ xâu ban đầu
.compare(str)	So sánh xâu với xâu hiện tại
.find(str)	Tìm xâu con trong xâu hiện tại

http://www.cplusplus.com/reference/string/string/

CÁC HÀM TRÊN KIỂU XÂU

Hàm	Mô tả
. **insert(pos, str2)**	Chèn xâu str2 vào vị trí pos
. **insert(pos, n, c)**	Chèn kí tự c n lần vào vị trí pos
.erase (pos, len)	Xóa len kí tự bắt đầu từ vị trí pos
.clear()	Xóa xâu
.find(str)	Trả ra vị trí tìm thấy xâu str trong xâu chính
. find(str, pos)	Trả ra vị trí tìm thấy xâu str trong xâu chính, bắt đầu tìm từ vị trí pos

CẤU TRÚC CHƯƠNG TRÌNH KHI SỬ DỤNG HÀM

HÀM TRẢ VỀ GIÁ TRỊ

CÚ PHÁP ĐỊNH NGHĨA HÀM

- Hàm trả về giá trị
- Thường dùng để thực hiện các tính toán
- Trong thân hàm CÓ có lệnh **return giá trị**, để trả giá trị cho hàm

```
kieutrave  tenham (kieudulieu  thamso1, kieudulieu   thamso2,..)
{
        //các câu lệnh xử lý
        return   giatri;        //câu lệnh trả về giá trị
}
```

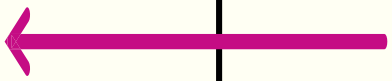
Tên hàm và số tham số phải trùng với nguyên mẫu hàm

Khi định nghĩa hàm không có dấu ;

CẤU TRÚC CHƯƠNG TRÌNH

```
//Phần khai báo thư viện
#include <iostream>
using namespace std;

//Phần định nghĩa hàm

//Hàm chính
int main () {

    //Lời gọi hàm

    return 0;
}
```

Viết định nghĩa hàm tại đây



Gọi hàm

CÚ PHÁP KHAI BÁO NGUYÊN MẪU HÀM

- **Chỉ là mô tả mẫu hàm**
  - Tên hàm
  - Các tham số
- **Cú pháp:**

kieutrave tenham (kieudulieu thamso1, kieudulieu thamso2,..);

kieutrave tenham (kieudulieu, kieudulieu ….);

- **kieutrave**: *kiểu dữ liệu trả về của hàm*
- **tenham**: *đặt theo quy tắc định danh*
- **kieudulieu**: *là kiểu dữ liệu của các tham số đầu vào*
- **thamso1, thamso2**: *tên các tham số đầu vào, sử dụng trong hàm, đặt theo quy tắc định danh*

CẤU TRÚC CHƯƠNG TRÌNH

```
//Phần khai báo thư viện
#include <iostream>
using namespace std;

//Phần khai báo nguyên mẫu hàm

//Hàm chính
int main () {

    //Lời gọi hàm

    return 0;
}

//Phần định nghĩa hàm
```

Viết **nguyên mẫu** hàm tại đây

Gọi hàm

Viết **định nghĩa** hàm tại đây

VÍ DỤ: 2 CÁCH VIẾT

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

double binhPhuong(double x)
{
    double s = x*x;
    return s;
}

int main () {
  double x;
  cout<<"Nhap gia tri x=";
  cin>>x;
  cout<<"x binh phuong ="<<binhPhuong(x);
  return 0;
}
```

```
#include <iostream>
using namespace std;
double binhPhuong(double x );

int main () {
  double x;
  cout<<"Nhap gia tri x=";
  cin>>x;
  cout<<"x binh phuong ="<<binhPhuong(x);
  return 0;
}

double binhPhuong(double x)
{
    double s = x*x;
    return s;
}
```

LỜI GỌI HÀM

- **Khi cần dùng hàm → gọi hàm**
- **Cú pháp:** **tenham (giatri1, giatri2…)**

- **Khi gọi hàm cần lưu ý:**
  - Gọi đúng **tên** hàm
  - Đảm bảo đủ **số lượng** tham số truyền vào
  - **Kiểu** của các tham số truyền vào phải tương ứng với kiểu đã khai báo

LỜI GỌI HÀM

- **Lời gọi hàm có thể xuất hiện ở các vị trí:**
  - Gọi trong lệnh gán giá trị cho biến
  - Gọi trong biểu thức toán học
  - Gọi trong câu lệnh **cout** ra màn hình

- **Ví dụ:**

```
double s = tong(a, 10, 2.5); //Goi ham tinh tong 3 so
double tb = tong(a, b, c)/3;
cout<<tong(a,b,c);
```

LỜI GỌI HÀM

- **Ví dụ:**

gọi hàm

```
#include <iostream>
using namespace std;
double binhPhuong(double);
int main () {
  double x, s;
  cout<<"Nhap gia tri x="; cin>>x;
  s= binhPhuong(x);
  cout<<"x*x ="<<s;
  return 0;
}

double binhPhuong(double x)
{
    double s = x*x;
    return s;
}
```

LỆNH return

- Trả về giá trị cho hàm
- Có tác dụng kết thúc hàm
- Có thể trả về giá trị của cả biểu thức
- Có thể xuất hiện lệnh return nhiều lần trong hàm

- **Ví dụ:**

```
double tuyetDoi(double u, double v)
{
      double s = u + v ;
      if(s>0)
            return s;
      else
            return -s;
}
```

TÌM LỖI SAI TRONG CHƯƠNG TRÌNH SAU?

```
1.  #include <iostream>
2.  using namespace std;
3.  double bieuThuc(double, double)
4.
5.  int main () {
6.    double x, y;
7.    cout<<"Nhap gia tri x, y:";
8.    cin>>x>>y;
9.    cout<<"Bieu thuc voi x= "x<<" va "<<20<<" la:" <<bieuThuc(x, 20)<<endl;
10.   cout<<"Bieu thuc voi x= "<<x<<" va "<<y<<" la:" <<bieuThuc(z, y)<<endl;
11.   cout<<"Bieu thuc voi x= "<<x<<" va "<<x<<" la:" <<bienThuc(x)<<endl;
12.   return 0;
13. }
14.
15. double biThuc(double x, int y)
16. {
17.     return x*x - 5*x + y;
18. }
```

TÌM LỖI SAI TRONG CHƯƠNG TRÌNH SAU?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main ()
4.  {
5.    int t; double vtoc;
6.    cout<<"Nhap gia tri van toc va thoi gian:";
7.    cin>>x>>y;
8.    cout<<"Quang duong= "<<quangduong(t, vtoc)<<endl;
9.    return 0;
10. }
11.
12. double quangduong(double v, int t);
13. {
14.     double s = v*t;
15.     return s;
16. }
```

BÀI TẬP

- **Bài 1:**

Cho 2 số r1=2 và r2=6. Viết chương trình tính diện tích 2 hình tròn bán kính r1 và r2 và hiển thị kết quả ra màn hình.

Trong chương trình có sử dụng hàm tính diện tích của hình tròn bán kính R.

- **Bài 2: Lập trình nhập vào tọa độ 3 điểm A, B, C. Tính các đoạn thẳng AB, AC, BC và đưa kết quả ra màn hình. Trong chương trình sử dụng hàm tính độ dài đoạn thẳng MN khi biết tọa độ 2 điểm M và N.**

HÀM KHÔNG TRẢ VỀ GIÁ TRỊ

CÚ PHÁP ĐỊNH NGHĨA HÀM

- Dùng từ khóa **void** để thay cho kiểu trả về của hàm
- Dùng để thực hiện 1 hành động
- Trong thân hàm KHÔNG có lệnh **return giá trị**
- Lời gọi tên hàm đứng độc lập, bản thân nó là 1 lệnh.
- Không được gọi trong câu lệnh **cout** ra màn hình, hay gán giá trị cho biến.

```
void tenham (kieudulieu thamso1, kieudulieu thamso2,..)
{
        //các câu lệnh xử lý
}
```

VÍ DỤ 1

```
#include <iostream>
using namespace std;
void trungBinh(double x, double y)
{
    double s = x + y ;
    cout<<“TB = “ << s/2;
}
int main () {
  double x, y;
  cout<<"Nhap gia tri x, y:";
  cin>>x>>y;
  trungBinh(x,y); //Loi goi ham trungBinh
  return 0;
}
```

VÍ DỤ 2

- Nhập 3 số x,y,z
- Tính và đưa ra màn hình kết quả của:

f(x) = x^3+2*x^2+3
f(y) = y^3+2*y^2+3
f(z) = z^3+2*z^2+3

VÍ DỤ 2

```cpp
#include <iostream>
using namespace std;
void f(int x)
{
    int fx;
    cout << "nhap x = "; cin >> x;
    fx = x*x*x + 2*x*x + 3;
    cout<< "f" << x << " = "<< fx << endl;
}
int main()
{
    int x;
    f(x);
    f(x);
    f(x);
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
void f(int x)
{  int s;
   s = x*x*x + 2*x*x +3;
   cout << "f(" << x << ") = " << s;
}
int main()
{ int x,y,z;
  cout <<"Nhap x = "; cin >>x;
  f(x);
  cout <<"\nNhap y = "; cin >>y;
  f(y);
  cout <<"\nNhap z = "; cin >>z;
  f(z);
  return 0;
}
```

CÚ PHÁP KHAI BÁO NGUYÊN MẪU HÀM

- **Chỉ là mô tả mẫu hàm**
  - Tên hàm
  - Các tham số
- **Cú pháp:**

```
void tenham (kieudulieu thamso1, kieudulieu thamso2,..);
```

```
void tenham (kieudulieu, kieudulieu ….);
```

- **kieutrave**: *kiểu dữ liệu trả về của hàm*
- **tenham**: *đặt theo quy tắc định danh*
- **kieudulieu**: *là kiểu dữ liệu của các tham số đầu vào*
- **thamso1, thamso2**: *tên các tham số đầu vào, sử dụng trong hàm, đặt theo quy tắc định danh*

TÌM LỖI SAI TRONG CHƯƠNG TRÌNH SAU?

```
1.  #include <iostream>
2.  using namespace std;
3.  void hienthi(double s)
4.  {
5.      cout<<"Gia tri vua nhap ="<<s<<endl;
6.  }
7.  int main ()
8.  {
9.    double x, y, s;
10.   cout<<"Nhap x= ";cin>>x;
11.   cout<<"Nhap y= ";cin>>y;
12.   hienthi(x,y);
13.   hienthi(y);
14.   hienthi(x+y);  return 0;
15. }
16.
```

TÌM LỖI SAI TRONG CHƯƠNG TRÌNH SAU?

```
1.  #include <iostream>
2.  using namespace std;
3.  double dientroTD(double, double );
4.  int main ()
5.  {
6.    double r1, r2;
7.    cout<<"Nhap r1= ";cin>>r1;
8.    cout<<"Nhap r2= ";cin>>r2;
9.    cout<<"Dien tro tuong duong=";
10.   dientroTD(r1,r2);
11.   return 0;
12. }
13. double dientroTD(double r1, double  r2)
14. {
15.     double s;
16.     s = 1/(1/r1+1/r2);
    }
```

PHẠM VI CỦA BIẾN

BIẾN CỤC BỘ VÀ BIẾN TOÀN CỤC

- **Biến cục bộ**:
  - Là những biến được khai báo và sử dụng bên trong một hàm/ khối lệnh
  - Chỉ có tác dụng trong hàm/khối lệnh đó và bị hủy khi hàm/khối lệnh kết thúc.

```
for (int i=1; i<=5; i++) s=s+i;
cout<<i;
```

- **Biến toàn cục**:
  - Là những biến được khai báo ở ngoài hàm/ khối lệnh
  - Có tác dụng trong toàn bộ chương trình.

PHẠM VI KHỐI

- Biến khai báo trong khối nào thì chỉ có tác dụng trong khối đó.
- Phạm vi khối còn gọi là **cục bộ**, và biến trong khối là **biến cục bộ**.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int i=20;
7      {
8      int i=10;
9      cout<<"Gia tri i ben trong khoi: "<<i<<endl;
10     }
11     cout<<"Gia tri i ben ngoai khoi: "<<i;
12 }
```

```
Gia tri i ben trong khoi: 10
Gia tri i ben ngoai khoi: 20
```

PHẠM VI HÀM

- Biến khai báo trong hàm nào thì hoạt động trong hàm đó.
- Phạm vi hàm còn gọi là **cục bộ**, và biến trong hàm là **biến cục bộ**.

```
2  int main()
3  {
4  int k;
5  float m;
6  double x;
7  //Các lệnh khác
8  //…
9  }
```

PHẠM VI CHƯƠNG TRÌNH

- Biến được khai báo bên ngoài các hàm gọi là biến toàn cục, có tác dụng cho toàn bộ chương trình.
- Phạm vi chương trình là phạm vi toàn cục.
- Sử dụng biến toàn cục?

Có thể sử dụng nhưng hiếm khi vì khó kiểm soát các hàm sẽ dùng nó như thế nào.

```
1  #include <iostream>
2  using namespace std;
3  int a, b;
4  void Nhap()
5  {
6  cout<<"Nhap a: ";cin>>a;
7  cout<<"Nhap b: ";cin>>b;
8  }
9  int main()
10 {
11 int c;
12 Nhap();
13 c=a+b;
14 cout<<"Tong = "<<c;
15 return 0;
16 }
```

THAM SỐ, ĐỐI SỐ VÀ CÁCH TRUYỀN THAM SỐ

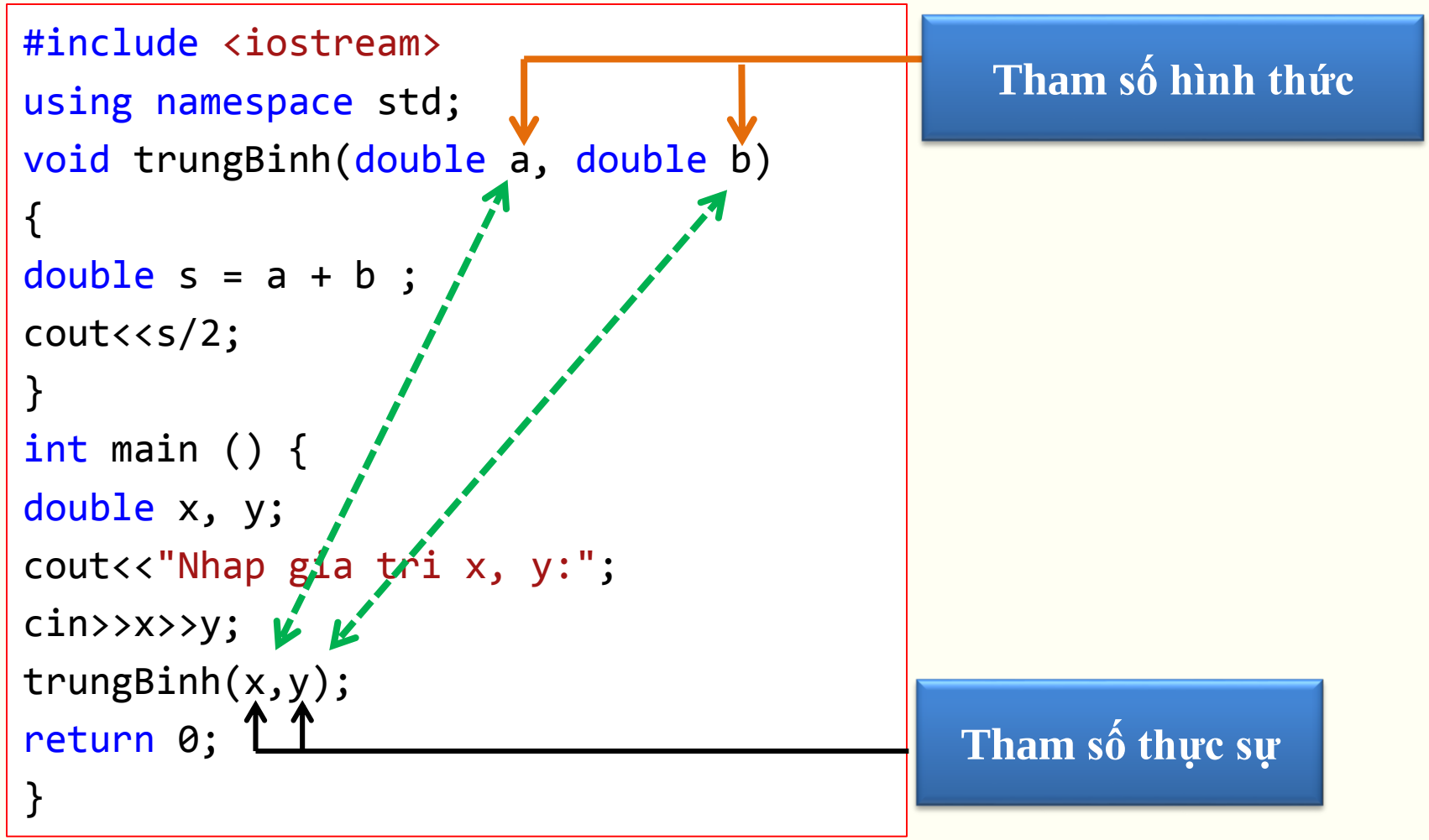
THAM SỐ

- Tham số hình thức (tham số)
- Tham số thực sự (đối số)

Tham số hình thức	Tham số thực sự
Là tham số được liệt kê trong phần định nghĩa và khai báo nguyên mẫu hàm	Là biến hoặc giá trị được viết trong lời gọi hàm
Là các tham số để nhận giá trị từ lời gọi hàm.	Là các giá trị thật sẽ được sao chép vào các tham số hình thức

VÍ DỤ

```
#include <iostream>
using namespace std;
void trungBinh(double a, double b)
{
double s = a + b ;
cout<<s/2;
}
int main () {
double x, y;
cout<<"Nhap gia tri x, y:";
cin>>x>>y;
trungBinh(x,y);
return 0;
}
```



HÀM KHÔNG CÓ THAM SỐ

- Không cần khai báo tham số trong định nghĩa hàm và khai báo nguyên mẫu hàm

- **Ví dụ:**

```
void hienThi()
{
      cout<<"Chao mung ban den voi khoa hoc!";
}
```

CÁCH TRUYỀN

Có 2 cách truyền tham số khi gọi hàm:

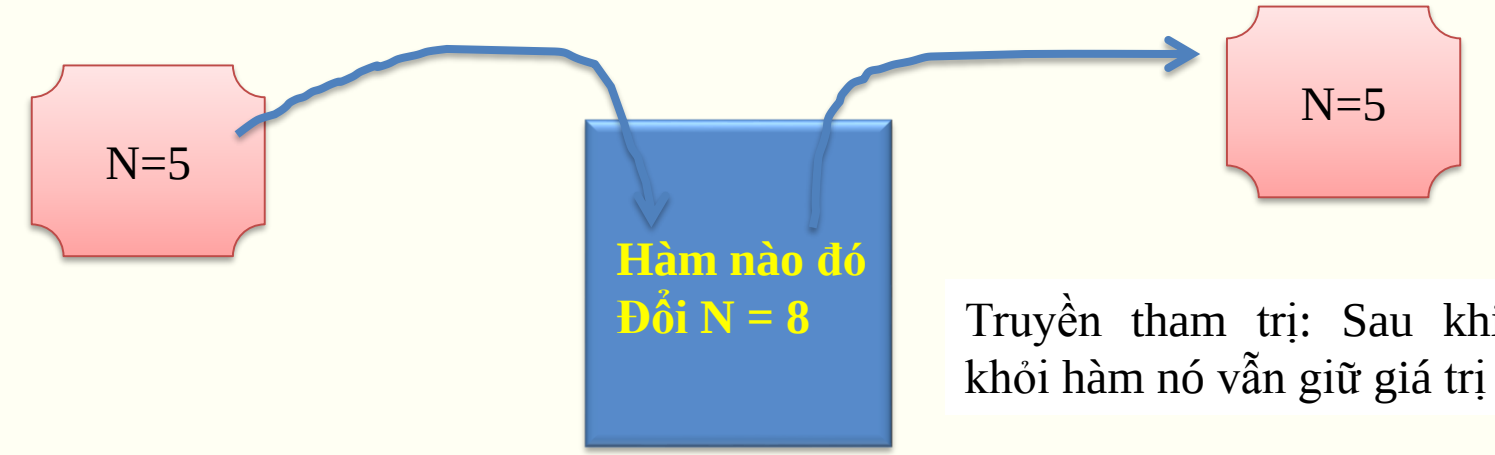
Truyền tham trị

- Truyền trực tiếp giá trị vào lời gọi hàm
- Truyền bản sao của biến vào lời gọi hàm, ra khỏi hàm biến KHÔNG thay đổi giá trị

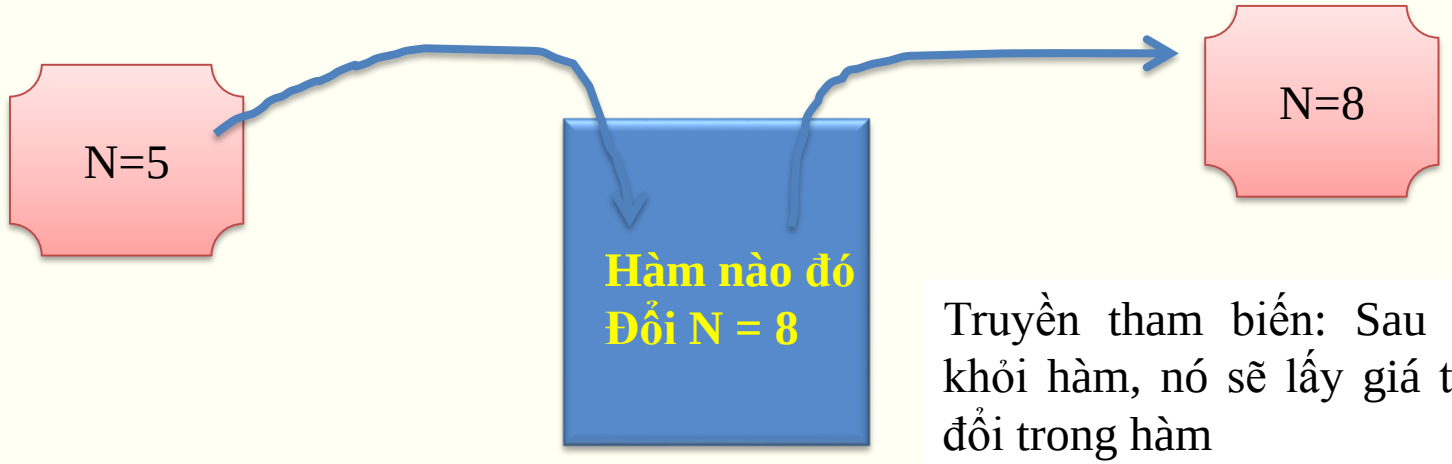
Truyền tham chiếu

- Truyền địa chỉ của biến vào lời gọi hàm
- CÓ làm thay đổi giá trị của biến truyền vào

TRUYỀN THAM SỐ



Truyền tham trị: Sau khi thoát khỏi hàm nó vẫn giữ giá trị gốc



Truyền tham biến: Sau khi thoát khỏi hàm, nó sẽ lấy giá trị bị thay đổi trong hàm

TRUYỀN THAM SỐ

Truyền tham trị (call by value)	Truyền tham chiếu *(call by reference)*
Sao chép giá trị của ***đối số*** vào ***tham số hình thức*** của hàm.	**Sao chép địa chỉ** của **đối số** vào ***tham số hình thức*** của hàm.
Những thay đổi của tham số không ảnh hưởng đến giá trị của đối số	Những thay đổi của tham số sẽ có tác dụng trên đối số

HÀM TRUYỀN THAM TRỊ

- Nguyên mẫu hàm truyền tham trị:

```
kieutrave  tenham (kieudulieu , kieudulieu,….);
```

- Định nghĩa hàm truyền tham trị:

```
 kieutrave  tenham (kieudulieu  thamso1, kieudulieu thamso2,..)
{
        //các câu lệnh xử lý
}
```

VÍ DỤ

```
#include <iostream>
using namespace std;
void hoanVi(int x, int y)
{
      int temp =x;
      x=y;  y=temp;
}
int main () {
  int x, y;
  cout<<"Nhap gia tri x, y:";
  cin>>x>>y;
  cout<<"Gia tri x, y truoc khi hoan doi "<<x<<"   "<<y<<endl;
  hoanVi(x,y); //Loi goi ham hoan vi
  cout<<"Gia tri x, y sau khi hoan doi "<<x<<"   "<<y<<endl;
  return 0;
}
```

```
Nhap gia tri x, y: 3 4
Gia tri x, y truoc khi hoan doi 3 4
Gia tri x, y sau khi hoan doi 3 4
```

HÀM TRUYỀN THAM CHIẾU

- Nguyên mẫu hàm truyền tham chiếu:

```
kieutrave  tenham (kieudulieu &, kieudulieu  &,….);
```

- Định nghĩa hàm truyền tham chiếu:

```
kieutrave  tenham (kieudulieu  &thamso1, kieudulieu  &thamso2,..)
{
        //các câu lệnh xử lý
}
```

VÍ DỤ

```
#include <iostream>
using namespace std;
void hoanVi(int &x, int &y)
{
      int temp =x;
      x=y;  y=temp;
}
int main () {
  int x, y;
  cout<<"Nhap gia tri x, y:";
  cin>>x>>y;
  cout<<"Gia tri x, y truoc khi hoan doi "<<x<<"   "<<y<<endl;
  hoanVi(x,y); //Loi goi ham hoan vi
  cout<<"Gia tri x, y sau khi hoan doi "<<x<<"   "<<y<<endl;
  return 0;
}
```

```
Nhap gia tri x, y: 3 4
Gia tri x, y truoc khi hoan doi 3 4
Gia tri x, y sau khi hoan doi 4 3
```

KHAI BÁO VỚI GIÁ TRỊ MẶC ĐỊNH

- Truyền giá trị mặc định khi định nghĩa hàm
- Giá trị mặc định được sử dụng khi không truyền tham số trong lời gọi hàm
- **Định nghĩa hàm:**

```
kieutrave  tenham (kieudulieu  thamso1 = giatri, kieudulieu  thamso2 = giatri,..)
{
        //các câu lệnh xử lý
}
```

KHAI BÁO VỚI GIÁ TRỊ MẶC ĐỊNH

- Ví dụ:

```
#include <iostream>
using namespace std;
int phepChia(int a, int b=2)
{
  int r;
  r=a/b;
  return (r);
}
int main ()
{
  cout << phepChia(12)<<endl; // goi ham voi gia tri mac dinh
  cout << phepChia(20,4);
  return 0;
}
```

TỔNG HỢP

- **Khi viết hàm, xác định bài toán:**

  - Hàm có trả về giá trị hay không?
  - Cần bao nhiêu tham số đầu vào?
  - Kiểu của các tham số là gì?
  - Truyền tham trị hay tham chiếu cho hàm?
  - Gọi hàm ở đâu?

BÀI TẬP

Bài 1: Viết chương trình nhập vào 1 số nguyên dương N. Tính và đưa ra màn hình N! Và N!!. Trong đó: N!=1.2.3...N; N!!=1.3.5....

Yêu cầu viết 2 hàm

Bài 2: Viết chương trình nhập vào số nguyên dương N. Viết hàm kiểm tra xem N có phải là số chính phương hay không?

(Số chính phương là bình phương của 1 số tự nhiên)

Bài 3: Viết chương trình nhập vào số nguyên dương N. Viết hàm kiểm tra xem N có phải là số nguyên tố hay không?

(Số nguyên tố là số >1 và chỉ có ước là 1 và chính nó)

BÀI TẬP

Bài 4: Viết hàm tính diện tích tam giác ABC khi biết độ dài 3 cạnh. Viết chương trình nhập độ dài 3 cạnh tam giác, gọi hàm tính diện tích trên và in kết quả S ra màn hình.

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ với } p = \frac{a+b+c}{2}$$

Bài 5: Viết chương trình con tính độ dài đoạn thẳng khi biết tọa độ 2 điểm sử dụng hàm sau:

void dodai(double xa,double ya, double xb, double yb, double &AB)

Viết chương trình nhập tọa độ 3 điểm A, B, C, sử dụng chương trình con trên để tính các đoạn thẳng AB, AC, BC.

NẠP CHỒNG HÀM

ĐẶT VẤN ĐỀ

Ví dụ 1:
Giả sử có chương trình như hình bên.
- Kết quả khi chạy CT là nho nhat = 4

```
1  #include <iostream>
2  using namespace std;
3  int nhonhat(int a, int b)
4   {
5       if (a<b) return a;
6       else return b;
7   }
8  int main()
9  {
10 cout<<"nho nhat = "<<nhonhat(4,5);
11 return 0;
12 }
```

- Nhưng nếu thay lời gọi hàm ở dòng 10 là nhonhat(4.3, 5.5) thì kết quả vẫn là nho nhat = 4
- Lý do là hàm nhonhat có 2 tham số là kiểu int nên chỉ nhận giá trị nguyên

ĐẶT VẤN ĐỀ

- Muốn có kết quả đúng ta phải viết 1 hàm khác có tham số là kiểu thực.

```
1  #include <iostream>
2  using namespace std;
3  double nhonhat(double a, double b)
4   {
5       if (a<b) return a;
6       else return b;
7   }
8  int main()
9  {
10 cout<<"nho nhat = "<<nhonhat(4.3,5.5);
11 return 0;
12 }
```

- Như vậy ta phải viết 2 hàm khác nhau chỉ nhằm cùng giải quyết 1 vấn đề là tìm giá trị nhỏ nhất của 2 số.

➔ Nạp chồng hàm ra đời

KHÁI NIỆM

- Nạp chồng hàm là kỹ thuật cho phép sử dụng cùng một tên gọi cho các hàm “giống nhau” (có cùng mục đích). Nhưng khác nhau về **kiểu dữ liệu** tham số hoặc **số lượng** tham số.
- Nạp chồng hàm cho phép ta khai báo và định nghĩa các hàm với **cùng một tên hàm.**

VÍ DỤ 2: Khác nhau về số lượng tham số

```
1  #include <iostream>
2  using namespace std;
3
4  double average(double n1, double n2)
5  { return ((n1 + n2) / 2.0);}
6
7  double average(double n1, double n2, double n3)
8  { return ((n1 + n2 + n3) / 3.0);}
9
10 int main() {
11 double x, y, z;
12 cout<<"Nhap gia tri x, y, z:";
13 cin>>x>>y>>z;
14 cout<<"Gia tri trung binh cua "<<x<<" va "<<y<<" la: " <<average(x, y)<<endl;
15 cout<<"Gia tri trung binh cua "<<x<<", "<<y<<" va "<<z<<" la: " <<average(x, y,z);
16 return 0;
17 }
```

```
Nhap gia tri x, y, z:3 4 5
Gia tri trung binh cua 3 va 4 la: 3.5
Gia tri trung binh cua 3, 4 va 5 la: 4
```

HÀM ĐỆ QUI

KHÁI NIỆM

- Một hàm được gọi là đệ quy nếu một lệnh trong thân hàm gọi đến chính hàm đó.
- Đệ quy phải xác định được điểm dừng. Nếu không xác định chính xác thì làm bài toán bị sai và có thể bị lặp vô hạn (Stack Overhead)

VÍ DỤ

Định nghĩa giai thừa của một số nguyên dương n như sau:

5!=5*4!

4!=4*3!

Tức là nếu biết được (n-1) giai thừa thì sẽ tính được n giai thừa, vì n!=n*(n-1)!

Qui ước nếu n=0 thì 0!= 1 $\rightarrow$ đây chính là điểm dừng

Vậy, có thể định nghĩa **n!**= $\begin{cases} \mathbf{1} & \boldsymbol{n}\text{ếu } \boldsymbol{n} = \mathbf{0} \\ (\boldsymbol{n}-\mathbf{1})! * \boldsymbol{n} & \boldsymbol{n}\text{ếu } \boldsymbol{n} > \mathbf{0} \end{cases}$

VÍ DỤ

```
1   #include <iostream>
2   using namespace std;
3
4   int giaiThua(int n)
5   {
6   if(n<=1)
7       return(1);
8   return n*giaiThua(n-1);
9   }
10
11  int main() {
12  int n;
13  cout<<"Nhap gia tri n:";
14  cin>>n;
15  cout<<n<<" != "<<giaiThua(n);
16  return 0;
17  }
```

HÀM ĐỆ QUI

Một hàm đệ quy căn bản gồm hai phần.

- **Phần cơ sở (phần neo)**: Điều kiện thoát khỏi đệ quy (điểm dừng đệ qui)
- **Phần đệ quy**: Thân hàm có chứa lời gọi đến chính nó nhưng với kích thước bé hơn.

```
{
   if(điều kiện dừng đệ quy) // phần cơ sở
        // Làm gì đó
   else
   {
    ....
        // Thực hiện gọi đệ quy
    ....
   }
}
```

VÍ DỤ

Hàm tìm Ước chung lớn nhất và Bội chung nhỏ nhất của 2 số

```
 * Tim uoc so chung lon nhat (USCLN)
 */
int USCLN(int a, int b) {
    if (b == 0) return a;
    return USCLN(b, a % b);
}

/**
 * Tim boi so chung nho nhat (BSCNN)
 */
int BSCNN(int a, int b) {
    return (a * b) / USCLN(a, b);
}
```

PHƯƠNG PHÁP THIẾT KẾ GIẢI THUẬT ĐỆ QUI

- Phân rã bài toán thành các bài toán nhỏ hơn.
- Trong đó có ít nhất một nhiệm vụ con là một trường hợp nhỏ hơn của nhiệm vụ cha (giống hệt bài toán ban đầu)
- Phân tích cho đến khi bài toán đủ nhỏ để cho ra kết quả đúng => đó là điểm dừng

ƯU NHƯỢC ĐIỂM CỦA ĐỆ QUI

- Hàm viết bằng đệ qui ngắn gọn, ít lệnh.
- Đệ qui không phải luôn luôn cần thiết
- Chạy chậm, sử dụng nhiều bộ nhớ.
- Bất kể công việc nào được giải bằng đệ qui đều có thể giải được theo cách không đệ qui, gọi là khử đệ qui
- Khử đệ qui bằng lặp: Dùng vòng lặp để thay thế cho đệ qui.

BÀI TẬP

Bài 1: Cho a và n là 2 số nguyên dương. Viết hàm tính a^n

Bài 2: Tính tổng của dãy số

$S(n) = 1 + 2 + 3 + \ldots\ldots + n$ *với n>=1*

Bài 3: Tính tổng của n số nguyên dương lẻ đầu tiên

$S(n)=1+3+5+\ldots+(2n-1)$ *với n>=1*

Bài 4: Tính số Fibonaci thứ n, biết rằng:

f(0) = 0

f(1) = 1

f(n) = f(n-1) + f(n+1)