Hàm

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC MỞ TPHCM

Mục tiêu

- Hiểu hàm là gì và tại sao nên phân chia chương trình thành các hàm.
- Biết cách sử dụng hàm có sẵn trong thư viện
- Hiểu cách truyền dữ liệu cho hàm và cách trả về giá trị do hàm tính toán.
- Biết cách cài đặt hàm trong C++ và sử dụng hàm đã cài đặt.
- Phân biệt cách truyền tham số bằng giá trị (pass-by-value) và truyền tham số bằng tham chiếu (pass-by-reference).

Hàm

- Khái niệm
- Định nghĩa hàm
- Truyền tham số cho hàm
- Phạm vi/ tầm vực

Khái niệm

- Khái niệm hàm
- Ý nghĩa của hàm
- Cách thức hoạt động của hàm

Khái niệm hàm

- *Hàm* (*function*) là chuỗi các câu lệnh được đặt tên.
- Mỗi hàm thực hiện **một công việc** nào đó, nhận vào dữ liệu, xử lý, và trả về kết quả.
- Hàm thường được xem là một *hộp đen* (*black box*).

Ý nghĩa hàm

- Phân chia chương trình thành nhiều phần để thuận tiện kiểm soát, logic hơn.
- Có thể sử dụng lại nhiều lần trong chương trình.
- Dễ kiểm thử, chỉnh sửa, cải tiến.

Hàm

- **Hàm có sẵn trong thư viện (*predefined function*)**
  Để sử dụng các hàm có sẵn trong thư viện, phải thêm header file vào đầu chương trình:
  - **`#include <iostream>`** (các hàm về nhập/xuất)
  - **`#include <iomanip>`** (các hàm về định dạng nhập/ xuất)
  - **`#include <cmath>`** (các hàm về toán)
  - **`#include <cctype>`** (các hàm về ký tự)
  - **`#include <ctime>`** (các hàm về ngày, giờ)
  - **`#include <string>`** (các hàm về chuỗi)
  - ….
- **Hàm do người dùng định nghĩa (*user-defined function*)**

Cách thức hoạt động của hàm

- Một chương trình C++ gồm có một hàm main và các hàm khác.
- Chương trình bắt đầu thực hiện từ hàm main, hàm main gọi các hàm khác, các hàm khi được gọi, có thể gọi các hàm khác nữa...
- **Khi sử dụng 1 hàm đã được xây dựng (gọi hàm) ta cần biết**:
  - Tên hàm
  - Chức năng của hàm
  - Danh sách tham số truyền vào hàm
  - Kết quả hàm trả về có kiểu dữ liệu gì?

Cách thức hoạt động của hàm

- **Khi một hàm được gọi (call function):**
  - chương trình chuyển điều khiển đến hàm (bắt đầu từ dòng lệnh đầu tiên);
  - thực hiện tất cả các lệnh bên trong hàm;
  - trả về dữ liệu đã tính toán cho nơi gọi;
  - chương trình tiếp tục thực hiện lệnh kế tiếp.

Cách thức hoạt động của hàm

- **Ví dụ**: Viết chương trình dùng hàm có sẵn trong thư viện <cmath> để tính x^y (x, y là 2 số nguyên dương).

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int x, y;
    double luyThua;
    cout << "Nhap co so: ";
    cin >> x;
    cout << "Nhap so mu: ";
    cin >> y;
    luyThua = pow(x*1.0, y);
    cout << "Ket qua = " << luyThua << endl;
    return 0;
}
```

Định nghĩa hàm

- Các bước viết một hàm
- Định nghĩa hàm (function definition)
- function prototype (nguyên mẫu hàm)
- Giá trị trả về (return value)
- Hàm void
- Gọi hàm thực thi
- Cấu trúc chương trình đề xuất khi có nhiều hàm
- Ví dụ

Các bước viết một hàm

- Bước 1: Xác định mục đích/mục tiêu của hàm.
- Bước 2: Định nghĩa dữ liệu hàm sẽ nhận từ nơi gọi (*paremeters*).
- Bước 3: Định nghĩa dữ liệu hàm sẽ tính toán và trả về (*return value*).
- Bước 4: Định nghĩa các bước được thực hiện để đạt được mục tiêu (*algorithm*)

Các bước viết một hàm

- **Ví dụ 1:** Xây dựng hàm tính diện tích hình chữ nhật.

1. Xác định mục đích/mục tiêu của hàm: Tính diện tích hình chữ nhật
2. Định nghĩa dữ liệu hàm sẽ nhận từ nơi gọi (*paremeters*).
   + Chiều dài hình chữ nhật (số thực)
   + Chiều rộng hình chữ nhật (số thực)
3. Định nghĩa dữ liệu hàm sẽ tính toán và trả về (*return value*): diện tích hình chữ nhật (số thực)
4. Định nghĩa các bước được thực hiện để đạt được mục tiêu (*algorithm*)
   + Tính chiều dài * chiều rộng và gán kết quả cho dienTich;
   + Trả về giá trị của dienTich

Định nghĩa hàm (function definition)

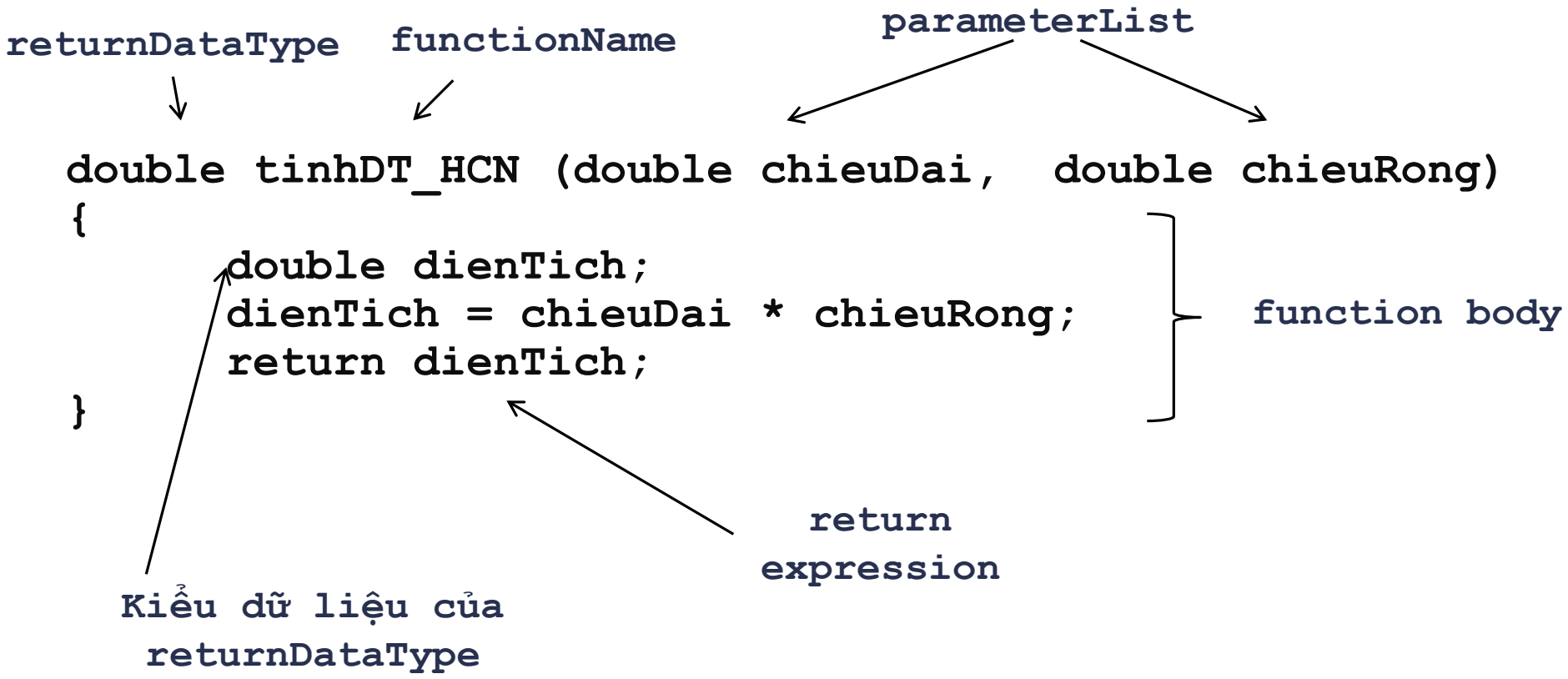
Cú pháp

```
returnDataType functionName ([parameterList])
{
    statements;
    [return expression;]
}
```

- **`returnDataType`**: kiểu dữ liệu của giá trị hàm trả về.
- **`functionName`**: tên hàm, theo luật đặt tên biến.
- **`parameterList`**: chứa *tham số hình thức* (formal parameters). Mỗi tham số hình thức gồm: kiểu dữ liệu và tên của tham số. Mỗi tham số cách nhau bằng dấu phẩy.
- **`statements`**: các câu lệnh được thực hiện.
- **`return expression;`** : trả về giá trị của *expression* do hàm tính, kiểu của expression giống với returnDataType.

Định nghĩa hàm (function definition)

- **Ví dụ 1:** Xây dựng hàm tính diện tích hình chữ nhật.



Định nghĩa hàm (function definition)

- **Ví dụ 2:** Xây dựng hàm tìm số lớn nhất của hai số nguyên.
  1. Mục đích: tìm số lớn nhất của 2 số nguyên
  2. Dữ liệu nhận vào: 2 số nguyên a và b
  3. Dữ liệu hàm trả về: số lớn nhất (số nguyên)
  4. Thuật giải:
     - Nếu a >= b thì max = a;
     - Ngược lại thì max = b;
     - Trả về kết quả max

Định nghĩa hàm (function definition)

- **Ví dụ 2:** Xây dựng hàm tìm số lớn nhất của hai số nguyên.

```
int timSoLonNhat (int a, int b)
{
    int max;
    if (a >= b)
        max = a;
    else
        max = b;
    return max;
}
```

Định nghĩa hàm (function definition)

- **Ví dụ 3:** Xây dựng hàm tính x^y (x là số thực, y là số nguyên)
  1. Mục đích: tính lũy thừa x^y
  2. Dữ liệu nhận vào: số thực x, số nguyên y
  3. Dữ liệu hàm trả về: kết quả lũy thừa (số thực)
  4. Thuật giải:

     Ngược lại thì nếu y >= 0 thì luyThua = x * x * … * x (y lần)

     Ngược lại (tức là y < 0) thì luyThua = x / x /… / x (-y lần)

     Trả về kết quả luyThua

Định nghĩa hàm (function definition)

- **Ví dụ 3:** Xây dựng hàm tính x^y (x là số thực, y là số nguyên)

```
double tinhLuyThua(double x, int y)
{
    double luyThua = 1.0;
    if (y >= 0)
        for (int i = 1; i <= y; i++)
            luyThua *= x;
    else
        for (int i = 1; i <= -y; i++)
            luyThua /= x;
    return luyThua;
}
```

function prototype (nguyên mẫu hàm)

- Hàm có thể được khai báo trước khi sử dụng hoặc định nghĩa bằng cách sử dụng nguyên mẫu hàm (*function prototype)*:

```
returnDataType  functionName  ([parameterList]);
```

Ví dụ:

```
double tinhLuyThua(double x, int y);
```

Hoặc

```
double tinhLuyThua(double, int);
```

- Khi *đặt function prototype của một hàm ở đầu* file chương trình nguồn (trước hàm main), trình biên dịch (compiler) có thể biên dịch chương trình, *bất kể* hàm đã được định nghĩa hay chưa.

Giá trị trả về (return value)

- Hàm có khai báo kiểu trả về (*return data type*) không phải là **`void`** thì phải trả về giá trị có cùng kiểu đã khai báo trong tiêu đề hàm (*function header*).
- Giá trị trả về được tính bên trong thân hàm (*function body*).
- Câu lệnh return kết thúc hàm và trả về giá trị do hàm tính.

```
return expression;
```

  - ✓expression: biến, giá trị hằng hoặc là biểu thức, phải có kiểu trùng với kiểu của hàm.
- Khi câu lệnh **`return`** được thực hiện, hàm kết thúc ngay lập tức và chuyển điều khiển về nơi gọi hàm.

Lưu ý về giá trị trả về

- Hàm được khai báo trả về giá trị thì phải trả về giá trị.
- Lỗi xảy ra nếu hàm đã thực thi xong nhưng không trả về giá trị.

```
double tinhDT_HCN (double chieuDai,  double chieuRong)
{
        if (chieuDai >= 0 && chieuRong >= 0)
             return chieuDai * chieuRong;
}     //loi
```

→ chương trình dịch cảnh báo (warning): not all control paths return a value.

Lưu ý về giá trị trả về

- Nếu hàm trả về giá trị trong câu lệnh rẽ nhánh thì phải đảm bảo mỗi nhánh đều có giá trị trả về.

```
int tinhTriTuyetDoi(int x)
{
    if (x < 0)
        return -x;
    else
        if (x > 0)
            return x;
}  //khong tra ve gia tri neu x bang 0
```

Hàm không trả về giá trị (void)

- Khi hàm thực hiện một chuỗi câu lệnh và không tạo ra giá trị thì khai báo kiểu trả về của hàm là **`void`**.
- Hàm **`void`** không trả về giá trị, vì vậy không được sử dụng trong biểu thức.
- Khi cần kết thúc hàm void, sử dụng câu lệnh:

```
return;
```

Hàm không trả về giá trị (void)

- **Ví dụ 4**: Viết hàm nhận vào số đo cạnh của hình vuông. Xuất hình vuông dưới dạng các dấu * ứng với số đo cạnh.
  1. Mục đích: xuất hình vuông
  2. Dữ liệu nhận vào: số đo cạnh hình vuông **`canh`** (số nguyên)
  3. Dữ liệu hàm trả về: không có
  4. Thuật giải:
     - Duyệt **`canh`** lần, mỗi lần duyệt:
       - Duyệt **`canh`** lần, mỗi lần xuất "*"
       - Xuống dòng

Hàm không trả về giá trị (void)

- **Ví dụ 4**: Viết hàm nhận vào số đo cạnh của hình vuông. Xuất hình vuông dưới dạng các dấu * ứng với số đo cạnh.

```
void xuatHinhVuong(int canh)
{
    for (int i = 1; i <= canh; i++)
    {
        for (int j = 1; j <= canh; j++)
            cout << "*";
        cout << endl;
    }
}
```

Gọi hàm thực thi

- Gọi hàm cần phải:
  - Gọi đúng tên hàm.
  - Truyền đủ tham số.
  - Nếu hàm có kiểu trả về thì:
    - Lời gọi hàm phải được gán kết quả cho một biến khác (biến này phải cùng kiểu với kiểu trả về của hàm); hoặc
    - Kết quả của hàm khi gọi sẽ tiếp tục được đem đi tính toán trong biểu thức tính toán; hoặc
    - Kết quả của hàm khi gọi sẽ được trả về trong câu lệnh xuất (cout) để xuất kết quả ra màn hình.
  - Nếu hàm không có kiểu trả về thì: chỉ gọi đúng tên và truyền đủ tham số kèm dấu ; (*như một câu lệnh*).

Gọi hàm thực thi

- Ví dụ về gọi hàm thực thi:

<table>
<tr><th>Nguyên mẫu hàm</th><th>Lời gọi hàm thực thi</th></tr>
<tr><td rowspan="2">

```
double tinhLuyThua(double
x, int y);
```

</td><td>

```
double m;
int n;
cout << "Nhap co so: ";
cin >> m;
cout << "Nhap so mu: ";
cin >> n;
cout << m << "^" << n << " = " <<
tinhLuyThua(m, n) << endl;
```

</td></tr>
<tr><td>

```
double m;
int n;
cout << "Nhap co so: ";
cin >> m;
cout << "Nhap so mu: ";
cin >> n;
double ketQua = tinhLuyThua(m, n);
cout << m << "^" << n << " = " <<
ketQua << endl;
```

</td></tr>
</table>

Gọi hàm thực thi

- Ví dụ về gọi hàm thực thi:

Nguyên mẫu hàm	Lời gọi hàm thực thi

`double tinhLuyThua(double x, int y);`	`//tinh am + bn`
	`double a, b;`
	`int m, n;`
	`cout << "Nhap co so: ";`
	`cin >> a;`
	`cout << "Nhap so mu: ";`
	`cin >> m;`
	`cout << "Nhap co so: ";`
	`cin >> b;`
	`cout << "Nhap so mu: ";`
	`cin >> n;`
	`double ketQua = tinhLuyThua(a, m)+ tinhLuyThua(b, n);`
	`cout << "Tong luy thua = " << ketQua << endl;`

`void xuatHinhVuong(int canh);`	`int n;`
	`cout << "Nhap so do canh: ";`
	`cin >> n;`
	`xuatHinhVuong(n);`

Cấu trúc chương trình đề xuất khi có nhiều hàm

```
//Tên chương trình
//Các chỉ thị tiền xử lý
//namespace

//Định nghĩa các hàm

//Hàm  main
```

```
//Tên chương trình
//Các chỉ thị tiền xử lý
//namespace

//Các function prototype
(nguyên mẫu hàm): có kết
thúc bằng dấu ;

//Hàm  main

//Định nghĩa các hàm
```

Cấu trúc chương trình đề xuất khi có nhiều hàm

- **Ví dụ 5:** Viết chương trình tính tổng các số từ 1 đến n (n là số nguyên dương)

```
#include <iostream>
using namespace std;
int tinhTong(int x)
{
   int tong = 0;
   for (int i = 1; i <= x; i++)
       tong += i;
   return tong;
} //het ham tinhTong
int main()
{
    int n;
    cout << "Nhap so: ";
    cin >> n;
    cout << "Tong tu 1 den " << n << " la " <<
    tinhTong(n) << endl;
    return 0;
}
```

Cấu trúc chương trình đề xuất khi có nhiều hàm

- **Ví dụ 5:** Viết chương trình tính tổng các số từ 1 đến n (n là số nguyên dương)

```
#include <iostream>
using namespace std;
int tinhTong(int x);
int main()
{
   int n;
   cout << "Nhap so: ";
   cin >> n;
   cout << "Tong tu 1 den " << n << " la " <<
   tinhTong(n) << endl;
   return 0;
}
int tinhTong(int x)
{
   int tong = 0;
   for (int i = 1; i <= x; i++)
      tong += i;
   return tong;
} //dinh nghia ham tinhTong
```

Truyền tham số cho hàm

- Khi gọi hàm phải truyền các đối số cho hàm.
- Các đối số được truyền cho hàm phải tương ứng với danh sách tham số trong định nghĩa hàm.
- Có 2 cơ chế truyền đối số cho hàm:
  - *truyền bằng giá trị* (*pass-by-value*)
  - *truyền bằng tham chiếu* (*pass-by-reference*)

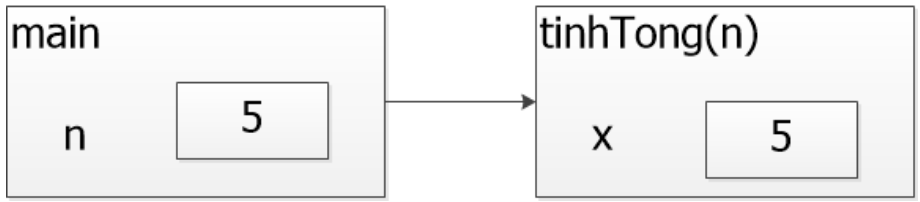
Truyền bằng giá trị (pass-by-value)

- Giá trị của đối số được sao chép và truyền cho hàm.
- Các thay đổi tham số bên trong hàm không ảnh hưởng đến đối số ban đầu do nơi gọi truyền đến.
- Với hàm tính tổng từ 1 đến n (n là số nguyên dương): n được truyền bằng giá trị.

```
int tinhTong(int n)
{
   int tong = 0;
   for (int i = 1;i <= n;i++)
      tong += i;
   return tong;
}
```

Truyền bằng giá trị (pass-by-value)

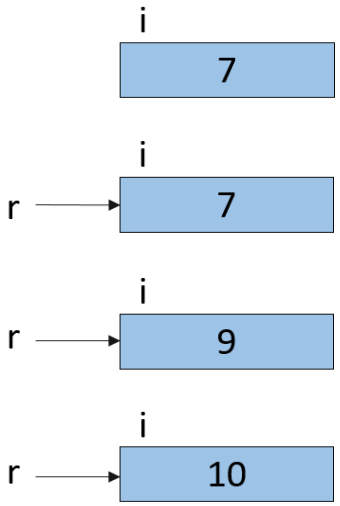
```
int main()
{
   int n;
   cout << "Nhap so: ";
   cin >> n;
   cout << "Tong tu 1 den " << n << " la "
   << tinhTong(n) << endl;
   return 0;
}
```



Truyền bằng tham chiếu (pass-by-reference)

- *Tham chiếu* (*reference*): tên khác (alias) của biến.

```
int i = 7;
int& r = i; //r tham chieu den i
r = 9;      //i = 9
i = 10;     //r = 10
cout << r << ' ' << i << '\n';
//in ra 10 10
```



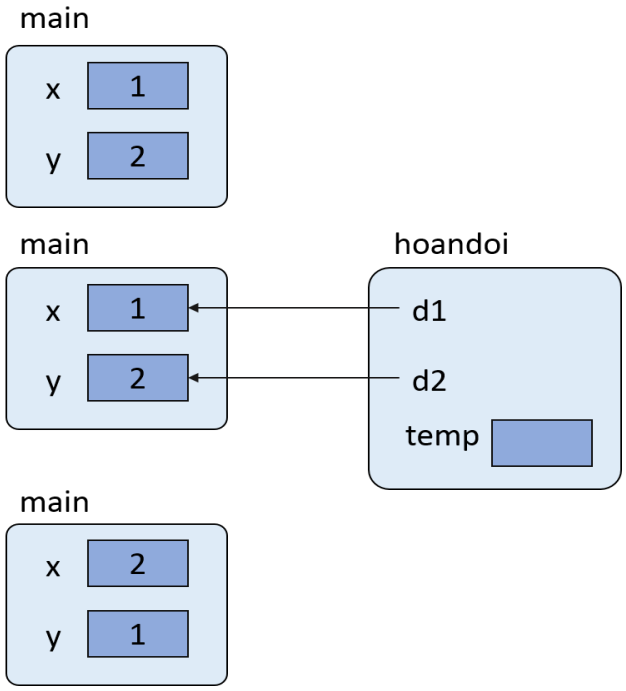
Truyền bằng tham chiếu (pass-by-reference)

- Không sao chép giá trị của đối số mà tham chiếu đến đối số.
- Tham số của hàm là tên khác (alias) của đối số, các thay đổi với tham số bên trong hàm sẽ ảnh hưởng trực tiếp đến đối số nơi gọi.

Truyền bằng tham chiếu (pass-by-reference)

- **Ví dụ 6**: Hàm hoán đổi 2 giá trị số nguyên cho nhau.

```
void hoanDoi(int  &d1, int  &d2)
{
    int temp;
    temp = d1;
    d1 = d2;
    d2 = temp;
}
int main()
{
   int x, y;
   cout << "Nhap so nguyen: ";
   cin >> x;
   cout << "Nhap so nguyen: ";
   cin >> y;
   cout << "Truoc khi goi ham x = " << x << " va y = "
   << y << endl;
   hoanDoi(x, y);
   cout << "Sau khi goi ham x = " << x << " va y = " <<
   y << endl;
}
```



Phạm vi/ tầm vực

- *Phạm vi/Tầm vực* (*scope*) là một vùng của chương trình.
- Tên được khai báo trong một phạm vi và có hiệu lực kể từ điểm khai báo cho đến hết phạm vi được khai báo.
- **Các loại phạm vi:**
  - phạm vi toàn cục (global scope)
  - phạm vi cục bộ (local scope): giữa **{** và **}**
  - phạm vi phát biểu (statement scope): bên trong câu lệnh for.

Phạm vi/ tầm vực

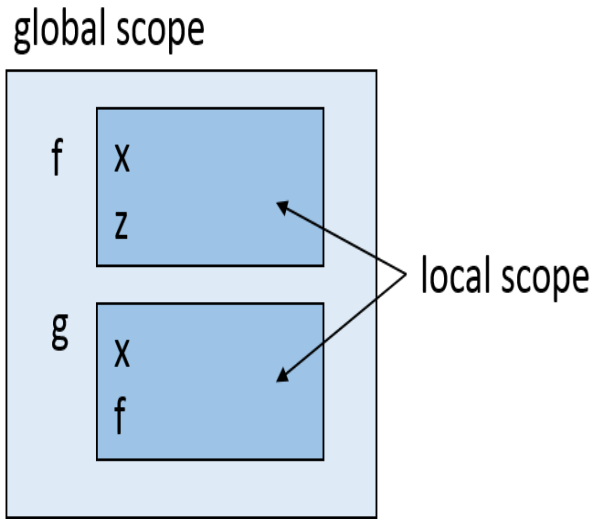
Ví dụ:

```
void f()
{
   g();  //loi: g() khong thuoc pham vi
}
void g()
{
   f();  //OK: f() trong pham vi
}
void h()
{
    int x = y;//loi: y khong thuoc pham vi
    int y = x;//OK
    g();      //OK
}
```

Phạm vi/ tầm vực

- Phạm vi cục bộ: chỉ tồn tại trong phạm vi nhất định

```
void f(int x)
{
    int z = x + 7;
}
int g(int x)
{
    int f = x + 2;
    return 2 * f;
}
```



Phạm vi/ tầm vực

- Các tham biến của hàm có phạm vi cục bộ
- Nên tránh việc khai báo trùng tên giữa biến toàn cục và cục bộ.

```
int lonNhat(int a, int b) //a, b: cuc bo
{
    return (a >= b) ? a : b;
}
int tinhTriTuyetDoi(int a)
{
    return (a < 0) ? -a : a;
    //khong phai a cua main, a la cuc bo cua
    ham
}
```

Phạm vi/ tầm vực

- Phạm vi cục bộ câu lệnh:

```
for (int i = 1; i <= 10; i++)
    cout << i << endl;
cout << "Ket thuc vong lap i = " << i <<
endl;

//lỗi vì i chỉ là biến cục bộ câu lệnh for
```

Q & A