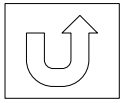
Chương 01.9: Hàm trong C++

I. Khai báo hàm

1. Giới thiệu về hàm
2. Cú pháp khai báo hàm
3. Các tham số trong khai báo hàm

1. Giới thiệu về hàm

- Trong C++ tất cả các chương trình con đều gọi là hàm.
- Ngoài các hàm thư viện có sẵn, người lập trình có thể tự tạo ra các hàm. Để tạo ra một hàm người lập trình phải khai báo và định nghĩa nó.
- Khai báo hàm (function declaration or prototype) là xác định tên của hàm, kiểu dữ liệu trả về, số lượng tham số và kiểu của từng tham số.
- Định nghĩa hàm (function definition) là xác định công việc mà hàm sẽ thực hiện thông qua các lệnh của hàm.
- Các hàm trong C++ không lồng nhau, tức là trong một hàm ta không thể định nghĩa một hàm khác.



2. Cú pháp khai báo hàm

- Cú pháp khai báo hàm nằm trên một dòng, kết thúc bằng dấu chấm phẩy.

  Kiểu_trả_về Tên_hàm(Kiểu_1 Tên_tham_số_1, Kiểu_2 Tên_tham_số_2,…);

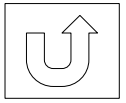
  Ví dụ: float inchtomet(float x);

  float cong(float a, float b);

- Một khai báo hàm không cho biết những gì có trong thân hàm. Nó chỉ báo cho trình biên dịch biết về tên hàm, kiểu của hàm, số lượng các tham số và kiểu của các tham số.

2. Cú pháp khai báo hàm (tiếp)

- Khai báo hàm có thể đặt ở bất kỳ đâu trước khi gọi hàm. Tốt nhất là để ở đầu tệp chứa chương trình chính (chứa hàm main) hoặc để trước một hàm sẽ gọi nó. Trong các chương trình nhiều file thì các khai báo hàm thường để trong các file header có đuôi .h, còn các định nghĩa hàm để trong các file thư viện có đuôi obj hoặc lib.
- Nếu hàm được định nghĩa ở đâu đó trước khi gọi hàm thì có thể không cần khai báo hàm. Tuy nhiên vẫn nên có khai báo hàm nhất là trong các chương trình có nhiều hàm lớn hay các chương trình nằm trên nhiều file.



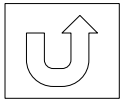
3. Các tham số trong khai báo hàm

- Nếu hàm không có tham số thì trong dấu ngoặc đơn của khai báo hàm để trống. Ví dụ:

  int xoa();

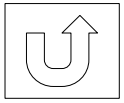
- Tên của các tham số trong khai báo hàm có thể không cần xác định. Ví dụ:

  float inchtomet(float, float);



II. Định nghĩa hàm

1. Cú pháp định nghĩa hàm
2. Lệnh return
3. Hàm không trả về giá trị



1. Cú pháp định nghĩa hàm

```
Kiểu_trả_về Tên_hàm(Kiểu_1 Tên_tham_số_1, Kiểu_2 Tên_tham_số_2,…)
{
        //Các lệnh của hàm để đây
}
```

Thân hàm

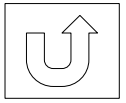
Không có dấu chấm phẩy

Ví dụ:

```
float cong(float a, float b)
{
    float z;
    z = a + b;
    return z;
}
```

1. Cú pháp định nghĩa hàm (tiếp)

- Dòng đầu tiên trong định nghĩa hàm giống trong khai báo hàm, chỉ khác là không có dấu chấm phẩy và các tham số bắt buộc phải có tên.
- Khi đã có khai báo hàm thì định nghĩa hàm thường để sau hàm main hoặc để trong một tệp obj (lib). Để quen dần với việc viết các chương trình lớn, khi thực hành chúng ta viết các khai báo hàm trong tệp .h, còn các định nghĩa hàm để trong tệp .obj (lib).

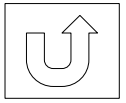


2. Lệnh return

- Lệnh return được sử dụng trong một hàm. Lệnh return thực hiện hai chức năng:
  - Làm cho một hàm trở về chương trình gọi nó.
  - Được dùng để trả về một giá trị.
- Cú pháp dùng lệnh return như sau:

  return Giá_trị_trả_về;

  hoặc return;
- Lệnh return có thể dùng ở bất kỳ vị trí nào trong hàm nhưng thường ở cuối hàm.
- Với các hàm có trả về giá trị thì lệnh return bắt buộc phải có.



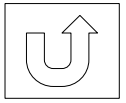
3. Hàm không trả về giá trị

- Với các hàm không trả về giá trị thì khi khai báo và định nghĩa hàm ta phải khai báo kiểu trả về là void. Ví dụ:

  void chao();

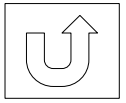
- Nếu sử dụng lệnh return trong hàm không trả về giá trị thì chỉ dùng được dạng:

  return;



III. Sử dụng hàm

1. Lời gọi hàm
2. Truyền đối số theo giá trị
3. Truyền đối số theo tham chiếu
4. Truyền con trỏ tới hàm
5. Truyền mảng tới hàm
6. Hàm có đối số mặc định



1. Lời gọi hàm

- Một hàm, sau khi được định nghĩa và khai báo, có thể được thực hiện bằng một lệnh gọi hàm (lời gọi hàm) ở đâu đó trong chương trình. Có thể gọi từ hàm main, có thể gọi từ một hàm khác hoặc có thể gọi từ một hàm thành viên của lớp.
- Cú pháp gọi hàm như sau:

  Tên_hàm(Danh sách các đối số, nếu có);
- Nếu hàm được khai báo và định nghĩa là có các tham số thì khi gọi hàm ta phải truyền giá trị cho hàm qua các tham số. Các giá trị truyền cho hàm gọi là các đối số. Các đối số có thể là hằng, biến, mảng, con trỏ,…

1. Lời gọi hàm (tiếp)

- Ví dụ: giả sử ta khai báo một hàm cộng hai giá trị float

  float cong(float a, float b);

  Ta gọi hàm này như sau:

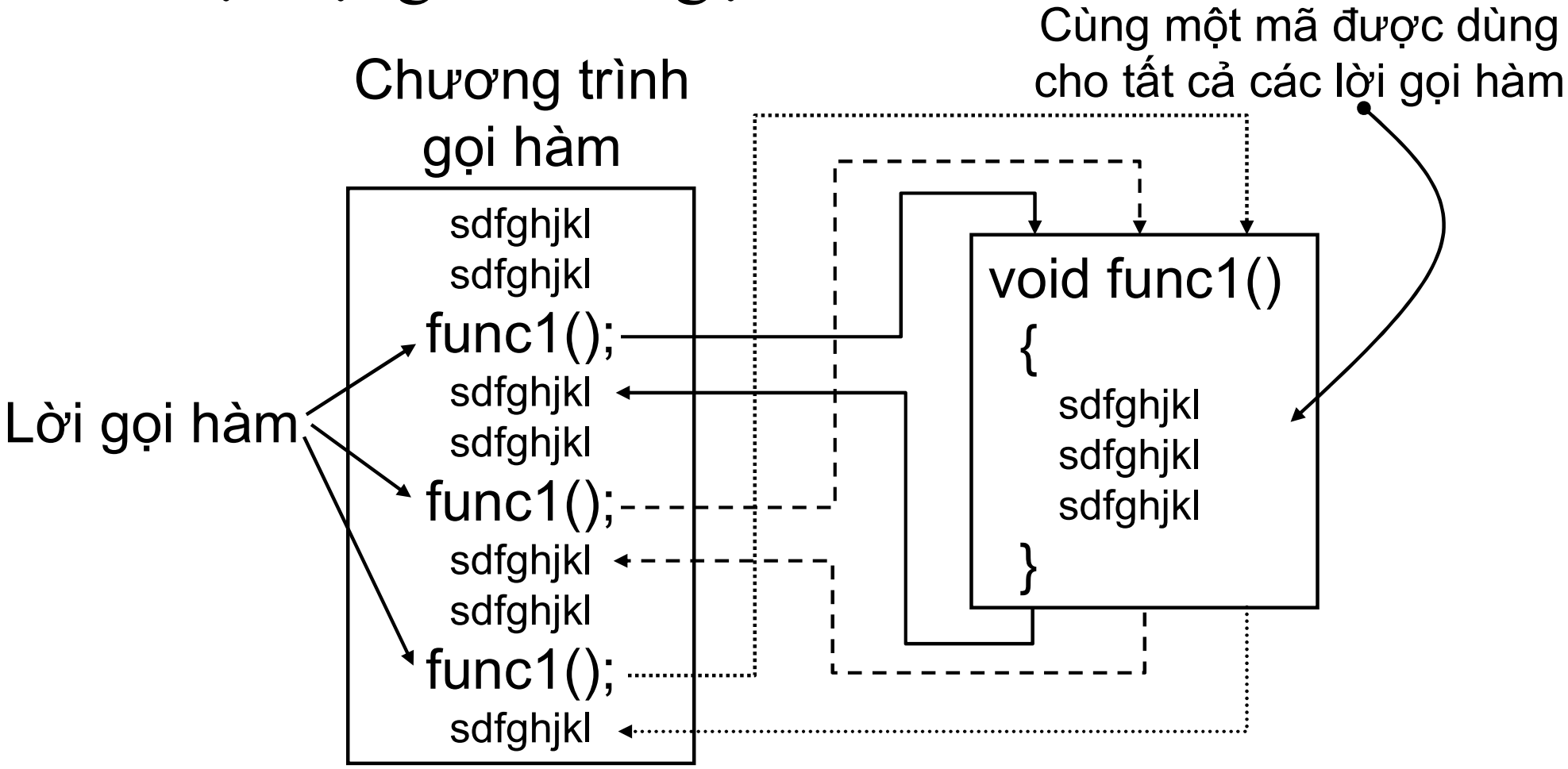
  cong(7,8);

- Lời gọi một hàm có trả về giá trị có thể sử dụng trong các biểu thức, còn lời gọi một hàm không trả về giá trị không dùng được trong biểu thức. Khi dùng trong biểu thức thì không có dấu chấm phẩy sau lời gọi hàm. Ví dụ:

  a = tong(7,8) +2; cout<<a;

1. Lời gọi hàm (tiếp)

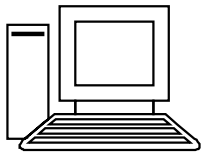
✧ Hoạt động của lời gọi hàm



Ví dụ về hàm

✧Viết chương trình tính số các chỉnh hợp chập k từ n phần tử. Chương trình phải sử dụng hàm để tính giai thừa và một hàm tính chỉnh hợp.

$$A_n^k = \frac{n!}{(n-k)!}$$



2. Truyền đối số theo giá trị

- Khi khai báo và định nghĩa hàm ta có hai cách khai báo các tham số của hàm:
  - Khai báo để khi gọi hàm truyền đối số cho hàm theo giá trị.
  - Khai báo để khi gọi hàm truyền đối số cho hàm theo tham chiếu.
- Khai báo để truyền đối số theo giá trị giống như khai báo biến thông thường:

  Kiểu Tên_tham_số

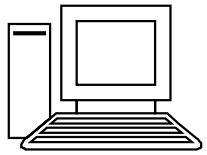
  Ví dụ: void DoiCho(int a, int b);

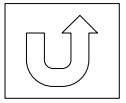
2. Truyền đối số theo giá trị (tiếp)

- Khi truyền đối số cho hàm theo giá trị thì hàm sẽ tạo ra các biến mới (tên các biến này là tên của các tham số), copy giá trị của các đối số vào các biến mới và thao tác trên các biến mới này. Bởi vậy sau khi gọi hàm các đối số không bị thay đổi giá trị mặc dù bên trong hàm giá trị của đối số bị thay đổi.
- Ví dụ: Để đổi chỗ giá trị trong hai biến ta viết hàm như sau: *(Xem trang sau)*

2. Truyền đối số theo giá trị (tiếp)

```
#include<iostream.h>
#include<conio.h>
void DoiCho(int,int);
void main()
  {
    int x=12,y=15;
    clrscr();
    cout<<"Truoc khi doi cho: x= "<<x<<", y= "<<y;
    DoiCho(x,y);
    cout<<"\nSau khi doi cho: x= "<<x<<", y= "<<y;
    getch();
  }
void DoiCho(int a,int b)  //Khai bao de truyen doi so theo gia tri
  {
    int tmp=a;
    a=b;
    b=tmp;
  }
```





3. Truyền đối số theo tham chiếu

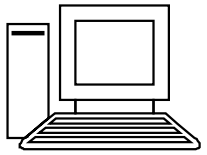
- Tham chiếu (reference) là một tên khác của cùng một biến.
- Khi truyền đối số theo tham chiếu hàm sẽ không tạo ra biến mới mà thao tác trực tiếp trên biến đối số. Kết quả là những tác động của hàm sẽ làm thay đổi giá trị của đối số.
- Để truyền đối số cho hàm theo tham chiếu thì khi khai báo hàm ta phải thêm dấu & vào bên phải tên kiểu của tham số.

  Ví dụ: void DoiCho(int &a, int &b);
- Các đối số truyền tới hàm theo tham chiếu chỉ có thể là biến không được là giá trị.

3. Truyền đối số theo tham chiếu (tiếp)

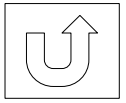
Ví dụ: Đổi chỗ giá trị của 2 biến

```
. . . . .
void DoiCho(int &a,int &b);
. . . . .
DoiCho(x,y);
.......
void DoiCho(int& a,int& b)
  {
    int tmp=a;
    a=b;
    b=tmp;
  }
```



3. Truyền đối số theo tham chiếu (tiếp)

- Khi đối số là đối tượng thì truyền theo tham chiếu là tốt nhất. Bởi vì truyền theo tham chiếu hàm sẽ không phải copy đối tượng mà thao tác trực tiếp trên đối tượng đối số. Với các đối tượng lớn thì đây là cách tiết kiệm bộ nhớ và thời gian thực hiện chương trình.



4. Truyền con trỏ tới hàm

- Để truyền con trỏ tới hàm ta phải thực hiện hai bước:
  - Khai báo các tham số (khi khai báo và định nghĩa) là con trỏ.
  - Khi gọi hàm thì đối số truyền cho hàm là địa chỉ.

Ví dụ:

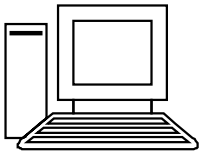
```
void DoiCho(int* a, int* b);
int x = 12,y = 15;
DoiCho(&x,&y);
```

4. Truyền con trỏ tới hàm (tiếp)

✧ Khi truyền con trỏ tới hàm thì biến do con trỏ trỏ tới có thể bị thay đổi bởi hàm.

Ví dụ: Đổi chỗ giá trị của hai biến

```
void DoiCho(int* a,int* b);
. . . .
DoiCho(&x,&y);
. . . .
void DoiCho(int* a,int* b)
  {
    int tmp = *a;
    *a = *b;
    *b = tmp;
  }
```



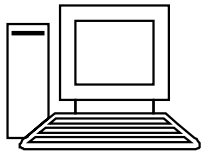
5. Truyền mảng tới hàm

- Khi tên mảng được sử dụng mà không có chỉ số kèm theo thì nó là địa chỉ bắt đầu của mảng. Do đó, nếu dùng mảng làm đối số truyền tới một hàm thì chỉ có địa chỉ của mảng được truyền tới hàm chứ không phải toàn bộ mảng. Điều này có nghĩa rằng khi khai báo tham số của hàm thì tham số phải có kiểu con trỏ.
- Bởi vì địa chỉ của mảng được truyền tới hàm nên mọi thay đổi của hàm lên mảng sẽ giữ nguyên khi hàm kết thúc.

5. Truyền mảng tới hàm (tiếp)

✧ *Ví dụ:* Viết một hàm đưa ra các phần tử của mảng

```
void print(int* m, int n);
. . . . .
int x[7]={2,5,8,1,6,7,10};
. . . . .
print(x,7);
. . . . .
void print(int* m, int n)
  {
    for(int i=0;i<n;i++) cout<<m[i]<<" ";
  }
```

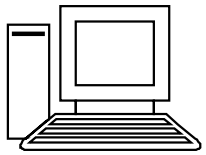


6. Hàm có đối số mặc định

- Một đặc điểm mới được đưa vào C++ liên quan tới hàm là đối số mặc định. Đó là khi khai báo và định nghĩa hàm ta có thể gán giá trị mặc định cho một tham số. Giá trị mặc định này sẽ được sử dụng khi không có đối số tương ứng với tham số đó trong lời gọi hàm.
- Như vậy, nếu sử dụng đối số mặc định thì khi gọi hàm có thể truyền đủ hoặc không đủ số lượng đối số.

6. Hàm có đối số mặc định (tiếp)

✧ *Ví dụ:* Khai báo và sử dụng hàm có đối số mặc định.



void f(int a=0, int b=5)

Với khai báo này ta có ba cách gọi hàm khác nhau:

- Cách 1: có thể gọi hàm với cả hai đối số: f(7,8);
- Cách 2: có thể gọi hàm với một đối số đầu tiên, khi đó đối số thứ hai sẽ có giá trị mặc định bằng 5: f(7);
- Cách 3: có thể gọi hàm mà không có đối số nào: f();

6. Hàm có đối số mặc định (tiếp)

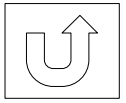
✧ Một số chú ý khi tạo hàm có đối số mặc định:

- Các giá trị mặc định chỉ được xác định duy nhất một lần ngay khi khai báo hàm (prototype), không xác định lại trong định nghĩa hàm. Ví dụ:

  ```
  void f(int a=0, int b=5);  //Khai báo hàm
  void f(int a=0, int b=5)   //Sai
    {   }
  ```

- Các tham số có giá trị mặc định phải nằm bên phải các tham số không có giá trị mặc định. Ví dụ:

  ```
  void f(int a, int b=1,int c);   //Sai
  ```



IV. Con trỏ trỏ tới hàm

- Một đặc điểm rất mạnh của C++ là con trỏ hàm. Mặc dù hàm không phải là biến nhưng nó vẫn có địa chỉ trong bộ nhớ. Địa chỉ này có thể chứa trong một con trỏ. Vì địa chỉ của hàm chứa trong con trỏ nên ta có thể sử dụng con trỏ thay cho tên hàm.
- Để có địa chỉ của hàm ta dùng tên hàm không có đối số (giống như tên biến mảng là địa chỉ của mảng).

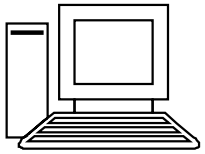
IV. Con trỏ trỏ tới hàm (tiếp)

- Để có con trỏ có thể chứa địa chỉ của hàm ta khai báo con trỏ trỏ tới kiểu giống kiểu trả về của hàm, theo sau là các tham số của hàm đặt trong ngoặc đơn. Ví dụ: giả sử hàm Tong có hai tham số kiểu int, kiểu trả về cũng là int. Khi đó ta khai báo con trỏ trỏ tới hàm này như sau:

```
int Tong(int a, int b); //Khai bao hàm
int (*p) (int a, int b);  //Khai báo con trỏ hàm
p = Tong;  //Gán địa chỉ của hàm Tong cho p
(*p)(10,15);  //Gọi hàm Tong qua con trỏ
```

IV. Con trỏ trỏ tới hàm (tiếp)

- Ví dụ: Viết chương trình tính tổng, hiệu, tích và thương của hai số nguyên nhập vào từ bàn phím. Chương trình yêu cầu người sử dụng lựa chọn một trong các cách tính.



Ví dụ

1. Cho dãy số nguyên có n phần tử. Sắp xếp dãy khóa tăng dần theo giải thuật sắp xếp chọn (Selection Sort). Dãy khóa là các số nguyên có n phần tử đọc vào mảng động từ tệp văn bản "daykhoa.txt". Yêu cầu viết 1 hàm đưa dãy khóa từ mảng ra màn hình; 1 hàm sắp xếp dãy khóa tăng dần; 1 hàm hoán đổi nội dung 2 ô nhớ.

Bài tập