

Chương 5

Hàm

*Khoa công nghệ thông tin
Trường Đại học mở tp hcm*

1

Mục tiêu

- Ý nghĩa của hàm trong lập trình.
- Cách sử dụng hàm có sẵn trong thư viện
- Cách truyền dữ liệu cho hàm và cách trả về kết quả do hàm tính toán.
- Cách cài đặt hàm trong C++ và sử dụng hàm đã cài đặt.
- Cách truyền tham số cho hàm bằng giá trị (pass-by-value) và truyền tham số bằng tham chiếu (pass-by-reference).

2

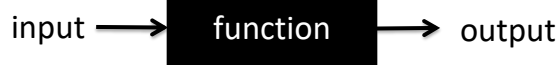
Nội dung

1. Giới thiệu
2. Định nghĩa hàm
3. Sự thực thi của hàm
4. Hàm trả trị và không trả trị
5. Truyền tham số cho hàm

3

5.1 Giới thiệu

- 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 của 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.

4

- Hàm thư viện:
 - Được định nghĩa 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)

- 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 (parameters).
- 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)

7

Các bước viết một hàm

- Ví dụ 1: Xây dựng hàm tính diện tích hình chữ nhật.
 - Xác định mục đích/mục tiêu của hàm: Tính diện tích hình chữ nhật
 - Định nghĩa dữ liệu hàm sẽ nhận từ nơi gọi (parameters).
 - Chiều dài hình chữ nhật (số thực)
 - Chiều rộng hình chữ nhật (số thực)
 - Định nghĩa dữ liệu hàm sẽ tính toán và trả về (return value): diện tích hình chữ nhật (số thực)
 - 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

8

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:
 - 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.

returnDataType functionName parameterList

```
double    tinhDT_HCN(double chieuDai, double chieuRong)
{
    double dienTich;
    dienTich = chieuDai * chieuRong;
    return dienTich;
}
```

↓ ↓ ↓ ↓

Kiểu dữ liệu của returnDataType return expression function body

Định nghĩa hàm (function definition)

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

- Mục đích: tìm số lớn nhất của 2 số nguyên
- Dữ liệu nhận vào: 2 số nguyên a và b
- Dữ liệu hàm trả về: số lớn nhất (số nguyên)
- Thuật giải:
 - Nếu $a \geq b$ thì $\text{max} = a$;
 - Ngược lại thì $\text{max} = b$;
 - Trả về kết quả max

```
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)

- Mục đích: tính x^y
- Dữ liệu nhận vào:
 - số thực x
 - số nguyên y
- Dữ liệu hàm trả về: số thực x^y
- Thuật giải:
 - Nếu $y \geq 0$ thì $\text{luyThua} = x * x * \dots * x$ (y lần)
 - Ngược lại (tức là $y < 0$) thì $\text{luyThua} = x / x / \dots / x$ (-y lần)
 - Trả về kết quả luyThua

```
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;
}
```

Định nghĩa hàm (function definition)

- 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

5.3 Sự thực thi 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 hàm (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?

Sự thực thi 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

15

Cách thức hoạt động của hàm

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

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int x, y;
    double luyThua;
    cout << "Nhập cơ số: ";
    cin >> x;
    cout << "Nhập số mũ: ";
    cin >> y;
    luyThua = pow(x * 1.0, y);
    cout << "Kết quả = " << luyThua << endl;
    return 0;
}
```

16

5.4.1 Hàm có trả về giá trị

- Trước khai báo tên hàm có chỉ rõ kiểu kiểu trả về (return data type)
- Giá trị trả về được tính bên trong thân hàm (function body) và phải có cùng kiểu với kiểu đã khai báo trước tên hàm.
- Phải có câu lệnh return trả về giá trị của hàm:
return expression;
 - expression: biến, biểu thức, phải có kiểu trùng với kiểu trả về của hàm.
- Khi câu lệnh return được thực hiện, hàm kết thúc ngay lập tức và chuyển điều khiển về nơi gọi hàm.

17

Hàm có trả về giá trị

- Lưu ý về giá trị trả về:
 - Thiếu lệnh return trong hàm có trả về giá trị.

```
double tinhDT_HCN(double chieuDai, double chieuRong)
{
    if (chieuDai >= 0 && chieuRong >= 0)
        return chieuDai * chieuRong;
} // (warning): not all control paths return a value
```

- 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 tri neu x bang 0
```

18

Hàm không trả về giá trị (void)

- Trước khai báo tên hàm có từ khóa void
- Hàm void không trả về giá trị, các xử lý tính toán thực hiện trực tiếp bên trong hàm.
- Khi cần kết thúc hàm void, sử dụng câu lệnh:
return;

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

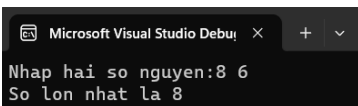
```
void xuatHinhVuong(int canh)
{
    for (int i = 1; i <= canh; i++)
    {
        for (int j = 1; j <= canh; j++)
            cout << "*";
        cout << endl;
    }
}
```

Gọi hàm

- Quy tắc: gọi đúng tên hàm, truyền đủ tham số.
 - Gọi hàm có trả trị: nhiều cách:
 - Lời gọi hàm phải được gán kết quả cho một biến khác (biến này phải cùng kiểu với kiểu trả về của hàm)
 - Sử dụng kết quả trả về của hàm để tính toán trong biểu thức khác
 - Lời gọi hàm có thể đặt trong câu lệnh xuất (cout) để xuất kết quả ra màn hình.
 - Gọi hàm không trả trị (void): chỉ gọi đúng tên và truyền đủ tham số kèm dấu ; (như một câu lệnh).

Gọi hàm

- Gọi hàm có trả trị:



```
Microsoft Visual Studio Debug
Nhap hai so nguyen:8 6
So lon nhat la 8
```

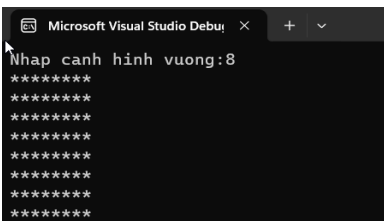
```
#include <iostream>
#include <cmath>
using namespace std;
//function prototype (nguyên mẫu hàm)
int timSoLonNhat(int a, int b);
int main()
{
    int a, b;
    cout << "Nhap hai so nguyen:";
    cin >> a >> b;
    int max = timSoLonNhat(a, b); //gọi hàm
    cout << "So lon nhat la " << max << endl;
    return 0;
}

int timSoLonNhat(int a, int b)
{
    return (a > b ? a : b);
}
```

21

Gọi hàm

- Gọi hàm không trả trị:



```
Microsoft Visual Studio Debug
Nhap canh hinh vuong:8
*****
*****
*****
*****
*****
*****
*****
*****
```

```
#include <iostream>
#include <cmath>
using namespace std;
//function prototype (nguyên mẫu hàm)
void xuatHinhVuong(int canh);
int main()
{
    int n;
    cout << "Nhap canh hinh vuong:";
    cin >> n;
    xuatHinhVuong(n);
    return 0;
}

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

22

Cấu trúc chương trình có nhiều hàm

- | | |
|---|--|
| <ul style="list-style-type: none"> • //Tên chương trình • //Các chỉ thị tiền xử lý • //khai báo namespace • //Các function prototype (nguyên mẫu hàm) kết thúc bằng dấu ; • //Hàm main • //Định nghĩa các hàm | <pre>//Ví dụ #include <iostream> #include <cmath> using namespace std; //function prototype (nguyên mẫu hàm) void func1(); int func2(int, int); double func3(int, double); int main() { } void func1() { } int func2(int a, int b) { } double func3(int n, double x) { }</pre> |
|---|--|

23

5.5 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 cách truyền đối số cho hàm:
 - Truyền bằng giá trị (pass-by-value)
 - Truyền bằng tham chiếu (pass-by-reference)

24

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.
- Ví dụ: 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;
}
```



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

Cơ sở lập trình - Chương 5: Hàm

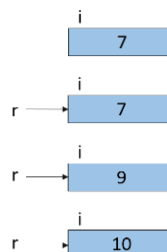
25

25

Truyền bằng tham chiếu (pass-by-reference)

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

```
int i = 7;
int& r = i; //r tham chiếu đến i
r = 9;      //i = 9
i = 10;     //r = 10
cout << r << " " << i << '\n';
//in ra 10 10
```



Cơ sở lập trình - Chương 5: Hàm

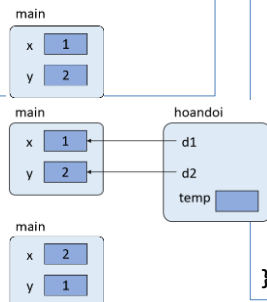
26

26

Truyền bằng tham chiếu (pass-by-reference)

- Ví dụ: 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 gọi hàm x = "
          << x << " và y = "
          << y << endl;
    hoanDoi(x, y);
    cout << "Sau khi gọi hàm x = "
          << x << " và y = "
          << y << endl;
}
```

27

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.

28

Phạm vi/ tầm vực

– Ví dụ:

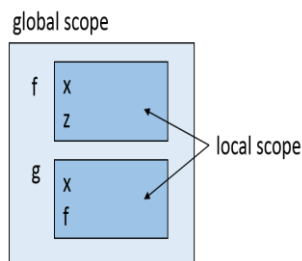
```
void f()
{
    g(); //lỗi: g() không thuộc phạm vi
}
void g()
{
    f(); //OK: f() trong phạm vi
}
void h()
{
    int x = y; //lỗi: y không thuộc phạm vi
    int y = x; //OK
    g(); //OK
}
```

29

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;
}
```



30

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: cục bộ
{
    return (a >= b) ? a : b;
}
int tinhTriTuyetDoi(int a)
{
    return (a < 0) ? -a : a;
    //biến a là biến cục bộ của hàm
}
```

- 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
```

31

Q & A

32