

Bài 3. Hàm

Giảng viên:

Hệ thống bài cũ

- Giới thiệu các cấu trúc điều khiển
- Giới thiệu chi tiết lệnh if và if-else
- Biểu thức điều kiện
- Giới thiệu chi tiết lệnh switch
- Giới thiệu chi tiết lệnh do...while
- Giới thiệu chi tiết lệnh while
- Giới thiệu chi tiết lệnh for

MỤC TIÊU

- Để giúp sinh viên có thể dễ quản lý chương trình lớn bằng mô đun hóa chúng thành các chương trình con.
- Giúp sinh viên hiểu các chương trình con này được gọi là hàm và có thể được biên dịch, kiểm tra độc lập và tái sử dụng trong các chương trình khác.

NỘI DUNG

- Định nghĩa hàm, gọi hàm.
- Nguyên mẫu hàm.
- Các bước thiết kế hàm.
- Phạm vi.
- Hàm không trả trị.
- Một số hàm toán học.
- Xét chi tiết cách truyền tham số.
- Hàm nội tuyến - hàm inline.
- Nạp chồng hàm.
- Hàm `main()`, hàm nhị phân.

Vấn đề

Ví dụ

Viết chương trình chuyển đổi nhiệt độ từ thang nhiệt Fahrenheit (F) thành Celsius (C). Biết công thức:

$$C = \frac{F - 32.0}{1.8}$$

Thực hiện



Giải Thuật

- Đưa thông báo nhắc người sử dụng nhập nhiệt độ F.
- Đọc giá trị thực nhiệt độ F và gán vào biến `temp_f`.
- Tính nhiệt độ C: `tep_c=(temp_f - 32.0)/1.8`
- Đưa giá trị của `tep_c` ra màn hình.

Viết Mã, Thực Thi và Kiểm Tra

ChuyenDoiFtoiC.cpp

(Global Scope)

main()

```
# include <iostream>
using namespace std;
//chuong trinh chuyen doi nhiet do F toi nhiet do C

int main()
{
    //khai bao bien nhiet do c
    double nhietdoF;
    cout << "Moi ban nhap vao nhiet do F: ";
    cin >> nhietdoF;
    cout << "Nhiet do F: "<<nhietdoF
         << " chuyen sang nhiet do C: "
         <<(nhietdoF - 32.0)/1.8 <<endl;
    system ("pause");
    return 0;
}
```

C:\Users\hieunh3\Documents\Visual Studio...

```
Moi ban nhap vao nhiet do F: 120
Nhiet do F: 120 chuyen sang nhiet do C: 48.8889
Press any key to continue . . . _
```

Sử dụng Hàm

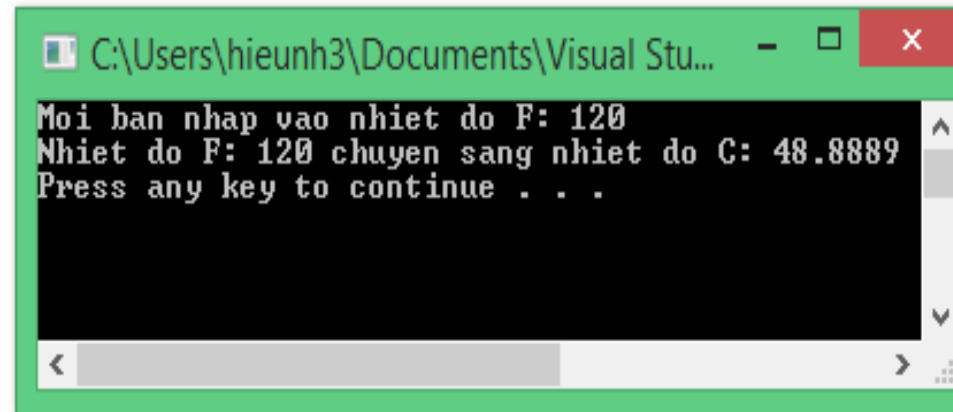
- Sự định nghĩa hàm (thay vì viết mã lệnh trong `main()`) để chuyển đổi nhiệt độ từ `Fahrenheit` thành `Celsius` làm cho các hàm hay các chương trình khác có thể **tái sử dụng** nó.
- Tránh lặp lại mã lệnh nếu có nhiều nơi trong chương trình sử dụng các mã lệnh này.
- Xem mã nguồn:

ChuyenDoiFtoiC.cpp

(Global Scope)

ChuyenNhietDoFtoC(double temp_f)

```
# include <iostream>
using namespace std;
//khai bao nguyen mau ham
double ChuyenNhietDoFtoC(double temp_f);
int main()
{
    //khai bao bien nhiet do c
    double nhietdoF;
    cout << "Moi ban nhap vao nhiet do F: ";
    cin >> nhietdoF;
    cout << "Nhiet do F: "<<nhietdoF
        << " chuyen sang nhiet do C: "
        << ChuyenNhietDoFtoC(nhietdoF) <<endl;
    system ("pause");
    return 0;
}
//chuong tring chuyen doi nhiet do F toi nhiet do C
double ChuyenNhietDoFtoC(double temp_f)
{
    return (temp_f - 32)/1.8;
}
```



```
C:\Users\hieunh3\Documents\Visual Stu...
Moi ban nhap vao nhiet do F: 120
Nhiet do F: 120 chuyen sang nhiet do C: 48.8889
Press any key to continue . . .
```

Gọi hàm bằng tên của nó: **ChuyenNhietDoFtoC** (nhietdoF)

Hàm **ChuyenNhietDoFtoC** được viết ra

Định Nghĩa Hàm

Tập những câu lệnh để thực hiện hành vi của nó khi nó được gọi.

Định nghĩa hàm có dạng:

```
return_type name (parameter- Declarations)
{
    statementList
}
```

Ví dụ:

```
const double PI = 3.14159;
double area (double length, double width)
{
    double halfLength = length/2.0, halfWidth = width/2.0;
    return PI * halfLength * halfWidth;
}
```

Nguyên Mẫu Hàm

- Dùng để khai báo hàm. Cho phép hàm đó có thể được gọi.
- Nguyên mẫu hàm phải được xuất hiện trước bất kỳ lời gọi nào hay định nghĩa hàm:
 - Ngược lại, trình biên dịch sẽ báo lỗi.
 - Trình biên dịch phải biết sự tồn tại của hàm đó.
- Dạng nguyên mẫu hàm:

`return_type name (parameter- Declarations);`

Nguyên Mẫu Hàm - Ví dụ

```
# include <iostream>
using namespace std;
//khai bao nguyen mau ham
double ChuyenNhietDoFtoC(double temp_f);
int main()
{
    //khai bao bien nhiet do c
    double nhietdoF;
    cout << "Moi ban nhap vao nhiet do F: ";
    cin >> nhietdoF;
    cout << "Nhiet do F: "<<nhietdoF
        << " chuyen sang nhiet do C: "
        << ChuyenNhietDoFtoC(nhietdoF) <<endl;
    system("pause");
    return 0;
}
//chuong trinh chuyen doi nhiet do F toi nhiet do C
double ChuyenNhietDoFtoC(double temp_f)
{
    return (temp_f - 32)/1.8;
}
```

Hàm là chương trình con

Các bước để thiết kế chương trình có thể được sử dụng để thiết kế hàm:

1. Hành vi.
2. Giải thuật.
3. Viết mã.
4. Kiểm tra, thực thi, phát hiện lỗi.
5. Bảo trì.

Hành vi của hàm gồm:

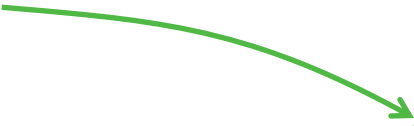
Nhận các giá trị từ hàm gọi

Trả về giá trị cho hàm gọi

Tham Số

Là các biến của hàm bị gọi mà hàm gọi có thể chỉ định giá trị cho nó.

Tham số được định nghĩa giữa cặp dấu ngoặc đơn trong phần định nghĩa hàm.



```
double ChuyenNhietDoFtoC (double temp_f)
{
    return (temp_f - 32)/1.8;
}
```

Đối Số

- Khi một hàm được gọi:

Hàm gọi có thể truyền giá trị của nó cho hàm bị gọi, các giá trị này được gọi là đối số.

Giá trị của đối số sẽ được chứa trong tham số của hàm bị gọi

```
double ChuyenNhietDoFtoC (140);  
double ChuyenNhietDoFtoC (double temp_f)  
{ return (temp_f - 32)/1.8; }
```




- Khi hàm này thực thi, nó sẽ sử dụng giá trị tham số của nó.

Phạm vi - Biến Cục Bộ

- Chương trình ví dụ của chúng ta chỉ sử dụng tham số `temp_f`.
- Tuy nhiên, nhiều hàm khác cần thêm các biến bên trong nó gọi là **biến cục bộ**.

```
int giatrilonnhat (int x, int y, int z)
{
    int max = x;
    if (y > max) max = y;
    if (z > max) max = z;
    return max;
}
```



- Biến cục bộ bị hủy đi khi sự thực thi thoát khỏi hàm (hay khối) chứa nó

Phạm vi - Biến Cục Bộ

- Tham số cũng được xem là biến cục bộ.
- Do đó, bên trong một hàm nếu biến cục bộ và tham số có cùng tên là lỗi cú pháp
- Ví dụ

```
Int tong2so (int a, int b)
```

```
{
```

```
int a;
```

```
return a+b;
```

```
}
```



Lỗi

Phạm vi - Biến Toàn Cục

Được tạo ra bằng cách đặt khai báo biến bên ngoài tất cả định nghĩa hàm

```
# include <iostream>
```

```
using namespace std;
```

```
int x = 1; // x là biến toàn cục
```

```
int main()
```

```
{
```

```
    //...
```

```
}
```

Biến toàn cục duy trì giá trị của nó trong suốt thời gian thực thi chương trình.



DEMO

Hàm trả giá trị,
Phạm vi biến



Hàm Không Trả Trị

Nhiều chương trình có các tác vụ thường được lặp lại:

- Hiển thị giá trị, thông báo.
- Đọc giá trị từ bàn phím, tập tin.

Các tác vụ này có thể được thực hiện bởi hàm không trả về một giá trị hay còn gọi là hàm **void**. Một hàm **void** đơn giản chỉ là **hàm không có giá trị trả về**.

Ví dụ:

```
void printDate (int,int,int);  
// in ra ngày dưới dạng hằng;  
int main()  
{ // kiểm tra hàm printDate    int month, day, year;  
    do{  
        cin >> month >> day >> year;  
        printDate(month,day,year);  
    }  
    while (month > 0);  
}
```

Gọi Hàm Không Trả Trị

- Cần phân biệt lời gọi hàm trả trị với lời gọi hàm không trả trị.
- Lời gọi hàm trả trị là biểu thức:
`double temp_C = ChuyenNhietDoFtoC (nhietdoF);`
- Lời gọi hàm không trả trị là câu lệnh:
`printDate (month, day, year);`

Một Số Hàm Toán Học

Hàm	Mô tả	Ví dụ
<code>acos(x)</code>	tính ngược cosin của x (theo radians)	<code>acos(0.2)</code> trả về 1.36944
<code>asin(x)</code>	tính ngược sin của x (theo radians)	<code>asin(0.2)</code> trả về 0.201358
<code>atan(x)</code>	tính ngược tang của x (theo radians)	<code>atan(0.2)</code> trả về 0.197396
<code>ceil(x)</code>	làm tròn x (làm tròn lên)	<code>ceil(3.141593)</code> trả về 4.0
<code>cos(x)</code>	cosin của x (theo radians)	<code>cos(2)</code> trả về -0.416147
<code>exp(x)</code>	hàm mũ của x (cơ số e)	<code>exp(2)</code> trả về 7.38906
<code>fabs(x)</code>	giá trị tuyệt đối của x	<code>fabs(-2)</code> trả về 2.0
<code>floor(x)</code>	làm tròn x (làm tròn xuống)	<code>floor(3.141593)</code> trả về 3.0
<code>log(x)</code>	logarithm tự nhiên của x (cơ số e)	<code>log(2)</code> trả về 0.693147
<code>log10(x)</code>	logarithm thường của x (cơ số 10)	<code>log10(2)</code> trả về 0.30103
<code>pow(x,p)</code>	x mũ p	<code>pow(2,3)</code> trả về 8.0
<code>sin(x)</code>	sin của x (theo radians)	<code>sin(2)</code> trả về 0.909297
<code>sqrt(x)</code>	căn bậc hai của x	<code>sqrt(2)</code> trả về 1.41421
<code>tan(x)</code>	tang của x (theo radians)	<code>tan(2)</code> trả về -2.18504



Hàm không trả giá trị và
một số hàm toán học

Truyền tham số

- Xét phép chia nguyên:
 - Cho kết quả là thương và phần dư.
 - Chúng ta muốn định nghĩa một hàm mà nó trả về hai giá trị này.
- Tuy nhiên, nếu sử dụng lệnh return thì mỗi hàm chỉ có thể trả về chỉ một giá trị.
- Các tham số mà chúng ta biết trước đây được gọi là tham trị (value parameter), nó là bản sao của các đối số của nó.
- Sự thay đổi giá trị của tham trị trong hàm bị gọi, chính là thay đổi giá trị của bản sao chứ không phải đối số thực của nó.

Giải quyết

- Tham số tham chiếu:
 - Các tham số được khai báo với dấu **&**
 - Dấu **&** được đặt sau kiểu của tham số nhưng trước tên của nó.
- Tham số tham chiếu là một bí danh (alias) cho đối số tương ứng của nó.
- Sự thay đổi giá trị của tham số tham chiếu sẽ thay đổi giá trị của đối số tương ứng của nó.

Ví dụ

Một hàm với các tham số tham chiếu:

```
void divideInts ( int op1, int op2, int& quotient, int& remainder)  
{ ... }
```

Các tham số **quotient** và **remainder** nhận giá trị thích hợp của các câu lệnh gán trong hàm bị gọi.

Các đối số tương ứng trong lời gọi hàm sẽ nhận cùng các giá trị này.

Tham Số

Có hai cách truyền đối số đến tham số:

- **Truyền bằng tham số giá trị:** tham số tương ứng được gọi là tham trị/tham số giá trị.
- **Truyền bằng tham chiếu:** tham số tương ứng được gọi là tham số tham chiếu.

Tham Trị

Là những biến riêng biệt trong bản sao của các đối số

Trong ví dụ:

```
void divideInts ( int op1, int op2, int& quotient, int& remainder)  
{ ... }
```

Ta thấy **op1** và **op2** là các tham trị.

Những đối tượng lớn được truyền bằng tham trị sẽ tốn thời gian và bộ nhớ đáng kể.

Tham Số Tham Chiếu

Một bí danh (tên thay thế) của đối số tương ứng trong lời gọi hàm.

Thay đổi giá trị của tham số tham chiếu sẽ thay đổi giá trị của đối số tương ứng.

Đối số tương ứng với tham số tham chiếu **phải là biến** và có cùng kiểu với tham số

```
void divideInts (int op1, int op2, int& quotient, int& remainder)
{
    assert(op2 != 0); quotient = op1 / op2;
    remainder = op1 % op2;
}
```

```
divideInts ( 5, 3, quot, rem );
```

Tham Chiếu hằng - const

- Tránh phí tổn thời gian truyền bản sao của đối số trong truyền bằng trị.
- Tạo sự an toàn (không cho phép sửa đổi giá trị đối số) trong cách truyền bằng tham chiếu.

```
void const_Ref (const int& x)
{
    ...
}
```

Ví dụ:

```
void f(int x, int& y, const int& z)
{
    x += z;
    y += z;
    cout << "x = " << x << ", y = " << y << ", z = " << z << endl;
}
```

Sử Dụng Tham Số

- Nếu hàm bị gọi chỉ nhận giá trị từ hàm gọi mà kiểu giá trị đó là kiểu dữ liệu cơ bản thì sử dụng tham trị.
- Nếu hàm bị gọi chỉ nhận giá trị từ hàm gọi mà giá trị đó là những đối tượng có kiểu dữ liệu lớn thì sử dụng tham số tham chiếu **const**.
- Nếu chỉ có một giá trị được truyền trở lại hàm gọi thì hàm bị gọi là hàm trả trị thông qua lệnh **return**.
- Nếu có nhiều giá trị được truyền trở lại hàm gọi thì hàm bị gọi là hàm không trả trị (**void**) và sử dụng tham số tham chiếu cho những giá trị đó.

Truyền qua giá trị và truyền qua tham chiếu

Truyền qua giá trị	Truyền qua tham chiếu
<ul style="list-style-type: none">- <code>int x;</code>- Tham số <code>x</code> là một biến cục bộ.- Nó chỉ là bản sao của đối số tương ứng.- Nó không thể thay đổi giá trị của đối số.- Đối số được truyền qua giá trị có thể là một hằng số, một biến hay một biểu thức.- Đối số là chỉ-đọc.	<ul style="list-style-type: none">- <code>int &x;</code>- Tham số <code>x</code> là tham chiếu cục bộ.- Nó là một <u>bí danh</u> của đối số.- Nó có thể thay đổi giá trị của đối số.- Đối số truyền qua tham chiếu phải là một biến.- Đối số có thể đọc-và-ghi.



DEMO

Truyền tham số



Hàm nội tuyến - Hàm inline

- Khi một hàm gọi hàm khác

```
void f ( int n )  
{  
    ...  
    x = g(n);  
    ...  
}
```

sự thực thi chuyển từ hàm này sang hàm khác, do đó thời gian thực thi tăng.

- Để tránh chi phí thời gian này, C++ cung cấp hàm inline.
- Hàm inline nên sử dụng cho những hàm nhỏ, được gọi thường xuyên.

Hàm nội tuyến - Hàm inline

- Để định nghĩa hàm inline, sử dụng từ khóa inline trong nguyên mẫu và trong định nghĩa hàm.

```
inline double ChuyenNhietDoFtoC (double temp);
```

...

```
inline double ChuyenNhietDoFtoC (double temp)  
{ return (temp – 32.0)/1.8; }
```

- Trình biên dịch sẽ đặt mã lệnh của hàm inline vào mỗi lời gọi hàm trong hàm gọi để tránh sự gọi hàm.

Nạp Chồng Hàm

- Các hàm được phân biệt với nhau bằng chữ ký (tên hàm và các tham số) của nó.
- Nhiều hàm có thể có cùng tên nhưng phải:
 - Khác nhau về số tham số, hoặc
 - Khác nhau về kiểu của các tham số.
- Khi điều này xảy ra, chúng ta nói rằng các hàm được nạp chồng.

Ví dụ

```
int tong( int n );
```

```
int tong( int m, int n );
```



DEMO về nạp chồng hàm

Hàm `main()`

- Mọi chương trình C++ đều bắt buộc có một hàm `main()`.
- Thực chất, chương trình đầy đủ được tạo thành từ hàm `main()` kết hợp với tất cả các hàm khác được gọi trực tiếp hay gián tiếp từ hàm này.
- Chương trình bắt đầu từ việc gọi hàm `main()` đầu tiên.
- Hàm `main()` là một hàm có kiểu trả về là `int`, nên nó thường được kết thúc bằng lệnh `return 0;`
- Có tất cả 4 cách để kết thúc một chương trình đột ngột:
 - Sử dụng lệnh `return` trong hàm `main()`;
 - Gọi hàm `exit()`;
 - Gọi hàm `abort()`;
 - Ném ra một ngoại lệ tự do.

Hàm nhị phân

- Là hàm trả về giá trị **true** or **false**
- Hàm có sẵn và hàm tự định nghĩa
- Hàm có sẵn như: **isdigit()**, **islower()**, **isupper()**, **isspace()**, **iscntrl()**, **ispunct()**;
- Khai báo hàm tự định nghĩa:

```
bool ten_ham (tham số)
```

```
{
```

```
    statement;
```

```
    return true or false;
```

```
}
```



DEMO

Hàm nhị phân



Tổng kết

- Định nghĩa hàm, gọi hàm.
- Nguyên mẫu hàm.
- Các bước thiết kế hàm.
- Phạm vi.
- Hàm không trả trị.
- Một số hàm toán học.
- Xét chi tiết cách truyền tham số.
- Hàm nội tuyến - hàm inline.
- Nạp chồng hàm.
- Hàm `main()`, hàm nhị phân

Chuẩn bị bài sau

Sinh viên đọc sách và slide trước bài học kế tiếp về Mảng và Xâu ký tự gồm:

- Khai báo, khởi tạo, và truy xuất mảng một chiều
- Xử lý mảng bằng vòng lặp for.
- Truyền mảng đến hàm.
- Cơ chế typedef
- Một số phương pháp sắp xếp, tìm kiếm trên mảng
- Mảng đa chiều
- Xâu ký tự chuẩn
- Tập (file)



FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn