ĐẠI NAM
UNIVERSITY

Phần trình bày của:

ĐẬU HẢI PHONG

Giảng viên

Đại Nam, ngày 19 tháng 01 năm 2023

LƯU Ý

KHÔNG NÓI CHUYỆN RIÊNG

KHÔNG SỬ DỤNG ĐIỆN THOẠI

KHÔNG NGỦ GẬT

GHI CHÉP ĐẦY ĐỦ

ĐẠI NAM
UNIVERSITY

LẬP TRÌNH CƠ BẢN

http://dainam.edu.vn

CHƯƠNG 5

HÀM - FUNCTION

NỘI DUNG

- Giới thiệu hàm
- Định nghĩa và Gọi hàm
- Truyền dữ liệu vào trong hàm
- Một số vấn đề liên quan
- Bài tập

Giới thiệu hàm

- Lập trình theo mô đun: chia chương trình thành các hàm, mô đun nhỏ để dễ dàng quản lý.
- Hàm: là tập hợp các câu lệnh thực hiện 1 nhiệm vụ
- **Mục tiêu:**
  - Cải thiện khả năng duy trì, phục hồi của chương trình
  - Đơn giản quá trình viết chươn trình.

This program has one long, complex function containing all of the statements necessary to solve a problem.

```
int main()
{
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
}
```

In this program the problem has been divided into smaller problems, each of which is handled by a separate function.

```
int main()
{
   statement;
   statement;
   statement;
}
```

main function

```
void function2()
{
   statement;
   statement;
   statement;
}
```

function 2

```
void function3()
{
   statement;
   statement;
   statement;
}
```

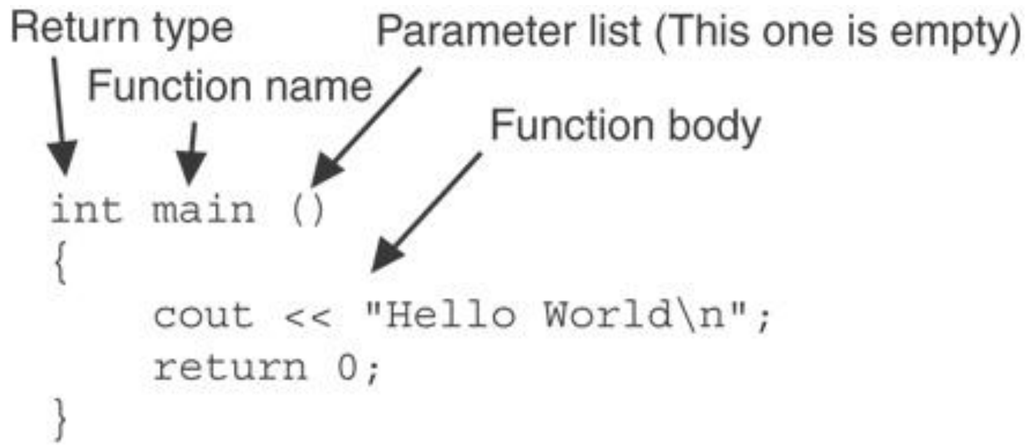
function 3

```
void function4()
{
   statement;
   statement;
   statement;
}
```

function 4

Định nghĩa & Gọi hàm

- Gọi hàm: câu lệnh để hàm thực hiện
- Định nghĩa hàm: câu lệnh tạo ra 1 hàm
- Định nghĩa hàm gồm:
  - *Kiểu trả về*: kiểu dữ liệu của giá trị mà hàm trả về.
  - *Tên hàm*: tên của hàm, đặt tên hàm theo như tắc đặt tên biến.
  - *Tham số*: Các biến chứa giá trị được truyền cho hàm
  - *Thân hàm*: Các câu lệnh thực hiện nhiệm vụ của hàm. Bao bọc bởi {}



Kiểu trả về của hàm & Lời gọi hàm

- Nếu một hàm trả về một giá trị thì kiểu giá trị phải chỉ ra:
  - int main()
- Nếu một hàm không trả về một giá trị thì kiểu là void

```
void printHeading()
{
  cout << "Monthly Sales\n";
}
```

- Để gọi hàm, sử dụng tên hàm theo sau là () và ;
  - printHeading();
- Khi gọi hàm thì các câu lệnh trong thân hàm được gọi sẽ thực hiện.
- Sau khi hàm kết thúc thì thực hiện tiếp tại nơi gọi hàm

Kiểu trả về của hàm & Lời gọi hàm

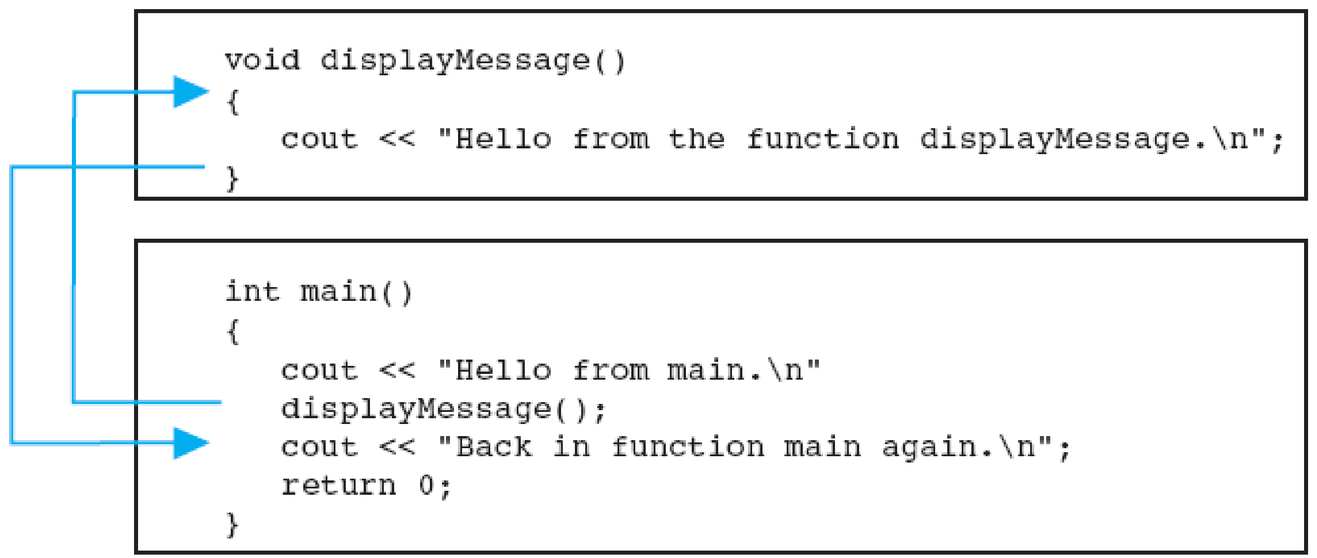
Program 6-1

```
1  // This program has two functions: main and displayMessage
2  #include <iostream>
3  using namespace std;
4
5  //*****************************************
6  // Definition of function displayMessage  *
7  // This function displays a greeting.     *
8  //*****************************************
9
10 void displayMessage()
11 {
12    cout << "Hello from the function displayMessage.\n";
13 }
14
15 //*****************************************
16 // Function main                          *
17 //*****************************************
18
19 int main()
20 {
21    cout << "Hello from main.\n";
22    displayMessage();
23    cout << "Back in function main again.\n";
24    return 0;
25 }
```

Program Output

```
Hello from main.
Hello from the function displayMessage.
Back in function main again.
```

Dòng điều khiển trong Program 6-1



Lời gọi hàm

- Hàm main có thể gọi bất kỳ hàm nào
- Hàm có thể gọi bởi 1 hàm khác
- Trình biên dịch cần biết về một hàm trước khi gọi:
  - Tên hàm
  - Kiểu trả về
  - Số lượng các tham số
  - Kiểu của từng tham số

Hàm nguyên mẫu - Function Prototypes

- Cách thông báo cho trình biên dịch về một hàm trước khi gọi hàm đến hàm:
  - Nơi định nghĩa hàm phải trước lời gọi hàm
  - Hàm nguyên mẫu – giống như định nghĩa hàm nhưng chưa có thân hàm:
    - Tiêu đề hàm: void printHeading()
    - Hàm nguyên mẫu: void printHeading();
- Chú ý:
  - Đặt hàm nguyên mẫu gần đầu chương trình
  - Chương trình phải bao gồm hàm nguyên mẫu và định nghĩa hàm trước khi có lời gọi hàm.
  - Khi có hàm nguyên mẫu thì có thể định nghĩa bất kỳ ở đâu trong tệp chương trình.

Hàm nguyên mẫu - Function Prototypes

Program 6-5

```
1  // This program has three functions: main, First, and Second.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototypes
6  void first();
7  void second();
8
9  int main()
10 {
11    cout << "I am starting in function main.\n";
12    first();    // Call function first
13    second();   // Call function second
14    cout << "Back in function main again.\n";
15    return 0;
16 }
17
```

```
18 //*****************************************
19 // Definition of function first.          *
20 // This function displays a message.      *
21 //*****************************************
22
23 void first()
24 {
25    cout << "I am now inside the function first.\n";
26 }
27
28 //*****************************************
29 // Definition of function second.         *
30 // This function displays a message.      *
31 //*****************************************
32
33 void second()
34 {
35    cout << "I am now inside the function second.\n";
36 }
```

Truyền dữ liệu vào trong hàm

- Có thể truyền giá trị vào trong hàm lúc gọi hàm
  - c = pow(a, b); d = pow(3,6);
- Giá trị truyền cho hàm là các đối số
- Các biến trong hàm chứa các giá trị truyền như các đối số là các tham số.

```
void displayValue(int num)
{
   cout << "The value is " << num << endl;
}
```

- Biến num là *tham số*, chấp nhận mọi giá trị nguyên được truyền cho hàm.

Truyền dữ liệu vào trong hàm

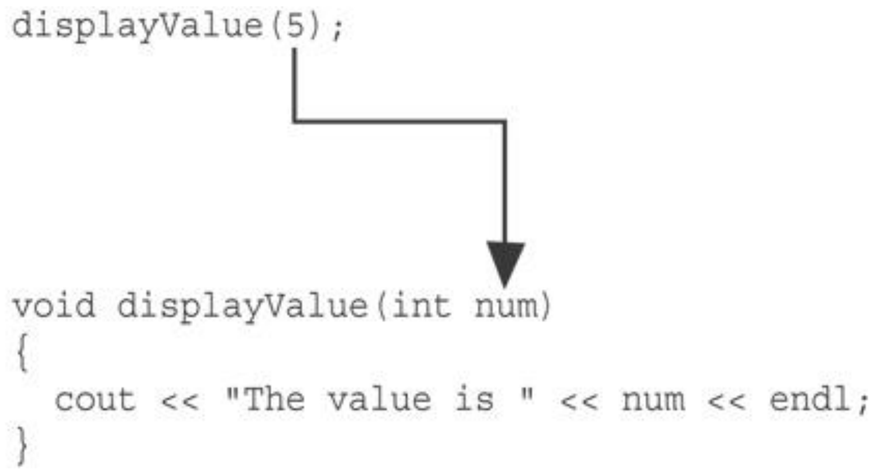
Program 6-6

```
1  // This program demonstrates a function with a parameter.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototype
6  void displayValue(int);
7
8  int main()
9  {
10    cout << "I am passing 5 to displayValue.\n";
11    displayValue(5);  // Call displayValue with argument 5
12    cout << "Now I am back in main.\n";
13    return 0;
14 }
15
16 //*************************************************************
17 // Definition of function displayValue.                       *
18 // It uses an integer parameter whose value is displayed.     *
19 //*************************************************************
20
21 void displayValue(int num)
22 {
23    cout << "The value is " << num << endl;
24 }
```

Program Output

```
I am passing 5 to displayValue.
The value is 5
Now I am back in main.
```

Lời gọi hàm ở dòng 11 truyền giá trị 5 là một đối số của hàm



Thuật ngữ khác của tham số

- Tham số cũng có thể gọi là tham số hình thức hoặc đối số hình thức
- Đối số có thể được gọi là tham số thực sự hoặc đối số thực sự
- Với mỗi đối số của hàm:
  - Hàm nguyên mẫu phải bao gồm kiểu dữ liệu của mỗi tham số bên trong dấu ngoặc.
  - Tiêu đề hàm phải bảo gồm khai báo tên các tham số bên trong
- Ví dụ:

```
void evenOrOdd(int);  //Hàm nguyên mẫu
void evenOrOdd(int num) //Tiêu đề hàm
evenOrOdd(val);       //Lời gọi hàm
```

Chú ý khi gọi hàm

- Giá trị của đối số sẽ được *truyền* vào cho tham số khi gọi hàm
- Tham số chỉ có phạm vi trong hàm sử dụng nó.
- Hàm có thể có nhiều tham số
- Với mỗi tham số phải có kiểu dữ liệu được liệt kê trong nguyên mẫu và khai báo đối số trong tiêu đề hàm
- Khi gọi hàm và truyền nhiều tham số:
  - Số lượng đối số phải trùng với nguyên mẫu và định nghĩa
  - Đối số đầu sẽ khởi tạo cho tham số đầu, đối số thứ 2 sẽ khởi tạo cho tham số thứ 2,…

Chú ý khi gọi hàm

Program 6-8

```
1  // This program demonstrates a function with three parameters.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototype
6  void showSum(int, int, int);
7
8  int main()
9  {
10    int value1, value2, value3;
11
12    // Get three integers.
13    cout << "Enter three integers and I will display ";
14    cout << "their sum: ";
15    cin >> value1 >> value2 >> value3;
16
17    // Call showSum passing three arguments.
18    showSum(value1, value2, value3);
19    return 0;
20 }
21
```

Lời gọi hàm ở dòng 18 sẽ truyền giá trị value1, value2, value3 như là một đối số của hàm

Function Call → `showSum(value1, value2, value3)`

```
void showSum(int num1, int num2, int num3)
{
      cout << (num1 + num2 + num3) << endl;
}
```

```
22 //*********************************************************
23 // Definition of function showSum.                        *
24 // It uses three integer parameters. Their sum is displayed. *
25 //*********************************************************
26
27 void showSum(int num1, int num2, int num3)
28 {
29    cout << (num1 + num2 + num3) << endl;
30 }
```

Program Output with Example Input Shown in Bold

```
Enter three integers and I will display their sum: 4 8 7 [Enter]
19
```

Truyền dữ liệu bằng giá trị

- Truyền bằng giá trị: khi một đối số được truyền cho hàm thì giá trị của nó sẽ truyền cho tham số
- Thay đổi tham số trong hàm không ảnh hưởng đến giá trị của đối số
- Ví dụ:

```
int val=5;
evenOrOdd(val);
```

  - **evenOrOdd** có thể thay đổi biến **num**, nhưng nó không ảnh hưởng đến biến **val**

Kiểm tra

- *5.5. Trong các dòng dưới đây, đâu là hàm nguyên mẫu, lời gọi hàm và tiêu đề hàm?*

  A. void showNum(double num)

  B. void showNum(double);

  C. showNum(45.67);

- *5.6. Viết một hàm có tên là* timesTen, *hàm có tham số tên là number kiểu nguyên. Khi hàm* timesTen *được gọi sẽ hiển thị 10 lần số number.*
- *5.7. Viết hàm nguyên mẫu cho hàm timesTen ở câu 5.6.*

Kiểm tra

- *5.8. Đầu ra của các đoạn chương trình sau là gì?*

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

void showDouble(int); // Function prototype

int main()
{
   int num;

   for (num = 0; num < 10; num++)
      showDouble(num);
   return 0;
}

// Definition of function showDouble.
void showDouble(int value)
{
   cout << value << "\t" << (value * 2) << endl;
}
```

```
int main()
{
   int x = 0;
   double y = 1.5;

   cout << x << " " << y << endl;
   func1(y, x);
   cout << x << " " << y << endl;
   return 0;
}

 void func1(double a, int b)
 {
    cout << a << " " << b << endl;
    a = 0.0;
    b = 10;
    cout << a << " " << b << endl;
 }
```

Sử dụng hàm trong thực đơn chương trình

- Hàm có thể được sử dụng để:
  - Thực hiện sự lựa chọn của người dùng từ thực đơn
  - Thực hiện các nhiệm vụ chung, giảm thiểu số lượng hàm, thời gian và tốc độ phát triển chương trình.
- Giới thiệu Program 6-10 (pr6-14.cpp):

Câu lệnh return

- Sử dụng để kết thúc thực hiện của 1 hàm
- Có thể để ở mọi nơi trong hàm.
- Khi gặp return thì các câu lệnh phía dưới không thực hiện.
- Hàm trả về void thì không có return ở cuối hàm

Program 6-11

```
1  // This program uses a function to perform division. If di
2  // by zero is detected, the function returns.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype.
7  void divide(double, double);
8
9  int main()
10 {
11    double num1, num2;
12
13    cout << "Enter two numbers and I will divide the first\
14    cout << "number by the second number: ";
15    cin >> num1 >> num2;
16    divide(num1, num2);
17    return 0;
18 }
```

```
20 //*************************************************************
21 // Definition of function divide.                            *
22 // Uses two parameters: arg1 and arg2. The function divides arg1*
23 // by arg2 and shows the result. If arg2 is zero, however, the  *
24 // function returns.                                          *
25 //*************************************************************
26
27 void divide(double arg1, double arg2)
28 {
29    if (arg2 == 0.0)
30    {
31       cout << "Sorry, I cannot divide by zero.\n";
32       return;
33    }
34    cout << "The quotient is " << (arg1 / arg2) << endl;
35 }
```

Program Output with Example Input Shown in Bold

```
Enter two numbers and I will divide the first
number by the second number: 12 0 [Enter]
Sorry, I cannot divide by zero.
```

Trả về một giá trị từ tên hàm

- Một hàm có thể trả về một giá trị qua câu lệnh gọi hàm
- VD: như hàm pow dưới đây trả về 1 giá trị.

```
double x;
x = pow(2.0,10.0);
```

- Trong hàm có giá trị trả về thì câu lệnh return được sử dụng để trả về giá trị từ hàm.

Return Type

```
int sum(int num1, int num2)
{
  int result;
  result = num1 + num2;
  return result;
}
```

Value Being Returned

```
int sum(int num1, int num2)
{
    return num1 + num2;
}
```

Trả về một giá trị từ tên hàm

Program 6-12

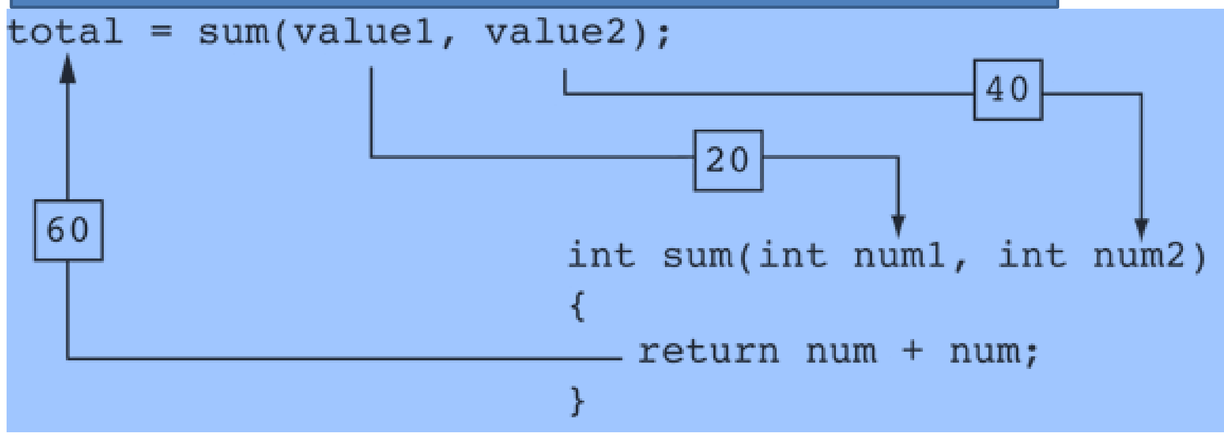
```
1  // This program uses a function that returns a value.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  int sum(int, int);
7
8  int main()
9  {
10    int value1 = 20,    // The first value
11        value2 = 40,    // The second value
12        total;          // To hold the total
13
14    // Call the sum function, passing the contents of
15    // value1 and value2 as arguments. Assign the return
16    // value to the total variable.
17    total = sum(value1, value2);
18
19    // Display the sum of the values.
20    cout << "The sum of " << value1 << " and "
21         << value2 << " is " << total << endl;
22    return 0;
23 }
```

```
24
25 //*********************************************************
26 // Definition of function sum. This function returns     *
27 // the sum of its two parameters.                         *
28 //*********************************************************
29
30 int sum(int num1, int num2)
31 {
32    return num1 + num2;
33 }
```

Program Output

```
The sum of 20 and 40 is 60
```

Dòng 17: value1, value2 là các tham số. Giá trị trả về gán cho total



Trả về một giá trị từ tên hàm

pr6-12.cpp

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

//Function prototypes
double getRadius();
double square(double number);

int main()
{
    const double PI = 3.14159;   // Constant for pi
    double radius;               // Holds the circle's radius
    double area;                 // Holds the circle's area

    // Set the numeric output formatting
    cout << fixed << showpoint << setprecision(2);

    // Get the radius of the circle
    cout << "This program calculates the area of a circle.\n";
    radius = getRadius();

    // Caclulate the area of the circle
    area = PI * square(radius);

    // Display the area
    cout << "The area is " << area << endl;
    return 0;
}
```

```cpp
/*****************************************
*               getRadius                *
* This function returns the circle radius *
* input by the user.                      *
*****************************************/
double getRadius()
{
    double rad;

    cout << "Enter the radius of the circle: ";
    cin  >> rad;
    return rad;
}

/*****************************************
*                 square                 *
* This function returns the square of the *
* double argument sent to it              *
*****************************************/
double square(double number)
{
    return number * number;
}
```

```
area = PI * square(radius);
                    10
          100
                    double square(double number)
                    {
                        return number * number;
                    }
```

Trả về một giá trị từ tên hàm

- Nguyên mẫu và định nghĩa hàm phải xác định kiểu giá trị trả về. Trừ hàm trả về void (không có giá trị trả về)
- Lời gọi hàm có giá trị trả về:
  - Gán cho 1 biến
    - VD: double x = pow(2.0,10.0);
  - Gửi cho hàm cout
    - VD: cout<<pow(2.0,10.0);
  - Sử dụng trong 1 biểu thức
    - VD: double x = y * pow(2.0,10.0);

Hàm trả về kiểu bool

- Hàm có thể trả về kiểu true hoặc false
- Khai báo kiểu trả về ở nguyên mẫu hoặc phần tiêu đề hàm sẽ là bool.
- Trong thân hàm phải có câu lệnh return true hoặc false
- Lời gọi hàm có thể sử dụng giá trị trả về trong 1 biểu thức lôgic

Hàm trả về kiểu bool

Program 6-15

```
1  // This program uses a function that returns true or false.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  bool isEven(int);
7
8  int main()
9  {
10    int val;
11
12    // Get a number from the user.
13    cout << "Enter an integer and I will tell you
14    cout << "if it is even or odd: ";
15    cin >> val;
16
17    // Indicate whether it is even or odd.
18    if (isEven(val))
19       cout << val << " is even.\n";
20    else
21       cout << val << " is odd.\n";
22    return 0;
23 }
24
```

```
25 //*************************************************************
26 // Definition of function isEven. This function accepts an    *
27 // integer argument and tests it to be even or odd. The function  *
28 // returns true if the argument is even or false if the argument  *
29 // is odd. The return value is a bool.                        *
30 //*************************************************************
31
32 bool isEven(int number)
33 {
34    bool status;
35
36    if (number % 2 == 0)
37       status = true;  // The number is even if there is no remainder.
38    else
39       status = false; // Otherwise, the number is odd.
40    return status;
41 }
```

Program Output with Example Input Shown in Bold

```
Enter an integer and I will tell you if it is even or odd: 5 [Enter]
5 is odd.
```

Kiểm tra

- *5.11. Một hàm có thể có thể trả về bao nhiêu giá trị qua tên hàm?*
- *5.12. Viết tiêu đề cho hàm có tên là distance. Hàm trả về 1 giá trị double và có 2 tham số: rate, time kiểu double.*
  - *double distance(double rate, double time)*
- *5.13. Viết tiêu đề cho hàm có tên là days. Hàm trả về 1 giá trị int và có 3 tham số: year, month, weeks kiểu int.*
  - *int days(int year, int month, int weeks)*
- *5.14. Viết tiêu đề cho hàm có tên là getKey. Hàm trả về 1 giá trị kiểu char và không có tham số.*
- *5.15. Viết tiêu đề cho hàm có tên là lightYears. Hàm trả về 1 giá trị long và có 1 tham số: miles kiểu long.*

Biến toàn cục và biến cục bộ

- **Biến cục bộ:**
  - Các biến định nghĩa bên trong hàm gọi là biến cục bộ.
    - Không thể sử dụng ở 1 hàm khác.
    - Trong các hàm khác nhau có thể đặt cùng 1 tên biến
  - Biến cục bộ chỉ tồn tại khi chạy hàm chứa nó
  - Khi bắt đầu 1 hàm thì biến cục bộ và biến tham số được tạo trong bộ nhớ. Khi hàm kết thúc thì sẽ phá hủy.

Biến toàn cục và biến cục bộ

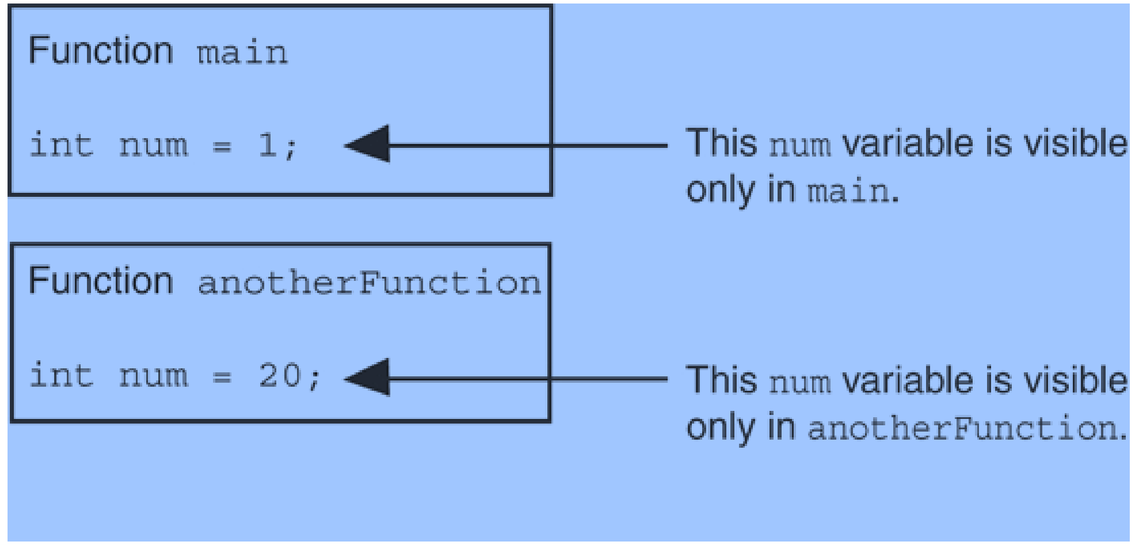
Program 6-16

```
1  // This program shows that variables defined in a function
2  // are hidden from other functions.
3  #include <iostream>
4  using namespace std;
5
6  void anotherFunction(); // Function prototype
7
8  int main()
9  {
10    int num = 1;    // Local variable
11
12    cout << "In main, num is " << num << endl;
13    anotherFunction();
14    cout << "Back in main, num is " << num << endl;
15    return 0;
16 }
17
18 //*********************************************************
19 // Definition of anotherFunction                          *
20 // It has a local variable, num, whose initial value  *
21 // is displayed.                                          *
22 //*********************************************************
23
24 void anotherFunction()
25 {
26    int num = 20;  // Local variable
27
28    cout << "In anotherFunction, num is " << num << endl;
29 }
```

Program Output

```
In main, num is 1
In anotherFunction, num is 20
Back in main, num is 1
```

Khi chương trình đang thực hiện trong hàm main thì biến num trong main có tác dụng. Khi hàm `anotherFunction` được gọi thì biến num trong hàm `anotherFunction` có tác dụng và biến num trong hàm main bị ẩn đi.



Biến toàn cục và biến cục bộ

- **Biến toàn cục:**
  - Là biến có tác dụng lên tất cả các hàm trong chương trình.
  - Phạm vi hoạt động từ đầu đến cuối chương trình
  - Có nghĩa là biến toàn cục có thể truy cập trong tất cả các hàm sau khi biến toàn cục được khai báo.
  - Nên hạn chế biến toàn cục vì làm cho gỡ lỗi khó hơn.
  - Tương tự, có thể tạo hằng toàn cục.
- **Khởi tạo biến Toàn cục và Cục bộ:**
  - Biến cục bộ không tự động khởi tạo, mà thường do lập trình viên.
  - Biến toàn cục, khi định nghĩa tự động khởi tạo giá trị bằng 0 (số) hoặc NULL (kí tự)

Biến toàn cục và biến cục bộ

Program 6-19

```
1  // This program calculates gross pay.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  // Global constants
7  const double PAY_RATE = 22.55;     // Hourly pay rate
8  const double BASE_HOURS = 40.0;    // Max non-overtime hours
9  const double OT_MULTIPLIER = 1.5; // Overtime multiplier
10
11 // Function prototypes
12 double getBasePay(double);
13 double getOvertimePay(double);
14
15 int main()
16 {
17    double hours,              // Hours worked
18           basePay,            // Base pay
19           overtime = 0.0,     // Overtime pay
20           totalPay;           // Total pay
```

Định nghĩa hằng toàn cục cho những giá trị không thay đổi trong suốt quá trình chương trình thực hiện.

```
29       // Get overtime pay, if any.
30       if (hours > BASE_HOURS)
31          overtime = getOvertimePay(hours);
```

```
56    // Determine base pay.
57    if (hoursWorked > BASE_HOURS)
58       basePay = BASE_HOURS * PAY_RATE;
59    else
60       basePay = hoursWorked * PAY_RATE;
```

```
75    // Determine overtime pay.
76    if (hoursWorked > BASE_HOURS)
77    {
78       overtimePay = (hoursWorked - BASE_HOURS) *
79                PAY_RATE * OT_MULTIPLIER;
```

Biến cục bộ tĩnh – Static Local variables

- Biến cục bộ chỉ tồn tại khi hàm thực hiện, sau khi hàm kết thúc thì nội dung của biến sẽ mất.
- Biến cục bộ tĩnh (static): sẽ giữ lại giá trị giữa các lời gọi hàm.
- Biến cục bộ tĩnh chỉ được định nghĩa và khởi tạo lần đầu tiên khi hàm thực hiện. Giá trị khởi tạo mặc định là 0.

Biến cục bộ tĩnh – Static Local variables

Program 6-21

```
1  // This program shows that local variables do not retain
2  // their values between function calls.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype
7  void showLocal();
8
9  int main()
10 {
11    showLocal();
12    showLocal();
13    return 0;
14 }
15
```

Mỗi lần `showLocal` được gọi thì biến `localNum` được tạo lại và khởi tạo giá trị là 5.

Program 6-21 *(continued)*

```
16 //*********************************************************
17 // Definition of function showLocal.                      *
18 // The initial value of localNum, which is 5, is displayed. *
19 // The value of localNum is then changed to 99 before the   *
20 // function returns.                                       *
21 //*********************************************************
22
23 void showLocal()
24 {
25    int localNum = 5; // Local variable
26
27    cout << "localNum is " << localNum << endl;
28    localNum = 99;
29 }
```

Program Output

```
localNum is 5
localNum is 5
```

Biến cục bộ tĩnh – Static Local variables

Program 6-22

```
1  // This program uses a static local variable.
2  #include <iostream>
3  using namespace std;
4
5  void showStatic(); // Function prototype
6
7  int main()
8  {
9     // Call the showStatic function five tim
10    for (int count = 0; count < 5; count++)
11       showStatic();
12    return 0;
13 }
14
```

Program 6-21 *(continued)*

Program 6-22

```
16 // Definition of function showStatic.                          *
17 // statNum is a static local variable. Its value is displayed  *
18 // and then incremented just before the function returns.       *
19 //*************************************************************
20
21 void showStatic()
22 {
23    static int statNum;
24
25    cout << "statNum is " << statNum << endl;
26    statNum++;
27 }
```

Program Output

```
statNum is 0
statNum is 1
statNum is 2
statNum is 3
statNum is 4
```

`statNum` tự độ khởi tạo giá trị 0. Chú ý: statNum duy trì giá trị của nó giữa các lần gọi hàm.

Biến cục bộ tĩnh – Static Local variables

- Nếu không khởi tạo biến cục bộ tĩnh thì biến chỉ khởi tạo 1 lần. Xem Program 6-23 dưới đây.

```
16  //*********************************************************
17  // Definition of function showStatic.                     *
18  // statNum is a static local variable. Its value is displayed *
19  // and then incremented just before the function returns.  *
20  //*********************************************************
21
22  void showStatic()
23  {
24     static int statNum = 5;
25
26     cout << "statNum is " << statNum << endl;
27     statNum++;
28  }
```

Program Output

```
statNum is 5
statNum is 6
statNum is 7
statNum is 8
statNum is 9
```

Kiểm tra

- *5.16. Sự khác nhau giữa biến cục bộ tĩnh và biến toàn cục?*
- *5.17. Đầu ra của những đoạn chương trình sau là sau là gì?*

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

void myFunc();  // Function prototype

int main()
{
   int var = 100;

   cout << var << endl;
   myFunc();
   cout << var << endl;
   return 0;
}

// Definition of function myFunc
void myFunc()
{
   int var = 50;

   cout << var << endl;
}
```

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

void showVar(); // Function prototype

int main()
{
   for (int count = 0; count < 10; count++)
      showVar();
   return 0;
}

// Definition of function showVar
void showVar()
{
   static int var = 10;

   cout << var << endl;
   var++;
}
```

Đối số mặc định

- Đối số mặc định là đối số tự động truyền cho các tham số khi *quên* trong lời gọi hàm.
- Trong hàm nguyên mẫu phải có khai báo
  - void evenOrOdd(int = 0);
- Có thể được khai báo trong phần đầu của hàm nếu không có hàm nguyên mẫu.
- Có thể có đối số cho nhiều tham số của hàm
  - int getSum(int, int=0, int=0);

Đối số mặc định

Default arguments specified in the prototype

Program 6-24

```
1  // This program demonstrates default function arguments.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype with default arguments
6  void displayStars(int = 10, int = 1);
7
8  int main()
9  {
10    displayStars();      // Use default values for cols and
11    cout << endl;
12    displayStars(5);     // Use default value for rows.
13    cout << endl;
14    displayStars(7, 3);  // Use 7 for cols and 3 for rows.
15    return 0;
16 }
```

```
18 //**********************************************************
19 // Definition of function displayStars.                    *
20 // The default argument for cols is 10 and for rows is 1.*
21 // This function displays a square made of asterisks.      *
22 //**********************************************************
23
24 void displayStars(int cols, int rows)
25 {
26    // Nested loop. The outer loop controls the rows
27    // and the inner loop controls the columns.
28    for (int down = 0; down < rows; down++)
29    {
30       for (int across = 0; across < cols; across++)
31          cout << "*";
32       cout << endl;
33    }
34 }
```

Program Output

```
**********

*****

*******
*******
*******
```

Đối số mặc định

- Nếu tất cả các tham số của hàm không có giá trị mặc định thì các tham số không có giá trị mặc định được khai báo đầu tiên:
  - int getSum(int, int=0, int=0); // OK
  - int getSum(int, int=0, int); // NO
- Khi gọi hàm có đối số mặc định được bỏ qua từ lời gọi hàm thì tất cả các đối số sau nó cũng được bỏ qua.
  - sum = getSum(num1, num2); // OK
  - sum = getSum(num1, , num3); // NO

Tham chiếu

- Sử dụng biến tham chiếu như tham số:
  - Một cơ chế cho phép hàm làm việc với đối số ban đầu từ lời gọi hàm (không phải là copy đối số)
  - Cho phép hàm **thay đổi** giá trị lưu trữ trong quá trình **gọi**.
  - Cho phép hàm trả về (return) hơn 1 giá trị.
- Truyền qua tham chiếu:
  - Một biến tham chiếu như **1 bí danh** cho **1 biến khác**
  - Định nghĩa với dấu và (&).
  - Thay đổi đối với biến tham chiếu được thực hiện cho biến mà nó tham chiếu đến.
  - Sử dụng biến tham chiếu để thực hiện truyền tham số bằng tham chiếu.

Tham chiếu

& trong hàm nguyên mẫu để xác định tham số là 1 tham chiếu

Program 6-25

```
1  // This program uses a reference variable as a function
2  // parameter.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype. The parameter is a reference variable.
7  void doubleNum(int &);
8
9  int main()
10 {
11    int value = 4;
12
13    cout << "In main, value is " << value << endl;
14    cout << "Now calling doubleNum..." << endl;
15    doubleNum(value);
16    cout << "Now back in main. value is " << value << endl
17    return 0;
18 }
19
```

Truyền giá trị bằng tham chiếu.

Dấu & xuất hiện trong tiêu đề hàm.

```
20 //*************************************************************
21 // Definition of doubleNum.                                   *
22 // The parameter refVar is a reference variable. The value    *
23 // in refVar is doubled.                                      *
24 //*************************************************************
25
26 void doubleNum (int &refVar)
27 {
28    refVar *= 2;
29 }
```

Program Output

```
In main, value is 4
Now calling doubleNum...
Now back in main. value is 8
```

Tham chiếu

- **Chú ý:**
  - Mỗi tham số là tham chiếu cần có dấu & phía trước
  - Khoảng cách giữa kiểu và dấu & là không quan trọng
  - Phải sử dụng dấu & ở cả hàm nguyên mẫu và tiêu đề hàm
  - Đối số truyền vào cho tham chiếu **phải là 1 biến**, không thể là 1 biểu thức hoặc hằng số
  - Sử dụng khi thích thích hợp, không sử dụng khi đối số không thay đổi trong hàm hoặc hàm cần trả về hơn 1 giá trị.

Kiểm tra

- *5.19. Những kiểu giá trị nào có thể được dùng như đối số mặc định?*
- *5.20. Viết hàm nguyên mẫu và tiêu đề hàm có tên là computer. Hàm có 3 tham số là 3 kiểu: int, double, long (không nhất thiết theo thứ tự). Tham số int có đối số mặc định là 5, long là 65536, double không có đối số mặc định.*
  - *void computer(double, int =5,long=65536); //Hàm nguyên mẫu*
  - *void computer(double d, int num=5, long lnum=65536) //Tiêu đề hàm*
- *5.21. Viết hàm nguyên mẫu và tiêu đề cho hàm calculate. Hàm có 3 tham số: int, double, long (không nhất thiết theo thứ tự). Chỉ có kiểu int có đối số mặc định là 47.*
  - *void calculate(double, long,int=47); //Hàm nguyên mẫu*
  - *void calculate(double x, long y,int z=47); //Tiêu đề hàm*
  - *calculate(4,5,6); hoặc calculate(4,5); hoặc calculate(4);//SAI*
- *5.22. Đầu ra của đoạn chương trình sau là gì?*

Kiểm tra


```
#include <iostream>
using namespace std;
void func1(int &, int &);
void func2(int &, int &, int &);
void func3(int, int, int);
int main()
{
    int x = 0, y = 0, z = 0;
    cout << x << " " << y << " " << z << endl;
    func1(x, y);
    cout << x << " " << y << " " << z << endl;
    func2(x, y, z);
    cout << x << " " << y << " " << z << endl;
    func3(x, y, z);
    cout << x << " " << y << " " << z << endl;
    return 0;
}
void func1(int &a, int &b)
{
    cout << "Enter two numbers: ";
    cin >> a >> b;
}
void func2(int &a, int &b, int &c)
{
    b++;
    c--;
    a = b + c;
}
void func3(int a, int b, int c)
{
    a = b - c;
}
```

Nếu người dùng nhập vào 12 và 14

Đầu ra của đoạn chương trình sau là gì?

```
#include <iostream>
using namespace std;
void test(int = 2, int = 4, int = 6);
int main()
{
    test();
    test(6);
    test(3, 9);
    test(1, 5, 7);
    return 0;
}
void test (int first, int second, int third)
{
    first += 3;
    second += 6;
    third += 9;
    cout << first << " " << second << " " << third <<endl;
}
```

Quá tải hàm

- Quá tải hàm là có các tên hàm giống nhau nhưng khác nhau về tham số.
- Có thể sử dụng để tạo các hàm thực hiện nhiệm vụ giống nhau nhưng khác nhau về kiểu tham số hoặc số lượng tham số.
- Trình biên dịch sẽ xác định hàm thích hợp bằng danh sách đối số và tham số.
- Ví dụ: định nghĩa
  - void getDimensions(int); // 1
  - void getDimensions(int, int); // 2
  - void getDimensions(int, double); // 3
  - void getDimensions(double, double);// 4
- Ví dụ: gọi hàm
  - int length, width;
  - double base, height;
  - getDimensions(length); // 1
  - getDimensions(length, width); // 2
  - getDimensions(length, height); // 3
  - getDimensions(height, base); // 4

Quá tải hàm

Program 6-27

```
1  // This program uses overloaded functions.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  // Function prototypes
7  int square(int);
8  double square(double);
9
10 int main()
11 {
12    int userInt;
13    double userFloat;
14
15    // Get an int and a double.
16    cout << fixed << showpoint << setprecision(2);
17    cout << "Enter an integer and a floating-point value: ";
18    cin >> userInt >> userFloat;
19
20    // Display their squares.
21    cout << "Here are their squares: ";
22    cout << square(userInt) << " and " << square(userFloat);
23    return 0;
24 }
```

Quá tải hàm có kiểu tham số khác nhau.

Truyền giá trị kiểu double

Truyền giá trị kiểu int

```
26 //*************************************************************
27 // Definition of overloaded function square.                  *
28 // This function uses an int parameter, number. It returns the *
29 // square of number as an int.                                 *
30 //*************************************************************
31
32 int square(int number)
33 {
34    return number * number;
35 }
36
37 //*************************************************************
38 // Definition of overloaded function square.                  *
39 // This function uses a double parameter, number. It returns   *
40 // the square of number as a double.                           *
41 //*************************************************************
42
43 double square(double number)
44 {
45    return number * number;
46 }
```

Program Output with Example Input Shown in Bold

```
Enter an integer and a floating-point value: 12 4.2 [Enter]
Here are their squares: 144 and 17.64
```

Hàm Exit()

- Kết thúc thực hiện chương trình
- Có thể gọi bất cứ từ hàm nào
- Có thể truyền 1 giá trị kiểu int để hệ điều hành xác định tình trạng kết thúc chương trình.
  - VD: Exit(1), Exit(0);
- Thường được dùng để kết thúc chương trình
- Yêu cầu có `#include<cstdlib>`
- Các hằng số được truyền để xác định thành công hay thất bại
  - exit(EXIT_SUCCESS);
  - exit(EXIT_FAILURE);

HỎI ĐÁP

Bài tập

- **Bài tập chương 5:**
  - Chapter5_Programming Challenges.pdf

ĐẠI NAM
UNIVERSITY

Trân trọng cảm ơn!

ĐẬU HẢI PHONG
Giảng viên
dauhaiphong@dainam.edu.vn
0912441435