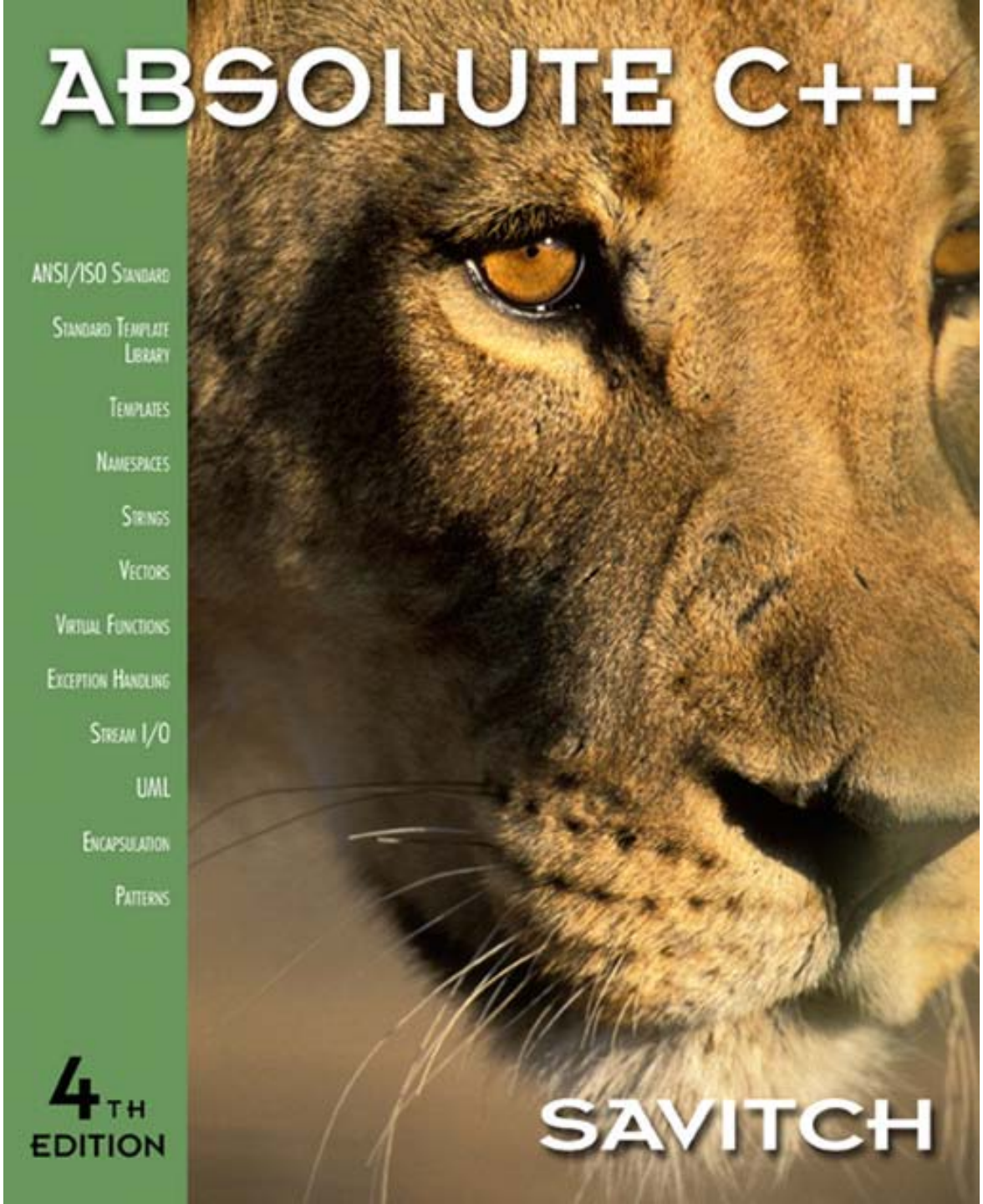
Bài 4: Tham số của hàm và Nạp chồng hàm

Giảng viên: Hoàng Thị Điệp
Khoa Công nghệ Thông tin – ĐH Công Nghệ



Chapter 4

Parameters and Overloading

PEARSON Addison Wesley

Mục tiêu bài học

- Tham số
  - Truyền giá trị
  - Truyền tham chiếu
  - Phối hợp 2 kiểu trong danh sách tham số
- Nạp chồng hàm và Đối số mặc định
  - Ví dụ, Quy tắc
- Chạy thử và gỡ lỗi cho hàm
  - Macro assert
  - Stub và Driver

Tham số

- Hai phương thức truyền tham số cho hàm
- Truyền giá trị
  - “bản sao" của đối số thực sự được truyền vào
- Truyền tham chiếu
  - “địa chỉ“ của đối số thực sự được truyền vào

Truyền giá trị

- Bản sao của đối số thực sự được truyền vào
- Bên trong hàm, chúng được xem như biến cục bộ
- Nếu bị biến đổi thì chỉ bản sao này chịu ảnh hưởng
  - Hàm không tác động lên đối số thực sự ở nơi gọi hàm
- Đây là cách thức mặc định
  - Được dùng trong tất cả các ví dụ ở các bài trước

Ví dụ truyền giá trị:
Display 4.1 Dùng tham số hình thức như biến cục bộ (1/3)

Display 4.1 Formal Parameter Used as a Local Variable

```
1   //Law office billing program.
2   #include <iostream>
3   using namespace std;

4   const double RATE = 150.00; //Dollars per quarter hour.

5   double fee(int hoursWorked, int minutesWorked);
6   //Returns the charges for hoursWorked hours and
7   //minutesWorked minutes of legal services.

8   int main( )
9   {
10      int hours, minutes;
11      double bill;
```

Ví dụ truyền giá trị:
Display 4.1 Dùng tham số hình thức như biến cục bộ (2/3)

```
12        cout << "Welcome to the law office of\n"
13             << "Dewey, Cheatham, and Howe.\n"
14             << "The law office with a heart.\n"
15             << "Enter the hours and minutes"
16             << " of your consultation:\n";
17        cin >> hours >> minutes;

18        bill = fee(hours, minutes);

19        cout.setf(ios::fixed);
20        cout.setf(ios::showpoint);
21        cout.precision(2);
22        cout << "For " << hours << " hours and " << minutes
23             << " minutes, your bill is $" << bill << endl;

24        return 0;
25   }
```

The value of `minutes` *is not changed by the call to* `fee`.

(continued)

Ví dụ truyền giá trị:
Display 4.1 Dùng tham số hình thức như biến cục bộ (3/3)

Display 4.1 Formal Parameter Used as a Local Variable

```
26  double fee(int hoursWorked, int minutesWorked)
27  {
28      int quarterHours;

29      minutesWorked = hoursWorked*60 + minutesWorked;
30      quarterHours = minutesWorked/15;
31      return (quarterHours*RATE);
32  }
```

`minutesWorked` *is a local variable initialized to the value of* `minutes`.

SAMPLE DIALOGUE

```
Welcome to the law office of
Dewey, Cheatham, and Howe.
The law office with a heart.
Enter the hours and minutes of your consultation:
5 46
For 5 hours and 46 minutes, your bill is $3450.00
```

Truyền giá trị: Lỗi thường gặp

- Lỗi thường gặp:
  - Lặp lại khai báo tham số trong thân hàm:

```
double fee(int hoursWorked, int minutesWorked)
{
  int quarterHours;                 // biến cục bộ
  int minutesWorked                 // KHÔNG ĐƯỢC!
}
```

  - Kết quả báo lỗi biên dịch
    - "Redefinition error…"
- Đối số giá trị được dùng như biến cục bộ trong thân hàm
  - Nhưng hàm “tự động” có được chúng

Truyền tham chiếu

- Cung cấp truy cập tới đối số thực sự
- Hàm được gọi tới có thể biến đổi dữ liệu của nơi gọi hàm!
- Ví dụ điển hình: hàm nhập dữ liệu
  - Để lấy dữ liệu cho nơi gọi
  - Dữ liệu sẽ được "gửi" cho nơi gọi
- Chỉ định truyền tham chiếu bằng cách thêm dấu và (&) vào sau kiểu dữ liệu trong danh sách tham số

Ví dụ truyền tham chiếu:
Display 4.1 Truyền tham chiếu (1/3)

Display 4.2 **Call-by-Reference Parameters**

```
1  //Program to demonstrate call-by-reference parameters.
2  #include <iostream>
3  using namespace std;

4  void getNumbers(int& input1, int& input2);
5  //Reads two integers from the keyboard.

6  void swapValues(int& variable1, int& variable2);
7  //Interchanges the values of variable1 and variable2.

8  void showResults(int output1, int output2);
9  //Shows the values of variable1 and variable2, in that order.

10 int main( )
11 {
12     int firstNum, secondNum;

13     getNumbers(firstNum, secondNum);
14     swapValues(firstNum, secondNum);
15     showResults(firstNum, secondNum);
16     return 0;
17 }
```

Ví dụ truyền tham chiếu:
Display 4.1 Truyền tham chiếu (2/3)

```
18  void getNumbers(int& input1, int& input2)
19  {
20      cout << "Enter two integers: ";
21      cin >> input1
22          >> input2;
23  }

24  void swapValues(int& variable1, int& variable2)
25  {
26      int temp;

27      temp = variable1;
28      variable1 = variable2;
29      variable2 = temp;
30  }
31
32  void showResults(int output1, int output2)
33  {
34      cout << "In reverse order the numbers are: "
35           << output1 << " " << output2 << endl;
36  }
```

Ví dụ truyền tham chiếu:
Display 4.1 Truyền tham chiếu (3/3)

Display 4.2 **Call-by-Reference Parameters**

SAMPLE DIALOGUE

```
Enter two integers: 5 6
In reverse order the numbers are: 6 5
```

Chi tiết truyền tham chiếu

- Thực sự thì cái gì được truyền vào?
- "Tham chiếu" tới đối số thực sự ở nơi gọi hàm!
  - Trỏ tới địa chỉ nhớ của đối số thực sự
  - Được gọi là "địa chỉ", là một con số duy nhất chỉ một địa điểm cụ thể trong bộ nhớ

Tham số tham chiếu hằng

- Đối số tham chiếu ẩn chứa nguy hiểm
  - Dữ liệu ở nơi gọi hàm có thể bị thay đổi
  - Thường thì đây là điều được mong đợi, nhưng đôi khi ngoài mong đợi
- Để "bảo vệ" dữ liệu và vẫn dùng truyền tham chiếu:
  - Hãy sử dụng từ khóa const
    - void sendConstRef(const int &par1,
      const int &par2);
    - Với khai báo này, hàm chỉ có thể đọc tham số
    - Thân hàm không được phép thay đổi chúng

Tham số và đối số

- Các thuật ngữ này dễ gây nhầm lẫn, thường dùng lẫn lộn
- Ý nghĩa thực sự:
  - Tham số hình thức
    - Trong khai báo hàm và định nghĩa hàm
  - Đối số
    - Dùng để "điền vào" tham số hình thức
    - Trong lời gọi hàm (danh sách đối số)
  - Truyền giá trị và truyền tham chiếu
    - Là cơ chế của quá trình lắp ghép dữ liệu vào trong hàm

Danh sách tham số phối hợp hai kiểu truyền

- Có thể phối hợp các cơ chế truyền tham số
- Danh sách tham số có thể có cả tham số tham chiếu và tham số giá trị
- Trong danh sách này, thứ tự đối số rất quan trọng:
  void mixedCall(int & par1, int par2, double & par3);
  - Lời gọi hàm:
    mixedCall(arg1, arg2, arg3);
    - arg1 phải có kiểu int, được truyền tham chiếu
    - arg2 phải có kiểu int, được truyền giá trị
    - arg3 phải có kiểu double, được truyền tham chiếu

Lựa chọn tên tham số hình thức

- Giống quy tắc đặt tên định danh:
  - Tên phải có nghĩa!
- Hàm là một "đơn vị khép kín"
  - Được thiết kế riêng biệt với phần còn lại của chương trình
  - Giao cho các nhóm lập trình viên khác nhau
  - Tất cả cần "hiểu" đúng cách sử dụng hàm
  - Có thể chấp nhận tên tham số hình thức trùng với tên đối số
- Lựa chọn tên hàm cũng dùng các quy tắc như trên

Nạp chồng hàm

- Các hàm có trùng tên
- Danh sách tham số khác nhau
- Hai định nghĩa riêng biệt
- "Chữ kí" của hàm
  - Tên hàm và danh sách tham số
  - Phải là "duy nhất" cho mỗi định nghĩa hàm
- Cho phép cùng một công việc thực hiện trên những dữ liệu khác nhau

Ví dụ nạp chồng: hàm **average()**

- Hàm tính trung bình cộng của 2 số:

```
double average(double n1, double n2)
{
        return ((n1 + n2) / 2.0);
}
```

- Hàm tính trung bình cộng của 3 số:

```
double average(double n1, double n2, double n3)
{
        return ((n1 + n2 + n3) / 3.0);
}
```

- Cùng tên nhưng là 2 hàm riêng biệt

Nạp chồng hàm **average()** <tiếp>

- Hàm nào sẽ được gọi?
- Tùy vào bản thân lời gọi:
  - avg = average(5.2, 6.7);
    - Gọi tới hàm average() có hai tham số
  - avg = average(6.5, 8.5, 4.2);
    - Gọi tới hàm average() có ba tham số
- Trình biên dịch phân tích lời gọi dựa trên chữ ký của lời gọi
  - "Ghép đôi" lời gọi với hàm phù hợp
  - Các hàm này là tách biệt

Các lỗi thường gặp khi nạp chồng hàm

- Chỉ nạp chồng những hàm cùng công việc
  - Một hàm mpg() nên luôn thực hiện cùng công việc, trong tất cả các phiên bản nạp chồng
  - Nếu không thì rất khó đoán ý nghĩa kết quả
- Phân tích lời gọi hàm C++:
  - Bước 1: tìm chữ kí chính xác
  - Bước 2: tìm chữ kí "có khả năng tương thích"

Phân tích lời gọi nạp chồng

- Bước 1: Ghép đôi chính xác
  - Tìm chữ kí chính xác
    - là nơi ta không cần chuyển đổi kiểu đối số
- Bước 2: Ghép đôi "tương thích"
  - Tìm chữ kí "tương thích" (Có thể cần tới chuyển đổi kiểu tự động)
    - Phương án 1: "nâng cấp" kiểu (ví dụ int→double)
      - Không mất dữ liệu
    - Phương án 2: "hạ cấp" kiểu (ví dụ double→int)
      - Có thể mất dữ liệu

Ví dụ phân tích lời gọi nạp chồng

- Cho các hàm sau đây:
  - 1. void f(int n, double m);
    2. void f(double n, int m);
    3. void f(int n, int m);
  - Lời gọi:
    f(98, 99); → gọi #3
    f(5.3, 4); → gọi #2
    f(4.3, 5.2); → gọi ???
- Cần tránh việc nạp chồng dễ gây nhầm lẫn này

Tự động chuyển đổi kiểu và nạp chồng

- Các tham số hình thức dạng số thường được đặt kiểu double
- Dùng được với bất cứ kiểu dữ liệu số nào
  - Các dữ liệu "dưới cấp" sẽ được "nâng cấp"
    - int → double
    - float → double
    - char → double *Sau này sẽ bàn thêm!
- Tránh nạp chồng những kiểu dữ liệu số khác nhau

Ví dụ tự động chuyển đổi kiểu và nạp chồng

- double mpg(double miles, double gallons)
  {
  return (miles/gallons);
  }
- Ví dụ lời gọi hàm:
  - mpgComputed = mpg(5, 20);
    - Tự động chuyển 5 & 20 thành double rồi truyền vào
  - mpgComputed = mpg(5.8, 20.2);
    - Không cần chuyển đổi kiểu
  - mpgComputed = mpg(5, 2.4);
    - Tự động chuyển 5 thành 5.0 rồi truyền vào

Đối số mặc định

- Cho phép bỏ qua một số đối số
- Chỉ định trong khai báo/nguyên mẫu hàm
  - void showVolume(int length,
    int width = 1,
    int height = 1);
    - Hai đối số sau được đặt giá trị mặc định
  - Những lời gọi hợp lệ:
    - showVolume(2, 4, 6); //Truyền cả 3 đối số vào
    - showVolume(3, 5); //height nhận giá trị mặc định là 1
    - showVolume(7); //width & height mặc định là 1

Ví dụ đối số mặc định:
Display 4.1 Đối số mặc định (1/2)

Display 4.8 Default Arguments

```
1
2   #include <iostream>
3   using namespace std;

4   void showVolume(int length, int width = 1, int height = 1);
5   //Returns the volume of a box.
6   //If no height is given, the height is assumed to be 1.
7   //If neither height nor width is given, both are assumed to be 1.

8   int main( )
9   {
10      showVolume(4, 6, 2);
11      showVolume(4, 6);
12      showVolume(4);

13      return 0;
14  }

15  void showVolume(int length, int width, int height)
```

Default arguments

A default argument should not be given a second time.

Ví dụ đối số mặc định:
Display 4.1 Đối số mặc định (2/2)

```
16  {
17      cout << "Volume of a box with \n"
18           << "Length = " << length << ", Width = " << width << endl
19           << "and Height = " << height
20           << " is " << length*width*height << endl;
21  }
```

SAMPLE DIALOGUE

```
Volume of a box with
Length = 4, Width = 6
and Height = 2 is 48
Volume of a box with
Length = 4, Width = 6
and Height = 1 is 24
Volume of a box with
Length = 4, Width = 1
and Height = 1 is 4
```

Chạy thử và gỡ lỗi cho hàm

- Rất nhiều kĩ thuật:
  - Dùng nhiều lệnh cout
    - Trong lời gọi và định nghĩa
    - Dùng để "lần vết" thực thi chương trình
  - Trình gỡ lỗi của IDE
    - Tùy thuộc môi trường
  - Dùng macro assert
    - Kết thúc chương trình sớm hơn cần thiết
  - Stubs và drivers
    - Phát triển tăng dần

Macro assert

- Một khẳng định trong assert: là một biểu thức true hay false
- Dùng khi viết tài liệu và kiểm tra tính đúng đắn của hàm
  - Điều kiện trước và sau
    - Thường dùng assert: kiểm tra tính hợp lệ của chúng
  - Cú pháp:
    assert(<điều_kiện_assert>);
    - Không có giá trị trả về
    - Tính giá trị của biểu thức điều_kiện_assert
    - Kết thúc chương trình nếu false, tiếp tục nếu true
- Định nghĩa trong thư viện <cassert>
  - Macro được dùng tương tự như hàm

Ví dụ macro assert

- Cho khai báo hàm sau:

```
void computeCoin(    int coinValue,
                     int& number,
                     int& amountLeft);
//Điều kiện trước:   0 < coinValue < 100
                     0 <= amountLeft <100
//Điều kiện sau: number set to max. number
                                of coins
```

- Kiểm tra điều kiện trước:
  - assert ((0 < currentCoin) && (currentCoin < 100)
    && (0 <= currentAmountLeft) && (currentAmountLeft < 100));
  - Nếu không thỏa mãn điều kiện trước → điều kiện bằng false → chương trình kết thúc!

Ví dụ macro assert <tiếp>

- Hữu ích khi gỡ lỗi
- Kết thúc chương trình do đó ta có thể phán đoán lỗi

Bật/tắt assert

- Chỉ thị tiền xử lý cho ta một cách để bật/tắt assert
- #define NDEBUG
  #include <cassert>
- Bổ sung dòng "#define" trước dòng “#include”
  - TẮT tất cả assert trong chương trình
- Loại bỏ dòng "#define" (hoặc đặt dấu chú thích dòng đó)
  - BẬT các assert trở lại

Stub và Driver

- Là các kĩ thuật để tách rời các đơn vị cần biên dịch
  - Mỗi hàm được thiết kế, cài đặt, chạy thử riêng rẽ
  - Đảm bảo tính hợp lệ của mỗi đơn vị
  - Chia để trị
    - Chuyển một tác vụ lớn → nhiều tác vụ nhỏ hơn, dễ quản lý
- Nhưng làm sao để chạy thử chúng độc lập với nhau?
  - Sử dụng các chương trình driver

Ví dụ chương trình driver:
Display 4.9 Chương trình driver (1/3)

Display 4.9 Driver Program

```
1
2   //Driver program for the function unitPrice.
3   #include <iostream>
4   using namespace std;

5   double unitPrice(int diameter, double price);
6   //Returns the price per square inch of a pizza.
7   //Precondition: The diameter parameter is the diameter of the pizza
8   //in inches. The price parameter is the price of the pizza.

9   int main( )
10  {
11      double diameter, price;
12      char ans;

13      do
14      {
15          cout << "Enter diameter and price:\n";
16          cin >> diameter >> price;
```

Ví dụ chương trình driver:
Display 4.9 Chương trình driver (2/3)

```
17          cout << "unit Price is $"
18               << unitPrice(diameter, price) << endl;

19          cout << "Test again? (y/n)";
20          cin >> ans;
21          cout << endl;
22      } while (ans == 'y' || ans == 'Y');

23      return 0;
24  }
25
26  double unitPrice(int diameter, double price)
27  {
28      const double PI = 3.14159;
29      double radius, area;

30      radius = diameter/static_cast<double>(2);
31      area = PI * radius * radius;
32      return (price/area);
33  }
```

(continued)

Ví dụ chương trình driver:
Display 4.9 Chương trình driver (3/3)

Display 4.9 Driver Program

SAMPLE DIALOGUE

```
Enter diameter and price:
13 14.75
Unit price is: $0.111126
Test again? (y/n): y

Enter diameter and price:
2 3.15
Unit price is: $1.00268
Test again? (y/n): n
```

Stub

- Trong mô hình phát triển tăng dần
- Trước tiên, viết các hàm mô phỏng "bức tranh toàn cảnh"
  - Các hàm ở mức thấp thì viết sau
  - Dùng các hàm thân trống để chạy thử rồi cài đặt dần
  - Ví dụ:

```
double unitPrice(int diameter, double price)
{
    return (9.99);   // không chính xác, nhưng chú ý
                     // nó chỉ là một giá trị "tạm thời"
}
```

  - Lời gọi tới hàm vẫn thực hiện được

Quy tắc chạy thử cơ bản

- Nhằm viết chương trình đúng đắn
- Giảm thiểu lỗi, "bugs"
- Đảm bảo tính hợp lệ của dữ liệu
  - Chạy thử mọi hàm trong chương trình có tất cả những hàm khác đã chạy thử và gỡ lỗi suôn sẻ
  - Tránh hiện tượng lỗi này kéo theo lỗi khác và kết quả xung đột

Tóm tắt 1

- Tham số hình thức là chỗ đặt trước, sẽ được điền bằng đối số thực sự trong lời gọi hàm
- Tham số được truyền bằng giá trị là những bản sao trong thân hàm được gọi
  - Hàm không thao tác trên đối số thực sự
- Truyền bằng tham chiếu cung cấp cho hàm địa chỉ nhớ của đối số thực sự
  - Hàm thao tác trên (do đó có thể biến đổi) đối số thực sự
  - Đối số phải là một biến, không thể là hằng số

Tóm tắt 2

- Bạn có thể định nghĩa nhiều hàm trùng tên: việc này gọi là nạp chồng hàm
- Đối số mặc định cho phép lời gọi hàm được "bỏ qua" một số hoặc tất cả đối số trong danh sách
  - Nếu lời gọi không cung cấp đối số → sử dụng giá trị mặc định
- Macro assert kích hoạt kết thúc chương trình nếu biểu thức điều kiện của assert không thỏa mãn
- Các hàm nên được chạy thử độc lập
  - Như các đơn vị biên dịch riêng rẽ, cùng với driver của chúng

Chuẩn bị bài tới

- Đọc chương 5 giáo trình: Mảng