

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**MÔN**  
**NGÔN NGỮ LẬP TRÌNH C++**

**CHƯƠNG 4 – MẢNG**

Giảng viên	:	<b>ThS. PHAN NGHĨA HIỆP</b>
Khoa	:	<b>CÔNG NGHỆ THÔNG TIN 2</b>
Bộ môn	:	<b>AN TOÀN THÔNG TIN</b>
Điện thoại/ Email	:	<b>hieppn@ptithcm.edu.vn</b>

# Chương 4 – Mảng

## Đề mục

- 4.1 Giới thiệu
- 4.2 Mảng
- 4.3 Khai báo mảng
- 4.4 Ví dụ về sử dụng mảng
- 4.5 Truyền tham số cho hàm
- 4.6 Sắp xếp mảng
- 4.7 Ví dụ: Dùng mảng tính Mean, Median và Mode
- 4.8 Tìm kiếm trên mảng: Tìm kiếm Tuyến tính và tìm kiếm Nhị phân
- 4.9 Mảng nhiều chiều

## 4.1 Giới thiệu

- Mảng (array)
  - Cấu trúc của những phần tử dữ liệu có liên quan
  - Thực thể tĩnh (giữ nguyên kích thước trong suốt chương trình)
- Một vài loại mảng
  - mảng dựa vào con trỏ (Pointer-based arrays) (C-like)
  - mảng là đối tượng (Arrays as objects) (C++)

## 4.2 Mảng

- Mảng
  - Tập hợp các vùng nhớ liên tiếp
  - Cùng tên, cùng kiểu (**int**, **char**, ...)
- Truy nhập đến 1 phần tử
  - Chỉ ra tên mảng và vị trí - position (chỉ số - index)
  - Cú pháp: **tên\_mảng[ chỉ\_số ]**
  - Phần tử đầu tiên ở vị trí 0
- Mảng c có n phần tử  
**c[ 0 ], c[ 1 ] ... c[ n - 1 ]**
  - Phần tử thứ N ở vị trí thứ N-1

## 4.2 Mảng

- Phần tử của mảng cũng như các biến khác
  - Gán giá trị và in mảng số nguyên **c**  
**c[ 0 ] = 3;**  
**cout << c[ 0 ];**
- Có thể sử dụng các phép toán trong cặp ngoặc vuông  
**c[ 5 - 2 ]** cũng giống **c[3]**

Tên mảng  
(Lưu ý rằng mọi phần tử  
của mảng này đều có cùng  
tên, **c**)

<b>c[0]</b>	<b>-45</b>
<b>c[1]</b>	<b>6</b>
<b>c[2]</b>	<b>0</b>
<b>c[3]</b>	<b>72</b>
<b>c[4]</b>	<b>1543</b>
<b>c[5]</b>	<b>-89</b>
<b>c[6]</b>	<b>0</b>
<b>c[7]</b>	<b>62</b>
<b>c[8]</b>	<b>-3</b>
<b>c[9]</b>	<b>1</b>
<b>c[10]</b>	<b>6453</b>
<b>c[11]</b>	<b>78</b>

Chỉ số của phần tử  
trong mảng **c**

## 4.3 Khai báo mảng

- Khi khai báo mảng, chỉ rõ
  - Tên
  - Kiểu của mảng
    - Bất cứ kiểu dữ liệu nào
  - Số phần tử
  - *type arrayName [ arraySize ] ;*
    - `int c[ 10 ]; // mảng của 10 số nguyên`
    - `float d[ 3284 ]; // mảng của 3284 số thực`
- Khai báo nhiều mảng cùng kiểu
  - Sử dụng dấu phẩy như với các biến bình thường
    - `int b[ 100 ], x[ 27 ];`

## 4.4 Ví dụ về sử dụng mảng

- Khởi tạo mảng

- Dùng vòng lặp khởi tạo từng phần tử

- Khởi tạo cả danh sách

- Chỉ rõ từng phần tử khi khai báo mảng

- ```
int n[ 5 ] = { 1, 2, 3, 4, 5 };
```

- Nếu trong danh sách không có đủ số giá trị khởi tạo, các phần tử ở bên phải nhất sẽ nhận giá trị 0

- Nếu danh sách thừa sẽ gây lỗi cú pháp

- Khởi tạo giá trị bằng 0 cho tất cả các phần tử

- ```
int n[ 5 ] = { 0 };
```

- Nếu không khai báo kích thước mảng, kích thước của danh sách các giá trị khởi tạo sẽ quyết định kích thước mảng

- ```
int n[] = { 1, 2, 3, 4, 5 };
```

- Có 5 giá trị khởi tạo, do đó mảng có 5 phần tử

- Nếu không khai báo kích thước mảng thì phải khởi tạo khi khai báo



```
1 // Fig. 4.3: fig04_03.cpp
2 // Initializing an array.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     int n[ 10 ]; // n is an array of 10 integers
15
16     // initialize elements of array n to 0
17     for ( int i = 0; i < 10; i++ )
18         n[ i ] = 0; // set element at location i to 0
19
20     cout << "Element" << setw( 13 ) << "Value" << endl;
21
22     // output contents of array n in tabular format
23     for ( int j = 0; j < 10; j++ )
24         cout << setw( 7 ) << j << setw( 13 ) << n[ j ] << endl;
25 }
```

Khai báo mảng 10 phần tử số nguyên.

Khởi tạo mảng bằng vòng lặp **for**.  
Chú ý rằng mảng gồm các phần tử  
từ **n[0]** đến **n[9]**.

```
26     return 0;  // indicates successful termination
27
28 } // end main
```

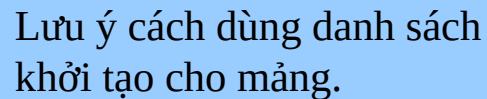
**fig04\_03.cpp**  
(2 of 2)

| Element | Value |
|---------|-------|
| 0       | 0     |
| 1       | 0     |
| 2       | 0     |
| 3       | 0     |
| 4       | 0     |
| 5       | 0     |
| 6       | 0     |
| 7       | 0     |
| 8       | 0     |
| 9       | 0     |

**fig04\_03.cpp**  
output (1 of 1)

```
1 // Fig. 4.4: fig04_04.cpp
2 // Initializing an array with a declaration.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     // use initializer list to initialize array n
15     int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
16
17     cout << "Element" << setw( 13 ) << "Value" << endl;
18
19     // output contents of array n in tabular format
20     for ( int i = 0; i < 10; i++ )
21         cout << setw( 7 ) << i << setw( 13 ) << n[ i ] << endl;
22
23     return 0; // indicates successful termination
24
25 }
```

Lưu ý cách dùng danh sách  
khởi tạo cho mảng.



| Element | Value |
|---------|-------|
| 0       | 32    |
| 1       | 27    |
| 2       | 64    |
| 3       | 18    |
| 4       | 95    |
| 5       | 14    |
| 6       | 90    |
| 7       | 70    |
| 8       | 60    |
| 9       | 37    |

**fig04\_04.cpp**  
**output (1 of 1)**

## 4.4 Ví dụ về sử dụng mảng

- Kích thước của mảng
  - Có thể được xác định bằng hằng số (**const**)
    - **const int size = 20;**
  - Hằng số không thể thay đổi
  - Hằng phải được khởi tạo khi khai báo
  - Còn được gọi là “named constant” (giá trị được đặt tên) hoặc “read-only variable” (biến chỉ đọc)

```
1 // Fig. 4.5: fig04_05.cpp
2 // Initialize array s to the even integers from 2 to 20.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     // constant variable can be used
15     const int arraySize = 10;
16
17     int s[ arraySize ]; // array s has 10 elements
18
19     for ( int i = 0; i < arraySize; i++ ) // s
20         s[ i ] = 2 + 2 * i;
21
22     cout << "Element" << setw( 13 ) << "Value"
23
```

Chú ý từ khoá **const**. Chỉ có các biến **const** được dùng để khai báo kích thước mảng.

Chương trình dễ thay đổi hơn khi ta dùng hằng (**const**) cho kích thước của mảng.  
Ta có thể thay đổi **arraySize**, và tất cả các vòng lặp vẫn hoạt động bình thường (nếu không, ta phải sửa mọi vòng lặp trong chương trình).

```
24 // output contents of array s in tabular format
25 for ( int j = 0; j < arraySize; j++ )
26     cout << setw( 7 ) << j << setw( 13 ) << s[ j ] << endl;
27
28 return 0; // indicates successful termination
29
30 } // end main
```

fig04\_05.cpp  
(2 of 2)

fig04\_05.cpp  
output (1 of 1)

| Element | Value |
|---------|-------|
| 0       | 2     |
| 1       | 4     |
| 2       | 6     |
| 3       | 8     |
| 4       | 10    |
| 5       | 12    |
| 6       | 14    |
| 7       | 16    |
| 8       | 18    |
| 9       | 20    |

```
1 // Fig. 4.6: fig04_06.cpp
2 // Using a properly initialized constant variable.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 int main()
9 {
10     const int x = 7; // initialized constant variable
11
12     cout << "The value of constant variable x is: "
13         << x << endl;
14
15     return 0; // indicates successful termination
16
17 } // end main
```

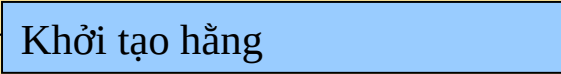


fig04\_06.cpp  
(1 of 1)

fig04\_06.cpp  
output (1 of 1)

The value of constant variable x is: 7



```

1  // Fig. 4.7: fig04_07.cpp
2  // A const object must be initialized.
3
4  int main()
5  {
6      const int x; // Error: x must be initialized
7
8      x = 7; // Error: cannot modify a const variable
9
10     return 0; // indicates successful termination
11
12 } // end main

```

Lỗi cú pháp do không khởi tạo hằng.  
Sửa giá trị của hằng cũng là một lỗi.

fig04\_07.cpp  
(1 of 1)

fig04\_07.cpp  
output (1 of 1)

```

d:\cpphttp4_examples\ch04\Fig04_07.cpp(6) : error C2734: 'x' :
const object must be initialized if not extern
d:\cpphttp4_examples\ch04\Fig04_07.cpp(8) : error C2166:
l-value specifies const object

```

```

1  // Fig. 4.8: fig04_08.cpp
2  // Compute the sum of the elements of the array.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main()
9  {
10     const int arraySize = 10;
11
12     int a[ arraySize ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
13
14     int total = 0;
15
16     // sum contents of array a
17     for ( int i = 0; i < arraySize; i++ )
18         total += a[ i ];
19
20     cout << "Total of array element values is " << total << endl;
21
22     return 0; // indicates successful termination
23
24 } // end main

```

fig04\_08.cpp  
of 1)

fig04\_08.cpp  
put (1 of 1)

Total of array element values is 55

```

1  // Fig. 4.9: fig04_09.cpp
2  // Histogram printing program.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     const int arraySize = 10;
15     int n[ arraySize ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
16
17     cout << "Element" << setw( 13 ) << "Value"
18         << setw( 17 ) << "Histogram" << endl;
19

```

| Element | Value | Histogram |
|---------|-------|-----------|
| 0       | 19    | *****     |
| 1       | 3     | ***       |
| 2       | 15    | *****     |
| 3       | 7     | *****     |
| 4       | 11    | *****     |
| 5       | 9     | *****     |
| 6       | 13    | *****     |
| 7       | 5     | *****     |
| 8       | 17    | *****     |
| 9       | 1     | *         |

```

20 // for each element of array n, output a bar in histogram
21 for ( int i = 0; i < arraySize; i++ ) {
22     cout << setw( 7 ) << i << setw( 13 )
23         << n[ i ] << setw( 9 );
24
25     for ( int j = 0; j < n[ i ]; j++ ) // print one bar
26         cout << '*';
27
28     cout << endl; // start next line of output
29
30 } // end outer for structure
31
32 return 0; // indicates successful termination
33
34 } // end main

```

In số dấu sao (\*) tương ứng với giá trị của phần tử **n[i]**.

fig04\_09.cpp  
output (1 of 1)

| Element | Value | Histogram |
|---------|-------|-----------|
| 0       | 19    | *****     |
| 1       | 3     | ***       |
| 2       | 15    | *****     |
| 3       | 7     | *****     |
| 4       | 11    | *****     |
| 5       | 9     | *****     |
| 6       | 13    | *****     |
| 7       | 5     | *****     |
| 8       | 17    | *****     |
| 9       | 1     | *         |

fig04\_10.cpp  
(1 of 2)

```
1 // Fig. 4.10: fig04_10.cpp
2 // Roll a six-sided die 6000 times.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 #include <cstdlib>
13 #include <ctime>
14
15 int main()
16 {
17     const int arraySize = 7;
18     int frequency[ arraySize ] = { 0 };
19
20     srand( time( 0 ) ); // seed random-number generator
21
22     // roll die 6000 times
23     for ( int roll = 1; roll <= 6000; roll++ )
24         ++frequency[ 1 + rand() % 6 ]; // replaces 20-line switch
25   // of Fig. 3.8
```

Viết lại một chương trình cũ. Một mảng được sử dụng thay cho 6 biến thường, và các phần tử dễ dàng cập nhật hơn (không cần sử dụng **switch**).

Dòng lệnh này tạo ra một số trong khoảng 1 đến 6 và tăng phần tử **frequency[]** có chỉ số đó.

```

26
27     cout << "Face" << setw( 13 ) << "Frequency" << endl;
28
29     // output frequency elements 1-6 in tabular format
30     for ( int face = 1; face < arraySize; face++ )
31         cout << setw( 4 ) << face
32             << setw( 13 ) << frequency[ face ] << endl;
33
34     return 0; // indicates successful termination
35
36 } // end main

```

fig04\_10.cpp  
(2 of 2)

fig04\_10.cpp  
output (1 of 1)

| Face | Frequency |
|------|-----------|
| 1    | 1003      |
| 2    | 1004      |
| 3    | 999       |
| 4    | 980       |
| 5    | 1013      |
| 6    | 1001      |

```

1  // Fig. 4.11: fig04_11.cpp    ***modified***
2  // Student mark statistic program.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     // define array sizes
15     const int markSize = 40;    // size of array of marks
16     const int frequencySize = 11; // size of array frequency
17
18     // place student marks in array of marks
19     int marks[ markSize ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8,
20                             10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7,
21                             5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
22
23     // initialize frequency counters to 0
24     int frequency[ frequencySize ] = { 0 };
25

```

g04\_11.cpp  
of 2)

```
26 // for each student's mark, select value of an element of array
27 // responses and use that value as subscript in array
28 // frequency to determine element to increment
29 for ( int student = 0; student < markSize; student++ )
30     ++frequency[ marks[student] ];
31
32 // display results
33 cout << "Rating" << setw( 17 ) << "Frequency" << endl;
34
35 // output frequencies in tabular format
36 for ( int rating = 1; rating < frequencySize; rating++ )
37     cout << setw( 6 ) << rating
38         << setw( 17 ) << frequency[ rating ] << endl;
39
40 return 0; // indicates successful termination
41
42 } // end main
```

**marks[student]** là điểm (từ 1 đến 10).  
Giá trị này quyết định chỉ số của phần tử  
**frequency[]** cần tăng.

| Rating | Frequency |
|--------|-----------|
| 1      | 2         |
| 2      | 2         |
| 3      | 2         |
| 4      | 2         |
| 5      | 5         |
| 6      | 11        |
| 7      | 5         |
| 8      | 7         |
| 9      | 1         |
| 10     | 3         |



## 4.4 Ví dụ về sử dụng mảng

- Xâu - string (xem thêm ở chương 5)
  - Mảng của các ký tự
  - Mọi xâu đều kết thúc với ký tự **null** ('\\0')
  - Ví dụ
    - **char string1[] = "hello";**
      - Ký tự **null** tự động được thêm vào, xâu có 6 phần tử
    - **char string1[] = { 'h', 'e', 'l', 'l', 'o', '\\0' };**
  - Chỉ số cũng giống như đối với mảng
    - string1[ 0 ]** bằng **'h'**
    - string1[ 2 ]** bằng **'l'**

## 4.4 Ví dụ về sử dụng mảng

- Nhập từ bàn phím bằng **cin**  

```
char string2[ 10 ];  
cin >> string2;
```

  - Ghi dữ liệu vào của người dùng vào chuỗi
    - Dừng lại ở ký tự trắng đầu tiên (tab, newline, blank...)
    - Thêm vào ký tự **null**
  - Nếu nhập quá nhiều, dữ liệu sẽ tràn mảng
    - Ta cần phải tránh điều này (mục 5.12 sẽ giải thích phương pháp)
- In chuỗi
  - **cout << string2 << endl;**
    - Không sử dụng được với các mảng có kiểu dữ liệu khác
  - In các ký tự cho đến khi gặp **null**

04\_12.cpp  
f 2)

```
1 // Fig. 4_12: fig04_12.cpp
2 // Treating character arrays as strings.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 int main()
10 {
11     char string1[ 20 ],           // reserves 20 characters
12     char string2[] = "string literal"; // reserves 15 characters
13
14     // read string from user into array string2
15     cout << "Enter the string \"hello there\": ";
16     cin >> string1; // reads "hello" [space terminates input]
17
18     // output strings
19     cout << "string1 is: " << string1
20          << "\nstring2 is: " << string2;
21
22     cout << "\nstring1 with spaces between characters is:\n";
23 }
```

Hai cách khác nhau để khai báo xâu.  
**string2** được khởi tạo và kích thước được xác định tự động.

Ví dụ về đọc xâu từ bàn phím và in ra.

```

24 // output characters until null character is reached
25 for ( int i = 0; string1[ i ] != '\0'; i++ )
26     cout << string1[ i ] << ' ';
27
28 cin >> string1; // reads "there"
29 cout << "\nstring1 is: " << string1 << endl;
30
31 return 0; // indicates successful termination
32
33 } // end main

```

Có thể truy nhập chuỗi giống như đối với mảng. Vòng lặp kết thúc khi gặp ký tự **null**.

fig04\_12.cpp  
output (1 of 1)

```

Enter the string "hello there": hello there
string1 is: hello
string2 is: string literal
string1 with spaces between characters is:
h e l l o
string1 is: there

```

## 4.4 Ví dụ về sử dụng mảng

- Kiểu lưu trữ tĩnh – static storage (chương 3)
    - Nếu là **static**, các biến địa phương lưu lại giá trị giữa các lần gọi hàm
    - chỉ được nhìn thấy trong thân hàm
    - Có thể khai báo mảng địa phương là static
      - được khởi tạo về 0
- static int array[3];**
- Nếu không phải static
    - Được tạo (và huỷ) tại mỗi lần gọi hàm

```
1 // Fig. 4.13: fig04_13.cpp
2 // Static arrays are initialized to zero.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 void staticArrayInit( void );    // function prototype
9 void automaticArrayInit( void ); // function prototype
10
11 int main()
12 {
13     cout << "First call to each function:\n";
14     staticArrayInit();
15     automaticArrayInit();
16
17     cout << "\n\nSecond call to each function:\n";
18     staticArrayInit();
19     automaticArrayInit();
20     cout << endl;
21
22     return 0; // indicates successful termination
23
24 } // end main
25
```

fig04\_13.cpp  
(1 of 3)

```

26 // function to demonstrate a static local array
27 void staticArrayInit( void )
28 {
29     // initializes elements to 0 first time function is called
30     static int array1[ 3 ];
31
32     cout << "\nValues on entering staticArrayInit:\n";
33
34     // output contents of array1
35     for ( int i = 0; i < 3; i++ )
36         cout << "array1[" << i << "] = " << array1[ i ] << " ";
37
38     cout << "\nValues on exiting staticArrayInit:\n";
39
40     // modify and output contents of array1
41     for ( int j = 0; j < 3; j++ )
42         cout << "array1[" << j << "] = "
43             << ( array1[ j ] += 5 ) << " ";
44
45 } // end function staticArrayInit
46

```

Mảng static, khởi tạo về 0 tại lần gọi hàm đầu tiên.

fig04\_13.cpp  
(2 of 3)

Dữ liệu trong mảng bị thay đổi, các thay đổi được bảo toàn.

```
47 // function to demonstrate an automatic local array
48 void automaticArrayInit( void )
49 {
50     // initializes elements each time function is called
51     int array2[ 3 ] = { 1, 2, 3 };
52
53     cout << "\n\nValues on entering automaticArrayInit:\n";
54
55     // output contents of array2
56     for ( int i = 0; i < 3; i++ )
57         cout << "array2[" << i << "] = " << array2[ i ] << " ";
58
59     cout << "\nValues on exiting automaticArrayInit:\n";
60
61     // modify and output contents of array2
62     for ( int j = 0; j < 3; j++ )
63         cout << "array2[" << j << "] = "
64             << ( array2[ j ] += 5 ) << " ";
65
66 }
```

Mảng automatic, được tạo lại  
tại mỗi lần gọi hàm.

Tuy mảng bị thay đổi, nó sẽ bị  
huỷ khi hàm kết thúc và thayt  
đổi trong dữ liệu sẽ bị mất.



First call to each function:

Values on entering staticArrayInit:

array1[0] = 0   array1[1] = 0   array1[2] = 0

Values on exiting staticArrayInit:

array1[0] = 5   array1[1] = 5   array1[2] = 5

Values on entering automaticArrayInit:

array2[0] = 1   array2[1] = 2   array2[2] = 3

Values on exiting automaticArrayInit:

array2[0] = 6   array2[1] = 7   array2[2] = 8

Second call to each function:

Values on entering staticArrayInit:

array1[0] = 5   array1[1] = 5   array1[2] = 5

Values on exiting staticArrayInit:

array1[0] = 10   array1[1] = 10   array1[2] = 10

Values on entering automaticArrayInit:

array2[0] = 1   array2[1] = 2   array2[2] = 3

Values on exiting automaticArrayInit:

array2[0] = 6   array2[1] = 7   array2[2] = 8

## 4.5 Truyền tham số cho hàm

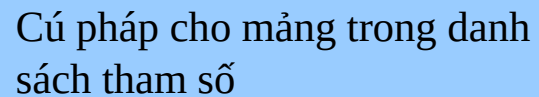
- Dùng tên mảng, bỏ cặp ngoặc vuông
  - Truyền mảng **myArray** cho hàm **myFunction**  

```
int myArray[ 24 ];  
myFunction( myArray, 24 );
```
  - Kích thước mảng thường được truyền, nhưng không nhất thiết
    - Có ích khi dùng để duyệt tất cả các phần tử
- Mảng được truyền bằng tham chiếu (passed-by-reference)
  - Hàm có thể thay đổi dữ liệu gốc của mảng
  - Tên mảng có giá trị bằng địa chỉ của phần tử đầu tiên
    - Hàm biết mảng được lưu ở đâu.
    - Hàm có thể sửa đổi dữ liệu ghi trong mảng
- Các phần tử mảng được truyền bằng giá trị (passed-by-value)
  - Như các biến thông thường
  - **square( myArray[3] );**

## 4.5 Truyền tham số cho hàm

- Các hàm dùng mảng làm đối số
  - Function prototype
    - `void modifyArray( int b[], int arraySize );`
    - `void modifyArray( int [], int );`
      - Trong prototype, tên không bắt buộc
    - cả hai hàm lấy đối số là một mảng số nguyên và 1 số nguyên
  - Không ghi cần kích thước mảng trong cặp ngoặc
    - Trình biên dịch bỏ qua
  - Nếu khai báo 1 tham số là **const**
    - đối số đó sẽ không thể bị thay đổi (chương trình dịch báo lỗi)
    - `void doNotModify( const int [] );`

```
1 // Fig. 4.14: fig04_14.cpp
2 // Passing arrays and individual array elements to functions.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 void modifyArray( int [], int ); // appears strange
13 void modifyElement( int );
14
15 int main()
16 {
17     const int arraySize = 5; // size of array a
18     int a[ arraySize ] = { 0, 1, 2, 3, 4 }; // initialize a
19
20     cout << "Effects of passing entire array by reference:"
21         << "\n\nThe values of the original array are:\n";
22
23     // output original array
24     for ( int i = 0; i < arraySize; i++ )
25         cout << setw( 3 ) << a[ i ];
```



Cú pháp cho mảng trong danh  
sách tham số

Truyền tên mảng (**a**) và kích thước cho hàm. Mảng truyền bằng tham chiếu

fig04\_14.cpp  
(2 of 3)

```
27  cout << endl;
28
29  // pass array a to modifyArray by reference
30  modifyArray( a, arraySize );
31
32  cout << "The values of the modified array are:\n";
33
34  // output modified array
35  for ( int j = 0; j < arraySize; j++ )
36      cout << setw( 3 ) << a[ j ];
37
38  // output value of a[ 3 ]
39  cout << "\n\n\n"
40      << "Effects of passing array element by value:"
41      << "\n\nThe value of a[3] is " << a[ 3 ] << '\n';
42
43  // pass array element a[ 3 ] by value
44  modifyElement( a[ 3 ] );
45
46  // output value of a[ 3 ]
47  cout << "The value of a[3] is " << a[ 3 ] << endl;
48
49  return 0; // indicates successful termination
50
51 }
```

1 phần tử mảng được truyền bằng giá trị;  
giá trị phần tử gốc không thể bị thay đổi.

Tuy đặt tên là **b**, khi được gọi, mảng chỉ đến mảng **a**, nên hàm có thể thay đổi dữ liệu của **a**.

fig04\_14.cpp  
(3 of 3)

```
52
53 // in function modifyArray, "b" points to
54 // the original array ← "a" in memory
55 void modifyArray( int b[], int sizeofArray )
56 {
57     // multiply each array element by 2
58     for ( int k = 0; k < sizeofArray; k++ )
59         b[ k ] *= 2;
60
61 } // end function modifyArray
62
63 // in function modifyElement, "e" is a local copy of
64 // array element a[ 3 ] passed from main
65 void modifyElement( int e )
66 {
67     // multiply parameter by 2
68     cout << "Value in modifyElement is "
69         << ( e *= 2 ) << endl;
70
71 } // end function modifyElement
```

Các phần tử đơn lẻ của mảng được truyền bằng giá trị, và các giá trị gốc không thể bị thay đổi.

Effects of passing entire array by reference:

The values of the original array are:

0 1 2 3 4

The values of the modified array are:

0 2 4 6 8

Effects of passing array element by value:

The value of a[3] is 6

Value in modifyElement is 12

The value of a[3] is 6

fig04\_14.cpp  
output (1 of 1)

```
1 // Fig. 4.15: fig04_15.cpp
2 // Demonstrating the const type qualifier.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 void tryToModifyArray( const int [] ); // function prototype
9
10 int main()
11 {
12     int a[] = { 10, 20, 30 };
13
14     tryToModifyArray( a );
15
16     cout << a[ 0 ] << ' ' << a[ 1 ] << ' ' << a[ 2 ] << '\n';
17
18     return 0; // indicates successful termination
19
20 } // end main
21
```

Tham số mảng được khai báo là **const**. Mảng không thể bị sửa đổi, kể cả khi nó được truyền bằng tham chiếu.



```
22 // In function tryToModifyArray, "b" cannot be used
23 // to modify the original array "a" in main.
24 void tryToModifyArray( const int b[] )
25 {
26     b[ 0 ] /= 2;    // error
27     b[ 1 ] /= 2;    // error
28     b[ 2 ] /= 2;    // error
29
30 } // end function tryToModifyArray
```

**fig04\_15.cpp**  
(2 of 2)

## 4.6 Sắp xếp mảng

- Sắp xếp dữ liệu
  - Là một ứng dụng quan trọng
  - Hầu hết mọi cơ quan/tổ chức đều phải sắp xếp dữ liệu
    - Một khối lượng khổng lồ dữ liệu cần được sắp xếp
- Xếp nổi bọt (Bubble sort)
  - Duyệt mảng vài lần
  - So sánh cặp phần tử liên tiếp
    - Nếu thứ tự tăng (hoặc bằng nhau), không thay đổi gì
    - Nếu thứ tự giảm, trao đổi hai phần tử
  - Lặp lại các bước trên cho mọi phần tử

## 4.6 Sắp xếp mảng

- Ví dụ:
  - Đi từ trái sang phải, và trao các phần tử khi cần thiết
    - Một lần duyệt cho mỗi phần tử
  - Dãy gốc: 3 4 2 7 6
  - Lần duyệt 1: 3 2 4 6 7 (trao đổi phần tử)
  - Lần duyệt 2: 2 3 4 6 7
  - Lần duyệt 3: 2 3 4 6 7 (không cần thay đổi)
  - Lần duyệt 4: 2 3 4 6 7
  - Lần duyệt 5: 2 3 4 6 7
  - Phần tử nhỏ “nổi” lên trên (như số 2 trong ví dụ)

## 4.6 Sắp xếp mảng

- Trao đổi các biến

```
int x = 3, y = 4;  
y = x;  
x = y;
```

- Cái gì xảy ra?
  - Cả x và y đều là 3!
  - Cần có biến tạm

- Giải pháp

```
int x = 3, y = 4, temp = 0;  
temp = x; // temp là 3  
x = y;    // x là 4  
y = temp; // y là 3
```

```
1 // Fig. 4.16: fig04_16.cpp
2 // This program sorts an array's values into ascending order.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14     const int arraySize = 10; // size of array a
15     int a[ arraySize ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
16     int hold; // temporary location used to swap array elements
17
18     cout << "Data items in original order\n";
19
20     // output original array
21     for ( int i = 0; i < arraySize; i++ )
22         cout << setw( 4 ) << a[ i ];
23 }
```

Duyệt 1 lần cho mỗi phần tử của mảng.

fig04\_16.cpp  
(2 of 3)

```
24 // bubble sort
25 // loop to control number of passes
26 for ( int pass = 0; pass < arraySize - 1; pass++ )
27
28     // loop to control number of comparisons per pass
29     for ( int j = 0; j < arraySize - 1; j++ )
30
31         // compare side-by-side elements and swap them if
32         // first element is greater than second element
33         if ( a[ j ] > a[ j + 1 ] ) {
34             hold = a[ j ];
35             a[ j ] = a[ j + 1 ];
36             a[ j + 1 ] = hold;
37
38         } // end if
39
```

Nếu phần tử bên trái (chỉ số **j**) lớn hơn phần tử bên phải (chỉ số **j + 1**), thì ta trao đổi chúng. Nhớ sử dụng biến tạm.

```

40     cout << "\nData items in ascending order\n";
41
42     // output sorted array
43     for ( int k = 0; k < arraySize; k++ )
44         cout << setw( 4 ) << a[ k ];
45
46     cout << endl;
47
48     return 0; // indicates successful termination
49
50 } // end main

```

fig04\_16.cpp  
(3 of 3)

fig04\_16.cpp  
output (1 of 1)

```

Data items in original order
  2   6   4   8  10  12  89  68  45  37
Data items in ascending order
  2   4   6   8  10  12  37  45  68  89

```

## 4.7 Ví dụ: sử dụng mảng để tính Mean, Median và Mode

- Mean
  - Giá trị trung bình (tổng/số phần tử)
- Median
  - Giá trị ở giữa dãy đã được sắp xếp
  - 1, 2, 3, 4, 5 (3 là median)
  - Nếu số phần tử là số chẵn, lấy trung bình của 2 số giữa
- Mode
  - Giá trị xuất hiện nhiều nhất
  - 1, 1, 1, 2, 3, 3, 4, 5 (1 là mode)



```

1  // Fig. 4.17: fig04_17.cpp
2  // This program introduces the topic of survey data
   analysis.
3  // It computes the mean, median, and mode of the data.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8  using std::fixed;
9  using std::showpoint;
10
11 #include <iomanip>
12
13 using std::setw;
14 using std::setprecision;
15
16 void mean( const int [], int );
17 void median( int [], int );
18 void mode( int [], int [], int );
19 void bubbleSort( int[], int );
20 void printArray( const int[], int );
21
22 int main()
23 {
24     const int responseSize = 99;  // size of array responses
25

```

fig04\_17.cpp  
(1 of 8)

04\_17.cpp  
of 8)

```
26  int frequency[ 10 ] = { 0 };  // initialize array frequency
27
28  // initialize array responses
29  int response[ responseSize ] =
30      { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
31        7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
32        6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
33        7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
34        6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
35        7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
36        5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
37        7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
38        7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
39        4, 5, 6, 1, 6, 5, 7, 8, 7 };
40
41  // process responses
42  mean( response, responseSize );
43  median( response, responseSize );
44  mode( frequency, response, responseSize );
45
46  return 0;  // indicates successful termination
47
48 } // end main
49
```

```

50 // calculate average of all response values
51 void mean( const int answer[], int arraySize )
52 {
53     int total = 0;
54
55     cout << "*****\n  Mean\n*****\n";
56
57     // total response values
58     for ( int i = 0; i < arraySize; i++ )
59         total += answer[ i ];
60
61     // format and output results
62     cout << fixed << setprecision( 4 );
63
64     cout << "The mean is the average value of the data\n"
65           << "items. The mean is equal to the total of\n"
66           << "all the data items divided by the number\n"
67           << "of data items (" << arraySize
68           << "). The mean value for\nthis run is: "
69           << total << " / " << arraySize << " = "
70           << static_cast< double >( total ) / arraySize
71           << "\n\n";
72
73 } // end function mean
74

```

fig04\_17.cpp  
(3 of 8)

Đổi sang **double** để được giá trị trung bình bằng số thực (thay vì giá trị nguyên).

```

75 // sort array and determine median element's value
76 void median( int answer[], int size )
77 {
78     cout << "\n*****\n Median\n*****\n"
79         << "The unsorted array of responses is";
80
81     printArray( answer, size ); // output unsorted array
82
83     bubbleSort( answer, size ); // sort array
84
85     cout << "\n\nThe sorted array is";
86     printArray( answer, size ); // output sorted array
87
88     // display median element
89     cout << "\n\nThe median is element " << size / 2
90         << " of\nthe sorted " << size
91         << " element array.\nFor this run the median is "
92         << answer[ size / 2 ] << "\n\n";
93
94 } // end function median
95

```

fig04\_17.cpp  
(4 of 8)

Sắp xếp mảng bằng cách  
truyền nó cho một hàm.  
Bảo vệ tính modun của  
chương trình

```
96 // determine most frequent response
97 void mode( int freq[], int answer[], int size )
98 {
99     int largest = 0;    // represents largest frequency
100    int modeValue = 0;  // represents most frequent response
101
102    cout << "\n*****\n  Mode\n*****\n";
103
104    // initialize frequencies to 0
105    for ( int i = 1; i <= 9; i++ )
106        freq[ i ] = 0;
107
108    // summarize frequencies
109    for ( int j = 0; j < size; j++ )
110        ++freq[ answer[ j ] ];
111
112    // output headers for result columns
113    cout << "Response" << setw( 11 ) << "Frequency"
114          << setw( 19 ) << "Histogram\n\n" << setw( 55 )
115          << "1      1      2      2\n" << setw( 56 )
116          << "5      0      5      0      5\n\n";
117
```

```
118 // output results
119 for ( int rating = 1; rating <= 9; rating++ ) {
120     cout << setw( 8 ) << rating << setw( 11 )
121         << freq[ rating ] << "          ";
122
123     // keep track of mode value and largest frequency value
124     if ( freq[ rating ] > largest ) {
125         largest = freq[ rating ];
126         modeValue = rating;
127
128     } // end if
129
130     // output histogram bar representing frequency value
131     for ( int k = 1; k <= freq[ rating ]; k++ )
132         cout << '*';
133
134     cout << '\n'; // begin new line of output
135
136 } // end outer for
137
138 // display the mode value
139 cout << "The mode is the most frequent value.\n"
140     << "For this run the mode is " << modeValue
141     << " which occurred " << largest << " times." << endl;
142
143 } // end function mode
```

**mode** là giá trị xuất hiện  
nhiều nhất (có giá trị cao nhất  
trong mảng **freq**).

```
144
145 // function that sorts an array with bubble sort algorithm
146 void bubbleSort( int a[], int size )
147 {
148     int hold; // temporary location used to swap elements
149
150     // loop to control number of passes
151     for ( int pass = 1; pass < size; pass++ )
152
153         // loop to control number of comparisons per pass
154         for ( int j = 0; j < size - 1; j++ )
155
156             // swap elements if out of order
157             if ( a[ j ] > a[ j + 1 ] ) {
158                 hold = a[ j ];
159                 a[ j ] = a[ j + 1 ];
160                 a[ j + 1 ] = hold;
161
162             } // end if
163
164 } // end function bubbleSort
165
```

```
166 // output array contents (20 values per row)
167 void printArray( const int a[], int size )
168 {
169     for ( int i = 0; i < size; i++ ) {
170
171         if ( i % 20 == 0 ) // begin new line every 20 values
172             cout << endl;
173
174         cout << setw( 2 ) << a[ i ];
175
176     } // end for
177
178 } // end function printArray
```

fig04\_17.cpp  
(8 of 8)



\*\*\*\*\*

Mean

\*\*\*\*\*

The mean is the average value of the data items. The mean is equal to the total of all the data items divided by the number of data items (99). The mean value for this run is:  $681 / 99 = 6.8788$

\*\*\*\*\*

Median

\*\*\*\*\*

The unsorted array of responses is

6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8  
6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9  
6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3  
5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 8  
7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7

The sorted array is

1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 5  
5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7  
7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8  
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

The median is element 49 of the sorted 99 element array. For this run the median is 7

\*\*\*\*\*  
Mode  
\*\*\*\*\*

| Response | Frequency | Histogram            |
|----------|-----------|----------------------|
|          |           | 1 1 2 2<br>5 0 5 0 5 |
| 1        | 1         | *                    |
| 2        | 3         | ***                  |
| 3        | 4         | ****                 |
| 4        | 5         | *****                |
| 5        | 8         | *****                |
| 6        | 9         | *****                |
| 7        | 23        | *****                |
| 8        | 27        | *****                |
| 9        | 19        | *****                |

The mode is the most frequent value.  
For this run the mode is 8 which occurred 27 times.

fig04\_17.cpp  
output (2 of 2)

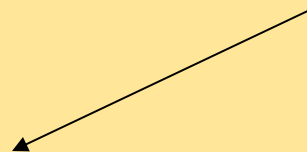
## 4.8 Tìm kiếm trên mảng: Tìm kiếm Tuyến tính và tìm kiếm Nhị phân

- Tìm một giá trị khoá (key value) trên mảng
- Tìm kiếm tuyến tính
  - So sánh từng phần tử của mảng với key
    - Bắt đầu từ một đầu, đi đến đầu kia của mảng
  - Hữu dụng cho mảng nhỏ và chưa sắp xếp
    - Không hiệu quả
    - Nếu giá trị cần tìm không có trong mảng thì phải kiểm tra tất cả các phần tử

## 4.8 Tìm kiếm trên mảng: Tìm kiếm Tuyến tính và tìm kiếm Nhị phân

- Tìm kiếm nhị phân
  - Chỉ sử dụng cho mảng đã sắp xếp
  - So sánh phần tử ở giữa (middle) với key
    - Nếu bằng, tìm thấy
    - Nếu  $key < middle$ 
      - Lặp lại ở nửa đầu của mảng
    - Nếu  $key > middle$ 
      - Lặp lại ở nửa cuối
  - Rất nhanh
    - Nhiều nhất là N bước với  $2^N > \text{số phần tử của mảng}$
    - mảng 30 phần tử cần nhiều nhất 5 bước
$$2^5 > 30$$

Lấy đối số là một mảng, khoá cần tìm, và kích thước mảng.



```
1  // Fig. 4.19: fig04_19.cpp
2  // Linear search of an array.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int linearSearch( const int [], int, int ); // prototype
10
11 int main()
12 {
13     const int arraySize = 100; // size of array a
14     int a[ arraySize ];        // create array a
15     int searchKey;             // value to locate in a
16
17     for ( int i = 0; i < arraySize; i++ ) // create some
data
18         a[ i ] = 2 * i;
19
20     cout << "Enter integer search key: ";
21     cin >> searchKey;
22
23     // attempt to locate searchKey in array a
24     int element = linearSearch( a, searchKey, arraySize );
25
```

fig04\_19.cpp

```
26 // display results
27 if ( element != -1 )
28     cout << "Found value in element " << element << endl;
29 else
30     cout << "Value not found" << endl;
31
32 return 0; // indicates successful termination
33
34 } // end main
35
36 // compare key to every element of array until location is
37 // found or until end of array is reached; return subscript of
38 // element if key or -1 if key not found
39 int linearSearch( const int array[], int key, int sizeOfArray )
40 {
41     for ( int j = 0; j < sizeOfArray; j++ )
42
43         if ( array[ j ] == key ) // if found,
44             return j;           // return location of key
45
46     return -1; // key not found
47
48 } // end function linearSearch
```

Enter integer search key: 36  
Found value in element 18

Enter integer search key: 37  
Value not found

```
1  // Fig. 4.20: fig04_20.cpp
2  // Binary search of an array.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  #include <iomanip>
10
11 using std::setw;
12
13 // function prototypes
14 int binarySearch( const int [], int, int, int, int );
15 void printHeader( int );
16 void printRow( const int [], int, int, int, int );
17
18 int main()
19 {
20     const int arraySize = 15; // size of array a
21     int a[ arraySize ];      // create array a
22     int key;                  // value to locate in a
23
24     for ( int i = 0; i < arraySize; i++ ) // create some
data
25         a[ i ] = 2 * i;
```

```
27     cout << "Enter a number between 0 and 28: ";
28     cin >> key;
29
30     printHeader( arraySize );
31
32     // search for key in array a
33     int result =
34         binarySearch( a, key, 0, arraySize - 1, arraySize );
35
36     // display results
37     if ( result != -1 )
38         cout << '\n' << key << " found in array element "
39             << result << endl;
40     else
41         cout << '\n' << key << " not found" << endl;
42
43     return 0; // indicates successful termination
44
45 } // end main
46
```

fig04\_20.cpp  
(2 of 6)



```
47 // function to perform binary search of an array
48 int binarySearch( const int b[], int searchKey, int low,
49     int high, int size )
50 {
51     int middle;
52
53     // loop until low subscript is greater than high subscript
54     while ( low <= high ) {
55
56         // determine middle element of subarray being searched
57         middle = ( low + high ) / 2;
58
59         // display subarray used in this loop iteration
60         printRow( b, low, middle, high, size );
61     }
```

Xác định phần tử ở giữa

fig04\_20.cpp  
(3 of 6)

```
62 // if searchKey matches middle element, return middle
63 if ( searchKey == b[ middle ] ) // match
64     return middle;
65
66 else
67
68     // if searchKey less than middle element,
69     // set new high element
70     if ( searchKey < b[ middle ] )
71         high = middle - 1; // search low end of array
72
73     // if searchKey greater than middle element,
74     // set new low element
75     else
76         low = middle + 1; // search high end of array
77 }
78
79 return -1; // searchKey not found
80
81 } // end function binarySearch
```

Sử dụng tìm Nhị phân:  
Nếu key bằng middle, tìm thấy  
  
Nếu nhỏ hơn, tìm nửa thấp  
  
Nếu lớn hơn, tìm nửa cao

Vòng lặp tạo low, middle và high tự động. Nếu tìm nửa cao, thì phần tử low mới sẽ cao hơn middle.

```
82
83 // print header for output
84 void printHeader( int size )
85 {
86     cout << "\nSubscripts:\n";
87
88     // output column heads
89     for ( int j = 0; j < size; j++ )
90         cout << setw( 3 ) << j << ' ';
91
92     cout << '\n'; // start new line of output
93
94     // output line of - characters
95     for ( int k = 1; k <= 4 * size; k++ )
96         cout << '-';
97
98     cout << endl; // start new line of output
99
100 } // end function printHeader
101
```

fig04\_20.cpp  
(5 of 6)

```
102 // print one row of output showing the current
103 // part of the array being processed
104 void printRow( const int b[], int low, int mid,
105     int high, int size )
106 {
107     // loop through entire array
108     for ( int m = 0; m < size; m++ )
109
110         // display spaces if outside current subarray range
111         if ( m < low || m > high )
112             cout << "    ";
113
114         // display middle element marked with a *
115         else
116
117             if ( m == mid )           // mark middle value
118                 cout << setw( 3 ) << b[ m ] << '*';
119
120         // display other elements in subarray
121         else
122             cout << setw( 3 ) << b[ m ] << ' ';
123
124     cout << endl; // start new line of output
125
126 } // end function printRow
```

Enter a number between 0 and 28: 6

Subscripts:

| 0     | 1 | 2 | 3  | 4 | 5  | 6  | 7   | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
|-------|---|---|----|---|----|----|-----|----|----|----|----|----|----|----|
| ----- |   |   |    |   |    |    |     |    |    |    |    |    |    |    |
| 0     | 2 | 4 | 6  | 8 | 10 | 12 | 14* | 16 | 18 | 20 | 22 | 24 | 26 | 28 |
| 0     | 2 | 4 | 6* | 8 | 10 | 12 |     |    |    |    |    |    |    |    |

6 found in array element 3

Enter a number between 0 and 28: 25

Subscripts:

| 0     | 1 | 2 | 3 | 4 | 5  | 6  | 7   | 8  | 9  | 10 | 11  | 12  | 13  | 14 |
|-------|---|---|---|---|----|----|-----|----|----|----|-----|-----|-----|----|
| ----- |   |   |   |   |    |    |     |    |    |    |     |     |     |    |
| 0     | 2 | 4 | 6 | 8 | 10 | 12 | 14* | 16 | 18 | 20 | 22  | 24  | 26  | 28 |
|       |   |   |   |   |    |    |     | 16 | 18 | 20 | 22* | 24  | 26  | 28 |
|       |   |   |   |   |    |    |     |    |    |    |     | 24  | 26* | 28 |
|       |   |   |   |   |    |    |     |    |    |    |     | 24* |     |    |

25 not found

Enter a number between 0 and 28: 8

Subscripts:

| 0     | 1 | 2 | 3  | 4  | 5   | 6  | 7   | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
|-------|---|---|----|----|-----|----|-----|----|----|----|----|----|----|----|
| ----- |   |   |    |    |     |    |     |    |    |    |    |    |    |    |
| 0     | 2 | 4 | 6  | 8  | 10  | 12 | 14* | 16 | 18 | 20 | 22 | 24 | 26 | 28 |
| 0     | 2 | 4 | 6* | 8  | 10  | 12 |     |    |    |    |    |    |    |    |
|       |   |   |    | 8  | 10* | 12 |     |    |    |    |    |    |    |    |
|       |   |   |    | 8* |     |    |     |    |    |    |    |    |    |    |

8 found in array element 4

## 4.9 Mảng nhiều chiều

- Đa chỉ số
  - `int a[ 3 ][ 4 ];`
  - `a[ i ][ j ]`
  - Các bảng có dòng và cột
  - Dòng trước, cột sau
  - “Mảng của mảng”
    - `a[0]` là một mảng 4 phần tử
    - `a[0][0]` là phần tử đầu tiên của mảng

|       | Column 0                 | Column 1                 | Column 2                 | Column 3                 |
|-------|--------------------------|--------------------------|--------------------------|--------------------------|
| Row 0 | <code>a[ 0 ][ 0 ]</code> | <code>a[ 0 ][ 1 ]</code> | <code>a[ 0 ][ 2 ]</code> | <code>a[ 0 ][ 3 ]</code> |
| Row 1 | <code>a[ 1 ][ 0 ]</code> | <code>a[ 1 ][ 1 ]</code> | <code>a[ 1 ][ 2 ]</code> | <code>a[ 1 ][ 3 ]</code> |
| Row 2 | <code>a[ 2 ][ 0 ]</code> | <code>a[ 2 ][ 1 ]</code> | <code>a[ 2 ][ 2 ]</code> | <code>a[ 2 ][ 3 ]</code> |

Array name

Row subscript  
(chỉ số dòng)

Column subscript  
(chỉ số cột)

## 4.9 Mảng nhiều chiều

- Khởi tạo
  - Mặc định là **0**
  - Khởi tạo, mỗi dòng trong 1 cặp ngoặc

```
int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```

Row 0                      Row 1

|   |   |
|---|---|
| 1 | 2 |
| 3 | 4 |

```
int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };
```

|   |   |
|---|---|
| 1 | 0 |
| 3 | 4 |



## 4.9 Mảng nhiều chiều

- Truy nhập đến như bình thường

```
cout << b[ 0 ][ 1 ];
```

- In ra **0**
- Không sử dụng dấu phẩy (,)

|   |   |
|---|---|
| 1 | 0 |
| 3 | 4 |

```
cout << b[ 0, 1 ];
```

- Lỗi cú pháp

- Function prototype

- Phải chỉ rõ kích thước của các chỉ số
  - Không đòi hỏi kích thước cho chỉ số đầu tiên, cũng như mảng 1 chiều
- **void printArray( int [][][ 3 ] );**

Chú ý cấu trúc của prototype.

Chú ý nhiều cách khởi tạo.  
Các phần tử trong **array2**  
được gán từ dòng thứ nhất  
rồi đến dòng thứ hai.

```
1  // Fig. 4.22: fig04_22.cpp
2  // Initializing multidimensional arrays.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  void printArray( int [][] [ 3 ] );
9
10 int main()
11 {
12     int array1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
13     int array2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5};
14     int array3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };
15
16     cout << "Values in array1 by row are:" << endl;
17     printArray( array1 );
18
19     cout << "Values in array2 by row are:" << endl;
20     printArray( array2 );
21
22     cout << "Values in array3 by row are:" << endl;
23     printArray( array3 );
24
25     return 0; // indicates successful termination
26 } // end main
```

```

28
29 // function to output array with two rows and three columns
30 void printArray( int a[][ 3 ] )
31 {
32     for ( int i = 0; i < 2; i++ ) {    // for each row
33
34         for ( int j = 0; j < 3; j++ )    // output column values
35             cout << a[ i ][ j ] << ' ';
36
37         cout << endl;    // start new line of output
38
39     } // end outer for structure
40
41 } // end function printArray

```

Vòng lặp for thường được dùng để quét qua mảng. Sử dụng vòng lặp lồng nhau cho mảng nhiều chiều.

ng04\_22.cpp  
output (1 of 1)

Values in array1 by row are:

1 2 3

4 5 6

Values in array2 by row are:

1 2 3

4 5 0

Values in array3 by row are:

1 2 0

4 0 0

## 4.9 Mảng nhiều chiều

- Tiếp theo: chương trình ví dụ về khởi tạo mảng
  - Chương trình lưu trữ điểm của sinh viên
  - Mảng nhiều chiều (bảng)
  - Dòng là sinh viên
  - Cột là điểm

|          | Quiz1 | Quiz2 |
|----------|-------|-------|
| Student0 | 95    | 85    |
| Student1 | 89    | 80    |

```
1 // Fig. 4.23: fig04_23.cpp
2 // Double-subscripted array example.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7 using std::fixed;
8 using std::left;
9
10 #include <iomanip>
11
12 using std::setw;
13 using std::setprecision;
14
15 const int students = 3; // number of students
16 const int exams = 4;    // number of exams
17
18 // function prototypes
19 int minimum( int [][] exams, int, int );
20 int maximum( int [][] exams, int, int );
21 double average( int [], int );
22 void printArray( int [][] exams, int, int );
23
```

**fig04\_23.cpp**  
**(1 of 6)**

```

24 int main()
25 {
26     // initialize student grades for three students (rows)
27     int studentGrades[ students ][ exams ] =
28         { { 77, 68, 86, 73 },
29           { 96, 87, 89, 78 },
30           { 70, 90, 86, 81 } };
31
32     // output array studentGrades
33     cout << "The array is:\n";
34     printArray( studentGrades, students, exams );
35
36     // determine smallest and largest grade values
37     cout << "\n\nLowest grade: "
38           << minimum( studentGrades, students, exams )
39           << "\nHighest grade: "
40           << maximum( studentGrades, students, exams ) << '\n';
41
42     cout << fixed << setprecision( 2 );
43

```

fig04\_23.cpp  
(2 of 6)

```
44 // calculate average grade for each student
45 for ( int person = 0; person < students; person++ )
46     cout << "The average grade for student " << person
47         << " is "
48         << average( studentGrades[ person ], exams )
49         << endl;
50
51 return 0; // indicates successful termination
52
53 } // end main
54
55 // find minimum grade
56 int minimum( int grades[][ exams ], int pupils, int tests )
57 {
58     int lowGrade = 100; // initialize to highest possible grade
59
60     for ( int i = 0; i < pupils; i++ )
61
62         for ( int j = 0; j < tests; j++ )
63
64             if ( grades[ i ][ j ] < lowGrade )
65                 lowGrade = grades[ i ][ j ];
66
67     return lowGrade;
68
69 } // end function minimum
```

Tính điểm trung bình cho sinh viên. Ta truyền dòng chứa điểm của sinh viên vào hàm. Chú ý: **studentGrades[0]** cũng là một mảng.

```
70
71 // find maximum grade
72 int maximum( int grades[][ exams ], int pupils, int tests )
73 {
74     int highGrade = 0; // initialize to lowest possible grade
75
76     for ( int i = 0; i < pupils; i++ )
77
78         for ( int j = 0; j < tests; j++ )
79
80             if ( grades[ i ][ j ] > highGrade )
81                 highGrade = grades[ i ][ j ];
82
83     return highGrade;
84
85 } // end function maximum
86
```

fig04\_23.cpp  
(4 of 6)



```
87 // determine average grade for particular student
88 double average( int setOfGrades[], int tests )
89 {
90     int total = 0;
91
92     // total all grades for one student
93     for ( int i = 0; i < tests; i++ )
94         total += setOfGrades[ i ];
95
96     return static_cast< double >( total ) / tests; // average
97
98 } // end function maximum
```

fig04\_23.cpp  
(5 of 6)

```
99
100 // Print the array
101 void printArray( int grades[][ exams ], int pupils, int tests )
102 {
103     // set left justification and output column heads
104     cout << left << "                [0]  [1]  [2]  [3]";
105
106     // output grades in tabular format
107     for ( int i = 0; i < pupils; i++ ) {
108
109         // output label for row
110         cout << "\nstudentGrades[" << i << "]" << " ";
111
112         // output one grades for one student
113         for ( int j = 0; j < tests; j++ )
114             cout << setw( 5 ) << grades[ i ][ j ];
115
116     } // end outer for
117
118 } // end function printArray
```

The array is:

|                  | [0] | [1] | [2] | [3] |
|------------------|-----|-----|-----|-----|
| studentGrades[0] | 77  | 68  | 86  | 73  |
| studentGrades[1] | 96  | 87  | 89  | 78  |
| studentGrades[2] | 70  | 90  | 86  | 81  |

Lowest grade: 68

Highest grade: 96

The average grade for student 0 is 76.00

The average grade for student 1 is 87.50

The average grade for student 2 is 81.75

fig04\_23.cpp  
output (1 of 1)