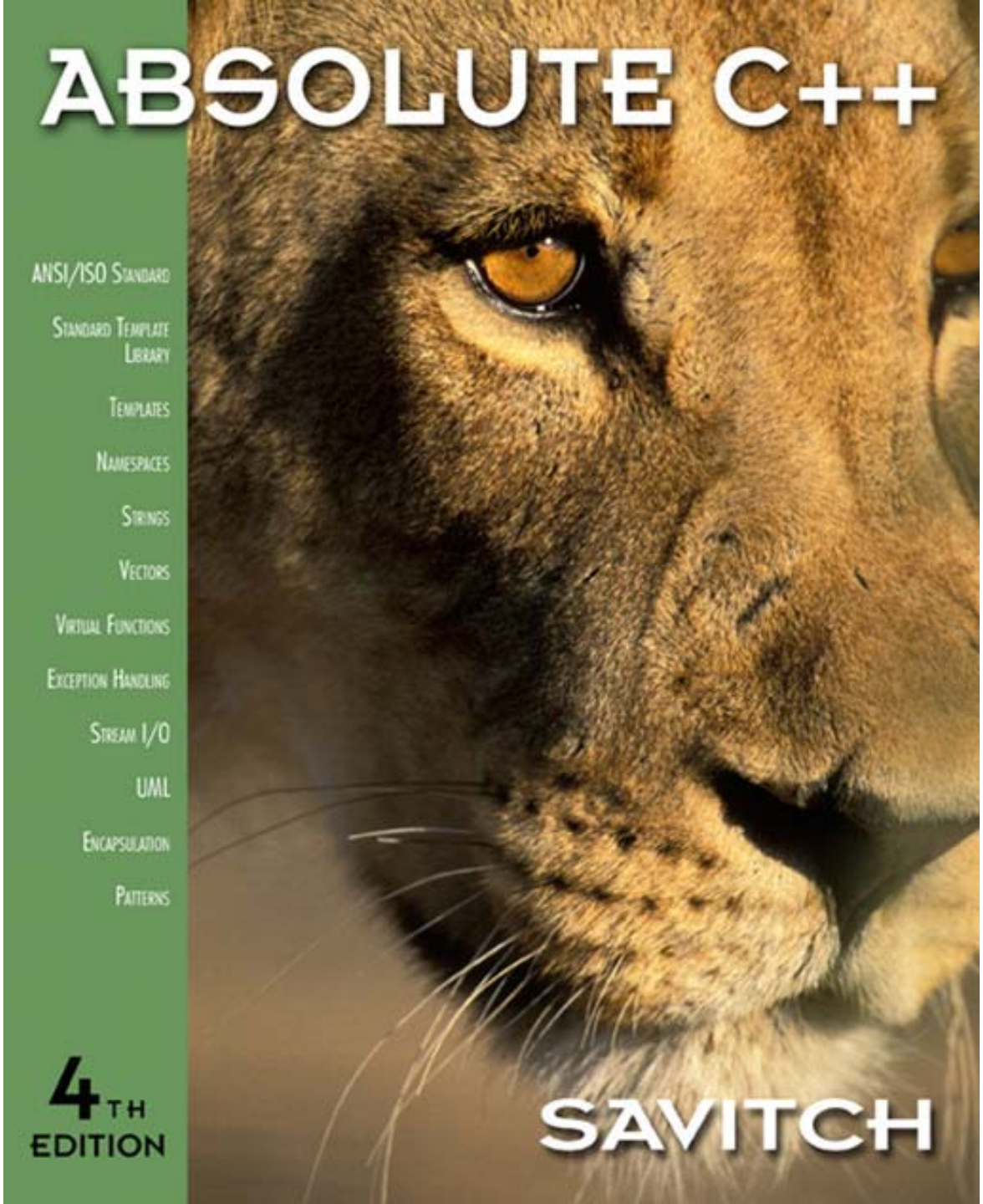
Bài 5: Mảng

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



Chapter 5

Arrays

PEARSON Addison Wesley

Mục tiêu bài học

- Giới thiệu mảng
  - Khai báo và tham chiếu mảng
  - Lệnh lặp for và mảng
  - Mảng trong bộ nhớ
- Mảng và hàm
  - Hàm có đối số là mảng
  - Hàm có giá trị trả về là mảng
- Lập trình với mảng
  - Mảng chưa đầy (Partially Filled Arrays)
  - Tìm kiếm, sắp xếp
- Mảng nhiều chiều

Giới thiệu mảng

- Định nghĩa mảng:
  - Tập hợp các phần tử dữ liệu cùng kiểu
- Đây là kiểu dữ liệu "nhóm" đầu tiên ta học
  - int, float, double, char là những kiểu dữ liệu đơn
- Dùng biểu diễn danh sách các phần tử giống nhau
  - Danh sách điểm thi, nhiệt độ, tên, …
  - Tránh khai báo nhiều biến đơn
  - Có thể thao tác với "danh sách" này như với một thực thể

Khai báo mảng

- Khai báo mảng → cấp phát bộ nhớ
  int score[5];
  - Khai báo mảng 5 số nguyên, có tên là "score"
  - Tương tự như khai báo 5 biến:
    int score[0], score[1], score[2], score[3], score[4]
- Mỗi cá thể trong mảng được gọi bằng rất nhiều tên:
  - Biến được đánh chỉ mục hoặc chỉ số
  - “Phần tử" của mảng
  - Giá trị trong cặp ngoặc vuông gọi là chỉ số
    - Miền giá trị từ 0 tới size - 1

Truy cập mảng

- Phép truy cập sử dụng chỉ số
  - cout << score[3];
- Lưu ý cách dùng cặp ngoặc vuông:
  - Trong lệnh khai báo, nó chỉ định kích thước của mảng
  - Ở những nơi khác, nó xác định chỉ số
- Kích thước và chỉ số không nhất thiết phải là giá trị hằng
  - int score[MAX_SCORES];
  - score[n+1] = 99;
    - Nếu n là 2, tương đương với score[3]

Sử dụng mảng

- Cơ chế mạnh dùng cho lưu trữ
- Có thể thực hiện những công việc như:
  - “Làm việc này với biến có chỉ số thứ i"
    trong đó i được tính bởi chương trình
  - “Hiển thị tất cả các phần tử của mảng score"
  - “Điền cho mảng score dữ liệu người dùng nhập vào"
  - “Tìm giá trị lớn nhất trong mảng score"
  - “Tìm giá trị nhỏ nhất trong mảng score"

Ví dụ chương trình dùng mảng:
Display 5.1 Chương trình dùng mảng (1/2)

Display 5.1 Program Using an Array

```
1  //Reads in five scores and shows how much each
2  //score differs from the highest score.
3  #include <iostream>
4  using namespace std;
5  int main( )
6  {
7      int i, score[5], max;
8      cout << "Enter 5 scores:\n";
9      cin >> score[0];
10     max = score[0];
11     for (i = 1; i < 5; i++)
12     {
13         cin >> score[i];
14         if (score[i] > max)
15             max = score[i];
16         //max is the largest of the values score[0],..., score[i].
17     }
```

Ví dụ chương trình dùng mảng:
Display 5.1 Chương trình dùng mảng (2/2)

```
18        cout << "The highest score is " << max << endl
19             << "The scores and their\n"
20             << "differences from the highest are:\n";
21        for (i = 0; i < 5; i++)
22            cout << score[i] << " off by "
23                 << (max - score[i]) << endl;
24        return 0;
25    }
```

SAMPLE DIALOGUE

Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4

Lệnh lặp for và mảng

- for là lệnh lặp đếm tự nhiên
  - Có thể khảo sát lần lượt các phần tử trong mảng
- Ví dụ:

```
for (idx = 0; idx<5; idx++)
{
      cout << score[idx] << "off by "
                  << max – score[idx] << endl;
}
```

  - Biến điều khiển vòng lặp (idx) đếm từ 0 – 5

Lỗi lớn khi dùng mảng

- Các chỉ số của mảng luôn bắt đầu từ 0
- 0 là con số "đầu tiên" với người làm công nghệ thông tin
- C++ sẽ "cho phép" bạn vượt ra ngoài miền này
  - Kết quả là không đoán trước được
  - Trình biên dịch sẽ không phát hiện ra những lỗi này!
- Lập trình viên phải tự kiểm soát "miền" của chỉ số

Ví dụ về lỗi lớn khi dùng mảng

- Miền chỉ số từ 0 tới (array_size – 1)
  - Ví dụ:
    double temperature[24]; // cỡ của mảng là 24
    // Khai báo mảng 24 giá trị double có tên là temperature
    - Chúng được đánh chỉ số là:
      temperature[0], temperature[1] … temperature[23]
  - Lỗi thường gặp:
    temperature[24] = 5;
    - Chỉ số 24 nằm ngoài miền!
    - Không có cảnh báo, kết quả có thể rất tàn khốc

Dùng hằng có tên chỉ định kích thước mảng

- Hãy dùng hằng có tên để chỉ định kích thước mảng
- Ví dụ:
  ```
  const int NUMBER_OF_STUDENTS = 5;
  int score[NUMBER_OF_STUDENTS];
  ```
- Dễ đọc hơn
- Linh hoạt hơn
- Dễ bảo trì hơn

Dùng hằng có tên

- Dùng ở mọi nơi cần tới kích thước của mảng
  - Khi duyệt vòng lặp for:

```
for (idx = 0; idx < NUMBER_OF_STUDENTS; idx++)
{
    // Thao tác với mảng
}
```

  - Trong các phép tính liên quan kích thước:

```
lastIndex = (NUMBER_OF_STUDENTS – 1);
```

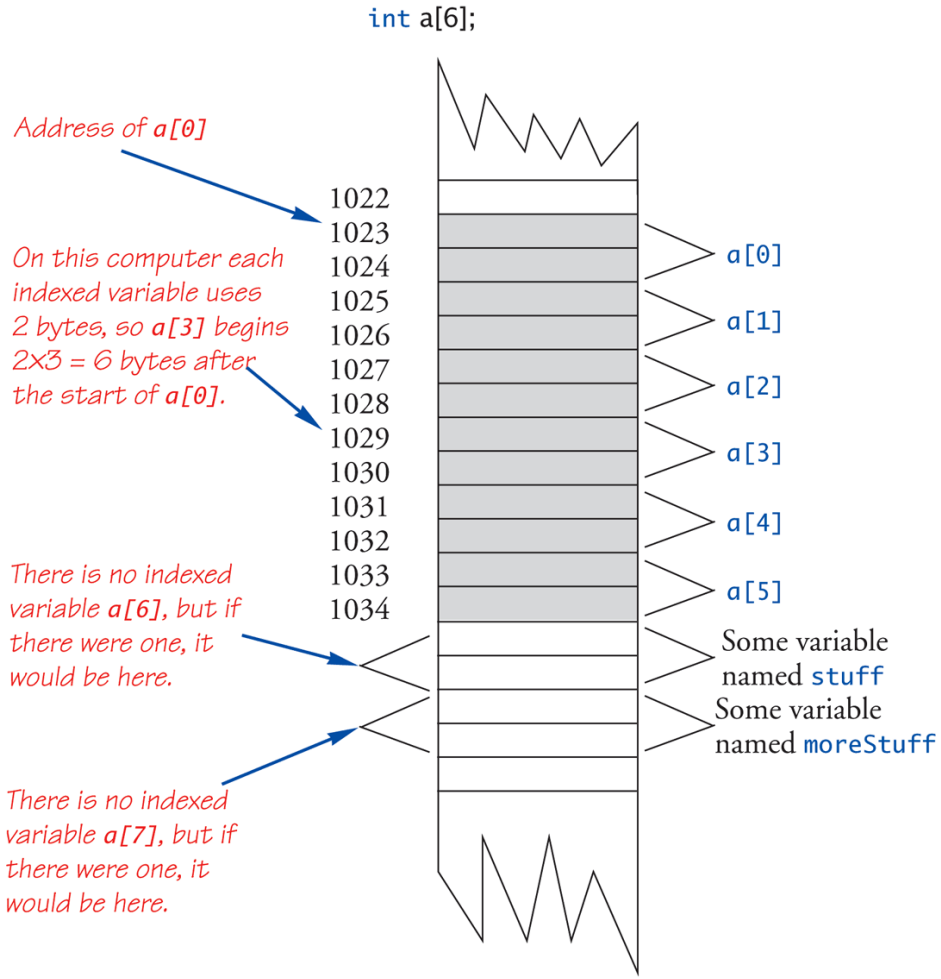
  - Khi truyền mảng vào hàm (sẽ bàn sau)
- Nếu kích thước thay đổi → chỉ cần sửa mã nguồn ở một nơi trong chương trình!

Mảng trong bộ nhớ

- Nhắc lại: Những biến đơn
  - được cấp phát bộ nhớ bằng một "địa chỉ"
- Khai báo mảng cấp phát bộ nhớ cho toàn bộ mảng
- Cấp phát tuần tự
  - Nghĩa là các địa chỉ được cấp phát liền kề nhau
  - Có thể làm phép tính trên chỉ số
    - "Phép cộng" đơn giản từ địa chỉ đầu mảng (chỉ số 0)

Một mảng trong bộ nhớ

Display 5.2 **An Array in Memory**



Khởi tạo mảng

- Các biến đơn có thể khởi tạo lúc khai báo:
  ```
  int price = 0;     // 0 là giá trị khởi tạo
  ```
- Cũng có thể làm vậy với mảng:
  ```
  int children[3] = {2, 12, 1};
  ```
  - Tương đương với:
    ```
    int children[3];
    children[0] = 2;
    children[1] = 12;
    children[2] = 1;
    ```

Mảng khởi tạo tự động

- Nếu số giá trị khởi tạo bạn cung cấp ít hơn kích thước mảng:
  - Chương trình sẽ điền các giá trị này từ đầu mảng
  - Điền "phần còn lại" với giá trị 0 của kiểu dữ liệu chỉ định cho mảng
- Nếu không chỉ định cỡ của mảng
  - Khai báo mảng với cỡ đủ để chứa các giá trị khởi tạo
  - Ví dụ:
    int b[] = {5, 12, 11};
    - Cấp phát mảng b cỡ là 3

Mảng và hàm

- Mảng là đối số
  - Các biến được đánh chỉ số
    - Mỗi phần tử đơn lẻ trong mảng có thể là một tham số hàm
  - Toàn bộ mảng
    - Tất cả các phần tử trong mảng có thể được truyền như “một thực thể”
- Mảng là giá trị trả về
  - Có thể làm được việc này → xem chương 10 giáo trình

Biến đánh chỉ số làm đối số

- Ta xử lý biến đánh chỉ số giống như các biến đơn cùng kiểu với mảng
- Cho khai báo hàm:

  ```
  void myFunction(double par1);
  ```

- Và những khai báo:

  ```
  int i;

  double n, a[10];
  ```

- Ta có thể có những lời gọi sau:

  ```
  myFunction(i);      // i được chuyển thành kiểu double
  myFunction(a[3]);// a[3] có kiểu double
  myFunction(n);      // n có kiểu double
  ```

Khéo léo trong việc dùng chỉ số

- Xem xét các lời gọi:
  myFunction(a[i]);
  - Giá trị của i được xác định trước
    - Chương trình quyết định xem biến đánh chỉ số nào sẽ được truyền vào hàm

  myFunction(a[i*5]);
  - Hoàn toàn hợp lệ từ góc nhìn của trình biên dịch
  - Lập trình viên chịu trách nhiệm giữ chỉ số trong miền có nghĩa

Mảng làm đối số

- Tham số hình thức có thể là một mảng
  - Đối số trong lời gọi hàm sẽ là một tên mảng
  - Gọi là tham số mảng
- Hãy truyền cả kích cỡ của mảng
  - Thường là tham số thứ 2
  - Có thể viết đơn giản là tham số hình thức kiểu int

Ví dụ mảng làm đối số:
Display 5.3 Hàm với một tham số mảng

Display 5.3 Function with an Array Parameter

SAMPLE DIALOGUEFUNCTION DECLARATION

```
void fillUp(int a[], int size);
//Precondition: size is the declared size of the array a.
//The user will type in size integers.
//Postcondition: The array a is filled with size integers
//from the keyboard.
```

SAMPLE DIALOGUEFUNCTION DEFINITION

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index used is " << (size - 1) << endl;
}
```

Ví dụ mảng làm đối số

- Xét ví dụ ở slide trước:
- Trong định nghĩa main() nào đó, xem xét những lời gọi sau:

```
int score[5], numberOfScores = 5;
fillUp(score, numberOfScores);
```

  - Đối số thứ nhất là một mảng
  - Đối số thứ 2 là một giá trị nguyên
  - Lưu ý không có cặp ngoặc vuông trong đối số mảng!

Mảng làm đối số: Chi tiết các bước

- Cái gì thực sự được truyền vào?
- Tưởng tượng mảng có 3 “phần”
  - Địa chỉ của biến đánh chỉ số đầu tiên (arrName[0])
  - Kiểu của mảng
  - Kích thước của mảng
- Chỉ có một “phần” được truyền vào hàm!
  - Là địa chỉ bắt đầu mảng
  - Rất giống với việc truyền tham chiếu

Tham số mảng

- Có vẻ khác lạ
  - Không có ngoặc vuông trong đối số mảng
  - Phải truyền kích thước riêng biệt
- Một tính chất hữu ích:
  - Có thể dùng cùng một hàm để điền dữ liệu cho bất cứ kích cỡ mảng nào!
  - Là ví dụ điển hình cho tính chất dùng lại của hàm
  - Ví dụ:

```
int score[5], time[10];
fillUp(score, 5);
fillUp(time, 10);
```

Tham số const

- Nhắc lại: tham số mảng thực sự truyền địa chỉ của phần tử đầu tiên
  - Tương tự với việc truyền tham chiếu
- Hàm do đó có thể biến đổi dữ liệu trong mảng!
  - Thường là trong tình huống mong đợi, đôi khi không!
- Khi cần bảo vệ nội dung của mảng khỏi việc biến đổi không mong muốn này
  - Hãy dùng từ khóa "const" trước tham số mảng
    - Gọi là “tham số mảng hằng”
    - Báo cho trình biên dịch “ngăn” các biến đổi

Hàm trả về một mảng

- Hàm không thể trả về mảng theo cách thức nó trả về giá trị cho biến đơn
- Cần dùng một “con trỏ”
- Được thảo luận trong chương 10 giáo trình

Lập trình với mảng

- Nhiều ứng dụng
  - Mảng không đầy
    - Phải khai báo “kích thước tối đa”
  - Sắp xếp
  - Tìm kiếm

Mảng không đầy

- Rất khó biết chính xác ta cần bao nhiêu phần tử mảng
- Phải khai báo một mảng với cỡ lớn nhất có thể cần
  - Phải theo dõi phần nào của mảng chứa dữ liệu hợp lệ
  - Cần thêm một biến lưu thông tin này
    - int numberUsed;
    - Lưu số phần tử hợp lệ hiện thời trong mảng

Ví dụ mảng không đầy:
Display 5.5 Mảng không đầy (1/5)

Display 5.5 Partially Filled Array

```
1   //Shows the difference between each of a list of golf scores and their average.
2   #include <iostream>
3   using namespace std;
4   const int MAX_NUMBER_SCORES = 10;

5   void fillArray(int a[], int size, int& numberUsed);
6   //Precondition: size is the declared size of the array a.
7   //Postcondition: numberUsed is the number of values stored in a.
8   //a[0] through a[numberUsed-1] have been filled with
9   //nonnegative integers read from the keyboard.

10  double computeAverage(const int a[], int numberUsed);
11  //Precondition: a[0] through a[numberUsed-1] have values; numberUsed > 0.
12  //Returns the average of numbers a[0] through a[numberUsed-1].

13  void showDifference(const int a[], int numberUsed);
14  //Precondition: The first numberUsed indexed variables of a have values.
15  //Postcondition: Gives screen output showing how much each of the first
16  //numberUsed elements of the array a differs from their average.
```

(continued)

Ví dụ mảng không đầy:
Display 5.5 Mảng không đầy (2/5)

Display 5.5 Partially Filled Array

```
17   int main( )
18   {
19       int score[MAX_NUMBER_SCORES], numberUsed;

20       cout << "This program reads golf scores and shows\n"
21            << "how much each differs from the average.\n";

22       cout << "Enter golf scores:\n";
23      fillArray(score, MAX_NUMBER_SCORES, numberUsed);
24       showDifference(score, numberUsed);

25       return 0;
26   }
```

Ví dụ mảng không đầy:
Display 5.5 Mảng không đầy (3/5)

```
27  void fillArray(int a[], int size, int& numberUsed)
28  {
29      cout << "Enter up to " << size << " nonnegative whole numbers.\n"
30           << "Mark the end of the list with a negative number.\n";
31      int next, index = 0;
32      cin >> next;
33      while ((next >= 0) && (index < size))
34      {
35          a[index] = next;
36          index++;
37          cin >> next;
38      }
39      numberUsed = index;
40  }
```

Ví dụ mảng không đầy:
Display 5.5 Mảng không đầy (4/5)

```
41  double computeAverage(const int a[], int numberUsed)
42  {
43      double total = 0;
44      for (int index = 0; index < numberUsed; index++)
45          total = total + a[index];
46      if (numberUsed > 0)
47      {
48          return (total/numberUsed);
49      }
50      else
51      {
52          cout << "ERROR: number of elements is 0 in computeAverage.\n"
53               << "computeAverage returns 0.\n";
54          return 0;
55      }
56  }
```

Ví dụ mảng không đầy:
Display 5.5 Mảng không đầy (5/5)

Display 5.5 Partially Filled Array

```
57  void showDifference(const int a[], int numberUsed)
58  {
59      double average = computeAverage(a, numberUsed);
60      cout << "Average of the " << numberUsed
61           << " scores = " << average << endl
62           << "The scores are:\n";
63      for (int index = 0; index < numberUsed; index++)
64      cout << a[index] << " differs from average by "
65           << (a[index] - average) << endl;
66  }
```

SAMPLE DIALOGUE

```
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
69 74 68 -1
Average of the 3 scores = 70.3333
The scores are:
69 differs from average by -1.33333
74 differs from average by 3.66667
68 differs from average by -2.33333
```

So sánh: Hằng toàn cục và tham số

- Hằng thường được khai báo “toàn cục”
  - Phía trên định nghĩa main()
- Do đó, khi bạn khai báo kích thước mảng là hằng toàn cục, hàm có quyền truy cập tới thông tin đó
  - Liệu có cần truyền thêm tham số kích thước?
    - Về lý thuyết: có
  - Vì sao ta vẫn nên có tham số kích thước?
    - Định nghĩa hàm có thể nằm ở một tệp riêng biệt
    - Hàm có thể được dùng bởi chương trình khác!

Tìm kiếm trên mảng

- Là ứng dụng rất hay gặp của mảng
- Xem Display 5.6 ở slide sau

Display 5.6
Tìm kiếm trên mảng (1/4)

Display 5.6 Searching an Array

```
1   //Searches a partially filled array of nonnegative integers.
2   #include <iostream>
3   using namespace std;
4   const int DECLARED_SIZE = 20;

5   void fillArray(int a[], int size, int& numberUsed);
6   //Precondition: size is the declared size of the array a.
7   //Postcondition: numberUsed is the number of values stored in a.
8   //a[0] through a[numberUsed-1] have been filled with
9   //nonnegative integers read from the keyboard.

10  int search(const int a[], int numberUsed, int target);
11  //Precondition: numberUsed is <= the declared size of a.
12  //Also, a[0] through a[numberUsed -1] have values.
13  //Returns the first index such that a[index] == target,
14  //provided there is such an index; otherwise, returns -1.
```

Display 5.6
Tìm kiếm trên mảng (2/4)

```
15  int main( )
16  {
17      int arr[DECLARED_SIZE], listSize, target;

18      fillArray(arr, DECLARED_SIZE, listSize);

19      char ans;
20      int result;
21      do
22      {
23          cout << "Enter a number to search for: ";
24          cin >> target;

25          result = search(arr, listSize, target);
26          if (result == -1)
27              cout << target << " is not on the list.\n";
28          else
29              cout << target << " is stored in array position "
30                   << result << endl
31                   << "(Remember: The first position is 0.)\n";
```

Display 5.6
Tìm kiếm trên mảng (3/4)

Display 5.6 Searching an Array

```
32          cout << "Search again?(y/n followed by Return): ";
33          cin >> ans;
34      } while ((ans != 'n') && (ans != 'N'));
35      cout << "End of program.\n";
36      return 0;
37  }

38  void fillArray(int a[], int size, int& numberUsed)
39  <The rest of the definition of fillArray is given in Display 5.5>
40  int search(const int a[], int numberUsed, int target)
41  {
42      int index = 0;
43      bool found = false;
44      while ((!found) && (index < numberUsed))
45      if (target == a[index])
46          found = true;
47      else
48          index++;
```

Display 5.6
Tìm kiếm trên mảng (4/4)

```
49        if (found)
50            return index;
51        else
52            return -1;
53    }
```

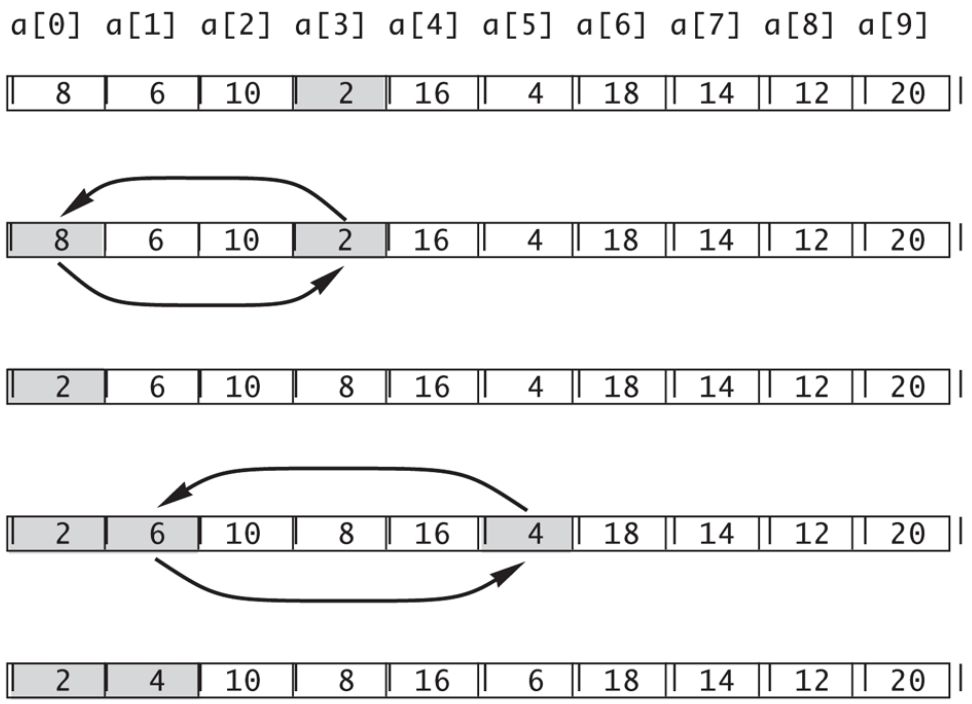
SAMPLE DIALOGUE

```
Enter up to 20 nonnegative whole numbers.
Mark the end of the list with a negative number.
10 20 30 40 50 60 70 80 -1
Enter a number to search for: 10
10 is stored in array position 0
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 40
40 is stored in array position 3
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 42
42 is not on the list.
Search again?(y/n followed by Return): n
End of program.
```

Sắp xếp một mảng:
Display 5.7 Sắp xếp lựa chọn

- Thuật toán sắp xếp lựa chọn

Display 5.7 Selection Sort



Ví dụ sắp xếp mảng:
Display 5.8 Sắp xếp mảng (1/4)

Display 5.8 Sorting an Array

```
1   //Tests the procedure sort.
2   #include <iostream>
3   using namespace std;

4   void fillArray(int a[], int size, int& numberUsed);
5   //Precondition: size is the declared size of the array a.
6   //Postcondition: numberUsed is the number of values stored in a.
7   //a[0] through a[numberUsed - 1] have been filled with
8   //nonnegative integers read from the keyboard.
9   void sort(int a[], int numberUsed);
10  //Precondition: numberUsed <= declared size of the array a.
```

(continued)

Ví dụ sắp xếp mảng:
Display 5.8 Sắp xếp mảng (2/4)

Display 5.8 Sorting an Array

```
11  //The array elements a[0] through a[numberUsed - 1] have values.
12  //Postcondition: The values of a[0] through a[numberUsed - 1] have
13  //been rearranged so that a[0] <= a[1] <= ... <= a[numberUsed - 1].

14  void swapValues(int& v1, int& v2);
15  //Interchanges the values of v1 and v2.

16  int indexOfSmallest(const int a[], int startIndex, int numberUsed);
17  //Precondition: 0 <= startIndex < numberUsed. Reference array elements
18  //have values. Returns the index i such that a[i] is the smallest of the
19  //values a[startIndex], a[startIndex + 1], ..., a[numberUsed - 1].

20  int main( )
21  {
22      cout << "This program sorts numbers from lowest to highest.\n";

23      int sampleArray[10], numberUsed;
24      fillArray(sampleArray, 10, numberUsed);
25      sort(sampleArray, numberUsed);

26      cout << "In sorted order the numbers are:\n";
27      for (int index = 0; index < numberUsed; index++)
28          cout << sampleArray[index] << " ";
29      cout << endl;

30      return 0;
31  }
```

Ví dụ sắp xếp mảng:
Display 5.8 Sắp xếp mảng (3/4)

```
32  void fillArray(int a[], int size, int& numberUsed)
33            <The rest of the definition of fillArray is given in Display 5.5.>

34  void sort(int a[], int numberUsed)
35  {
36      int indexOfNextSmallest;
37      for (int index = 0; index < numberUsed - 1; index++)
38      {//Place the correct value in a[index]:
39          indexOfNextSmallest =
40                        indexOfSmallest(a, index, numberUsed);
41          swapValues(a[index], a[indexOfNextSmallest]);
42          //a[0] <= a[1] <=...<= a[index] are the smallest of the original array
43          //elements. The rest of the elements are in the remaining positions.
44      }
45  }

46  void swapValues(int& v1, int& v2)
47  {
48      int temp;
49      temp = v1;
50      v1 = v2;
```

Ví dụ sắp xếp mảng:
Display 5.8 Sắp xếp mảng (4/4)

Display 5.8 Sorting an Array

```
51      v2 = temp;
52  }
53
54  int indexOfSmallest(const int a[], int startIndex, int numberUsed)
55  {
56      int min = a[startIndex],
57          indexOfMin = startIndex;
58      for (int index = startIndex + 1; index < numberUsed; index++)
59          if (a[index] < min)
60          {
61              min = a[index];
62              indexOfMin = index;
63              //min is the smallest of a[startIndex] through a[index]
64          }
65      return indexOfMin;
66  }
```

SAMPLE DIALOGUE

```
This program sorts numbers from lowest to highest.
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
80 30 50 70 60 90 20 30 40 -1
In sorted order the numbers are:
20 30 30 40 50 60 70 80 90
```

Mảng nhiều chiều

- Là mảng có nhiều hơn một chỉ số
  - char page[30][100];
    - Hai chỉ số. Đây là một mảng của các mảng một chiều.
    - Có thể minh họa như sau:
      page[0][0], page[0][1], …, page[0][99]
      page[1][0], page[1][1], …, page[1][99]
      …
      page[29][0], page[29][1], …, page[29][99]
- C++ cho phép số lượng chỉ số bất kì
  - Thường thì không quá hai

Tham số mảng nhiều chiều

- Tương tự với mảng một chiều
  - Bỏ qua kích thước chiều thứ nhất
    - Truyền vào dưới dạng một tham số riêng
  - Chỉ định kích thước chiều thứ hai
- Ví dụ:

```
void DisplayPage(const char p[][100], int sizeDimension1)
{
        for (int index1=0; index1<sizeDimension1; index1++)
        {
                for (int index2=0; index2 < 100; index2++)
                        cout << p[index1][index2];
                cout << endl;
        }
}
```

Tóm tắt 1

- Mảng là một tập hợp các phần tử dữ liệu cùng kiểu
- Các biến đánh chỉ số hợp thành mảng được dùng như các biến đơn khác
- Lệnh lặp for cho ta cách “tự nhiên”để duyệt mảng
- Lập trình viên có trách nhiệm kiểm soát miền giá trị của chỉ số
- Tham số mảng là “một kiểu mới”
  - Tương tự như truyền tham chiếu

Tóm tắt 2

- Các phần tử của mảng được lưu trữ tuần tự
  - Các phần bộ nhớ cận kề nhau
  - Chỉ có địa chỉ của phần tử đầu tiên được truyền vào hàm
- Hàm không đầy → Cần kiểm soát nhiều hơn
- Dùng từ khóa const với tham số mảng
  - Ngăn chặn việc biến đổi nội dung của mảng
- Mảng nhiều chiều
  - Tạo ra mảng của mảng

Chuẩn bị bài tới

- Đọc chương 5 giáo trình: struct và class