NGÔN NGỮ LẬP TRÌNH

Bài 2b: Mảng

Giảng viên: Lý Anh Tuấn

Email: tuanla@tlu.edu.vn

Nội dung

1. Giới thiệu mảng
   - Khai báo và tham chiếu mảng
   - Vòng lặp for và mảng
   - Mảng trong bộ nhớ
2. Mảng trong hàm
   - Mảng là tham số hàm, là giá trị trả về
3. Lập trình với mảng
   - Mảng được nhập giá trị một phần
   - Tìm kiếm
   - Sắp xếp
4. Mảng nhiều chiều

Giới thiệu mảng

- Định nghĩa mảng
  - Một tập dữ liệu có cùng kiểu
- Là kiểu dữ liệu kết hợp đầu tiên
  - int, float, double, char là các kiểu dữ liệu đơn giản
- Sử dụng cho các danh sách:
  - Điểm kiểm tra, nhiệt độ, tên, vân vân
  - Tránh khai báo nhiều biến đơn giản
  - Có thể điều khiển "danh sách" như một thực thể

Khai báo mảng

- Khai báo mảng → cấp phát bộ nhớ
  int score[5];
  - Khai báo mảng 5 số nguyên tên là score
  - Tương tự như khai báo 5 biến:

    int score[0], score[1], score[2], score[3], score[4]
- Các phần tử độc lập được gọi là
  - Biến có chỉ số
  - Các phần tử mảng
  - Giá trị trong ngoặc vuông được gọi là chỉ số, được đánh số từ 0 đến size - 1

Truy cập mảng

- Truy cập bằng cách sử dụng chỉ số
  - cout << score[3];
- Lưu ý hai trường hợp sử dụng cặp dấu ngoặc vuông
  - Trong khai báo, là kích thước của mảng
  - Ở chỗ khác, là một chỉ số
- Chỉ số không cần phải là một hằng nguyên
  - score[n+1] = 99; //nếu n là 2: score[3]

Sử dụng mảng

- Là kỹ thuật lưu trữ hiệu quả
- Có thể nêu ra các yêu cầu:
  - Thực hiện một công việc với biến chỉ số i trong đó i được tính toán bởi chương trình
  - Hiển thị tất cả các phần tử của mảng score
  - Nhập giá trị cho các phần tử của mảng score
  - Tìm giá trị lớn nhất trong mảng score
  - Tìm giá trị bé nhất trong mảng score

Ví dụ về mảng

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 về mảng

```
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

Vòng lặp for với mảng

- Vòng lặp đếm tự nhiên
  - Rất phù hợp với việc đếm các phần tử của một 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 thường gặp với mảng

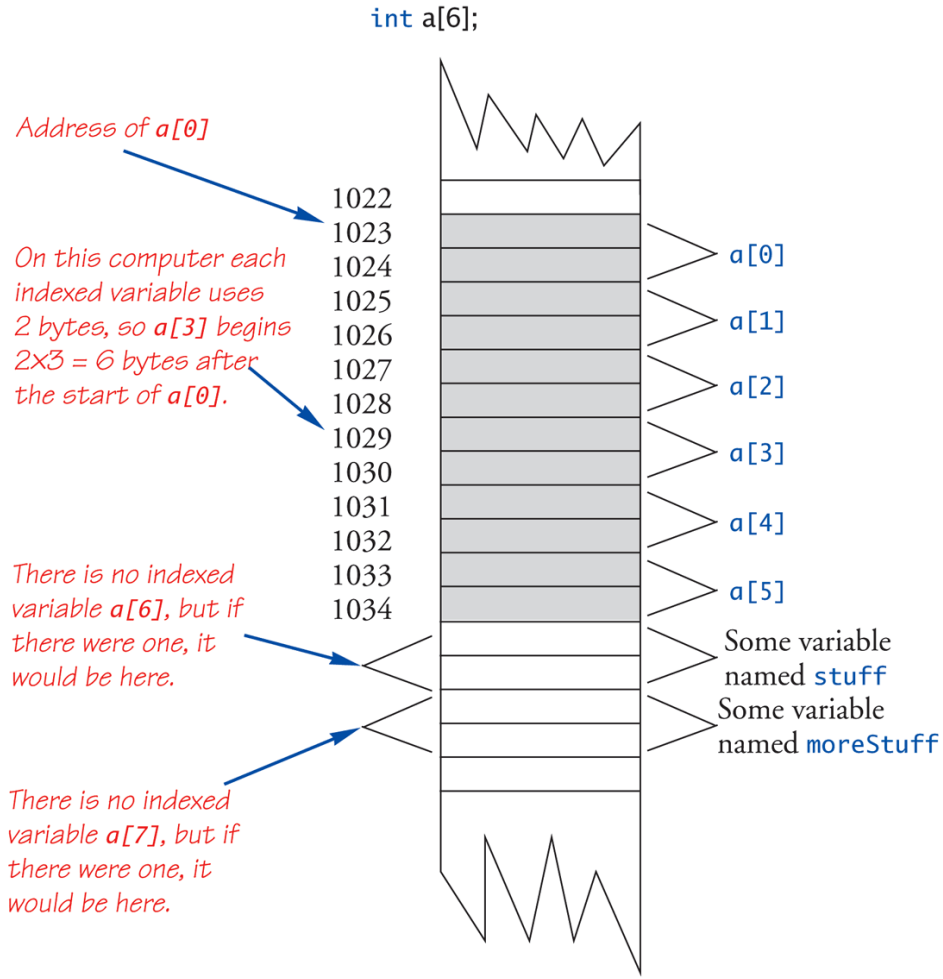
- Chỉ số mảng luôn bắt đầu bằng 0
- 0 là số đầu tiên với các nhà khoa học máy tính
- C++ cho phép vượt ra ngoài phạm vi mảng
  - Không dự đoán được kết quả
  - Trình biên dịch không phát hiện ra những lỗi này
- Lưu ý không vượt ra ngoài phạm vi mảng
- VD: Khai báo mảng
  - double temperature[24]; // 24 là kích thước mảng
  - Lỗi thường gặp:
    temperature[24] = 5; // chỉ số 24 là nằm ngoài phạm vi mảng

Mảng trong bộ nhớ

- Các biến đơn giản được cấp phát bộ nhớ bằng một địa chỉ
- Khai báo mảng cấp phát bộ nhớ cho toàn bộ mảng
- Được cấp phát liên tục
  - Các địa chỉ được cấp phát liên tiếp nhau
  - Cho phép tính địa chỉ của các phần tử mảng thông qua chỉ số

Mảng trong bộ nhớ

Display 5.2 **An Array in Memory**



Khởi tạo mảng

- Các biến đơn giản có thể được khởi tạo khi khai báo:

  int price = 0; // 0 là giá trị khởi tạo

- Mảng cũng có thể được khởi tạo:

  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 tập có ít giá trị hơn kích thước mảng:
  - Gán từ phần tử đầu tiên
  - Gán các phần tử còn lại bằng giá trị 0 trong kiểu dữ liệu của mảng
- Nếu không chỉ rõ kích thước mảng
  - Khai báo mảng có kích thước dựa trên số lượng giá trị khởi tạo
  - Ví dụ:

```
int b[] = {5, 12, 11};   // cấp phát mảng b kích thước
                            là 3
```

Mảng trong hàm

- Làm đối số của hàm
  - Biến có chỉ số:
    - Một phần tử mảng có thể là tham số hàm
  - Toàn bộ mảng:
    - Tất cả các phần tử mảng có thể được truyền như một thực thể
- Làm giá trị trả về từ hàm

Biến có chỉ số là đối số

- Vận hành như biến đơn giản thuộc kiểu dữ liệu của mảng
- Cho khai báo hàm sau đây:
  void myFunction(double par1);
- Và các khai báo:
  int i; double n, a[10];
- Có thể viết các lời gọi hàm sau đây:
  myFunction(i); // i được chuyển thành double
  myFunction(a[3]); // a[3] là double
  myFunction(n); // n là double

Toàn bộ mảng là đối số

- Tham số hình thức là toàn bộ mảng
  - Đối số được truyền trong lời gọi hàm là tên mảng
  - Được gọi là tham số mảng
- Cũng truyền kích thước của mảng
  - Thường được đặt là tham số thứ hai
  - Là tham số hình thức kiểu int

Ví dụ mảng là đối số

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 là đối số

- Xét ví dụ trước:
- Trong định nghĩa hàm main(), xét lời gọi sau:

  int score[5], numberOfScores = 5;
  fillup(score, numberOfScores);
  - Tham số thứ nhất là toàn bộ mảng
  - Tham số thứ hai là giá trị nguyên
- Lưu ý không có cặp dấu ngoặc vuông trong đối số mảng
- Sẽ thực sự truyền địa chỉ bắt đầu của mảng (địa chỉ của phần tử đầu tiên)

Bổ từ const cho tham số mảng

- Tham số mảng thực sự truyền địa chỉ của phần tử đầu tiên
  - Tương tự như truyền tham biến
- Hàm sau đó có thể sửa đổi mảng
  - Đôi khi chúng ta không mong muốn điều này
- Để bảo vệ nội dung mảng khỏi việc sửa đổi
  - Sử dụng bổ từ const trước tham số mảng

Mảng được nhập giá trị một phần

- Khó biết chính xác kích thước mảng cần thiết
- Phải khai báo kích thước lớn nhất có thể có
- Thêm biến cần thiết để theo dõi dữ liệu đúng trong mảng:
  - int numberUsed;
  - Theo dõi số phần tử hiện có trong mảng

Mảng được nhập giá trị một phần

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)

Mảng được nhập giá trị một phần

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   }
```

Mảng được nhập giá trị một phần

```
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  }
```

Mảng được nhập giá trị một phần

```
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  }
```

Mảng được nhập giá trị một phần

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
```

Tìm kiếm với mảng

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.
```

Tìm kiếm với mảng

```
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";
```

Tìm kiếm với mảng

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++;
```

Tìm kiếm với mảng

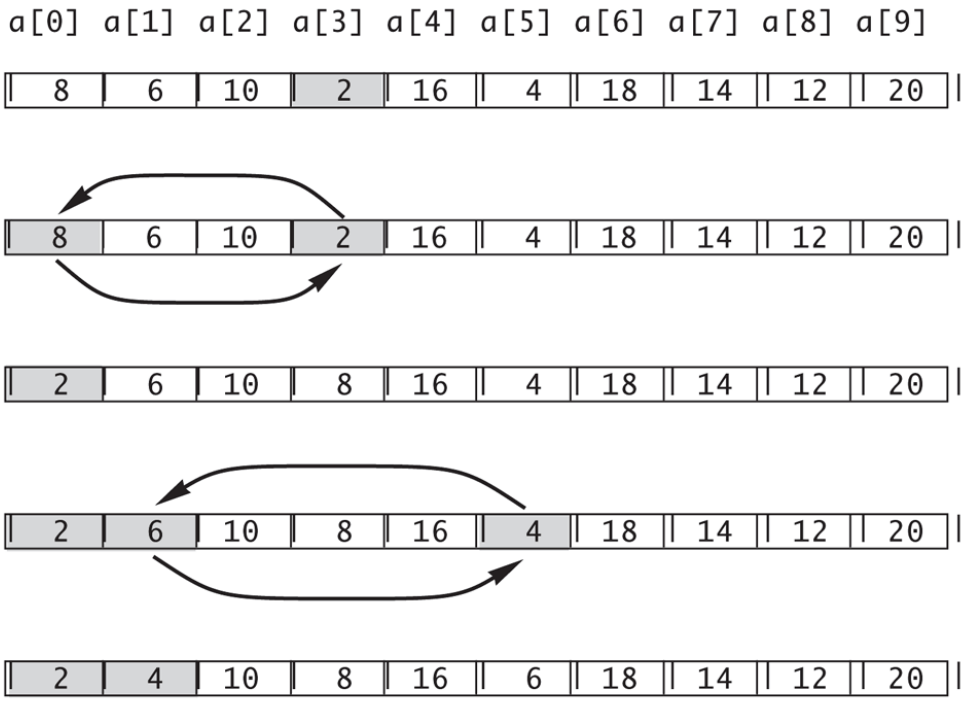
```
49        if (found)
50            return index;
51        else
52            return -1;
53    }
```

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 mảng

Display 5.7 **Selection Sort**



Sắp xếp mảng

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)

Sắp xếp mảng

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  }
```

Sắp xếp mảng

```
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;
```

Sắp xếp mảng

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

- Mảng có nhiều hơn một chỉ số
  - char page[30][100];
  - Có dạng:

    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 tùy ý
  - Thường không vượt quá hai

Tóm tắt

- Mảng là một tập dữ liệu cùng kiểu
- Biến có chỉ số của mảng được sử dụng giống như các biến đơn giản
- Vòng lặp for là cách tự nhiên để duyệt mảng
- Lưu ý không vượt ra ngoài phạm vi mảng
- Tham số mảng: tương tự như truyền tham biến
- Các phần tử mảng được lưu trữ liên tiếp
- Mảng được nhập giá trị một phần → cần biến theo dõi
- Tham số mảng hằng: ngăn sửa đổi nội dung mảng
- Mảng nhiều chiều: mảng của các mảng