DISPLAY 7.9 Partially Filled Array

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

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

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

16  int main( )
17  {
18      using namespace std;
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
23      cout << "Enter golf scores:\n";
24      fillArray(score, MAX_NUMBER_SCORES, numberUsed);
25      showDifference(score, numberUsed);
26      return 0;
27  }
28  //Uses iostream:
29  void fillArray(int a[], int size, int & numberUsed)
30  {
31      using namespace std;
32      cout << "Enter up to " << size << " nonnegative whole numbers.\n"
33           << "Mark the end of the list with a negative number.\n";
34      int next, index = 0;
35      cin >> next;
36      while ((next >= 0) && (index < size))
37      {
38          a[index] = next;
39          index++;
40          cin >> next;
41      }
42      numberUsed = index;
43  }
```

AND

(con

```
44    double computeAverage(const int a[], int numberUsed)
45    {
46        double total = 0;
47        for (int index = 0; index < numberUsed; index++)
48            total = total + a[index];
49        if (numberUsed> 0)
50        {
51            return (total/numberUsed);
52        }
53        else
54        {
55            using namespace std;
56            cout << "ERROR: number of elements is 0 in computeAverage.\n"
57                 << "computeAverage returns 0.\n";
58            return 0;
59        }
60    }
61    void showDifference(const int a[], int numberUsed)
62    {
63        using namespace std;
64        double average = computeAverage(a, numberUsed);
65        cout << "Average of the " << numberUsed
66             << " scores = " << average << endl
67             << "The scores are:\n";
68        for (int index = 0; index < numberUsed; index++)
69        cout << a[index] << " differs from average by "
70             << (a[index] - average) << endl;
71    }
```

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
```

DISPLAY 7.10 Searching an Array ..

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

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

14  int main( )
15  {
16      using namespace std;
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";

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  //Uses iostream:
```

```
39    void fillArray(int a[], int size, int& numberUsed)
    <The rest of the definition of fillArray is given in  Display 7.9.>
41    int search(const int a[], int numberUsed, int target)
42    {
43
44        int index = 0;
45        bool found = false;
46        while ((!found) && (index < numberUsed))
47            if (target == a[index])
48                found = true;
49            else
50                index++;
51
52        if (found)
53            return index;
54        else
55            return -1;
56    }
```

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.
```

```
3   #include <iostream>
4
5   void bubblesort(int arr[], int length);
6   //Precondition: length <= declared size of the array arr.
7   //The array elements arr[0] through a[length - 1] have values.
8   //Postcondition: The values of arr[0] through arr[length - 1] have
9   //been rearranged so that arr[0] <= a[1] <=  <= arr[length - 1].
10
11  int main()
12  {
13      using namespace std;
14      int a[] = {3, 10, 9, 2, 5, 1};
15
16      bubblesort(a, 6);
17      for (int i=0; i<6; i++)
18      {
19           cout << a[i] << " ";
20      }
21      cout << endl;
22      return 0;
23  }
24
25  void bubblesort(int arr[], int length)
26  {
27          // Bubble largest number toward the right
28          for (int i = length-1; j > 0; i--)
29                  for (int j = 0; j < i; j++)
30                          if (arr[j] > arr[j+1])
31                          {
32                                  // Swap the numbers
33                                  int temp = arr[j+1];
34                                  arr[j+1] = arr[j];
35                                  arr[j] = temp;
36                          }
37  }
```

DISPLAY 7.14 Two-Dimensional Array

```
1   //Reads quiz scores for each student into the two-dimensional array grade (but
2   //the input code is not shown in this display). Computes the average score
3   //for each student and the average score for each quiz. Displays the quiz scores
4   //and the averages.
5   #include <iostream>
6   #include <iomanip>
7   const int NUMBER_STUDENTS = 4, NUMBER_QUIZZES = 3;
8
9   void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[]);
10  //Precondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES
11  //are the dimensions of the array grade. Each of the indexed variables
12  //grade[stNum - 1, quizNum - 1] contains the score for student stNum on quiz
13  //quizNum.
14  //Postcondition: Each stAve[stNum - 1] contains the average for student
15  //number stuNum.
16
17  void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[]);
18  //Precondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES
19  //are the dimensions of the array grade. Each of the indexed variables
20  //grade[stNum - 1, quizNum - 1] contains the score for student stNum on quiz
21  //quizNum.
22  //Postcondition: Each quizAve[quizNum - 1] contains the average for quiz number
23  //quizNum.
24
25  void display(const int grade[][NUMBER_QUIZZES],
26  const double stAve[], const double quizAve[]);
27  //Precondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES are the
28  //dimensions of the array grade. Each of the indexed variables grade[stNum - 1,
29  //quizNum - 1] contains the score for student stNum on quiz quizNum. Each
30  //stAve[stNum - 1] contains the average for student stuNum. Each
31  //quizAve[quizNum - 1] contains the average for quiz number quizNum.
32  //Postcondition: All the data in grade, stAve, and quizAve has been output.
33
34  int main( )
35  {
36      using namespace std;
37      int grade[NUMBER_STUDENTS][NUMBER_QUIZZES];
38      double stAve[NUMBER_STUDENTS];
39      double quizAve[NUMBER_QUIZZES];
40
```

```
41          computeStAve(grade, stAve);
42          computeQuizAve(grade, quizAve);
43          display(grade, stAve, quizAve);
44          return 0;
45      }
46      void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[])
47      {
48          for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
49          {//Process one stNum:
50              double sum = 0;
51              for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
52                  sum = sum + grade[stNum - 1][quizNum - 1];
53              //sum contains the sum of the quiz scores for student number stNum.
54              stAve[stNum - 1] = sum/NUMBER_QUIZZES;
55              //Average for student stNum is the value of stAve[stNum-1]
56          }
57      }
58
59
60      void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[])
61      {
62          for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
63          {//Process one quiz (for all students):
64              double sum = 0;
65              for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
66                  sum = sum + grade[stNum - 1][quizNum - 1];
67              //sum contains the sum of all student scores on quiz number quizNum.
68              quizAve[quizNum - 1] = sum/NUMBER_STUDENTS;
69              //Average for quiz quizNum is the value of quizAve[quizNum - 1]
70          }
71      }
72
73
74      //Uses iostream and iomanip:
75      void display(const int grade[][NUMBER_QUIZZES],
76          const double stAve[], const double quizAve[])
77      {
78          using namespace std;
79          cout.setf(ios::fixed);
80          cout.setf(ios::showpoint);
81          cout.precision(1);
82          cout << setw(10) << "Student"
83               << setw(5) << "Ave"
84               << setw(15) << "Quizzes\n";
85          for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
86          {//Display for one stNum:
```