```cpp
// Chapter 15 - Assignment 1, Analysis of Sorting Algorithms
// This Program uses a class named AbstractSort as a framework
// for sorting algorithms that sort by comparing pairs of array
entries.
// To obtain a concrete class for sorting, one first subclasses
// the AbstractSort class and then overrides the pure virtual
function
// sort. The sort() function must call the compare(int, int) function
// to compare pairs of array entries.

#include <iostream>
#include <algorithm>
#include <fstream>
#include <cstdlib> // For random
#include <time.h>  // time
#include <conio.h>
using namespace std;

class AbstractSort
{
public:
    virtual void sort(int arr[], int size) = 0;
    int getComparisonCount()
    {
        return comparisonCount;
    }
    void resetComparisonCount()
```

```
    {
        comparisonCount = 0;
    }
protected:
    int compare(int x, int y);
private:
    int comparisonCount;
};

//*************************************************
//          AbstractSort::compare                 *
//   Returns -1, 0, or 1 a la strcmp              *
// This also keeps track of the number of comparisons *
// performed.                                     *
//*************************************************
int AbstractSort::compare(int x, int y)
{
    comparisonCount++;
    return x - y;
}

// Derived class
class MaxSort : public AbstractSort
{
public:
    void sort(int arr[], int size);
};
```

```
53
54  //*************************************************************
55  //                  MaxSort::sort                              *
56  // Sort the given array with the given number of elements.    *
57  //*************************************************************
58  void MaxSort::sort(int arr[], int size)
59  {
60      resetComparisonCount();
61      for (int k = size - 1; k >= 1; k--)
62      {
63          // Arr[k+1..size-1] contains the biggest items in the array
    is sorted order
64          // Find biggest item in arr[0..k] and move it to a[k]
65
66          int indexOfLargest = 0;
67          for (int ix = 1; ix <= k; ix++)
68              if (compare(arr[ix], arr[indexOfLargest]) > 0)
69                  indexOfLargest = ix;
70
71          swap(arr[indexOfLargest], arr[k]);
72      }
73  }
74
75  int main()
76  {
77      const int MAX_SIZE = 1000;
78      int arr[MAX_SIZE];
```

```
79      int size = 0;
80      int val;
81      int temp;
82      ifstream inputFile;
83
84      // Explain the program
85      cout << "This program keeps track of the number of comparisons
    required to\n";
86      cout << "to sort an array found in file Project 7 data
    file.txt.\n";
87      cout << "\n\nPress <Enter> to continue";
88      _getch();
89
90      // Get the size of the array
91      inputFile.open("Project 7 data file.txt");
92      while (inputFile >> val)
93          size++;
94      inputFile.close();
95
96      if (size > MAX_SIZE)
97      {
98          cout << "The size of the array must be no greater than
    1000.";
99          exit(1);
100     }
101
102     // Initialize random number generator
```

```
103     srand(time(0));
104
105     // Open the input file
106     inputFile.open("Project 7 data file.txt");
107
108     // Fill the array with random numbers
109     for (int k = 0; k < size; k++)
110         //arr[k] = rand() % 1000;
111         inputFile >> arr[k];
112
113     // Output array to be sorted
114     cout << "Array to be sorted is: \n";
115     for (int k = 0; k < size; k++)
116         cout << arr[k] << "  ";
117
118     // Sort and output results
119     MaxSort maxSort;
120     maxSort.sort(arr, size);
121     cout << "\nThe sorted array is: \n";
122     for (int k = 0; k < size; k++)
123         cout << arr[k] << "  ";
124     cout << "\nNumber of comparisons performed is: " << maxSort.
    getComparisonCount();
125     cout << "\n\nPress <Enter> to exit";
126     _getch();
127     return 0;
128 }
```