#include<iostream>  
#include "input.h"  
using namespace std;  
  
void challenge3();  
  
int main()  
{  
   do  
    {  
        system("cls");  
        cout << "\n\tCMPR121: Chapter 9 Pointer - assignments by your name";  
        cout << "\n\t" << string(80, char(205));  
        cout << "\n\t 3. Drop the Lowest Score";  
        cout << "\n\t 6. Case Study Modification #1";  
        cout << "\n\t 7. Case Study Modification #2";  
        cout << "\n\t 9. Median Function";  
        cout << "\n\t10. Reverse Array";  
        cout << "\n\t11. Array Expander";  
        cout << "\n\t12. Element Shifter";  
  
        cout << "\n\t" << string(80, char(196));  
        cout << "\n\t 0. Exit";  
        cout << "\n\t" << string(80, char(205));  
        cout << "\n";  
        switch (inputInteger("\tOption: ", 0, 12))  
        {  
        case 0: exit(0);  
        case 3: challenge3(); break;  
       case 6: challenge6(); break;  
       case 7: challenge7(); break;  
       case 9: challenge9(); break;  
       case 10: challenge10(); break;  
       case 11: challenge11(); break;  
       case 12: challenge12(); break;  
  
        default: cout << "\n\tERROR: Invalid option.\n";  
        }  
  
        system("pause");  
    } while (true);  
    return 0;  
}  
void challenge3()  
{  
    int size = inputInteger("\nEnter the size of the dynamic allocated array: ", true);  
    double\* scores = new double[size];  
    srand(time(0));  
  
    double sum = 0.0;  
  
    for (int i = 0; i < size; i++)  
    {  
        \*(scores+i) = rand() % 100 + 1;  
        cout << \*(scores + i) << '\n';  
        sum += \*(scores + i);  
    }  
  
    // sort the array   
    // recalcuLate AVERAGE without the lowest score  
  
    cout << "\nAverage: " << sum / size << "\n";  
  
    delete[] scores;  
}

**Instructions:**

1. Create a MS Visual Studio solution named **Chapter9**.
2. Create projects according to the assigned problems.
3. Use techniques and experience that you have obtained from previous chapters and this chapter.
4. Code and test your programs. They **MUST** be completed without syntax, logic and run-time errors.
5. Include comments of your name, date, and brief descriptions of all functions.
6. Compress C**hapter9**folder into **ONE** zipped file.
7. Submit or re-submit your zipped file before its due date&time.

**View Rubric**

3 --Drop Lowest Score

Modify Problem 2 above so the lowest test score is dropped. This score should not be included in the calculation of the average.

6--Case Study Modification #1   
Modify Program 9-19 (the United Cause case study program – source code at end of document) so it can be used with any set of donations. The program should dynamically allocate the donations array and ask the user to input its values.

7--Case Study Modification #2 (end of page)

Modify Program 9-19 (the United Cause case study program – source code at end of document) so the arrptr array is sorted in descending order instead of ascending order.

9--Median Function

In statistics, when a set of values is sorted in ascending or descending order, its median is the middle value. If the set contains an even number of values, the median is the mean, or average, of the two middle values. Write a function that accepts as arguments the following:

An array of integers

An integer that indicates the number of elements in the array

The function should determine the median of the array. This value should be returned as a double. (Assume the values in the array are already sorted.)

Demonstrate your pointer prowess by using pointer notation instead of array notation in this function.

10-Reverse Array

Write a function that accepts an int array and the array’s size as arguments. The function should create a copy of the array, except that the element values should be reversed in the copy. The function should return a pointer to the new array. Demonstrate the function in a complete program.

11-- Array Expander

Write a function that accepts an int array and the array’s size as arguments. The function should create a new array that is twice the size of the argument array. The function should copy the contents of the argument array to the new array and initialize the unused elements of the second array with 0. The function should return a pointer to the new array.

12 -- Element Shifter

Write a function that accepts an int array and the array’s size as arguments. The function should create a new array that is one element larger than the argument array. The first element of the new array should be set to 0. Element 0 of the argument array should be copied to element 1 of the new array, element 1 of the argument array should be copied to element 2 of the new array, and so forth. The function should return a pointer to the new array.

Program 9-19

1 // This program shows the donations made to the United Cause

2 // by the employees of CK Graphics, Inc. It displays

3 // the donations in order from lowest to highest

4 // and in the original order they were received.

5 #include <iostream>

6 using namespace std;

7

8 // Function prototypes

9 void arrSelectSort(int \*[], int);

10 void showArray(const int [], int);

11 void showArrPtr(int \*[], int);

12

13 int main()

14 {

15 const int NUM\_DONATIONS = 15; // Number of donations

16

17 // An array containing the donation amounts.

18 int donations[NUM\_DONATIONS] = { 5, 100, 5, 25, 10,

19 5, 25, 5, 5, 100,

20 10, 15, 10, 5, 10 };

21

22 // An array of pointers to int.

23 int \*arrPtr[NUM\_DONATIONS] = { nullptr, nullptr, nullptr, nullptr, nullptr,

24 nullptr, nullptr, nullptr, nullptr, nullptr,

25 nullptr, nullptr, nullptr, nullptr, nullptr};

26

27 // Each element of arrPtr is a pointer to int. Make each

28 // element point to an element in the donations array.

29 for (int count = 0; count < NUM\_DONATIONS; count++)

30 arrPtr[count] = &donations[count];

31

32 // Sort the elements of the array of pointers.

33 arrSelectSort(arrPtr, NUM\_DONATIONS);

34

35 // Display the donations using the array of pointers. This

36 // will display them in sorted order.

37 cout << "The donations, sorted in ascending order, are: \n";

38 showArrPtr(arrPtr, NUM\_DONATIONS);

39

40 // Display the donations in their original order.

41 cout << "The donations, in their original order, are: \n";

42 showArray(donations, NUM\_DONATIONS);

43 return 0;

44 }

45

46 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

47 // Definition of function arrSelectSort. \*

48 // This function performs an ascending order selection sort on \*

49 // arr, which is an array of pointers. Each element of array \*

50 // points to an element of a second array. After the sort, \*

51 // arr will point to the elements of the second array in \*

52 // ascending order. \*

53 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

54

55 void arrSelectSort(int \*arr[], int size)

56 {

57 int startScan, minIndex;

58 int \*minElem;

59

60 for (startScan = 0; startScan < (size − 1); startScan++)

61 {

62 minIndex = startScan;

63 minElem = arr[startScan];

64 for(int index = startScan + 1; index < size; index++)

65 {

66 if (\*(arr[index]) < \*minElem)

67 {

68 minElem = arr[index];

69 minIndex = index;

70 }

71 }

72 arr[minIndex] = arr[startScan];

73 arr[startScan] = minElem;

74 }

75 }

76

77 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

78 // Definition of function showArray. \*

79 // This function displays the contents of arr. size is the \*

80 // number of elements. \*

81 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

82

83 void showArray(const int arr[], int size)

84 {

85 for (int count = 0; count < size; count++)

86 cout << arr[count] << " ";

87 cout << endl;

88 }

89

90 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

91 // Definition of function showArrPtr. \*

92 // This function displays the contents of the array pointed to \*

93 // by arr. size is the number of elements. \*

94 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

95

96 void showArrPtr(int \*arr[], int size)

97 {

98 for (int count = 0; count < size; count++)

99 cout << \*(arr[count]) << " ";

100 cout << endl;

101 }

Program Output

The donations, sorted in ascending order, are:

5 5 5 5 5 5 10 10 10 10 15 25 25 100 100

The donations, in their original order, are:

5 100 5 25 10 5 25 5 5 100 10 15 10 5 10