**Programming Fundamentals**

**Lab 9**

**Submitted To:**

Mr. Dilshad Sabir

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Finding Minimum and Maximum Values in an Array**

**Your task is to perform some functions on integer arrays. Specifically you will write a C program that does the following:**

**1. Declare an array of size 20.**

**2. Initialize the array with random values (use loop, and rand() function).**

**3. Print all the elements in the array.**

**4. Print all the elements in the array in the reverse order.**

**5. Print the array such that every Nth element gets printed. N is user input.**

**Program:** In this program, first the array of size 20 has been declared. Array has been initialized using function **“rand( )”** such that the values range between 0 and 99. Then a for loop prints the array. The next for loop prints the array in reverse order. After that the program asks the value of **N** from the user and prints every **Nth** element of the array.

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 **int** main()

5 {

6 **int** my\_array[20], N;

7

8 **for**(**int** i =0; i < 20; i++)

9 {

10 my\_array[i] = rand()%100;

11 }

12

13 printf("\nThis is the array:\n\n");

14 **for**(**int** i=0; i < 20; i++)

15 printf("%d ",my\_array[i]);

16

17 printf("\nThis is the array in reverse order:\n\n");

18 **for**(**int** i=19; i >= 0; i--)

19 printf("%d ",my\_array[i]);

20

21 printf("\n\nEnter the vaalue of N: ");

22 scanf("%d",&N);

23

24 printf("\nThis is the array with Nth element printed:\n\n");

25 **for**(**int** i=N-1; i < 20; i=i+N)

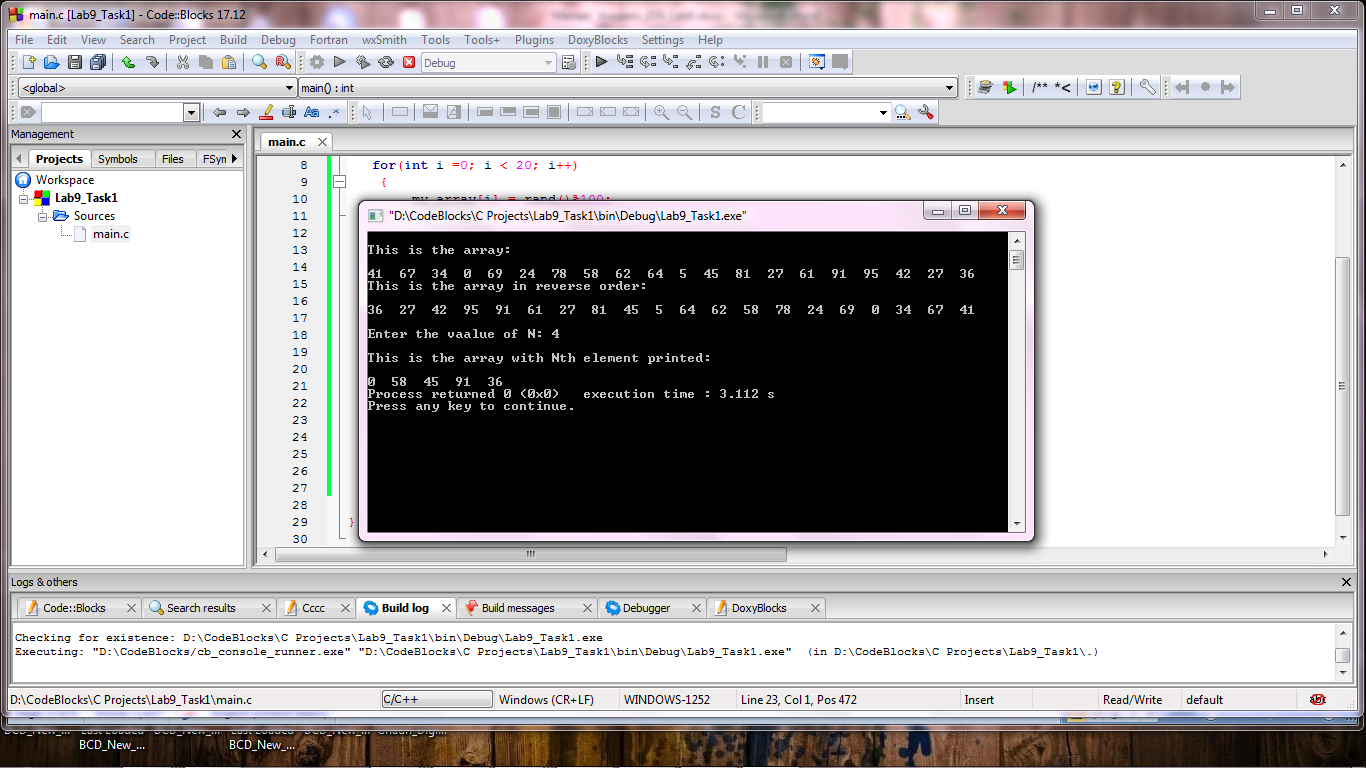
26 printf("%d ",my\_array[i]);

27

28 **return** 0;

29 }

**Output:**



**Task 2:**

**Implementing Selection Sort**

**You are given a C program in Code Listing 1, that does the following:**

**1. Declares an integer array with 50 elements (not initialized).**

**2. Populates the array with random positive numbers. (Uses a loop and rand() function)**

**3. Calls the function ‘int find\_max(int \* ptr\_array, int size)’ and prints the**

**value and index of the largest number.**

**Code Listing 1**

#include <stdio.h>

#include <stdlib.h>

#define ASCENDING 0

#define DESCENDING 1

#define ARRAY\_SIZE 50

int find\_max(int \* ptr\_array, int size);

int find\_min(int \* ptr\_array, int size);

void selection\_sort(int \* ptr\_array, int size, int order);

int main()

{

int num\_array[ARRAY\_SIZE]; /// Declare an integer array

int \* ptr\_ar = &num\_array[0]; /// A pointer to the start of the array

for(int i =0; i < ARRAY\_SIZE; i++)

{

num\_array[i] = rand()%100; /// Initialize the array with random numbers in range 0 to 99

printf("%d ", num\_array[i]);/// and print it.

}

int mx\_idx = find\_max(ptr\_ar, ARRAY\_SIZE); /// Print the maximum value and its index

printf("\nThe maximum number is %d at index %d \n", num\_array[mx\_idx], mx\_idx);

int mn\_idx = find\_min(ptr\_ar, ARRAY\_SIZE); /// Print the minimum value and its index

printf("\nThe minimum number is %d at index %d \n", num\_array[mn\_idx], mn\_idx);

selection\_sort(num\_array, ARRAY\_SIZE, ASCENDING); /// Sort the array using Selection Sort

for(int i=0; i<ARRAY\_SIZE; i++) /// Print the sorted array

printf("%d ", num\_array[i]);

return 0;

}

int find\_max(int \* ptr\_array, int size)

{

int max\_val = 0;

int max\_idx = 0;

for(int i=0; i<size; i++)

{

if(\*(ptr\_array+i) > max\_val)

{

max\_val = \*(ptr\_array+i);

max\_idx = i;

}

}

return(max\_idx);

}

int find\_min(int \* ptr\_array, int size)

{

return 0;

}

void selection\_sort(int \* ptr\_array, int size, int order)

{

}

**Program:** In this program, **“find\_min”** and **“selection\_sort”** are completed as per instructions in the reading material. In **“find\_min”** function **“min\_val”** is initialized with the value 99 which is the maximum possible value among all random value. By means of a for loop each value in array is compared to the value stored in **“min\_val”** and if it is less than **“min\_val”** then **“min\_val”** is assigned that value and its index is stored in **“min\_idx”**. Finally the index of minimum value is returned to the main function and value at that index is displayed as output on the console.

In **“selection\_sort”** function, first a selection is made depending on the value of **“order”** parameter. If sorting in ascending order is required then a loop is started which finds the maximum value of the unsorted list using the **“max\_find”** function then this value is swapped with the topmost element of the sorted list. If sorting in descending order is required then a loop is started which finds the minimum value of the unsorted list using the **“min\_find”** function then this value is swapped with the topmost element of the sorted list. Henceforth, the sorted array is printed in the **“main()”** function as output on the console.

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 #define ASCENDING 0

5 #define DESCENDING 1

6

7 #define ARRAY\_SIZE 50

8

9 **int** find\_max(**int** \* ptr\_array, **int** size);

10 **int** find\_min(**int** \* ptr\_array, **int** size);

11 **void** selection\_sort(**int** \* ptr\_array, **int** size, **int** order);

12

13 **int** main()

14 {

15

16 **int** num\_array[ARRAY\_SIZE]; **/// Declare an integer array**

17

18 **int** \* ptr\_ar = &num\_array[0]; **/// A pointer to the start of the array**

19

20 **for**(**int** i =0; i < ARRAY\_SIZE; i++)

21 {

22 num\_array[i] = rand()%100; **/// Initialize the array with random numbers in range 0 to 99**

23 printf("%d ", num\_array[i]);**/// and print it.**

24 }

25

26 **int** mx\_idx = find\_max(ptr\_ar, ARRAY\_SIZE); **/// Print the maximum value and its index**

27 printf("\nThe maximum number is %d at index %d \n", num\_array[mx\_idx], mx\_idx);

28

29 **int** mn\_idx = find\_min(ptr\_ar, ARRAY\_SIZE); **/// Print the minimum value and its index**

30 printf("\nThe minimum number is %d at index %d \n", num\_array[mn\_idx], mn\_idx);

31

32 selection\_sort(num\_array, ARRAY\_SIZE, DESCENDING); **/// Sort the array using Selection Sort**

33 **for**(**int** i=0; i<ARRAY\_SIZE; i++) **/// Print the sorted array**

34 printf("%d ", num\_array[i]);

35

36 **return** 0;

37 }

38

39 **int** find\_max(**int** \* ptr\_array, **int** size)

40 {

41 **int** max\_val = 0;

42 **int** max\_idx = 0;

43

44 **for**(**int** i=0; i<size; i++)

45 {

46

47 **if**(\*(ptr\_array+i) > max\_val)

48 {

49 max\_val = \*(ptr\_array+i);

50 max\_idx = i;

51 }

52 }

53

54 **return**(max\_idx);

55

56 }

57 **int** find\_min(**int** \* ptr\_array, **int** size)

58 {

59 **int** min\_val = 99;

60 **int** min\_idx = 0;

61

62 **for**(**int** i=0; i<size; i++)

63 {

64

65 **if**(\*(ptr\_array+i) < min\_val)

66 {

67 min\_val = \*(ptr\_array+i);

68 min\_idx = i;

69 }

70 }

71

72 **return**(min\_idx);

73 }

74 **void** selection\_sort(**int** \* ptr\_array, **int** size, **int** order)

75 {

76 **if** (order == ASCENDING)

77 {

78 **int** temp, max, j;

79

80 **for**(j=size-1; j >0; j--)

81 {

82 max = find\_max(ptr\_array, j+1);

83 temp = \*(ptr\_array+max);

84 \*(ptr\_array+max) = \*(ptr\_array+j);

85 \*(ptr\_array+j) = temp;

86 }

87 }

88

89 **else if** (order == DESCENDING)

90 {

91 **int** temp, min, j;

92 **for**(j=size-1; j>=0; j--)

93 {

94 min = find\_min(ptr\_array, j+1);

95 temp = \*(ptr\_array+min);

96 \*(ptr\_array+min) = \*(ptr\_array+j);

97 \*(ptr\_array+j) = temp;

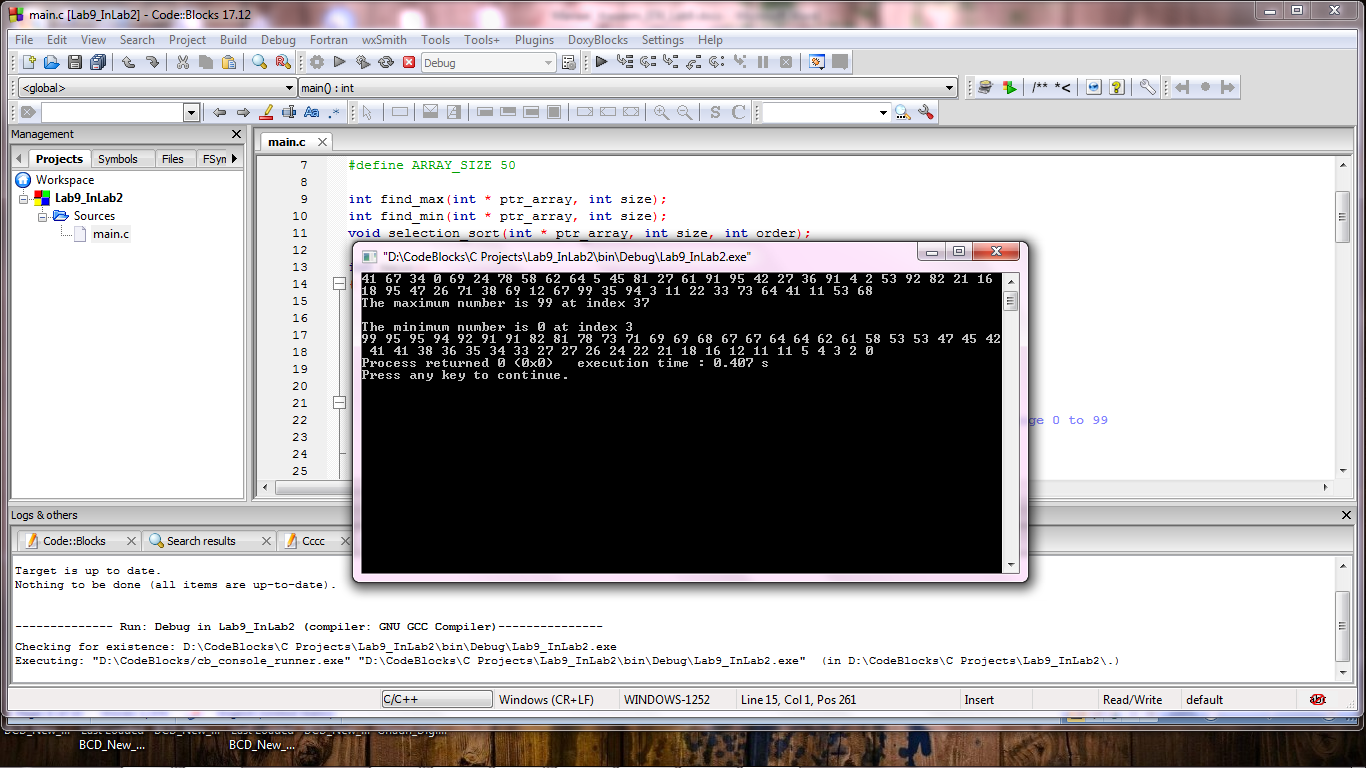
98 }

99 }

100 }

101

**Output:**



**Post Lab:**

**Implement Insertion Sort Algorithm**

**Your second task is to implement the Insertion Sort algorithm by making a function with the following prototype;**

**void insertion\_sort(int \* ptr\_array, int size, int order);**

**This function takes as input a pointer to the start of the array, and the array size and sorts it in-place.**

**The last input to the function is the sorting order (0 for ascending and 1 for descending).**

**Program:** In this program, a function **“insertion\_sort”** is defined. In **“insertion\_sort”** function, first a selection is made depending on the value of **“order”** parameter. Depending upon this selection, the first pairs of numbers are compared with each other and hence sorted. Then first element of the unsorted list is compared with the elements of the sorted list and inserted in the sort order in it. This process continues until all elements of the array are arranged in the defined order. The elements alter their positions by swapping the values. Hence, the sorted array is printed in the **“main( )”** function as output on the console.

1 #include <stdio.h>

2 #include <stdlib.h>

3 #define ASCENDING 0

4 #define DESCENDING 1

5 #define ARRAY\_SIZE 50

6

7 **void** insertion\_sort(**int** \* ptr\_array, **int** size, **int** order);

8

9 **int** main()

10 {

11 **int** num\_array[ARRAY\_SIZE]; **/// Declare an integer array**

12 **int** \* ptr\_array = &num\_array[0]; **/// A pointer to the start of the array**

13

14 printf("\n");

15

16 **for**(**int** i =0; i < ARRAY\_SIZE; i++)

17 {

18 num\_array[i] = rand()%100; **/// Initialize the array with random numbers in range 0 to 99**

19 printf("%d ", num\_array[i]);**/// and print it.**

20 }

21 insertion\_sort(num\_array, ARRAY\_SIZE, ASCENDING); **/// Sort the array using Insertion Sort**

22 printf("\n\nThe sorted array is: \n\n");

23 **for**(**int** i=0; i<ARRAY\_SIZE; i++) **/// Print the sorted array**

24 printf("%d ", num\_array[i]);

25

26

27 **return** 0;

28 }

29

30 **void** insertion\_sort(**int** \* ptr\_array, **int** size, **int** order)

31 {

32 **if** (order == ASCENDING)

33 {

34 **int** i, j, ins\_element;

35

36 **for** (i=1; i<size; i++)

37 {

38 ins\_element= \*(ptr\_array+i);

39 j=i-1;

40 **while** (j>=0 && \*(ptr\_array+j)>ins\_element)

41 {

42 \*(ptr\_array+j+1)=\*(ptr\_array+j);

43 j=j-1;

44 }

45 \*(ptr\_array+j+1) = ins\_element;

46 }

47 }

48

49 **if** (order == DESCENDING)

50 {

51 **int** i, j, ins\_element;

52

53 **for** (i=1; i<size; i++)

54 {

55 ins\_element= \*(ptr\_array+i);

56 j=i-1;

57 **while** (j>=0 && \*(ptr\_array+j)<ins\_element)

58 {

59 \*(ptr\_array+j+1)=\*(ptr\_array+j);

60 j=j-1;

61 }

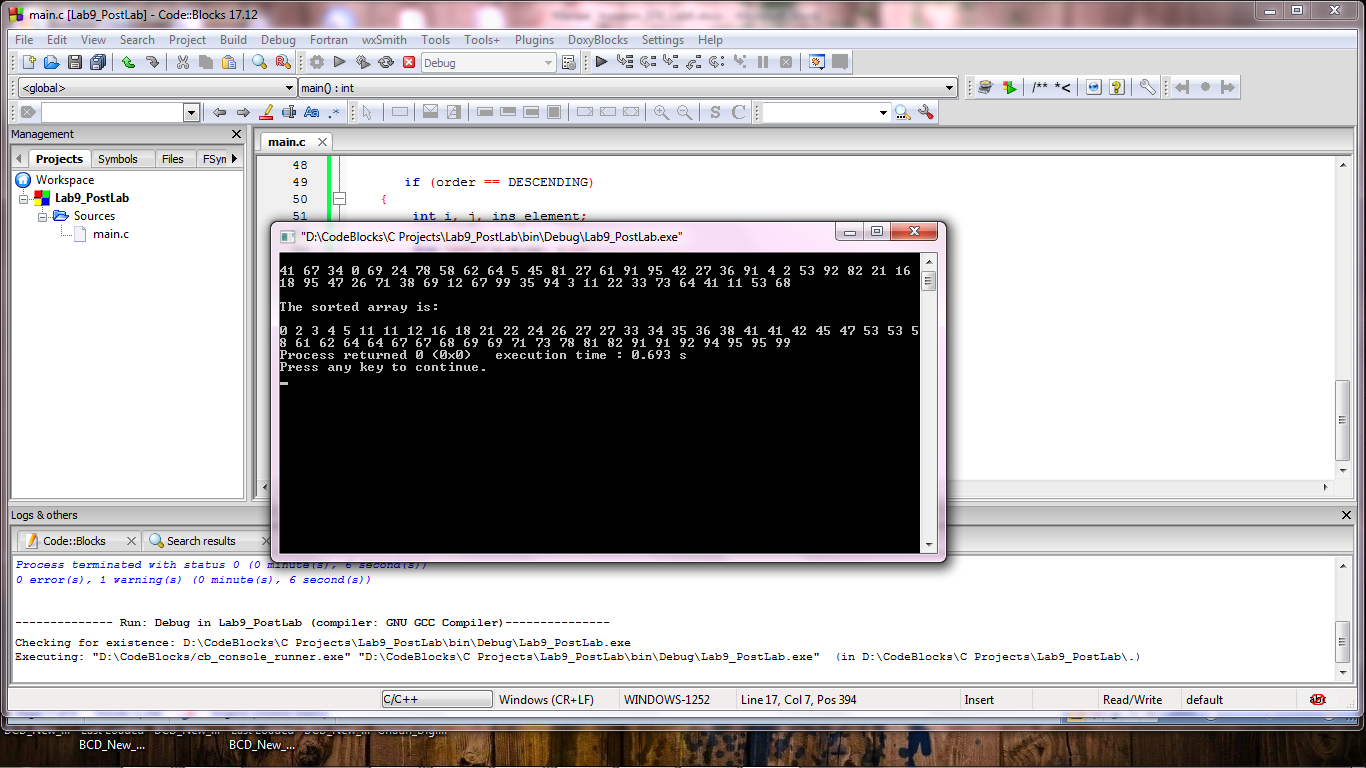
62 \*(ptr\_array+j+1) = ins\_element;

63 }

64 }

65 }

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**