

## Assignment – 2 (Array Concepts)

1. An **array** is a data structure containing a collection of values or variables. The simplest type of array is a linear array or one-dimensional array. An array can be defined in C with the following syntax:  
`int Arr[5] = {12, 56, 34, 78, 100};`

The screenshot shows a C++ IDE with a project named 'array.cpp'. The code defines an array of 5 integers: {12, 56, 34, 78, 100}. It then iterates through the array to find the largest element, which is 100. The program is executed, and the output is displayed in a console window.

```
1 #include<stdio.h>
2 int main(){
3     int arr[5] = {12,56,34,78,100};
4     int largest = arr[0];
5     for(int i = 1; i < 5; i++){
6         if(arr[i] > largest){
7             largest = arr[i];
8         }
9     }
10    printf("the largest element %d",largest);
11    return 0;
12 }
13 }
```

the largest element 100

-----

Process exited after 0.04016 seconds with return value 0

Press any key to continue . . .

Compilation result

-----

- Errors: 0

- Warnings: 0

- Output Filename:

- Output Size: 127

- Compilation Time:

## 2. Problem Description

We have to write a program in C such that the program will read the elements of a one-dimensional array, then compares the elements and finds which are the largest two elements in a given array.

### Expected Input and Output

#### 1. Finding Largest 2 numbers in an array with unique elements:

If we are entering 5 elements (N = 5), with array element values as 2,4,5,8 and 7 then,

**The FIRST LARGEST = 8**

**THE SECOND LARGEST = 7**

#### 2. Finding Largest 2 numbers in an array with recurring elements:

If we are entering 6 elements (N = 6), with array element values as 2,1,1,2,1 and 2 then,

**The FIRST LARGEST = 2**

**THE SECOND LARGEST = 1**

The screenshot displays a C++ IDE with the source code for a program that finds the two largest numbers in an array. The code is as follows:

```
1 int n,i,largest1,largest2,arr[500];
2 printf("enter the number of elements in array");
3 scanf("%d",&n);
4 printf("enter the element of the array");
5 for(i=0;i<n;i++){
6     scanf("%d",&arr[i]);
7 }
8 largest1 = arr[0];
9 largest2 = arr[1];
10 if(largest1 < largest2){
11     int temp = largest1;
12     largest1 = largest2;
13     largest2 = temp;
14 }
15 for(i=2;i<n;i++){
16     if(arr[i] > largest1){
17         largest2 = largest1;
18         largest1 = arr[i];
19     }
20     else if ( arr[i] > largest2 && arr[i] != largest1){
21         largest2 = arr[i];
22     }
23 }
24 printf("the first largest number %d\n",largest1);
25 printf("the first largest number %d\n",largest2);
26 return 0;
```

The execution output shows the program running with the following input and output:

```
enter the number of elements in array10,11,12,13,14
enter the element of the arraythe first largest number 1977226368
the first largest number 1976463872

Process exited after 10.9 seconds with return value 0
Press any key to continue . . .
```

The IDE also shows the compilation results, indicating that the program compiled successfully with no errors or warnings.

### 3. C Program finds second largest & smallest elements in an Array.

#### Problem Description

The program will implement a one dimensional array and sort the array in descending order. Then it finds the second largest and smallest element in an array and also find the average of these two array elements. Later it checks if the resultant average number is present in a given array. If found, display appropriate message.

```
1 #include <stdio.h>
2
3 #define MAX_SIZE 100 // Maximum array size
4
5 int main()
6 {
7     int arr[MAX_SIZE], n, i, j, temp;
8     int largest, second_largest, smallest, second_smallest;
9     float avg;
10
11     // Read array size and elements from user
12     printf("Enter size of array: ");
13     scanf("%d", &n);
14
15     printf("Enter %d elements in array: ", n);
16     for(i=0; i<n; i++)
17         scanf("%d", &arr[i]);
18
19     // Sort array in descending order
20     for(i=0; i<n; i++)
21     {
22         for(j=i+1; j<n; j++)
23         {
24             if(arr[i] < arr[j])
25             {
26                 temp = arr[i];
27                 arr[i] = arr[j];
28                 arr[j] = temp;
29             }
30         }
31     }
32 }
```

Enter size of array: 12,13,14  
Enter 12 elements in array:  
780140736.00 is not present in the array.  
-----  
Process exited after 19.48 seconds with return value 0  
Press any key to continue . . .

Compilation results...  
- Errors: 0  
- Warnings: 0  
- Output Filename: C:\Users\akhi1\OneDrive\Desktop\c\3.exe  
- Output Size: 129.1015625 KiB  
- Compilation Time: 0.20s

#### 4. C Program To Find Maximum Difference Between Two Elements in an Array

Example:

Consider the Following Array

```
int array[] = {10, 15, 90, 200, 110};
```

Output:

Maximum difference is 190

That is  $200 - 10 = 190$

The screenshot shows a C program in Dev-C++ that finds the maximum difference between two elements in an array. The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int arr[] = {10, 15, 90, 200, 110};
6     int n = sizeof(arr)/sizeof(arr[0]);
7     int max_diff = arr[1] - arr[0];
8     int min_element = arr[0];
9
10    for(int i=1; i<n; i++)
11    {
12        if(arr[i] - min_element > max_diff)
13            max_diff = arr[i] - min_element;
14
15        if(arr[i] < min_element)
16            min_element = arr[i];
17    }
18
19    printf("Maximum difference is %d", max_diff);
20
21    return 0;
22 }
```

The program is compiled and executed, showing the output: "Maximum difference is 190". The compilation results show 0 errors and 0 warnings.

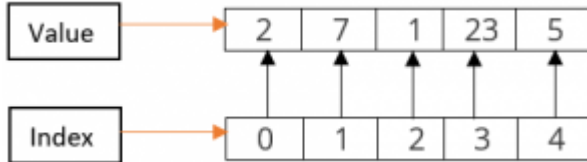
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\akhil\OneDrive\Desktop\c\max diff
- Output Size: 127.953125 KIB
- Compilation Time: 0.20s

## 5. C program to remove duplicate elements in an Array?

An **array** is a collection of similar data elements stored in a contiguous memory location.

Example:  $\text{arr}[5] = \{2, 7, 1, 23, 5\}$



Example:

Input Array: 1,2,4,5,4,2,7,5

Output: Resultant Array after removing duplicates: 1,2,4,5,7

```
1 #include<stdio.h>
2 int main(){
3     int arr[]={1,2,3,4,5,4,2,7,5};
4     int n= sizeof(arr)/sizeof(arr[0]);
5     int i,j,k;
6     for(i=0; i<n; i++){
7         for(j=i+1; j<n; j++){
8             if(arr[j] == arr[i]){
9                 for(k= j; k < n; k++){
10                     arr[k] = arr[k+1];
11                 }
12                 n--;
13             }
14             else{
15                 j++;
16             }
17         }
18     }
19     printf("resultant array after removing duplicate");
20     for(i = 0; i<n; i++){
21         printf("%d",arr[i]);
22     }
23     return 0;
}
```

resultant array after removing duplicate123457

Process exited after 0.0413 seconds with return value 0

Press any key to continue . . .

Compilation results...

- Errors: 0

- Warnings: 0

- Output Filename: C:\Users\akhi\OneDrive\Desktop\c\z

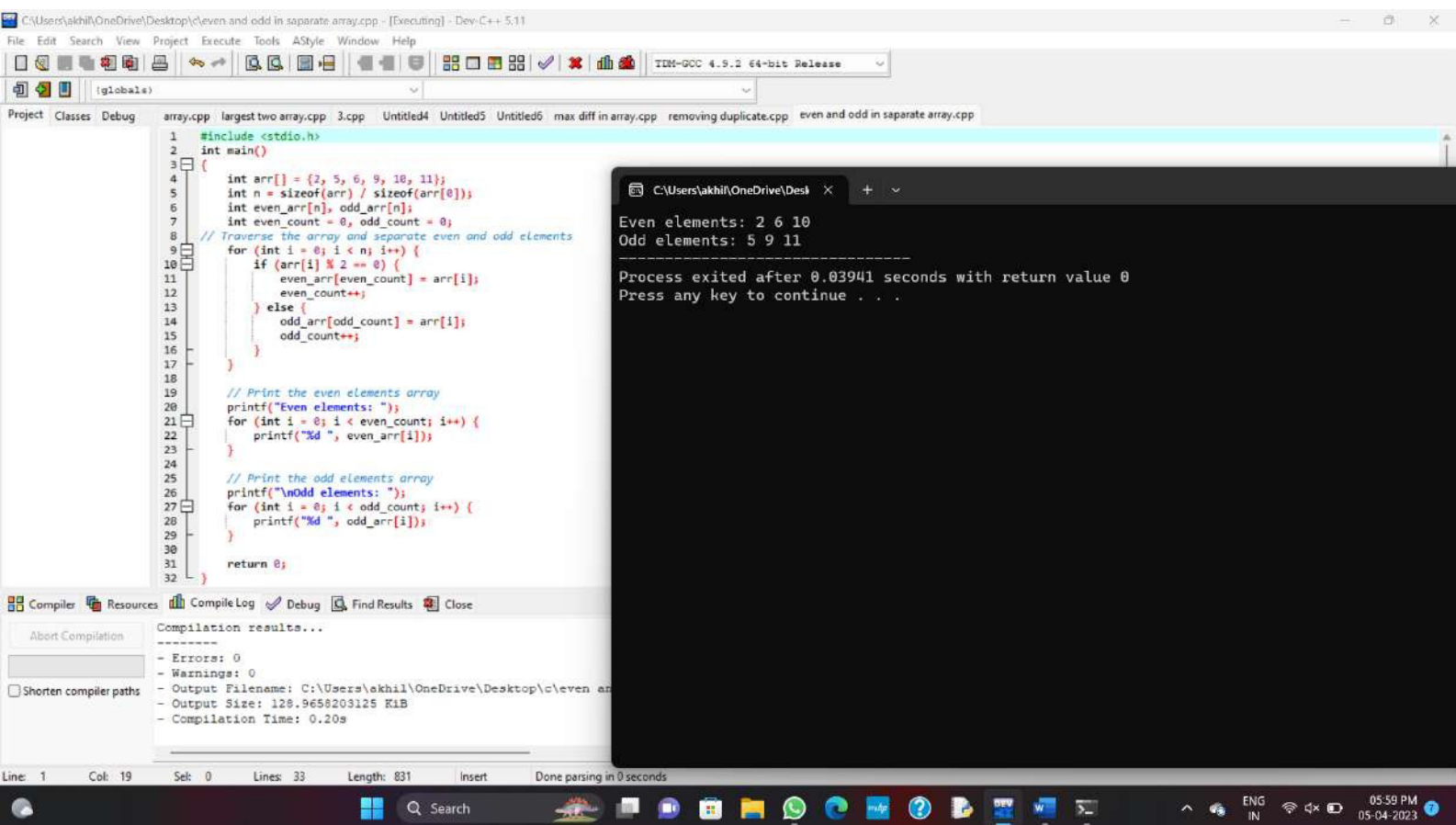
- Output Size: 127.9541015625 KiB

- Compilation Time: 0.20s

## 6. C Program to put even & odd elements of an array in 2 separate arrays.

### Problem Description

The program first finds the odd and even elements of the array. Then the odd elements of an array is stored in one array and even elements of an array is stored in another array.



```
1 #include <stdio.h>
2 int main()
3 {
4     int arr[] = {2, 5, 6, 9, 10, 11};
5     int n = sizeof(arr) / sizeof(arr[0]);
6     int even_arr[n], odd_arr[n];
7     int even_count = 0, odd_count = 0;
8     // Traverse the array and separate even and odd elements
9     for (int i = 0; i < n; i++) {
10         if (arr[i] % 2 == 0) {
11             even_arr[even_count] = arr[i];
12             even_count++;
13         } else {
14             odd_arr[odd_count] = arr[i];
15             odd_count++;
16         }
17     }
18     // Print the even elements array
19     printf("Even elements: ");
20     for (int i = 0; i < even_count; i++) {
21         printf("%d ", even_arr[i]);
22     }
23     // Print the odd elements array
24     printf("\nOdd elements: ");
25     for (int i = 0; i < odd_count; i++) {
26         printf("%d ", odd_arr[i]);
27     }
28     return 0;
29 }
```

Even elements: 2 6 10  
Odd elements: 5 9 11

Process exited after 0.03941 seconds with return value 0  
Press any key to continue . . .

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\akhil\OneDrive\Desktop\c\even and odd in separate array.cpp
- Output Size: 128.9658203125 KIB
- Compilation Time: 0.20s



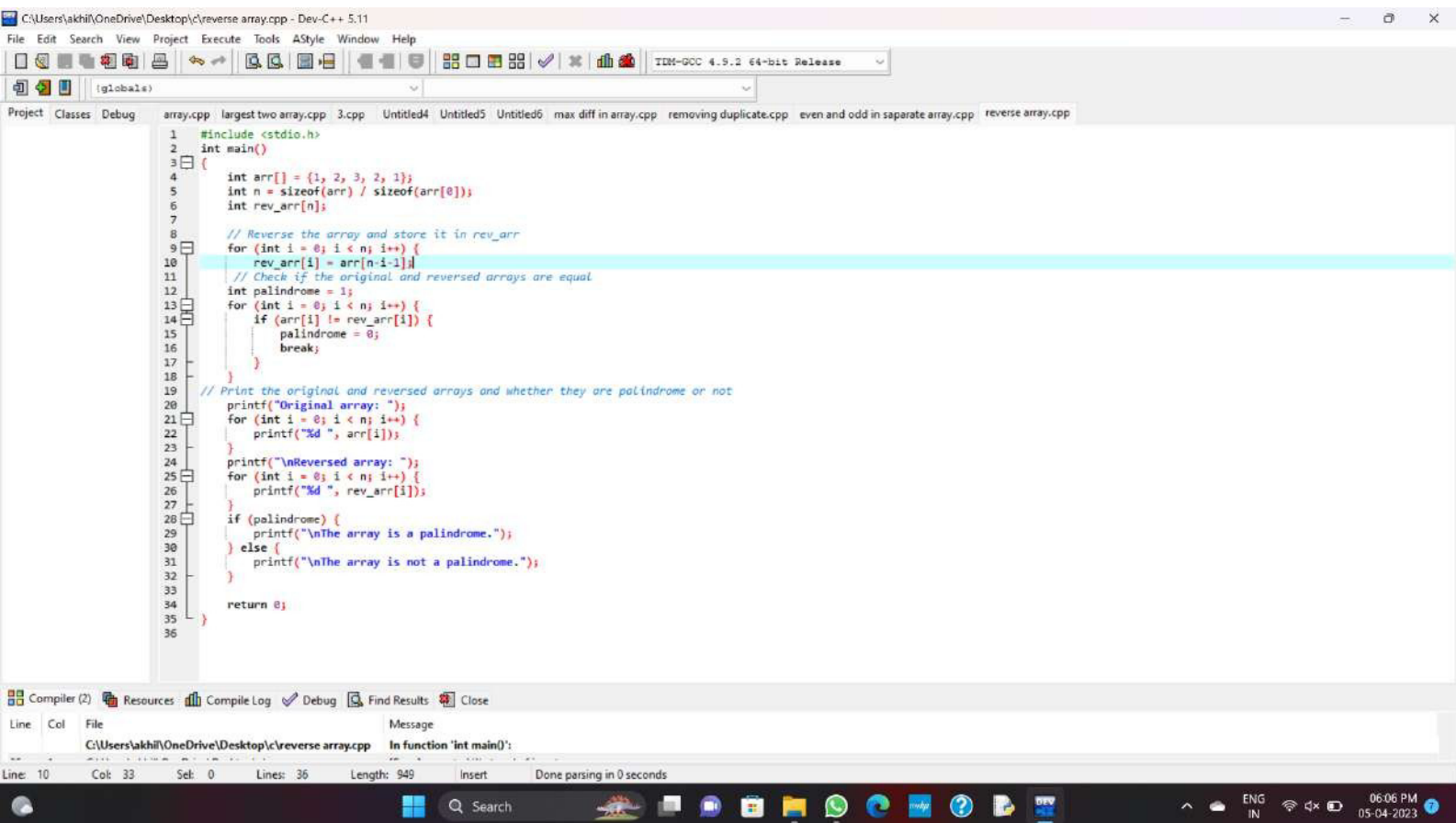
7. **Reversing an array** means substituting the last element in the first position and vice versa and doing such a thing for all elements of the array. For **example**, first element is swapped with last, second element is swapped by second last and so on.

Such arrays where the original and reversed arrays are equal are called palindrome arrays.

**Examples:**

Input array: [1,2,3,4]

Reversed array: [4,3,2,1]



```
1 #include <stdio.h>
2 int main()
3 {
4     int arr[] = {1, 2, 3, 2, 1};
5     int n = sizeof(arr) / sizeof(arr[0]);
6     int rev_arr[n];
7
8     // Reverse the array and store it in rev_arr
9     for (int i = 0; i < n; i++) {
10         rev_arr[i] = arr[n-i-1];
11     }
12     // Check if the original and reversed arrays are equal
13     int palindrome = 1;
14     for (int i = 0; i < n; i++) {
15         if (arr[i] != rev_arr[i]) {
16             palindrome = 0;
17             break;
18         }
19     }
20     // Print the original and reversed arrays and whether they are palindrome or not
21     printf("Original array: ");
22     for (int i = 0; i < n; i++) {
23         printf("%d ", arr[i]);
24     }
25     printf("\nReversed array: ");
26     for (int i = 0; i < n; i++) {
27         printf("%d ", rev_arr[i]);
28     }
29     if (palindrome) {
30         printf("\nThe array is a palindrome.");
31     } else {
32         printf("\nThe array is not a palindrome.");
33     }
34     return 0;
35 }
```

## 8. C Program to sort an array in descending order.

### Problem Description

This program will implement a one-dimensional array of some fixed size, filled with some random numbers, then will sort all the filled elements of the array.

Enter the value of N

5

Enter the numbers

234

780

130

56

90

The numbers arranged in descending order are given below

780

234

130

90

56

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int arr[100], n, i, j, temp;
6
7     printf("Enter the value of N: ");
8     scanf("%d", &n);
9
10    printf("Enter the numbers: ");
11    for(i = 0; i < n; i++)
12        scanf("%d", &arr[i]);
13
14    // Sort the array in descending order
15    for(i = 0; i < n; i++) {
16        for(j = i+1; j < n; j++) {
17            if(arr[i] < arr[j]) {
18                temp = arr[i];
19                arr[i] = arr[j];
20                arr[j] = temp;
21            }
22        }
23    }
24
25    printf("The numbers arranged in descending order are given below:\n");
26    for(i = 0; i < n; i++) {
27        printf("%d\n", arr[i]);
28    }
29
30    return 0;
31 }
```

Enter the value of N: 1  
Enter the numbers: 233  
The numbers arranged in descending order are given below:  
233

Process exited after 5.469 seconds with return value 0  
Press any key to continue . . .

Compiler: GCC 4.9.2 64-bit Release  
Errors: 0  
Warnings: 0  
Output Filename: C:\Users\akhil\OneDrive\Desktop\c\arrainging in decreasing.exe  
Output Size: 128.798828125 KiB  
Compilation Time: 0.22s



9. Given an array `arr[]` where each element represents the max number of steps that can be made forward from that index. The task is to find the minimum number of jumps to reach the end of the array starting from index 0. If the end isn't reachable, return -1.

Examples:

Input: `arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}`

Output: 3 (1-> 3 -> 9 -> 9)

Explanation: Jump from 1st element to 2nd element as there is only 1 step.

Now there are three options 5, 8 or 9. I

f 8 or 9 is chosen then the end node 9 can be reached. So 3 jumps are made.

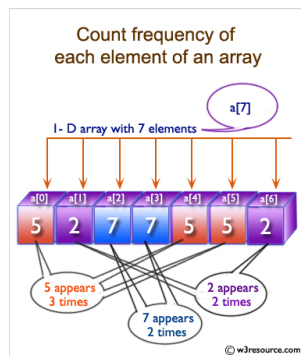
Input: `arr[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}`

Output: 10

Explanation: In every step a jump is needed so the count of jumps is 10.



10. Write a program in C to count the frequency of each element of an array.



```
C:\Users\Viwith\OneDrive\Documents\assignment 2 question number 8.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

(globals)
assignment 2 question number 8.cpp
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 int main()
4 {
5     int arr[MAX_SIZE], freq[MAX_SIZE];
6     int n, i, j, count;
7
8     printf("Enter the size of the array (max %d): ", MAX_SIZE);
9     scanf("%d", &n);
10    printf("Enter %d elements: ", n);
11    for(i=0; i<n; i++)
12    {
13        scanf("%d", &arr[i]);
14        freq[i] = -1;
15    }
16    for(i=0; i<n; i++)
17    {
18        count = 1;
19        for(j=i+1; j<n; j++)
20        {
21            if(arr[j] == arr[i])
22            {
23                count++;
24                freq[j] = 0;
25            }
26        }
27        if(freq[i] != 0)
28        {
29            freq[i] = count;
30        }
31    }
32    printf("Frequency of all elements in the array:\n");
33    for(i=0; i<n; i++)
34    {
35        if(freq[i] != 0)
36        {
37            printf("%d occurs %d times\n", arr[i], freq[i]);
38        }
39    }
40    return 0;
41 }
```

```
C:\Users\Viwith\OneDrive\Documents\assignment 2 question number 8.cpp
Enter the size of the array (max 100): 8
Enter 8 elements: 1
2
3
4
5
4
3
2
Frequency of all elements in the array:
1 occurs 1 times
2 occurs 2 times
3 occurs 2 times
4 occurs 2 times
5 occurs 1 times

-----
Process exited after 9.272 seconds with return value 0
Press any key to continue . . .
```