

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 in the editor is as follows:

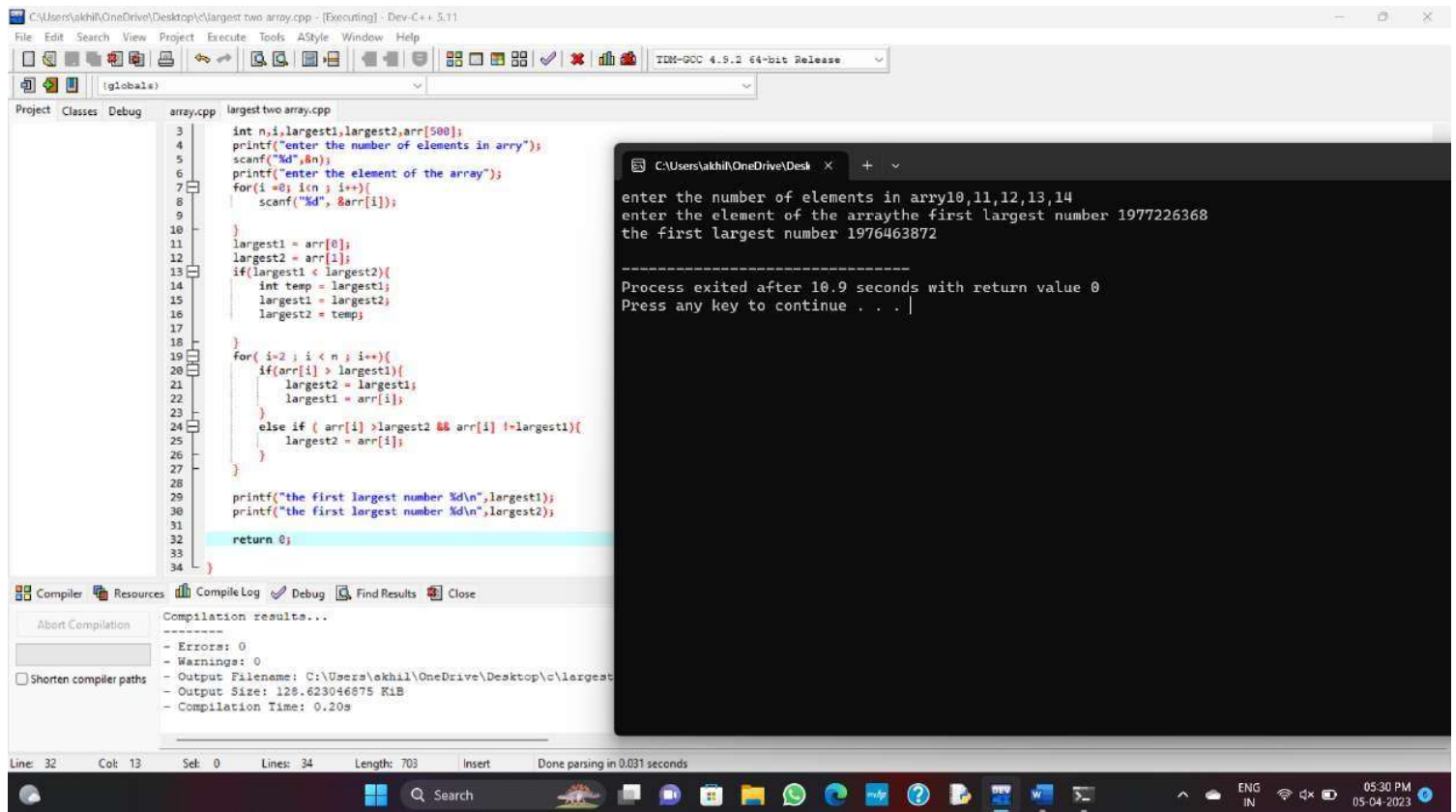
```
1 #include<stdio.h>
2 int main()
3 {
4     int arr[5] = {12,56,34,78,100};
5     int largest = arr[0];
6     for(int i = 1; i < 5; i++){
7         if(arr[i] > largest){
8             largest = arr[i];
9         }
10    }
11    printf("the largest element %d",largest);
12    return 0;
13 }
```

The output window shows the following text:

```
the largest element 100
-----
Process exited after 0.04016 seconds with return value 0
Press any key to continue . . .
```

The bottom status bar indicates the current line is 3, column is 27, and the selection is 0 lines.

2. 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.



```
3  int n,i,largest1,largest2,arr[500];
4  printf("enter the number of elements in array");
5  scanf("%d",&n);
6  printf("enter the element of the array");
7  for(i=0; i<n; i++){
8      scanf("%d", &arr[i]);
9  }
10
11  largest1 = arr[0];
12  largest2 = arr[1];
13  if(largest1 < largest2){
14      int temp = largest1;
15      largest1 = largest2;
16      largest2 = temp;
17  }
18
19  for(i=2; i < n; i++){
20      if(arr[i] > largest1){
21          largest2 = largest1;
22          largest1 = arr[i];
23      }
24      else if ( arr[i] > largest2 && arr[i] != largest1){
25          largest2 = arr[i];
26      }
27  }
28
29  printf("the first largest number %d\n",largest1);
30  printf("the first largest number %d\n",largest2);
31
32  return 0;
33
34 }
```

```
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 . . . |
```

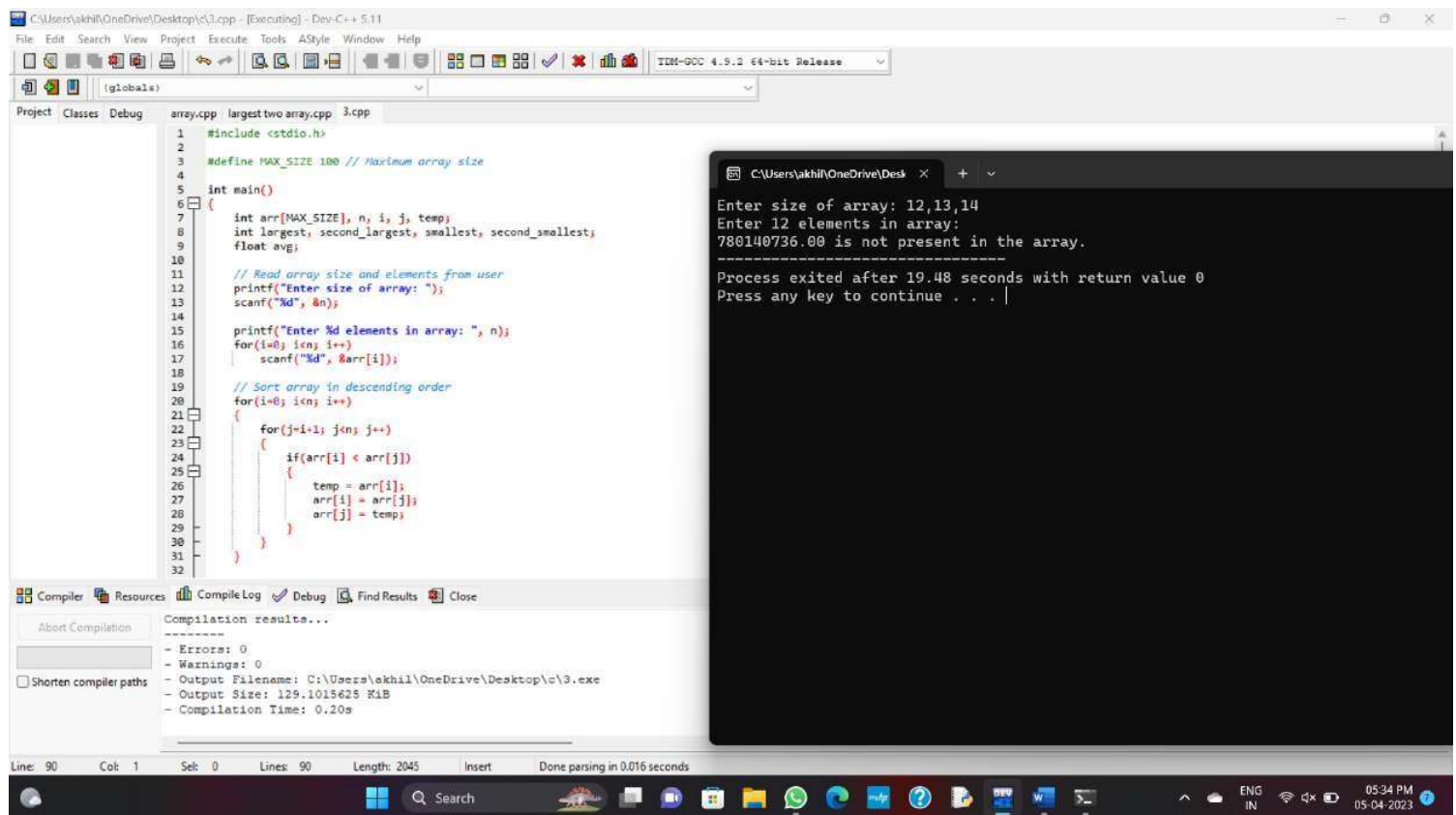
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\akhil\OneDrive\Desktop\c\largest
- Output Size: 128.623046875 KiB
- Compilation Time: 0.20s

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 }
```

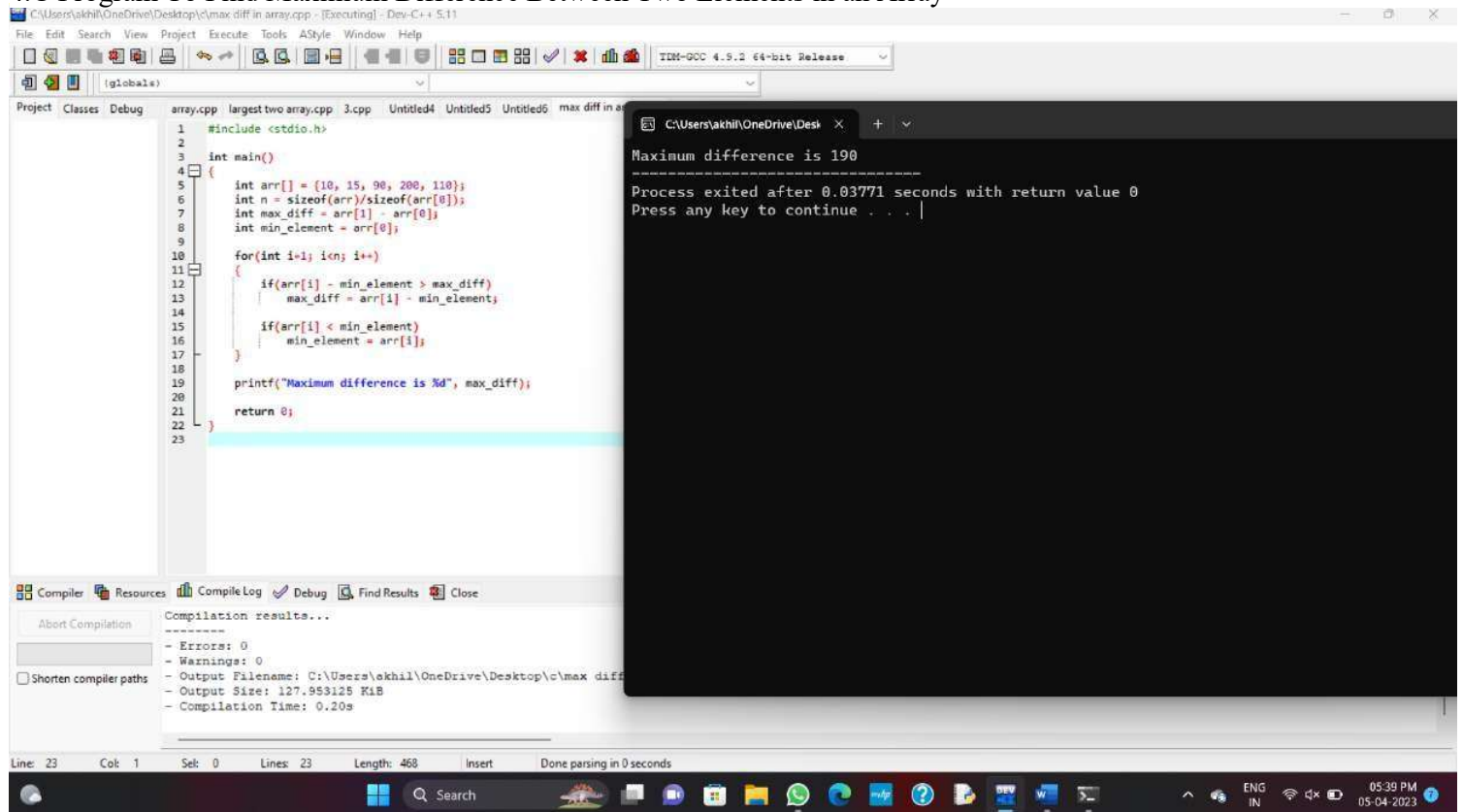
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\akhil\OneDrive\Desktop\c3.exe
- Output Size: 129.1015625 KiB
- Compilation Time: 0.20s

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 . . .

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



The screenshot displays the Dev-C++ IDE with a C++ program open in the editor. The program is titled "array.cpp" and is located at "C:\Users\akhil\OneDrive\Desktop\c\max diff in array.cpp". 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. The output window shows the following message:

```
Maximum difference is 190
-----
Process exited after 0.03771 seconds with return value 0
Press any key to continue . . .
```

The compilation results are also visible in the bottom-left pane:

```
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
```

The status bar at the bottom indicates the current line is 23, column is 1, and the file is "array.cpp". The system tray shows the time as 05:39 PM on 05-04-2023.

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

The screenshot displays the Dev-C++ IDE with a C program designed to remove duplicate elements from an array. The program uses a nested loop to compare each element with the subsequent ones. If a duplicate is found, it shifts the remaining elements one position to the right and decrements the array size. The output window shows the resulting array after removing duplicates.

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

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\akhil\OneDrive\Desktop\c\...
- 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
19     // Print the even elements array
20     printf("Even elements: ");
21     for (int i = 0; i < even_count; i++) {
22         printf("%d ", even_arr[i]);
23     }
24
25     // Print the odd elements array
26     printf("\nOdd elements: ");
27     for (int i = 0; i < odd_count; i++) {
28         printf("%d ", odd_arr[i]);
29     }
30
31     return 0;
32 }
```

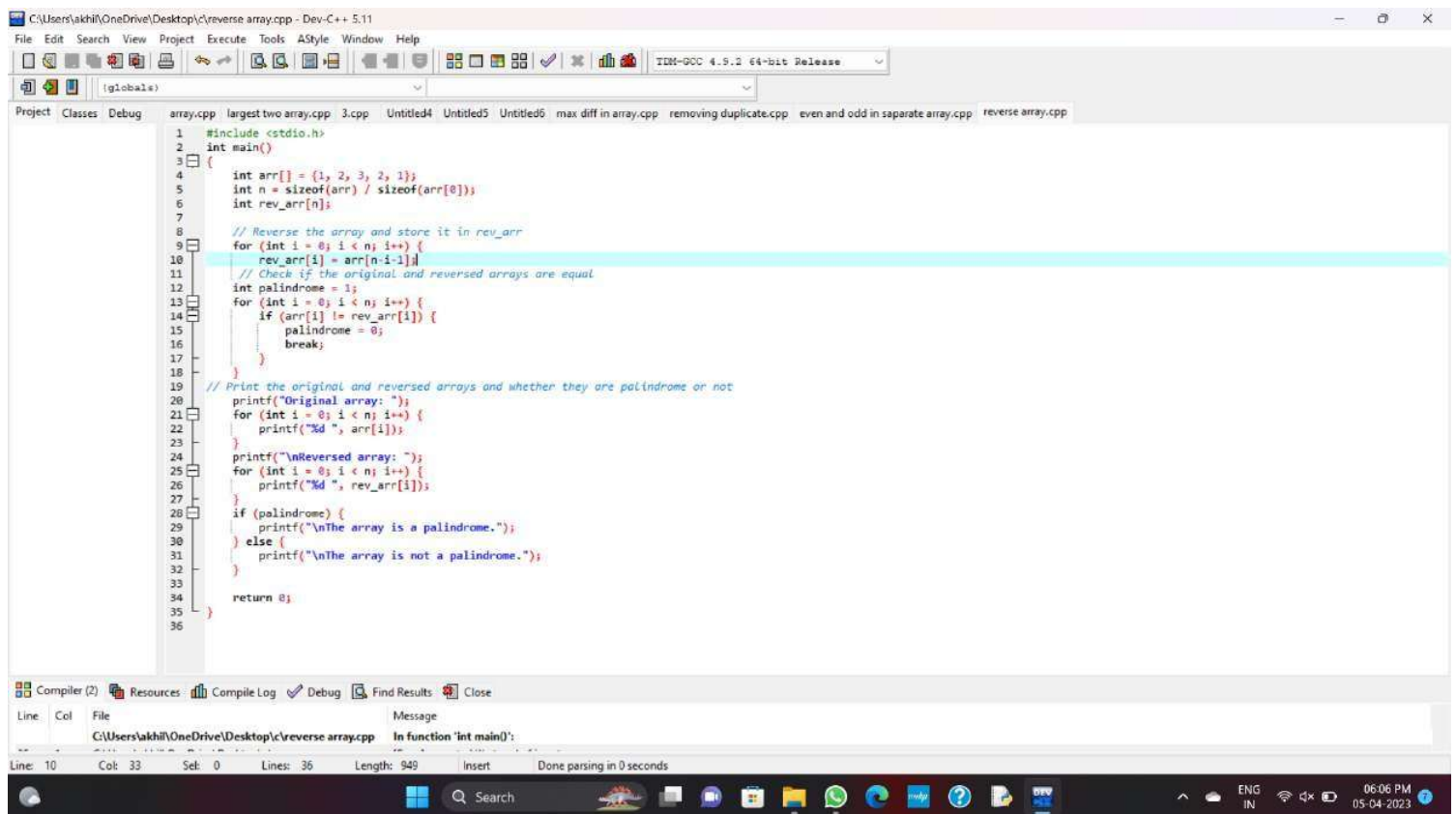
Even elements: 2 6 10
Odd elements: 5 9 11

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 saparate.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.



```
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 }
```

The screenshot shows the Dev-C++ IDE with the file 'reverse array.cpp' open. The code implements a function to reverse an array and check if it is a palindrome. The array {1, 2, 3, 2, 1} is used as an example. The program prints the original array, the reversed array, and a message indicating whether it is a palindrome. The status bar at the bottom shows 'Line: 10, Col: 33, Sel: 0, Lines: 36, Length: 940, Insert, Done parsing in 0 seconds'. The Windows taskbar at the bottom shows the date and time as 05-04-2023, 06:06 PM.

9. 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.

The screenshot shows a C program in Dev-C++ that sorts an array of 100 integers in descending order. The program uses a nested loop to compare adjacent elements and swap them if they are in the wrong order. The output window shows the user inputting the value of N as 1 and the numbers as 233. The program then displays the numbers arranged in descending order, which are all 233. The process exited after 5.469 seconds with a return value of 0.

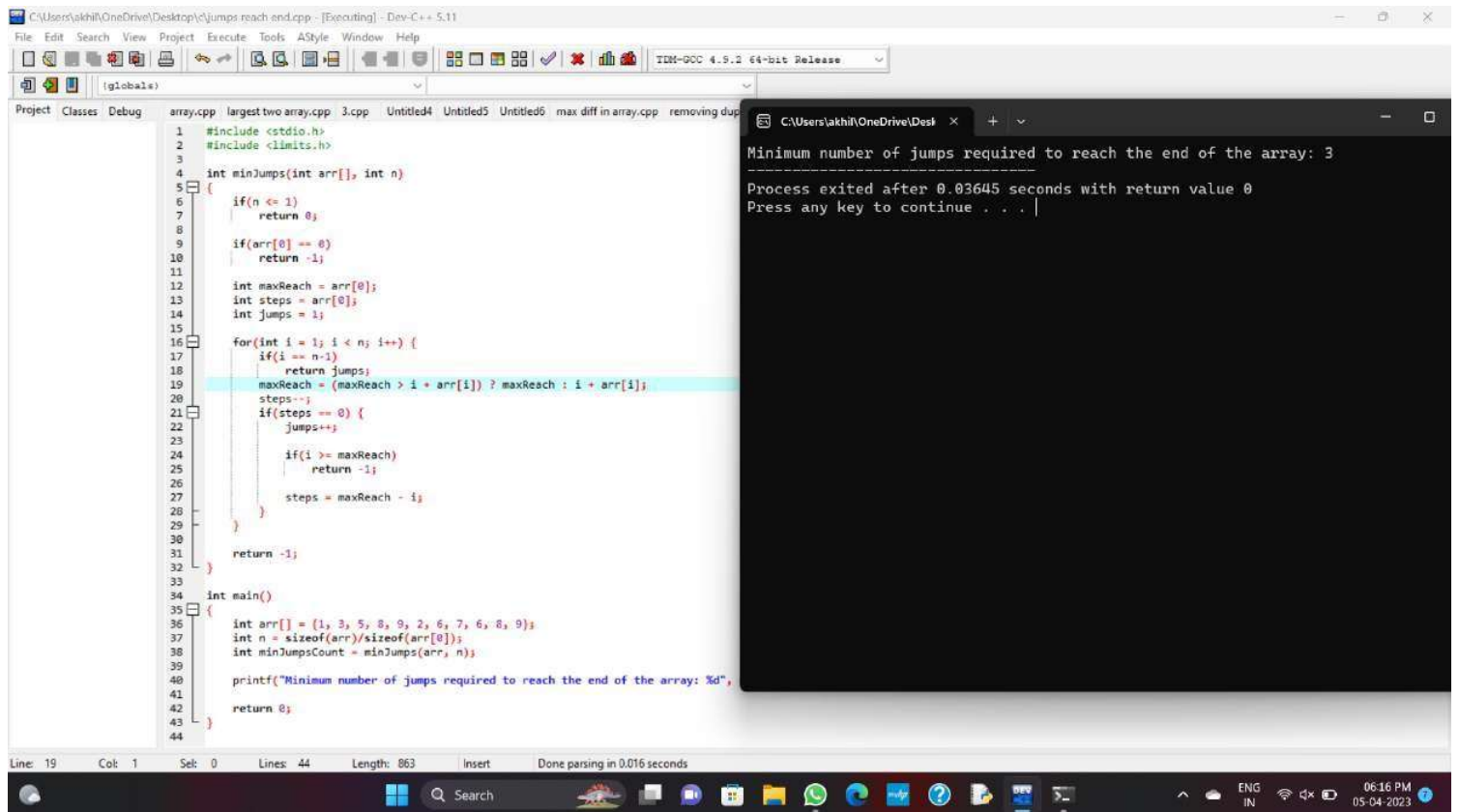
```
1 #include <stdio.h>
2
3 int main()
4 {
5     int arr[100], n, i, j, temp;
6
7     printf("Enter the value of N: ");
8     scanf("%d", &n);
9
10    printf("Enter the numbers: ");
11    for(i = 0; i < n; i++)
12        scanf("%d", &arr[i]);
13
14    // Sort the array in descending order
15    for(i = 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
Output Filename: C:\Users\akhil\OneDrive\Desktop\c\arrainging in decreasing.exe
Output Size: 128.798828125 KiB
Compilation Time: 0.22s

10. 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.



```
1 #include <stdio.h>
2 #include <limits.h>
3
4 int minJumps(int arr[], int n)
5 {
6     if(n <= 1)
7         return 0;
8
9     if(arr[0] == 0)
10        return -1;
11
12    int maxReach = arr[0];
13    int steps = arr[0];
14    int jumps = 1;
15
16    for(int i = 1; i < n; i++) {
17        if(i == n-1)
18            return jumps;
19        maxReach = (maxReach > i + arr[i]) ? maxReach : i + arr[i];
20        steps--;
21        if(steps == 0) {
22            jumps++;
23
24            if(i >= maxReach)
25                return -1;
26
27            steps = maxReach - i;
28        }
29    }
30    return -1;
31 }
32
33
34 int main()
35 {
36     int arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9};
37     int n = sizeof(arr)/sizeof(arr[0]);
38     int minJumpsCount = minJumps(arr, n);
39
40     printf("Minimum number of jumps required to reach the end of the array: %d",
41           minJumpsCount);
42     return 0;
43 }
44
```

Minimum number of jumps required to reach the end of the array: 3

Process exited after 0.03645 seconds with return value 0
Press any key to continue . . .

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

The screenshot displays the Dev-C++ IDE with a C program open in the editor. The program, named 'assignment 2 question number 8.cpp', is designed to count the frequency of each element in an array. It includes standard headers, defines a maximum array size of 100, and uses nested loops to compare elements and update a frequency array. The output window shows the program's execution, where the user enters an array size of 8 and the elements 1, 2, 3, 4, 5, 4, 3, 2. The program then prints the frequency of each element: 1 occurs 1 time, 2 occurs 2 times, 3 occurs 2 times, 4 occurs 2 times, and 5 occurs 1 time.

```
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 }
```

Execution Output:

```
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 . . .
```

