

CSA02- C Programming

N PAVAN SAI

192225023

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};
```

/* here 12,56,34,78,100 are the elements at indices 0,1,2,3,4 respectively */

In this example, array Arr is a collection of 5 integers. Each integer can be identified and accessed by its index. The indices of the array start with 0, so the first element of the array will have index 0, the next will have index 1 and so on.

Largest element of the array is the array element which has the largest numerical value among all the array elements.

Examples:

If we are entering 5 elements (N = 5), with array element values as 12, 56, 34, 78 and 100

Then, largest element present in the given array is: 100

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int arr[100], n, i, max;
6     printf("Enter the number of elements in the array: ");
7     scanf("%d", &n);
8
9     printf("Enter the elements of the array: \n");
10    for (i = 0; i < n; i++) {
11        scanf("%d", &arr[i]);
12    }
13
14    max = arr[0];
15
16    for (i = 1; i < n; i++) {
17        if (arr[i] > max) {
18            max = arr[i];
19        }
20    }
21
22    printf("The largest element in the array is %d\n", max);
23
24    return 0;
25 }
```

Enter the elements of the array:

10
15
13
20
22

The largest element in the array is 22

Process exited after 16.06 seconds with return value 0
Press any key to continue . . .

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 shows a C program in a code editor. The program prompts the user to enter the size of the array (N) and then the elements. It uses a loop to read the elements into an array. After reading, it prints the array elements. Then, it uses a loop to find the first and second largest elements. The output window shows the following: Enter the size of the array: 5, Enter the elements: 25, 55, 87, 98, 76, The array elements are: 25 55 87 98 76, The FIRST LARGEST = 98, THE SECOND LARGEST = 87.

The screenshot shows a C program in a code editor. The program prompts the user to enter the size of the array (N) and then the elements. It uses a loop to read the elements into an array. After reading, it prints the array elements. Then, it uses a loop to find the first and second largest elements. The output window shows the following: Enter the size of the array: 6, Enter the elements: 25, 55, 87, 98, 76, The array elements are: 25 55 87 98 76, The FIRST LARGEST = 98, THE SECOND LARGEST = 87.

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

The screenshot shows a C program in Dev-C++ that finds the second largest and smallest elements in an array. The program is titled "C 26.cpp" and is located at "C:\Users\Pawan Sai\OneDrive\Documents\C 26.cpp". The code is as follows:

```
1 #include <stdio.h>
2 int main()
3 {
4     int a[20], b[20], n, sm1=0, pos, i, j, temp;
5     printf("Enter the Numbers of terms: ");
6     scanf("%d", &n);
7     printf("\nEnter the terms: \n");
8     for (i = 1; i <= n; i++)
9     {
10         scanf("%d", &a[i]);
11         b[i] = a[i];
12     }
13     for (i = 1; i <= n; i++)
14     {
15         for (j = 1; j <= n; j++)
16         {
17             if (a[i] <= a[j])
18             {
19                 temp = a[i];
20                 a[i] = a[j];
21                 a[j] = temp;
22             }
23         }
24     }
25     printf("\n The Array Elements are: \n");
26     for (i = 1; i <= n; i++)
27     {
28         printf("%d\t", b[i]);
29     }
30     printf("\n Second Smallest Element is : %d", a[2]);
31     printf("\n Second Largest Element is : %d", a[n-1]);
32     return (0);
33 }
```

The program prompts the user to enter the number of terms (5) and then the terms themselves (16, 14, 12, 18, 19). It then displays the array elements and the second smallest and largest elements.

Compilation results:

```
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 26.exe
- Output Size: 128.6005859375 Kib
- Compilation Time: 0.33s
```

Process exited after 17.44 seconds with return value 0
Press any key to continue . . .

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

Example:

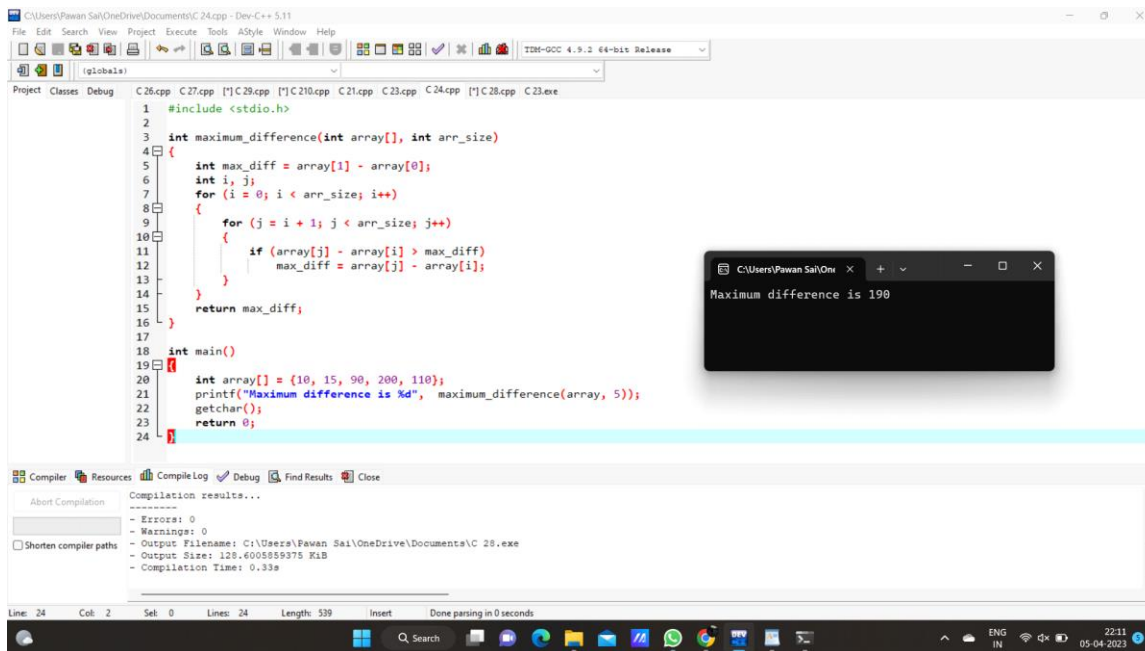
Consider the Following Array

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

Output:

Maximum difference is 190

That is $200-10=190$



```
1 #include <stdio.h>
2
3 int maximum_difference(int array[], int arr_size)
4 {
5     int max_diff = array[1] - array[0];
6     int i, j;
7     for (i = 0; i < arr_size; i++)
8     {
9         for (j = i + 1; j < arr_size; j++)
10        {
11            if (array[j] - array[i] > max_diff)
12                max_diff = array[j] - array[i];
13        }
14    }
15    return max_diff;
16 }
17
18 int main()
19 {
20     int array[] = {10, 15, 90, 200, 110};
21     printf("Maximum difference is %d", maximum_difference(array, 5));
22     getchar();
23     return 0;
24 }
```

Maximum difference is 190

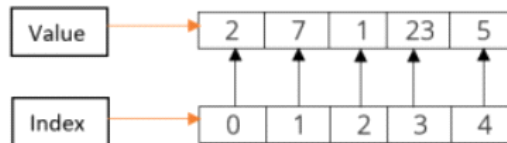
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 28.exe
- Output Size: 128.4005859375 KiB
- Compilation Time: 0.33s

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

The screenshot shows a C program in the Dev-C++ IDE. The program defines an array `arr` with the values `{1, 2, 4, 5, 4, 2, 7, 5}` and uses a nested loop to remove duplicates. The output window shows the result: "Resultant Array after removing duplicates: 1 2 4 5 7". The compiler output at the bottom indicates that the program compiled successfully with no errors or warnings.

```
1 #include <stdio.h>
2 int main()
3 {
4     int arr[] = {1, 2, 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             } else {
15                 j++;
16             }
17         }
18     }
19     printf("Resultant Array after removing duplicates: ");
20     for (i = 0; i < n; i++) {
21         printf("%d ", arr[i]);
22     }
23     return 0;
24 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 28.exe
- Output Size: 128.6005859375 KiB
- Compilation Time: 0.33s

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

The screenshot shows a C program in Dev-C++ that separates even and odd elements of an array into two separate arrays. The program is titled "C 26.cpp" and is located at "C:\Users\Pawan Sai\OneDrive\Documents\C 26.cpp". The code is as follows:

```
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     for (int i = 0; i < n; i++) {
9         if (arr[i] % 2 == 0) {
10             even_arr[even_count] = arr[i];
11             even_count++;
12         } else {
13             odd_arr[odd_count] = arr[i];
14             odd_count++;
15         }
16     }
17     printf("Even elements: ");
18     for (int i = 0; i < even_count; i++) {
19         printf("%d ", even_arr[i]);
20     }
21     printf("\nOdd elements: ");
22     for (int i = 0; i < odd_count; i++) {
23         printf("%d ", odd_arr[i]);
24     }
25     return 0;
26 }
```

The program output is displayed in a separate window titled "C:\Users\Pawan Sai\OneDrive":

```
Even elements: 2 6 10
Odd elements: 5 9 11
-----
Process exited after 0.02357 seconds with return value 0
Press any key to continue . . .
```

The Dev-C++ interface also shows the "Compiler" window with the following output:

```
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 26.exe
- Output Size: 128.6005859375 Kib
- Compilation Time: 0.33s
```

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.

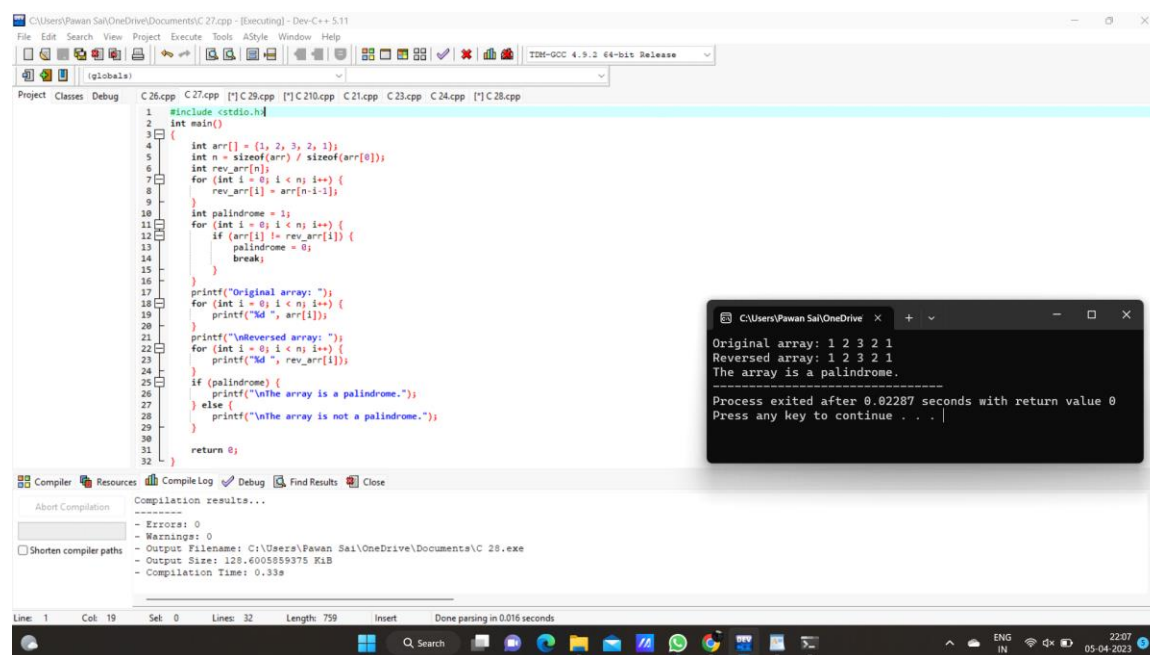
Examples:

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

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

Input array: [3,2,1]

Reversed array: [1,2,3]



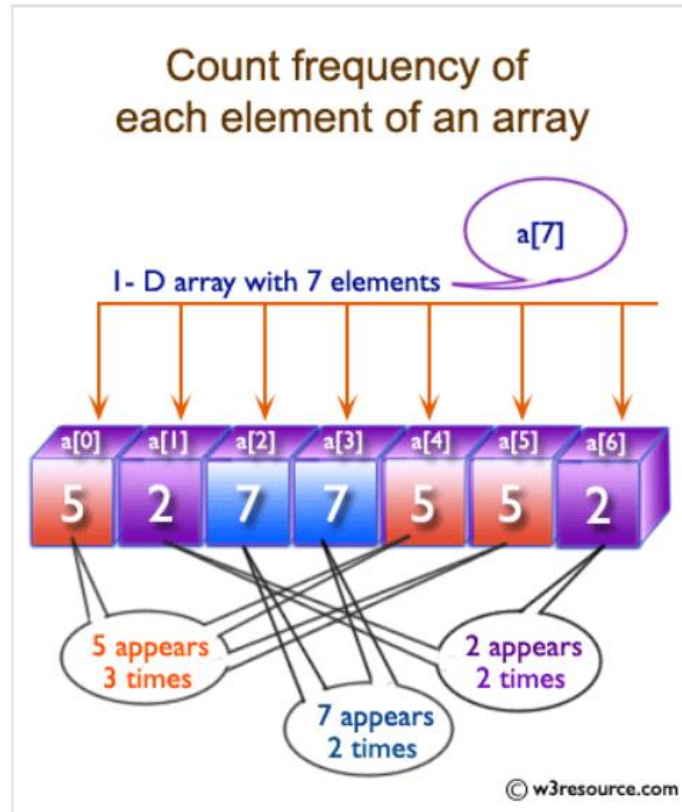
```
1 #include <stdio.h>
2 int main()
3 {
4     int arr[] = {1, 2, 3, 2, 1};
5     int n = sizeof(arr) / sizeof(arr[0]);
6     int rev_arr[n];
7     for (int i = 0; i < n; i++) {
8         rev_arr[i] = arr[n-i-1];
9     }
10    int palindrome = 1;
11    for (int i = 0; i < n; i++) {
12        if (arr[i] != rev_arr[i]) {
13            palindrome = 0;
14            break;
15        }
16    }
17    printf("Original array: ");
18    for (int i = 0; i < n; i++) {
19        printf("%d ", arr[i]);
20    }
21    printf("\nReversed array: ");
22    for (int i = 0; i < n; i++) {
23        printf("%d ", rev_arr[i]);
24    }
25    if (palindrome) {
26        printf("\nThe array is a palindrome.");
27    } else {
28        printf("\nThe array is not a palindrome.");
29    }
30    return 0;
31 }
```

```
C:\Users\Pawan Sai\OneDrive\Documents>
Original array: 1 2 3 2 1
Reversed array: 1 2 3 2 1
The array is a palindrome.
=====
Process exited after 0.02287 seconds with return value 0
Press any key to continue . . .
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 28.exe
- Output Size: 128.600599375 KB
- Compilation Time: 0.33s

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



The screenshot shows a Windows desktop with a taskbar at the bottom. The main application is Visual Studio, which is open to a C++ project. The code editor displays a file named `C28.cpp` with the following content:

```

1 #include <stdio.h>
2 int main()
3 {
4     int arr[] = {5, 2, 7, 7, 5, 2, 3};
5     int length = sizeof(arr)/sizeof(arr[0]);
6     int fr[length];
7     int visited = -1;
8
9     for(int i = 0; i < length; i++){
10         int count = 1;
11         for(int j = i+1; j < length; j++){
12             if(arr[i] == arr[j]){
13                 count++;
14                 fr[j] = visited;
15             }
16             if(fr[i] != visited)
17                 fr[i] = count;
18         }
19         printf("-----\n");
20         printf(" Element | Frequency\n");
21         printf("-----\n");
22         for(int i = 0; i < length; i++){
23             if(fr[i] != visited){
24                 printf(" %d | %d\n", arr[i], fr[i]);
25                 printf("-----\n");
26             }
27         }
28     }
29     printf("-----\n");
30     return 0;
31 }

```

The IDE's status bar at the bottom indicates "Done parsing in 0.016 seconds".

To the right of the IDE, a terminal window is open, showing the output of the program. It displays a table of Element and Frequency, followed by a message indicating the process exited after 0.02289 seconds with return value 0.

Element	Frequency
5	3
2	2
7	2

Process exited after 0.02289 seconds with return value 0
Press any key to continue . . .

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.

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 int main()
3 {
4     int arr[100], n, i, j, temp;
5
6     printf("Enter the value of N: ");
7     scanf("%d", &n);
8
9     printf("Enter the numbers: ");
10    for(i = 0; i < n; i++)
11        scanf("%d", &arr[i]);
12    for(i = 0; i < n; i++) {
13        for(j = i+1; j < n; j++) {
```

Enter the value of N: 5
Enter the numbers: 15
22
13
12
24
The numbers arranged in descending order are given below:
24
22
15
13
12
Process exited after 17.54 seconds with return value 0
Press any key to continue . . .

```
13        for(j = i+1; j < n; j++) {
14            if(arr[i] < arr[j]) {
15                temp = arr[i];
16                arr[i] = arr[j];
17                arr[j] = temp;
18            }
19        }
20    }
21
22    printf("The numbers arranged in descending order are given below:\n");
23    for(i = 0; i < n; i++) {
24        printf("%d\n", arr[i]);
25    }
26
27    return 0;
```

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.

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.

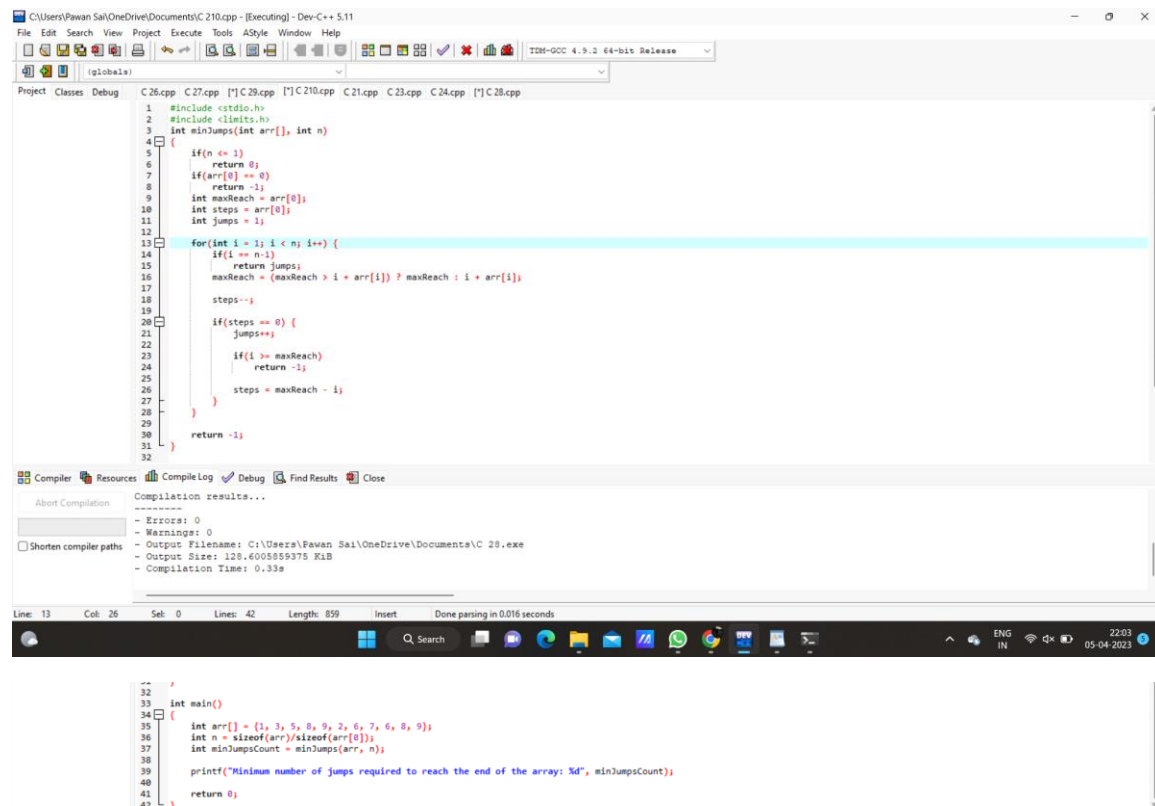
Now there are three options 5, 8 or 9. I

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}`

Output: 10

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



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

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Pawan Sai\OneDrive\Documents\C 28.exe
- Output Size: 128.400859375 Kib
- Compilation Time: 0.33s

Line: 13 Col: 26 Sel: 0 Lines: 42 Length: 859 Insert Done parsing in 0.016 seconds

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

```
-----
```

```
Process exited after 0.01813 seconds with return value 0
```

```
Press any key to continue . . .
```