One dimensional array in C

Arrays are a fundamental concept in programming, and they come in different dimensions. ***One-dimensional arrays***, also known as single arrays, are arrays with only one dimension or a single row. In this article, we'll dive deep into one-dimensional arrays in C programming language, including their ***syntax, examples***, and ***output***.

Syntax of One-Dimensional Array in C

The syntax of a ***one-dimensional array*** in C programming language is as follows:

1. dataType arrayName[arraySize];
   - **dataType** specifies the data type of the array. It can be any valid data type in C programming language, such as ***int, float, char, double***, etc.
   - **arrayName** is the name of the array, which is used to refer to the array in the program.
   - **arraySize** specifies the number of elements in the array. It must be a ***positive integer value***.

Example of One-Dimensional Array in C

Let's take a simple example of a one-dimensional array in C programming language to understand its syntax and usage.

```
#include <stdio.h>
 int main() {
   int numbers[5] = {10, 20, 30, 40, 50};
for(int i=0; i<5; i++) {
printf("numbers[%d] = %d\n", i, numbers[i]);
  }
    return 0;
}
```

Output:

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

Explanation:

In the above example, we have declared a one-dimensional array of integers named ***numbers***. The array contains five elements, and each element is initialized with a value.

We have used a ***for loop*** to iterate over the elements of the array and print their values using the ***printf*** function.

Accessing Elements of One-Dimensional Array in C

In a one-dimensional array, each element is identified by its ***index*** or ***position*** in the array. The index of the first element in the array is ***0***, and the index of the last element is ***arraySize - 1***.

To access an element of a one-dimensional array in C programming language, we use the following syntax:

1. arrayName[index]

- **arrayName** is the name of the array.
- **index** is the index of the element we want to access.

Example of Accessing Elements of One-Dimensional Array in C

Let's take an example to understand how to access elements of a one-dimensional array in C programming language.

```
1. #include <stdio.h>
2.  int main() {
3.    int numbers[5] = {10, 20, 30, 40, 50};
```

```
4. printf("The first element of the array is: %d\n", numbers[0]);
5. printf("The third element of the array is: %d\n", numbers[2]);
6. 
7.    return 0;
8. }
```

Output:

```
The first element of the array is: 10
The third element of the array is: 30
```

Explanation:

In the above example, we have declared a one-dimensional array of integers named ***numbers***. We have accessed the first element of the array using the ***index 0*** and the third element of the array using the ***index 2***.

Accessing Elements of One-Dimensional Arrays

We can access individual elements of a one-dimensional array using their index, which is an integer value that represents the position of the element in the array. The index of the first element in the array is ***0***, and the index of the last element is ***arraySize-1***.

The syntax to access an element of a one-dimensional array in C is as follows:

1. arrayName[index]

- **arrayName** is the name of the array.
- **index** is the index of the element we want to access.

Example of Accessing Elements of One-Dimensional Array in C

```
1. #include <stdio.h>
2. int main() {
3.    int numbers[5] = {10, 20, 30, 40, 50};
4. printf("The third element of the array is %d\n", numbers[2]);
```

```
    return 0;
}
```

Output:

```
The third element of the array is 30
```

Explanation:

In the above example, we have declared a one-dimensional array of integers named ***numbers*** and initialized it with the values ***{10, 20, 30, 40, 50}***. We have used the ***printf*** function to print the third element of the array, which is accessed using the ***index 2***.

Modifying Elements of One-Dimensional Arrays

We can modify the value of individual elements of a one-dimensional array using their index. To modify an element, we simply assign a new value to it using the ***assignment operator =***.

Example of Modifying Elements of One-Dimensional Array in C

```
#include <stdio.h>

int main() {
   int numbers[5] = {10, 20, 30, 40, 50};
printf("The third element of the array is %d\n", numbers[2]);
numbers[2] = 35;
    printf("The third element of the array is now %d\n", numbers[2]);

    return 0;

}
```

ADVERTISEMENT

Output:

```
The third element of the array is 30
The third element of the array is now 35
```

Explanation:

In the above example, we have declared a one-dimensional array of integers named ***numbers*** and initialized it with the values ***{10, 20, 30, 40, 50}***. We have used the ***printf*** function to print the third element of the array, which is accessed using the ***index 2***.

After that, we modified the value of the ***third element*** by assigning a new value of ***35*** to it. Finally, we have used the ***printf*** function again to print the new value of the third element.

Initializing One Dimensional Array in C

In C programming language, we can initialize a one-dimensional array while declaring it or later in the program. We can initialize a one-dimensional array while declaring it by using the following syntax:

```
1. dataType arrayName[arraySize] = {element1, element2, ..., elementN};
```

- **"dataType'** specifies the data type of the array.

- **"arrayName'** is the name of the array.

- **"arraySize'** specifies the number of elements in the array.

- **"{element1, element2, ..., elementN}'** specifies the values of the elements in the array. The number of elements must be equal to ***"arraySize'***.

Example of Initializing One Dimensional Array in C

Let's take an example to understand how to initialize a one-dimensional array in C programming language.

```
1. #include <stdio.h>
```

```
2.  int main() {
3.     int numbers[5] = {10, 20, 30, 40, 50};
4.       for(int i=0; i<5; i++) {
5.        printf("numbers[%d] = %d\n", i, numbers[i]);
6.     }
7.      return 0;
8.  }
```

Output of the code:

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

Explanation:

In the above example, we have declared a one-dimensional array of integers named ***numbers*** and initialized it with the values ***{10, 20, 30, 40, 50}***. We have used a ***for loop*** to iterate over the elements of the array and print their values using the ***printf*** function.

We can also initialize a ***one-dimensional array*** later in the program by assigning values to its elements using the following syntax:

```
1.  arrayName[index] = value;
```

- ***arrayName*** is the name of the array.

- **index** is the index of the element we want to assign a value to.

- **value** is the value we want to assign to the element.

Example of Initializing One Dimensional Array in C

Let's take an example to understand how to initialize a one-dimensional array later in the program in C programming language.

```c
#include <stdio.h>
int main() {
   int numbers[5];
      numbers[0] = 10;
   numbers[1] = 20;
   numbers[2] = 30;
   numbers[3] = 40;
   numbers[4] = 50;
     for(int i=0; i<5; i++) {
     printf("numbers[%d] = %d\n", i, numbers[i]);
  }
     return 0;
}
```

Output of the code:

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

Explanation:

In the above example, we have declared a one-dimensional array of integers ***named*** numbers. We have initialized the elements of the array later in the program by assigning values to them. We have used a ***for loop*** to iterate over the elements of the array and print their values using the ***printf*** function.

Conclusion:

In this article, we have covered ***one-dimensional arrays*** in C programming language, including their ***syntax, examples***, and ***output***. One-dimensional arrays are an essential

concept in programming, and they allow us to store multiple values of the same data type in a single variable. We can access, initialize, and manipulate one-dimensional arrays using various techniques and operations, such as loops and assignments. By mastering one-dimensional arrays, we can solve a wide range of programming problems and implement many useful algorithms and data structures.

Overall, understanding one-dimensional arrays in C is a fundamental building block of programming, and it is a skill that every aspiring programmer should possess. By applying the principles and techniques covered in this article, you can become proficient in working with one-dimensional arrays and apply this knowledge to more complex programming challenges.

Here is the full code example from the article :

```
1. #include <stdio.h>
2. int main() {
3.    int numbers[5] = {10, 20, 30, 40, 50};
4.   for(int i=0; i<5; i++) {
5.       printf("numbers[%d] = %d\n", i, numbers[i]);
6.    }
7.   return 0 ;
8. }
```

Output:

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

Two Dimensional Array in C

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

```
data_type array_name[rows][columns];
```

Consider the following example.

```
int twodimen[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns.

Initialization of 2D Array in C

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

Two-dimensional array example in C

```
#include<stdio.h>
int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

```
5.  //traversing 2D array
6.  for(i=0;i<4;i++){
7.   for(j=0;j<3;j++){
8.     printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
9.   }//end of j
10. }//end of i
11. return 0;
12. }
```

Output

```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```

C 2D array example: Storing elements in a matrix and printing it.

```
1.  #include <stdio.h>
2.  void main ()
3.  {
4.     int arr[3][3],i,j;
5.     for (i=0;i<3;i++)
6.     {
7.        for (j=0;j<3;j++)
```

```
        {
            printf("Enter a[%d][%d]: ",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("\n printing the elements ....\n");
    for(i=0;i<3;i++)
    {
        printf("\n");
        for (j=0;j<3;j++)
        {
            printf("%d\t",arr[i][j]);
        }
    }
}
```

Output

```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34

Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78

 printing the elements ....

56      10      30
34      21      34
45      56      78
```