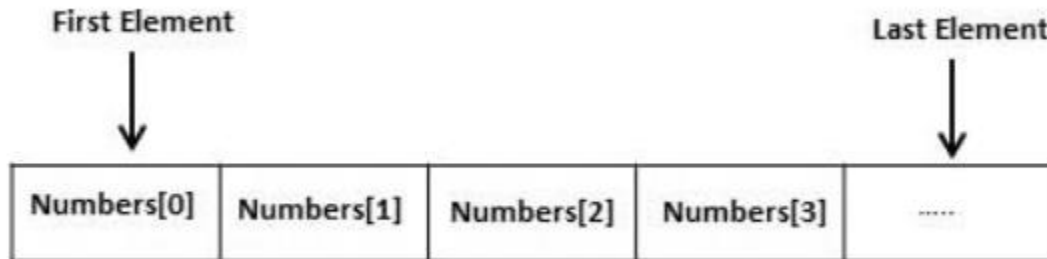# Arrays in C

- An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it.
- All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.
- A specific element in an array is accessed by an index.

Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

This is called a single-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C data type. For example, to declare a 10-element array called balance of type double, use this statement −

double balance[10];

How to access element of an array in C

You can use **array index** to access any element stored in array. Subscript starts with 0, which means arr[0] represents the first element in the array arr.

In general, arr[n-1] can be used to access nth element of an array. where n is any integer number.

For example:

```
int mydata[20];

mydata[0] /* first element of array mydata*/

mydata[19] /* last (20th) element of array mydata*/
```

Example 1: Program to take 5 values from the user and store them in an array. Print the elements stored in the array

```c
#include <stdio.h>

int main() {
  int values[5];

  printf("Enter 5 integers: ");

  // taking input and storing it in an array
  for(int i = 0; i < 5; ++i) {
     scanf("%d", &values[i]);
  }

  printf("Displaying integers: ");

  // printing elements of an array
  for(int i = 0; i < 5; ++i) {
     printf("%d\n", values[i]);
  }
  return 0;
}
```

Output

```
Enter 5 integers: 1
-3
34
0
3
Displaying integers: 1
-3
34
0
3
```

Here, we have used a `for` loop to take 5 inputs from the user and store them in an array. Then, using another `for` loop, these elements are displayed on the screen.

Example 2: Program to find the average of n numbers using arrays

```c
#include <stdio.h>
int main()
{
    int marks[10], i, n, sum = 0, average;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    for(i=0; i<n; ++i)
    {
        printf("Enter number%d: ",i+1);
        scanf("%d", &marks[i]);

        // adding integers entered by the user to the sum variable
        sum += marks[i];
    }

    average = sum/n;
    printf("Average = %d", average);

    return 0;
}
```

Output

```
Enter n: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39
```

C Multidimensional Arrays

```
int x[3][4];
```

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as a table with 3 rows and each row has 4 columns.

Similarly, you can declare a Three-Dimensional (3D) array. For example,

```
int a[2][4][3];
```

Here, the array a can hold 24 elements.

Here is the general form of a multidimensional array declaration −

```
type name[size 1][size 2]...[size N];
```

Initializing Two-Dimensional Arrays

Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

```
int a[3][4] = {
   {0, 1, 2, 3} ,    /*  initializers for row indexed by 0 */
   {4, 5, 6, 7} ,    /*  initializers for row indexed by 1 */
   {8, 9, 10, 11}    /*  initializers for row indexed by 2 */
};
```

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous example −

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example −

```
int val = a[2][3];
```

```c
#include <stdio.h>

int main () {

   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {

      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }

   return 0;
}
```

OUTPUT :

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```