**How to declare arrays in C Language**

For array declaration in C Programming, you need to specify the datatype and the count of elements you want the array to be. Here, we are discussing about one-dimensional arrays. For example, write [3], if you want to declare an array with 3 elements.

The following syntax will make the concept clearer,

|  |  |
| --- | --- |
| 1  2  3 | datatype name\_of\_array[size\_of\_array]; |

Here,  
**datatype:** C data type such as int, float, etc.  
**name\_of\_array:** The name of an array (variable)  
**size\_of\_array:** The size of the array under square brackets. Add a value greater than zero.

The following example shows the perfect way of declaring arrays in C,

|  |  |
| --- | --- |
| 1  2  3 | int marks[3]; |

Above, we declared an array, with datatype **int,**name of array **marks**, with 3 elements, since the size we specified is 3. The variable array **marks**now hold 3 elements.

**How to initialize arrays**

To initialize arrays, you need to add the array values in a single line. Continuing the previous example,

|  |  |
| --- | --- |
| 1  2  3 | int marks[3]; |

To initialize the above, add the array values like this in braces,

|  |  |
| --- | --- |
| 1  2  3 | int marks[3] = {90, 75, 88} ; |

Above, we added marks for 3 students by initializing our array. Always remember, element one would come at index 0 and the last element at **size – 1**.

For example,

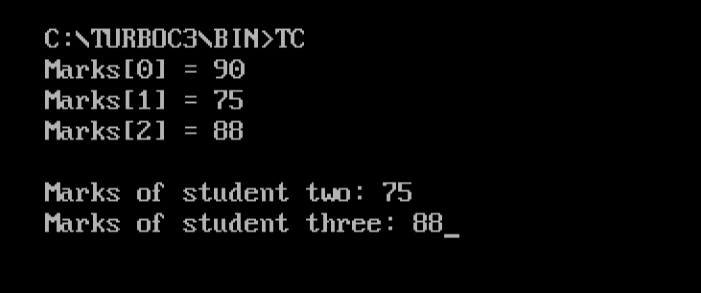
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // index 0 for first element  marks[0] = 90;    // index (size-1) for last element  marks[2] = 88 |

**First C Program with Arrays**

To move further, let us see a complete C Program, and learn how to declare and initialize arrays. We will also see how to access array elements,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>    void main () {       // marks array with 3 elements     int marks[ 3 ] = {90, 75, 88};     int i,j;       // displaying all the elements     for ( i = 0; i < 3; i++ ) {        printf("Marks[%d] = %d\n", i, marks[i] );     }       // accessing array elements     printf("\nMarks of student two: %d",marks[1]);     printf("\nMarks of student three: %d",marks[2]);       getch();  } |

The following is the output, displaying declaration, initialization of one-dimensional array, and how to access individual array elements,



**Two Dimensional Arrays in C**

An array can have two or more dimensions. Here we will discuss two-dimensional arrays. A two-dimensional array as the name suggests has two dimensions, not one like we saw in one-dimensional arrays.

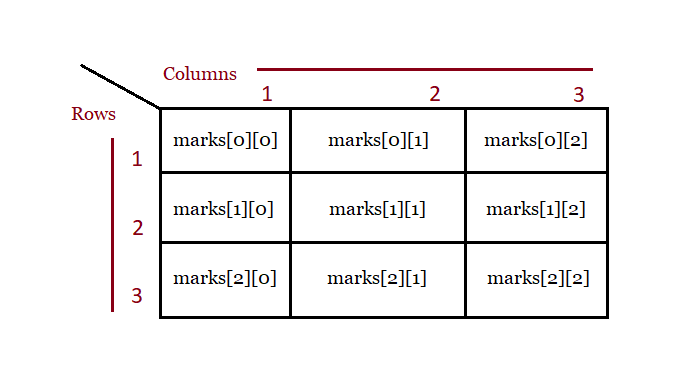
Before the example, let’s see the syntax,

|  |  |
| --- | --- |
| 1  2  3 | datatype name\_of\_array[rows\_count][columns\_count]; |

Considering the above syntax, let’s see an example, with 3 rows and columns in a two-dimensional array,

|  |  |
| --- | --- |
| 1  2  3 | int cricket[3][3]; |

Let’s now see how rows and columns work in a two-dimensional array,

**

Now, let’s see a complete example to learn more about two-dimensional arrays.

**Example**

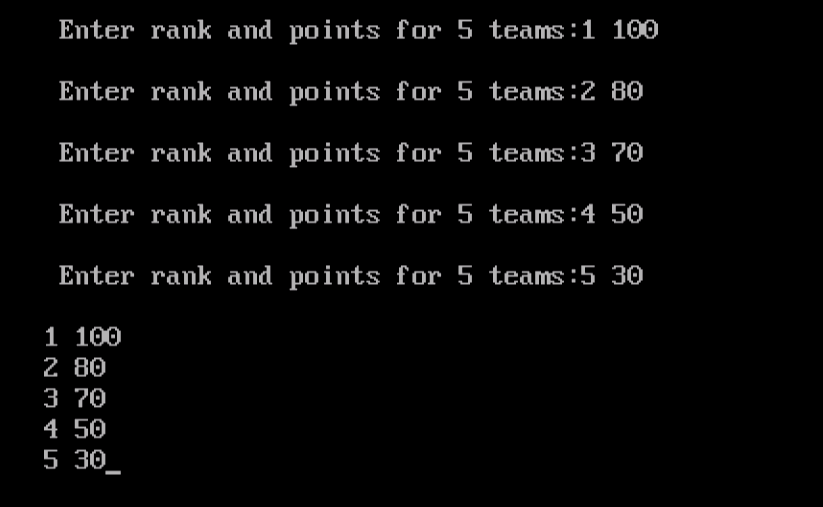
Let’s see a program where we will print the rank and points of 5 cricketers. Here, you can see an array with rows and columns i.e. two-dimensional.

We’re displaying the rank and points for 5 teams, but before that, you need to input the values. %d is a format specifier and % is a placeholder. Specifying %d means displaying values as integers, so we’ve used it.

Here, cric[0][1] would mean 1st row and 1st column, cric[0][2] is 1st row and 2nd column, etc

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>    void main () {       int cric[4][2];     int i, j;     clrsc();       for (i=0;i<5;i++)     {        printf("\n Enter rank and points for 5 teams:");        scanf("%d %d", &cric[i][0], &cric[i][1]);     }       for(i=0;i<5;i++)        printf("\n%d %d", cric[i][0],cric[i][1]);        getch();  } |

Following is the output,



Above, we saw the following two-dimensional arrays,

