**Arrays**

Table of Contents

[Integer Arrays 2](#_Toc44844797)

[Character Arrays 4](#_Toc44844798)

## Integer Arrays

int ar[10] = {1, 2, 3}; *// this is called initialization  
// prints 1 2 3*

C

If the size of the array is not declared but it is initialized, then the code will still work as the size of the array can be understood.

Array

|  |  |
| --- | --- |
|  | ar[0] |
|  | ar[1] |
|  | ar[2] |
|  | ar[3] |

If an index has no value, it still has memory allocated to it. Calling it will print the value 0 if the array was initialized. However, calling an index that is outside the declared size of an array or printing the array without initializing it will return a random value, whatever was stored in that piece of memory. This value is called a garbage value.

Int arrays written as such are called one dimensional arrays:

int ar[100]

C

Two dimensional arrays are written as such:

int ar[100][100]

C

where the first size is the number of lines and the second size the length of each line.

Any number of dimensions can be created in arrays.

Calling a two-dimensional array that does not have a size allocated to it may or may not cause an error depending on the scenario.

ar[][] = {1, 2, 3, 4} - this will cause an error since there are multiple possible ways to arrange this data

ar[3][3] = {1, 2, 3, 4} - this will print 1 2 3 on line 1, 4 0 0 on line 2 and 0 0 0 on line 3.

ar[3][] = {1, 2, 3, 4} - this will cause an error since there are multiple possible ways to arrange this data

ar[][3] = {1, 2, 3, 4} - this will print 2 lines, each with 3 integers. This works since the number of lines needed can be understood.

## Character Arrays

char str[100] = "string"; *// each character is stored in a separate index*

C

After all the characters are stored, a null character (\0) will be entered automatically in the next index. If a null character is manually entered in the string, any characters entered after the null character will be stored, but will be ignored.

Integer arrays do not have the null character.

Character arrays can be stored as two-dimensional arrays, with each line being a different string.

char ar[10][100] = {"string1", "string2"};  
puts(str[0]); *// prints 1st string (puts is same as printf)*

C

Functions related to character arrays

strlen(str[1]) *// finds length of string*strcpy(str[0], str[1]); *// replaces value of str[0] with value of str[1]*strncpy(str[0], str[1], 4); *// replaces first 4 characters of str[0]  
 // with first 4 characters of str[1]*strcat(str[1], str[0]); *// joins str[0] to end of str[1]*strncat(str[1], str[0], 3); *// first 3 characters of str[0]  
 // are joined to the end of str[1]*

C

For strncat(), strcat(), strncpy() and strcpy(), the second value can be input directly instead of using an index value.

strcmp (str[0], str[1]); *// this compares the two strings lexicographically*

C

Each character has an ANSI value. The values of each character of the strings starting from the first are compared. Whichever string is the first to have a character value greater than that of the other strings, is considered to be greater. So, b > a, bb > ba and so on.

In the above example, if str[0] > str[1], the value returned is 1. If str[0] < str[1], it is -1 and if the two strings are exactly the same with no differences whatsoever, then the value returned is 0.