Arrays in C

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Declaration of C Array

We can declare an array in the c language in the following way.

data_type array_name[array_size];

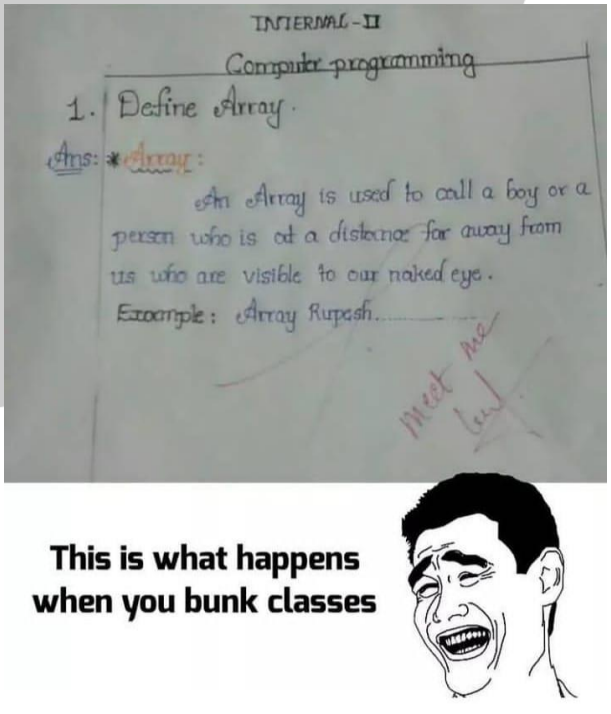
Now, let us see the example to declare the array.

int marks[5];
Here, int is the data_type, marks are the array_name, and 5 is the array_size.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```



C array example

```
#include<stdio.h>
int main(){
int i=0;
int marks[5];//declaration of array
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```

//traversal of array

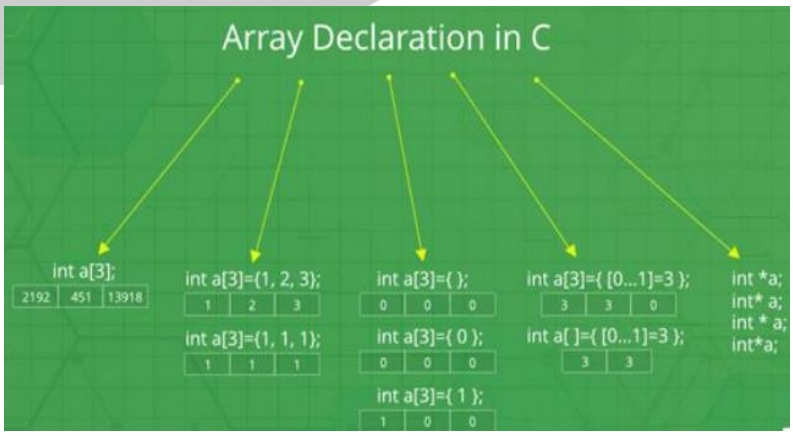
```
for(i=0;i<5;i++){ printf("%d \n",marks[i]);
}//end of for loop
return 0;
}
```

Output

```
80
60
70
85
75
```

ARRAY DECLARATION:

There are various ways in which we can declare an array. It can be done by specifying its type and size, by initializing it or both.



1. **Array declaration by specifying size**

```
// Array declaration by specifying size
int arr1[10];
// With recent C/C++ versions, we can also
// declare an array of user specified size
int n = 10; int arr2[n];
```

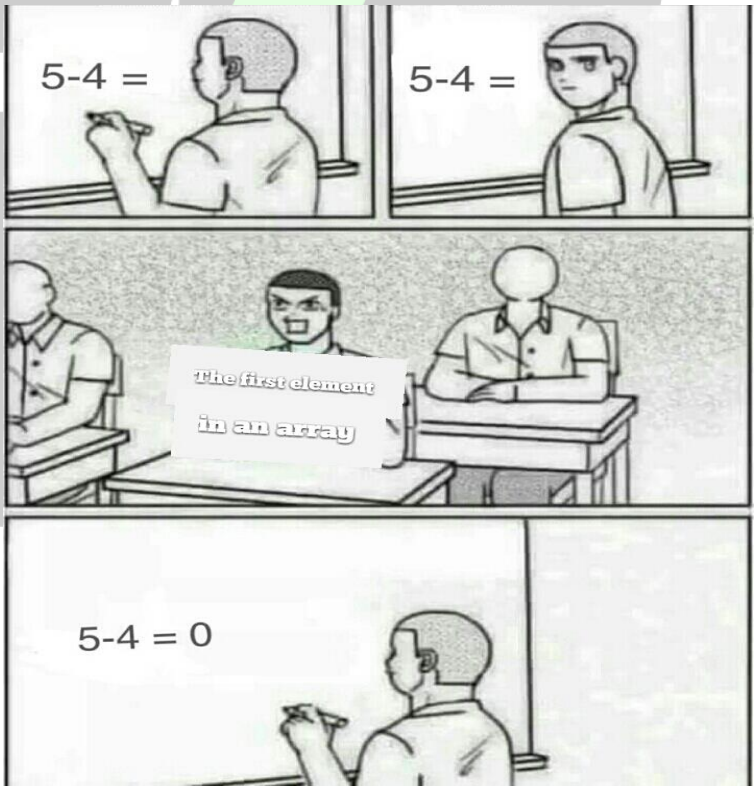
2. **Array declaration by initializing elements**

```
// Array declaration by initializing elements
int arr[] = { 10, 20, 30, 40 };

// Compiler creates an array of size 4.
// above is same as "int arr[4] = {10, 20, 30, 40};"
```

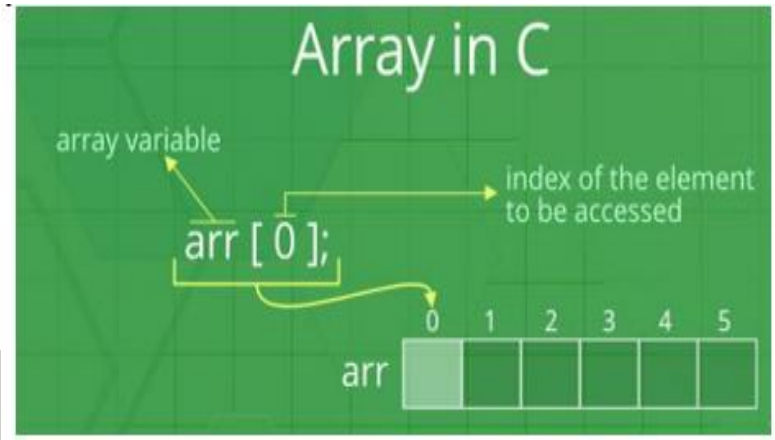
3. **Array declaration by specifying size and initializing elements**

```
// Array declaration by
specifying size and initializing
// elements
int arr[6] = { 10, 20, 30, 40 }
// Compiler creates an array of
size 6, initializes first
// 4 elements as specified by
user and rest two elements as 0.
// above is same as
"int arr[] = {10, 20, 30, 40, 0, 0};"
```



Accessing Array Elements

Array elements are accessed by using an integer index. Array index starts with 0 and goes till size of array minus 1.



C Multidimensional Arrays

In C programming, you can create an array of arrays. These arrays are known as multidimensional arrays. For example, **float x[3][4];**

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

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.

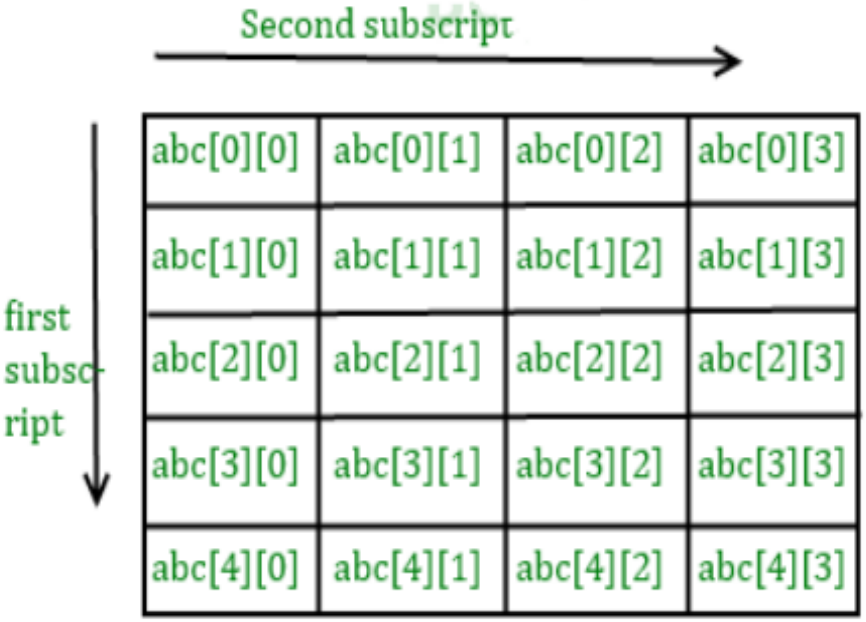
Initialization of a 2d array

```
// Different ways to initialize two-dimensional array.
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[2][3] = {1, 3, 0, -1, 5, 9};
```

Example 1: Two-dimensional array to store and print values

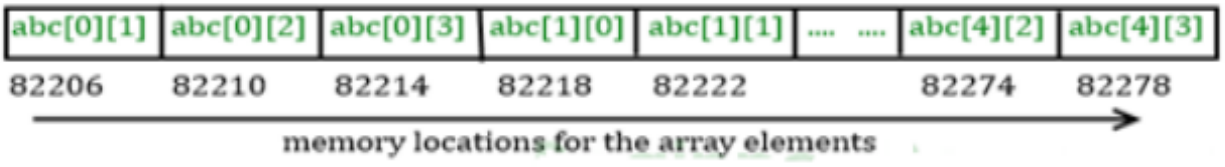
```
// C program to store temperature of two cities of a week and
display it.
#include <stdio.h>
const int CITY = 2;
const int WEEK = 7;
int main()
{
int temperature[CITY][WEEK];
// Using nested loop to store values in a 2d array
for (int i = 0; i < CITY; ++i)
{
for (int j = 0; j < WEEK; ++j)
{
printf("City %d, Day %d: ", i + 1, j + 1);
scanf("%d", &temperature[i][j]);
}
}
printf("\nDisplaying values: \n\n");
// Using nested loop to display values of a 2d array
for (int i = 0; i < CITY; ++i)
{
for (int j = 0; j < WEEK; ++j)
{ printf("City %d, Day %d = %d\n", i + 1, j + 1, temperature[i][j]);
}
}
return 0;
}
```

2D array conceptual memory representation



Here my array is abc[5][4], which can be conceptually viewed as a matrix of 5 rows and 4 columns. Point to note here is that subscript starts with zero, which means abc[0][0] would be the first element of the array.

However the actual representation of this array in memory would be something like



Array is of integer type so each element would use 4 bytes that's the reason there is a difference of 4 in element's addresses.