

# Programming in C

## Arrays

# Array

**s:**

An array in C is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array.

They are used to store similar type of elements as in the data type must be the same for all elements.

They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type.

# Array

s:

The size and type of arrays cannot be changed after its declaration.

**For Example :** If you want to store marks of 100 students you can create an array for it.

```
float marks[100];
```

▪

An array is a collection of data that holds fixed number of values of same data type.

(Or)

# **Some examples where the concept of an array can be used:**

- List of temperatures recorded every hour in a day, or a month or a year.
- List of employees in an organization.
- List of products and their cost sold by a store.
- Test scores of a class of students.
- List of customers and their telephone numbers.
- Table of daily rainfall data.

# How to declare an array in C?

## Syntax:

```
data_type array_name[array_  
size];
```

**For example:**

```
float mark[5];
```

Here we declared an array, mark, of floating-point type and size 5. That it holds floating-point values.

# Elements of an array and How to access them?

You can access elements of an array by indices.

Suppose you declared an array **mark** as above. The first element is **mark[0]**, second element is **mark[1]** and so on.

## Few Key points:

- Arrays have 0 as the first index not 1. In this example, **mark[0]**.
- If the size of an array is **n**, to access the last element, **(n-1)** index is used. In this example, **mark[4]**.
- Suppose the starting address of **mark[0]** is 2120d. Then, the next address, **mark[1]**, will be 2124d, address of **mark[2]** will be 2128d and so on. **Its because the size of a float is 4 bytes.**

# How to Initialize an

**array?** It is possible to initialize an array during declaration.

**For example:**

```
int mark[5] = {9,4,6,3,5};
```

Another method of initialize array during declaration

```
int mark[ ] ={9,4,6,3,5};
```

**Here,**

mark[0] is equal to 9

mark[1] is equal to 4

mark[2] is equal to 6

mark[3] is equal to 3

mark[4] is equal to 5



# Important thing to remember when working with C arrays:

Suppose you declared an array of 10 elements. Lets say,

```
int    testArray[10];
```

You can use the array members from **testArray[0]** to **testArray[9]**.

If you try to access array elements outside of its bound, lets say `testArray[12]`, the compiler may not show any error. However, this may cause unexpected output (undefined behavior).

## Array Declaration in C

2192	451	13918
------	-----	-------

0	0	0
---	---	---

```
int* a;
```

```
int*a;
```

# advantages

- Random access of elements using array index.
- Use of less line of code as it creates a single array of multiple elements.
- Easy access to all the elements.
- Traversal through the array becomes easy using a single loop.
- Sorting becomes easy as it can be accomplished by writing less line of code.

# Disadvantages

- Allows a fixed number of elements to be entered which is decided at the time of declaration. Unlike a linked list, an array in C is not dynamic.
- Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

# **Arrays are of three types:**

1. One-dimensional arrays.
2. Two-dimensional arrays.
3. Multidimensional arrays.

# One-dimensional Array:

A list of item can be given **one variable name** using **only one variable subscript** and such a variable is called a **single subscripted** or a **one dimensional array**.

**The Syntax for an array declaration is:**  
**type variable-name[size];**

## Example:

```
float height[50];  
int  group[10];  
char name[10];
```

The type specifies the type of the element that will be contained in the array, such as **int**, **float**, or **char** and the size indicates the maximum number of elements that can be stored inside the array.

Now as we declare a array

**int number[5];**

Then the computer reserves five storage locations as the size of the array is 5 as shown below.

## Reserved Space Storing values after Initialization

number[0]		number[0]	35
number[1]		number[1]	20
number[2]		number[2]	40
number[3]		number[3]	57
number[4]		number[4]	19

# Initialization of one dimensional array:

After an array is declared, its elements must be initialized. In C programming an array can be initialized at either of the following stages:

**At compile  
time   At run  
time**



# Compile Time

**Initialization:** The form of initialization of array is:

**type array-name[size] = {list of values};**

The values in the list are separated by **commas**.

**For example:**

**int number[3] = {0,5,4};**

size 3 and of will declare the variable 'number' as an array  
will assign the values to each elements.

number of If the number of values in the list is less than  
the initialized. elements, then only that many  
elements will be

automatically. The remaining elements will be set to zero

declared size, Remember, if we have more initializers than the  
the compiler will produce an error.

# Run time

## initialization:

**An array can also be explicitly initialized at run time.** For example consider the following segment of a c program.

```
for(i=0;i<10;i++)  
{  
    scanf("%d", &x[i]);  
}
```

In the run time initialization of the arrays looping statements are almost compulsory.

Looping statements are used to initialize the values of the arrays one by one using assignment operator or through the keyboard by the user.

# One dimensional Array

## Program:

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

```
int main()
{
    int arr[5];
    arr[0] = 5;
    arr[2] = -10;
    arr[3 / 2] = 2; // this is same as arr[1] = 2
    arr[3] = arr[0];

    printf("%d %d %d %d", arr[0],
           arr[1], arr[2], arr[3]);

    return 0;
}
```

# One dimensional Array

## Program:

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

```
int main()
```

```
{
```

```
    int arr[2];
```

```
    printf("%d ", arr[3]);
```

```
    printf("%d ", arr[-2]);
```

```
    return 0;
```

```
}
```

# One dimensional Array

## Program:

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

```
int main()
```

```
{
```

```
    int arr[2] = { 10, 20, 30, 40, 50 };
```

```
    return 0;
```

```
}
```

# One dimensional Array

## Program:

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

```
void main()
```

```
{
```

```
    int array[5];
```

```
    printf("Enter 5 numbers to store them in array \n");
```

```
    for(i=0;i<5;i++)
```

```
    {
```

```
        Scanf("%d", &array[i]);
```

```
    }
```

```
    printf("Element in the array are: \n");
```

```
    For(i=0;i<5;i++)
```

```
    {
```

```
        printf("Element stored at a[%d]=%d \n", i, array[i]);
```

```
    }
```

```
    getch();
```

```
}
```

## **Inpu**

**t:** Enter 5 elements in the array: 23 45 32 25 45

## **Output:**

Element in the array are:

Element stored at a[0]:23

Element stored at a[0]:45

Element stored at a[0]:32

Element stored at a[0]:25

Element stored at a[0]:45

# Two-dimensional Arrays:

The simplest form of multidimensional array is a two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays.

To declare a two-dimensional integer array of size `[x][y]`, you would write something as follows:

```
type arrayName[x][y];
```

Where **type** can be any valid C data type and **arrayName** will be a valid C identifier.



A two-dimensional array can be considered as a table which will have **x number o rows and y number o columns.**

A two-dimensional array **a**, which contains three rows and four columns can be shown as follows.

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Thus, every element in the array **a** is identified by an element name of the form **a[i][j]**.

where 'a' is the **name of the array**, and 'i' and 'j' **are the subscripts that uniquely identify each element in 'a'**.

## 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},  
                {4,5,6,7},  
                {8,9,10,11}  
            };
```

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};
```

# Accessing Two-Dimensional Array Elements:

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];
```

The above statement will take the 4<sup>th</sup> element from the 3<sup>rd</sup> row of the array.

# Accessing Two-Dimensional Array Elements:

\

# Two-Dimensional Arrays

**program:**  
#include <stdio.h>

```
int main()
{
    int a[5][2]={ {0,0},{1,2},{2,4},{3,6},{4,8}};
    int i,j;
    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

# Multi-Dimensional Arrays:

C programming language supports multidimensional Arrays.

- Multi dimensional arrays have more than one subscript variables.
- Multi dimensional array is also called as matrix.
- Multi dimensional arrays are array of arrays.



# **Declaration of Multidimensional Array**

# **Declaration of Multidimensional Array**

# Declaration of Multidimensional

## Array

For a C program accessing  $X[i][j][k]$ , the following intermediate code is generated by a compiler. Assume that the size of an integer is 32 bits and the size of a character is 8 bits. (GATE-CS-2014)

$t0 = i * 1024$   
 $t1 = j * 32$   
 $t2 = k * 4$   
 $t3 = t1 + t0$   
 $t4 = t3 + t2$   
 $t5 = X[t4]$  Which one of the following statement about the source code of C program is correct?

- (A) X is declared as “int X[32][32][8]”
- (B) X is declared as “int X[4][1024][32]”
- (C) X is declared as “char X[4][32][8]”
- (D) X is declared as “char X[32][16][2]”

# Multidimensional Array

For a program

accessing

`X[i][j][k]`, the

following

intermediate

code is

generated by a

compiler.

Assume that the

size of an integer

is 32 bits and the

size of a

character is 8

```
{  
printf("\n a[%d][%d]:",i+1,j+1);  
scanf("%d",&a[i][j]);  
}  
printf("\n Element of B  
matrix"); for(i=0;i<r;++i)  
for(j=0;j<c;++j)  
{  
printf("\n b[%d][%d]:",i+1,j+1);  
scanf("%d",&b[i][j]);  
}  
printf("\n Addition of matrix");  
for(i=0;i<r; i++)
```

```
{
for(j=0;j<c;j++)
sum[i][j]=a[i][j]+b[i][j
];
}
for(i=0;i<r;i++)
{
printf("\n");
for(j=0;j<c;j++)
)

{
printf("\t%d",sum[i][j])
;
}
```



2

11

9

12



**Thank You**