Arrays In C Language

Use of array in programming :

Or

Why we need array in C?

The fundamental data types, namely char int, float, and double are used to store only one value at any given time. Hence these fundamental data types can handle limited amounts of data.

In some cases we need to handle large volume of data in terms of reading, processing and printing. To process such large amounts of data, we need a powerful data type that would facilitate efficient storing, accessing and manipulation of data items. For example: If the user wants to store marks of 100 students. This can be done by creating 100 variables individually but, this process is rather tedious and impracticable. This type of problem can be handled in C programming using arrays.

ARRAY / VECTOR :

C Array is a collection of variables belongings to the same data type. You can store group of data of same data type in an array.(Array is Collection of symilar datatype that stores in continues memory location).

(Or)

An array is collection of homogeneous elements in a single variable.

- Array might be belonging to any of the data types
- Array size must be a constant value.
- Always, adjacent memory locations are used to store array elements in memory.
- Individual values are called as elements.
- It allocates sequential memory locations.

Types of Arrays :

We can use arrays to represent not only simple lists of values but also tables of data in two or three or more dimensions.

- **One–dimensional arrays**
- **Two–dimensional arrays**
- **Multi-dimensional arrays**

One - Dimensional Arrays

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

Declaration of One-Dimensional Arrays :
Like any other variables, arrays must be declared before they are used.
The general form of array declaration is

Syntax :

```
<data type> <array name>[sizeofarray];
```

The data type specifies the type of element that will be contained in the array, such as int, float, or char.

The size indicates the maximum number of elements that can be stored inside the array.

For example :

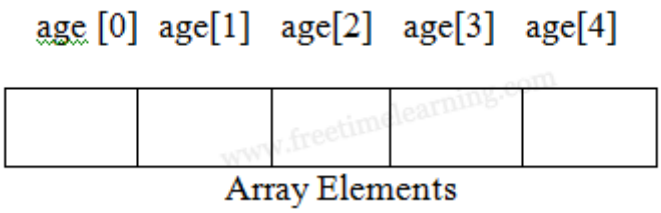
```
int age[5];
```

Here, the name of array is age. The size of array is 5, i.e., there are 5 items (elements) of array age. All elements in an array are of the same type (int, in this case).

Array elements :
Size of array defines the number of elements in an array. Each element of array can be accessed and used by user according to the need of program.

For example :

```
int age [5];
```



Array Elements

Note that, the first element is numbered 0 and so on.(size-1)

The first index is called **Lower Bound**, and the last index is called an **Upper Bound**. Upper Bound of a one dimensional is always Size – 1.

Here, the size of array age is 5 times the size of int because there are 5 elements.

Suppose, the starting address of age [0] is 2120d and the size of int be 2 bytes. Then, the next address (address of a [1]) will be 2122d, address of a [2] will be 2124d and so on.

Initialization of one-dimensional array :

After an array is declared, its elements must be initialized. Otherwise they will contain "garbage". We can initialize the elements of arrays in the same way as the ordinary variables when they are declared.

The general form of initialization of array is .

Syntax :

```
datatype array_name[size] = { list of values };
```

The values in the list are separated by commas.

You can initialize array in C either one by one or using a single statement as follows :

Array_name[0]=54;

Array_name[1]=32;

Array_name[2]=87;

Array_name[3]=90;

…….till size-1;

Or

int age[5]={2,6,34,5,1};

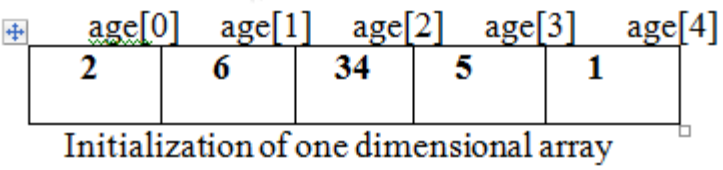
It is not necessary to define the size of arrays during initialization.

int age[]={2,6,34,5,1};

In this case, the compiler determines the size of array by calculating the number of elements of an array.

Following is an example to assign a single element of the array :

```
int age[2]=34;
int age[3]=5;
```



Initialization of one dimensional array

Program : To find out the largest and smallest element in an array.

```
#include<stdio.h>
void main()
{
        int a[5]={10,20,55,22,88},i;
        clrscr();
        for(i=0;i<5;i++)
        {
                printf("\n A[%d]%d",i,a[i]);
        }
        getch();
}
```

Output :

A[0]=10

A[1]=20

A[2]=55

A[3]=22

A[4]=88

Two - Dimensional Arrays

There could be situations where a table of values will have to be stored.
Consider a student table with marks in 3 subjects.

Student	Mathematics	Physics	Chemistry
Student # 1	89	77	84
Student # 2	98	89	80
Student # 3	75	70	80
Student # 4	60	75	80
Student # 5	84	80	75

• The above table contains a total of 15 values.

• We can think this table as a matrix consisting of 5 rows & 3 columns.

• Each row represents marks of student # 1 in all (different) subjects.

• Each column represents the subject wise marks of all students.

• In mathematics we represent a particular value in a matrix by using two subscripts such as Vij. Here V denotes the entire matrix Vij refers to the value in " i "th row and " j "th column.

Example :

In the above table V23 refers to the value "80".

C allows us to define such tables of items by using two-dimensional arrays.

Definition :

A list of items can be given one variable name using **two subscripts** and such a variable is called a two – subscripted variable or a two – dimensional array.

Two – Dimensional arrays can be declared as.

Syntax :

<datatype> <variable-name>[row-size][column-size];

The above table can be defined in “C” as

The above table can be defined in “C” as

Representation of a two Dimensional array in memory :

	Column-0	column-1	column-2
	[0][0]	[0][1]	[0][2]
Row-0	89	77	84
	[1][0]	[1][1]	[1][2]
Row-1	98	89	80
	[2][0]	[2][1]	[2][2]
Row-2	75	70	80
	[3][0]	[3][1]	[3][2]
Row-3	60	75	80
	[4][0]	[4][1]	[4][2]
Row-4	84	80	75

Initializing Two- Dimensional Arrays :

• Like the one-dimensional arrays, two-dimensional arrays may be initialized by following their declaration with a list of initial values enclosed in braces.

int table[2] [3] = {0,0,0,1,1,1};

• This initializes the elements of first row to zero and the second row to one.

• This initialization is done row by row.

• The above statement can be equivalently written as

int table[2][3] ={{0,0,0},{1,1,1}};

• we can also initialize a two-dimensional array in the form of a matrix as shown.

int table[2][3] = {
{0,0,0},

{1,1,1}
};

• Commas are required after each brace that closes of a row, except in case of last row.

• If the values are missing in an initialize, they are automatically set to zero.

Ex: int table [2] [3] = {
{1,1}, 1 1 0
{2} 2 0 0
};

Program : Write a program to display the elements of a two dimensional array.

```
# include<stdio.h>
# include<conio.h>
main( ) {
        int  i,j;
        int  a[3][3] = { { 1,2,3}, {4,5,6}, {7,8,9}};
        printf("elements of an array \n \n");
        for( i=0; i<3; i++)  {
            for ( j=0; j<3; j++){
                    printf ("%d\t", a[ i ][ j ]);
            }  printf("\n");
         }
getch();
}
```

Output :

```
Elements of an Array
1   2   3
4   5   6
7   8   9
```

Program : To perform addition and subtraction of two matrices.

```
# include<stdio.h>
 # include<conio.h>
 main( )
  {
 int i,j,r1,c1, a[10][10], b[10][10];
 clrscr( );
 printf("Enter Order of Matrix A & B up to 10 x 10:");
 scanf("%d  %d", &r1, &c1);
 printf("Enter Elements of Matrix of A: \n");
```

```
for( i=0; i<r1; i++)
{
    for( j=0; j<c1; j++)
    scanf(" %d ", &a[ i ][ j ]);
}
printf("Enter Elements of Matrix of B: \ n");
for( i=0; i<r1; i++)
 {
   for( j=0; j<c1; j++)
  scanf(" %d ", &b[ i ][ j ]);
 }
printf("\n Matrix Addition \n");
for( i=0; i<r1; i++)
 {
   for( j=0; j<c1; j++)
    printf("%d\t", a[ i ][ j ] + b[ i ][ j ]);
    printf (" \n");
 }
printf("n Matrix Subtraction \n");
for( i=0; i<r1; i++)
 {
   for( j=0; j<c1; j++)
   printf("%d\t", a[ i ][ j ] – b[ i ][ j ]);
   printf("\n");
 }
getch( );
 }
```

Output :

```
Enter order of Matrix A & B up to 10 x 10: 3   3
Enter Elements of Matrix of A :
4   5   8   2   9   8   2   9   4
Enter Elements of Matrix of B :
1   3   5   0   5   4   6   7   2
Matrix Addition

5   8   13
2   14   12
8   16   6
```

Matrix Subtraction

3 2 3
2 4 4
-4 2 2

Multi - Dimensional Arrays

A list of items can be given one variable name using **more than two subscripts** and such a variable is called Multi-dimensional array.
Three Dimensional Arrays :
The Three dimensional array can be declared as

Syntax :

```
<datatype> <array_name>[sizeofno.oftwoDimArray] [sizeofrow] [sizeofcolumn];
```

Initializing Three- Dimensional Arrays :

• Like the one-dimensional arrays, three-dimensional arrays may be initialized by following their declaration with a list of initial values enclosed in braces.

int table[2][2][3] = {0,0,0,1,1,1,6,6,6,7,7,7};

• This initializes the elements of first two dimensional(matrix) first row to zero's and the second row to one's and second matrix elements are first row to six's and the second row to seven's.

• This initialization is done row by row.

• The above statement can be equivalently written as

int table[2][3] = {{{0,0,0},{1,1,1}},{{0,0,0},{1,1,1}}}

• We can also initialize a two – dimensional array in the form of a matrix as shown.

```
int table[2][3] = {
    {
      {0,0,0},
            {1,1,1}
     },
     {
      {6,6,6},
            {7,7,7}
      }

   } ;
```

Advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

Disadvantage of C Array

1) **Fixed Size**: Whatever size, we define at the time of declaration of the array, we can't exceed the limit.