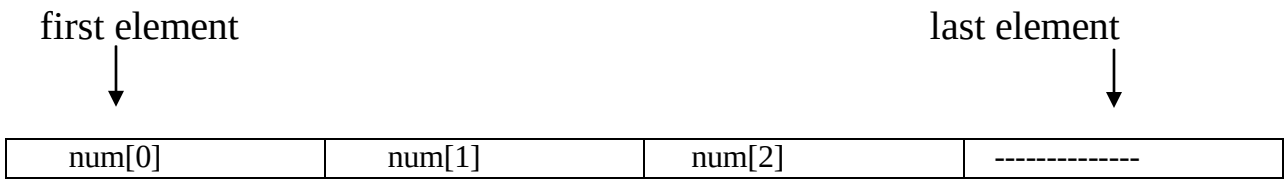
Array

•An array is a group of similar data items sharing a common name. Each element of an array has a unique Index number.

• Array is always stored at Contiguous (ones after the other) memory location

• The lowest address corresponds to the first element and the highest address to the last element.

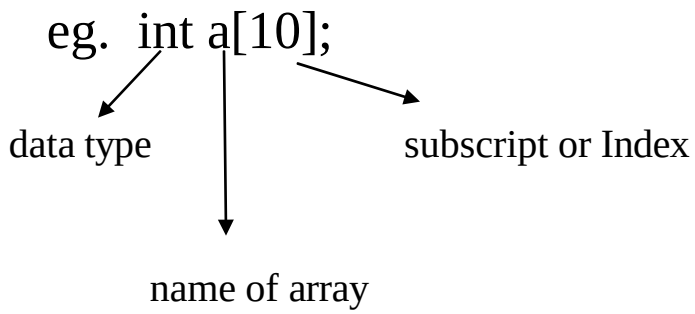


• Array is a derived data type.

• It is a collection of the same data elements. Each element of an array has a unique index number. Each element of an array can be accessed with the combination of array name and the Index number.

•An array with a single index number is called single dimensional array and the array with two subscripts for a single element is called two dimensional arrays.

The index is enclosed in brackets after array name



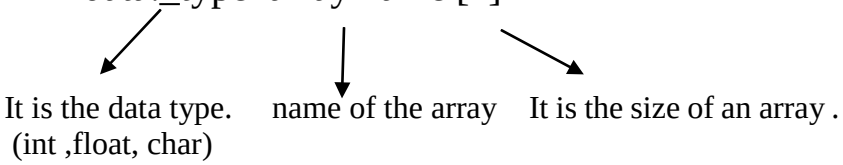
Array Declaration:

Like other variables, an array has to be declared before it is used to declare an array, we must provide the following information:-

1. Data type of an array.
2. The name of the array.
3. The number of indexes (subscripts) in an array.
4. The maximum value of each subscript.

Syntax for declaring an array

data._type array name [n]



eg(i) int a [10]; (one-dimensional array because of one subscript)

- int specifies the type of variable.
- a is the name of the array variable.
- [10] is the subscript
- 10 elements stored in this array.

eg(ii) int a [5] [6]; (two dimensional array because of two subscript)

Types of Array

1 Single dimensional Array. Or One dimensional array.
2. Two dimensional Array.
3.Multi dimensional Array.

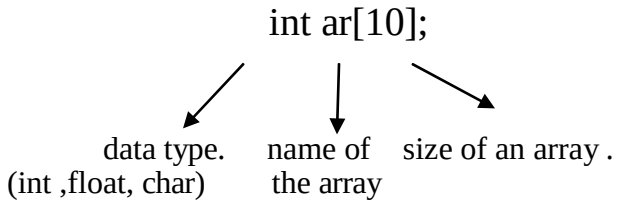
1. Single dimensional array

A Single dimensional array has only a single Subscript. The subscript number can identify the number of individual elements in the array. They take contiguous memory locations either Column wise or row wise.

Syntax:-

<datatype><arrayname> [size of an array];

for example



Memory Representation of Single Dimensional Array

int ar [10]={ 20,30,40,50,60,70,80,90,100,110};

Row wise

Storage	Index
20	ar[0]
30	ar[1]
40	ar[2]
50	ar[3]
60	ar[4]
70	ar[5]
80	ar[6]
90	ar[7]
100	ar[8]
110	ar[9]

Column wise

Index	0	1	2	3	4	5	6	7	8	9
Storage	20	30	40	50	60	70	80	90	100	110

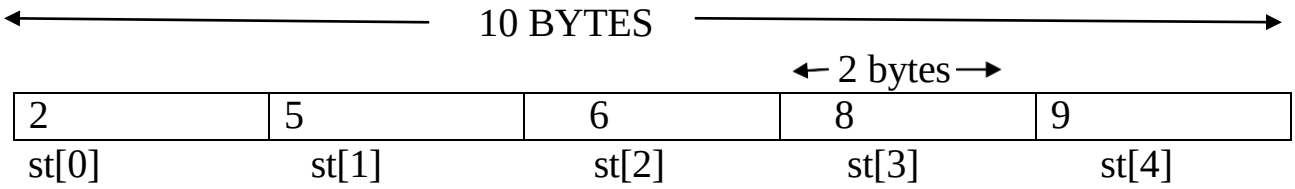
Initialization of One - dimensional Array

One dimensional array can be declared and Initialized at the same place following are the different ways to initialize arrays.

Syntax:

<datatype><name of the array> (size is optional>= {assigned value}

eg1:- int st [5] = {10,20,30,40,50}



eg2:- int i [10] = {10,20,30,40,50}

It will take the first five elements as it is and next five as the Garbage values.

Example of 1-D Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int arr[5];
int i;
for(i=0;i<5;i++)
{
printf("Enter the array elements");
scanf("%d",&arr[i]);
}
printf("\n Printing elements of array");
for(i=0;i<5;i++)
{
printf("%d\t",arr[i]);
}
getch();
}
```

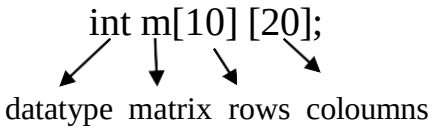
2 Two Dimensional Array

Two-dimensional array identified by two subscripts - A 2-d array is also referred as a matrix or grid. A matrix has two subscripts. One subscript tells us the number of rows and the second subscript tells us about the columns.

Syntax

<datatype><arrayname>[no of rows][no of coloumns];

for example



Here m is declared as a matrix having 10 rows and 20 Columns.

Memory Representation of Two-dimensional Array

Memory does not have any concept of rows and columns. It stores the elements in one continuous chain whether it is one-dimensional array or more than one dimensional array.

int at2] [3] = {10,20,30,40,50,60}

Memory map

	Col 1	Col 2	Col 3
Row 0	10	20	30
Row 1	40	50	60

Memory Map in actual Memory

10	a[0][0]
20	a[0][1]
30	a[0][2]
40	a[1][0]
50	a[1][1]
60	a[1][02]

Initialization Of Two dimensional Array

The general form of initializing 2-D array is :
data-type array-name [row size] [column Size]{{list of row values}, {list of Col values}};

eg1:- int st[2][3]={0,0,0,1,1,1};
OR
int st[2][3]={ {0,0,0},{1,1,1})
Note when we Initializing 2-c array, we have to mention the second dimension. Whereas first dimension is optional.
Example int [2] [ ] = {10,20,30} // Never work
int [] [3] = {10,20,30} // Yes

Example of 2-D Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int arr[2][2];
int I,j;
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("Enter the array elements");
scanf("%d",&arr[i][j]);
}
printf("\n");
}
printf("\n Printing elements of array");
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("%d\t",arr[i][j]);
}
printf("\n");
```

```
}
getch();
}
```

Multidimensional Array

- A multidimensional array has more than one subscript .A two-dimensional array has two subscripts, a three dimensional array has three subscripts and so on. There is no limit to the number of dimensions the C array can have.
- Two dimensional array and more than two dimensional arrays are under the category of multidimensional array.
- C allows more than 2-d arrays but in practice more than two dimensions are rarely used.
- Multidimensional array can be used in the same manner as two-dimensional arrays.

The General form of a Multidimensional Array: -

type array name [s1] [s2]... [sn];

The number of size specifies depend upon the dimension of the array

for example

```
int a [3] [5] [12];
float num [5] [4[5] [3] ;
```

- a is a three dimensional array.
- num is four dimensional array.

WAP to find the sum of two Matrices 


```
#include <stdio.h>
#include<conio.h>
void main()
{
 intat2] [2], b[2] [2], i, j;
 printf(“Enter the elements of A matrix \n”);
for (i=0; i< 2; i++)
{
for (j = 0; j <2; j++)
{
 printf ("Enter the number");
scanf ("%d",&a[i][j]);
}
}
printf(“ Enter the elements of B matrix \n”);
for (i=0; i< 2; i++)
{
for (j = 0; j <2; j++)
{
printf("Enter the number");
scanf("%d",&b[i][j]);
}
}
printf ("Addition of A and B Matrix \n”);
for (i=0; 1<2; i++)
{
 for(j=0; j <2; j++)
{
printf ("%d", a[i][j] + b[i][j]);
}
printf ("\n");
}
 getch();
}
```

Advantages of Array: -

1. It is the simplest kind of data Structure.
2. The calculation of matrices become easy with the use of arrays
3. Arrays make loops very effective.
4. We can refer to a group of elements by a single name.
5. The method of Accessing array elements is very simple.
6. Easy to declare array elements.
7. All array elements can store and accessed with a single variable name.

Disadvantages of Array:

1. Sometimes It's not easy to work with many Index arrays.
2. The size of the array should be known in advance.
3. Wastage of Memory
4. Arrays will always hold similar kinds of information So it is impossible to store unrelated information to the array.
5. Array are largely Static

Passing Array to function

We Pass arrays to function when we need to pass a list of values to a given function.

The array elements can be passed to a function by two methods.

1. Calling the function by value
2. Calling the function by reference

1 Calling the function by value:- In call by value Method, we pass the values of array elements to the function .

Example:

```
#include  <stdio.h>
#include <conio.h>
void show (int a)
{
printf("%d\t", a);
}
void main()
{
clrscr();
 int a[]={ 10,20,30,40 60};
printf ("The elements of Array : \n");
for (int i = 0; i < 5; i++)
{
show (a[i]);
}
getch();
}
```

2. Calling the function by Reference -

In call by Reference, we pass the address of array elements to the function.

Example:

```
#include <stdio.h>
#include <conio.h>
void show (int *a)
{
printf("%d\t", a);
}
```

```
void main()
{
clrscr();
 int a[]={ 10,20,30,40 60};
printf ("The elements of Array : \n");
for (int i = 0; i < 5; i++)
{
show (&a[i]);
}
getch();
}
```

Pass Entire Array to the function

In place of passing individual elements of an array one by one to a function, we can also pass an entire array to a function.

Example

```
#include <stdio.h>
#include <conio.h>
void main()
{
int arrl[]={10,20,30,40,50};
 clrscr();
 show (arr,5);
getch();
}
show (int *a, int size)
{
int i;
 for (i = 0; i<size; i++)
{
printf ("\n%d",*a);
a++;
}
}
```