Array:-

An **array** consists of a set of objects (called its **elements**), all of which are of the same type and are arranged contiguously in memory.
In general, only the array itself has a symbolic name, not its elements. Each element is identified by an **index** which denotes the position of the element in the array. The number of elements in an array is called its **dimension**. The dimension of an array is fixed and predetermined; it cannot be changed during program execution.
The following figure shows as integer array called A. This array contains 10 elements. Any one of these elements may be referred to by giving the name of the array. (i.e., here is A) followed by the position number of the particular element in square brackets ([ ]). Normally the position number started with 0, so our array of 10 elements started with the first element A[0] and ended with A[9] because from 0 to 9 means 10 elements. In general, the ith element of array A is referred to as A[i-1] and the position number specifies the number of elements from the beginning of the array. It is more formally called a ***subscript***.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
1	2	3	4	5	6	7	8	9	10

If we want to print the first element in array A, we would write:-
cout << A[0]<< endl;
If we want to print the sum of the values contained in the first three
elements of array A. we would write:-
cout <<A[0]+A[1]+A[2]<<,endl;

Array declaration

Like other variables, an array may have an initializer. Braces are used to specify a list of comma-separated initial values for array elements. For example:-
int nums[3] = {5, 10, 15};
initializes the three elements of nums to 5, 10, and 15, respectively. When the number of values in the initializer is less than the number of elements, the remaining elements are initialized to zero:
int nums[3] = {5, 10}; // nums[2] initializes to 0

Single dimensional array

The C++ arrays are declared with the data type name and the number of elements inside the square brackets. The general format is:-

Data type array name [number of element];

Where:-
_ Data type is any type that already exists such as (int, float, char.....)
_ array name is the name of the array (any legal identifier)
_ size(number of element) is the number of elements in the array.

Examples
char fname[24]; // an array named fname with 24 chars
int grade[35]; // an array named grade with 35 ints
int pixel[1024*768]; // an array named pixel with 1024*768 ints
double B[5];
The elements in an array are numbered from zero to one less than the size of the array.
For example, the first variable in the B array (the first element of the array) is B [0]. The last element is B [4]. The number in square brackets, the subscript, should be either an integer constant or variable, or an expression whose value is an integer.
Figure shows a diagram of how the array looks in memory.

2	70	7	678	653
B [0]	B [1]	B [2]	B [3]	B [4]

So for example, to set the third element to 7, we may write:-
B [2]=7;
We can declare several arrays with single declarations.
int b[100], c[10];

Things to remember about arrays:

1- The starting index of an array is 0, not 1.
2- The last index is one less than the size of the array.
3- If the array has size elements, the range is 0 to size-1.
4- Arrays contain data of a single type.
5- An array is a sequence of consecutive elements in memory and the start of the array is the address of its first element.
6- Because the starting address of the array in memory is the address of its first element, and all elements are the same size and type, the compiler can calculate the locations of the remaining elements. If B is the starting address of the array , and each element is 4 bytes long, the elements are at addresses B, B + 4, B + 8, B + 12, and so on, and in general, element array[k] is at address B + 12k.

Initializing Declarations:-

Arrays can be initialized at declaration time in two different ways:-

- **Data type array name [number of element] = { list with <= number of elements };**
- **Data type array name[ ] = { list with any number of values or elements };**

Examples:-

```
const int SIZE = 100;
double numbers[SIZE]; // not initialized
string fruit[NUMFRUIT] = {"apple","pear","peach","lemon","mango"};
int power[ ] = {0,1,2,4,9,16,25,36,49,64,81,100};
int counts[SIZE] = {0};
```

The first declaration declares but does not initialize the array named numbers. The next declares and initializes an array named fruit with five strings:-

apple	pear	peach	lemon	mango
0	1	2	3	4

The third initializes an array named power, whose size is determined by the number of values in the bracedelimited list. When the array size is given in square brackets but the number of values in the list is less than the size, the remainder of the array is initialized to 0. In the fourth example, all elements will be set to 0, and in the last, the first three are set to 1 and the rest, to 0.

Rules

- If the array size is given in brackets, then the initialized list must have at most that many values in it.
- If it has fewer, the rest of the array is initialized to 0's.
- If the size is not given in brackets, then the size is equal to the number of elements in the initializer list.
- _ If there is no initializer list, none of the array elements are initialized. They are not set to 0.

To initialize a ten-element integer array n to zero we will do that by the following:-

```
int n[10]={0};
```

Example:

```
#include <iostream.h>
int main()
{
int n[10]={0};
for (int i=0;i<10;++i)
cout <<n[i]<<endl;
return 0;}
```

Also we can initialize array elements to integer values as follows:

```
int n[6]={2,10,35,40,9,12};
```

the same as:

```
int n[]={2,10,35,40,9,12}; //Would create a six-element array.
```

Example: w.p. to reads one value at a time from the keyboard and stores the value in array n[5], and print its element out.

Solution:-

```
#include <iostream.h>
int main()
{
int n[5];
cout <<"Enter an integer elements:"<<endl;
for (int i=0; i<5;i++)
{
cin>>n[i];
}
for(int j=0; j<5; j++)
cout <<"Element "<<j<<" is "<<n[j]<<endl;
return 0;}
```

Example: assume that A[10] is an integer array its elements value is (1,2,3,4,……..10) , w.p. to print:-

1. The even value.

2. The summation of odd value.

Solution:

```
#include <iostream.h>
int main()
{
int sum=0,i;
int a[]={1,2,3,4,5,6,7,8,9,10};
cout <<"The even values is"<<endl;
for (i=0; i<10; i++)
{
if (a[i]%2==0)
cout << a[i]<<endl;
else if (a[i]%2!=0)
sum=sum+a[i];
}
cout <<"The summation of odd values is "<<sum<<endl;
return 0;
}
```

Example:- w.p. that calculates the sum and average of an initialized integer array b[5] = {9, 3, 11, 7, 1}

Solution:-

```
#include <iostream.h>
int main()
{
int sum=0;
int b[5]={9, 3, 11, 7, 1};
 for (int i=0; i<5; i++)
sum=sum+b[i];
cout <<"The summation is "<<sum<<endl;
cout<<" Average is" << sum/ 5;
return 0;
}
```

Multiple –subscripted Arrays:-

Arrays in C++ can have multiple subscripted (dimensions). A common use of multiple dimensions arrays is to represent tables of values consisting of information arranged in *row* and *columns*. Tables or arrays that require two subscript to identify a particular element are called double-subscripted arrays. C++ compilers support at least 12 array subscripts.
Ex:
Two dimensional array b[2][2] could be declared and initialized with:
int b[2][2]={{1,2},{3,4}};
it means that b is two dimensional array with (2) rows and (2) columns and will refer to any element by its position number of the particular element:
cout <<b[0][1]
that means print the element exist in row no.1 and column no.2.

$$b = \begin{array}{c} \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} \end{array} \begin{matrix} 0 \\ 1 \\ 2 \end{matrix}$$

$$b = \begin{bmatrix} 5 & 4 & 2 \\ 3 & 1 & 7 \\ 2 & 9 & 6 \end{bmatrix}$$

```
cout <<b[0][0]          //5
cout <<b[2][1]          //9
cout <<b[2][2]          //6
cout <<b[1][2]          //7
```

For read and print the two dimensional arrays we must use *nested for* as follow:

```
#include <iostream.h>
int main()
{
int a[3][3];
for (int i=0; i<3; i++)
{
for (int j=0; j<3; j++)
cin >> a[i][j];
}
for (int n=0; n<3; n++)
{
for (int m=0; m<3; m++)
cout << a[n][m]<<"\t";
cout <<endl;
}
return 0;
}
```

The output will be:

```
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
```

Initialization of two dimensional array:

$$A = \begin{bmatrix} 7 & 9 & 8 & 1 \\ 2 & 3 & 4 & 5 \\ 7 & 5 & 7 & 8 \end{bmatrix}$$

```
int A[3][4]={{7,5,7,8},{2,3,4,5},{7,9,8,1}}
```

$$b = \begin{bmatrix} 3 & 4 \\ 1 & 0 \end{bmatrix}$$

```
int b[2][2] ={{1},{3,4}};
```

Example: w.p. to add two matrices A(3×4) and B(3×4) and the result will be stored in matrix C. assume that the elements of A and B is (1,2,3,4,5,…..12)

Solution:

```
#include <iostream.h>
int main()
{
int A[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int B[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int C[3][4];
for(int i=0;i<3;i++)
{
for(int j=0; j<4;j++)
{
C[i][j]=A[i][j]+B[i][j];
cout<<C[i][j]<<"\t";
}
cout<<endl;

}

return 0;}
```

The output will be:

```
 2    4   6   8
10  12  14  16
18  20  22  24
```

Example: w.p. to find the average of each row of a matrix:

$$A = \begin{bmatrix} 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Solution:

```
#include <iostream.h>
int main()
{
int A[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int sum;
for(int i=0;i<3;i++)
{
sum=0;
for (int j=0;j<4;j++)
sum+=A[i][j];
cout <<float (sum)/4.0<<endl;
}
return 0;}
```

Example: if you have given matrix

$$a = \begin{bmatrix} 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 8 & 9 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 8 & 9 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

w.p. to calculate the following:

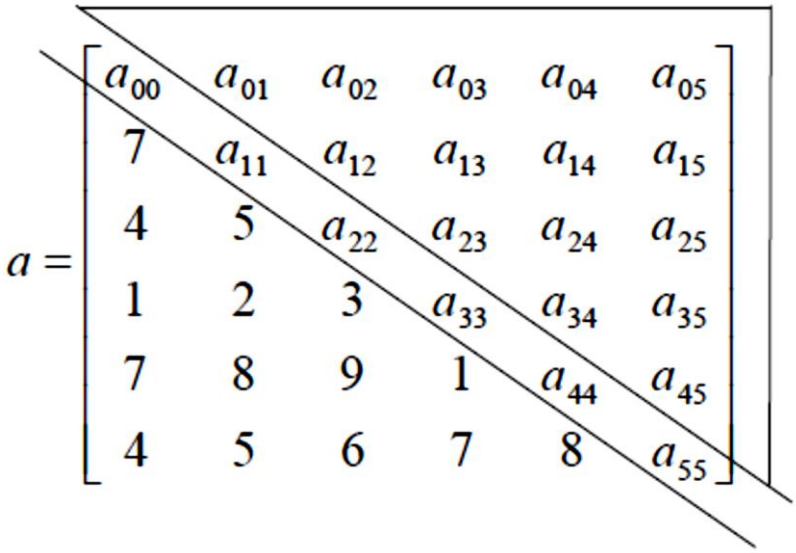
1. The summation of the primary diagonal of matrix.
2. The summation of the upper triangular matrix.
3. The summation of the lower triangular matrix.

Solution:

```
#include <iostream.h>
int main()
{
int
a[6][6]={1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,
7,8,9,1,2,3,4,5,6,7,8,9};
int sum=0;
for(int i=0;i<=5;i++)
for(int j=0;j<=5;j++)
if(i==j)
sum=sum+a[i][j];
cout <<"The sum of primary diagonal ="<<sum<<endl;
sum=0;
for(int n=0;n<=5;n++)
for(int m=0;m<=5;m++)
if(n<m)
sum=sum+a[n][m];
cout <<"The sum of upper triangular matrix ="<<sum<<endl;
sum=0;
for(int x=0;x<=5;x++)
for(int y=0;y<=5;y++)
if(x>y)
sum=sum+a[x][y];
cout <<"The sum of lower triangular matrix ="<<sum<<endl;
return 0;}
```

The output will:

```
The sum of primary diagonal =30
The sum of upper triangular matrix =73
The sum of lower triangular matrix =77
```



Example:- W.P. that exchanges row4 with row2 in a (4×4) integer matrix input by the user?

Solution:-

```
#include <iostream.h>
int main()
{
int A[4][4];
cout<< "Enter 4×4 integer matrix:";
for (int i=0; i<4; i++)
for(int j=0; j<4; j++)
cin>> A[i][j];
for(i=0; i<4;i++)
{
int temp=A[1][i];
A[1][i]=A[3][i];
A[3][i]=temp;
}
cout<<"matrix after exchanging="<<endl;
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
cout<<A[i][j]<<" ";
cout<<endl;
}
}
return 0;}
```

String (1D array of characters):-

String is a character array that is terminated with null. Null is zero and can be expressed as Null or '\0'. The compiler adds the null to the end of string automatically.

Example:- w.p that reads a string and then computes the number of capital letters in the string?
Solution:-

```
#include<iostream.h>
int main()
{
char str[30];
int count=0;
cout<<"Enter your string:";
cin>> str;
for(int i=0; str[i]; i++)
if(str[i]>='A'&& str[i]<='z')
count++;
cout<<"no. Of capital letters is"<<count<<endl;
}
return 0;}
```

Array of strings (2D array of characters):-

To creat an array of strings, we use a 2-D character array. The number of rows determines the number of strings and the number of columns specifies the maximum length of each string:-

Char a[30][80];

Char day[7][10]={ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday");

	0	1	2	3	4	5	6	7	8	9
0	S	U	N	D	A	Y	0			
1	M	O	N	D	A	Y	0			
2	T	U	E	S	D	A	Y	0		
3	W	E	D	N	E	S	D	A	Y	0
4	T	H	U	R	S	D	A	Y	0	
5	F	R	I	D	A	Y	0			
6	S	A	T	U	R	D	A	Y	0	

Example:- use array of string to w.p. that prints the week days?
Solution:-

```
#include<iostream.h>
int main()
{
char day[7][10]= { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday");
for (int i=0; i<7;i++)
cout<< day[i]<,endl;
}
return 0;}
```