Arrays in C++

Array

The array is a group(collection) of similar type element by sharing a common name, that are differentiated from one another by their position in the array .OR The array is a collection of similar type element stored under a unique name and can be accessed by index value.

Example If we want to store the marks of a student in 6 different subject at that moment we require 6 variables to store that value instead of that by using the array we can store all these detail using the single variable.

The array can be classified into 3 different types. They are

1) One dimensional array
2) Two dimensional array
3) Multi dimensional array

One dimensional array

A list of data item can be given one variable name using one subscript and such a variable is called one dimensional array OR It is an array can be accessed by single index value.

Declaration of array

Before using the array we have to declare it the syntax to declare the array is

data-type variable-name[max-size];

Example: int marks[6];

In array we can access the element by using its position value and the position value is known as array subscript and is always starts with value zero and ends with maximum size-1. While specifying the max-size it should always either integer value.

Eg. Program to accessing the array elements OR Program to read and print one dimensional array

```
#include<iostream.h>
void main()
{
int a[10]; // array declaration
int i;
// reading the 10 array elements
cout<<"enter 10 numbers"<<endl;
for(i=0;i<10;i++)
cin>>a[i];
//  to print the 10 array elements
for(i=0;i<10;i++)
cout<<a[i]<<endl;
}
```

Initialization of array : an array can be initialized along with its declaration

syntax: data_ type array _name [size]={val1, val2 , …};

Example: float height[4]={4.5,5.7,5.5,5.6};

In the above example the value 4.5 is stored in height[0] and 5.6 is stored at height[3] as shown in below figure

height

4.5	5.7	5.5	5.6	
0	1	2	3	← position value

Note: If there is less number of elements specified than the size of the array then the elements are filled by the 0 by the compiler

Example float height[7]={ 4.5, 5.7,5.5,5.6};

In the above example the value 4.5 is stored in height[0] and 5.6 is stored at height[3] and zero is stored from height[4] to height[6] as shown in below figure

height

4.5	5.7	5.5	5.6	0.0	0.0	0.0	
0	1	2	3	4	5	6	← position value

Example: float height[ ]={4.5,5.7,5.5,5.6};

In the above example the value 4.5 is stored in height[0] and 5.6 is stored at height[3] the maximum size is automatically assigned by the compiler as shown in below figure

height

4.5	5.7	5.5	5.6	
0	1	2	3	← position value

Two dimensional array: It is an array can be accessed by a pair of index value

Here while storing the values in array we store the values in the form of row and column format i.e. in the table like structure with 2 index value

Declaration of two dimensional array

The declaration of two dimensional array is same as one dimension with one additional subscript

Syntax: **data-type variable-name[row-size][col-size];**

Example: int marks[3][4];

Will declare a table with 3 row and 4 column

Initialization of 2 dimensional array

We can also initialize the 2 dimensional array same as one dimensional array

Syntax **data-type variable-name[row-size][col-size]={v1,v2,v3,….vn};**

Example:

1) int a[2][3]={1,2,3,4,5,6};

The a is a two dimensional array which contain 2 rows and 3 columns and a will assign the value as follows

```
a[0][0]=1   a[0][1]=2    a[0][2]=3
a[1][0]=4    a[1][1]=5    a[1][2]=6
```

```
2)  int a[2][3]={
                    {1,2},
                     {3}
                   };
```

The a is a two dimensional array which contain 2 rows and 3 columns and a will assign the value as follows

```
a[0][0]=1   a[0][1]=2    a[0][2]=0
a[1][0]=3    a[1][1]=0    a[1][2]=0
```

Note: While initializing if any value is missing then they are automatically initialized to zero

Accessing the elements: To read or to print the value of two dimensional array require nested for loop. The outer loop to keep track the row and the inner loop to keep track the column value

Eg. Write a program to read and print the values of two dimensional array or for matrix

```
#include<iostream.h>
```

```
void main()
{
int a[3][3]; // declaration of two dimensional array
int i;

// input or read the value to array

cout<<”enter 9 numbers”<<endl;
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
cin>>a[i][j];
}
}

// output or print the value of array

cout<<” the given elements are\n”;
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
cout<<a[i][j]<<”\t”;
}
cout<<endl;
}
getch();
}
```

Multidimensional array

If an array has more than one dimension then we call it as multi dimensions. The maximum dimension depends on the compiler.

Declaration syntax

data-type array-Name[size1][size2][size3]…[sizen];

Example : int a[2][3][4];

In the above example a is an array with 3 dimension

Memory representation of one dimensional array

The elements of one dimensional arrays are stored in contiguous memory location. So the amount of storage space required to hold an array is directly related to its type and size of the array.

The total size can be calculated using the relation

Total-size=sizeof(data-type)*max-size;

Example int a[6];

to store array a we require total size12 bytes (total-size= sizeof(int)*6)

Example char a[6];

to store array a we require total size 6 bytes (total-size= sizeof(char)*6)