C++ 4_1

4.1 Arrays
Array is a fixed size collection of similar data type items. Arrays are used to store and access group of data of same data type.
Arrays can of any data type. Arrays must have constant size. Continuous memory locations are used to store array.
It is an aggregate data type that lets you access multiple variables through a single name by use of an index. Array index always starts with 0.

Example for Arrays:
int a[5]; // integer array
char a[5]; // character(string) array
In the above example, we declare an array named a. When used in an array definition, the subscript operator ([]) is used to tell the compiler how many variables to allocate. In this case, we're allocating 5 integers/character. Each of these variables in an array is called an element.

Types of Arrays:
One Dimensional Array
Two Dimensional Array
Multi Dimensional Array

1 One Dimensional Array

Array declaration
int age [5];

Array initialization
int age[5]={0, 1, 2, 3, 4, 5};

Accessing array
age[0]; /*0_is_accessed*/
age[1]; /*1_is_accessed*/
age[2]; /*2_is_accessed*/

2 Two Dimensional Array

Two dimensional array is combination of rows n columns.
Array declaration
int arr[2][2];

Array initialization
int arr[2][2] = {{1,2}, {3,4}};

Accessing array
arr [0][0] = 1;
arr [0][1] = 2;
arr [1][0] = 3;

arr [1][1] = 4;

3 Multi Dimensional Array

C++ programming language allows programmer to create arrays of arrays known as multidimensional arrays.
For example:
float a[2][4][3];

Pointer to an array

Please go through pointers chapter first to understand this
An array name is a constant pointer to the first element of the array. Therefore, in the declaration:
double balance[50];

balance is a pointer to &balance[0], which is the address of the first element of the array balance. Thus, the following program fragment assigns p the address of the first element of balance:

```
double *p;
double balance[10];
p = balance;
```

It is legal to use array names as constant pointers, and vice versa. Therefore, *(balance + 4) is a legitimate way of accessing the data at balance[4].

Passing Array To Function

We can pass entire Arrays to functions as an argument.
For eg.

```
#include
void display(int a)
{
  int i;
  for(i=0;i < 4;i++){
    cout << a[i];
  }
}
int main(){
  int c[]={1,2,3,4};
  display(c);
  //Passing array to display.
  return 0;
}
```

Return array from functions

C++ does not allow to return an entire array as an argument to a function. However, You can

return a pointer to an array by specifying the array's name without an index.
If you want to return a single-dimension array from a function, you would have to declare a function returning a pointer as in the following example:

```
int * myFunction()
{
int c[]={1,2,3}
.
.
.
return c
}
```