**Chapter 3 Array and Function**

**Array:**

* Array is the collection of similar datatypes.
* array index starts from 0.
* n number of data will be get using array.

**Advantages of C++ Array**

* Code Optimization (less code)
* Random Access
* Easy to traverse data
* Easy to manipulate data
* Easy to sort data etc.

**Disadvantages of C++ Array**

* Fixed size

**DECLARATION OF AN ARRAY:**

The declaration of an array tells the compiler that, the data type, name of the array, size of the array and for each element it occupies memory space.

**Syntax:**

Data type array name [size];

**INITIALIZATION OF AN ARRAY:**

After declaration element of local array has garbage value. If it is global or static array then it will be automatically initialize with zero. An explicitly it can be initialize that

Data type array name [size] = {value1, value2, value3…}

Example: int arr[5]={20,60,90, 100,120}

**C++ Array Types**

There are 2 types of arrays in C++ programming:

1. Single Dimensional Array
2. Multidimensional Array

## C++ Single Dimensional Array

## One dimensional array are the simplest form of an array in C++ language.

## It can easily declare, initialize, and manipulate a one-dimensional array.

## A one-dimensional array can be a parameter for function and so on.

**EXAMPLES OF 1D ARRAY**

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** arr[5]={10, 0, 20, 0, 30};  //creating and initializing array

        //traversing array

**for** (**int** i = 0; i < 5; i++)

        {

            cout<<arr[i]<<"\n";

        }

}

**MULTIDIMENSIONAL ARRAY**

* The multidimensional array is also known as rectangular arrays in C++.
* It can be two dimensional or three dimensional.
* The data is stored in tabular form (row ∗ column) which is also known as matrix.

**EXAMPLES OF 2D ARRAY**

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** test[3][3];  //declaration of 2D array

    test[0][0]=5;  //initialization

    test[0][1]=10;

    test[1][1]=15;

    test[1][2]=20;

    test[2][0]=30;

    test[2][2]=10;

    //traversal

**for**(**int** i = 0; i < 3; ++i)

    {

**for**(**int** j = 0; j < 3; ++j)

        {

            cout<< test[i][j]<<" ";

        }

        cout<<"\n"; //new line at each row

    }

**return** 0;

}

**FUNCTIONS**

* A function in C++ is a group of statements that together perform a specific task.
* A function can be called many times.

**TYPES OF FUNCTION**

There are two types of functions in C programming:

* **Library Functions**
* **User-defined functions**

**Library Functions**

Its are the functions which are declared in the C++ header files such as ceil(x), cos(x), exp(x), etc.

**User-defined functions**

Its are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.

**HOW TO DECLARE THE FUNCTION**

return\_type function\_name (data\_type parameter...)

{

//code to be executed

}

**EXAMPLES**

Void main()

{

Cout<< “sandhiya”;

}

Here,

* the name of the function is main()
* the return type of the function is void
* the empty parentheses mean it doesn't have any parameters
* the function body is written inside {}

### **call by value**

* **original value is not modified.**
* If you change the value of function parameter, it is changed for the current function only.
* It will not change the value of variable inside the caller method such as main().

### **call by reference**

* original value is modified because we pass reference (address).
* Here, address of the value is passed in the function, so actual and formal arguments share the same address space.
* Hence, value changed inside the function, is reflected inside as well as outside the function.

# **C++ Recursion**

* When function is called within the same function, it is known as recursion in C++.
* The function which calls the same function, is known as recursive function.

## Calling a Function

* While creating a C function, you give a definition of what the function has to do.
* To use a function, you will have to call that function to perform the defined task.
* When a program calls a function, the program control is transferred to the called function.

## Function Arguments

* If a function is to use arguments, it must declare variables that accept the values of the arguments.
* These variables are called the **formal parameters** of the function.

## C++ String

* Strings are used for storing text.
* A string variable contains a collection of characters surrounded by double quotes.

**String function**

1. Str cpy ----->to copy the string
2. Str cat ----->to join the string
3. Str len ---->length
4. Str cmp ----> compare the two string
5. Str chr -----> searches for a character

**Numeric function**

1. abs ----> absolute value
2. ceil---->round off
3. sqrt---->squareroot
4. floor---->return the largest integer
5. atoi---->return the integer value stored as a string.