# Chapter 3

# Functions

## Concept of modular programming

Modular programming is the act of designing and writing programs as functions. A large program is broken down into **sub-programs**(**Modules**).The process of breaking large programs into smaller sub programs is called **modularizatio**n.The subprograms are also called functions.

Functions are classified into two **bulit-in** or **predefined functions** and **user defined functions**.Built-in functions are readily available to the users and are stored in header files.They are a part of C++ Library.User defined functions are written by the user for specific tasks.Two methods are used for modular programming **top-down** and **bottom-up method.**

## Advantages of Modular programming

Following are the advantages of Modular programming,

### a)Reduction in program size.

Modular approach helps to isolate repeated task .Instructions are grouped together to form functions and is invoked by using its name.Thus program size is reduced.

### b)Less chance of error

When size of program is reduced syntax error will be less.Modularity helps to reduce logical error.

### c)Reduce programming complexity

Modularity helps to reduce programm complexity by dividing larger programs into functions.

### d)Improve reusability

Once a function is written it can be used later.It helps to reduce development time.

## Dis-advantages of Modular programming

Modular programming requires more time and memory.

## Functions in C++

A function is a collection of statement that performs a specific task. Functions are classified into two bulit-in or predefined functions and user defined functions.Built-in functions are readily available to the users and are stored in header files.User defined functions are written by the user for specific tasks.

## Role of main( ) function

The main( ) function is a user defined function in C++ .The execution of a C++ program begins and ends at main( ).By default the main ( ) function returns an integer value to the Operating system.

## Predefined functions

Built-in or pre-defined functions are readily available to the users and are stored in header files.They are a part of C++ library.

The values(data) passed into a function is called **argument**s or **parameters**.The result returned(given) by a function is called return value.Some functions do not return any value,where as som efunctions return's value.

For Example sum(10,5)here 10 and 5 are arguments to the function sum.

## User defined functions

User defined functions are written by the user for specific tasks.

## Console functions for character I/O

The console I/O functions are functions which reads directly from the keyboard and outputs to the monitor. These functions require the inclusion of header file cstdio (stdio.h in Turbo C++ IDE) in the program.

getchar( ) function:The getchar( ) function returns a single character from the keyboard.

Example:

```
char ch=getchar( )
```

The above statement stores a single character on character **ch**

putchar( ) function:The putchar( ) function displays or prints a single character.

Example:

```
char ch='P';
putchar(ch); //Displays P on the screen.
```

## Stream functions for I/O operations

C++ supports a number of input/output operations.The stream acts as an intermediate between I/O devices and the user.A stream is agroup of bytes.Streams are available in the form of functions stored in iostream header file.Keyboard and mouse are treated as objects in C++.

## Input functions

Input functions allows to input character and string data.The **get( )** and **getline( )** functions allows to accept a stream from input object into memory.These functions uses cin as object.

For Example:

```
cin.get( )
cin.getline( )
```

The following are the input functions in C++,

### a)get( )

The get( ) function is accept a single character or stream through the keyboard.

```
char ch;
ch=cin.get(ch); //accepts a character and stores in ch
cin.get(ch);   //accepts a character and stores in ch.
```

### b)getline( )

The getline( ) function is used to accept a string through the keyboard.It reads a string until a newline character is read.

The format of getline function is

**cin.getline(line,size);**

Where line is the name of the variable which stores line of text and size specifies the maximum number of characters which can be stored in line.

**Example:**

```
#include<iostream>

using namespace std;

int main( )

  {

        char name[40];

        cout<<"Enter the name";

        cin.getline(name,40);

        cout<<name;

        return 0;

}
```

## Output functions

The output functions allows flow of bytes(stream) from memory to output object.The object **cout** is used with these functions.The following are the output functions in C++.

### a)Put( ) function

The put( ) function is used to display a single character.

Example:

```
        char ch='c';

      cout.put(ch);
```

## b)write( ) function

The write( ) function is used to displays a string.

Example:

```
char  str[10]="hello";
cout.write(str,10);
```

## String functions

A string is a group of characters.String functions are used to manipulate a string.The header file cstring is used for these functions.

### i) strlen( )

The strlen( ) function is used to find the length of a string(number of characters in the string).

The syntax is:

**int strlen(string);**

Example:

```
n = strlen("Computer");
```

Stores 8 in n(ie number of charactes in the String Computer).

### ii) strcpy( )

The strcpy( ) function is used to copy one string into another.

The syntax is:

**strcpy(string1,string2);**

Copy string2 to string1

### iii) strcat( )

The strcat function is used to join (concatenate) one string to another string. The length of the

resultant string is the total length of the two strings.

The syntax is:

**strcat(string1,string2);**

```
char s1[20] = "Welcome", s2[10] = "  All";

strcat(s1,s2);

cout << s1;//Displays Welcome All.
```

### iv) strcmp( )

The strcmp( ) function is used to compare two strings.

The syntax is

**strcmp(strinh1,string2)**

Compares string1 with string2.

The funtion returns,

**0** if both strings are equal.

**-ve value** if string1 is alphabetically lower than string2.

**+ve value** if string1 is alphabetically higher than string2 .

### v) strcmpi( )

The strcmpi( ) function is used to compare two strings ignoring cases.The function will treat both the upper case and lower case letters as same for comparison.The syntax is same as strcmp( ) function.

The function returns,

**0** if the two strings are identical(same).

a **negative value** if string1<string2.

a **positive value** if string1>string2.

## Mathematical functions

The commonly used mathematical functions in C++ are,

**i)abs( ):-**It returns the absolute value of an integer.

Example: abs(-25),Returns 25

**ii) sqrt():-**It returns the square root of a number.

Example:sqrt(36),Returns 6.

**iii)pow( ):-**It is used to find power of a number.

Example:-pow(2,3),Returns 8(ie,2*2*2=8).

**Note:-**The header file **cmath**(GCC compiler) or **math.h** (Turbo C++ Compiler)is to be used for the above mathematical functions.

## Character functions

Character functions are used to perform various operations on characters.The header file cctype(GCC Compiler) supports these functions.The commonly used character functions in C++ are,

**i)isupper( ):-**This function checks if a character is in upper case or not. The function returns 1 if the character is in uppercase, and 0 otherwise.

**ii) islower( ):-** This function checks if a character is in lower case or not. The function returns 1 if the character is in lowercase, and 0 otherwise.

**iii) isalpha( ):-**This function is used to check whether the given character is an alphabet or not. The function returns 1 if the character is an alphabet, and 0 otherwise.

**iv) isdigit( ):-**This function is used to check whether the given character is a digit or not. The function returns 1 if the character is a digit, and 0 otherwise.

**v) isalnum( ):-**This function is used to check whether a character is alphanumeric or not. The function returns 1 if the character is alphanumeric, and 0 otherwise.

**vi) toupper( ):-**This function is used to convert the given character into its uppercase.

**vii) tolower( ):-**This function is used to convert the given character into its lower case.

## User-defined functions

A user defined function is a group of code to perform a specific task.Every function in C++ function consists of two parts,A function header and a function body. A function definition consists of function header and function body.The function header specifies the type of value it returns.

**The general format of a function is**

```
data_type  function_name(argument_list)
{
    statements ;
}
```

The result of a function is called **return value**.A function can only return a value.A function which is defined cannot be executed by itself.It is invoked by another function such as main( ).Depending on type of communication ,between calling and called function ,functions are of four types ,

- **Function with no argument and no return value.**

```
void sum1( )
{
int a, b, s;
cout<<"Enter 2 numbers: ";
cin>>a>>b;
s=a+b;
cout<<"Sum="<<s;
}
```

- **Function with argument and no return value.**

```
void sum3(int a, int b)
{
int s;
```

```
s=a+b;
cout<<"Sum="<<s;
}
```

- **Function with argument and return value.**

```
int sum4(int a, int b)
{
int s;
s=a+b;
return s;
}
```

- **Function with no argument but with return value.**

```
int sum2()
{
int a, b, s;
cout<<"Enter 2 numbers: ";
cin>>a>>b;
s=a+b;
return s;
}
```

The return statement returns a value to the calling function and transfers the program control back to the calling function.

## Function prototype

Function prototype models the actual function.It specifies the compiler the type and number of arguments a function use.The main use of function prototype is to prevent errors due to mismatch between argumnents.A function should be defined before it is called.

## Actual and Formal parameters

The values which are passed into a function is called **argumnets** or **parameters**.The arguments or parameters present in function definition is called formal parameter and those present in function call is called actual parameter.

For example

```
int main( )
{
  int x,y;
void output(int x,int y);       //Actual parameters are x and y
}
void output(int a,int b)        //Formal parameters are a and b
{
 int c;
 c=a+b;
cout<<"Sum ="<<c;
}
```

**Note:**The number and type of Actual and formal parameters must be the same.

## Passing arguments to a function

Based on the method of passing arguments, function calling methods can be classified into two

1. Call by Value method .
2. Call by Reference method.

## Call by value (Pass by value) method

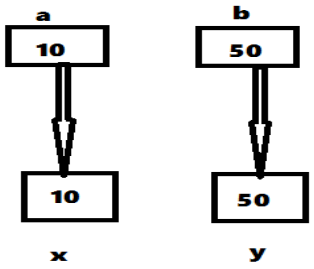
This is the default argument passing method in C++.In this method, the value contained in the actual argument is passed to the formal argument ie, a copy of the actual argument is passed to the function.Any changes made to the formal argument does not effect actual argument.

The limitation of this method is that only one value can be return from the function.

**Example:Swapping two values using call by value**

```
#include<iostream>
using namespace std;
void swap(int,int);              //function prototype declaration
int main( )
{
    int a=10,b=50;
    swap(a,b);//function call by value
   cout<<"The values of a and b are :";
   cout<<a<<" "<<b;
}
void swap(int x,int y)                  //function definition
{
    int t;
    t=x;
    x=y;
    y=t;
   cout<<"The values of x and y are :";
    cout<<x<<" "<<y;
}
```

## Working of the above program




t=10;

x=50;

y=10;

## Call by reference (Pass by reference) method

In this method a reference of the actual argument is passed to the function.The actual argument and formal argument shares the same memory location.Here any changes made to formal argument effects or is reflected back to the actual arguments.The formal argument is preceded by '**&**' Operator.

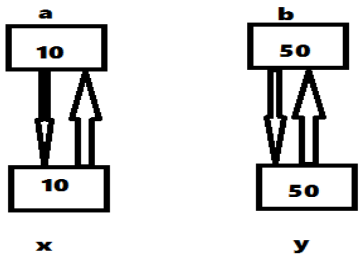
**Note:**A reference is an alternative name for an existing variable.A variable and its reference points to the same memory location.

**Example:Swapping two values using pass by reference**

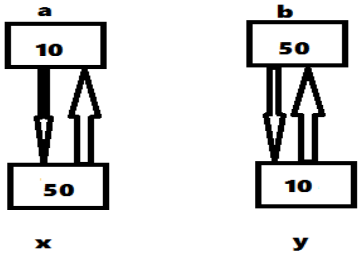
```
#include<iostream>

using namespace std;

int main( )

{

   void swap(int &x,int &y);          //function prototype declaration

    int a=10,b=50;

    cout<<"The values of a and b before swapping:";

    cout<<a<<" "<<b;
```

```
    swap(a,b); //function call by value
    cout<<"\n"<<"The values of a and b after swapping :";
    cout<<a<<" "<<b;
 }
void swap(int &x,int &y)                    //function definition
{
    int t;
    t=x;
    x=y;
    y=t;
}
```

**Before Swapping**




**After Swapping**




```
    t=10;
    x=50;
    y=10;
```

## Difference between Call by value and Call by reference methods.

| Call by value | Call by reference |
|---|---|
| Ordinary variables are used as formal parameters. | Reference variables are used as formal parameters. |
| Actual parameters can be a constant. | Actual parameters will be variables. |
| Changes made to formal arguments do not reflect actual arguments. | Changes made in the formal arguments do reflect in actual arguments. |
| Actual argument and formal arguments are stored in different memory locations. | Actual argument and formal arguments are stored in same memory locations. |

## Scope of variables and function

Scope of a variable is the portion of a program where the variable can be used(accessed).The variables declared within the body of a block(Function) are called local variables and can be used within the block.The life of local variables ends with execution of last instruction of the function.The variables declared outside any function and which are accessible to all functions are called global variables.Global variables have life time through out the program execution.

A function which is declared inside the body of another function is called a local function. A function declared outside the function body of any other function is called a global function. A global function can be used throughout the program.The scope of a global function is the entire program and that of a local function is only within the function where it is declared.