**Programming Fundamentals**

|  |  |
| --- | --- |
| Lab 03 | |
| **Topic** | Function Basic |
| **Objective** | * Functions   + Need and importance of functions     - Divide and conquer strategy     - Reusability   + Predefine functions, user define functions   + Value returning function, void function   + Function Prototypes, benefit of function prototype.   + Function Definition   + Function Calling   + Formal parameter, Actual parameter   + Value parameter, Reference parameter |

**Lab Description:**

This lab is basically designed for the basics of functions and the importance of functions.

**What is function?**

It is a block of code that is used for a specific task to reduce number of lines in a program.

**Importance of Function:**

* A program may need to repeat the same piece of code at various places.
* It may be required to perform certain task repeatedly.
* The program may become very large if functions are not used.
* The real reason for using function is to divide a problem into sub problems

**Advantages of Function:**

* Easier to Code
* Easier to Modify
* Easier to Maintain
* Reusability
* Less Programming Time
* Easier to Understand

**Types of Function:**

**Pre-define Function:**

In C++, predefined functions are organized into separate libraries. For example, the header file iostream contains I/O functions, and the header file cmath contains math functions.

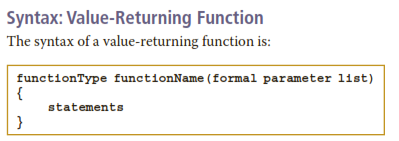
**User-define Function:**

C++ does not provide every function that you will ever need and designers cannot possibly know a user’s specific needs, you must learn to write your own functions. Those functions which are defined by user are called user define functions.

User-defined functions in C++ are classified into two categories:

**Value-returning functions**

Functions that have a return type. These functions return a value of a specific data type using the return statement, which we will explain shortly. Note that the function main has used a return statement to return the value 0 in every program we’ve seen so far.



**Void functions**

Functions that do not have a return type. These functions do not use a return statement to return a value.

We will first discuss value-returning functions. Many of the concepts discussed in regard to value-returning functions also apply to void functions.

**Function prototype:**

The function heading, terminated by a semicolon, without the body of the function. By reading the prototype you should understand the working of the function.

**Function call:**

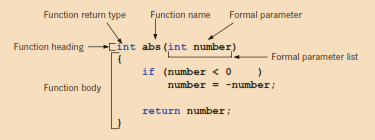
That activates a function is known as Function call. The following steps take place when a function is called:

* The control moves to the function that is called.
* All statements in function body are executed.
* Control returns back to calling statement.

In a function call, the number of actual parameters, together with their data types, must match with the formal parameters in the order given. That is, actual and formal parameters have a one-to-one correspondence.

**Function definition:**

A function is called (or invoked, or executed) by providing the function name, followed by the parameters being sent enclosed in parentheses. To invoke a function, see the syntax below



**Formal parameter and Actual parameter:**

When a function is called, the value of the actual parameter is copied into the corresponding formal parameter. If the formal parameter is a value parameter, then after copying the value of the actual parameter, there is no connection between the formal parameter and actual parameter; That is, the formal parameter is a separate variable with its own copy of the data. Therefore, during program execution, the formal parameter manipulates the data stored in its own memory space.

**Value Parameter:**

After copying data, a value parameter has no connection with the actual parameter, so a value parameter cannot pass any result back to the calling function. When the function executes, any changes made to the formal parameters do not in any way affect the actual parameters. The actual parameters have no knowledge of what is happening to the formal parameters. Thus, value parameters cannot pass information outside of the function. Value parameters provide only a one-way link from the actual parameters to the formal parameters.

**Reference Parameters:**

A reference parameter receives the address (memory location) of the actual parameter, reference parameters can pass one or more values from a function and can change the value of the actual parameter.

**Reference parameters are useful in three situations:**

When the value of the actual parameter needs to be changed, when you want to return more than one value from a function, when passing the address would save memory space and time relative to copying a large amount of data.

**Lab Tasks**

**Display Message**

Function that takes input in the form of message from user and display it on console.

**Calculate square**

Function to calculate square of a number and return the function value.

**Calculate Maximum value**

Function that takes two parameters and return the maximum between two.

**Swap numbers**

Implement a function-based C++ program to swap values of two integers. Do it twice, once by each call by value and call by reference and observe difference. Take initial values from user.

**Display array through function**

Program that takes 5 subject marks in an integers array from user. Your task is to display the marks of individual subjects by using functions.

**Calculate Sum Average and Percentage**

Program that takes 5 subject marks in an integers array and total marks of subject in integer variable from user. Your task is to display sum, average and percentage of these marks by using functions.

**Display Prime numbers**

Program that has a function that receives two numbers as an argument and display all prime numbers between these two numbers. Call this function from main( )

**Four Function Calculator Problem**

Program which works as a basic calculator. Make a function calculator() that accepts two integer values and a character value. Your function should perform the operation on the entered integer values depending on the character value entered by the user and return the output to main().

Make sure to create separate functions of sum(int,int), subtraction(int,int), multiplication(int, int) and division(int,int). The function calculator(int,int,char) should be able to call the necessary function as required.