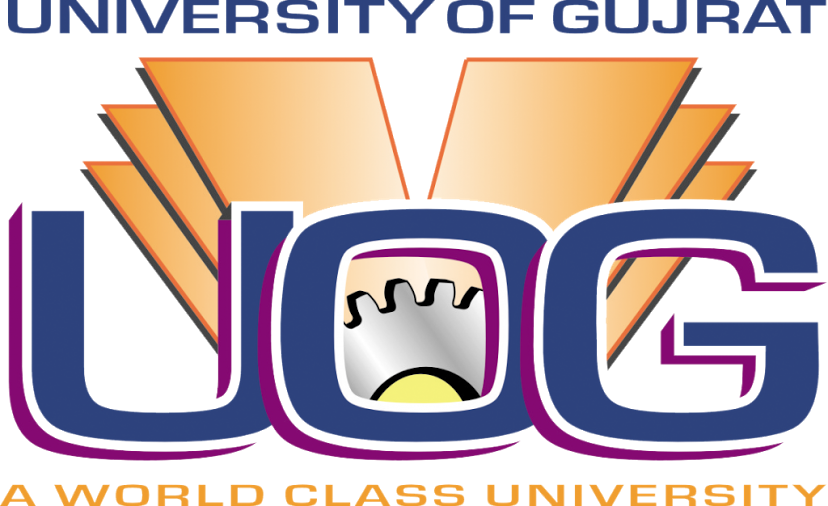
**Programming Fundamentals**

**Lab Manual**

**Week 6 – Lab 6**



**Repetition Control Structure**

**Session:**

**Faculty of Information Technology**

**UOG Gujrat Pakistan**

# Objectives

* To understand concept of functions
* Difference between void and value returning functions
* Calling functions form the main functions.

# Overview

Functions helps in making a programming code more organized and reduces the number of instructions. Use of functions is associated with reusability since the instructions enclosed by the function body can be reused just by calling the function wherever required instead of writing that set of instructions several times.

**FUNCTIONS**

The most commonly used function/method is void main(). This function is a necessary part of every program. There are several other function with which you are familiar like sqrt(), exp(), abs(), etc. These functions come from the header file math.h. The details of these functions have been implemented by the C++ developers. In this lab session, students will learn to implement their own functions and make function calls. Consider the main function:

void main() // function header

{

// function body

}

The function header is made up of the following parts:

**Return type:** In the above function, the return type is void. A function with void return type does not return any value to the place where the function is called. A function may have any other return type depending on the type of variable returned from the function.

**Name of the function:** Every function does have a name. The name of the function defined above is main. The names of functions should be descriptive about the purpose for which the function is implemented. For example, the function sqrt() indicates that it takes the square root of the variable placed between the round brackets.

**Parameter list:** Any variable written between the round brackets in the function header makes up the parameter list. The parameter list may have one or more variables. For example, the function sqrt() takes only one parameter. The function pow( , ) takes two parameters.

# Lab Task 1

Run the following code and observe the output

|  |
| --- |
| #include <iostream>  using namespace std;  void print()  {  cout<<”Hello world”;  }  int main()  {  print();  } |

# Lab Task 2

Run the following code and observe the output

|  |
| --- |
| #include <iostream>  using namespace std;  void print(char pr, int count){  for(int i=0; i<count; i++)  cout<<pr;  }  int main(){  print(‘\*’,5);  } |

# Lab Task 3

Change the above code to print a diamond.

# Lab Task 4

Now let’s move on to value returning functions. Run the following code and observe the output. Observe the use of **double** as a return type. This is to prevent overflow errors caused by the final answer not being able to fit inside a **float** type variable.

|  |
| --- |
| #include <iostream>  using namespace std;  double power(float base, int exponent)  {  double pow=1;  for(int i=0; i<exponent; i++)  pow\*=base;  return pow;  }  int main()  {  cout<<power(0.5,3);  } |

# Lab Task 5

Using similar logic as above write a function that computes the factorial of a number. Note: Ensure that the return type caters for overflow errors.

# Lab Task 6

Observe the following series for sin(x)

|  |
| --- |
| sin(x) = |

This series can be expanded and written as

|  |
| --- |
| sin(x) = |

Observe the powers of **-1**; they are incremented by **+1** for each term, also notice that the powers of **x** and the factorial get incremented by +2 in each iteration.

Now using the functions for power and factorial developed above write a function that computes the series sin(x). You can sum up to the first 10 terms.

#### Grading:

You will be graded on your originality and the precision of the algorithm.  The task should be clear to lab instructors with no ambiguity, and reproducible from the steps you list.

**NOTE: Any student not completing the task in class will be marked 0 for the lab.**

**Sample question**

Q1. Write a function that takes two Distance values as arguments and returns the larger one. Include a main() program that accepts two Distance values from the user, compares them, and displays the larger. (See the RETSTRC program for hints.)

Q2. Write a function called hms\_to\_secs() that takes three int values—for hours, minutes, and seconds—as arguments, and returns the equivalent time in seconds (type long).

Create a program that exercises this function by repeatedly obtaining a time value in hours, minutes, and seconds from the user (format 12:59:59), calling the function, and displaying the value of seconds it returns.