CL-1002 – Programming Fundamentals Lab

LAB MANUAL

DEPARTMENT OF ELECTRICAL ENGINEERING,

FAST-NU, LAHORE

Created by: Dr. Usman Shahid , Ms. Hina Tariq and Ms.Tooba Javed

Date: August 30, 2014

Last Updated by: Ms. Shazia Haque & Mr. Hamza Yousuf

Date: July 30, 2023

Approved by: Head of Electrical Engineering Department

Date: July, 2023

Table of Contents

List of Equipment

Sr. No.	Description
1	Workstations (PCs)
2	Visual Studio 2012 C++ (software)

Experiment No. 01

Introduction to Visual Studio 2012 and C++ Basics

OBJECTIVE:

- To be able to use Visual Studio for compiling simple C++Programs.
- To be able to understand I/O environment in C++
- To get familiar with datatypes
- To be able to understand and solve a programming problem using C++ arithmetic expressions.

Section – 1: Microsoft Visual Studio tutorial

1) Start menu->Microsoft Visual Studio 2012->Visual C++ Development Settings->Start Visual Studio as depicted in Figure 3-1

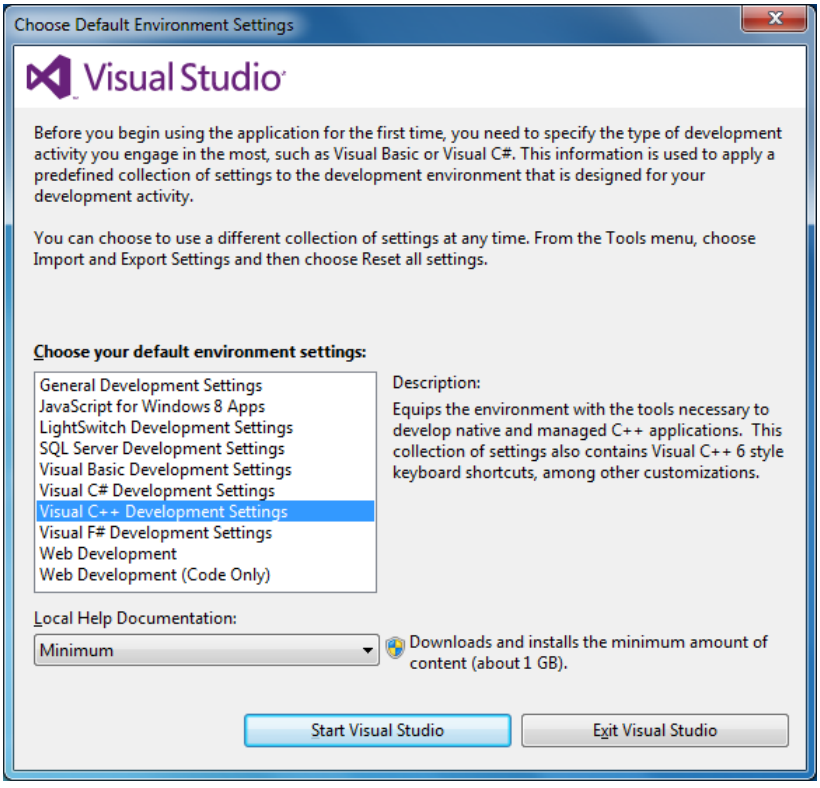

Figure 3-1

2) Microsoft Visual Studio->File->New->Project (shown in Figures3-2, 3-3 & 3-4)

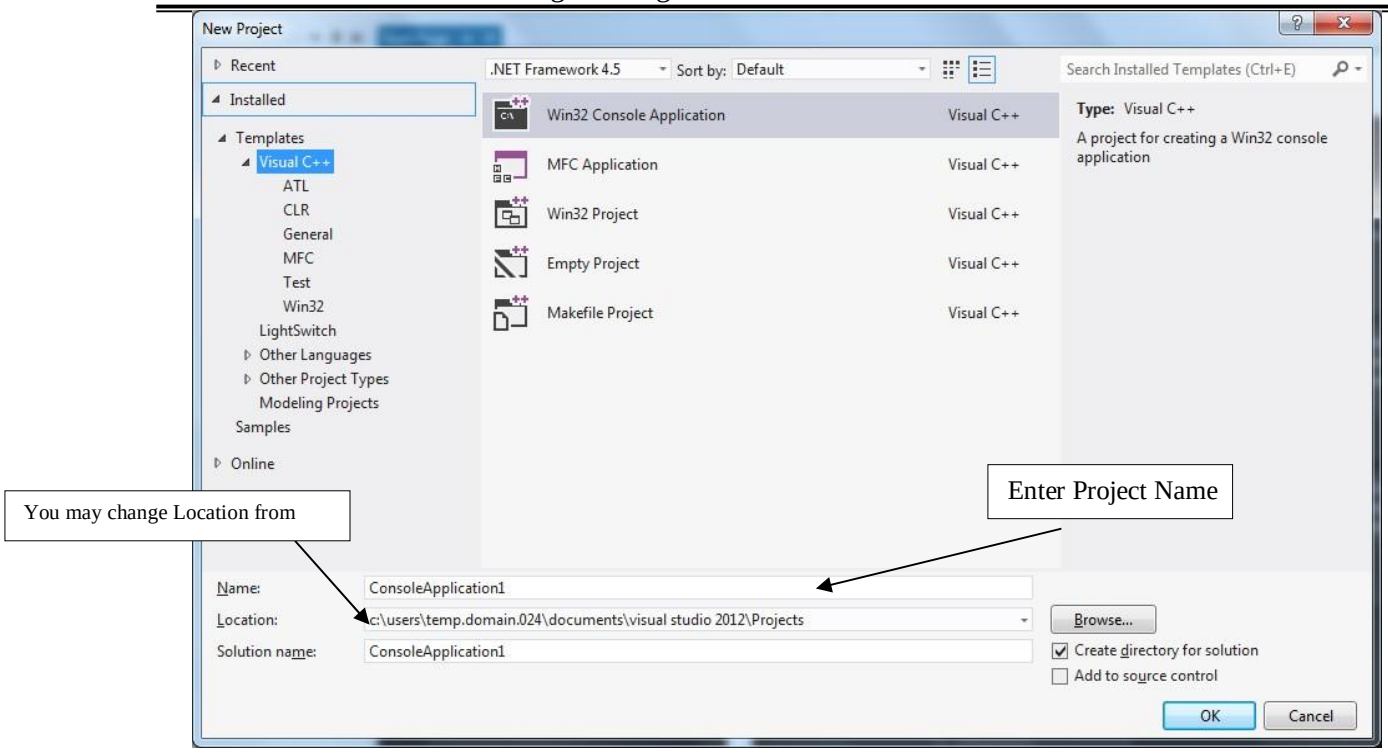

Figure 3-2

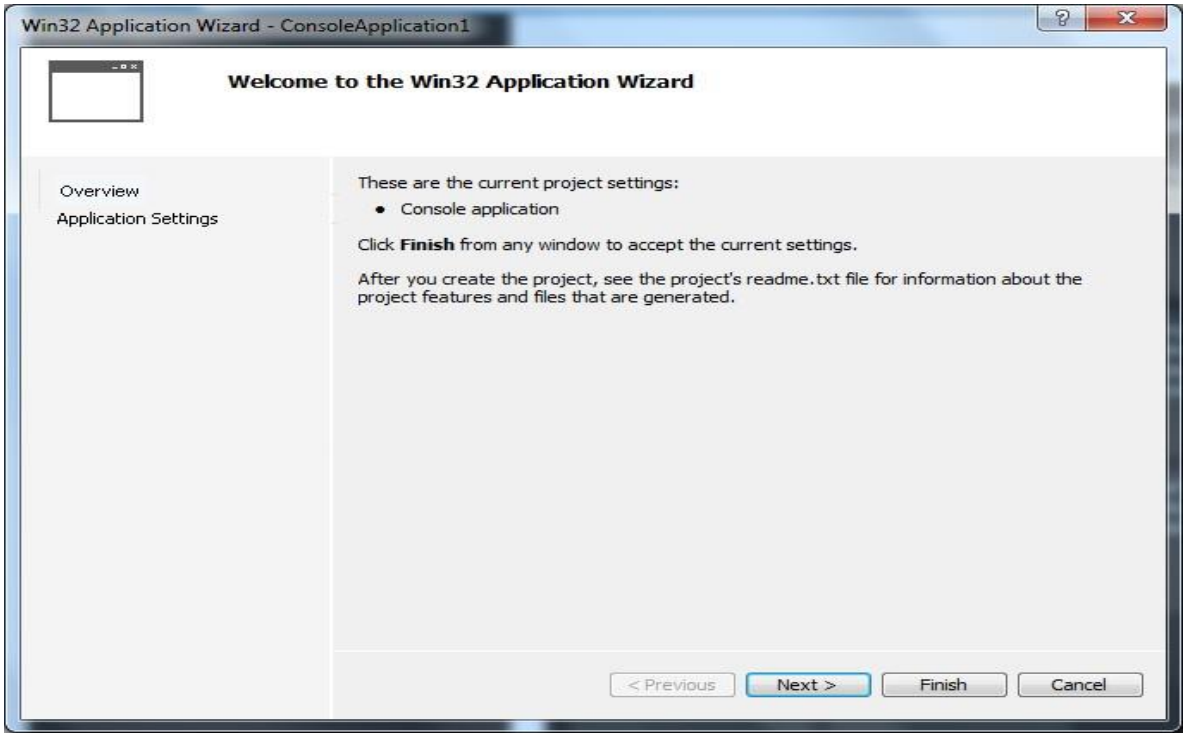

Figure 3-3

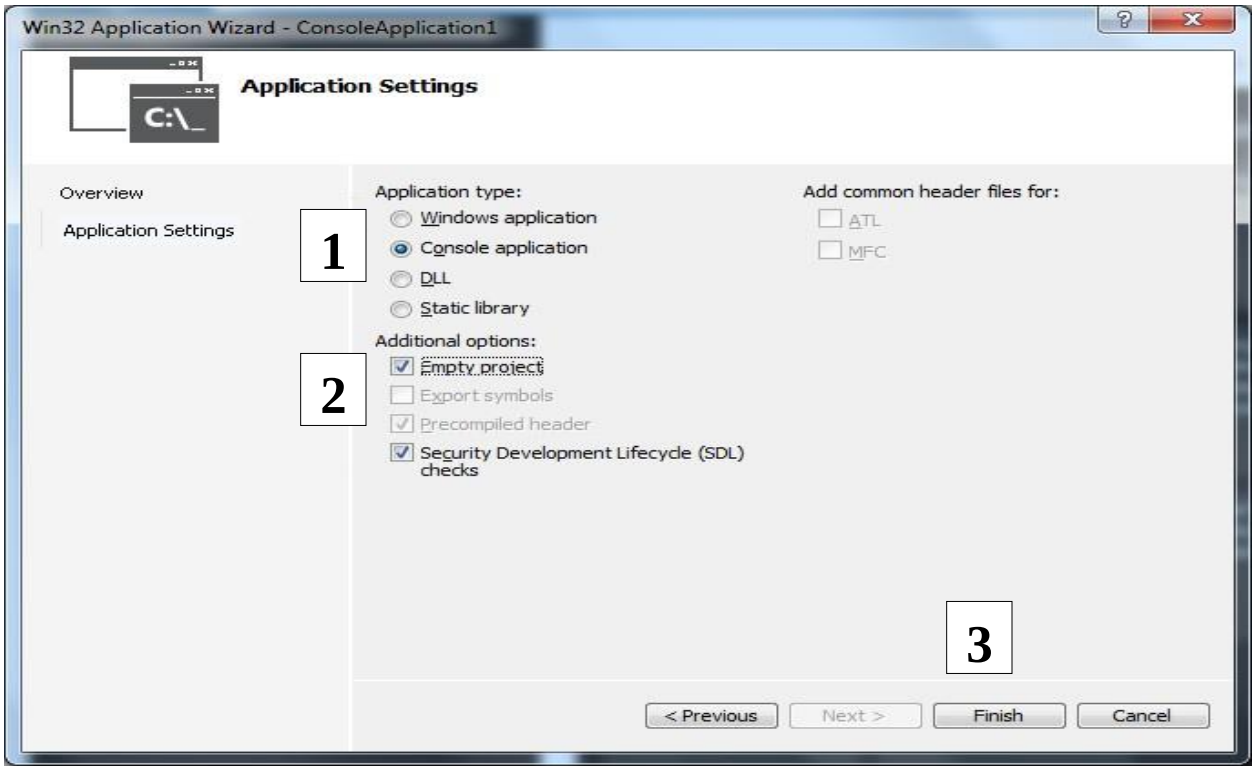

Figure 3-4

3) Creating a CPP file:

Right click on Source Folder-> Add->New Item then following window will be opened as shown in Figure 3-5

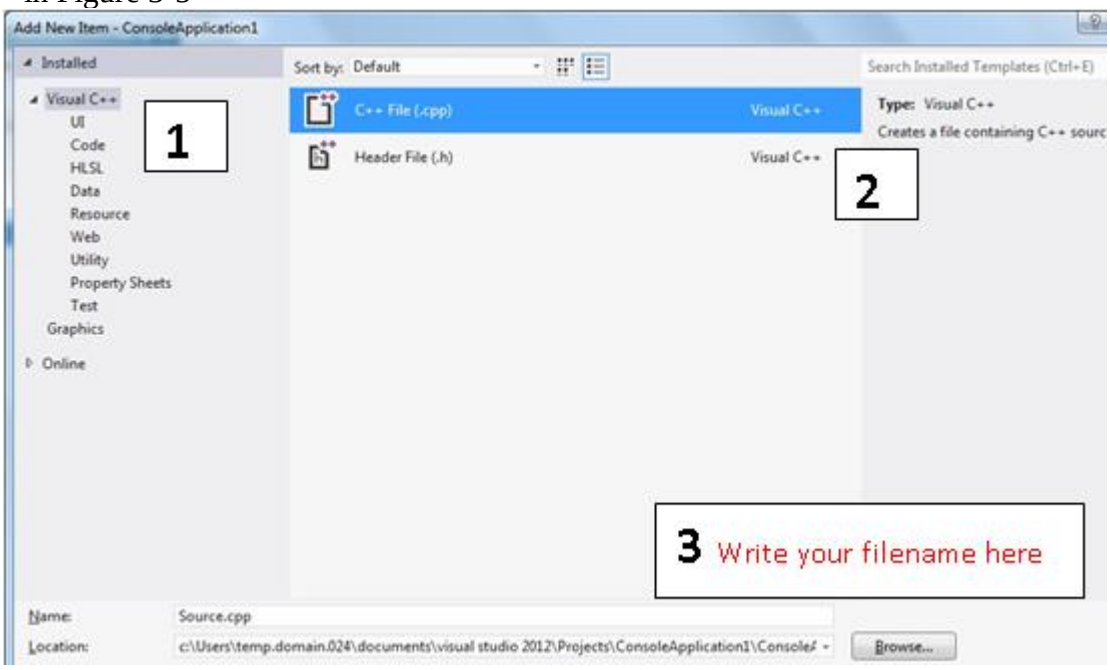

Figure 3-5

4) Type the following in code window**.** It will look like in the figure 3-6.

```
#include <iostream>
using namespace std;
 void main()
{
                cout<<"hello world";
}
```

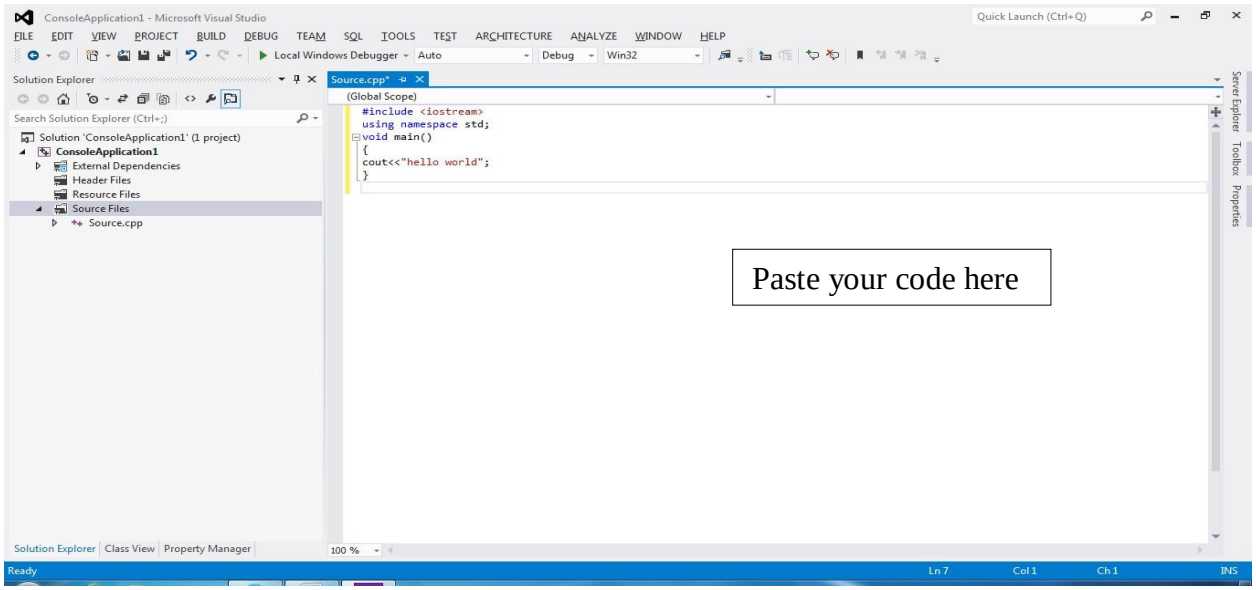

Figure 3-6

5) Compiling code:

Build Menu->Build Solution

6) Executing Code:

To see output of your code press Ctrl+F5, A black window will be opened. It is called console as shown below in Figure 3-7.

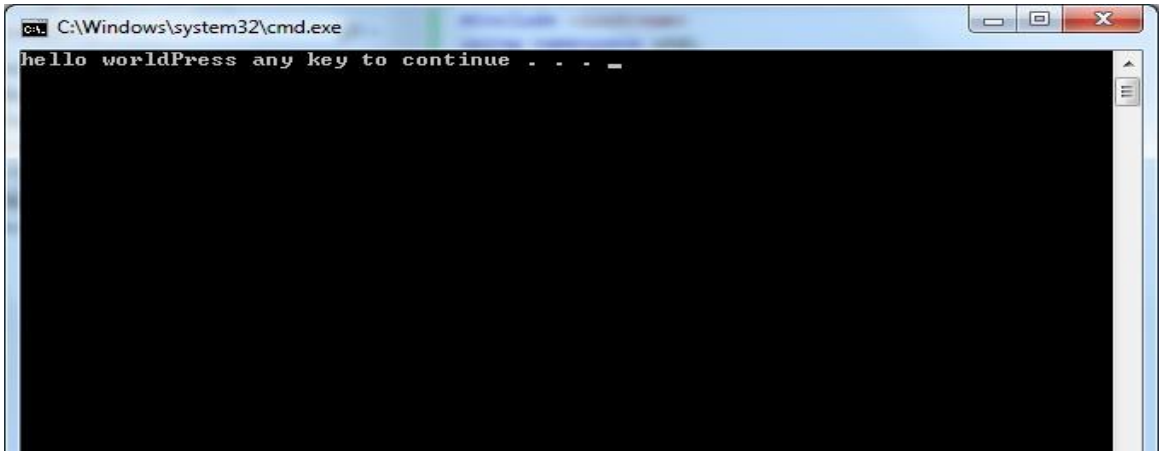

Figure 3-7

Basic Data Types:

The '*type'* of a variable determines what kind of values it may take on. There are only a few basic data types in C++. The first ones we'll be encountering and using are:

- 'char' a character
- 'int' an integer, in the range -32,767 to32,767
- 'long int' a larger integer (up to+-2,147,483,647)
- 'float' a floating-point number
- 'double' a floating-point number, with more precision and perhaps greater range than float

Declaration:

A *declaration* tells the compiler the name and type of a variable you'll be using in your program. In its simplest form, a declaration consists of the type, the name of the variable, and a terminating semicolon:

```
char c;
int i;
float f;
```

Exercise - 1 (5 points)

Write a code in C++ to display your name, roll number, degree and semester on console in following way. Use "\t" to give tab space between each field

```
*******************************

Name  Roll Number Degree Semester

*******************************
```

Please give your C++ code for solving the above question in the space below:

Exercise - 2 (10 points)

Declare three variables (A,B,C) of type **int** in C++. Take input from console in each variable. Now write code such that following is displayed on screen:

A	B	C
[A]*2	[B]*2	[C]*2
[A]+3	[B]+2	[C]+2

Please give your C++ code for solving the above question in the space below:

Exercise - 3 (5 points)

Write a code in C++ to display following on console:

```
   *
  **
 ***
****
```

Please give your C++ code for solving the above question in the space below:

Experiment No. 02

Basic Data Types and Variable Declaration

OBJECTIVE:

- To get familiar with data types
- To understand how to use basic arithmetic operators in C++

The '*type'* of a variable determines what kinds of values it may take on. An '*operator'* computes new values out of old ones. An '*expression'* consists of variables, constants, and operators combined to perform some useful computation. In this chapter, we'll learn about C++ basic types, how to declare variables of different types, and what the basic operators are.

Basic Data Types:

There are only a few basic data types in C++. The first ones we'll be encountering and using are:

- 'char' a character
- 'int' an integer, in the range -32,767 to 32,767
- 'long int' a larger integer (up to +-2,147,483,647)
- 'float' a floating-point number
- 'double' a floating-point number, with more precision and perhaps greater range than `float`

If you can look at this list of basic types and say to yourself, "Oh, how simple, there are only a few types, I won't have to worry much about choosing among them," you'll have an easy time with declarations.

Declaration:

A *declaration* tells the compiler the name and type of a variable you'll be using in your program. In its simplest form, a declaration consists of the type, the name of the variable, and a terminating semicolon:

```
char c;
int i;
float f;
```

Basic Operators:

The basic operators for performing arithmetic are the same in many computer languages:

```
+      addition
-      subtraction
*      multiplication
/      division
```

Exercise 1:
Write a C++ program that displays a table of 06 entries that contains square and cube of numbers. An excerpt from table is shown.

```
a       a^2     a^3
1       1       1
2       4       8
3       9       27
4       16      64
```

Paste screenshot of Code and Output here…

Exercise 2:
Write a C++ program that displays following pattern

```
  CCCC    +       +
 C        +       +
C      +++++++ +++++++
 C       +       +
  CCCC   +       +
```

Paste screenshot of Code and Output here…

Exercise –3

Convert following algebraic equation into C++ arithmetic expression:

$$V = IR$$

- Where "R", "I" and "V" is resistance, current and voltage respectively.
- **Ask user to enter values** for 'I' and 'R' and display the result of this expression. All variables can be integers only. *For example,* if x=2, and a=1, the y should be 15.

Paste screenshot of Code and Output here…

Exercise –4

Single room takes 2 liters of paint mixture whereas 2 liters' paint mixture cost is 1000 rupees.

Write a C++ program that **takes as input the number of rooms** to be painted and outputs the total cost to paint these rooms. (Assume that all rooms are of same standard size.)

Number of Rooms	Total cost
4	
3	
7	

Paste screenshot of Code and Output here…

Experiment No. 03

Data Types, Variable Declaration and Arithmetic Operators

OBJECTIVE:

- To get familiar with arithmetic operators
- To understand how to use basic arithmetic operators in C++

The '*type'* of a variable determines what kinds of values it may take on. An '*operator'* computes new values out of old ones. An '*expression'* consists of variables, constants, and operators combined to perform some useful computation. In this chapter, we'll learn about C++ basic types, how to declare variables of different types, and what the basic operators are.

Basic Data Types:

There are only a few basic data types in C++. The first ones we'll be encountering and using are:

- 'char' a character
- 'int' an integer, in the range -32,767 to 32,767
- 'long int' a larger integer (up to +-2,147,483,647)
- 'float' a floating-point number
- 'double' a floating-point number, with more precision and perhaps greater range than `float`

If you can look at this list of basic types and say to yourself, "Oh, how simple, there are only a few types, I won't have to worry much about choosing among them," you'll have an easy time with declarations.

Declaration:

A *declaration* tells the compiler the name and type of a variable you'll be using in your program. In its simplest form, a declaration consists of the type, the name of the variable, and a terminating semicolon:

```
char c;
int i;
float f;
```

Basic Operators:
The basic operators for performing arithmetic are the same in many computer languages:

```
+      addition
-      subtraction
*      multiplication
/      division
%      modulus (remainder)
```

Multiplication, division, and modulus all have higher *precedence* than addition and subtraction. The term ``precedence" refers to how ``tightly" operators bind to their operands (that is, to the things they operate on). In mathematics, multiplication has higher precedence than addition, so `1 + 2 * 3` is 7, not 9. In other words, `1 + 2 * 3` is equivalent to `1 + (2 * 3)`. C is the same way.

Exercise-1

Suppose 'x', 'y', and 'z' are **int** variables and 'w' and't' are **double** variables. What value is assigned to each of these variables after the last statement executes?

```
x = 17;
y = 15;
x = x + y / 4;
z = x % 3 + 4;
w = 17 / 3 + 6.5;
t = x / 4.0 + 15 % 4 - 3.5;
```

In the space below please solve on paper. Then write a C++ code to verify your answer (place a screen shot of your output as well)

Paste screenshot of Output here…

Exercise-2

Write a program, which takes an integer as a parameter and display the number in a reverse order. For example, if the user enters the value 12345, your program must display 54321 on the console window.

Paste screenshot of Code and Output here…

Exercise-3

Write a C++ code to make the following pattern (as given below). Input (starting character) will be given by user through console which is to be stored in a char variable. You are required to print next three characters according to the given pattern.

Starting Char	**Output**		
A	**C**	**F**	**J**
B	**D**	**G**	**K**
C	**E**	**H**	**L**

Paste screenshot of Code and Output here…

Exercise-4

You can convert temperature from degrees Celsius to degrees Fahrenheit by multiplying by 9/5 and adding 32. Write a C++ program that allows the user to enter a floating-point number representing degrees Celsius, and then displays the corresponding degrees Fahrenheit.

Paste screenshot of Output here…

Experiment No. 04

I/O and File Streams

OBJECTIVE:

Things that will be covered in today's lab:

- To be able to understand the working of files and functions.

Reading data from file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
Ifstream in_stream;
in_stream.open("infile.txt");

char i;

in_stream>>i;
cout<<i<<" ";

in_stream.close();
return 0;
 }
```

Step#1
fstream library is included in order to read/write from/to file.

Step#2
In order to read from file, a variable is declared of type **ifstream** (Input stream).

Step#3
Using the variable declared in step 2, open the file from which you want to read data.

Step#4
Just like you use cin to read data from console, the ifstream variable is used exactly the same way. This statement will read first character from file. We can read integers or any other data type too.

Step#5
After reading data from file, this statement is used for closing the file.

To read file till end:

```
while(!in_stream.eof())
{
in_stream>>i;
//do any required processing on data
}
```

Using built in function of fstream class
eof() is a function already defined in fstream class. It returns 0 if file handle has not reached to end of file

Writing data to file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
Ofstream o_stream;
o_stream.open("ofile.txt");
int a = 10;

      o_stream<<"Hello World";
      o_stream<<a;


o_stream.close();
return 0;
 }
```

Step#1
fstream library is included in order to read/write from/to file

Step#2
In order to write data to file, a variable is declared of type **ofstream** (output stream)

Step#3
Using the variable declared in step 2, open the file to which you want to write data. If file does not exist then a new file with name "ofile" is created. If a file exists already then previous data is removed.

Step#4
Just like you use cout to write data on console, the ofstream variable is used exactly the same way. This statement will write value of ain the beginning of file. We can read integers or any other data type too

Step#5
After writing data to file, this statement is used for closing the file

Example Codes

Please run the example codes and paste the screen shot of the output

Following code reads all integers from a file and displays on screen. Create a file with name "myfile.txt" Enter some integers in it separated by space and save it in the same folder in which you have saved your cpp file.

```
#include <iostream>
#include <fstream>
using namespace std;
void main(){
ifstream a;
a.open("myfile.txt");
int x;
while(!a.eof()){
a>>x;
cout<<x;}
a.close();}
```

Following code reads all integers from a file and writes in another file. Use myfile.txt of PART – A.

```
#include <iostream>
#include <fstream>
using namespace std;
void main()
{
ifstream in;
in.open("myfile.txt");
ofstream out;
out.open("output_file.txt");
int x;
while(!in.eof())
{
in>>x;
out<<x;
}
in.close();
out.close();
}
```

Exercise 1: (10 points)

Modify the code B in Exercise - 1 such that it reads all integers from an input file and writes **the number**, **square of that number** and **cube of that number** separated by tab to an output file. Sample input and output files are given below. Use **pow** function of **cmath** library.

Input file

```
1
2
3
4
5
6
7
```

Output file

```
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
```

Paste screenshot of Code here…

Exercise 2: (10 points)

File "quiz_result.txt" stores the result of three quizzes for 20 students. The format of the file is shown below. The first column in the file lists the marks of first quiz, second columns lists the marks of second quiz and third column is for third quiz. Write a program, which reads the marks of all the quizzes for each student, and calculate and display the total marks obtained by each student.

quiz_result.txt

```
2   3   2
10  9   8
4   5   7
2   1   0
3   3   5
.................
```

Paste screenshot of Code here…

Exercise 3: (10 points)

The file named **fact_data.txt** contains integers in each line. Write a program that reads each integer and then finds and displays the factorial of that integer as well as write it in a file named **fact_out.txt** in a new line.

Input file

```
1
2
3
4
5
```

fact_data.txt

Output file

```
1
2
6
24
120
```

fact_out.txt

Paste screenshot of Code here…

Exercise 4: (10 points)

The file named **employee_data.txt** contains 2 types of information in each line: employee id and the salary/month corresponding to that id. Using this information, create another file, **employee_annum_data.txt** that contains the employee id, the total salary earned and the total tax paid by that employee per year in each line. Assume that each employee is paying 5% of their salary as tax.

Input file

1	125.966
2	558.124
3	981.000
4	888.888
5	15.213

employee_data.txt

Paste screenshot of Code here…

Experiment No. 05

Control Structures: If-else statement and operators

OBJECTIVE:

- To implement nested "if else" statements.

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general from of a typical decision making structure found in most of the programming languages:

C++ programming language provides following types of decision making statements as shown in table 5-1

Table 5-1

Statement	Description
if statement	An if statement consists of a boolean expression followed by one or more statements.
if...else statement	An if statement can be followed by an optional else statement, which executes when the boolean expression is false.
nested if statements	You can use one if or else if statement inside another if or else if statement(s).

If Else Statement:

An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is false.

Syntax: The syntax of an if...else statement in C++ is:

```
if(boolean_expression)
{
// statement(s) will execute if the boolean expression is true
}
else
{
// statement(s) will execute if the boolean expression is false
}
```

If the boolean expression evaluates to **true**, then the **if block** of code will be executed, otherwise **else block** of code will be executed.

If Elseif Else Statements:

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.
When using if , else if , else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax: The syntax of an if...else if...else statement in C++ is:

```
if(boolean_expression 1)
{
// Executes when the boolean expression 1 is true
}
elseif( boolean_expression 2)
{
// Executes when the boolean expression 2 is true
}
elseif( boolean_expression 3)
{
// Executes when the boolean expression 3 is true
}
else
{
// executes when the none of the above condition is true.
}
```

Exercise - 1(5 points)

In a right triangle, the square of the length of one side is equal to the sum of the squares of the lengths of the other two sides.

Write a program that prompts the user to enter the lengths of three sides of a triangle and then outputs a message indicating whether the triangle is a right triangle.

Paste screenshot of Output here…

Exercise - 2(10 points)

Any year is input by the user. Write a program to determine whether the year is a leap year or not.
How to determine whether a year is a leap year
To determine whether a year is a leap year, follow these steps:

1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days).
5. The year is not a leap year (it has 365 days).

Paste screenshot of Output here…

Exercise –3(10 points)

Write a program in C++ using if else operator to find the grade of a student. The program should prompt the user to enter marks.
The details are as follows:

marks >= 90 Grade A
marks >= 75 Grade B
marks >= 60 Grade C
marks >= 45 Grade D
else Grade F

Paste screenshot of Output here…

Exercise –4(10 points)

Write a C++ program to design a calculator. The program should take2 integers ‘a’ and ‘b’ and an operator ‘ch’ of your choice. For example,

if a=2 and b=3 and ch=+, then output would be a+b=5

if a=2 and b=3 and ch=*, then output would be a*b=6

Your calculator should be able to perform all the operations below:

1. If ch = ‘+’ then calculate add = a + b and display the additionresult.
2. If ch= ‘- ‘then calculate sub = a – b and display the subtractionresult.
3. If ch= ‘*’ then calculate mul = a * b and display the multiplicationresult.
4. If ch= ‘/’ then calculate div = a / b and display the divisionresult.
5. If ch=’%’ then calculate mod = a % b and display the modulusresult.
6. Otherwise display invalidoperator

Paste screenshot of Output here…

Exercise 5:

The roots of the quadratic equation $ax^2 + bx + c = 0$, are given by the following formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Write a program that prompts the user to input the value of ‘a’ (the coefficient of x^2), ‘b’ (the coefficient of x), and ‘c’ (the constant term) and outputs the type of roots of the equation.

Case 1: $b^2 - 4ac = 0$	Single repeated roots
Case 2:$b^2 - 4ac > 0$	Two real roots
Case 3:$b^2 - 4ac < 0$	Two Complex roots

Paste screenshot of Output here…

Exercise 6:

Write a program to calculate the monthly telephone bills as per the following rule:
Minimum Rs. 200 for upto 100 calls.

- Plus Rs. 0.60 per call for next 50 calls.
- Plus Rs. 0.50 per call for next 50 calls.

Plus Rs. 0.40 per call for any call beyond 200 calls.

Paste screenshot of Output here…

Experiment No. 06

Nested – if and switch structure

OBJECTIVE:

- To be able to understand the working of switches.

If Else if Else Statements:

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.
When using if , else if , else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of he remaining else if's or else's will be tested.

Syntax: The syntax of an if...else if...else statement in C++ is:

```
if(boolean_expression 1)
{
// Executes when the boolean expression 1 is true
}
elseif( boolean_expression 2)
{
// Executes when the boolean expression 2 is true
}
elseif( boolean_expression 3)
{
// Executes when the boolean expression 3 is true
}
else
{
// executes when the none of the above condition is true.
}
```

Switch:
Switch case statements are a substitute for long if statements that compare a variable to several values. The basic format for using switch case is outlined below. The value of the variable given into switch is compared to the value following each of the cases, and when one value matches the value of the variable, the computer continues executing the program from that point.

Expression:

```
switch(variable){
case value1  :
       statement(s);
```

```
break;//optional
case value2  :
       statement(s);
break;//optional

// you can have any number of case statements.
default://Optional
       statement(s);
}
```

The condition of a switch statement is a value. The case says that if it has the value of whatever is after that case then do whatever follows the colon. The break is used to break out of the case statements. Break is a keyword that breaks out of the code block, usually surrounded by braces, which it is in. In this case, break prevents the program from falling through and executing the code in all the other case statements. An important thing to note about the switch statement is that the case values may only be constant integral expressions. Sadly, it isn't legal to use case like this:

The default case is optional, but it is wise to include it as it handles any unexpected cases. Switch statements serves as a simple way to write long if statements when the requirements are met. Often it can be used to process input from a user.

Exercise - 1

Any year is input by the user. Write a program to determine whether the year is a leap year or not.
How to determine whether a year is a leap year
To determine whether a year is a leap year, follow these steps:

1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days).
5. The year is not a leap year (it has 365 days).

Paste screenshot of Output here…

Exercise - 2

Write a C++ to perform arithmetic Operations using switch case.

1. Read two numbers ‘a’ and ‘b’ **and** 'Read your choice of operator ch.

2. If ch = ‘+’ then calculate add = a + b and display the addition result.
3. If ch= ‘- ‘then calculate sub = a – b and display the subtraction result.
4. If ch= ‘*’ then calculate mul = a * b and display the multiplication result.
5. If ch= ‘/’ then calculate div = a / b and display the division result.
6. If ch=’%’ then calculate mod = a % b and display the modulus result.
7. Otherwise display invalid operator.

Paste screenshot of Output here…

Exercise - 3

Write a program in C++ using if else operator to find the grade of a student.
The details are as follow

marks >= 90 Grade A
marks >= 75 Grade B
marks >= 60 Grade C
marks >= 45 Grade D
else Grade F

Paste screenshot of Output here…

Experiment No.07

Control Structures: Loops (while & for)

OBJECTIVE:

- To be able to understand the working of loops.

Repetition Statements

Repetition statements are called *loops*, and are used to repeat the same code multiple times in succession. The number of repetitions is based on criteria defined in the loop structure, usually a true/false expression.
The three loop structures in C++ are:

- **while** loops
- **for** loops

Three types of loops are not actually needed, but having the different forms is convenient

while and do-while loops: Format of while loop:

```
// while loop format
while (expression)
 {
    statement1;
    statement2;
    // ...
    statementN;
 }
```

How they work:
The expression is a test condition that is evaluated to decide whether the loop should repeat or not.

- true means run the loop body again.
- false means quit.

The while and do/while loops both follow the same basic flowchart -- the only exception is that:

- In a while loop, the expression is tested first
- In a do/while loop, the loop "body" is executed first.

For Loop:
The **for** loop is most convenient with *counting loops* -- i.e. loops that are based on a counting variable, usually a known number of iterations
Remember that the statement can be a single statement or a compound statement (block), so an alternate way to write the format might be:

```
for (initialCondition; testExpression; iterativeStatement)
 {
    statement1;
    statement2;
    // ...
    statementN;
 }
```

How it works

- The *initialCondition* runs once, at the start of the loop
- The *testExpression* is checked. (This is just like the expression in a while loop). If it's false, quit. If it's true, then:
- Run the loop body
- Run the *iterativeStatement*
- Go back to the *testExpression* step and repeat

Special statements: break and continue

- These statements can be used to alter the flow of control in loops, although they are not specifically *needed*. (Any loop can be made to exit by writing an appropriate *test expression*).
- **break**: This causes immediate exit from any loop (as well as from switch blocks)
- **continue**: When used in a loop, this statement causes the current loop iteration to end, but the loop then moves on to the next step.
    - In a while or do-while loop, the rest of the loop body is skipped, and execution moves on to the *test condition*
    - In a for loop, the rest of the loop body is skipped, and execution moves on to the *iterative statement*

Exercise –1

For loops can always be re-written as **while** loops, and vice-versa. Are the following two programs equivalent, and what is their output? Explain your answer in the space provided, and run the programs to check.

Program (a):

```
#include <iostream>
using namespace std;
int main()
{
        for (int count=1; count <= 5 ; count++)
        {
                int count = 1;
                cout << count << "\n";
        }
        return 0;
}
```

Program (b):

```
# include<iostream>
using namespace std;
        int main()
        {
                int count = 1;
                while(count<=5)
                {
                        int count = 1;
                        cout << count << "\n";
                        count++;
                }
                return 0;
        }
```

Exercise –2

Write a C++ program to compute the sum of first n even numbers. “n” should be taken as an input from the user. Display the result of the program on console.

Paste screenshot of Output here…

Exercise –3

Write a C ++ program that calculates the sum and factorial of the integers for each integer from 1 to n. Where n is a positive integer that you enter.

- You are required to only use **while loop** structure in this question.
- Your program should produce the output below

```
integer     sum       factorial
================================
   1          1               1
   2          3               2
   3          6               6
   4         10              24
   5         15             120
   6         21             720
   7         28            5040
```

Paste screenshot of Code here…

Exercise –4

Determine the pattern followed by the series below:
1, 2, 3, 5, 8, 13, 21, 34, …
Write a program that takes a number ‘n’ and prints first ‘n’ numbers of this series. For example, if user enters 6, your program should display 1, 2, 3, 5, 8, 13

Paste screenshot of Code here…

Experiment No.08

Nested Control Structures

OBJECTIVE:

- To be able to understand the working of nested control structures.

Exercise –1 (5 points)

Write a program that uses nested for – loops to print first 10 entries of tables of 1 to 10. For example the first 10 entries of tables of 1 to 5 are given below in figures 7-1 & 7-2.

```
----- Table of 1 -----
1 x 1   : 1
1 x 2   : 2
1 x 3   : 3
1 x 4   : 4
1 x 5   : 5
1 x 6   : 6
1 x 7   : 7
1 x 8   : 8
1 x 9   : 9
1 x 10  : 10
-----------------------
----- Table of 2 -----
2 x 1   : 2
2 x 2   : 4
2 x 3   : 6
2 x 4   : 8
2 x 5   : 10
2 x 6   : 12
2 x 7   : 14
2 x 8   : 16
2 x 9   : 18
2 x 10  : 20
-----------------------
----- Table of 3 -----
3 x 1   : 3
3 x 2   : 6
3 x 3   : 9
3 x 4   : 12
3 x 5   : 15
3 x 6   : 18
3 x 7   : 21
3 x 8   : 24
3 x 9   : 27
3 x 10  : 30
```

```
-----------------------
----- Table of 4 -----
4 x 1   : 4
4 x 2   : 8
4 x 3   : 12
4 x 4   : 16
4 x 5   : 20
4 x 6   : 24
4 x 7   : 28
4 x 8   : 32
4 x 9   : 36
4 x 10  : 40
-----------------------
----- Table of 5 -----
5 x 1   : 5
5 x 2   : 10
5 x 3   : 15
5 x 4   : 20
5 x 5   : 25
5 x 6   : 30
5 x 7   : 35
5 x 8   : 40
5 x 9   : 45
5 x 10  : 50
-----------------------
Press any key to continue . . .
```

Paste screenshot of Code here…

Exercise –2 (10 points)

Write a C++ program that takes 2 integers (i) a starting number and (ii) the number of elements in last row. Then print the data in form of a lower triangular matrix. For example (shown in table 7-2), if:

Table 7-2

Starting number=10 Elements in last row=3 10 11 12 13 14 15	Starting number=1 Elements in last row=4 1 2 3 4 5 6 7 8 9 10	Starting number=20 Elements in last row=2 20 21 22

Paste screenshot of Code here…

Exercise –3 (10 points)

Design and write a C++ program that takes as input an integer larger than 1 and calculates the sum of the squares from 1 to that integer. The output should be the value of the squares and the sum, properly labelled on the screen.

For example, if the integer equals **"4"**, your program would display:

1
4
9
16
Sum=30

Paste screenshot of Code here…

Exercise –4 (10 points)

An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3^3 + 7^3 + 1^3 = 371.Write a program to find all Armstrong number in the range of 0 and 999.

Paste screenshot of Code and Output here…

Exercise –5 (10 points)

Write a program to calculate the place value of digit in an integer.
For example, if the user inputs an integer **"6918"** and you want to determine the place value of 6, the output would be "Thousands". You can include a check for whether that specific digit is present or not. Your program should work for a maximum place value of "thousands".

*Units; tens; hundreds; thousands.

Paste screenshot of Code and Output here…

Experiment No.09

C++ Functions

OBJECTIVE:

- To understand how to use pre-defined functions
- To understand how to define your own functions
- To get familiar with value returning functions

Functions allow to structure programs in segments of code to perform individual tasks.

In C++, a function is a group of statements that is given a name, and which can be called from some point of the program. The most common syntax to define a function is:

```
type name ( parameter1, parameter2, ...) { statements }
```

Where:
- 'type' is the type of the value returned by the function.
- 'name' is the identifier by which the function can be called.
- 'parameters' (as many as needed): Each parameter consists of a type followed by an identifier, with each parameter being separated from the next by a comma. Each parameter looks very much like a regular variable declaration (for example: int x), and in fact acts within the function as a regular variable which is local to the function. The purpose of parameters is to allow passing arguments to the function from the location where it is called from.
- 'statements' is the function's body. It is a block of statements surrounded by braces { }that specify what the function actually does.

Let's have a look at an example:

```
// function example
#include <iostream>
using namespace
std;
int addition (int a, int b)
{
int r;
  r=a+b;
return r;
}
int main ()
{
int z;
  z = addition (5,3);
  cout <<"The result is "<< z;
}
```

This program is divided in two functions: addition and main. Remember that no matter the order in which they are defined, a C++ program always starts by calling main. In fact, main is the only function called automatically, and the code in any other function is only executed if its function is called from main (directly or indirectly).

Exercise –1 (5 points)

Write a program that will find the area and volume of sphere.Data type for all variable should be double

`double area (double radius):` function that will take radius as input and return the area.

`double volume (double radius):` takes radius as input argument and return the volume of sphere.

$$Area = 4\,\pi\,r^2$$

$$Volume = \frac{4}{3}\,\pi r^3$$

Write a main function that will take radius from the user and display the area and volume, don't display the area and volume in the function.

Paste screenshot of Output here…

Exercise –2 (5 points)

Write a void function **calculateRoots( )** that takes three input parameters 'a', 'b', 'c'; computes two roots and display them on console. You may use pre-defined functions for taking square and square root. Following expression calculates roots of a quadratic equation.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Write a main( ) to test your function. Take values of 'a', 'b', 'c' from user.

This program solve quadratic equation only for this condition.

$$b^2 - 4ac > 0$$

Paste screenshot of Output here…

Exercise –3 (5 points)

Write a function that takes a **four-digit number** as input in a single variable from user and returns the sum of its four digits. If user enters 2345, then output of program should be:14

- The input from user should be taken in main and then passed to the function as its parameter. The output from function should be displayed on console in main.
- Prototype should be `void four_digit_number(int number)`

Paste screenshot of Output here…

Exercise –4 (10 points)

Write a function that determines the median of three input numbers. The median is the middle number when the three numbers are arranged in order. However, the user can input the values in any order, so your program must determine which value is between the other two. For example, if the user enters:
41.52; 27.18; 96.03

Then the program would output:
The median of 41.52, 27.18 and 96.03 is 41.52

Paste screenshot of Output here…

Exercise –5 (10 points)

The sum of the squares of the first ten natural numbers is, $1^2 + 2^2 + ... + 10^2 = 385$.The square of the sum of the first ten natural numbers is, $(1 + 2 + ... + 10)^2 = 55^2 = 3025$.Hence the difference between the square of the sum of the first ten natural numbers and the sum of the squares of first ten natural numbers is $3025 − 385 = 2640$.

Find the difference between the square of the sum of the first ten natural numbers and the sum of the squares of first ten natural numbers

Paste screenshot of Output here…

Experiment No.10

Value and Reference parameters in Functions

OBJECTIVE:

- To get an understanding of reference parameters
- To learn the difference between value and reference parameters

Functions with Value Parameters

Function Declaration: A value parameter is used to pass information ***into*** a function to be processed. Remember the general form for a function declaration:

```
type function-name (formal parameter type list);
```

A void function with value parameters are declared by enclosing the list of types for the parameter list in the parentheses.

Example: A function that prints out a user specified number of horizontal lines is declared as:

```
// Purpose: Print out a number of lines
// Precondition: numOfLines has a value assigned.
// Postcondition: the number of lines are printed.
void PrintLines(int);
```

Function Activation: To activate a void function with value parameters, we specify the name of the function and provide the actual arguments enclosed in parentheses. The order and types of the list of arguments should correspond exactly to those of the formal parameters declared in the function prototype. The arguments can be constants, expressions, variables, or even function calls themselves. If an argument is a variable, at the time of function activation, the variable must have a value.

Example:

```
PrintLines(8);
            or

numOfLines = 5;
PrintLines(numOfLines);
```

Void Functions with Reference Parameters

Function Declaration: The second type of parameter in C++ is called a **reference parameter**. These parameters are used to send back a value (***output)***, or both to send in and out values (***input and output)*** from functions. Reference parameters have the ampersand (**&**) following their type identifier in the function prototype and function heading.

Example: A function that reads and returns (**output**) the length and width of a rectangle entered by the user, is declared as:

```
// Purpose: Read and return the length and width of a rectangle
// Postcondition: the values of the two arguements are read
void GetData(int& length, int& width);
```

Exercise – 1 (10 points)

Write the value of the variables in the boxes provided in Table 1.

Code – 1

```
# include <iostream.h>
int counter = 0;
int function1(int x, int& y)
{
   counter = counter +5;
   x = x*100;
   y = x*10;
   return x*y;
}
int function2(int& x,int& y,int &z)
{
   counter = counter * 10;
   x = y*z;
   y = x*y;
   z = z*z;
   return x+y+z;
}
```

Table 1

	Counter	Result	x	y	z

void main()
{
int x=10,y=100,z=1000;
int Result = 1;
counter++;					

Result = function1(x,y);					
x=1;y=2;z=3;					
Result =function2(x,y,z); }					

Code - 2 .Write answers in table 2

```
#include <iostream.h>
int GlobalVariable1 =100;
int GlobalVariable2 = 10;
int Area(int height,int width)
{
      height = 2+width;
      width = 5+height;
      int a = height*width;
      GlobalVariable1++;
      GlobalVariable2++;
      return a;
}
int Area1(int& height,int& width)
{
      int a = height*width;
      height = height/10;
      width = width/2;
```

```
            GlobalVariable1 = GlobalVariable1*10;
            GlobalVariable2 = GlobalVariable2*100;
            return a;
      }
```

Table 2

	GlobalVaraible1	GlobalVariable2	Width	Length	Area

`void main()`
`{`
`int Width = 10;`
`int Length = 20;`
`int area = 0;`					

`area = Area(Length,Width);`					
`area = Area(Length,Width);`					
`area = Area1(Width,Length);`					
`area = Area1(Length,Width);`					
`}`					

Exercise – 2 (10 points)

We need to design software for an ATM machine. An ATM card is used for money withdrawal. The systems needs following information of a card:

- Card Number
- Initial balance
- Type of card. (gold and silver)

InputCardDetails(..): This function should prompt the user to enter relevant details of ATM Card. Decide the prototype yourself. Remember that variables must be declared in main only.**The output from function should be displayed on console through main**. Write the main and test your program.

Paste screenshot of Output here…

Exercise – 3 (10 points)

In the previous exercise add the with draw money functionality.

WithdrawMoney(..): This function should take all the details of cards as input and ask the user the amount he/she wants to withdraw. Following rules should be observed:

- For Silver card: Money to be withdrawn cannot be more than 10000.
- For Gold card: Money to be withdrawn cannot be more than 25000.

Your program should keep displaying the menu after each function call unless user wants to exit.

Algorithm of main program is given below. Convert it into C++ to test your functions accordingly

- Declare all required variables in main
- Get input of a card from user
- Display the menu
- Withdraw some amount

Paste screenshot of Output here…

Exercise – 4 (10 points)

In the previous exercise add the functionality of balance enquiry or transfer money to other bank. For this add the following functions:

BalanceInquiry(..): This function generates a message that displays card #, and current balance.

TransferMoney(..): This function should ask the user the account number to which amount has to be transferred. It should keep a check on the balance before transferring any amount.

Your program should keep displaying the menu after each function call unless user wants to exit.

Algorithm of main program is given below. Convert it into C++ to test your functions accordingly

- Declare all required variables in main
- Get input of a card from user

- Display the menu
- Withdraw some amount
- Generate an inquiry about the balance
- Transfer some amount
- Generate an inquiry about the balance

Paste screenshot of Output here…

Experiment No.11

Introduction to arrays

OBJECTIVE:

Things that will be covered in today's lab:

- To learn how to declare a C++ array.
- To be able to know how to access values from the arrays.

An array is a collection of data elements of same data type. It is described by a single name and each element of an array is referenced by using array name and its subscript no.

Declaration of Array

Type arrayName[numberOfElements];

For example,

int Age [5];
float cost[30];

Initialization of One Dimensional Array

An array can be initialized along with declaration. For array initialization it is required to place the elements separated by commas enclosed within braces.

```
int A[5]={ 11,2,34,5,15};
```

It is possible to leave the array size open. The compiler will count the array size.

```
int B[]= { 6,7,8,9,15,12};
```

Referring to Array Elements

In any point of a program in which an array is visible, we can access the value of any of its elements individually as if it was a normal variable, thus being able to both read and modify its value. The format is as simple as:

```
name[index]
```

Examples:

```
cout<<age[4];
age[4]=55;
cin>>age[4];
```

Using loop to input an array from the user.

```
int age[10],i;
for( i=0; i<10; i++)
cin>>age[i];
```

Arrays as Parameters

At some moment we may need to pass an array to a function as a parameter. In C++ it is not possible to pass a complete block of memory by value as a parameter to a function, but we are allowed to pass its address.

For example, the following function:

```
void print( int A[])
```

accepts a parameter of type "array of int" called A. In order to pass to this function an array declared as:

```
int arr[20];
```

We need to write a call like this:

```
print(arr);
```

Exercise –1 (10 points)

Write a C++ program that includes the following

a) Declare an integer array of size 10
b) Ask user to enter 10 numbers for the array
c) Print out the array in index ascending order
d) Print out the array in index descending order
e) Find the sum of the array and print it out

Paste screenshot of Output here…

Exercise –2 (5 points)

Write a program that searches an array for a "**key value**" and returns the array's index for that element. If the element is not found, than the function should return -999. Write a main program that correctly tests this function. The main program should print the array and show the value of the search result.

Paste screenshot of Output here…

Exercise –3 (5 points)

Write a program that replaces the contents of each cell with the sum of the contents of all the cells in the original array from the left end to the cell in question.

Example:

Input:

5.8	2.6	9.1	3.4	7.0

Output:

5.8	8.4	17.5	20.9	27.9

Paste screenshot of Code here…

Exercise –4 (20 points)

A Polynomial of single variable is defined by its degree and coefficients for each term. For example, $5x^3+x^2+4$ has highest degree of 3(largest exponent of the variable), and coefficients for all terms are as follows:

degrees	Coefficient
0	4
1	0
2	1
3	5
4	0
5	0
6	0
7	0
8	0
9	0

a) Define an array of polynomial assuming maximum number of coefficients in a polynomial can be 10.
b) Add two polynomials and display their result in the main.
c) Take a polynomial and a number from the user, display true if that number is factor of that polynomial. For example for polynomial $5x^5+3x^2+2$ and number -1, polynomial evaluation is $=5(-1)^5 + 3(-1)^2 + 2 = 5(-1) +3(1) +2 =0$ => hence -1 is factor of above mentioned polynomial.

Paste screenshot of Output here…

Exercise –5 (20 points)

Suppose A, B, C are arrays of integers of size M, N and M+N respectively.

- The number in array A appear in ascending order
- While the numbers in array B appear in descending order.

Write a user program in C++ to produce third array C by emerging arrays s A and B in ascending order.

Paste screenshot of Output here…

Experiment No.12

Function and arrays

OBJECTIVE:

- To be able to work with character arrays.
- To be able to understand the working of arrays with functions.

String and Character array

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A **string** is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

For example: The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.

Declaring and Initializing a string variables

There are different ways to initialize a character array variable.

```
char name[10]="StudyTonight";   //valid character array initialization

char name[10]={'L','e','s','s','o','n','s','\0'};//valid initialization
```

Remember that when you initialize a character array by listings all its characters separately then you must supply the '\0' character explicitly.Some examples of illegal initialization of character array are,

```
char ch[3]="hell";   //Illegal
char str[4];
str="hell";   //Illegal
```

Arrays with Functions:

C++ does not allow to pass an entire array as an argument to a function. However, You can pass a pointer to an array by specifying the array's name without an index.

If you want to pass a single-dimension array as an argument in a function, you would have to declare function formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received.

Way-1

Formal parameters as a pointer as follows:

```
void myFunction(int*param)
{
      \\ Function definition
}
```

Way-2

Formal parameters as a sized array as follows:

```
void myFunction(int param[10])
{
      \\ Function definition
}
```

Way-3

Formal parameters as an unsized array as follows:

```
void myFunction(int param[])
{
      \\ Function definition
}
```

Now, consider the following function, which will take an array as an argument along with another argument and based on the passed arguments, it will return average of the numbers passed through the array as follows:

```
double getAverage(int arr[],int size)
{
int    i, sum =0;
double avg;
for(i =0; i < size;++i)
    sum += arr[i];
  avg =double(sum)/ size;

return avg;
}
```

Now, let us call the above function as follows:

```
#include<iostream>
usingnamespace std;
// function declaration:
double getAverage(int arr[],int size);
int main ()
{
// an int array with 5 elements.
int balance[5]={1000,2,3,17,50};
double avg;
// pass pointer to the array as an argument.
   avg = getAverage( balance,5);
// output the returned value
   cout <<"Average value is: "<< avg << endl;
return0;
}
```

When the above code is compiled together and executed, it produces the following result:

```
Average value is:214.44
```

As you can see, the length of the array doesn't matter as far as the function is concerned because C++ performs no bounds checking for the formal parameters.

Exercise –1 (10 points)

Write a C++ program that asks the user to enter a phrase and then a single letter. The program should then report how many times that letter occurred in the phrase, displaying the letter with double quotes around it. An example is shown below where the user searched for lower-case "r". **Your program should use functions appropriately.**

Example:
Input:
Phrase: Programming requires effort and perseverance.
letter : r
Output:
The letter "r" occurred 6 times.

Paste screenshot of Code and Output here…

Exercise –2 (10 points)

Write a program that will find the number of vowels in the string.
Define a function that must take the whole string from the main and print the number of vowels appears in the string.

Example:

Input: hard work is a key to success.
Output: 8

Paste screenshot of Code and Output here…

Exercise –3 (10 points)

You are required to write a program which will take input from user in two integer arrays. The program should compare both arrays for checking if both arrays are totally identical (Exactly same). Write a boolean function which return true if arrays are totally identical otherwise return false. Example of two identical (Exactly same) arrays:

Example of two identical (Exactly same)arrays :

Array1	Array2
1	1
2	2
3	3
5	5
7	7
9	9

Main function:
Declare and initialize two integer Arrays;
IsIdenticalArrays(Array1,Array2);
Output if arrays are equal or not;

Paste screenshot of Output here…

Exercise – 4 (10 points)

Write a function PickLarger( ) which compares the corresponding elements of array, picks the larger element among them and saves the larger element in the output array.

Sample Input and Output:

```
Input:
Array 1: 1 2 1 5 9   1   3 5
Array 2: 8 2 1 0 11 10 3 4

Output:
Output Array: 8 2 1 5 11 10 3 5
```

Paste screenshot of Output here…

Exercise – 5 (10 points)

Encryption can be very useful method for protecting sensitive data from intruders. Data is converted into meaningless form by the usage of a given key and it is converted back to original form by utilizing the key again when required. You need to design an encryption\decryption application.

Example:

Inputs:
Text: hello world , Key: 3
Outputs:
Encrypted text: khoor zruog , Decrypted text: hello world

Note:

1. Use only character arrays.
2. String Library functions are not allowed.
3. Make separate functions for encryption and decryption and write down the **main function** as well to make the code interactive.
4. Usage of temporary arrays is prohibited. Both encryption and decryption should be performed on original array.

Paste screenshot of Output here…

Appendix A: Lab Evaluation Criteria

Labs with projects

1. Class Participation 10%
2. Lab Work 40%
3. Quiz (2) 20%
4. Project 30%
    a. Project Demonstration 20%
    b. Project Report 5%
    c. Project Quiz 5%

Labs without projects

1. Class Participation 10%
2. Lab Work 40%
3. Quiz (2) 20%
4. Lab Final 30%
    a. Lab Final (Practical) 20%
    b. Lab Final (Written) 10%

Appendix B: Safety Around Electricity

In all the Electrical Engineering(EE) labs, with an aim to prevent any unforeseen accidents during conduct of lab experiments, following preventive measures and safe practices shall be adopted:

- Remember that the voltage of the electricity and the available electrical current in EE labs has enough power to cause death/injury by electrocution. It is around 50V/10 mA that the "cannot let go" level is reached. "The key to survival is to decrease our exposure to energized circuits."
- If a person touches an energized bare wire or faulty equipment while grounded, electricity will instantly pass through the body to the ground, causing a harmful, potentially fatal, shock.
- Eachcircuitmustbeprotectedbyafuseorcircuitbreakerthatwillblowor"trip"whenits safecarryingcapacityissurpassed.Ifafuseblowsorcircuitbreakertripsrepeatedlywhile in normal use (not overloaded), check for shorts and other faults in the line or devices. Do not resume use until the trouble is fixed.
- It is hazardous to overload electrical circuits by using extension cords and multi-plug outlets. Use extension cords only when necessary and make sure they are heavy enough for the job. Avoid creating an "octopus" by inserting several plugs into a multi-plug outlet connected to a single wall outlet. Extension cords should ONLY be used on a temporary basis in situations where fixed wiring is not feasible.
- Dimmed lights, reduced output from heaters and poor monitor pictures are all symptoms of an overloaded circuit. Keep the total load at any one time safely below maximum capacity.
- If wires are exposed, they may cause a shock to a person who comes into contact with them. Cords should not be hung on nails, run over or wrapped around objects, knotted or twisted. This may break the wire or insulation. Short circuits are usually caused by bare wires touching due to breakdown of insulation. Electrical tape or any other kind of tape is not adequate for insulation!
- Electrical cords should be examined visually before use for external defects such as: Fraying(wornout)andexposedwiring,looseparts,deformedormissingparts,damageto outer jacket or insulation, evidence of internal damage such as pinched or crushed outer jacket. If any defects are found the electric cords should be removed from service immediately.
- Pull the plug not the cord. Pulling the cord could break a wire, causing a short circuit.
- Plug your heavy current consuming or any other large appliances into an outlet that is not sharedwithotherappliances.Donottamperwithfusesasthisisapotentialfirehazard.Do not overload circuits as this may cause the wires to heat and ignite insulation or other combustibles.
- Keep lab equipment properly cleaned and maintained.
- Ensure lamps are free from contact with flammable material. Always use lights bulbs with the recommended wattage for your lamp and equipment.
- Be aware of the odor of burning plastic or wire.

- ALWAYS follow the manufacturer recommendations when using or installing new lab equipment. Wiring installations should always be made by a licensed electrician or other qualifiedperson.Allelectricallabequipmentshouldhavethelabelofatestinglaboratory.
- Be aware of missing ground prong and outlet cover, pinched wires, damaged casings on electrical outlets.
- Inform Lab engineer / Lab assistant of any failure of safety preventive measures and safe practices as soon you notice it. Be alert and proceed with caution at all times in the laboratory.
- Conduct yourself in a responsible manner at all times in the EE Labs.
- Follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ASK YOUR LAB ENGINEER / LAB ASSISTANT BEFORE PROCEEDING WITH THEACTIVITY.
- Neverworkaloneinthelaboratory.NostudentmayworkinEELabswithoutthepresence of the Lab engineer / Lab assistant.
- Perform only those experiments authorized by your teacher. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed.
- BepreparedforyourworkintheEELabs.Readallproceduresthoroughlybeforeentering the laboratory. Never fool around in the laboratory. Horseplay, practical jokes, and pranks are dangerous and prohibited.
- Always work in a well-ventilated area.
- Observe good housekeeping practices. Work areas should be kept clean and tidy at all times.
- Experiments must be personally monitored at all times. Do not wander around the room, distract other students, startle other students or interfere with the laboratory experiments of others.
- Dress properly during a laboratory activity. Long hair, dangling jewelry, and loose or baggy clothing are a hazard in the laboratory. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured. Shoes must completely cover the foot.
- Know the locations and operating procedures of all safety equipment including fire extinguisher. Know what to do if there is a fire during a lab period; “Turn off equipment, if possible and exit EE lab immediately.”