Kuwait University
College of Life Sciences
Department of Information Science

Lab Manual

ISC 115 Computing Foundations

Prepared by

Professor Mostafa Abd-El-Barr

Dr. Mariam Al-Otaibi

Eng. Aisha Al-Noori

Revised 2018

Table of Contents

Laboratory Hardware and Software/Tools Requirements

In this lab the students will be using Dev C++.

Laboratory Schedule

#	Lab Title	Lab activity
1	Introduction & Input/Output Statements	Exercise # 1
2	Variables, Constant & Operators	Exercise # 2
3	Control Structures (I)	Exercise # 3
4	Control Structures (II)	Exercise # 4
5	Arrays I	Exercise # 5
6	Arrays II	Exercise # 6
7	Functions	Exercise # 7
8	Applications I: C++ Program for Simple Statistical Applications	Exercise # 8
9	Applications II: C++ Program to Compute a Finite Series	Exercise # 9
10	Applications III: C++ Program for Matrix Applications	Exercise # 10

Laboratory Policy

- Please follow the laboratory rules listed in appendix “A”
- To pass this course, the student must pass the lab-component of the course.
- Cheating in whatever form will result in (F) grade.
- Attendance will be checked at the beginning of each Lab.
- Number of absence hours will be combined with the absence hours of the course and they are subject for applying the university absence regulations.
- Cheating in Lab Work or Lab Project will result (F) grade in Lab.
- There will be no make-up for any Quiz/Exam/Lab.
- Hard work and dedication are necessary ingredients for success in this course.

Laboratory Grading Policy

Activity	Weight
Lab Work (10 x 0.5%)	5%
Lab Quizzes (5 x 1%)	5%
Lab Final Exam	5%
Total	15%

Introduction

This lab is an integral part of the course ISC 115 Computing Foundation. The main objective of the lab is to introduce C++ programming language and to use it to solve different computing problems.

Familiarity with Lab Hardware and Software tools

General Introduction to the Laboratory as being an integral part of the course ISC 115 Computing Foundations.

Steps to start a C++ program in Visual C

1- From Menu: File -> New
2- Choose "Files" Tab, then "C++ Source File"
3- Put your "File Name" and "Location"
4- Write your code

Steps to run a C++ program in Visual C

1- From Menu: Build -> Compile program
2- From Menu: Build -> Build
3- From Menu: Build -> Execute program

Steps to print a C++ run in Visual C

1- From Menu: Build -> Execute program
2- Press Print-Scrn button
3- Open a word document
4- Paste
5- Write a reference for the pasted image

Laboratory Tools Setup

Install Microsoft Visual C++ Version 6.0

1. Start the Visual C++ Installation Wizard and click Next.

2. If you agree to the license, put the checkmark and click Next.

3. When asked for the installation location, choose `c:\Program Files\Microsoft Visual Studio\Common` and click Next.

4. To continue with the installation process, click Next.

5. Note down the Product ID and click OK.

6. When asked for the setup type, click on the Standard button.

7. When asked to register the environment variables, you can leave the option to register the environment variables unchecked, though it does no harm to check this option. Click OK.

8. Note that the Visual C++ setup has finished successfully and click OK.

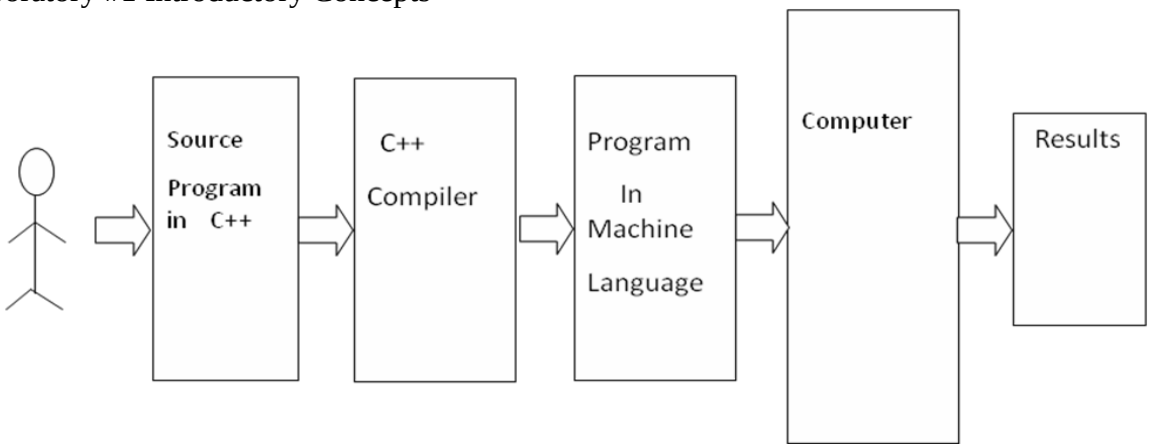
Laboratory #1 – Introduction & Input/Output Statements

1. Laboratory Objective

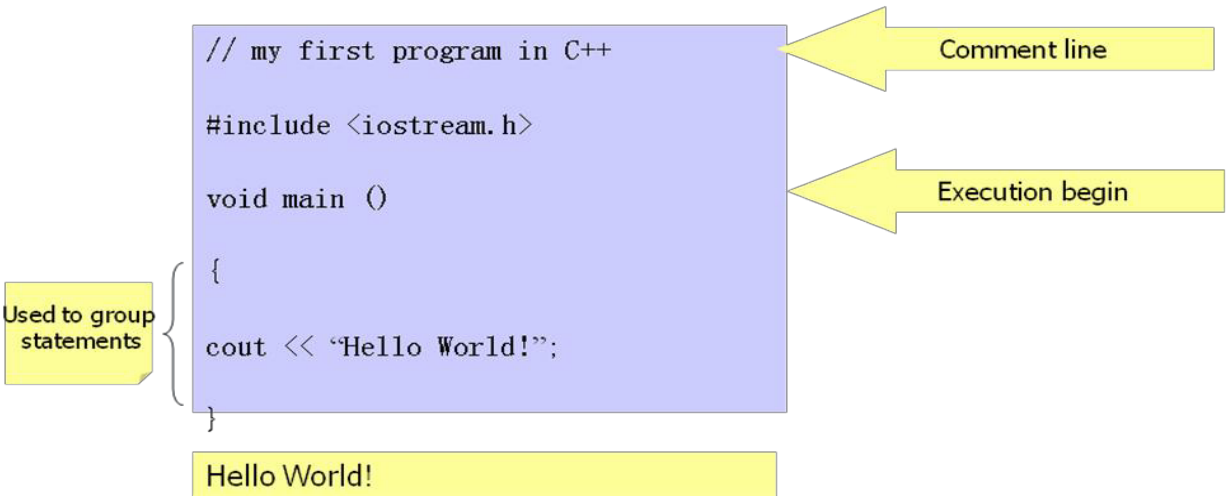
   The objective of this laboratory is to introduce students to the basic concepts in C++ Programming Language as well as the basic forms for the Input and the Output Statements in the Language.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:
   Type, edit, compile, debug, and run simple C++ Programs.
   b. Write, edit, compile, debug, and run a program that includes simple Input and Output Statements in the C++ Language.

3. Laboratory #1 Introductory Concepts



Cout << expression << expression ….;



```
/* my second program in C++
   With more comment */
#include <iostream.h>
void main ()
{
cout << "Hello World!"; // says Hello World
cout << "I am a C++ program"; // says I am a C++ program
}
```

\n	new line
\r	carriage return
\t	Tabulation
\v	vertical tabulation
\b	Backspace
\f	page feed
\a	alert (beep)
\'	single quotes (')
\"	double quotes (")
\?	question (?)
\\	inverted slash

```
#include <iostream.h>
/* Created By: Maryam Al-Otaibi
   Creation Date:  13 June 2006
   Description:  .. */
void main ()
{
int i;
cout << "Please enter an integer value: ";
cin >> i;
cout << "The value you entered is " << i;
cout << " and its double is " << 2*i << ".\n";
}
```

4. Laboratory Problem Description

In this laboratory, you are required to type, edit, compile and execute the following C++ program:

```
#include<iostream.h>

void main()

{

      cout<<"Welcome to ISC 115 Laboratory\n";

}
```

Use Microsoft visual C++ to compile and execute the above program.

Write a C++ program that inputs three integers by the user and displays the three integers, their sum, average and product.

Sample Output

```
Enter first integer: 2

Enter second integer: 4

Enter third integer: 6

The sum is 12

The product is 48

The average is 4
```

5. Laboratory Instructions

(a) Some Good Programming Practices

- Every program should begin with a comment that explains the purpose of program, author and creation date and last updated date.
- Use blank lines, space characters and tabs to enhance program readability.
- Many programmers make the last character printed a newline (`\n`). This ensures that the function will leave the screen cursor positioned at the beginning of a new line.
- Indent the entire body of each function one level within the braces that delimit the body of the function. This makes the program easier to read.
- Place a space after each comma (`,`) to make programs more readable.

(b) Some Common Programming Mistakes

- Forgetting to include the `<iostream>` header file in a program that inputs data from the keyboard or outputs data to the screen cause the compiler to issue an error message.
- Omitting the semicolon at the end of C++ statement is a syntax error.

6. Laboratory #1 Exercises

Write a C++ program that displays the following shape:

Output:

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

*       *

*       *

*       *

*       *

*       *

*       *

*       *

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

Laboratory #2– C++ Variables, Constant & Operators

1. Laboratory Objective

   The objective of this laboratory is to introduce students to the basics of declaring Variables, and Constants as well as the use of different Operators in the C++ Language.

2. Laboratory Learning Outcomes: After conducing this laboratory students will be able to:
   a. Declare Variables and Constants and use them in Simple C++ Programs.
   b. Use different types of Operators, such as Arithmetic, Relational, and Compound Operators in Simple C+++ Programs.

3. Laboratory #2 Introductory Concepts

   Data_Type identifier, identifier, …;
   Data_Type identifier = initial_value;
   const Data_Type identifier = value;
   Variable = expression;

Name	Bytes	Description	Range*
Char	1	character or integer 8 bits length.	**signed:** -128 to 127 **unsigned:** 0 to 255
short	2	integer 16 bits length.	**signed:** -32768 to 32767 **unsigned:** 0 to 65535
Long	4	integer 32 bits length.	**signed:**-2147483648 to 2147483647 **unsigned:** 0 to 4294967295
Int	*	Integer. Its length traditionally depends on the length of the system's Word **type**, thus in MSDOS it is 16 bits long, whereas in 32 bit systems (like Windows 9x/2000/NT and systems that work under protected mode in x86 systems) it is 32 bits long (4 bytes).	See **short**, **long**
float	4	floating point number.	3.4e + / - 38 (7 digits)
double	8	double precision floating point number.	1.7e + / - 308 (15 digits)
long double	10	long double precision floating point number.	1.2e + / - 4932 (19 digits)
bool	1	Boolean value. It can take one of two values: true or false NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it. Consult section bool type for compatibility information.	**true** or **false**

Summary of Operators

- Boolean Operators: &&, ||, !
- Arithmetic Operators: +, -, *, /, %
- Equality Operators: <, >, ==, <=, >=, !=
- Assignment Operators: =, +=, -=, *=, /=, %=

Operator	Name
!	Boolean NOT
*, /, %	Multiplication, Division, Mod
+, -	Addition, Subtraction
<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or Equal to
==, !=	Is equal to, is NOT equal to
&&	Boolean AND
||	Boolean OR
=, *=, /=, %=, +=, -=	Assignment, Multiply and assign, Divide and assign, mod and assign, Add and assign, Subtract and assign

4. Laboratory Problem Description

In this laboratory you are required to perform the following:

1. Write a C++ program that declares and initializes an integer then displays the integer raised to the power of 2.

2. Write a C++ program that declares and initializes three variables of type double. The program should display the sum, average, and product of the numbers in the command window.

5. Laboratory Instructions

(a) Some Good Programming Practices

- ✓ Many programmers prefer to declare each variable on a separate line. This format allows for easy insertion of a descriptive comment next to each declaration.
- ✓ Choosing meaningful variables makes a program self-documenting.
- ✓ Avoid using abbreviation in identifiers. This promotes program readability.
- ✓ Place spaces on either side of a binary operator. This makes the operator stand out and makes the program more readable.

(b) Some Common Programming Mistakes

- Attempting to use the modulus operator (%) with non-integer operands is a compilation error.
- A syntax error will occur if any of the operators ==, !=, >= and <= appears with spaces between its pair of symbols.
- Reversing the order of the pair of symbols in any of the operators !=, >= and <= is normally a syntax error.
- Confusing the equality operator == with the assignment operator = results in logic errors.

6. Laboratory #2 Exercises

Write a C++ program that declares and initializes the radius of a circle. The program should output the area and circumference. Define **π** as a constant in your program.

Area of a circle = **π radius^2**

Circumference of a circle = **2 π radius**

π = 3.14159

Sample Output

```
The radius of the circle is 3

The area of the circle is 28.2743

The circumference of the circle is 18.8495
```

Laboratory #3 – Control Structures I

1. Laboratory Objective

   The objective of this laboratory is to train students on how and when to use different C+++ control structures.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:
   a. Identify and practice using the different C++ Control Structures.
   b. Practice using the C++ Branching and Switch Statements in Simple C++ Program Applications.

3. Laboratory #3 Introductory Concepts
   - Control Structure are ways for programmers to control what pieces of a program are to be executed at certain times.
   - There are two basic types of Control Structures: Branching statements and Loops.

```
If (Condition) Statement;
If (Condition) Statement_1 else Statement_2

Switch (expression) {
Case constant_1;
        Block of Statements_1
        Break;
Case constant_2;
        Block of Statements_2
        Break;
……
 Default:
        Default Block of Statements
}
```

4. Laboratory Problem Description

Write a C++ program that prompts the user to input two integer variables. The program should prompt the user asking which operation need to be performed as follows:

1. Addition
2. Subtraction
3. Multiplication
4. Division (the second number should not be equal to zero)
5. Compare the two numbers (determining if the first number is larger, smaller or equal to the second number)

When the user inputs his/her choice a `switch` statement should b used to process the user's input and display the result.

Sample Output

```
Enter first integer:  3

Enter second integer:  8

choose from the menu

1- Addition

2- Subtraction

3- Multiplication

4- Division

5- Compare the two numbers

Enter your choice:  1

3 + 8 = 11
```

5. Laboratory Instructions

(a) Good Programming Practices

- ✓ Apply reasonable indentation in you program to make it more readable.
- ✓ Indent both body statements of an `if…else` statement.
- ✓ Always putting the braces in an `if…else` statement (or any control statement) helps prevent their accidental omission.

(b) Some Common Programming Mistakes

- Using a keyword as an identifier is a syntax error.
- Spelling a keyword with any uppercase letter is a syntax error. All of C++'s keyword contain only lowercase letters.
- Forgetting one or both of the braces that delimit a block can lead to syntax errors or logic errors in a program.
- Placing a semicolon after the condition in an `if` statement leads to logic error in single-selection `if` statements and a syntax error in double-selection `if…else` statements.

6. Laboratory #3 Exercises

Write a C++ program that prompts the user to input an integer variable. The program should then print whether the number is odd or even.

Sample Output

```
Enter an integer:  9

9 is odd
```

Laboratory #4 – Control Structures II

1. Laboratory Objective

   The objective of this laboratory is to train students on how and when to use different C+++ control structures.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:
   a. Identify and practice using the different C++ Control Structures.
   b. Practice using the C++ Loop Structure in Simple C++ Program Applications.

3. Laboratory #4 Introductory Concepts

   - **There are three types of loops: for, while, and do..while.**
   - **Each of loop statements has their specific uses.**

   for (variable initialization; condition; variable update) { Code to execute while the condition is true }

   while (condition) { Code to execute while the condition is true }

   do { Code to execute} while (condition);

4. Laboratory Problem Description

In this laboratory you are required to perform the following:

1 Write a C++ program that uses `while` loop to perform the following steps:

   a. Prompt the user to input two integers: `firstNum` and `secondNum` (`firstNum` must be less than `secondNum`). If the user enters the first number that is greater than or equal to the second number, an error message should be printed and the program should read the numbers again until the user satisfies the previous condition.
   b. Output all even numbers between `firstNum` and `secondNum`.
   c. Output the sum of all odd numbers between `firstNum` and `secondNum`.

Sample Output

```
Enter two integers, such that the first number is less than the second number

9  2

Try again ! first number should be less than second number

2  9
```

```
the even numbers are: 4  6  8

the sum of odd numbers equals to 15
```

2 Write a C++ program that inputs a series of 10 integers and determines and prints the largest integer. Your program should use at least the following three variables:

a. `counter`: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).
b. `number`: The integer most recently input by the user.
c. `largest`: The largest number found so far.

Sample Output

```
Enter 10 integers
Enter number: 45
Enter number: 2
Enter number: 5
Enter number: 11
Enter number: 2
Enter number: 100
Enter number: 2
Enter number: 99
Enter number: 100
Enter number: 33
The largest is 100
```

3 Write a C++ program that inputs the size of the triangle and the shape, then it outputs the triangle using the shape entered. Use a nested loop to output the triangle.

Sample Output

```
Enter the size of the triangle: 7
Enter the shape: #

#
##
###
####
#####
######
```

4 Write a C++ program that inputs a series of 10 integers and calculates and prints the average number. The sum and average should be of type double.

Sample Output

```
Enter 10 integers
Enter number: 20
Enter number: 19
Enter number: 12
Enter number: 9
Enter number: 17
Enter number: 15
```

```
Enter number: 16
Enter number: 18
Enter number:  8
The average is 13.4
```

5 Laboratory Instructions

(a) Some Good Programming Practices

- ✓ Declare each variable on a separate line with its own comment to make programs more readable.
- ✓ Prompt the user for each keyboard input. The prompt should indicate the form of the input and any special input values.

(b) Common Programming Mistakes

- Not providing, in the body of a `while` statement, an action that eventually causes the condition in the `while` to become false normally results in a logic error called an infinite loop, in which the repetition statement never terminates.
- Not initializing counters and totals can lead to logic errors.
- Using a loop's counter-control variable in a calculation after the loop often causes a common logic error called an off-by-one error. In a counter-controlled loop that counts up by one each time through the loop, the loop terminates when the counter's value is one higher than its last legitimate value.
- An attempt to divide by zero normally causes a fatal runtime error.
- Omitting the braces that delimit a block can lead to logic errors, such as infinite loops.
- Using an incorrect relational operator or using an incorrect final value of a loop counter in the condition of a `while` or `for` statement can cause off-by-one errors.
- When the control variable of a for statement is declared in the initialization section of the `for` statement header, using the control variable after the body of the statement is a compilation error.
- Using commas instead of the two required semicolon in a `for` header is a syntax error.

6 Laboratory #4 Exercises

Write a C++ program that calculates and prints the product of the odd integers from 1 to 15.

Sample Output

```
the product of the odd numbers from 1 to 15 is 2027025
```

Laboratory #5 – Arrays I

1. Laboratory Objective

   The objective of this laboratory is to train students on how to use Arrays data structure in the context of the C++ Programming Language.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

   a. Declare one-Dimensional Arrays.
   b. Use one-Dimensional Arrays in Simple Applications.
   c. Apply C++ basic concepts in dealing with Arrays.

3. Laboratory #5 Introductory Concepts

   - An array consists of a series of elements (variables) of the same type that are placed consecutively in the computer's memory such that the elements can be individually referenced by adding an index to the array name.
   - Example: A -12 36 -120 72 255
     A [0] A[1] A[2] A[3] A[4]

     Data_Type Array_Name [elements];
     Data_Type Array_Name [elements]= {value_1, value_2, …, value_n-1};

   - Example

```
#include <iostream.h>
/* Created By:        Maryam Al-Otaibi
    Creation Date:   13 June 2006
    Description:        Array example */
int item [] = {16, 2, 77, 40, 1207};
int n;
int result=0;
 void main ()
{
 for ( n=0 ; n<5 ; n++ )
  {
   result = result+item[n];
  }
 cout << result;
}
```

   - **Printing individual elements of an array**
     cout<<"item[2]= "<<item[2];
   - **Printing a whole array**
     for (i=0;i<5;i++)
     cout<<item[i]<<"\t ";
     cout<<"\n";

4. Laboratory Problem Description

In this laboratory you are required to perform the following:

1. Write a C++ program that creates an arithmetic sequence. It should prompt the user to input the first term in the sequence `first` and the difference between two successive terms `diff`. The program then creates a one-dimensional array with 10 elements ordered in an arithmetic sequence. It also outputs the arithmetic sequence.

Sample Output

```
Enter the first term of the sequence:  5
Enter the difference between successive terms:  10
5  15  25  35  45  55  65  75  85  95
```

2. Write a C++ program that inputs 10 integers, stores it in a one dimensional array and calculates and outputs the average of the integers. It also counts and outputs the number of integers that are greater than the average and the number of integers that are less than the average.

Sample Output

```
Enter number 1 :  58
Enter number 2 :  66
Enter number 3 :  75
Enter number 4 :  98
Enter number 5 :  100
Enter number 6 :  82
Enter number 7 :  94
Enter number 8 :  87
Enter number 9 :  71
Enter number 10 :  96
average = 82.7
5 numbers are greater than the average
5 numbers are less than the average
```

5. Laboratory Instructions
   Please follow our hints for "Good Programming Practices" and avoid our hints for "Common Programming Mistakes" that were mentioned in previous labs.

6. Laboratory #5 Exercises

Lab Exercise 7

Write a C++ program that inputs 10 integers, stores it in a one dimensional array and calculates and outputs the average of the integers. It also counts the number of integers that are greater than the average and the number of integers that are less than the average. It then displays the result using a bar charts.

Sample Output

```
Enter number 1 :  58
Enter number 2 :  66
Enter number 3 :  75
Enter number 4 :  98
Enter number 5 :  100
Enter number 6 :  82
Enter number 7 :  94
Enter number 8 :  87
Enter number 9 :  71
Enter number 10 :  96
average = 82.7
greater than average:*  *  *  *  *
less than average:   *  *  *  *  *
```

Laboratory #6 – Arrays II

1. Laboratory Objective

The objective of this laboratory is to train students on how to use Arrays data structure in the context of the C++ Programming Language.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

a. Declare two-Dimensional Arrays.
b. Use two-Dimensional Arrays in Simple Applications.
c. Apply C++ basic concepts in dealing with Arrays.

3. Laboratory #6 Introductory Concepts

- Two dimensional arrays can be described as "arrays of arrays".
- Example: List [3][5]

	0	1	2	3	4
0					
1					
2					

Data_Type Array_Name [rows][columns];
Data_Type Array_Name [rows][columns]= {{value_0,0,value_0,1,…},{…..},{value_n-1,0, ….,value_n-1,m-1}};

- Printing indivuidual array element
  cout<<"list[2][1]= "<<list[2][1];

- Printing a whole two-Dimensional array

```
for ( i=0;i<3;i++)
        { for ( j=0;j<4;j++)
                cout<<a[i][j]<<" ";
          cout<<"\n";
```

4. Laboratory Problem Description

In this laboratory you are required to perform the following:

1. Write a C++ program that declares a two-dimensional array that has 5 rows and 5 columns. Initialize the array such that if the sum of indices is even, the element should equal to the sum of indices. Otherwise, the element should equal to the product of indices. Print the two dimensional array. The output should be as close to a matrix form as possible.

Sample Output

```
0  0  2  0  4
0  2  2  4  4
2  2  4  6  6
0  4  6  6  12
4  4  6  12  8
```

2. Write a C++ program for performing operations on a 2 x 2 matrix. Your program should do the following operations:
   a. Input the elements of the matrix.
   b. Output the elements of the matrix.
   c. Find if the matrix is symmetric or not.

Sample Output

```
element (0 , 0): 2
element (0 , 1): 5
element (1 , 0): 5
element (1 , 1): 2
2  5
5  2
It is a symmetric matrix
```

5. Laboratory Instructions

Please follow our hints for "Good Programming Practices" and avoid our hints for "Common Programming Mistakes" that were mentioned in previous labs.

6. Laboratory #6 Exercises

Write a C++ program that output the 1 to 10 times tables. It should declare a two-dimensional array that has 11 rows and 11 columns. It should initialize the array such that each element of the array equals the product of indices. Print the 1 to 10 times table as shown in sample output.

Sample Output

```
The 1 times table
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
The 2 times table
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
```

```
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

…
…
…
The 10 times table
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

Laboratory #7– Functions

1. Laboratory Objective

The objective of this laboratory is to train students on how to use Functions as an important C++ Programming tool in modular programming.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

   a. Declare C++ functions in C+++ Programs.
   b. Apply the concept of functions in developing modular C++ Programs.

3. Laboratory #7 Introductory Concepts
   - Functions allow you to do a whole lot in one line of code.
   - In C++ Functions allow you to group a series of steps under one name.
   - Functions can make your code shorter when you call them repeatedly.
   - You can deal with a function as a black box

   - You can imagine that the "black box" takes some inputs, performs some processing on the input(s) inside the box, and produces some output(s). How a function will use the inputs to come up with the outputs is the essence of the function.

   - Sample function declaration

```
type function_name (argument_1, argument_2, argument_..) {
      // function body (code to execute inside function)
 }
```

   - Type: is the type of data returned by the function. Or void for a function with no value returned.
   - Function_name: is the name by which the function can be called.
   - Arguments: Each argument consists of a type of data followed by its identifier, like variable declaration (exe: int x). Or void for a function without parameters.
   - Function body: It can be a single instruction or a block of instructions delimited by curly brackets {}.

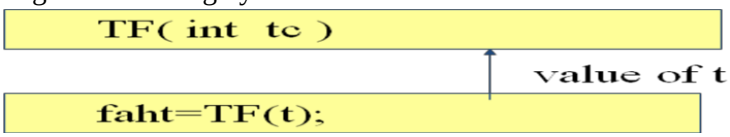
   - Example: A program that converts a temperature in Celsius into Fahrenheit

```
 #include <iostream.h>
int TF(int tc)
{
     int r;
     r=(9/5)*tc+32;
     return (r);
}
void main ()
{
     int t, faht;
     cout<< "Please enter temperature in Celsius to be converted to
Fahrenheit\n";
     cin>>t;          // read the temperature Celsius
     faht=TF(t);     // convert the Celsius  into Fahrenheit
     cout << "temperature in Fahrenheit = "<< faht<<"\n\n";
}
```

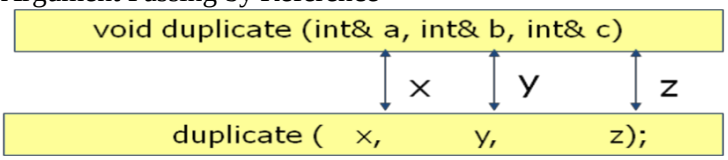
- Scope of variables

```
#include <iostream.h>
        int a, b,c; /* set of Global Variables */
int function_name
{
        int list[10], x, y; /* Local Variables */
        Int x, y;
……….
}
main()
        int n; /* Local Variable*/
        float number, age;
……..
}
```

- Argument Passing by Values



- Argument Passing by Reference



- Recursive Function

```
// factorial calculator
 #include <iostream.h>
long factorial (long a)
{
   if (a > 1)
      return (a * factorial (a-1));
  else
      return (1);
 }
void main ()
{
  long number;
  cout << "Please type a positive integer number: ";
  cin >> number;
  cout << number << "! = " << factorial (number);
}
```

4. Laboratory Problem Description

In this laboratory you are required to perform the following:

write a function `integerPower(base, exponent)` that returns the value of $base^{exponent}$. For example, `integerPower(3, 4)` = 3^4 = 3 * 3 * 3 * 3. Assume `exponent` is a positive, nonzero integer and that `base` is an integer. The function `integerPower` should use `for` or `while` to control the calculation. Use this function in a program that inputs the `base` and `exponent` then it calls the function `integerPower` to display the result.

Sample Output

```
enter the base 3
enter the exponent 4
3 raised to the power of 4 is 81
```

5. Laboratory Instructions

Some Common Programming Mistakes

- Declaring function parameters of the same type as `double x, y` instead of `double x, double y` is a syntax error.
- Compilation errors occur if the function prototype, function header and function calls do not all agree in the number, type and order of arguments and parameters, and in return type.
- If a function is defined before it is invoked, then the function's definition also serves as the function's prototype, so a separate prototype is unnecessary. If a function is invoked before it is defined, and that function does not have a function prototype, a compilation error occurs.
- It is a compilation error if two functions in the same scope have the same signature but different return types.

6. Laboratory #7 Exercises

Write a program that inputs 5 integers and passes them one at a time to function `even`, which uses the modulus operator to determine whether an integer is even. The function should take an integer argument and print whether the number is even or not.

Sample Output

```
Enter an integer:  5
5 is odd
Enter an integer:  100
100 is even
Enter an integer:  93
93 is odd
Enter an integer:  28
28 is even
Enter an integer:  10
10 is even
```

Laboratory #8 – Applications I

1. Laboratory Objective

   The objective of this laboratory is to train students on how to integrate the knowledge gained in the C++ programming and apply it skillfully in programming solutions for real life problems.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

   a. Apply the programming skills gained throughout the C++ Programming in solving real life problems.
   b. Gain appropriate experience in developing modular C++ Programs.

3. Laboratory #8 Introductory Concepts

4. Laboratory Problem Description

Write a C++ program that inputs 10 scores, stores it in a one dimensional array and computes the standard deviation of the 10 scores. The formula for computing the standard deviation is:

$$deviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean)^2}{n-1}} \qquad mean = \frac{\sum_{i=1}^{n} x_i}{n}$$

To compute `deviation` with this formula, you have to store the individual scores using an array, so that they can be used after the `mean` is obtained. Use `{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}` to test the program.

Sample Output

```
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

The mean is 5.5

The standard deviation is 3.028
```

5. Laboratory Instructions

   Please follow our hints for "Good Programming Practices" and avoid our hints for "Common Programming Mistakes" that were mentioned in previous labs.

6. Laboratory Exercises

Write a function `drawHistograms` that accepts one integer parameter `n` and displays `n` adjacent asterisks. Use this function in a program that inputs (reads) five integers, passes the integers to function `drawHistograms` one at a time to display the adjacent asterisks.

Sample Output

```
Enter a number 1
 *
Enter a number 2
 *  *
Enter a number 3
 *  *  *
Enter a number 4
 *  *  *  *
Enter a number 5
 *  *  *  *  *
```

Laboratory #9 – Applications II

1. Laboratory Objective

   The objective of this laboratory is to train students on how to integrate the knowledge gained in the C++ programming and apply it skillfully in programming solutions for real life problems II.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

   a. Apply the programming skills gained throughout the C++ Programming in solving real life problems.
   b. Gain appropriate experience in developing modular C++ Programs.

3. Laboratory #9 Introductory Concepts

4. Laboratory Problem Description

   Write a C++ program that calculates and displays the value of the following series: $\frac{1}{x}+\frac{1}{x^2}+\frac{1}{x^3}+\frac{1}{x^4}+\mathrm{L}\ +\frac{1}{x^n}$. You should use a `for` loop that counts up from 1 to n. The `sum` should be of type `double`.

   Sample Output

```
Enter the value of n: 3
Enter the value of x: 2
the value of the series is 0.875
```

5. Laboratory Instructions

   Please follow our hints for "Good Programming Practices" and avoid our hints for "Common Programming Mistakes" that were mentioned in previous labs.

6. **Laboratory Exercises**

   Write a C++ program that calculates and displays the value of the following series: $\frac{1}{x}+\frac{1}{x^3}+\frac{1}{x^5}+\frac{1}{x^7}+\mathrm{L}\ +\frac{1}{x^n}$. You should use a `for` loop that counts up from 1 to n. The `sum` should be of type `double`.

   Sample Output

```
Enter the value of n: 5
Enter the value of x: 2
the value of the series is 0.65625
```

Laboratory #10– Applications III

1. Laboratory Objective

   The objective of this laboratory is to train students on how to integrate the knowledge gained in the C++ programming and apply it skillfully in programming solutions for real life problems.

2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

   a. Apply the programming skills gained throughout the C++ Programming in solving real life problems.
   b. Gain appropriate experience in developing modular C++ Programs.

3. Laboratory #10 Introductory Concepts

4. Laboratory Problem Description

   Write a C++ program that creates a two-dimensional array with 5 rows and 5 columns. Fill the two-dimensional array with random numbers in the range 1 to 10, inclusive. Print the two-dimensional array. Then, find and display the maximum number in each row.

   Sample Output

```
2  8  5  1  10
5  9  9  3  5
6  6  2  8  2
2  6  3  8  7
2  5  3  4  3
row 0: max = 10
row 1: max = 9
row 2: max = 8
row 3: max = 8
row 4: max = 5
```

5. Laboratory Instructions

   Please follow our hints for "Good Programming Practices" and avoid our hints for "Common Programming Mistakes" that were mentioned in previous labs.

6. Laboratory #10 Exercises

Write a C++ program that input the elements of two 2 x 2 matrices. Then, it will output the sum of the two matrices.

Sample Output

```
input matrix 1
element (0 , 0): 2
element (0 , 1): 2
element (1 , 0): 2
element (1 , 1): 2
input matrix 2
element (0 , 0): 4
element (0 , 1): 4
element (1 , 0): 4
element (1 , 1): 4

the sum of the two matrices
6  6
6  6
```

Appendix A: Rules to follow by Computer Lab Users

- Loud conversations/discussion that disturbs other colleagues in the lab is prohibited.
- Audio CDs or applications with audio output may only be used with headphones with minimum volume such that it should not disturb others.
- All cell phones are to be turned off or set to silent while in the lab. If you receive a phone call, you should exit the lab before answering your cell phone.
- Do not bring food or beverages inside the lab.
- Any file saved on the computer hard drive will be deleted without notice. Students should save their work onto an external storage device such as USB drive or CD.
- Changing hardware and software configurations in the computer labs is prohibited. This includes modifications of the settings, modifications of system software, unplugging equipment, etc.
- Open labs are reserved for academic use only. Use of lab computers for other purposes, such as personal email, non-academic printing, instant messaging, playing games, and listening to music is not permitted.
- Please leave the computer bench ready for the next patron. Leave the monitor on the login screen, and do not forget to take all your personal belongings before leaving the lab. While leaving computer bench please push the chair inside the computer bench.
- Users are responsible for their own personal belongings and equipment. Do not leave anything in the Computer Lab unattended for any length of time. The Computer Labs staffs are not responsible for lost or stolen items.
- Users are not allowed to clear paper jams in the printer by themselves.
- Operate the lab equipments with care.
- After using white-board the user must clean for other user.

Thanks for your cooperation.

Information Science Department

Appendix B: Endorsement

LABORARTORY MANUAL FOR

ISC 115 Computing Foundations

#	Instructor name	Remarks	Signature	Date
1	Professor Mostafa Abd-El-Barr			
2	Professor Muhammad Sarfraz			
3	Dr. Mariam Al-Otaibi			
4	Eng. Aisha Al-Noori			