

**Programming Fundamentals**

**Fall 2019 Lab Manual**

**Submitted By:**

|  |  |
| --- | --- |
| Name |  |
| CMS ID |  |
| Department |  |
| Section |  |

# **Submitted To:** Mr. Muhammad Bilal Khan

**Lesson Set 2Data types, Variables, Variable Values & Console output**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Purpose** | | 1. To briefly introduce the C++ programming language 2. To show the use of memory in programming 3. To introduce variables and named constants 4. To introduce various data types:   a. Integer, Character, Floating point, Boolean and String  5. To introduce the assignment and **cout** statements | | | |
| **Procedure** | | 1. Students should read the Pre-lab Reading Assignment before coming to lab. 2. Students should complete the Pre-lab Writing Assignment before coming to lab. 3. In the lab, students should complete Labs 2.1 through 2.4 in sequence. Your 4. Instructor will give further instructions as to grading and completion of the lab. 5. Students should complete the set of lab tasks before the next lab and get them checked by their lab instructor. | | | |
| **Contents** | | **Pre-requisites** | **Completion Time** | **Page Number** | |
| Pre-lab Reading Assignment | | - | 20 min | 12 | |
| Pre-lab Writing Assignment | | Pre-lab Reading | 20 min | 15 | |
| **Lab 2** | | |  |  | |
| **Lab 2.1**  Working with the **cout** statement | | Pre-lab reading | 20 min | 16 | |
| **Lab 2.2**  Working with Variables | | Understanding Variables | 30 min | 16 | |
| **Lab 2.3**  Working with numeric data types | | Understanding Variables, their types and usage | 30 min | 17 | |
| **Lab 2.4**  Working with non-numeric data types | | Understanding Variables, their types and usage | 30 min | 17 | |
| **Lab 2.5**  Lab Tasks | | Understanding Variables, their types and usage | 30 min | 18 | |

## PRE-LAB READING ASSIGNMENT

|  |  |
| --- | --- |
| **The C++**  **Programming**  **Language** | Computer programming courses generally concentrate on program design that can be applied to any number of programming languages on the market. It is imperative, however, to apply that design to a particular language. This course uses C++, a popular object-oriented language, for that purpose. For now, we can think of a C++ program as consisting of two general divisions: header and main. The **header**, or **global** **section**, gives preliminary instructions to the compiler. It consists of comments that describe the purpose of the program, as well as information on which library routines will be used by the program. Given below is a basic c++ program. All c++ programs follow the following basic structure. |

1.

#include

iostream

<

>

2.

**using**

**namespace**

std;

3.

4.

/\*

These

are

5.

multi

line

comments

\*/

6.

7.

//

This

is

a

single

line

comment

8.

9.

**int**

main()

10.

{

11.

cout<<

"Our

First

Program"

;

12.

**return**

0

;

13.

}

|  |  |
| --- | --- |
|  | Everything in bold (everything above the int main() statement) is considered the header or global section. Everything in bold and italic (enclosed within /\* and \*/ or staring with //) are comments and Everything else is the main section. |
| **Preprocessor Directives** | The statements beginning with the **#** are called the preprocessor directives.The #include statement, indicates which library will be needed by the program.  #include <iostream> |
| **Libraries** | Recall from Lesson Set 1, that every program needs other modules attached so that it may execute properly. Libraries are the sort of other modules that are made part of your program by using #include preprocessor directive. IOSTREAM is a library that contains code for basic input or output related tasks. Your instructor will generally tell you which libraries are needed for each particular programming assignment. |
| **Comments** | Comments are included in every program to document what a program does and how it operates. These statements are ignored by the computer but are most valuable to the programmers who must update or fix the program. In C++ comments begin with // which is an indication to the compiler to ignore everything from the // to the end of the line. Comments can also cross line boundaries by beginning with /\* and ending with \*/. Notice that the first three |

lines of the previous program all begin with // and thus are comments. Those same lines could also have been written as the following:

***/\* These are multi line comments \*/***

***// This is a single line comment***

|  |  |
| --- | --- |
| **Main Function** | Every C++ program has a **main** function which indicates the start of the executable instructions. Every main must begin with a left brace { and end with a right brace }. The statements inside those braces will be explained as we progress through this lesson |
| **Memory** | Memory storage is the collection of locations where instructions and data that are used by the program are temporarily stored. Recall from Lesson Set 1 that a computer only understands a sequence of 1s and 0s. These are binary digits or **bits (Binary Digits)**. Eight of these brought together are called a **byte** which is the most common unit of storage.  **Analogy:** These chunks of memory can be thought of as hotel mailboxes at the registration desk. The size of each of those boxes indicates the type of mail that can be stored there. A very small mailbox can only hold notes or postcards. Larger mailboxes can hold letters, while even larger ones can hold packages. Each mailbox is identified by a number or name of an occupant. We have identified two very important attributes of these mailboxes: the name or number, which indicates the mailbox that is being referenced, and the size, which indicates what type of “data” can be placed there.  **Example:****Postcards Jim** is an indication that the mailbox called Jim can only hold postcards, while the statement **Packages Mary** indicates that the mailbox called Mary can hold large packages. Memory locations in a computer are identified by the same two attributes: data type and name. Much of programming is getting data to and from memory locations and thus it is imperative that the programmer tell the computer the name and data type of each memory location that he or she intends to use. In the sample program the statement **float radius** does just that, float is a data type that indicates what kind of data can be stored and radius is the name for that particular memory location. |
| **Variables and Constants** | The ability to change or not change the data stored can be a third attribute of these memory locations. Components of memory in which data values stored can change during the execution of the program are called **variables**. These usually should not be defined in the header or global section of the program. In our sample program, radius is defined in the main function. Components of memory in which data values stored are initialized once and never changed during the execution of the program are called **constants**. They are often defined in the global section and are preceded (in C++) by the word **const**. |
| **Identifiers in C++** | Identifiers are used to name variables, constants and many other components of a program. They consist exclusively of letters, digits and the underscore character. They cannot begin with a digit and cannot duplicate reserved words used in C++ such as **int** or **if**. All characters in C++ are case sensitive; thus, memory locations called simple, Simple, and SIMPLE are three distinct locations. It has become standard practice among programmers to make constants all uppercase and variables predominantly lowercase characters. |
| **Data types** | While doing programming in any programming language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. You may like to store information of various data types like character, wide character, integer, floating point, double floating point, boolean etc. Based on the data **type of** a variable, the operating system allocates memory and decides what can be stored in the reserved memory. C++ offer the programmer a rich assortment of built-in as well as user defined data types. Following table lists down five basic C++ data types |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Keyword** | **Size** | **Range** |
| Boolean | Bool | 1 Bit | True and False |
| Character | Char | 1 Byte | -127 to 127 or 0 to 255 |
| Integer | Int | 4 Bytes | -2147483648 to 2147483647 |
| Floating point | Float | 4 Bytes | +/- 3.4e +/- 38 7*digits* |
| Double | Double | 8 Butes | +/- 1.7e +/- 308 |

|  |  |
| --- | --- |
|  |  |
| **Integers** | Integers are real numbers that do not contain any fractional component. They take up less memory than numbers with fractional components.  The following statement defines integer variable in C++:  **int sum;** |
| **Float and Double** | C++ uses both **float** and **double** to indicate floating point numbers, with double using more memory than float. In computer science 3 = 3.0 is not a true statement. The number on the left is an integer and the number on the right is a real or floating point, number. Although mathematically the two are equal, the computer stores them as different data types. The following two statements define floating point variables in C++.  **float average; double nationaldebt;** |
| **Character** | Character data includes the letters of the alphabet (upper and lower cases), the digits 0–9 and special characters such as ! ? . , \*. All these symbols combined are called **alphanumeric**. Each character data is enclosed with single quotes to distinguish it from other data types. Thus '8' is different than 8. The first is a |

character while the second is an integer. The following statement defines a C++ character variable initialized to 'a'.

## char letter = 'a';

|  |  |
| --- | --- |
| **Boolean** | The Boolean data type, named after the mathematician George Boole, allows only two values: true or false, which are internally represented as 0 and nonzero, respectively. The following statement defines a Boolean variable initialized to false.  **bool found = false;** |
| **Assignment**  **Statements**                      **Output Statements** | These statements place values in memory locations. The left side of an assignment statement consists of one and only one variable. The right side consists of an **expression**. An expression can be any manipulation of **literal** numbers (actual numbers such as 7 or 38, etc.), or the contents of constants and/or variables, that will “boil down” to one value. That value is placed in the memory location of the variable on the left. C++ uses = as the separator between the left and right side of the assignment statement. Those new to programming often get this confused with equality; however = in C++ is not equality but rather the symbol to indicate assignment. The = in C++ is read as “is assigned the value of”.  **Example:**  **int total; total = 10;**  Here int is the data type and total is the name of the variable whereas 10 is a literal or a value that is placed in the variable  These instructions send information from the computer to the outside world. This information may be sent to the screen or to some file. In C++ the **cout <<** statement sends information to the screen. The #include <iostream> directive must be in the header for cout to be used.  **cout << total;**  The above statement sends whatever value is stored in the variable total to the screen. |
| **Other Statements** | C++ uses the semicolon as a statement terminator. We can output literal strings  (such as “Hello”) by enclosing them in double quotes. The << operator acts as a separator for multiple outputs. |

## cout << "The value of total is " << total << endl;

The endl statement causes the cursor to be moved to the beginning of the next line.

## Pre lab writing assignment

|  |  |
| --- | --- |
| **Fill in the blanks** | 1. A \_\_\_\_Constant\_\_\_ is a memory location whose value cannot change during the execution of the program. 2. \_\_\_\_Integer\_\_\_\_ is a data type that only holds numbers with no fractional component. 3. \_\_\_Float and Double\_\_\_ is a data type that holds numbers with fractional components. 4. The \_\_|Global\_\_\_ section gives preliminary instructions to the compiler 5. \_\_\_Comments\_\_\_ are included in every program to document what a program does and how it operates. |

## Lab Task 02

**Lab 2.1** • Once you have opened the visual studio copy the following code in the

code editor, try to run the code and observe the output. Write down the

**[[1]](#footnote-1) cout** output that appears on the screen in the space given below **Statement**

1.

#include

iostream

>

<

2.

**using**

**namespace**

std;

3.

4.

**int**

main

()

5.

{

6.

**int**

age;

7.

age

=

10

;

8.

cout

<<

"The

age

is

"

<<

age;

9.

**return**

0

;

10.

}

1.

#include

iostream

<

>

2.

**using**

**namespace**

std;

3.

4.

**int**

main

()

5.

{

6.

cout

<<

"Hello

World

"

<<

endl;

7.

**return**

0

;

8.

}

|  |
| --- |
| Hellow World |

**Lab 2.2** • Create a new project called variablesprogram.cpp, copy the following

code it and try to run the program. Observe the output that appears and

|  |  |  |
| --- | --- | --- |
| **Lab 2.3**  **Working with numeric data types** | • | Create a new project called numericprogram.cpp, copy the following code it and try to run the program. Observe the output that appears and write it in the box given below |

1. #include <iostream> 2. **using** **namespace** std;

3.

1. **int** main ()
2. {
3. **int** number;
4. **float** weight;
5. age = 10;
6. weight = 5.5f;
7. cout << " The age is " << age;
8. cout << " The weight is "<< weight;
9. **return** 0;
10. }

|  |
| --- |
| **The age is 10 The weight is 5.5F** |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Lab 2.4**  **Working with nonnumeric data types** |  | • | Create a new project called nonnumericprogram.cpp, copy the following code it and try to run the program. Observe the output that appears and write it in the box given below |

|  |  |  |
| --- | --- | --- |
| 1. | #include <iostream.h> | |
| 2. | **using** **namespace** std; | |
| 3. | **void** main() | |
| 4. | { | |
| 5. | **char** c = 'A'; | |
| 6. | cout << c << '\n'; | |
| 7. | } | |
| A | | |

### Lab 2.5 Lab Tasks

1. Write a program in C++ to print your name on the screen.

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  cout<<”IQRA MENGAL”<<endl;  } |

1. Write a program in C++ to print the following patterns on the screen.

* + 1. \*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int rows, i, j, space;  cout << "Enter number of rows: ";  cin >> rows;  for(i = 1; i <= rows; i++)  {  //for loop for displaying space  for(space = i; space < rows; space++)  {  cout << " ";  }  //for loop to display star equal to row number  for(j = 1; j <= (2 \* i - 1); j++)  {  cout << "\*";  }  cout << "\n";  }  return 0;  } |

* + 1. \*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int rows, i, j, space;  cout << "Enter number of rows: ";  cin >> rows;  for(i = 1; i <= rows; i++)  {  //for loop for displaying space  for(space = i; space < rows; space++)  {  cout << " ";  }  //for loop to display star equal to row number  for(j = 1; j <= (2 \* i - 1); j++)  {  cout << "\*";  }  cout << "\n";  }  return 0;  } |

1. Write a program that takes the age of user in Float format and prints 1/3rd of the current age of user in Double format?

Steps:

* 1. Create a variable of Float data type.
  2. Assign a value to the created variable.
  3. Create another variable of Double data type.
  4. Divide the variable of Float data type by 3 and assign it to the variable created in Step 3.
  5. Print the information of both variables using cout.

|  |
| --- |
| #include <iostream>  using namespace std;  int main ()  {    float age;  double current\_age;  cout<<"Enter Age : "<<endl;  cin>>age;  current\_age=age/3;  cout<<" The Age Entered Is "<<age<<endl;  cout<<"The 1/3rd of the current age of user in Double format is "<<current\_age;      } |

1. write it in the box given below

   |  |
   | --- |
   | The age is 10 |

   Variables

   [↑](#footnote-ref-1)