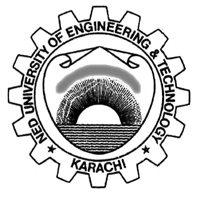
**PRACTICAL WORK BOOK**

**Computer and Programming (EE-163)**

**For**

**FE (ELECTRICAL)**

|  |
| --- |
| Name: MUHAMMAD MUNEEB WASEEM |
| Roll Number: EE-205 |
| Class: FIRST YEAR Semester: SECOND(2nd) |
| Batch: 2018 |
| Department : ELECTRICAL ENGINEERING |



**Department of Electrical Engineering  
NED University of Engineering & Technology, Karachi**

**CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab.**  **No.** | **Title of Experiments** | **Dated** | **Remarks** |
| 1 | Getting Started – Familiarization with Environment |  |  |
| 2 | C++ Building Blocks |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| **Lab Session 01** |

# **Objective:**

# **Getting Started – Familiarization with Environment.**

In this lab session, we shall cover the following objectives

* How to install Code::Blocks IDE on computer
* Use Code::Blocks IDE and built in MinGW GCC Compiler to run our first program
* Explore Command Prompt (cmd)
* Run our first program via cmd
* Run an existing program (GuessNumber.exe) via cmd
  1. **Installing Code::Blocks Integrated Development Environment (IDE)**

C++ (pronounced cee plus plus) is a compiled language. In order to get started, two requirements are essential. First is the compiler and second is text editor (for typing the program). These requirements often come under a single packaged software application termed as Integrated Development Environment (IDE). For the lab sessions of this course we shall be using an open source and free of cost IDE called **Code::Blocks**. Getting Code::Blocks is just a matter of few clicks (provided you have an internet connection). In order to download the IDE follow these steps

1. Access [www.codeblocks.org/downloads](http://www.codeblocks.org/downloads) from your favorite web browser.
2. Click ***Download the binary release***
3. Download the Code::Blocks with Mingw setup file, at the time of writing this text codeblocks-16.01mingw-setup.exe was available.
4. You are ready to go now.

Note: These instructions are for Windows users. If you are running any other operating system then download the version for your operating system.

*If you don’t have internet access you can get a copy of the binary release from the Computer Lab. For now, it’s only available for Windows users.*

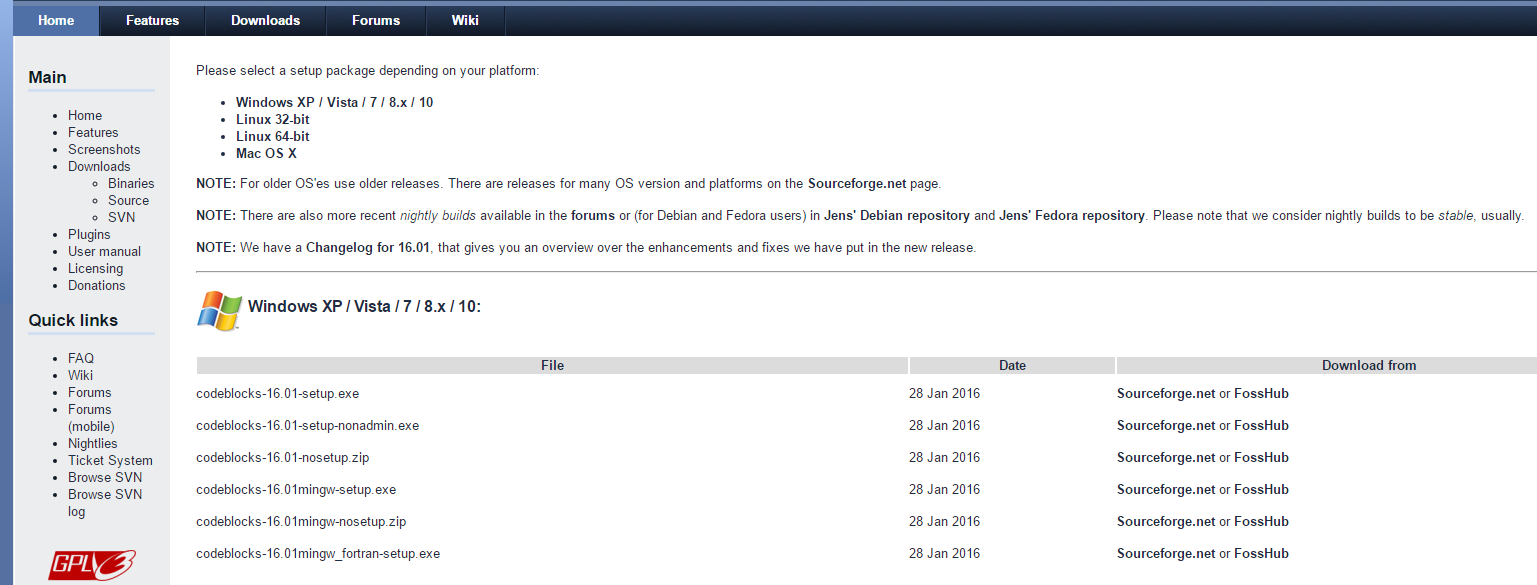


Figure 1 Screenshot of Downloads Page for Binary Release

Installation process is simple. Run the executable file you just downloaded (or acquired from Computer Lab). The installation Wizard will guide you through the whole process.

Once you run the setup file, the Wizard will get started.

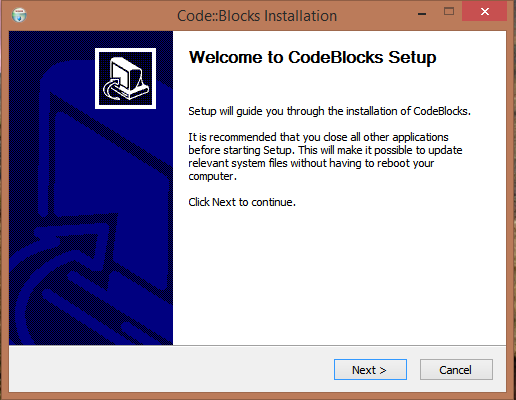


Figure 2 Step 1 of Installation Process Wizard Guide

Click Next to continue.

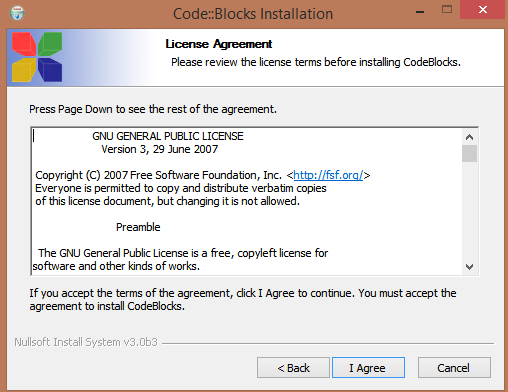


Figure 3 Step 2 of installation process License Agreement

You must agree with license terms to install and use Code::Blocks (read the terms provided and click I Agree). Once you are agreed with the terms, the installation wizard will now prompt to choose the components to install, check all components and click Next.

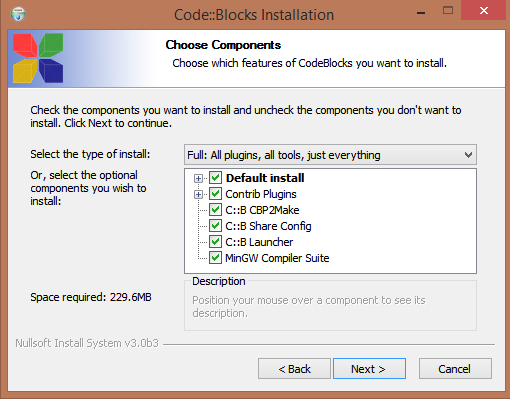


Figure 4 Step 3 of installation process, Components to install

Now select the hard disk location to install the Code::Blocks (using default is recommended)

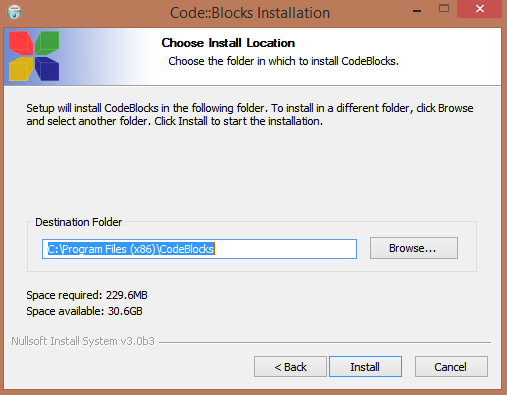
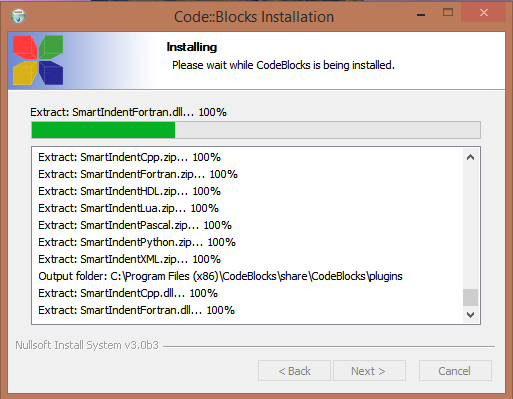


Figure 5 Step 4 of installation process, choose destination

Once you click the install button the installation will take place. Upon successful installation you will get

the message.

Figure 6 Installation in process

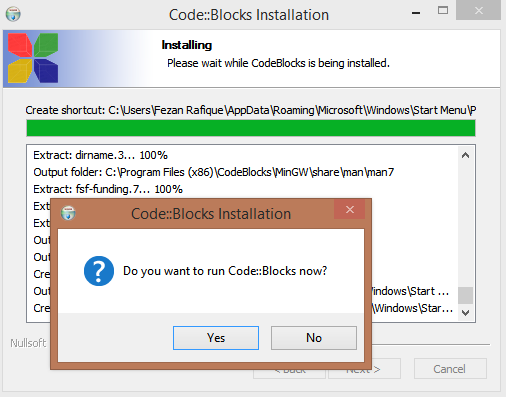


Figure 7 Installation successful

Once the installation process completed, click Finish button

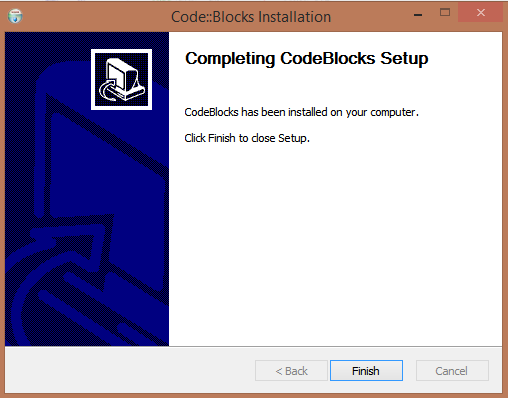


Figure 8 Installation process completed

* 1. **Running the First Program**

Once, the IDE is installed successfully we are now ready to develop our first C++ program. Follow the following steps

* Open Code::Blocks
* Create a new empty file (shortcut Ctrl + Shift + N )
* Save the file as lab\_01\_code\_01.cpp
* Beware about the format .cpp

**lab01\_code\_01.cpp**

#include<iostream>

using namespace std;

int main(void)

{

cout<<"Hello World";

return 0;

}

* Type the code as shown (don’t worry if you don’t understand it for now)
* After typing the code Go to BUILD>>BUILD and RUN (shortcut F9)
* If your program was successfully written, it will be executed otherwise you will get an error

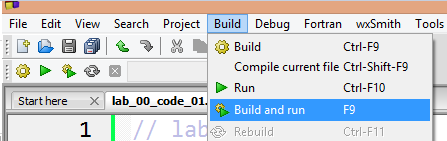


Figure 9 Step to Build Code

Screen Clipping

Figure 10 Console Log for Successful Build

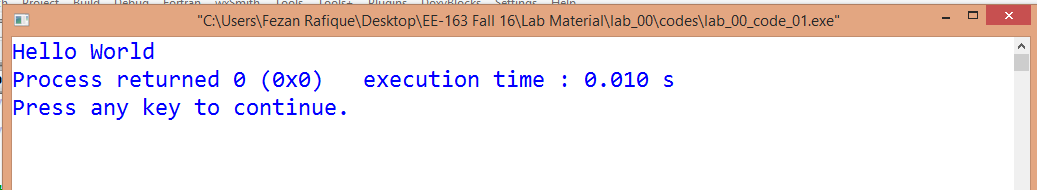


Figure 11 Output for lab\_01\_code\_01.cpp

* 1. **Exploring Command Prompt**

The target of our first program and all the other programs in this course is Console (command prompt or terminal). It is therefore necessary to have a brief introduction of command prompt.

To start command prompt, type “**cmd**” (without quotes) in Run.

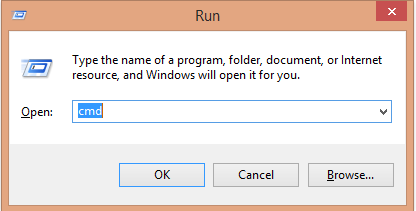


Figure 12 Run command for command prompt

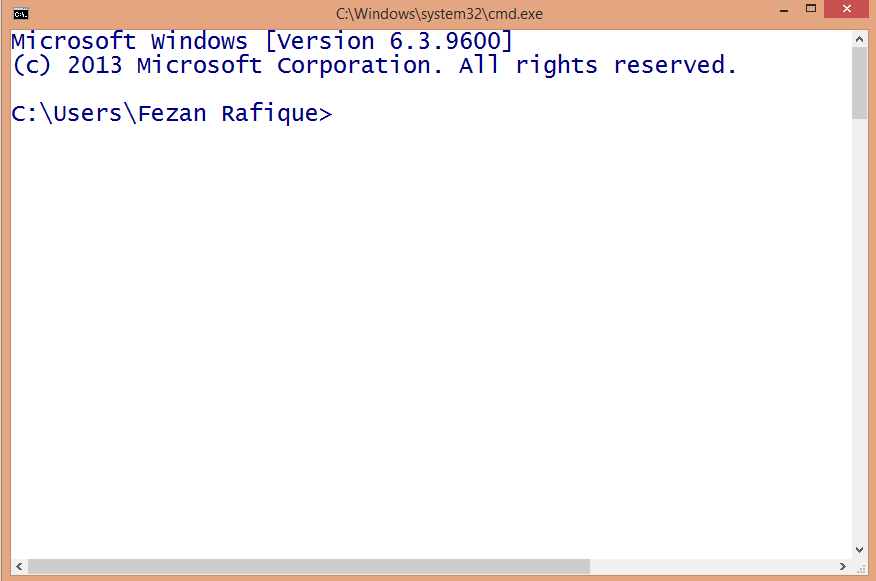
This will open the command prompt window. 

Figure 13 Command Prompt

To navigate through the directories, one can use **cd** command. A sample is shown in figure.

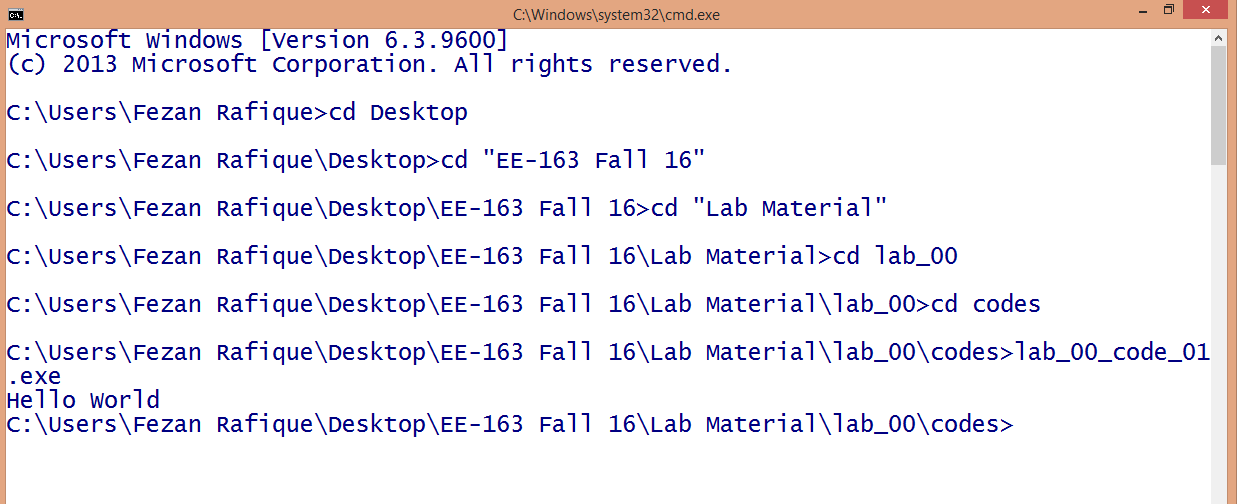


Figure 14 Navigating directories

There are many useful commands for command prompt, following links are helpful to get started.

* http://www.digitalcitizen.life/command-prompt-how-use-basic-commands
* http://www.computerhope.com/overview.htm
  1. **Run GuessNumber.exe**

As part of cmd exercise we shall now run an already developed program called **GuessNumber.exe** through cmd. This file is provided in the folder for Lab01

* GuessNumber.exe is already written program, the program asks the user to guess a number (which is in computer’s mind)
* The user will respond by typing and can do so, until correct number is guessed
* In the meanwhile for any wrong guess computer will give a hint
* Let’s try it

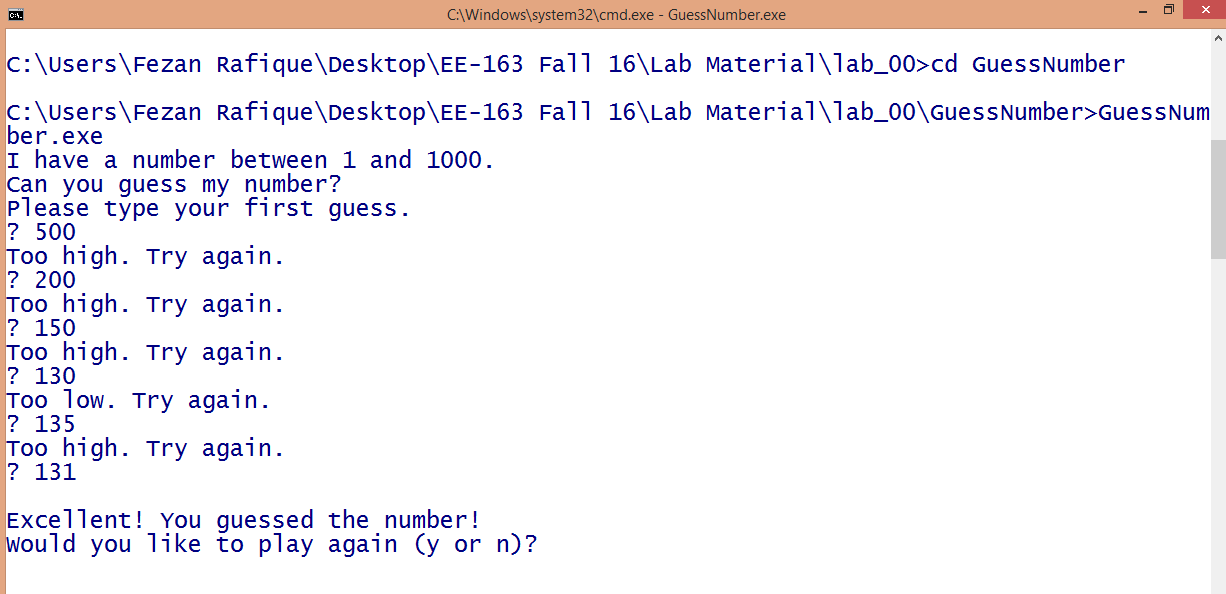


Figure 15 Running GuessNumber.exe

# Exercise

**Task 1:**

Write a program to print text in following pattern,

Hello World

Hello World

Hello World

#include <iostream>

using namespace std;

int main()

{

cout<<"Hello World\n\tHello World\nHello World";

}

|  |
| --- |
| **Lab Session 2** |

# **Objective**:

# C++ Building Blocks

In this lab session, we shall cover the following objectives

* Basic data types in C++
* Declaring and using variables
* Comments in a C++ Program
* Printing variable values with cout
* Interactive computing with cin
* Escape sequences

# **2.1 Basic Data Types in C++**

# Fundamental to any computer program is the data associated with its use. Based on the nature of data it can be classified into various categories. Data types are important to understand, they define proper use of an identifier and expression. In C++ data types can be categorized as following.

Figure Basic data types in C++

*Numeric*: This type contains the numbers including integers and floating point values. Following are the example of numeric data

* 100
* 895
* -237
* 6.022140857 × 10 ^ 23
* 6.62607004 × 10 -34
* -1.60217662 × 10-19

*Character*: Character data includes the alpha numeric characters and special symbols (enclosed in single quotes). Following are the examples

* ‘a’
* ‘F’
* ‘@’
* ‘%’
* ‘^’

*Strings*: Strings include all the text values (enclosed in double quotes). Following are the examples

* “Finland”
* “NED University”
* “PO Box No 341”
* “all along the watch tower”

*Boolean*: Boolean includes true and false values.

Following C++ statements show the possible use of these data types

cout<<100;

cout<<‘~’;

cout<<true;

cout<<“Mixing the stream ”<<200<<‘#’<<true<<“ ”<<false;

**2.2 Declaring and Using Variables**

* Variables are named objects with a specific type
* Variables can be used to store data of a certain type which can later be used, processed and/or updated in the program
* A variable must be declared using appropriate keyword
* There are some rules with variable naming

The following table shows the keyword and memory requirement of several data types

|  |  |  |
| --- | --- | --- |
| **Type** | **Keyword** | **Memory** |
| **Boolean** | bool | 1 Byte |
| **Character** | char | 1 Byte |
| **Integer** | int | 4 Bytes |
| **Floating point** | float | 4 Bytes |
| **Double floating point** | double | 8 Bytes |
| **String** | string | ? |

lab\_02\_code\_01.cpp

Following code can be used to check the memory requirements of various data types

#include<iostream>

using namespace std;

// sizeof() function calculates the Bytes

int main(void)

{

cout<<"Integer Bytes="<<sizeof(int);

cout<<"\nDouble Bytes="<<sizeof(double);

cout<<"\nCharacter Bytes="<<sizeof(char);

cout<<"\nBoolean Bytes="<<sizeof(bool);

return 0;

}

Identify the data types for the following items

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Type | Item | Type |
| TRUE |  | @ |  |
| 127 |  | 192.12 |  |
| Pakistan |  |  |  |

Variable Naming Rules: Following rules must be taken care while assigning a name to any variable.

* Variable name must start with a letter or \_ (underscore)
* May contains letter, numbers and the underscore character only
* Uppercase and lower case are distinct
* Name should not be a reserved keyword

Good Examples

salary, new\_name, myValue

Bad Examples

3name, my name, my-val, class, struct, while

A variable can be assigned a value with the assignment operator “=” . (Discussion about associativity will be the part of Lab03)

The following codes will be helpful to understand the use and role of variables in a C++ program

lab02\_code\_02.cpp

#include<iostream>

using namespace std;

int main(void)

{

int age; // declaring int variable

string name;// declaring string variable

float height\_in\_cms, weight\_in\_kg; // 2 float variables

age = 19; // now assigning values to variables

name = "Ahmed Khan";

height\_in\_cms = 123.8;

weight\_in\_kg = 58.7;

cout<<"Name:"<<name<<"\t Age:"<<age<<endl;

cout<<"Height(cm):"<<height\_in\_cms<<"\t Weight(kg):"<<weight\_in\_kg;

return 0;

}

lab02\_code\_03.cpp

#include<iostream>

using namespace std;

int main(void)

{

int Roll\_No = 123, salary = 40000;

float CGPA = 3.2;

double pi = 3.1214, x = 0.012, y;

string enrolment\_no = "ned/0145/14-15",name;

char section = 'D';

bool logical = 1;

cout<< "My Roll No:" <<Roll\_No<<"\t Pi="<<CGPA;

cout<< endl<<"Value of y is:"<<y<<endl;

cout<< "Name:"<<name<< endl;

cout << "Enrolment:"<< enrolment\_no;

return 0; }

**2.3 Comments in a C++ Program**

Program comments are explanatory statements that you can include in the C++ code that you write and helps anyone reading it's source code. All programming languages allow for some form of comments. C++ supports single-line and multi-line comments. All characters available inside any comment are ignored by C++ compiler.

C++ comments start with /\* and end with \*/. For example:

/\* This is a comment \*/

/\* C++ comments can also

\* span multiple lines

\*/

A comment can also start with //, extending to the end of the line. For example:

#include <iostream>

using namespace std;

main() {

cout << "Hello World"; // prints Hello World

return 0;

}

When the above code is compiled, it will ignore // prints Hello World and final executable will produce the following result:

Screen Clipping

Within a /\* and \*/ comment, // characters have no special meaning. Within a // comment, /\* and \*/ have no special meaning. Thus, you can "nest" one kind of comment within the other kind. For example:

/\* Comment out printing of Hello World:

cout << "Hello World"; // prints Hello World

\*/

**2.4 Idea of Interactive Computing**

In the above programs the value was directly assigned to the variable via assignment operator. This was done by the programmer. If it is needed to take input from the user and assign the user value to a particular variable. This is called interactive computing. C++ provides means to do so. One can use stream insertion via cin to assign value to a variable. This can be done like following

int value;

cout<<“Please enter the value ”;

cin>>value;

The following code further illustrates the idea of interactive computing

lab\_02\_code\_04.cpp

#include<iostream>

using namespace std;

int main(void)

{ // Starting braces of main

//\*\*\*Variable Declaration\*\*\*

string name, year, department ;

char section;

int roll\_no;

float cgpa;

//\*\*\*\*Taking user input\*\*\*\*

cout<<"Enter your name:";

cin>>name;

cout<<"Enter your Roll No.:";

cin>>roll\_no;

cout<<"Enter your department:";

cin>>department;

cout<<"Enter year of study:";

cin>>year;

cout<<"Enter your section:";

cin>>section;

cout<<"What is your CGPA?";

cin>>cgpa;

cout<<endl<<endl;

//\*\*\*\*\*Printing Output\*\*\*\*\*

cout<<"\t My Profile"<<endl;

cout<<"Name:"<<name<<"\tRoll No:"

<<roll\_no<<endl<<"Section:"

<<section<<"\tYear:"<<year<<

endl<<"Department:"<<

department<<"\tCGPA:"<<cgpa;

return 0;

}

**2.4 Escape Sequences**

You must have observed some difference in the last code, e.g. using \t in cout statements. This is called escape sequence. Escape sequences are used to represent certain special characters within string literals and character literals. Following escape sequences are commonly used in C++.

|  |  |
| --- | --- |
| **Sequence** | **Purpose** |
| \n | Next line |
| \r | Carriage return |
| \t | Horizontal tab |
| \b | Backspace |
| \a | Alert (beep) |
| \\ | Print \ |
| \’ | Print ’ |
| \” | Print “ |

Taking help from your textbook and online resources, try to figure out the purpose of these escape sequences and explain with the help of an example program.

# Exercise

**Task 1:**

How to insert single line and multiline comments in a C++ program.

MULTI LINE COMMENTS:

1. Starts with /\* and ends with \*/.

2. All Words and Statements written between /\* and \*/ are ignored.

3. Comment ends when \*/ Occurs.

4. E.g. /\* this is Multiline Comment \*/

SINGLE LINE COMMENTS:

1. Starts with //.

2. Statements after the symbol // up to the end of line are ignored.

3. Comment Ends whenever ENTER is Pressed and New Line Starts.

4. E.g. // Single line Comment.

**Task 2:**

Variable Declarations can appear almost anywhere in the body of C++ function (T/F).

If true, then discuss the situation in which variable declaration must be done prior to some specific task. Support you answer by giving example.

We declare a variable of primitive types (int, float, char, Boolean, long), or complex types (string, vector, YourFooClass), and give it a name. The compiler will know how much memory it has to allocate for the given type. It is also done when output (result) is required over some given input.

//EXAMPLE:ADDITION OF TWO NUMBERS

#include <iostream>

using namespace std;

int main()

{

int a,b,sum;

cout<<"Input two numbers:";

cin>>a>>b;

sum=a+b;

cout<<"the sum of "<<a<<" and "<<b<<" is "<<sum;

}

**Task 3:**

Calculate the maximum and minimum number that can be accommodated by *int* data type (calculate range).

Integer=4 bytes=32bits

N=32

For sign integer:

Min = (-1)\*2^ (N-1)

Min = (-1)\*2^ (31)

Min = -128

Max = 2^ (N-1)-1

Max = 2^ (31)-1

Max = 127

Range = -128 to 127

For un sign integer:

Min=0

Max=2^ (N)-1

Max=2^ (32)-1

Max= 4294967295

Range= 0 to 4294967295

For long integer:

Integer=4 bytes=32bits

N=32

Min = (-1)\*2^ (N-1)

Min = (-1)\*2^ (31)

Min = -128

Max = 2^ (N-1)-1

Max = 2^ (31)-1

Max = 127

Range = -128 to 127

For un sign long integer:

Min=0

Max=2^ (N)-1

Max=2^ (32)-1

Max= 4294967295

Range= 0 to 4294967295

For short integer:

Short integer=2 bytes=16 bits

Min = (-1)\*2^ (N-1)

Min = -32768

Max=2^ (N-1)-1

Max= 32767

Range= -32768 to 32767

For un sign short integer:

Min=0

Max= 2^ (N)-1

Max= 65535

Range= 0 to 65535

**Task 4:**

What do you mean by *Variable Declaration* and *Variable Definition* in C/C++?

DECLARATION OF VARIABLE: Declaration of a variable in C means that the variable is only declared and allocated a block of memory but still has no value.

DEFINITION OF VARIABLE: Definition of a variable means to assign or initialize it with some specific value.

For example:

int a; //declaration

int i = 1; //definition

**Task 5:**

Check the output of the following *cout* functions and write your comments.

1. cout << “I am a computer geek, \rits a \blie.”
2. cout <<"a"<<"\t"<<"b"<<"\t"<<"c"<<endl;

1.

#include<iostream>

using namespace std;

int main()

{

cout <<"I am a computer geek, \rits a \blie.";

return 0;

}

OUTPUT: its alie.mputer geek

2.

#include <iostream>

using namespace std;

int main()

{

cout<<"a"<<"\t"<<"b"<<"\t"<<"c"<<endl;

return 0;

}

OUTPUT: a b c

**Task 6:**

Temperature can be converted from Centigrade to Fahrenheit using following formula.

**F = (9.0/5.0)\*C + 32.0**

Write a program that ask user to input the temperature in degree Centigrade and calculates and displays the equivalent Fahrenheit value. Try the program with two possible data types, integer and float. Discuss the difference in result for the two programs.

//CONVERSION OF TEMPERATURE FROM CELSIUS TO FAHRENHEIT

//FLOAT (DATA TYPE)

#include<iostream>

using namespace std;

int main()

{

float c,f;

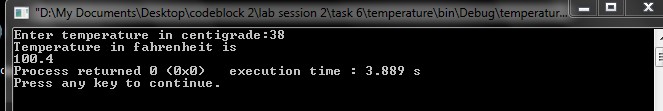
cout<<"Enter temperature in centigrade:";

cin>>c;

f=(9.0/5.0)\*c+32.0;

cout<<"Temperature in fahrenheit is"<<"\n"<<f;

}



//INT (DATA TYPE)

#include<iostream>

using namespace std;

int main()

{

int c,f;

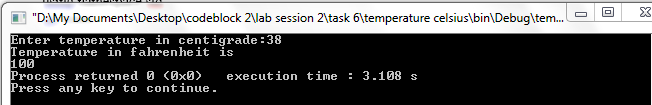
cout<<"Enter temperature in centigrade:";

cin>>c;

f=(9.0/5.0)\*c+32.0;

cout<<"Temperature in fahrenheit is"<<"\n"<<f;

}



With int data type, input and output will be in whole no (integer) whereas float data type is used for values with decimal points.