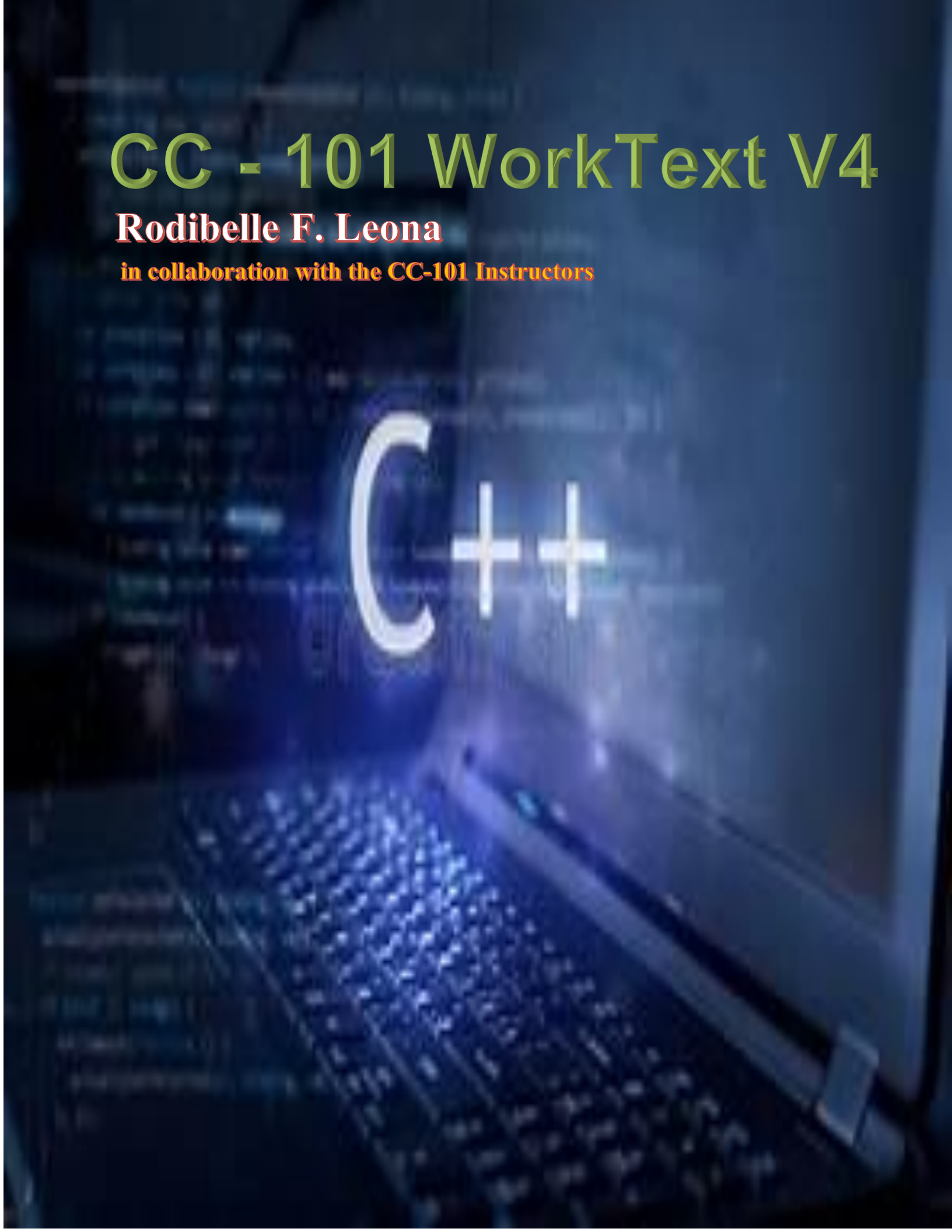
CC - 101 WorkText V4

Rodibelle F. Leona

in collaboration with the CC-101 Instructors



PREFACE

This worktext entitled “CC-101 WorkText” is a collaborative work of the instructors from the different campuses of Nueva Ecija University of Science and Technology who will be handling the subject “Computer Programming 1, Fundamentals” under the degree, Bachelor of Science in Information Technology which is offered every first semester of the academic year. The authors devised a way on how to present the topics of the subject in the simplest ways for the students to easily comprehend the lessons without agonizing over technical jargons related to computer programming.

The module is designed to introduce the basic programming concepts as well as to establish the foundation of programming through the use of the C++ language. It also includes discussions of terminologies, dynamics of flowcharting, and sample codes that can help the students learn and establish better appreciation of computer programming. Both objective and coding exercises are provided at the end of each chapter which can serve as a review and assessment of the knowledge gained.

The module is designed in such a way that it can be used in the different modalities of teaching-learning experience in the new normal. Most importantly, this material serves as a bridge to the teacher and students in achieving quality instruction and effective learning despite the crisis that we are experiencing nowadays.

Activity: What Do You Already Know About?

Pre-Assessment Test

Welcome to **CC101 – Computer Programming, Fundamentals**! Let's start with a little get-to-know-you quiz to assess your background in programming. Please answer the questions honestly by marking your responses with a check.

Do You Already Know...	***Yes, I do!***	***Maybe but unsure***	***I have no idea***
...what programming is?			
... how to draw flowcharts?			
... how to write codes in C++?			
...what variables are?			
...the different control structures?			
... if, if-else, nested if-else, and switch statements?			

TABLE OF CONTENTS

UNIT 1. PROGRAMMING CONCEPTS

Learning Objectives

At the end of the unit, I am able to:

1. *define software and hardware;*
2. *identify types of programmers;*
3. *explain the steps of problem solving;*
4. *identify the elements of programming; and*
5. *classify the programming paradigms.*

INTRODUCTION

To most individuals, ***computers*** may seem intelligent machines because they can do tasks efficiently and easily, but in actuality, they are just machines which will not do anything unless the tasks are specified by the users using programs that are executed into them.

A ***software*** is a collection of codes that instructs the computer on all the processes that it has to perform. Another term for software is ***computer program***. For complicated or large programs, say, the NEUST enrolment system, the program is consists of two sets of codes, the first are codes that instruct the computer and the second set is consists of codes which contain data. The lists of instructions are called ***program codes*** and the process of typing these codes into a computer is called ***program coding*** or simply ***coding***.

The people who write program codes are called ***programmers***. Programmers are classified as:

1. ***SYSTEM PROGRAMMERS***

   These are the people who create and maintain programs which are involved in the computer's basic operating functions like operating systems, utilities and device drivers.

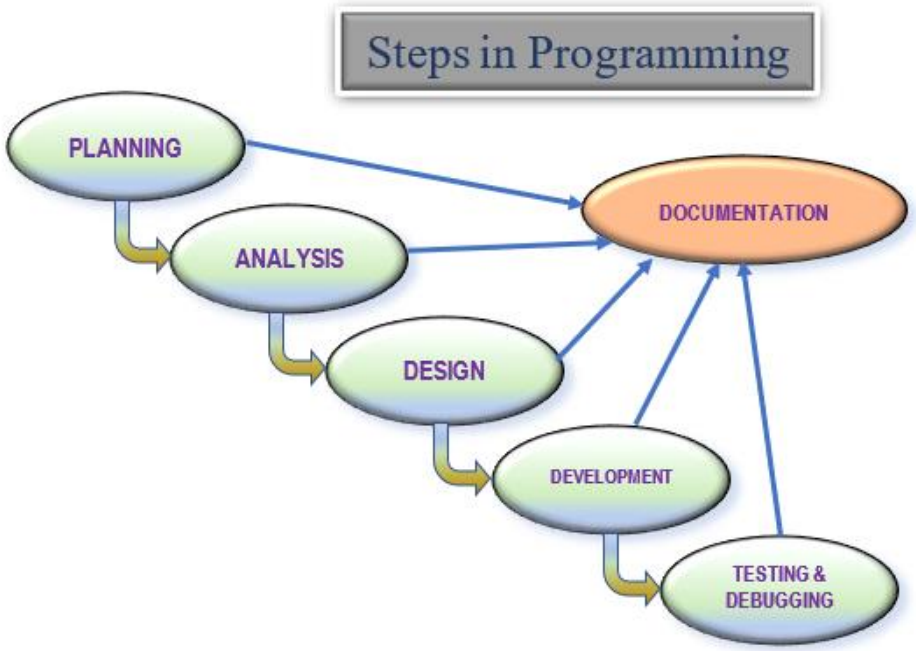
2. ***APPLICATION PROGRAMMERS***

   They are the ones who create and maintain programs that have specific functions such as ***Point Of Sale (POS)*** programs that calculate and generate the payroll report of a certain company, and others.

The physical components of a computer system is called ***hardware.*** It may be consisted of the CPU, the keyboard, the LCD screen and other peripherals like mouse, printer, webcam, etc. These are the tangible parts of the computer system which you can touch.

PROBLEM SOLVING

Just like when writing a term paper, the first step in creating an effective program is proper thought and preparation. Planning for the final output is very important so as to have the solution that you are picturing in your mind for a particular problem.



1. PLANNING

The planning stage revolves around the determination of the requirements, then setting the processes by which goals are to be achieved.

2. ANALYSIS

In the analysis stage, you must determine the precise objectives that you want to achieve to create a solution to the presented problem. The material that the program generates is called ***output***. A ***file***, which is stored by name, is a collection of information. A file may contain data, a program or some other document.

After deciding what the output will be, you should determine the information that is needed to achieve your objectives. These data that the program requires to produce the desired results are called ***input***.

Next will be to decide what will be the major steps required for processing the input to produce the output. A ***process*** is a mechanism for converting input to output.

3. DESIGN

After analyzing the problem, we use this analysis to design a solution. You can use modeling tools to guide you in designing a solution to the problem. Examples of such tools are flowcharts and algorithms.

4. DEVELOPMENT

In this step, you translate the design into a program. Writing computer instructions using any of the available programming languages is called ***CODING.*** These computer instructions or simply called statements are termed as ***CODES***.

5. TESTING AND DEBUGGING

Testing the problem's solution on the computer overlaps with the design and development phases. Testing is done so that all the computer errors or **BUGS** will be eliminated or corrected and the process of correcting errors is referred to as ***DEBUGGING***.

DOCUMENTATION

Documentation means that you have to create materials that will generally describe all the things you did in the whole process of developing a project or a program. All the steps in programming require documenting for the purpose of reference.

ELEMENTS OF PROGRAMMING

Almost all the programs are designed to solve problems. The design team is presented with the problem first to determine the requirements of the problem at hand. Using these requirements, they will consider several methods on how to reach the final output, utilizing the following elements of programming.

1. ***INPUT***

   The necessary information is collected at the start of programming. This means getting the ***INPUT*** into the program. The data you type, the click of the mouse, sources coming from compact disks or flashdrives are input values.

2. ***DATA***

All inputs are called ***data*** since nothing has been done yet to them. In other words, data are raw values which are not yet subjected into any manipulation.

3. ***OPERATIONS***

The data are then applied with the correct ***OPERATIONS*** to manipulate it. The ***OPERATIONS*** can be

a. Assigning values to constants and variables;
b. Subjecting values to different mathematical operations using the plus (+), minus (-), asterisk (*), slash (/) and modulo (%); and
c. Comparing values using relational operators and logical operators like OR (||), NOT (!), AND (&&), etc.

4. ***OUTPUT***

The manipulated data will be extracted from the program and sent back to the user. The extracted data is referred to as ***output***.

5. ***CONDITIONAL EXECUTION***

Data can also be manipulated using ***conditional execution*** or ***branching***. There may be codes that require comparison first before execution.

6. ***LOOPS***

Another way to manipulate data is by using looping. ***Loops*** execute a set of instructions until the condition inside the loop statement evaluates to false.

7. ***SUBROUTINES***

Data are sometimes manipulated by breaking them into small pieces and are executed at different locations in the program. These are called ***subroutines*** which are methods or sets of instructions that are executed by the program by calling their specific names throughout the program.

F. PROGRAMMING PARADIGMS

Programming paradigms are the styles of programming that a programmer uses to come up with the output out of a project or problem.

1. ***PROCEDURAL***

Procedural paradigm is running the codes line by line without skipping any of the statements, the order of which is which statement comes first, which is second and so on and so forth.

2. ***MODULAR***

   Modular paradigm is when you turn a huge program into small ones with simpler codes.

3. ***DATA ABSTRACTION***

   The data abstraction paradigm is hiding the attributes of the data making it appear more compound than more primitive.

4. ***OBJECT-ORIENTED PROGRAMMING***

   A programming methodology that represents concepts as "***objects***" that have specific attributes and methods.

Seat No.: _____ **Rating: _____**

Name: _____ **Date: _____**

Seatwork No. 1

ENCIRCLE THE LETTER OF THE CORRECT ANSWER.

1. It is a set of instructions in a computer.
   a. program c. hardware
   b. computer d. programmer
2. It is the data that the program requires to produce the desired results.
   a. output c. input
   b. process d. storage
3. It is the process of typing codes into a computer.
   a. testing c. design
   b. analysis d. coding
4. They are the individuals who write program codes.
   a. designers c. users
   b. librarians d. programmers
5. It is the material that the program generates.
   a. output c. input
   b. process d. storage
6. It refers to the computer itself and other equipment such as keyboard, mouse, monitor, central processing unit (CPU) and printer.
   a. program c. hardware
   b. computer d. programmer
7. It is a collection of information.
   a. file c. data
   b. computer d. requirements
8. It is a mechanism for converting input to output.
   a. output c. input
   b. process d. storage
9. It is another name for EXECUTES (as in executing a program).
   a. runs c. codes
   b. debugs d. tests
10. Complete name of your instructor.

_____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 2

ENUMERATE THE FOLLOWING:

A. PROGRAMMING PARADIGMS

1.
2.
3.
4.

B. ELEMENTS OF PROGRAMMING

5.
6.
7.
8.
9.
10.
11.

C. TYPES OF PROGRAMMERS

12.
13.

D. STEPS IN PROGRAMMING

14.
15.
16.
17.
18.

E. TYPES OF APPLICATION PROGRAMMERS

19.
20.
21.
22.

UNIT 2. NUMBER SYSTEMS

Learning Objectives

At the end of the unit, I am able to:

1. *develop knowledge about th different number systems; and*
2. *convert values into different number systems.*

A ***number system*** is a particular way on how to represent numbers. A number can be consisted of a digit or a series of digits. When you say digit, it is only ***one symbol*** that represent a number. The number ***143*** is consists of ***3*** digits, 1, 4 and 3. The value of each digit position is called ***weight***. ***Decimal weight*** represents the value of each digit position of a decimal number such as UNITS (10^0), TENS (10^1), HUNDREDS (10^2), THOUSANDS (10^3), etc. ***Binary weights*** are enumerated as UNITS (2^0), TWOS (2^1), FOURS (2^2), EIGHTS (2^3), and so on. ***HEXADECIMAL WEIGHTS*** are enumerated as UNITS (16^0), SIXTEENS (16^1), and so on.

The ***Decimal Number System*** is the most common out of all the number systems because it is used in our everyday living. Since it is using 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) therefore, it has a base of ***10***. The ***Binary Number System*** uses only two digits (0 and 1) so it has a base of ***2***. ***Hexadecimal*** means 16. The ***Hexadecimal Number System*** uses 16 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F) so the base is also ***16***.

CONVERSIONS

Any particular number system can be converted to another number system by following the steps in conversion.

1. BINARY NUMBER TO DECIMAL NUMBER CONVERSION

STEPS:

a. Write the ***Binary*** number downwards.
b. Multiply each digit with the corresponding binary weights starting with ***UNITS*** from the bottom.
c. Add the products. The ***sum*** is the corresponding decimal number.

Example No. 1. CONVERT 1011101_2 TO DECIMAL.

Solution:

1011101_2 = $\underline{\ 93\ }_{10}$

$1 \times 2^6 = 64$
$0 \times 2^5 = 0$
$1 \times 2^4 = 16$
$1 \times 2^3 = 8$
$1 \times 2^2 = 4$
$0 \times 2^1 = 0$
$1 \times 2^0 = 1$

93

Example No. 2. CONVERT 110101001_2 TO DECIMAL.

Solution:

2. DECIMAL NUMBER TO BINARY NUMBER CONVERSION

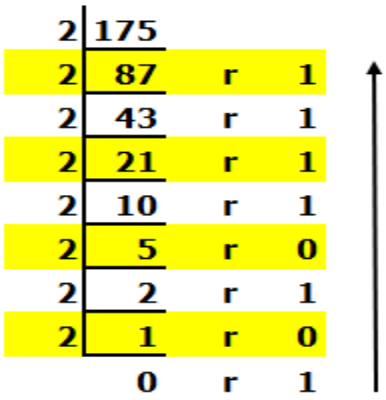
STEPS:

a. Write the given ***decimal*** number and divide it by 2.
b. Write the ***quotient*** and the ***remainder***.
c. Divide the ***quotient*** by 2.
d. Repeat steps b and c until the quotient is ***0***.
e. Write the digits from ***bottom to top***. The resulting string will be the ***binary*** number.

Example No. 1. CONVERT 175_{10} TO BINARY.

Solution:

175_{10} = $\underline{\quad 10101111 \quad}_{2}$



The process of successive division by 2 is called ***DOUBLE-DABBLE***.

Example No. 2. CONVERT 72_{10} TO BINARY.

Solution:

3. HEXADECIMAL NUMBER TO BINARY NUMBER CONVERSION

To convert ***Hexadecimal number*** to ***Binary***, the ***TABLE OF EQUIVALENCES*** should be used.

TABLE OF EQUIVALENCES

HEXADECIMAL	BINARY	DECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

STEPS:

a. Write the digits of the ***hexadecimal*** number separately.
b. Using the ***Table of Equivalences,*** write the corresponding ***binary equivalent*** under each digit.
c. ***Cross-out*** the unnecessary zeroes. The ***UNNECESSARY ZEROES*** can be found at the ***leftmost*** digit.
d. Write the ***binary*** digits as a string.

Example No. 1. CONVERT 143_{16} TO BINARY.

Solution:

$$143_{16} = 101000011_{2}$$

1	4	3
0001	0100	0011

Example No. 2. CONVERT $5F92_{16}$ TO BINARY.

Solution:

4. BINARY NUMBER TO HEXADECIMAL NUMBER CONVERSION

STEPS:

a. ***Group*** the digits of the ***binary*** number from the last digit by ***4's***.

b. If the ***last*** group is ***less*** than 4 digits, add zeros to the left to make it 4. Ex. 11, add 2 zeroes to make it 0011. All the groups should have 4 digits.

c. From the ***Table of Equivalences***, write below the corresponding hexadecimal number equivalent of each group.

d. Write the ***hexadecimal*** digits as a string.

Example No. 1. CONVERT 101100101_2 TO HEXADECIMAL.

Solution:

$$101100101_2 = 165_{16}$$

1 0 1 1 0 0 1 0 1

0 0 0 1 0 1 1 0 0 1 0 1

1 6 5

Example No. 2. CONVERT 101111010_2 TO HEXADECIMAL.

Solution:

5. HEXADECIMAL NUMBER TO DECIMAL NUMBER CONVERSION

STEPS:

a. Write the ***hexadecimal*** number downwards. If the ***hexadecimal*** digit is a letter, change it first to its corresponding decimal number.
b. Multiply each digit with the corresponding *hexadecimal weights* starting with *UNITS* at the bottom of the list.
c. Add the products. The *sum* is the corresponding decimal number.

Example No. 1. CONVERT $34B_{16}$ TO DECIMAL.

Solution:

$34B_{16}$ = $\underline{\quad 843 \quad}_{10}$

3	**x 16^2 =**	**768**
4	**x 16^1 =**	**64**
B = 11	**x 16^0 =**	**11**

		843

Example No. 2. CONVERT $C3B_{16}$ TO DECIMAL.

Solution:

6. DECIMAL NUMBER TO HEXADECIMAL NUMBER CONVERSION

STEPS:

a. Divide the ***hexadecimal*** number by ***16***.
b. Write the ***quotient*** and the ***remainder***.
c. Divide the quotient by 16.
d. Repeat steps b and c until the quotient is 0.
e. Write the digits from ***bottom to top***. The resulting string will be the ***binary*** number.

Example No. 1. CONVERT 195_{10} TO HEXADECIMAL.

Solution:

42195_{10} = $A4D3_{16}$

```
16 | 42195
16 |  2637    r     3
16 |   164    r    13 = D
16 |    10    r     4
      0      r    10 = A
```

Example No. 2. CONVERT 3175_{10} TO HEXADECIMAL.

Solution:

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 3

DO THE FOLLOWING CONVERSIONS:

1. 1010_{10} = _____$_2$ 2. 58_{10} = _____$_2$

3. 11000111_2 = _____$_{10}$ 4. 10111100_2 = _____$_{10}$

5. $AB8_{16}$ = _____$_{10}$ 6. $259F_{16}$ = _____$_{10}$

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 4

DO THE FOLLOWING CONVERSIONS:

1. $62E_{16}$ = _____$_2$

2. 115_{16} = _____$_2$

3. 984_{10} = _____$_{16}$

4. 3425_{10} = _____$_{16}$

5. 11100011101_2 = _____$_{16}$

6. 10001111_2 = _____$_{16}$

UNIT 3. PROGRAMMING LANGUAGES

Learning Objectives

At the end of the unit, I am able to:

1. *Understand the history of programming languages;*
2. *Identify the type of high-level programming;*
3. *Understand the control structures; and*
4. *Create a flowchart.*

HISTORY OF PROGRAMMING LANGUAGES

1. ***Machine Language***

   The first computer programming language ever developed was the ***machine language***. The machine language is written with ***binary*** digits, that is, 0s and 1s. Machine language is the sole programming language which is recognized by the computer, though it is very tedious and the tendency of making errors is extremely high. Machine language codes can only be written by expert low level programmers.

2. ***Assembly Language***

   The ***assembly language*** is a slightly high-level language than the machine language. This programming language uses ***mnemonics***. ***Mnemonics*** are memory aids which are abbreviated alphabetic instructions. The programmers developed this language so that they can easily understand their programs as compared to the ones they wrote using machine language. An ***assembler*** is used to convert codes written in assembly language into machine language.

3. ***High-level Language***

   The high-level language is a program written in English words that even common people can understand. Most of the programming languages today like C language, C++, Java and others are high-level languages. For the computer to understand and execute a program written in high-level language, it should first be compiled by a compiler program into object code and then combined into a single executable program using a linker.

TYPES OF HIGH-LEVEL PROGRAMS

1. Procedure-oriented programs

When you say procedure-oriented, the program is focused on its step by step process of solving a given problem.

Examples:

a. COmmon Business-Oriented Language (COBOL)
b. Pascal Language
c. C Language
d. Beginner's All-purpose Symbolic Instruction Code (BASIC)

2. Object-oriented programs

The object-oriented program focuses on transforming real-life objects into its digital representation. Examples of objects in programming are text boxes, frames, check boxes, buttons and combo boxes.

Examples:

C++ Visual Basic Java

CONTROL STRUCTURES

Control structures are statements which oversee the flow on how a program is executed. The program's logic depends on these control structures that is why they are sometimes called ***logic structures***.

1. Sequence Structure

In this control structure, the lines of the program are executed in a procedural method, meaning, the first statement will be executed by the computer first, followed by the second statement and so on and so forth until the last statement of the program is executed.

EXAMPLE:

Make an algorithm that will show the process by which a Grade I student does his daily routine from the time he wakes up to the time he goes to school.

SOLUTION:

1. Wake up early.
2. Eat breakfast.
3. Brush his teeth.
4. Take a shower.
5. Put on his uniform.
6. Go to school.

2. Selection Structure

In this structure, the execution of a statement or a block of statements will be dependent on outcome of comparing the values in an expression. The selection structure is also called as a ***decision structure***.

EXAMPLE:

From the previous example, make a selection structure in the part where the Grade I student is putting on his uniform. Every Wednesday is PE day, so the algorithm include a selection statement that will make him put on his PE uniform on a Wednesday.

SOLUTION:

1. Wake up early.
2. Eat breakfast.
3. Brush his teeth.
4. Take a shower.
5. If (today is Wednesday)
   Put on his PE uniform
   else
   Put on his school uniform
   end if
6. Go to school.

3. Repetition Structure

The Repetition Structure is sometimes called ***looping***. In this structure, the statement or block of statements are done again and again until the result of the comparison expression becomes "***false***".

EXAMPLE:

Make the algorithm above show that the process of going to school will be repeated 5 times to cover the 5 school days of the week.

SOLUTION:

1. Repeat 5 times

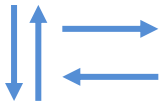
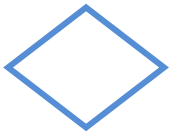
    - Wake up early.
    - Eat breakfast.
    - Brush his teeth.
    - Take a shower.

    - If (today is Wednesday)
      Put on his PE uniform
        else
          Put on his school uniform
      end if
    - Go to school.

    end repeat

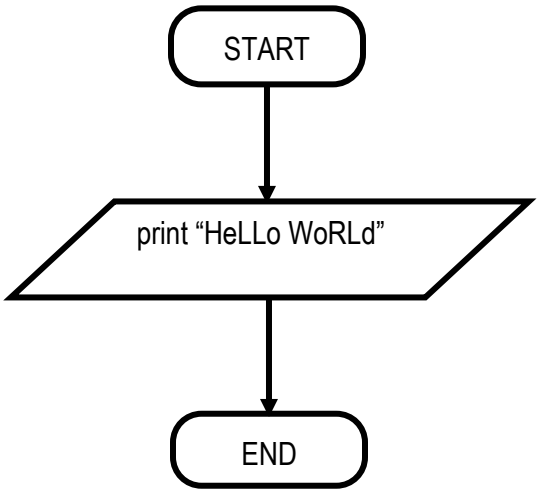
FLOWCHART

Flowchart is a guide in creating programs. It uses symbols to show the step by step process of how you can develop a solution to a certain problem. The table below lists the commonly used flowcharting symbols and their corresponding functions.

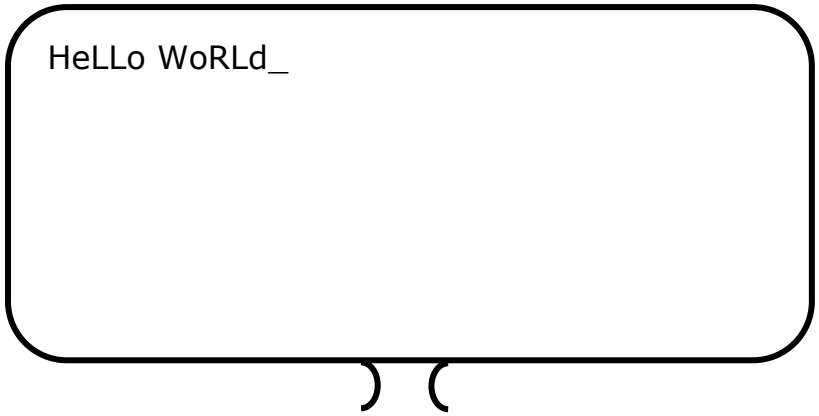
SYMBOL	NAME	FUNCTION
	***TERMINAL***	Used to denote the beginning and ending of a flowchart.
	***FLOWLINES***	Used to show the next step in the flowchart.
	***PREPARATION***	Used for declaring and initializing variables and/or constants.
	***INPUT/OUTPUT***	Used for entering data and for displaying the results.
	***PROCESS***	Used for assigning values and calculations.
	***DECISION-MAKING***	Used for branching. Shows which alternative will be followed by the flowchart.
	***CONNECTOR***	Used to connect a part of a flowchart to another at the same page.

EXAMPLE No. 1:

Create a flowchart that will display "HeLLo WoRLd".



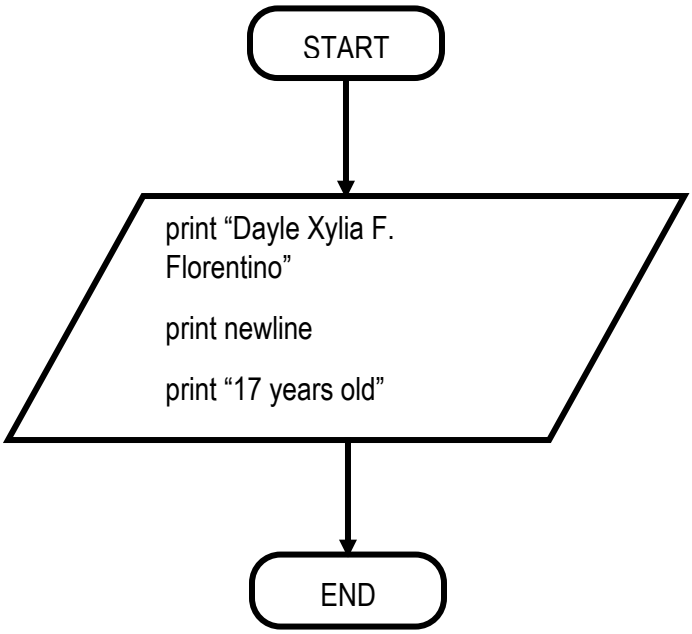
SAMPLE OUTPUT:



Notice that only the text inside the double quotations is printed. The quote-unquote were not printed on the monitor. Whatever form the text inside the quote-unquote takes, the same will be printed on the screen. The letters **HLLWRL** are displayed in capital letters while the letters **eood** are in small letters. This process is coined as **WYSIWYG** (***What You See Is What You Get***). Notice also that the cursor (represented by **_**) is displayed after the last symbol printed on the screen. It is where the next symbol will be printed. If you want the succeeding symbols to be printed on a new line, you should print a newline so that the cursor will go to the next line instead of to the right of the last printed symbol.

EXAMPLE No. 2:

Draw a flowchart that will show your name, and age in the monitor.

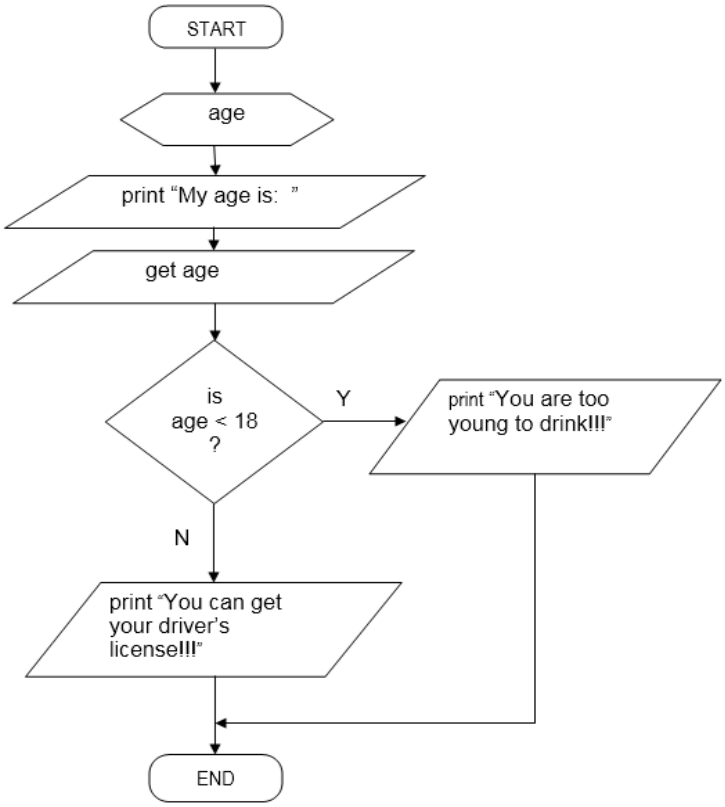


SAMPLE OUTPUT:

In this example, the first print statement (Dayle Xylia F. Florentino) was displayed on the screen, then a newline was printed, so the cursor transferred to the next line. The third print statement (17 years old) was displayed on the screen. A newline was next printed so the cursor appeared on the next line of the last printed line.

EXAMPLE No. 3:

Draw a flowchart that will input your age and display "You are too young to drink!!!" if your age 17 years or less, or print "You can get your driver's license!!!" if your age is 18 or more.



My age is: 16
You are too young to drink!!!_

In this flowchart, variable ***age*** was first declared. Then, the prompt (***My age is:***) is printed on the monitor. The user entered 16 and the value was stored in variable ***age***. 16 was compared to 18 (***is age < 18?***). Since the result is yes, the message (***You are too young to drink!!!***) was displayed and the cursor remained after the last printed character.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 5

I. ANSWER THE FOLLOWING QUESTIONS.

1. Which language uses binary digits?

_____

2. What is the flowcharting symbol used to declare variables?

_____

3. Which control structure chooses from alternatives?

_____

4. What are the three examples of Object-oriented program?

_____, _____, and _____

5. What type of program is focused on its step by step application?

_____

6. What is the flowcharting symbol used to connect parts of a program?

_____

7. What is the meaning of the acronym BASIC?

_____

8. What is the flowcharting symbol used to mark the start and end of a flowchart?

_____

9. What is another name for Loop Structure?

_____

10. What is the meaning of the acronym CoBOL?

_____

11. What is the visual representation of a program?

_____

12. What is the flowcharting symbol for branching?

_____

13. What is a slightly advanced language than the machine language?

_____

14. What are memory aids which are abbreviated alphabetic instructions?

_____

15. What is the flowcharting symbol used for computations and assigning values?

_____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 6

Create a flowchart that will print five lines of codes containing the translation of the words "I LOVE YOU" in different languages. Include the country in each line. Show the sample output.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 7

Create a flowchart that will print three lines of one-sentence messages on the screen. The first lines should be the fullname of your classmate (crush). The second and third lines should indicate why you like him or her. Show the sample output.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 8

Create a flowchart that will input a 3-digit number and display that number on the screen. Show the sample output.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 9

Two numbers will be input values from the user. Create a flowchart that will compute, then, display the sum of the numbers multiplied by 2 on the screen. Show the sample output.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 10

Draw a flowchart that will input a number, then display a message that tells if the input number is equal to 14, more than 14 or less than 14. Show the sample output.

HINT: If the number is not equal to 14 and not less than 14, it is definitely more than 14 so you don't have to ask anymore.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 11

Draw a flowchart that will input two numbers and display the bigger number. If the input numbers are the same, display an error message.

UNIT 4. C++ PROGRAMMING

Learning Objectives

At the end of the unit, I am able to:

1. *identify the parts of a simple C++ program;*
2. *learn about different C++ applications;*
3. *create and compile C++ programs;*
4. *debug a C++ source code; and*
5. *execute a C++ program.*

For the programs that you will be writing in this programming subject, you will be using C++. C++ is a high-level programming language which uses codes that are easier for common people to understand.

Step by step tracing of the code ***(debugging)*** is necessary so that you may know precisely where errors are occurring. A debugger is a tool that helps programmers trace programs and find probable causes of errors. It cannot correct the errors in your program. It only assists the programmer find errors while in the process of programming. The graphically based debugger environment is a part of the ***Integrated Development Environment (IDE).***

The simple structure of a C++ program is shown below.

```
1   #include <iostream>
2   #include <cstdlib>
3
4   using namespace std;
5
6   int main()
7   {
8       statement_1;
9       statement_2;
10      .
11      .
12      .
13      statement_n;
14
15      return EXIT_SUCCESS;
16  }
17
```

Examples of IDEs that you can use :

- for desktops and laptops: Code::Blocks, text editor with GNU compiler;
- for Android: cppDroid, cxxDroid, C++ Compiler;
- for iOS: cppCode

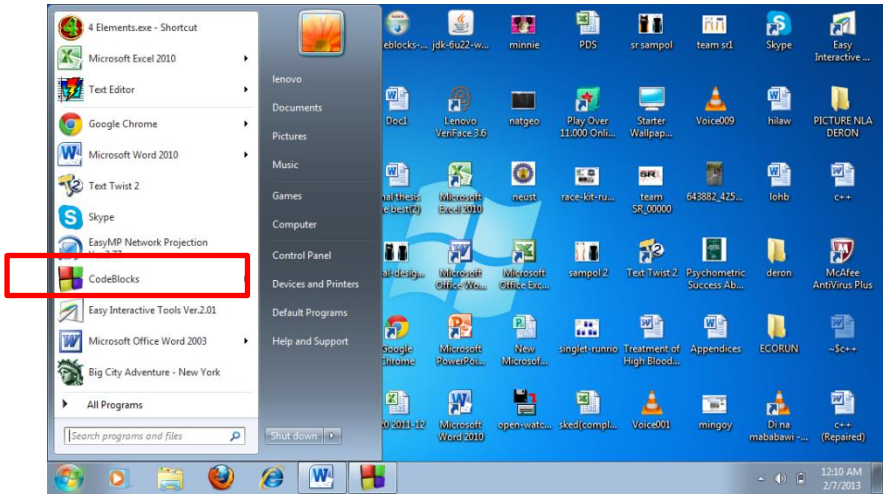
APPLICATIONS FOR DESKTOPS/LAPTOPS

1. CODE::BLOCKS

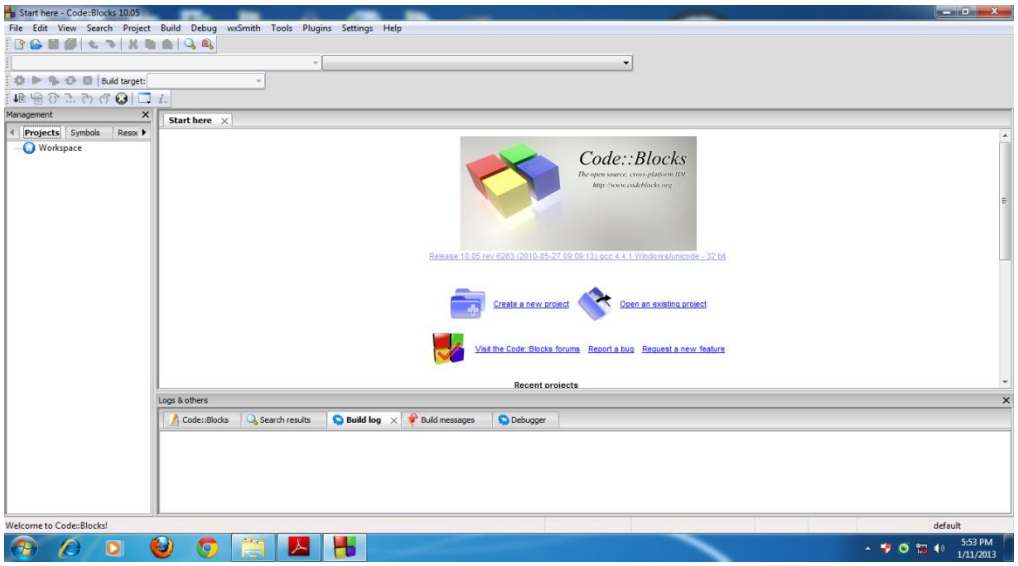
Code::Blocks is an open source free IDE which is used for C++ Programming which uses a text editor and a compiler. It will allow you to create and test your programs from an easy to use development environment.

OPENING Code::Blocks

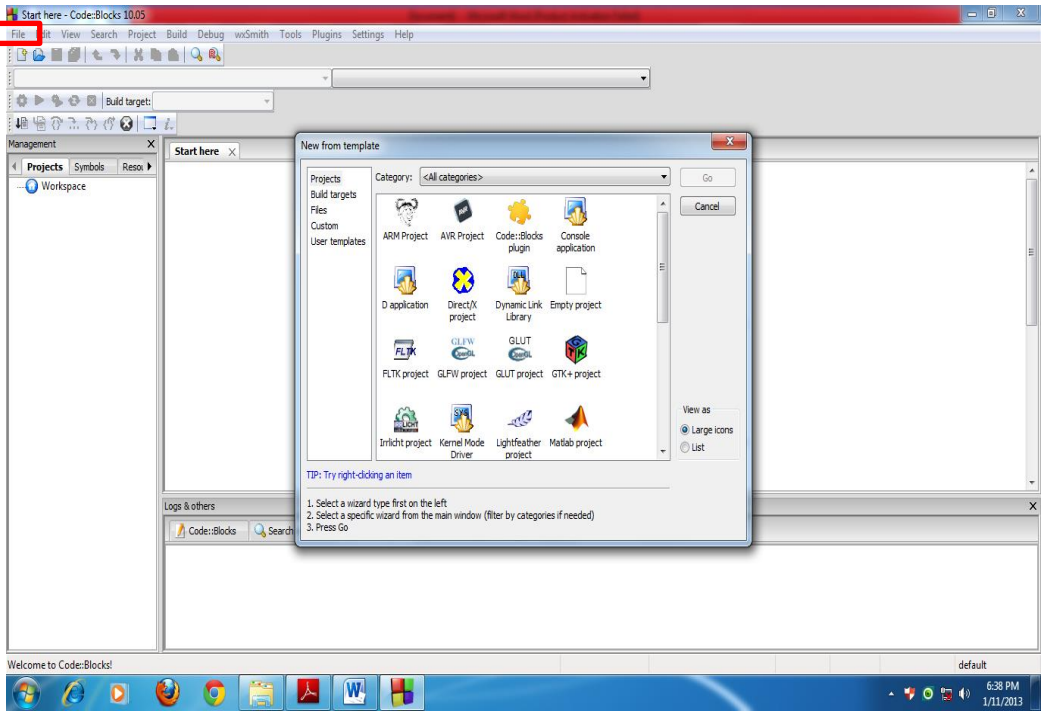
1. From the desktop, look for the ***Code::Blocks*** icon. Double click the icon. (If you cannot find the Code::Blocks icon in the desktop, you can click the ***START*** button and then look for ***Code::Blocks*** and click it).



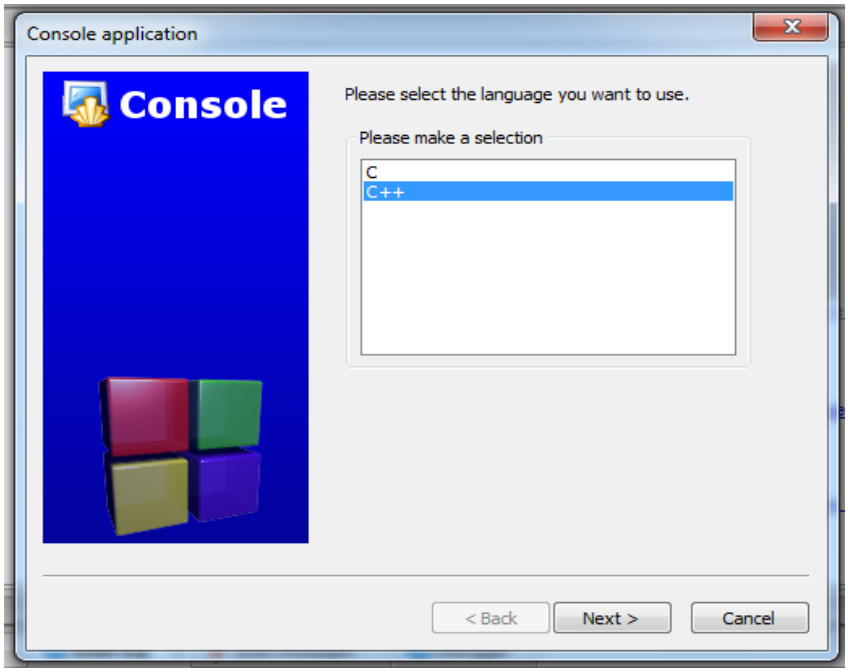
The figure below will be shown in your monitor.



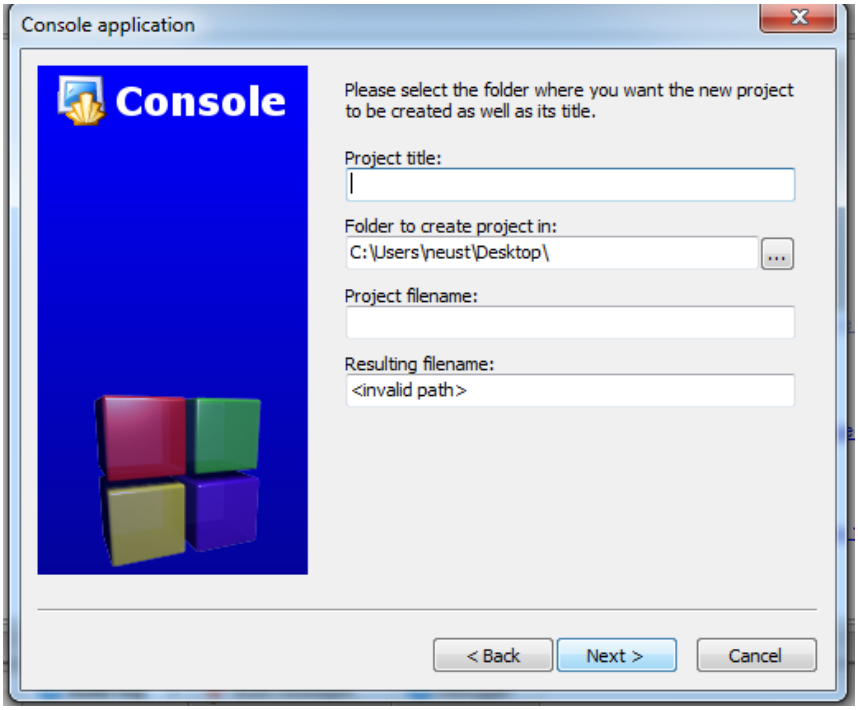
2. Click the ***File Menu*** from the Menu bar and then, click ***New***, and proceed to clicking ***Project***. You will be presented with a new window like the one below.



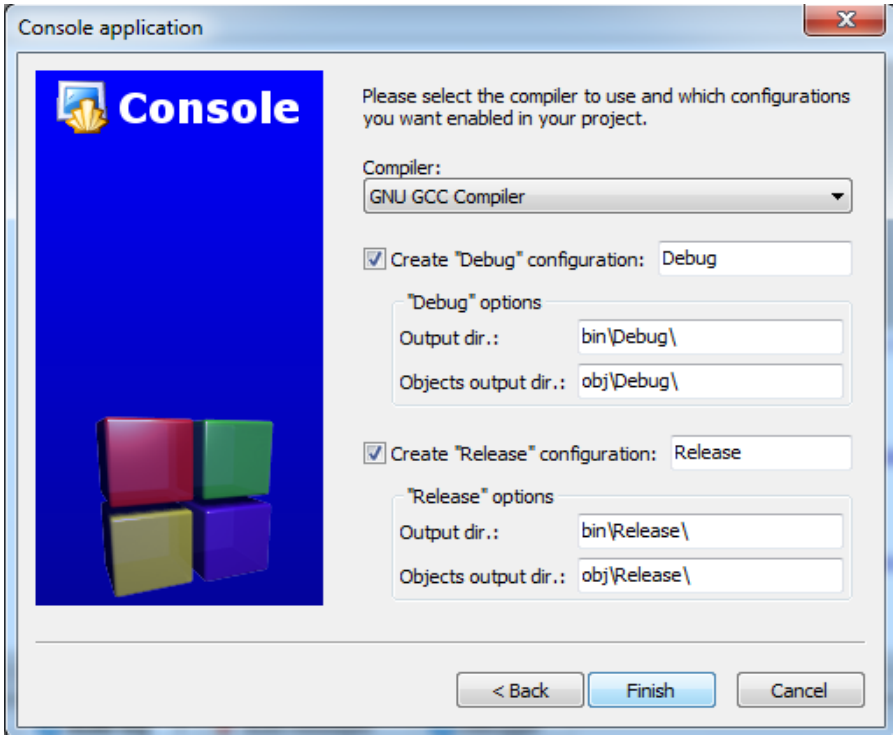
3. Choose ***Console Application*** by double clicking its icon. Your screen will now look like this Figure:



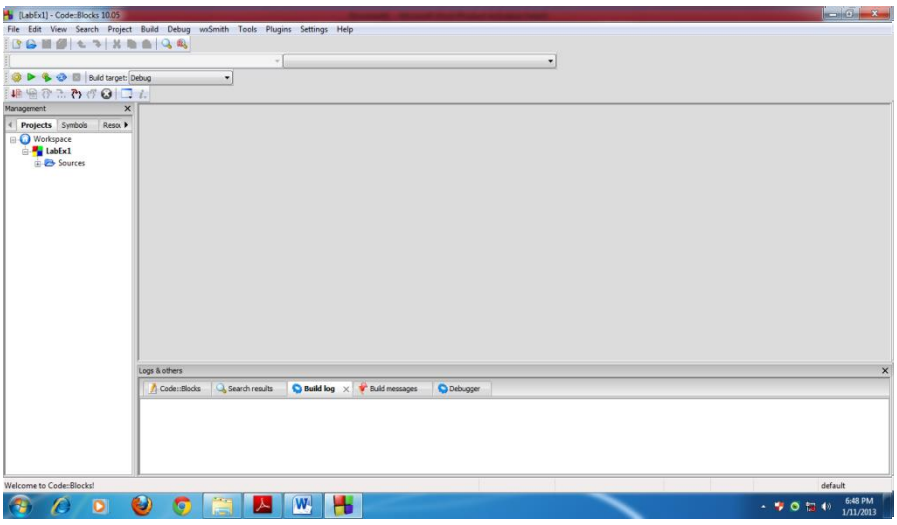
4. Choose ***C++*** and then, click ***Next***.



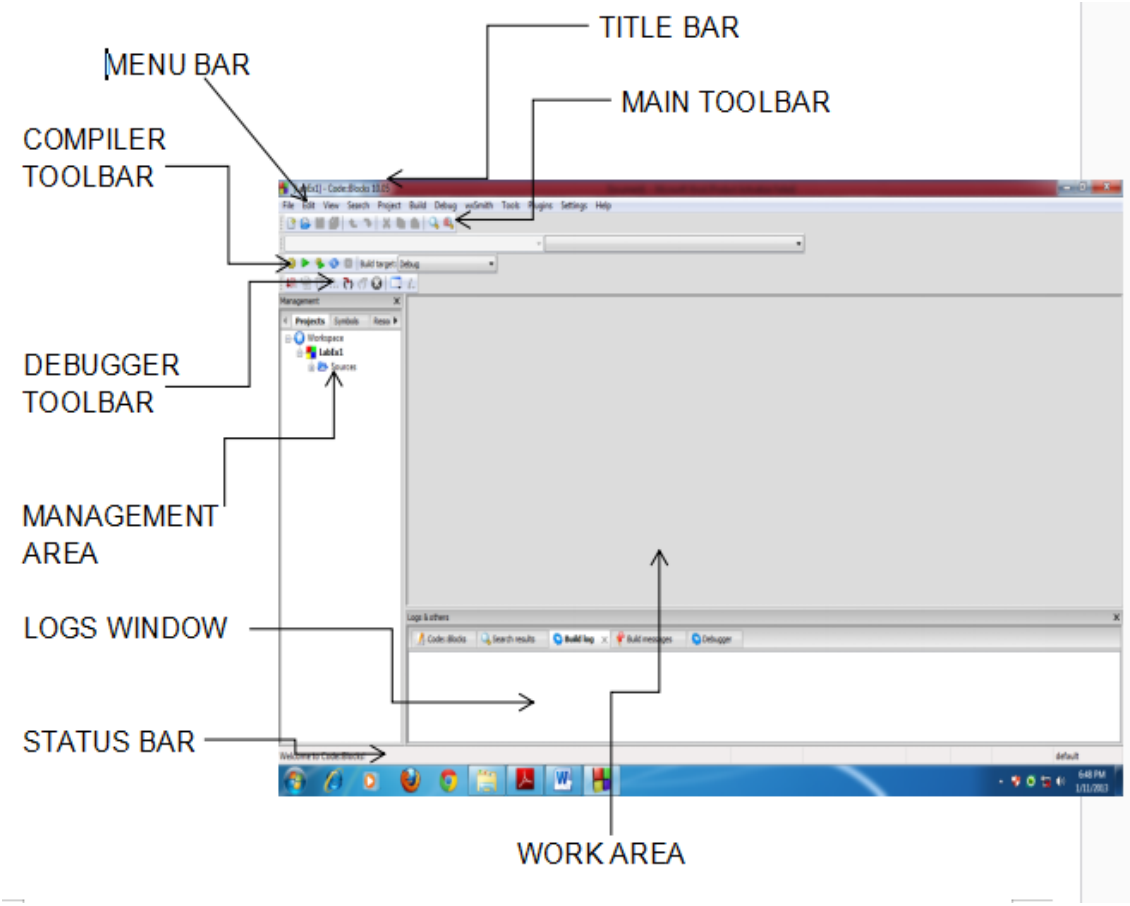
5. Type the title of the project (example, ***LabEx1***) and then, click ***Next***. The figure below will be your next screen.



6. Click ***Finish***. You will now be presented with the ***Code::Blocks Editor screen***.



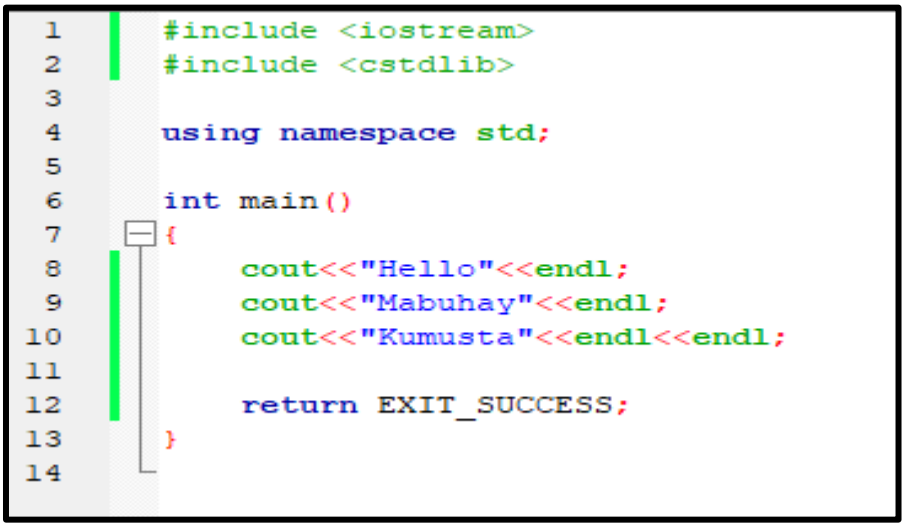
THE Code::Blocks SCREEN



OPENING Code::Blocks TEXT EDITOR

From the Code::Blocks Screen, click ***Projects*** on the ***Management Area***, then find the ***file*** you have created, double click ***Sources***, then double click ***main.cpp.*** The Work Area will display the basic structure of a simple C++ program. You can erase the unnecessary statements in the given program or you can completely erase all the data and create your own program. To familiarize yourself in creating a C++ program, erase the data completely. Type this simple program, following the structure as you can see it.

EXAMPLE OF A SIMPLE C++ PROGRAM



```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    cout<<"Hello"<<endl;
    cout<<"Mabuhay"<<endl;
    cout<<"Kumusta"<<endl<<endl;

    return EXIT_SUCCESS;
}
```

#include <iostream>
#include <cstdlib>

The first two lines are the declaration of libraries. The ***number sign*** or ***hash sign*** (#) indicates that the line is a directive for the compiler's preprocessor.

These libraries should be invoked in your program so that commands that you are bound to use in your codes will be able to function. The <iostream> contains the basic input and output commands of C++ while the <***cstdlib***> includes the ***namespace std***.

using namespace std;

A ***namespace*** is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols. ***namespace std*** is a ***namespace*** for the standard library where common C++ commands are included. This declaration says that we will be accessing the functionality of the ***standard namespace***. Also, you'll need to include **using namespace std;** to make the short name of the command visible instead of requiring the cumbersome long command. For example, instead of **std::cout**, you just have to type **cout** to display the text**.** (As a side note, std is a C++ *namespace* for many pieces of functionality that are provided in standard C++ libraries. For the purposes of this class, you won't need to otherwise know about namespaces.)

int main()

This line is the main method of your C++ program where all the statements will be executed by the program and is enclosed by ***opening*** (***{***) and ***closing*** (***}***) ***braces***.

cout<<"Hello"<<endl;
cout<<"Mabuhay"<<endl;
cout<<"Kumusta"<<endl<<endl;

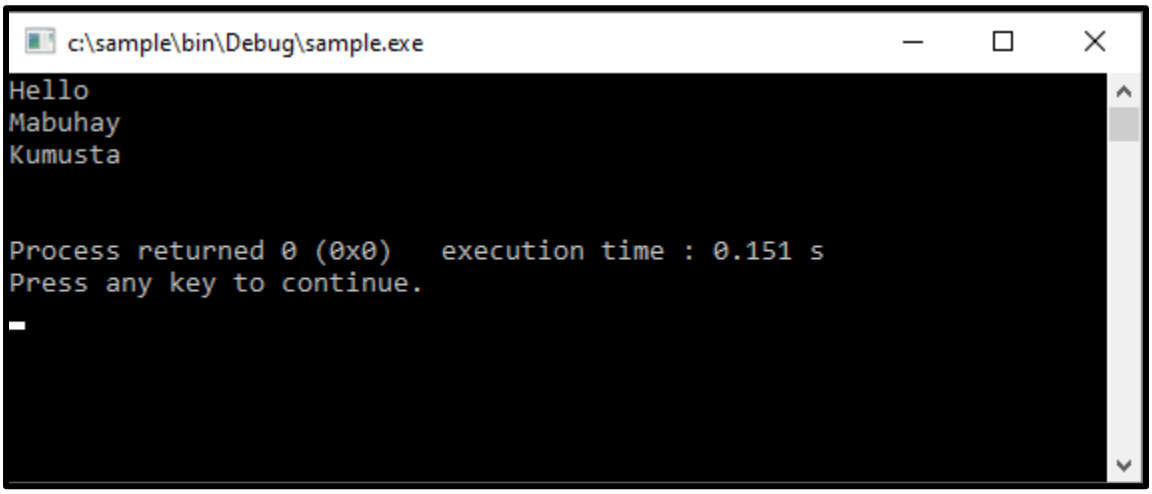
The sixth, seventh and eighth lines are C++ output ***statements***. Every C++ statement should end with a semicolon(***;***). ***cout*** is a command which will display anything inside the quote and unquote marks. It is followed immediately by an ***insertion operator*** (<<) and then, the ***quote***, the ***literal string*** or text to be printed, the ***unquote*** and a ***semicolon***. Literal strings are texts which are inside the quote-unquote symbols and will be displayed as is, or the way the strings are written or coded. These ***literal strings*** will be directed to the ***standard output stream***, usually, the ***screen*** or ***monitor***. The ***endl*** command will print a newline to the output. This means that the cursor will go down the next line of the screen. It is just like pressing the Enter Key when typing in a text editor.

return EXIT_SUCCESS;

The ninth line is a statement that causes a main function to end. This statement is translated as "***the program has functioned as expected and is deemed a success***."

- return ➡ exit function
- EXIT_SUCCESS ➡ constant with an integer value passed on to the Operating System when the program terminates

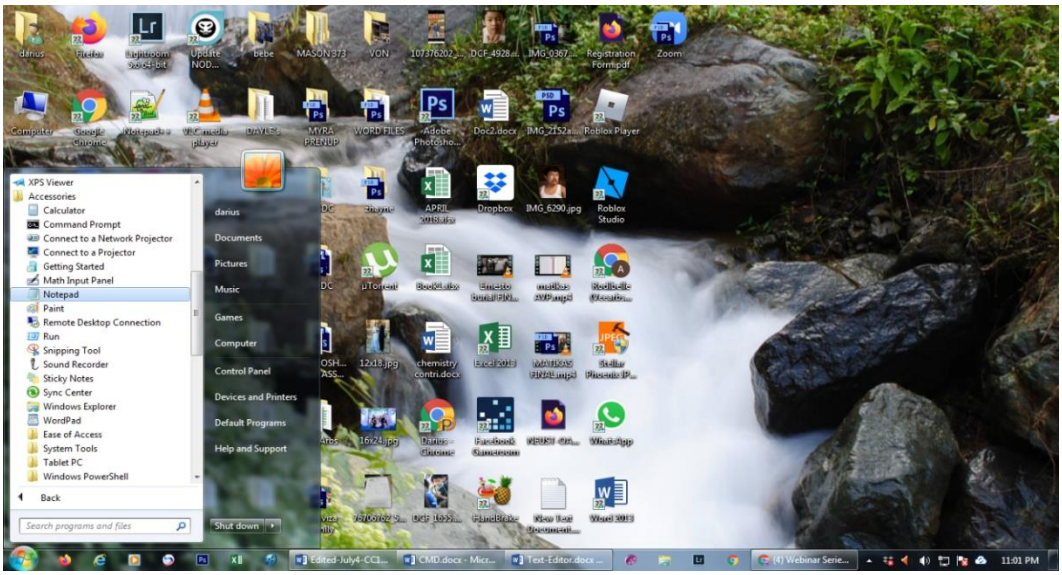
SAMPLE OUTPUT



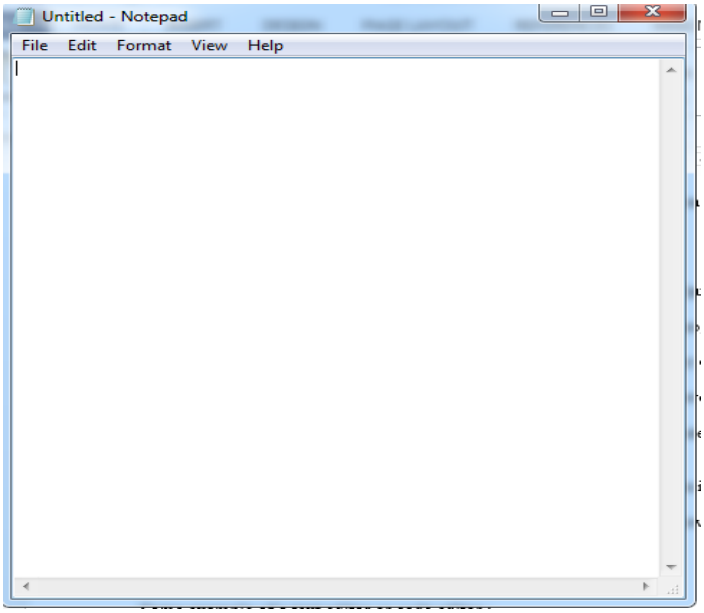
2. TEXT EDITOR AND COMMAND LINE INTERFACE

TEXT EDITOR

Use notepad as your text editor. To open notepad, click ***Start***, then choose ***All Programs*** and ***Accessories***. Then click on ***Notepad***.



The computer will display the Notepad window.

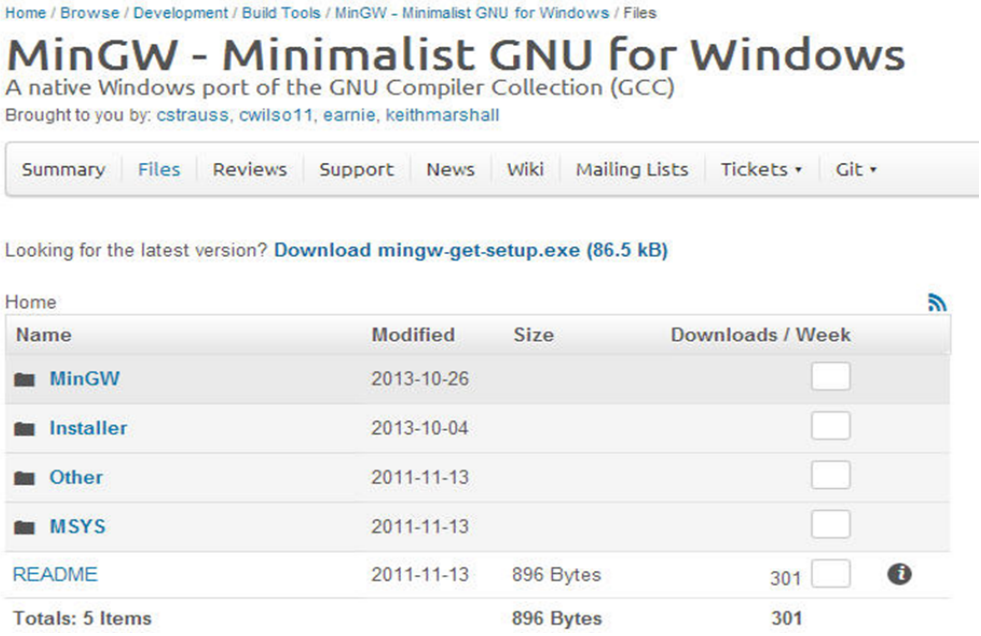


You can now write the source code in the Notepad window. To run the program that you have encoded using your text editor, you have to use a Compiler to execute the program.

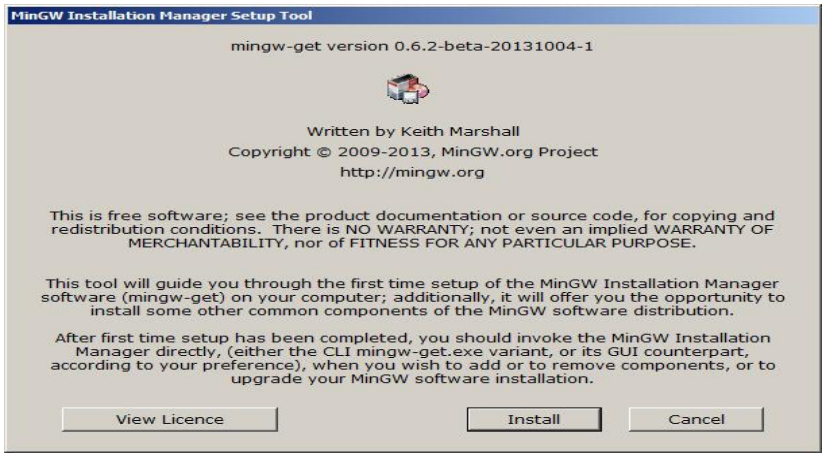
COMPILER

For the compiler, you can install MinGW in your desktop using the following steps:

1. Type https://sourceforge.net/projects/mingw/files/ in your browser.

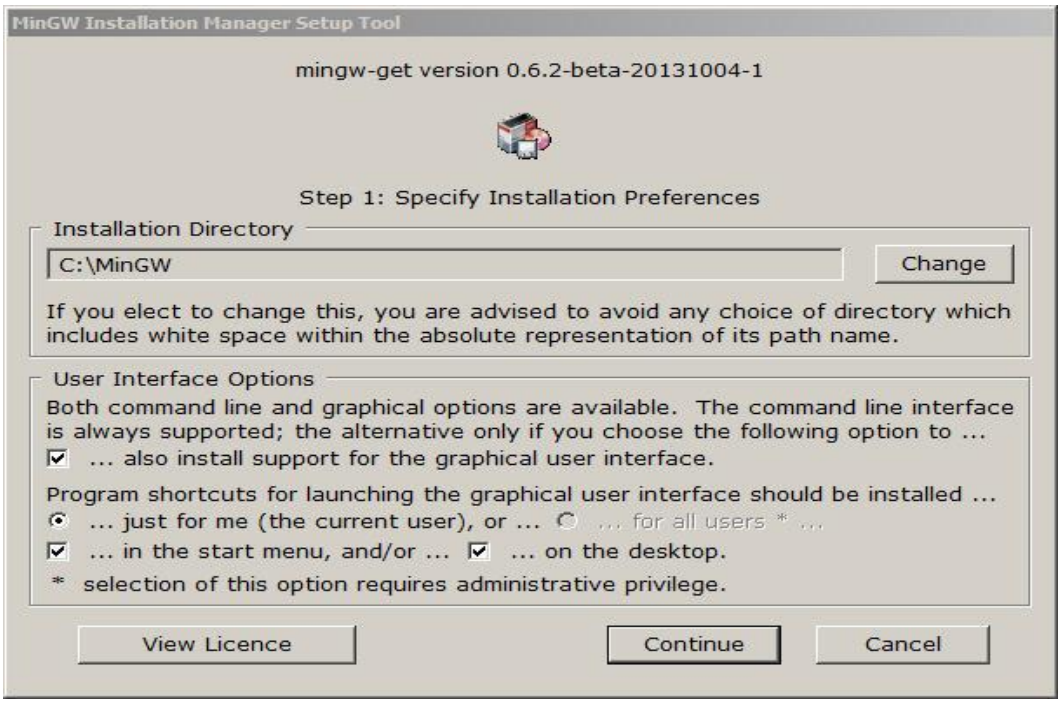


2. Click Download mingw-get-setup.exe.
3. Double-click the mingw-get-setup.exe icon.

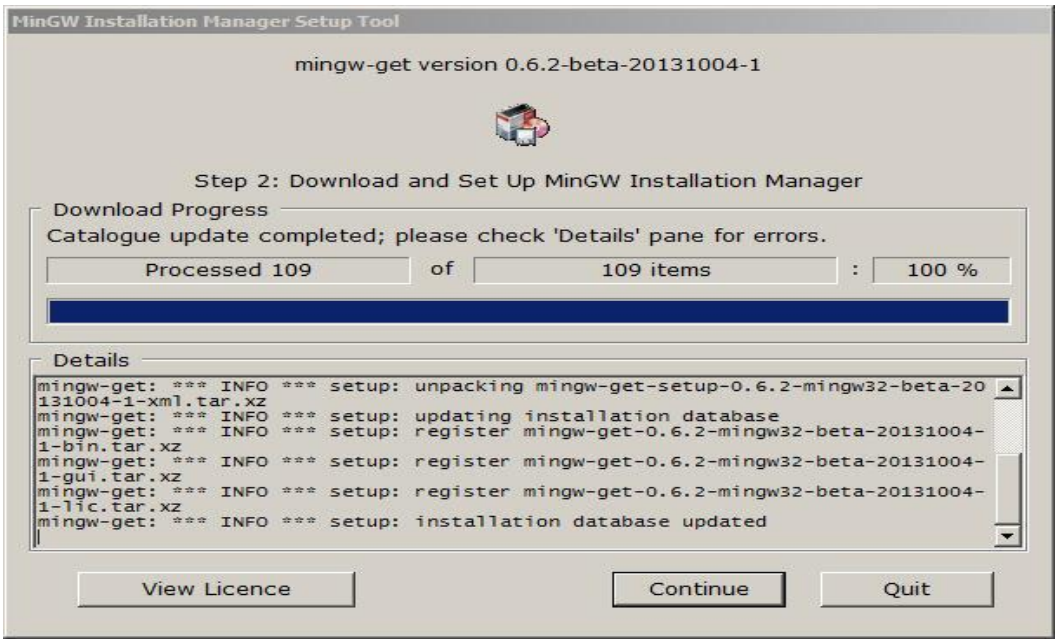


4. Click **Install**.

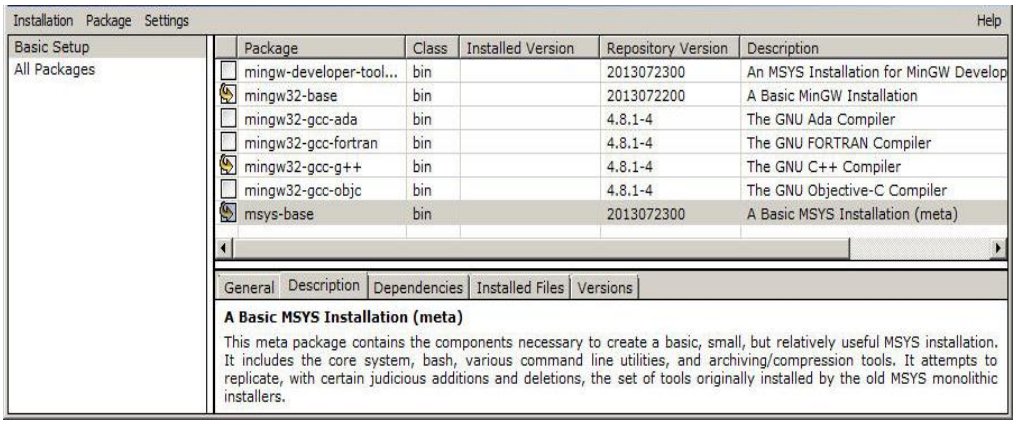
This window will be displayed on your screen.



5. Click **Continue**.

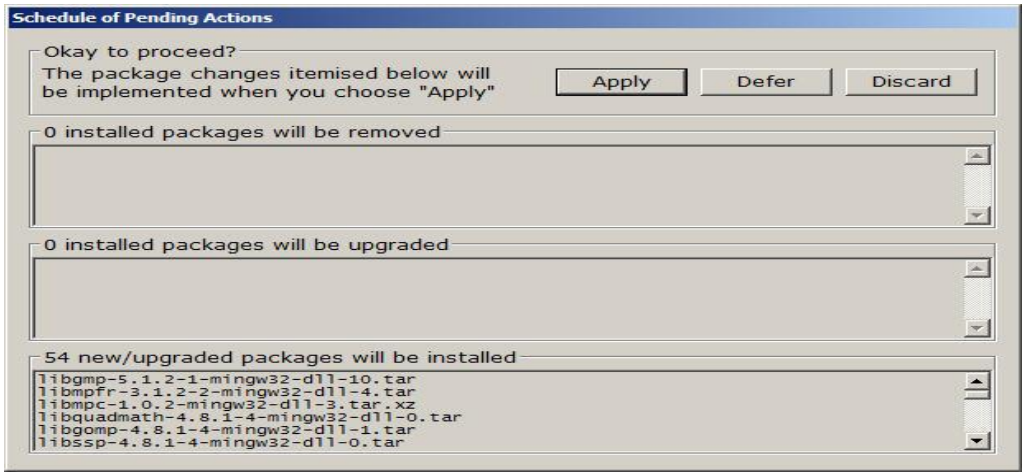


6. Click **Continue**.



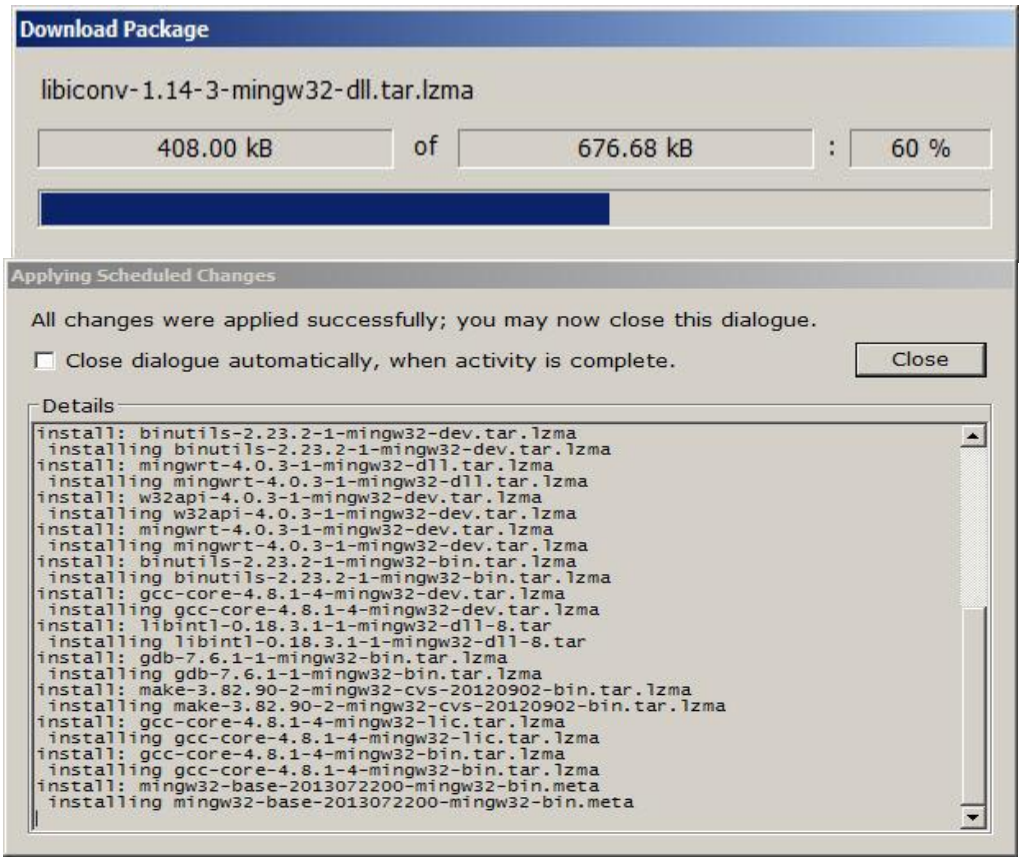
7. Click **x** in the MinGW Installation Manager.

8. Click **Review Changes**.



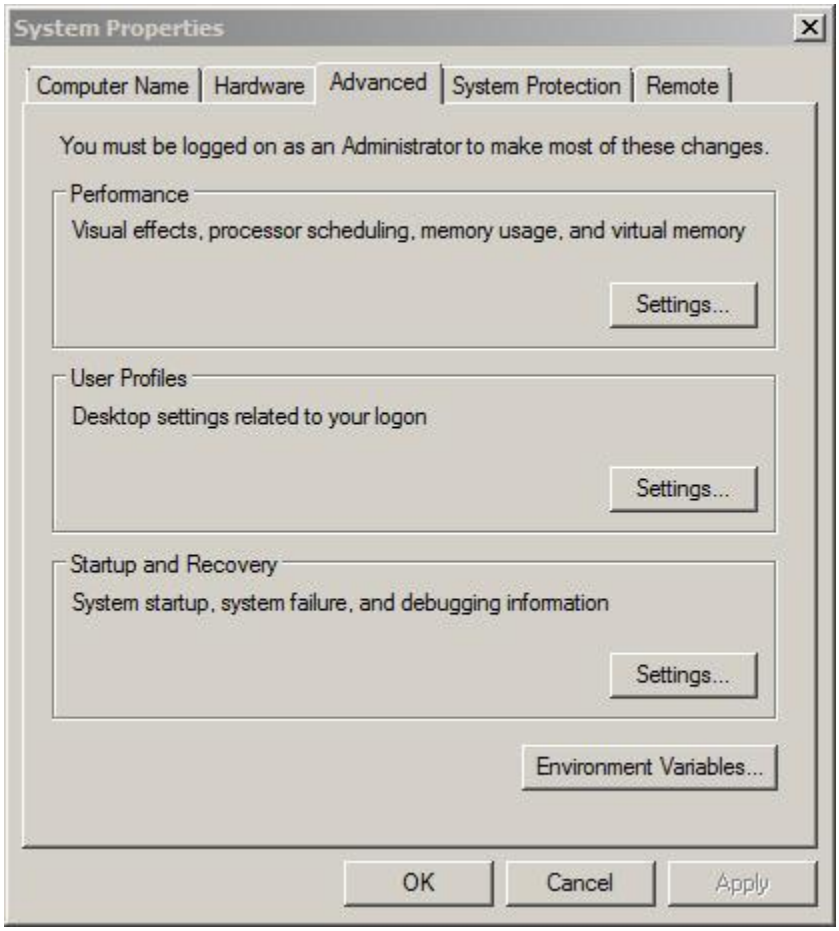
9. Click **Apply**.

The following windows will be displayed on your screen.

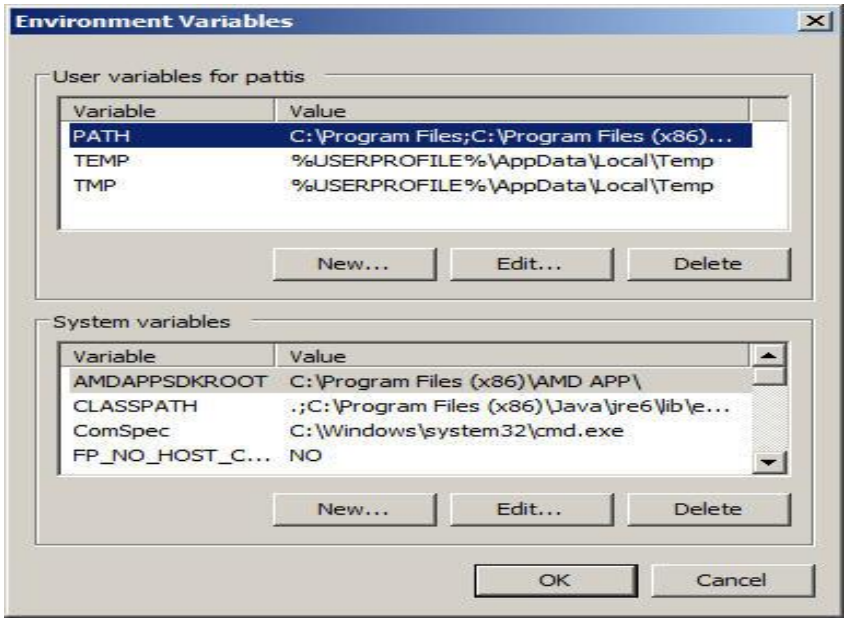


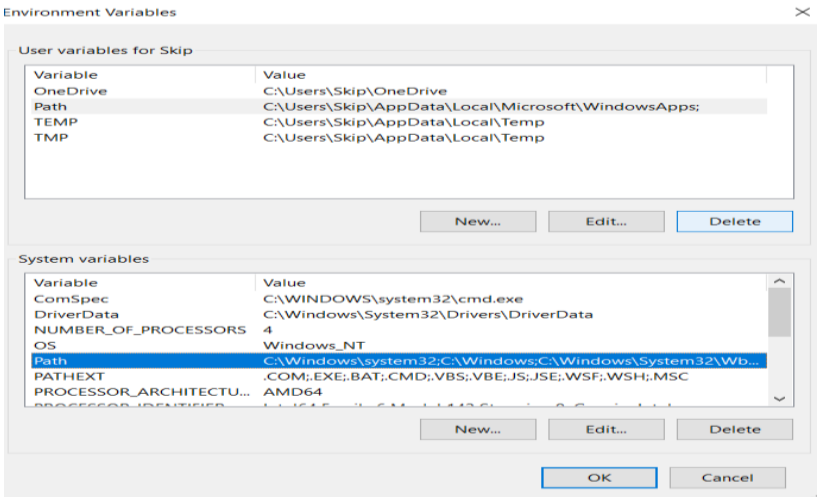
10. Click **Close**.

1. Edit the Environment Variable Path
    - Click **Start**
    - Click **Control Panel** / for windows 10- Search Control Panel
    - Click **System** / for windows 10 - Click System and Security , Click System
    - Click **Advance system settings**; you will see

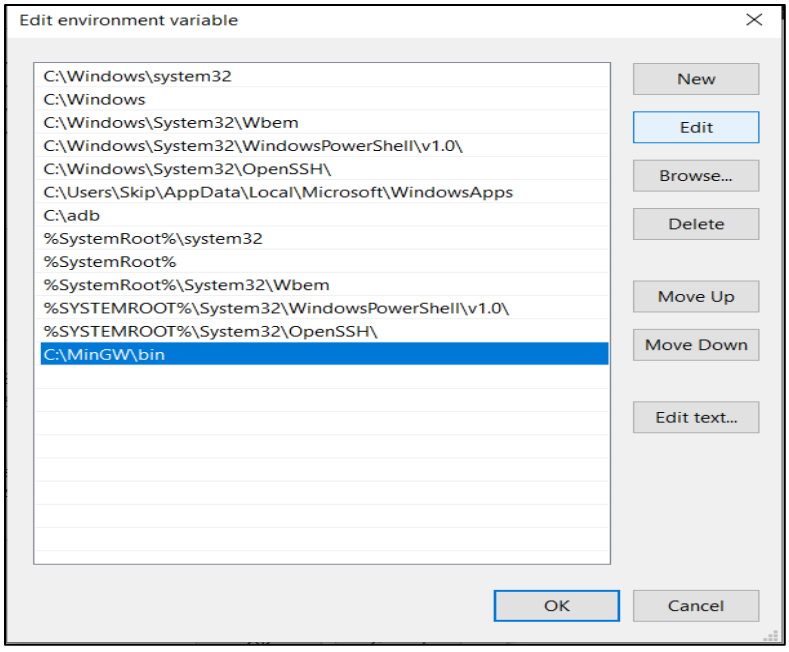


- Click **Enviroment Variables...**; you will see



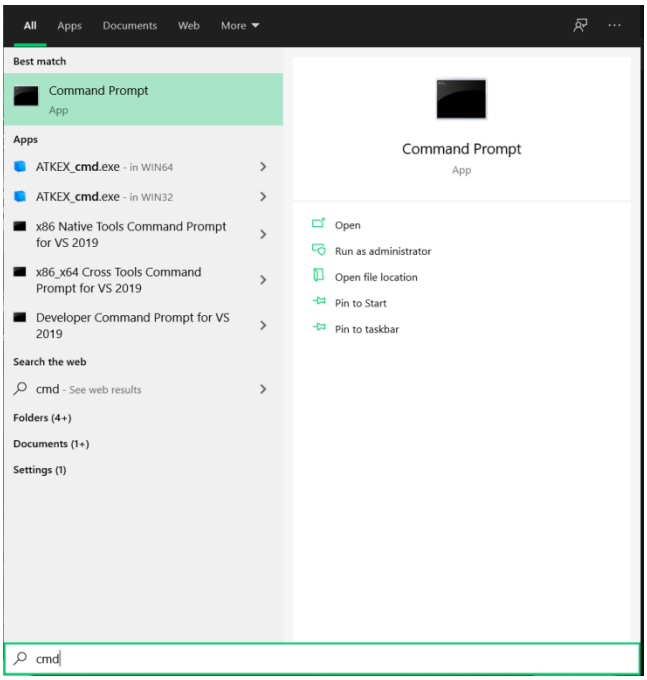


11. Click **OK** three times.

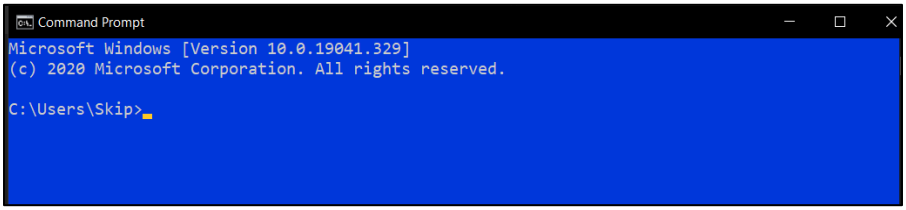


Create a source code and compile it on the Command Line Interface

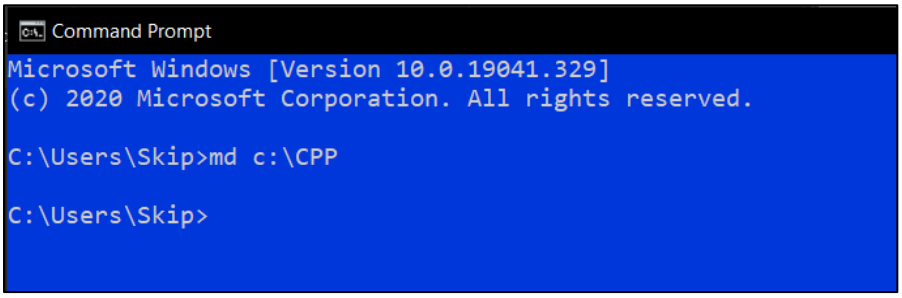
1. Search for the Command Prompt and click to select.



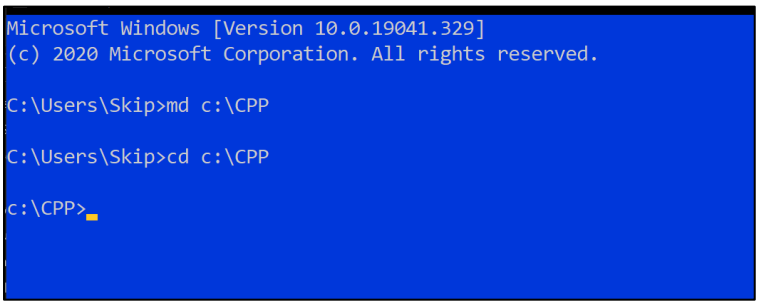
2. Your screen should look like this:



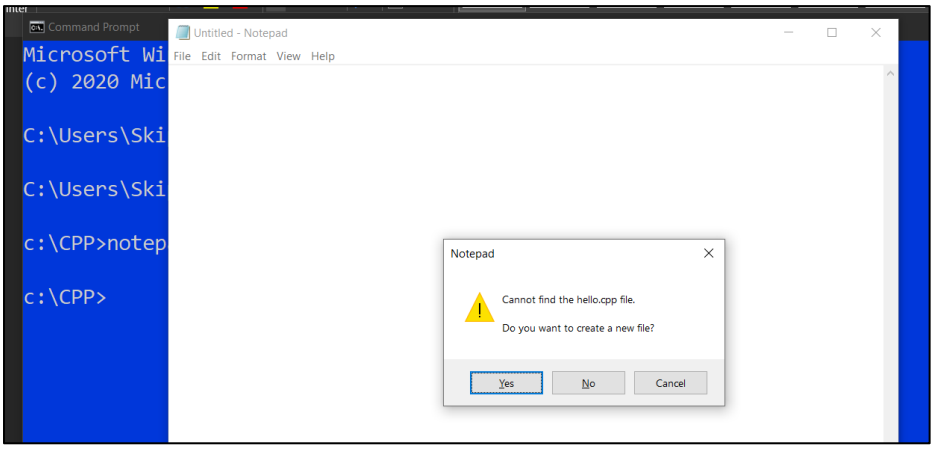
3. Type md c:\CPP to create a folder and hit Enter *md(make directory)



4. Type cd c:\CPP to change directory to change the directory to CPP so that the sorce file will be saved in that directory.



5. Type notepad hello.cpp



Click **Yes** to start coding using notepad.

Then, type this c++ code.

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello world, from C++!" << endl;
}
```

 Save or hit **CTRL S.**

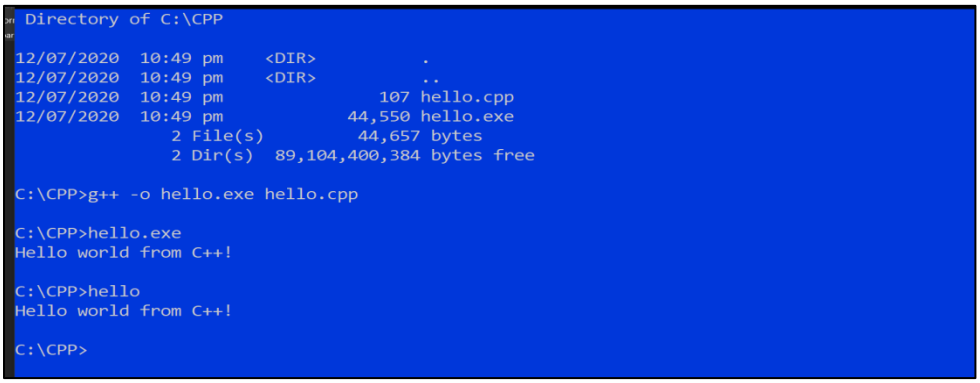
 You have successfully saved your source code.

Go back to the command prompt. Type dir to display the contents of a c:\hello directory.

Then type g++ -o hello.exe hello.cpp tOr type g++ hello.cpp.

Compiling with g++

 To run the hello.exe program, type ***hello.exe*** or ***hello*** or simply type ***a*** at the command prompt.
Output:



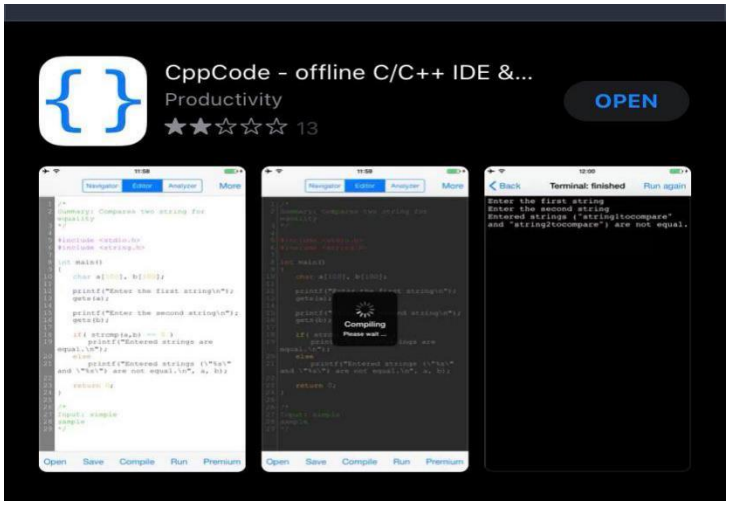
Congrats, you have compiled and run a C++ program by using the command-line tools.

C++ Application for iOS Devices

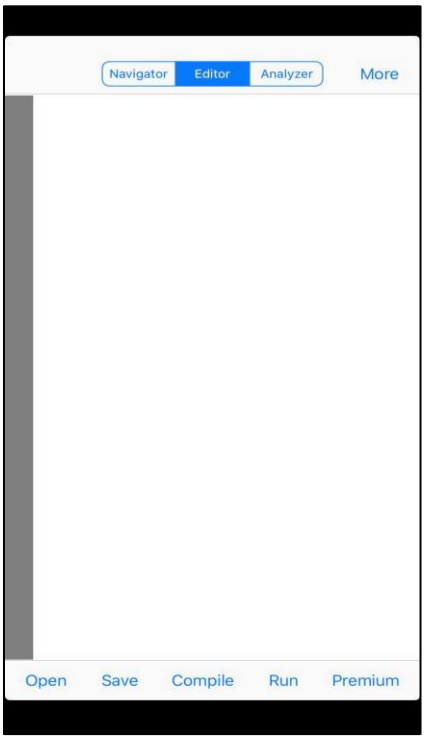
1. CppCode

Download, Installation, and Opening of CppCode

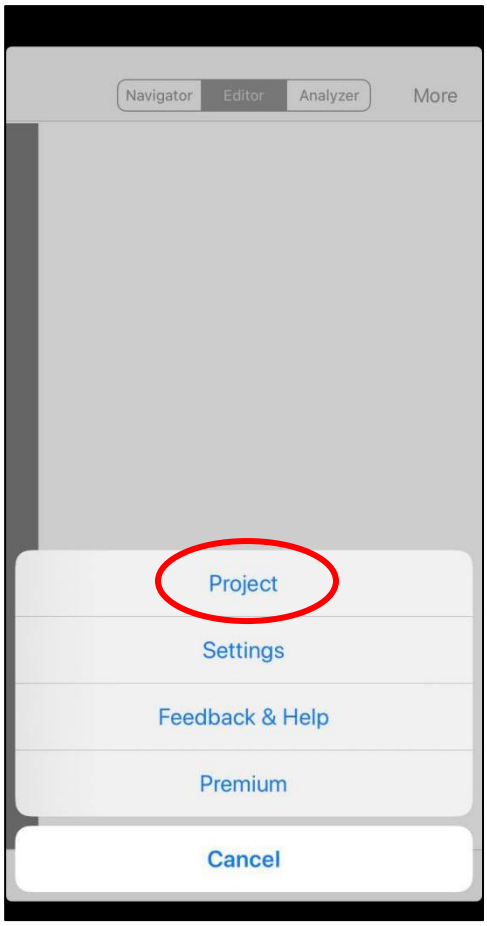
1. From the App Store, download and install CppCode.

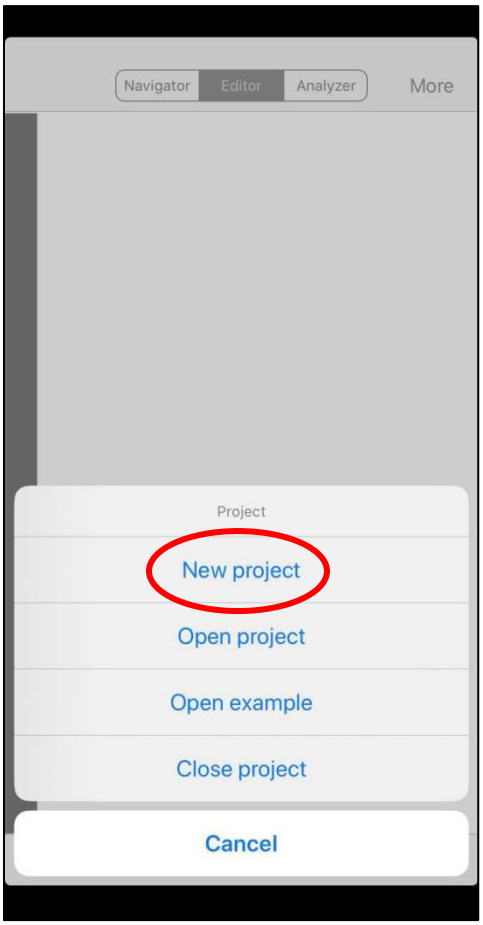


2. Upon opening the application, the main screen will appear as described in the figure below.

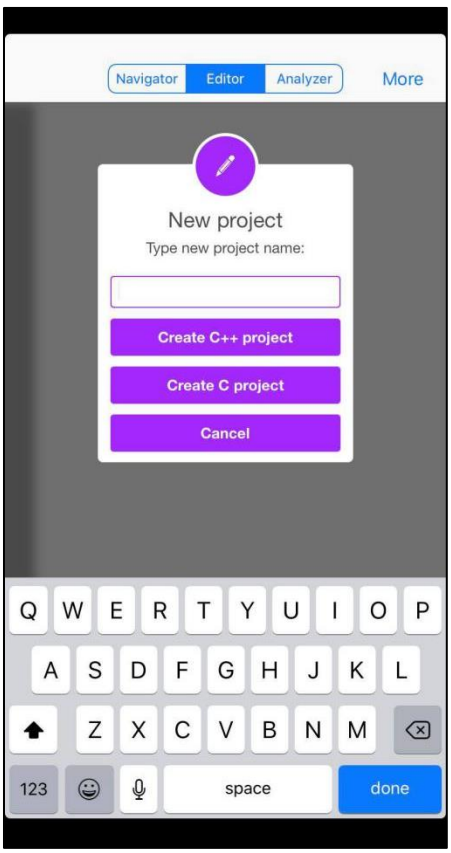


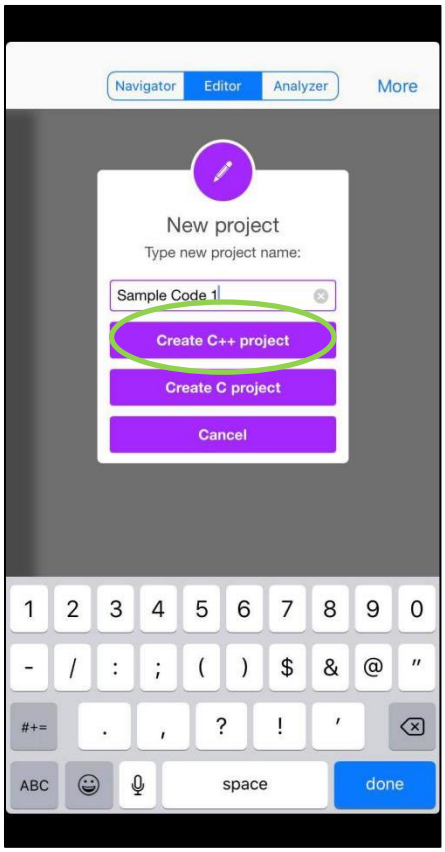
3. To create a new C++ project, click on ***More*** on the upper right hand of the screen then choose ***Project***. Next, choose ***New Project***.



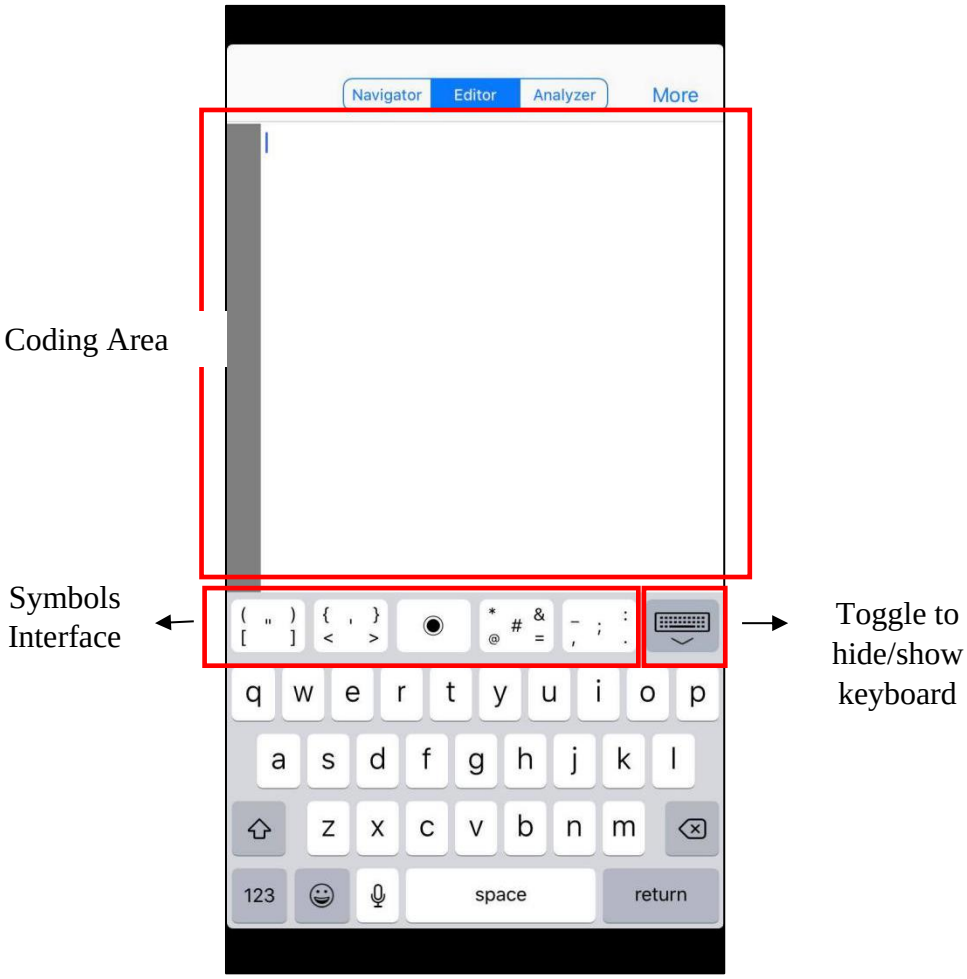


4. After tapping New Project, a new window will appear. Input the file name you want in the text field then choose ***Create C++ Project***.

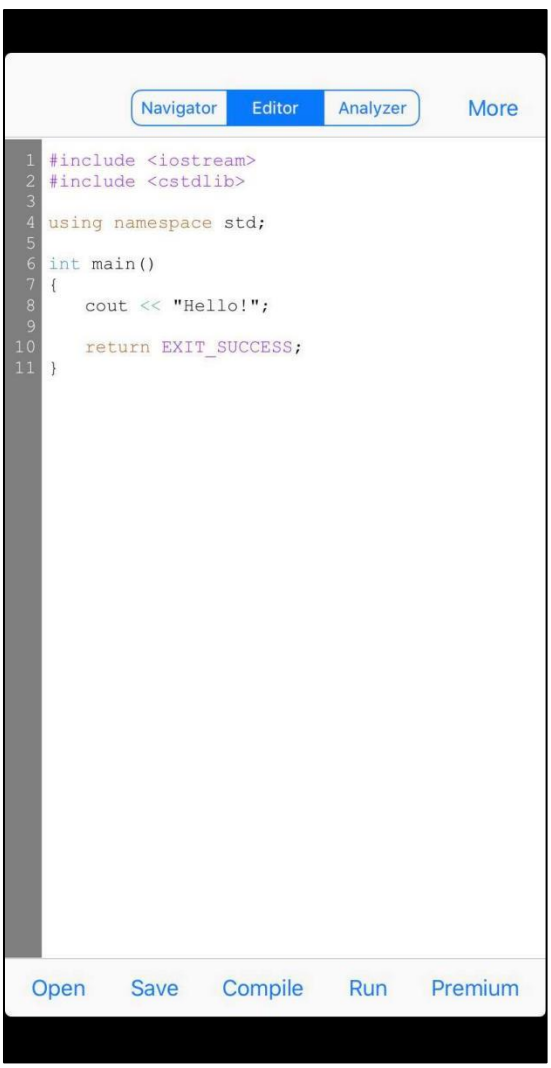


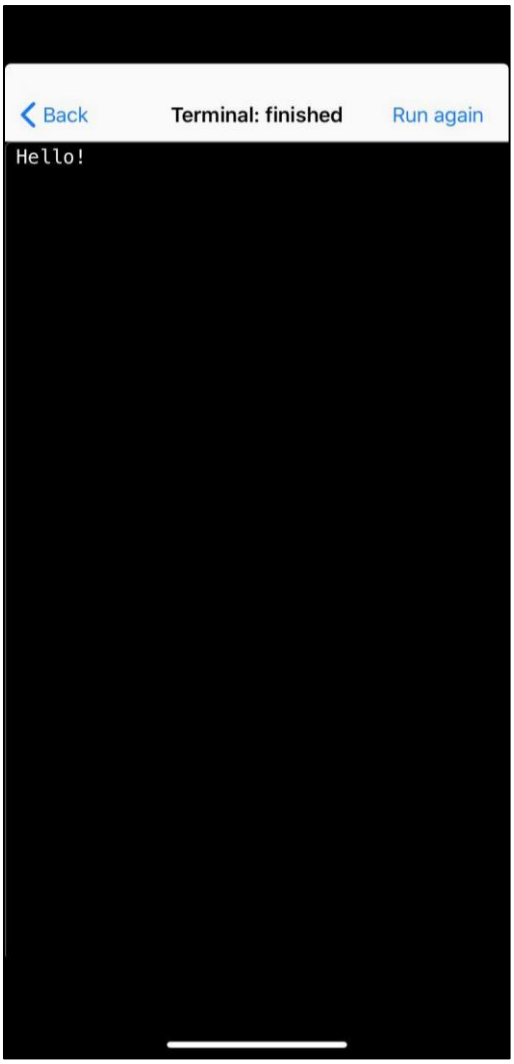


5. Upon choosing Create C++ Project, the text editor will appear and you can now type the C++ code. Please note that for special characters such as quotation marks (“ “) and single quotes (‘ ‘), please use the symbols interface that is provided with the text editor. Otherwise, you might encounter errors.



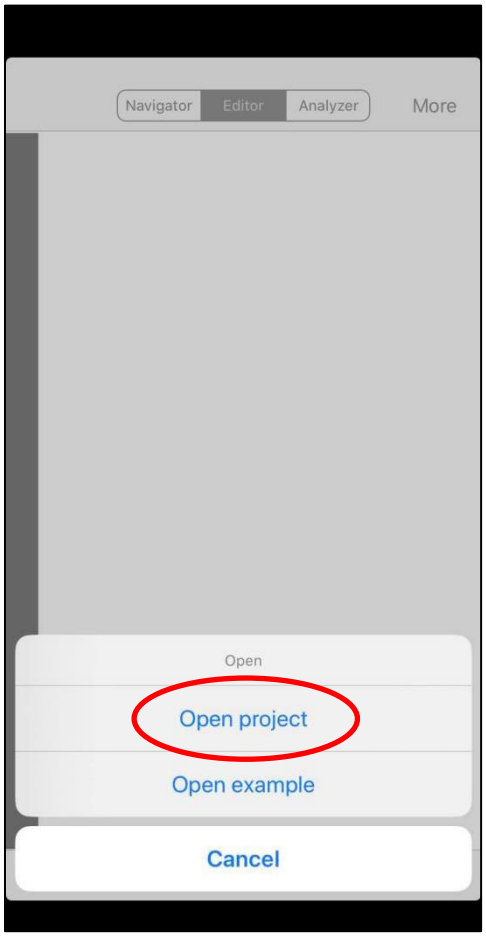
6. Once done, tap on the hide keyboard symbol, then tap ***Save***. You will be prompted if the save is successful. If there are errors, tap on the editor again to enable the keyboard and correct the errors. If there are no errors, tap ***Compile*** and wait for the prompt that the compilation is successful. Then, tap ***Run***. A terminal window will appear with the code's output.

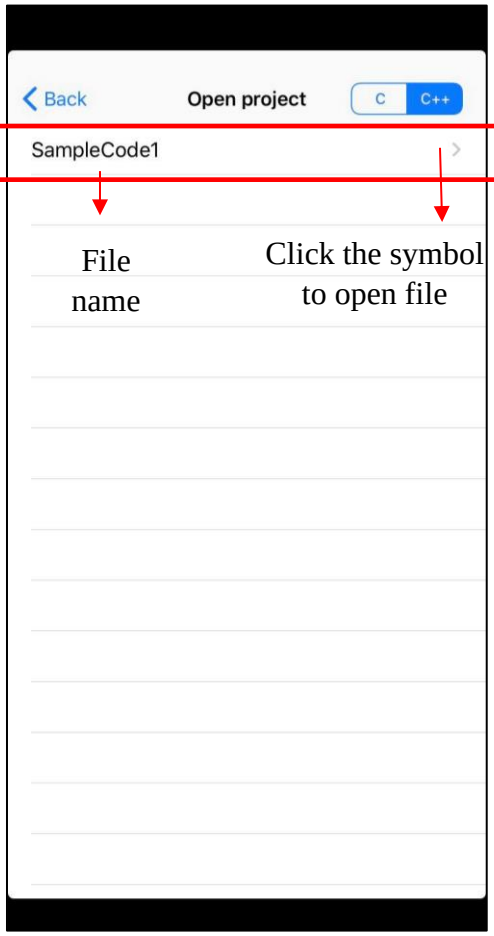




Opening Existing Projects

From the main screen, tap ***Open*** then ***Open project*** then choose ***C++*** on the upper right hand corner of the screen. All existing projects will be listed. Then tap the **>** symbol found at right of the file that you want open. The text editor that contains the code will appear.

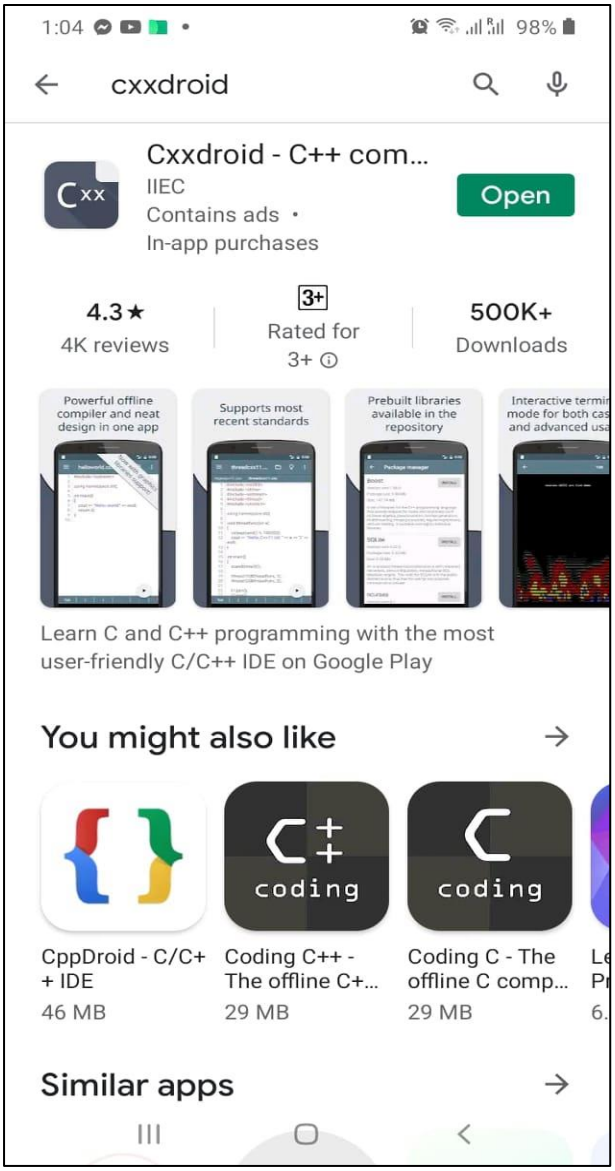


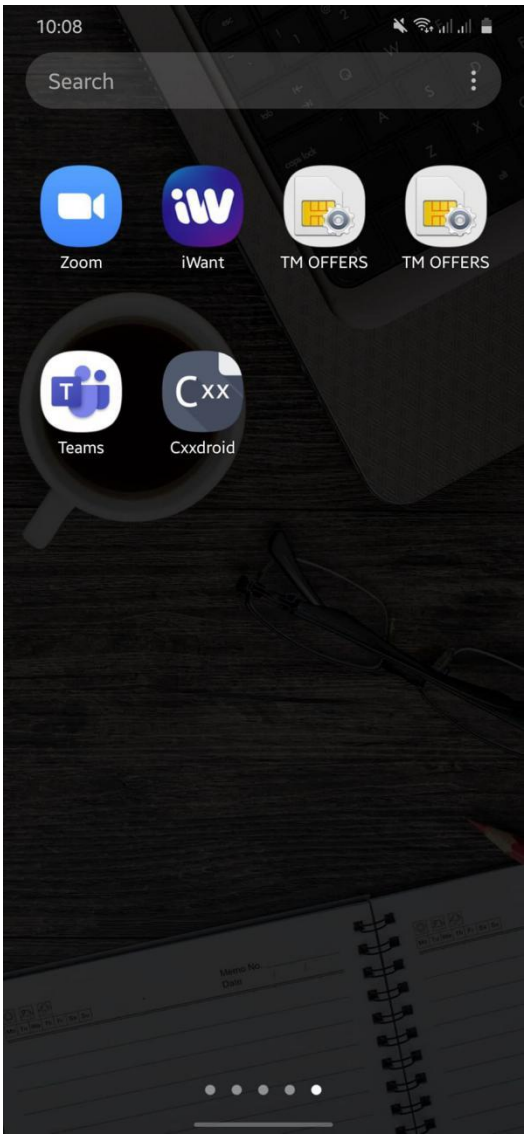


A. APPLICATIONS FOR ANDROID DEVICES

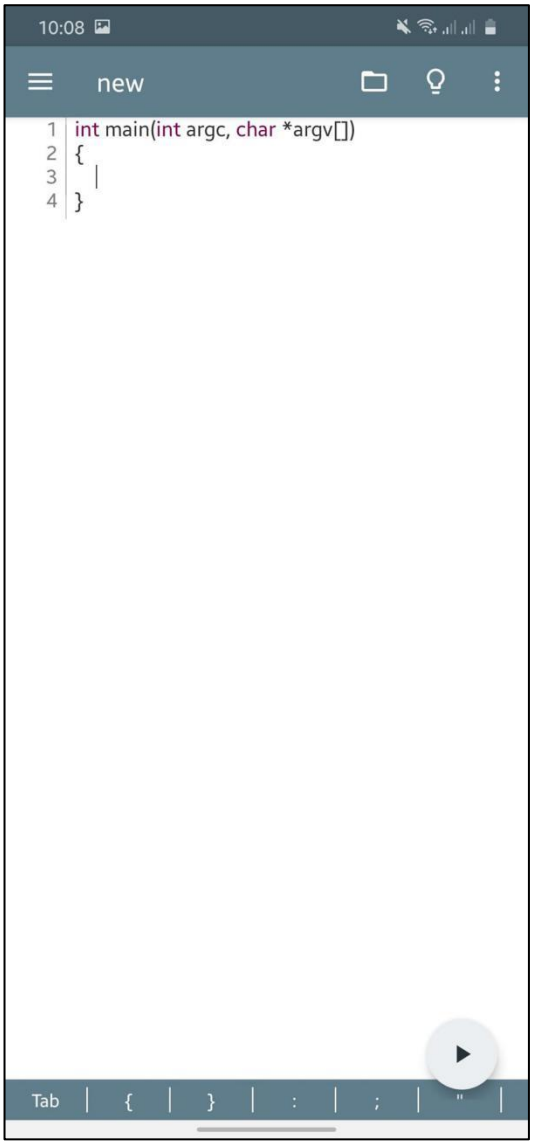
1. Cxxdroid

1. Download and install Cxxdroid from Google Play Store.

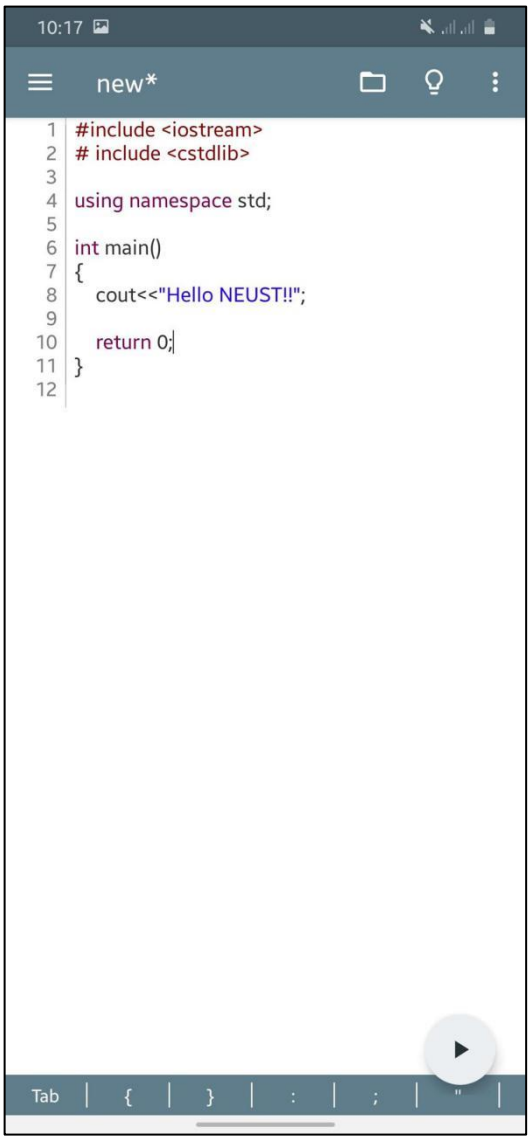


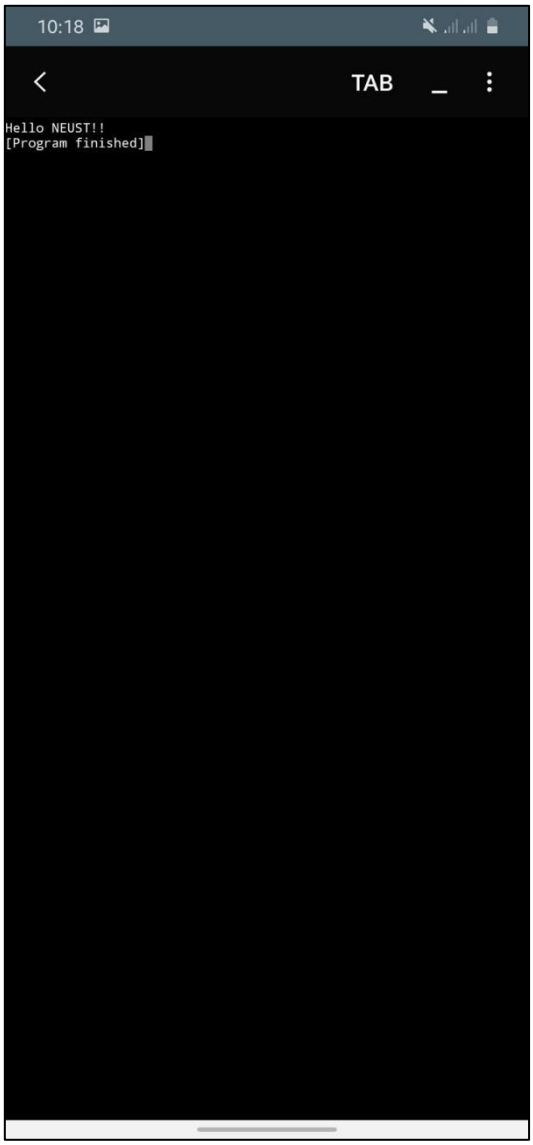


On your android device, locate and click the Cxx icon to start using the app.



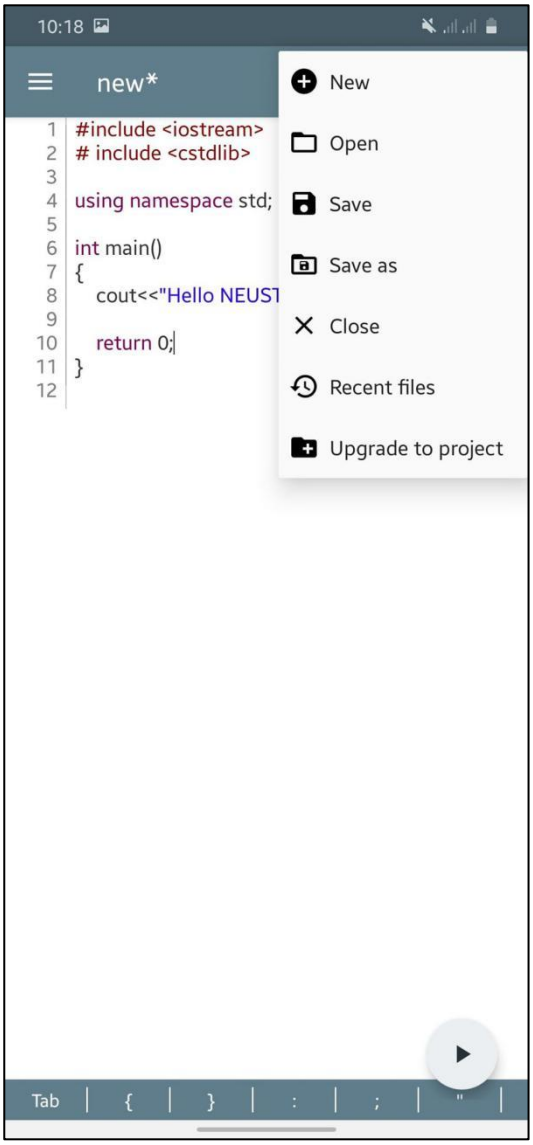
The image above is the default work area of Cxxdroid upon opening the app.



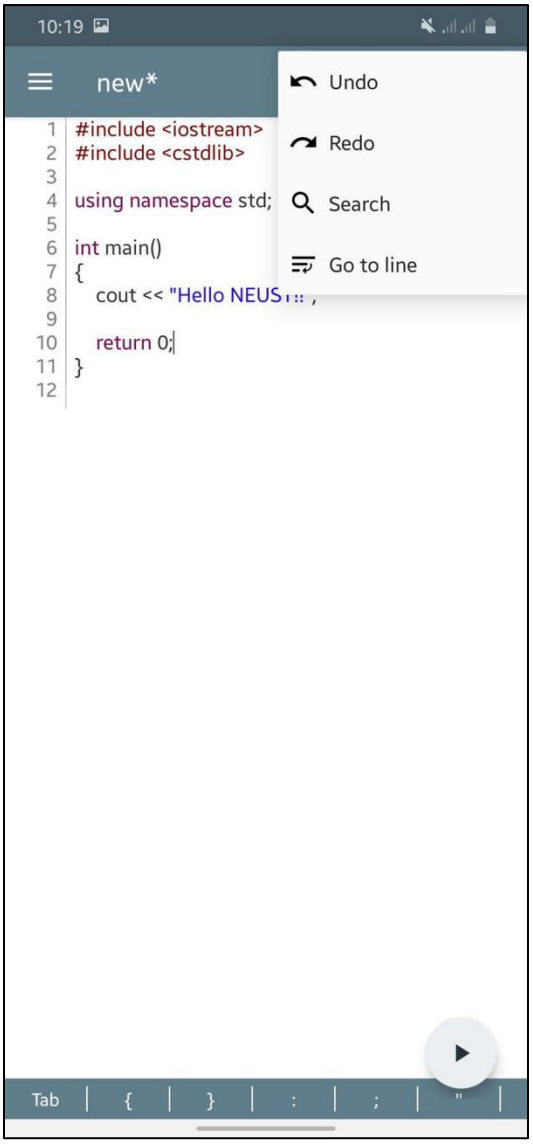


The default and existing codes on work area can be replaced by the desired coding of your own.

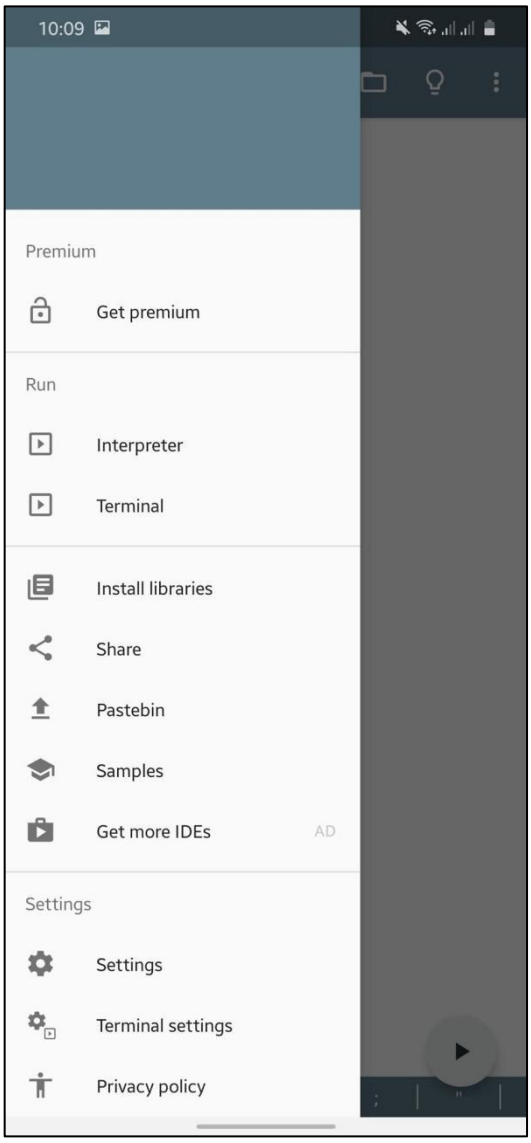
After running or executing the program by pressing the play button/icon on the lower right of the screen



On the upper right corner, click the folder icon to toggle the options on how to manipulate the project.

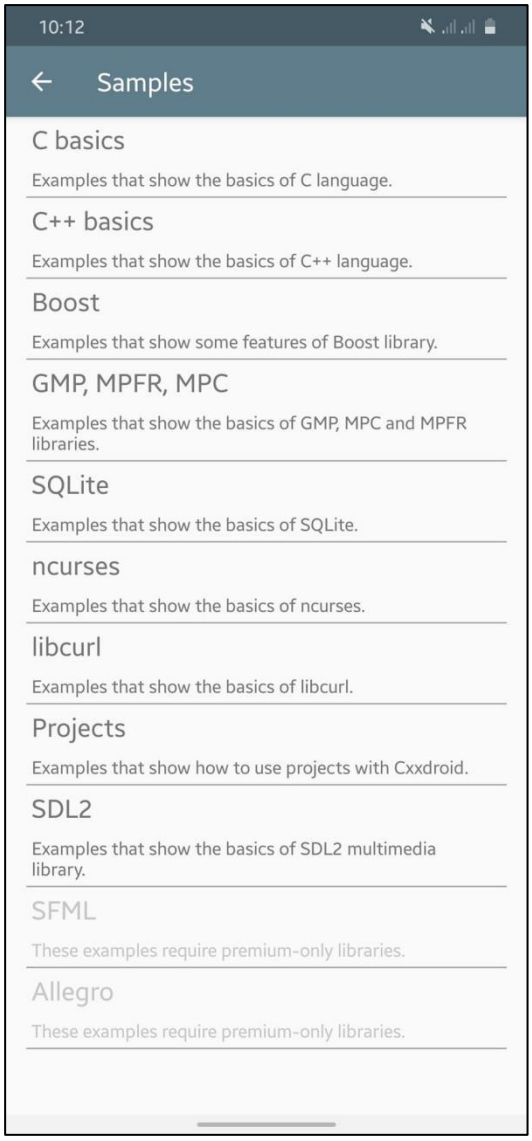


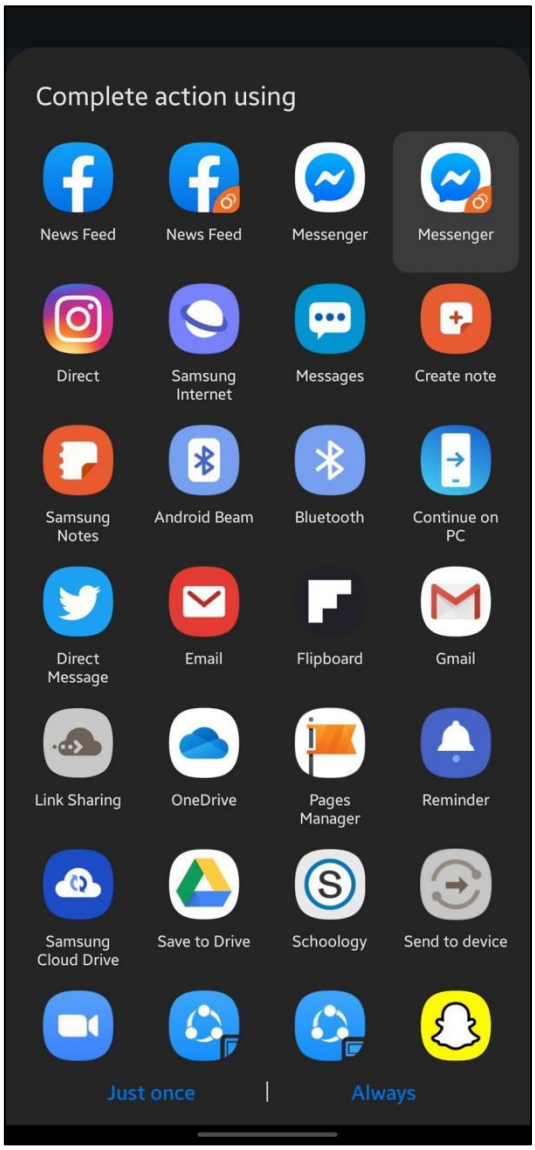
On the upper right corner, click the three dots (Vertical Ellipsis) to open another option set such as undo, redo, search and go to line.



Click the three dashes icon found on the upper left corner of the app to display other features of the app.

Appearance of the app such as themes, font size and font styles are customizable.





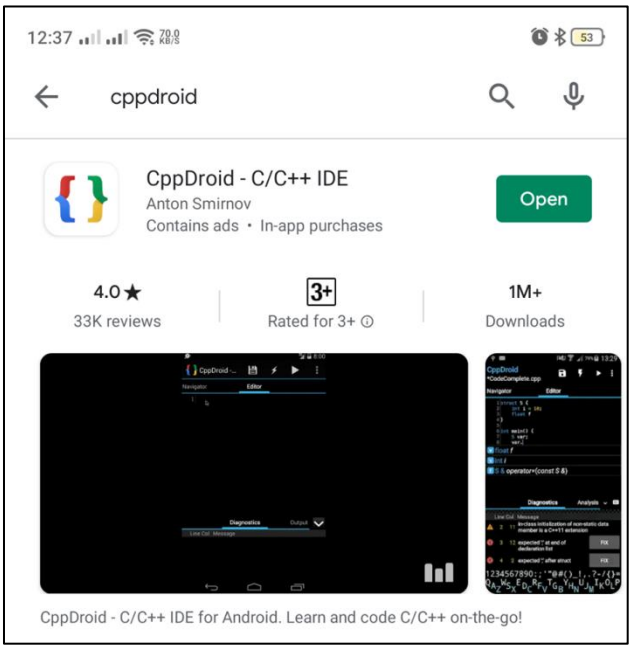
Ready made samples of basic C++ coding are also available in this app.

Sharing capability via text message, Bluetooth, social media platforms, etc. is also available in this app

2. CppDroid

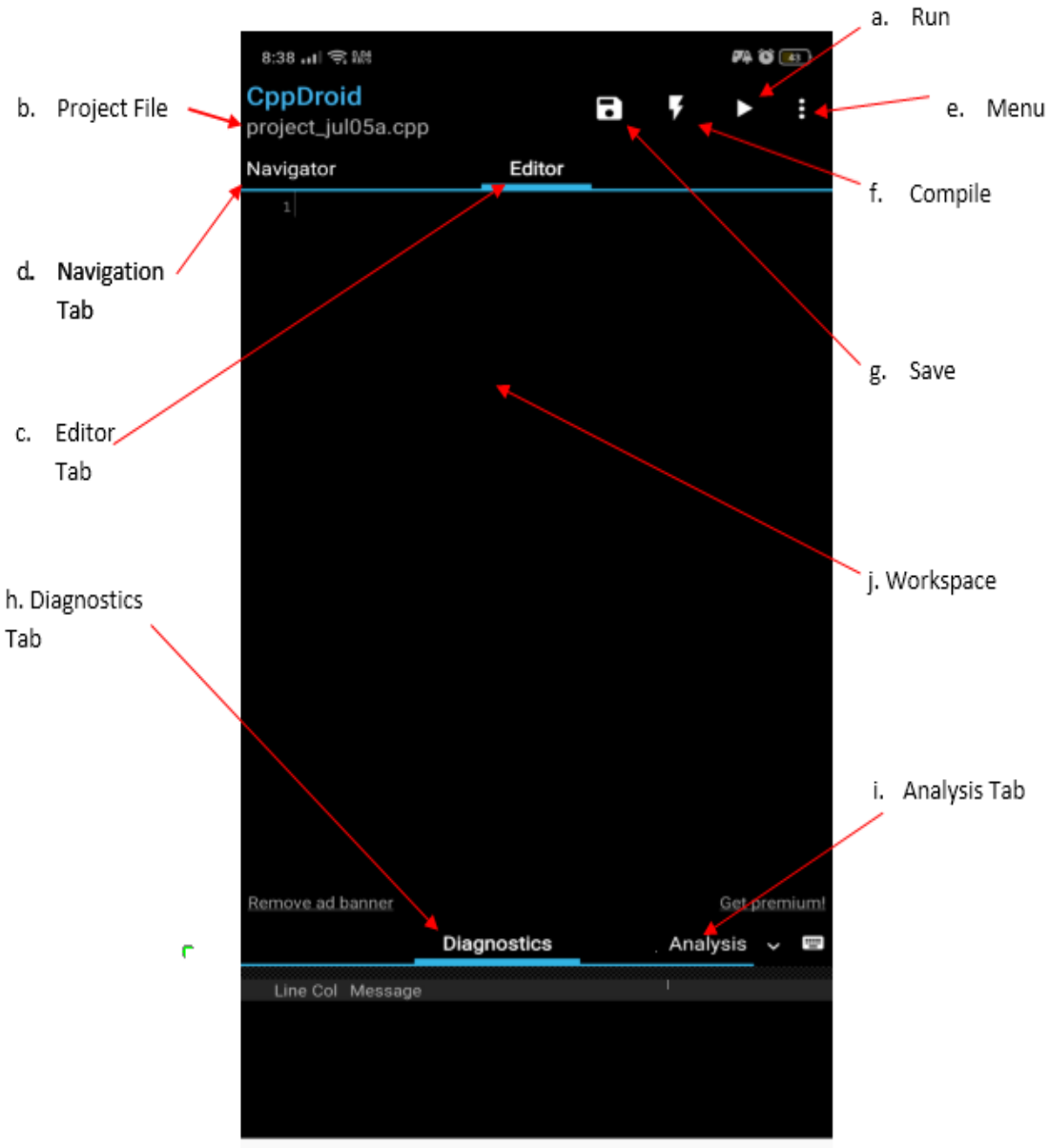
Download and install

Launch the Google Play application in your android device and search for CppDroid.



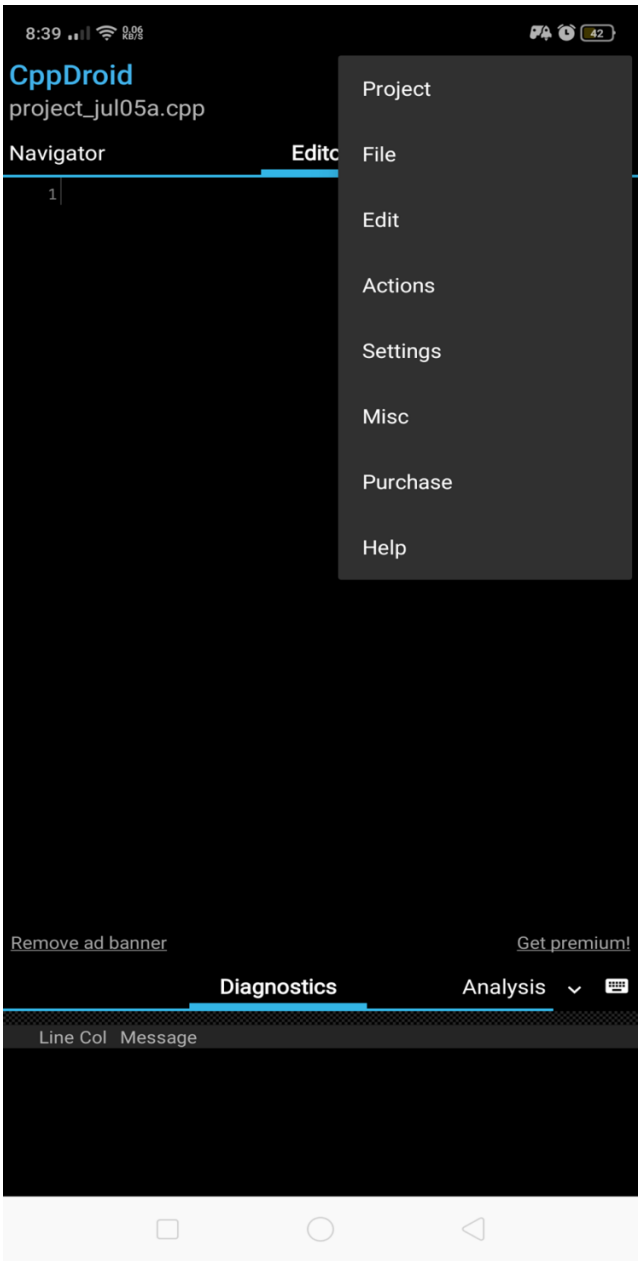
After installing CppDroid, you can see the launch icon of CppDroid in your screen.

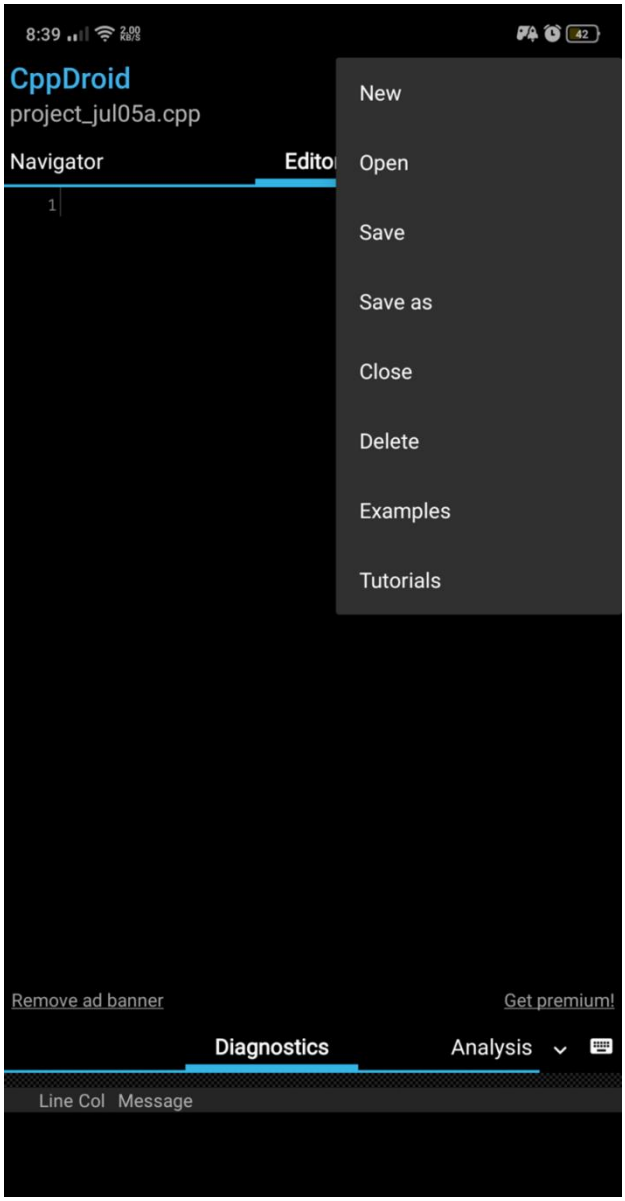
Upon opening the application, the main screen will appear as described in the figure below.



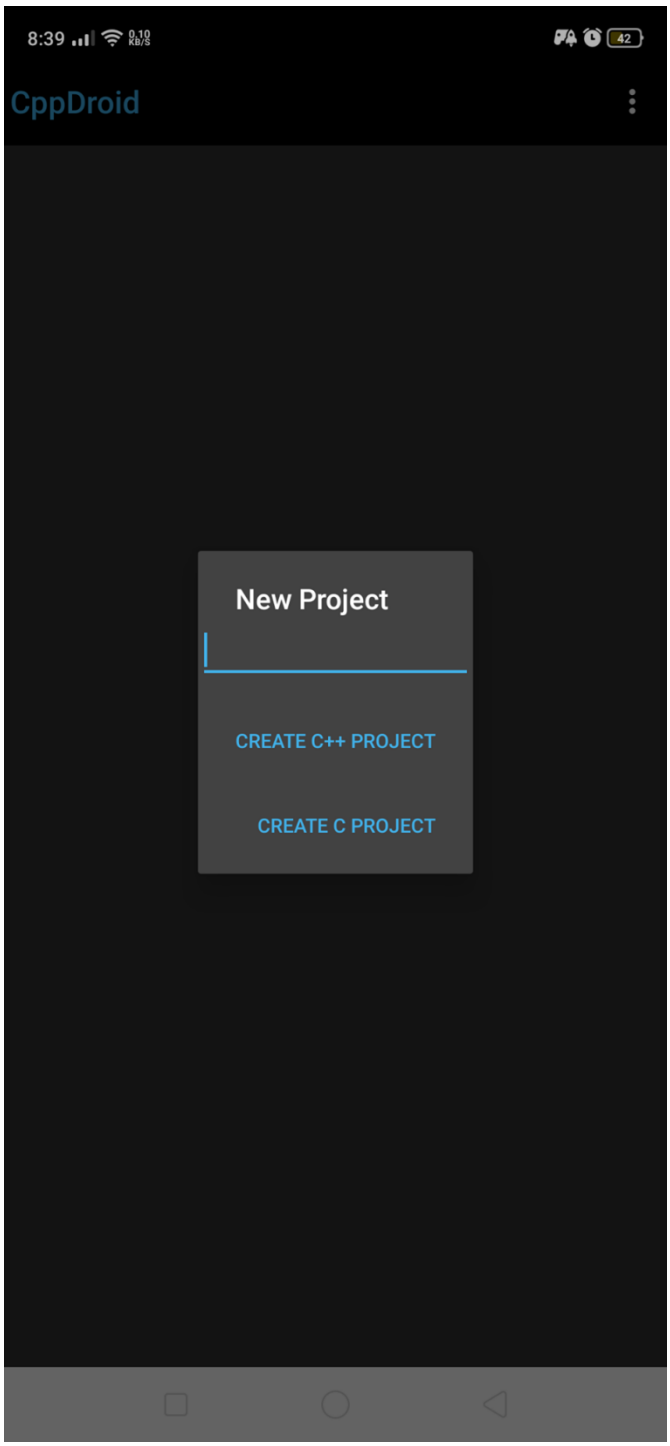
Creating Project

Click the Menu  Icon represented by three dots aligned vertically and select Project -> New.



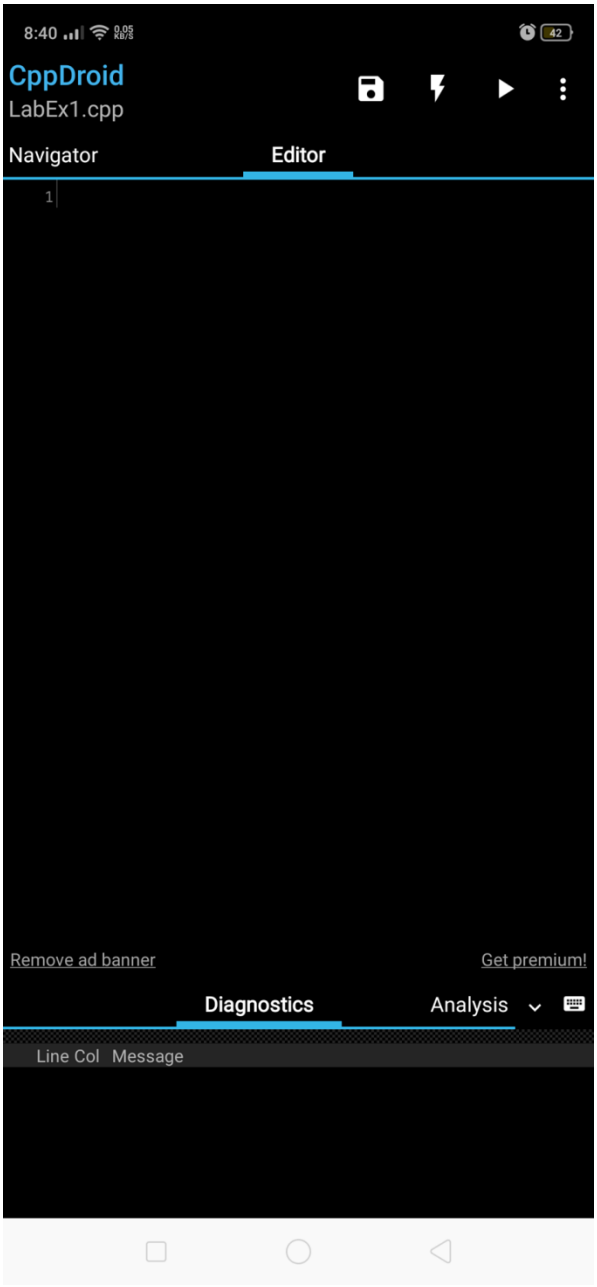


Input project name (ex. LabEx1), then click CREATE C++ PROJECT.

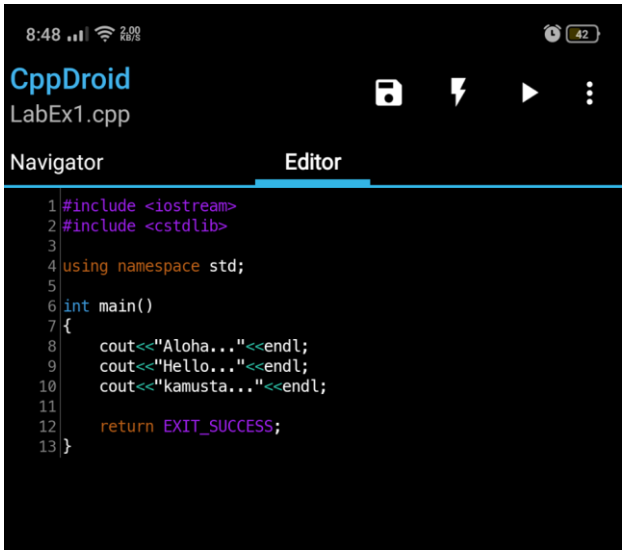


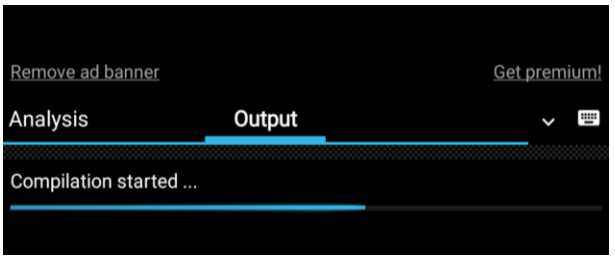
Writing, Compiling and Running a Code

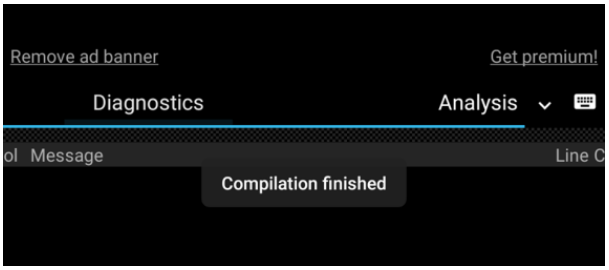
You can now start your coding in the IDE's Edior Tab and write codes in its workspace.



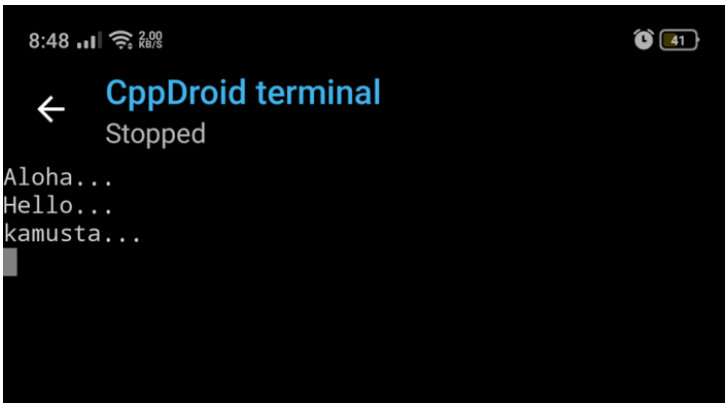
After writing the program, tap on Compile  (Lightning Icon) to compile the written code.







After the compilation is successful, tap on Run and it will execute the program. The figure presents the output preview:

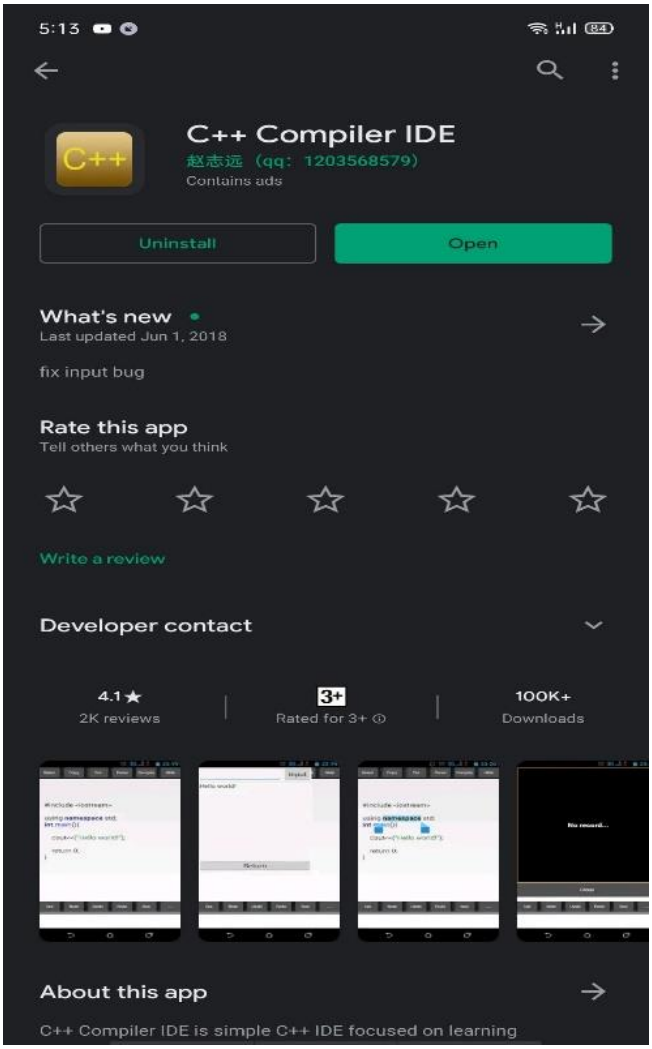


It can be helpful for you to run and test C/C++ on your device at any time.

3. C++ Compiler IDE

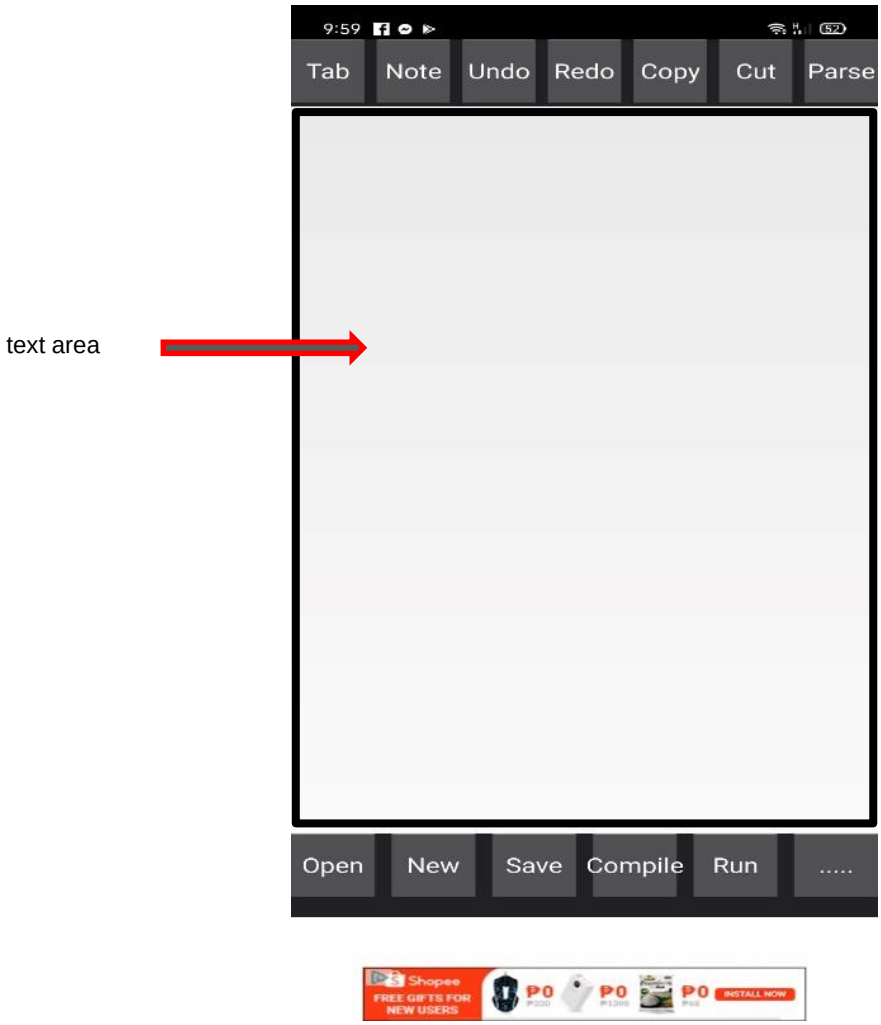
Installing C++ Compiler IDE

From Google Play Store, download and Install **C++ Compiler IDE**.

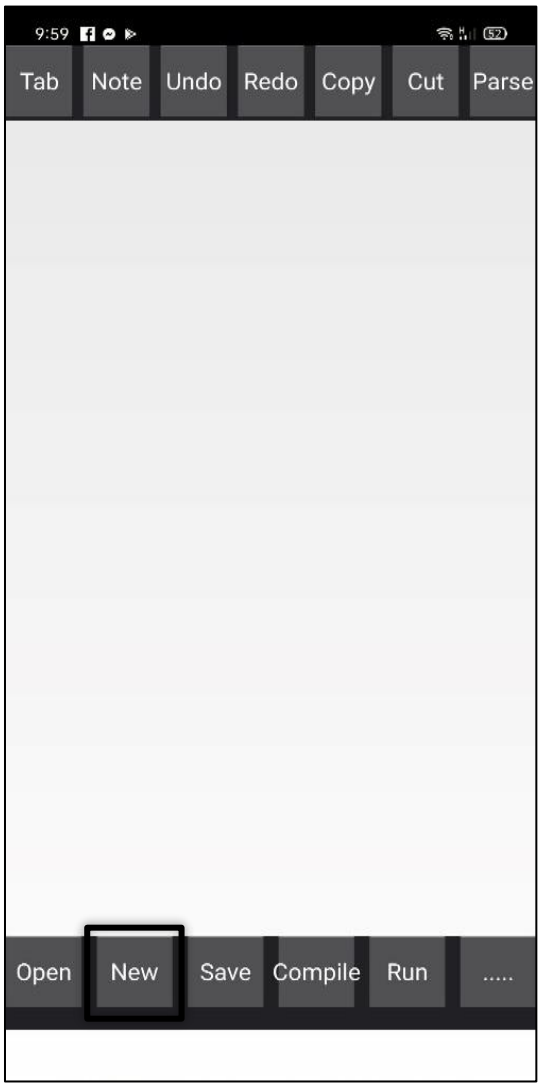


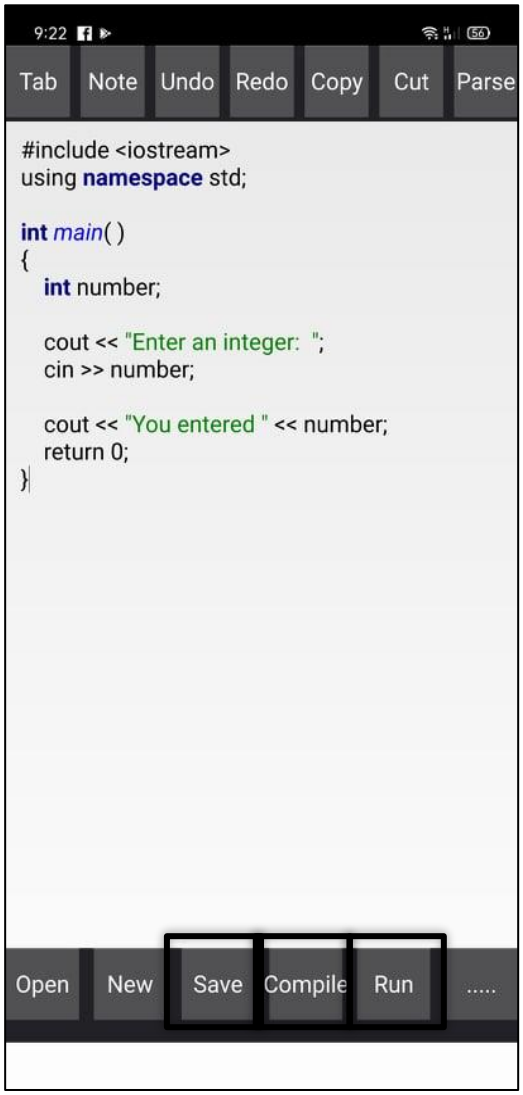
Using C++ Compiler IDE

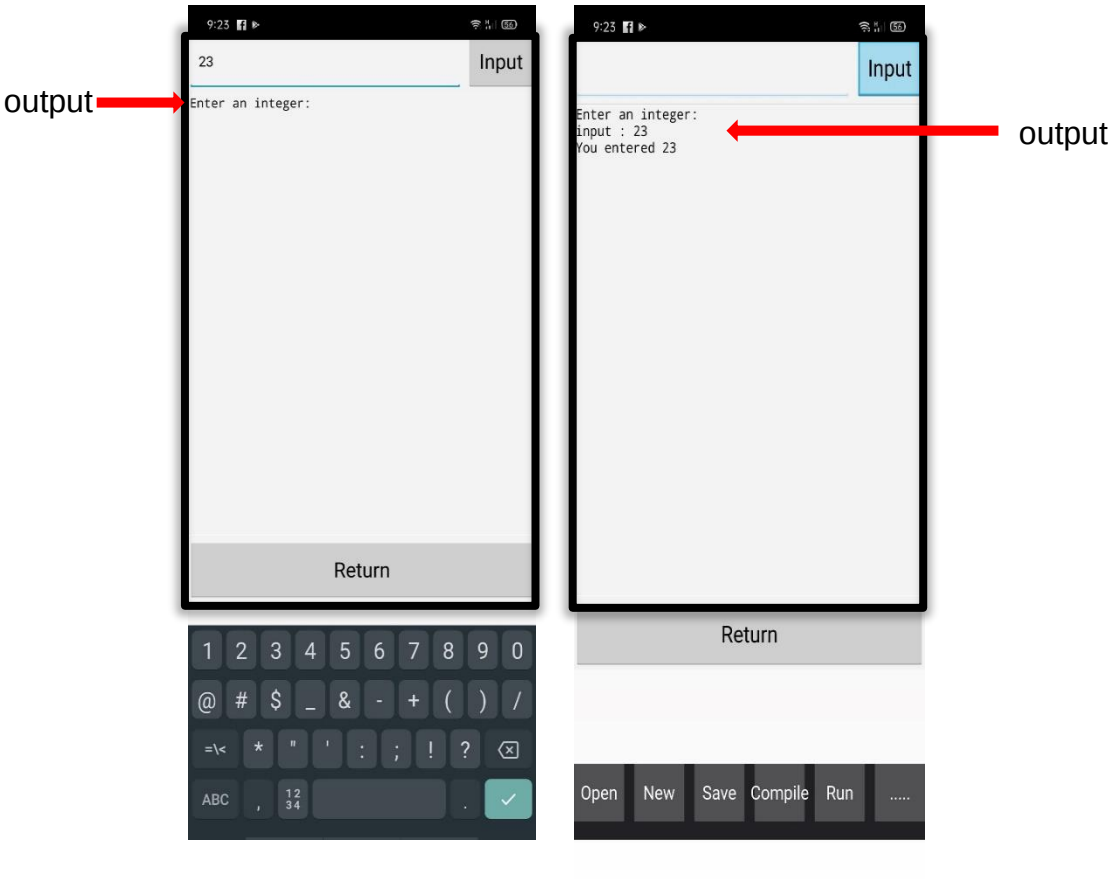
The main screen will be displayed as defined in the following figure when the application is opened.



To create a new project for C++, click “**New**” at the bottom on the phone screen. Now you can start to construct the code. After you have constructed the code, click “**Save**” so that project will be stored. If no errors exist, click “**Compile**” and wait for the prompt to successfully compile. When you click “**Run**” the output of the code will be displayed in the phone’s screen.

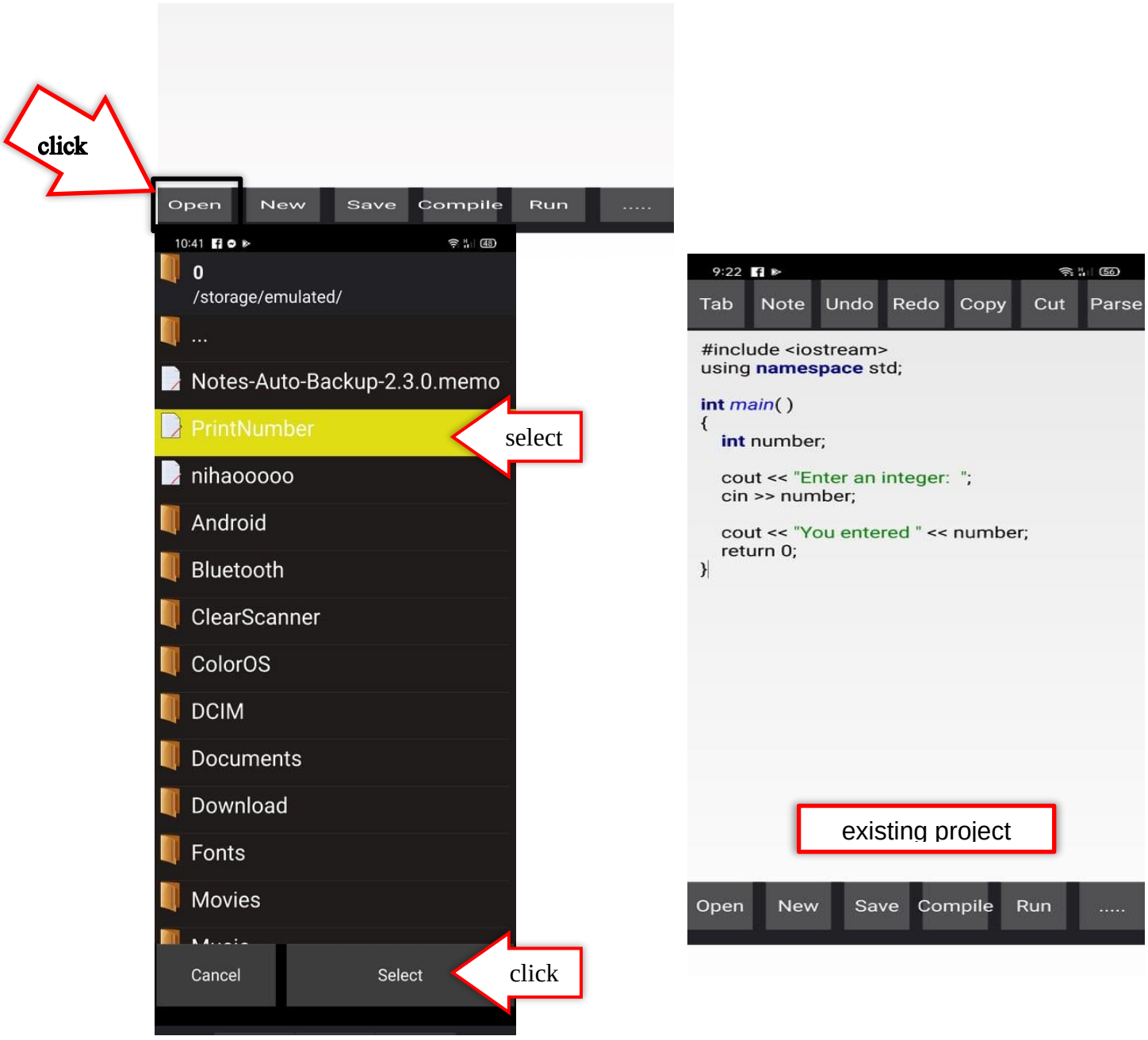





output
23
Input
Enter an integer:
Return
Enter an integer:
input : 23
You entered 23
Input
output
Return
Open
New
Save
Compile
Run
.....

Opening Existing Projects

From the phone screen, click “**Open**”, then **select** the existing file project you want to view.



Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 1

Write a C++ Program that will display your School Name and Campus.
Save it as *LabEx1.cpp*

Sample Output:
Nueva Ecija University of Science and Technology
Sumacab Campus

UNIT 5. IDENTIFIERS, VARIABLES AND CONSTANTS

Learning Objectives

At the end of the unit, I am able to:

1. *define and classify identifiers;*
2. *use and apply appropriate data types for variables and constants;*
3. *name, declare and initialize variables and constants;*
4. *use assignment statements and expressions; and*
5. *apply mathematical operators in building a program.*

A. IDENTIFIERS

Identifiers are names given to ***variables*** and ***constants***. In C++, an ***identifier*** can be a letter or series of letters, numbers or underscore. No other symbol can be used for valid ***identifiers***. The ***identifiers*** should not start with a number. You cannot use ***reserved words*** in C++ as identifiers since they all have specific functions. The list of ***reserved words*** is given below:

asm	**auto**	**bool**	**break**	**case**
catch	**char**	**class**	**const**	**const_cast**
continue	**default**	**delete**	**do**	**double**
dynamic_cast	**else**	**enum**	**explicit**	**export**
extern	**false**	**float**	**friend**	**goto**
if	**inline**	**int**	**long**	**mutable**
namespace	**new**	**operator**	**private**	**protected**
public	**register**	**reinterpret_cast**	**return**	**short**
signed	**sizeof**	**static**	**static_cast**	**struct**
switch	**template**	**this**	**throw**	**typedef**
try	**true**	**type_id**	**typename**	**union**
unsigned	**using**	**virtual**	**void**	**while**
volatile	**wchar_t**	**and**	**and_eq**	**bitand**
bitor	**compl**	**not**	**not_eq**	**or**
or_eq	**xor**	**xor_eq**		

C++ is a case sensitive programming language, therefore, identifier rate, Rate and RATE are not the same identifiers.

EXAMPLES OF VALID IDENTIFIERS

Age redRibbon year123 ako_ikaw_tayo ARVEE

B. DATA TYPES

Variables and constants use memory locations to store data and the size of which will be depending on the data type that these two identifiers will hold. This is the reason why you cannot use variables and constants unless you have declared them first. The primitive data types which are built-in in the C++ programming language are listed below with their corresponding size.

DATA TYPE	MEMORY SPACE ALLOCATION	DESCRIPTION
short	2 bytes	A whole number within the range of -32,768 to 32,768
int	2 bytes	A whole number within the range of -32,768 to 32,768
long	4 bytes	A whole number within the range of -2,147,483,648 to 2,147,483,648
float	4 bytes	A decimal number within the range of -3.4×10^{38} to 3.4×10^{38}
double	8 bytes	A decimal number within the range of -1.7×10^{308} to 1.7×10^{308}
char	1 byte	A letter, number or symbol
bool	1 byte	true or false
string	1 byte per character	0 or more characters

The ***short, long*** and the ***int*** data type will store integers or ***whole*** numbers with no decimal place values. The ***float*** and the ***double*** data type will hold ***real*** numbers or numbers with ***decimal*** parts. You cannot use the value of a ***char*** data type in mathematical and logical computations. Character values are enclosed with a pair of ***apostrophes***(‘ ‘). The ***bool*** data type will store the values true or false. The ***string*** data type can hold values which are series of letters, numbers and symbols, or it can hold a ***null string*** which is a space inside the quote-unquote symbols **(“ “).** The value of a string data type is enclosed with ***quote-unquote***.

C. VARIABLES

Variables are names given to memory locations that you will be utilizing in order for you to store data. You can change the value of a variable as you go along in your coding but as mentioned earlier, a variable should be declared first at the start of the program before you can use it. As for conventionality, if a variable name you want to use is consisted of two or more words, use small letters to the first word then capitalize the first letter/s of the succeeding word/s. The variable will appear like it has ***humps*** thus, it is said to be using ***camel case*** format.

Examples:

interestRate numberOfHoursWorked taxRate

To declare a variable (so that the computer can allocate memory space for it), use the following syntax:

data_type variable_name;

Examples::

int x; /* Declares **x** as a variable which can only store whole numbers */

float interest_rate; //Declares ***interest_rate*** to be a ***float*** variable

char alias; /* Declares ***alias*** as a variable which can only hold a single character (number, letter or symbol)*/

If you are using variables with the same data type, you can declare individually or together in one declaration statement:

Example:

short rv; **short df;** **short dx;**	**short rv, df, dx;**

In declaring variables, bear in mind that it would store values of certain entities, therefore, give variable names in relation to the entity it will represent. For example, if you want storage for the rate of interest, do not use ***hourlyRate*** as variable. Instead you can use ***interestRate***, so that if for some instance another person other than you will look at your program, it will not be difficult for him to comprehend your code.

INITIALIZATION OF VARIABLES

You can also declare a variable and at the same time put an initial value to it. To do that, follow this syntax:

data_type variable_name = initial_value;

Examples:

```
int number = 10;
double rating = 7.53;
char x = 'R';
```

D. CONSTANTS

A ***constant*** is an identifier which can store unchanging values. The same thing as in variables, constants should be declared first at the beginning of your program before you can utilize them. You can declare constants before the main method of a C++ program like this:

```
#define PI 3.1416
#define LETTER 'a'
```

If you want to declare a constant inside the main method, use the following syntax:

const data_type CONSTANT_NAME = value;

Examples:

```
const double PI = 3.1416;
const char LETTER = 'a';
```

To easily distinguished constant from variable, use capital letters when naming constants. If two or more words are used, separate the words with underscores(_).

E. ASSIGNMENT STATEMENTS AND ASSIGNMENT EXPRESSIONS

You can also assign values to your variables using the ***assignment operator*** (=).

Example No. 1: SYNTAX: variable = value;

```
a = 5;
r = 20;
v = 'c';
e = 3.28;
e = 0.1;
```

Example No. 2: SYNTAX: variable = variable;

```
t = a;
age = num;
nextRate = oldRate;
```

Example No. 3: SYNTAX: variable = expression;

```
x = 5 * (3 / 2) + 3 * 2;
area = PI * radius * radius;
```

Notice that in the assignment statement, the variable is located at the left of the assignment operator. The right side, however, can be a value, another variable or an expression. An expression in C++ is a mathematical expression which is consists of values, variables and operators which evaluates to a single value.

INCREMENT/DECREMENT

A variable can have an increment or decrement. Increment is the process of increasing the value of a variable by addition. The usual increment is by 1:

```
r = r + 1;
```

The shortcut form of the above statement is as follows:

```
r++;
```

On the other hand, decrement is the process of decreasing the value of a variable. Just like in the increment, the usual value of decrement is 1. For example:

```
w = w – 1;
```

The decrement shortcut are like these:

w--;

Increment and decrement can also use other numbers aside from **1**. If variable **x** is incremented by **3**, the corresponding statement will be like this.

x = x + 3;

or

x += 3;

Consequently, if variable **y** is to be decremented by **5**, the statement will be:

y = y – 5;

or

y -= 5;

F. MATHEMATICAL OPERATORS

C++ also uses mathematical operators the list of which appears in the table below:

SYMBOL	NAME	OPERATION	FUNCTION
+	plus	Addition	Used in getting the sum of the numbers
-	minus	Subtraction	Used in getting the difference of the numbers
*	asterisk	Multiplication	Used in getting the product of the numbers
/	slash	Integer division	Used in getting the quotient of the numbers
%	modulo	Integer division	Used in getting the remainder of the division of two numbers

ORDER OF PRECEDENCE

Just like in College Algebra, the ***mathematical operators*** in C++ have an ***order of precedence***. The ***Order of Precedence*** tells which operator should be executed first. The following table lists the mathematical operators and their levels in the Order of Precedence.

LEVEL	OPERATORS
I	**( )**
II	*** / %**
III	**+ -**

The computer will first evaluate the ***level I operator*** which is a pair of parentheses. After all the operators inside the parentheses were evaluated, the level II operators will be processed next, then followed by the level III operators.

If the are operators under the same level, evaluate first the operator on the left of the expression until all of the same-level operators are evaluated.

EXAMPLES:

1. x = 4324 % 7 – 2 * (15 – **30 / 6**) + 84 / 3
   x = 4324 % 7 – 2 * (**15 – 5**) + 84 / 3
   x = **4324 % 7** – 2 * 10 + 84 / 3
   x = 5 – **2 * 10** + 84 / 3
   x = 5 – 20 + **84 / 3**
   x = **5 – 20** + 28
   x = **-15 + 28**
   x = **13**

2. x = **(28 – 4)** * 3 / 8 – 82 % 7
   x = **24 * 3** / 8 – 82 % 7
   x = **72 / 8** – 82 % 7
   x = 9 – **82 % 7**
   x = **9 - 5**
   x = **4**

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 12

I. BOX ALL THE VALID IDENTIFIERS.

areaofacircle	1ABCD	PI	age-18
x	C	char	true_blue
interest rate	RADIUS	friend	RATE_HOUR
MAX VALUE	virtual	2013	false

II. WRITE THE VALID DECLARATION STATEMENTS OF THE FOLLOWING.

1. Variable ***x*** with initial value of 3.75.

_____

2. Constant ***RANGE*** with value of 200.

_____

3. Variable ***age*** with will hold a whole number.

_____

4. Variable ***y*** which will hold a character value.

_____

5. Constant ***LIMIT*** with a value of Z.

_____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 13

EVALUATE THE FOLLOWING:

1. **a = (2540 – 13) / 41 * 6 – 450 + 182 % 14**

2. **r = 25 + 500 / 4 – 22 * 15 % 23**

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 14

MAKE C++ STATEMENTS THAT WILL DO THE FOLLOWING:

1. Assign the value of variable ***m*** to variable ***n***;

   _____

2. Assign the value you will get if you add 5 and variable ***a*** to variable ***b***.

   _____

3. Assign the value 436 to variable ***x***.

   _____

4. Assign the value of variable ***rate*** to variable ***due***.

   _____

5. Get the product of the values of variables ***x*** and ***y*** and assign it to variable ***z***.

   _____

DECLARE THE FOLLOWING AND MAKE THE CORRESPONDING ASSIGNMENT STATEMENTS:

1. Variable ***num1*** with a value of 5.55

   _____

   _____

2. Constant ***name*** with value DERON XYRUS

   _____

   _____

3. Variable ***interest rate*** with the value of 7.5%

   _____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 15

Draw a flowchart and make a C++ program that will compute and display the area of a circle. Assign 50 to the radius and declare pi as a constant with a value of 3.1416. The area of a circle is computed by multiplying pi to the square of the radius.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 16

Draw a flowchart and make a C++ Program that will compute and display the sum and the average of 615, 512 and 48.

UNIT 6. INPUT/ OUTPUT STATEMENTS

Learning Objectives

At the end of the unit, I am able to:

1. *create a C++ program using input/output statements.*

In C++, a ***stream*** is referred to as a series of letters, characters or symbols. The stream is handled by objects which are inside the ***iostream*** library. That is why, you use ***#include <iostream>*** at the start of a C++ program. This ***#include*** statement will let you use the input and output stream commands of C++.

A. THE OUTPUT STATEMENT

The ***output*** statement in C++ is consist of the reserved word ***cout*** (pronounced as ***see out***), followed by the ***insertion operator*** (**<<**), then a variable or a value or a literal string then a semicolon. It can also be followed by a series of variables, values and literal strings, each of which is separated by the insertion operator. The syntax for a ***cout*** statement are as follows:

```
cout<<value;
cout<<variable;
cout<<"Literal String";
```

COMBINATION:

```
cout<<value<<variable<<"Literal string";
```

EXAMPLE:

```
cout<<10;
cout<<age;
cout<<"Choobahnez";

cout<<"The sum is "<<sum<<endl;
```

The command ***endl*** (pronounced as ***endline***) will print a line break on the output screen. This is just like pressing the ***ENTER*** key. Another way to print a line break is by using the ***quote-backslash-n-unquote ("\n")*** command. \n will not function if it is not inside the quote-unquote symbols.

B. STRING MANIPULATION

One of the most useful data types supplied in the C++ libraries is the string. A string is a variable that stores a sequence of letters or other characters, such as "**Hello**" or "**December 31st is my birthday!**". Just like the other data types, to create a string we first declare it, then we can store a value in it.

string testString;

testString = "This is a string.";

We can combine these two statements into one line:

string testString = "This is a string.";

Often, we use strings as output, and **cout** works exactly like one would expect:

cout << testString << endl;

will print the same result as

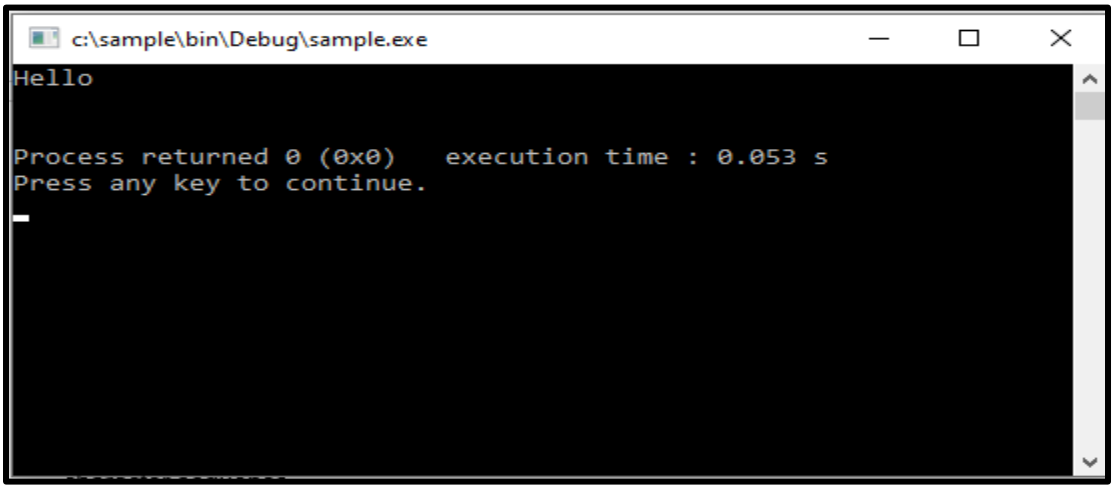
cout << "This is a string." << endl;

In order to use the string data type, the C++ string header **<string>** must be included at the top of the program. Thus, you would have the following **#include**'s in your program in order to use the **string** type.

Source Code:

```
1   #include <iostream>
2   #include <cstdlib>
3   #include <string>
4
5   using namespace std;
6
7   int main()
8   {
9       string greeting = "Hello";
10
11      cout<<greeting<<endl<<endl;
12
13      return EXIT_SUCCESS;
14  }
15
```

Output:



C++ strings are designed to behave like ordinary primitive types with regard to assignment. Assigning one string to another makes a deep copy of the character sequence.

```
string str1 = "hello";
string str2 = str1; // makes a new copy
```

C. THE INPUT STATEMENT

There are times when a program requires user input in order to have data to manipulate and produce the output it desired. The C++ input stream object is ***cin*** (***see in***). The ***cin*** object is followed by the ***extraction operator (>>)*** and the ***variable*** where you want to store the value to be extracted by the cin object. The extraction operator will stop removing ***cin*** object characters once it encounters a linebreak (when you press the ***ENTER*** key) or a space(when you press the ***spacebar***) or a tab(when you press the ***TAB*** key). These are called the ***whitespaces*** in programming. The syntax for the ***cin*** statement is:

cin>>variable;

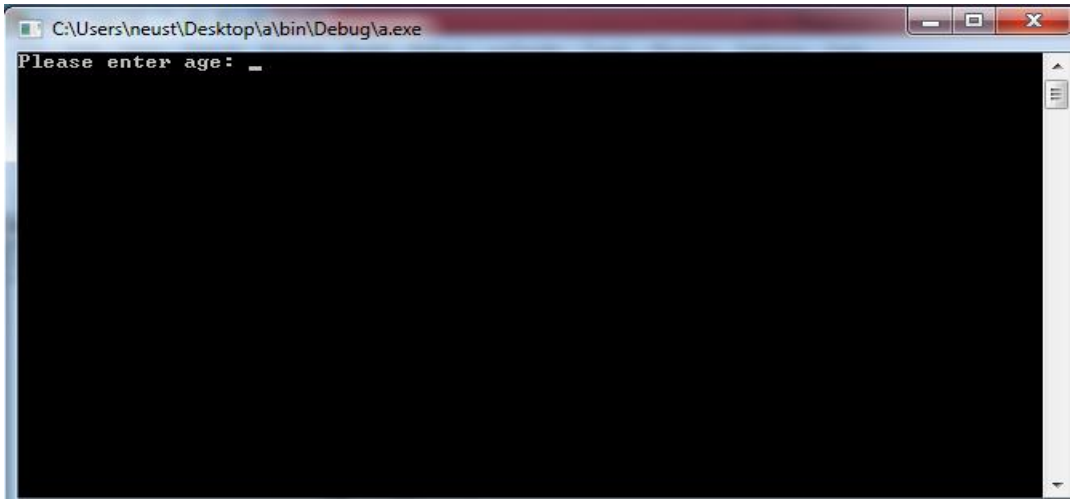
Before using the ***cin*** statement; you should first prompt the user as to what data should be entered. For example, you want the user to key-in the age of a person. In the program, you will type:

cin>>age;

As seen in the output screen, only a blinking cursor will be displayed since it is waiting for the user to type the age. The problem with this lies on the fact that if you are not the programmer of this code, you will not have any idea as to what it is you are supposed to input. The solution for this will be to prompt the user by using the ***cout*** object like this:

```
cout<<"Please enter age: ";      //prompts the user to enter age
cin>>age;                        /*gets the age given by the user and
                                   assign it to variable age*/
```

Your sample screen will be like this:



So now, even if you are not the programmer of the code, you will know from the prompt that you have to input an integer value which corresponds to age.

BASIC STRING OPERATIONS

1. Counting the number of characters in a string

The **length** method returns the number of characters in a string, including spaces and punctuation. Like many of the string operations, **length** is a *member function*, and we invoke member functions using *dot notation*. The string that is the receiver is to the left of the dot, the member function we are invoking is to the right, (e.g. **str.length()**). In such an expression, we are requesting the length from the variable **str**.

Example Program 1:

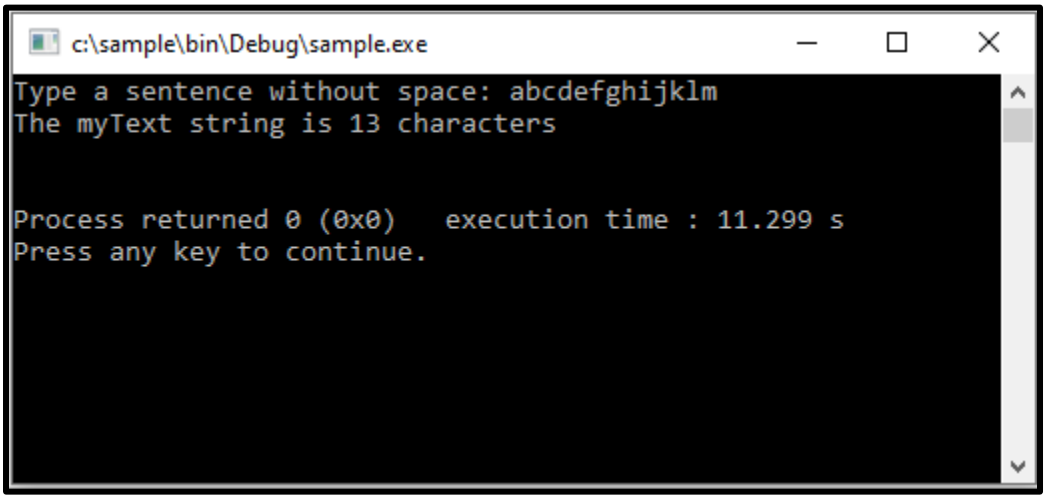
```
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;

int main()
{
    string small, large;

    small = "Iamshort";
    large = "My friendIamalongandelaboratestringindeed";

    cout<<"The short string is "<<small.length()<<" charaters"<<endl;
    cout<<"The long string is "<<large.length()<<" characters"<<endl<<endl;

    return EXIT_SUCCESS;
}
```

```
c:\sample\bin\Debug\sample.exe
The short string is 8 charaters
The long string is 40 characters


Process returned 0 (0x0)   execution time : 0.013 s
Press any key to continue.
```

Example Program 2:

```
1   #include <iostream>
2   #include <cstdlib>
3   #include <string>
4
5   using namespace std;
6
7   int main()
8   {
9       string myText;
10
11      cout<<"Type a sentence without space: ";
12      cin>>myText;
13
14      cout<<"The myText string is "<<myText.length()<<" characters"<<endl<<endl;
15
16      return EXIT_SUCCESS;
17  }
18
```

In the example program above, try to type **abcdefghijklm**. The output will be:



Tip: You might see some C++ programs that use the **size()** function to get the length of a string. This is just an alias of l**ength()**. It is completely up to you if you want to use **length()** or **size()**.

2. String Concatenation

Suppose you have two strings, str1 and str2 and you want to create a new string by combining them. This is called **concatenation.**

Conveniently, you can just write str1 + str2, and you'll get the result you'd expect.

Example Program 1:

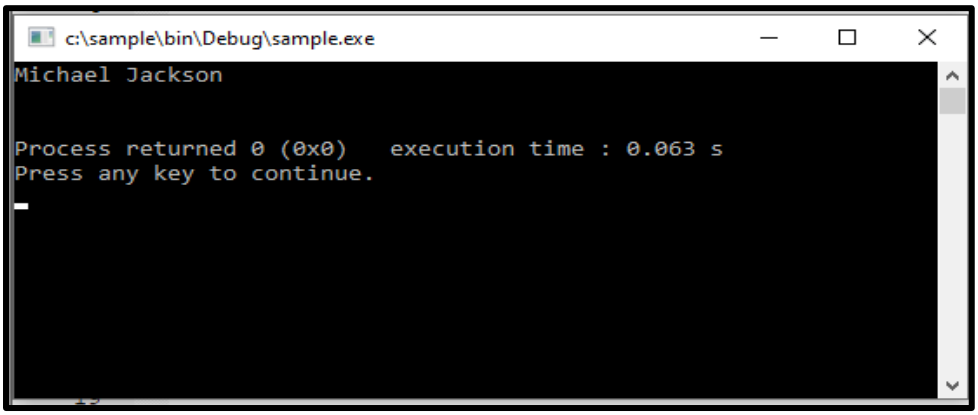
```
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;

int main()
{
    string str1, str2, str3;

    str1 = "Michael ";
    str2 = "Jackson";
    str3 = str1 + str2;

    cout<<str3<<endl<<endl;

    return EXIT_SUCCESS;
}
```

```
c:\sample\bin\Debug\sample.exe
Michael Jackson


Process returned 0 (0x0)   execution time : 0.063 s
Press any key to continue.
```

In the example above, we added a space after **str1** to create a space between Michael and Jackson on output. However, you could also add a space using quotes (" ") or apostrophes (' ').

Example Program 2:

```
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;

int main()
{
    string str1, str2, str3;

    str1 = "Michael";
    str2 = "Jackson";
    str3 = str1 + " " + str2;

    cout<<str3<<endl<<endl;

    return EXIT_SUCCESS;
}
```



Similarly, if you want to append to the end of string, you can use the **+=** operator. You can append either another string or a single character to the end of a string.

Example Program 3:

```
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;

int main()
{
    string str1, str2, str3;

    str1 = "Michael ";
    str2 = "Jackson";
    str3 = str1 + str2;
    str3 += " Jr";
    str3 += ".";

    cout<<str3<<endl<<endl;

    return EXIT_SUCCESS;
}
```

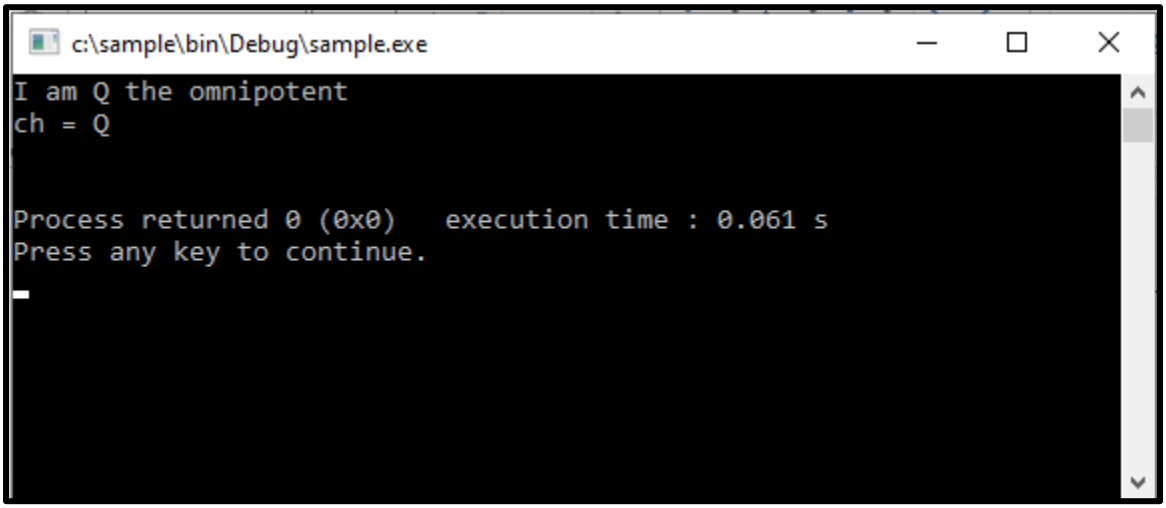
```
c:\sample\bin\Debug\sample.exe
Michael Jackson Jr.


Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```

3. Accessing individual characters

Using square brackets, you can access individual characters within a string as if it is a char array. Positions within a string str are numbered from 0 through str.length() - 1. You can read and write to characters within a string using [ ]. Take note that string indeces start with 0 therefore [0] is the first character, [1] is the second character and so on.

Example Program:

```cpp
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;

int main()
{
    string test;
    char ch;

    test = "I am Q the omnipot3nt";
    ch = test[5];     //ch is 'Q'
    test[18] = 'e';   //we correct misspelling  of omnipotent

    cout<<test<<endl;
    cout<<"ch = "<<ch<<endl<<endl;

    return EXIT_SUCCESS;
}
```



Be careful not to access positions outside the bounds of the string. The square bracket operator is not range-checked and thus reading from or writing to an out-of-bounds index tends to produce difficult-to-track-down errors.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 17

MAKE C++ STATEMENTS THAT WILL DO THE FOLLOWING:

1. Print the sentence, “It Is A Good Day”.

   _____

2. Input a number using variable, num.

   _____

   _____

3. Input a character and store it to variable, oneSymbol.

   _____

   _____

4. Display the sum of two numbers giving it a proper description.

   _____

5. Input the first letter of your name.

   _____

   _____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 2

Open the code for *LabEx1.cpp* and modify it by adding syntax that will ask the user to:

a. Input last name;
b. Input first name;
c. Input middle initial;
d. Input address;
e. Input contact number and
f. Input birth year; and
g. Compute for the age based on the birth year; and
h. Display all the input data.

Sample Output:

Nueva Ecija University of Science and Technology
Sumacab Campus

Input Last Name: *Rizal*
Input First Name: *Jose*
Input Middle Initial: *P.*
Input Address: *Cabanatuan City*
Input Contact Number: *09151234567*
Input Birth Year: *1991*

Full Name; Jose P. Rizal
Address: Cabanatuan City
Contact Number: 09151234567
Birth Year: 1991
Age: 30

Save it as *LabEx2.cpp*

UNIT 7. DECISION MAKING

Learning Objectives

At the end of the unit, I am able to:

1. *write decision making statements in C++, including an if statement and an if else statement;*
2. *use relational and logical operators to make decision in a program; and*
3. *write other decision making statements in C++ including nested if else and switch statement.*

In making decisions in programming, the programmer should use ***comparison expression*** and then choose a path in which to take depending on the answer to the comparison. The ***SELECTION*** structure or ***DECISION*** structure is used for this style of programming.

A. RELATIONAL OPERATORS

Relational operators are also called comparison operators because they compare values. The result of of the comparison is a ***Boolean value***, either a TRUE or a FALSE.

COMPARISON OPERATORS			
OPERATOR	**NAME**	**EXAMPLE**	**ANSWER**
==	equal to	25 == 25	true
!=	not equal to	80 !=80	false
<	less than	78 < 77	false
<=	less than or equal to	26 <= 30	true
>	greater than	45 > 45	false
>=	greater than or equal to	11 >= 11	true

You call a variable which stores Boolean values as a ***Boolean variable***.

B. BOOLEAN OPERATORS

Boolean operators in C++ are operators used in Boolean values which will result to a single Boolean value. They are also called ***logical operators***.

BOOLEAN OPERATORS		
OPERATOR	**NAME**	**DESCRIPTION**
!	**NOT**	**Logical negation**
&&	**AND**	**Logical conjuction**
||	**OR**	**Logical disjuction**

1. The ***NOT (!)*** operator negates the value of the Boolean variable.

TRUTH TABLE FOR OPERATOR !			
p	**!p**	**Example**	**Answer**
true	**false**	**!(1 < 2)** !(true)	**false**
false	**true**	**!(5 > 8)** !(false)	**true**

In the first example, the value of (1 < 2) is ***true***. Applying the NOT operator, !(true) will evaluate to ***false***. In the second example, the value of (5 > 8) is ***false***. Applying the NOT operator, !(false) will evaluate to ***true***.

2. The ***AND* (&&)** operator in a Boolean comparison will result to ***true*** if and only if all of the operands that are being compared have ***true*** values.

TRUTH TABLE FOR OPERATOR &&				
p1	**p2**	**p1 && p2**	**Example**	**Answer**
false	**false**	**false**	**(5 == 2) && (5 > 5)** (false) && (false)	**false**
false	**true**	**false**	**(43 < 27) && (17 >= 2)** (false) && (true)	**false**
true	**false**	**false**	**(50 != 12) && (8 > 15)** (true) && (false)	**false**
true	**true**	**true**	**(22 <= 22) && (11 > 5)** (true) && (true)	**true**

3. The ***OR* (||)** operator in a Boolean comparison will result to ***true*** if one of the operands that are being compared has a ***true*** value.

TRUTH TABLE FOR OPERATOR ||				
p1	**p2**	**p1 || p2**	**Example**	**Answer**
false	**false**	**false**	**(5 == 2) || (5 > 5)** (false) || (false)	**false**
false	**true**	**false**	**(43 < 27) || (17 >= 2)** (false) || (true)	**true**
true	**false**	**false**	**(50 != 12) || (8 > 15)** (true) || (false)	**true**
true	**true**	**true**	**(22 <= 22) || (11 > 5)** (true) || (true)	**true**

C. THE if STATEMENT

The ***if statement*** is one of the selection structures that will execute the statements or block of statements following it when the comparison expression evaluates to true. However, if the comparison expression evaluates to false, it will go out of the if structure without doing anything. The if statement syntax is as follows:

```
if (comparison expression)
{
      statement_1;
      statement_2;
    .
     .
     .
    statement_n;
}
```

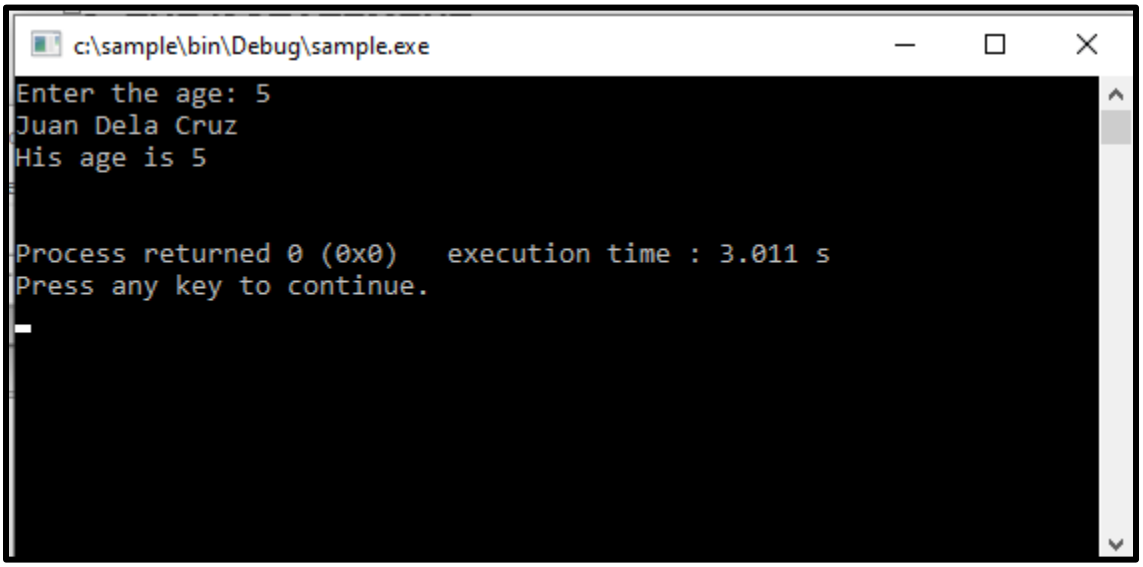
SAMPLE PROBLEM:

Make a C++ program that will print the name and age of a person if the input age is less than 10.

SAMPLE PROGRAM:

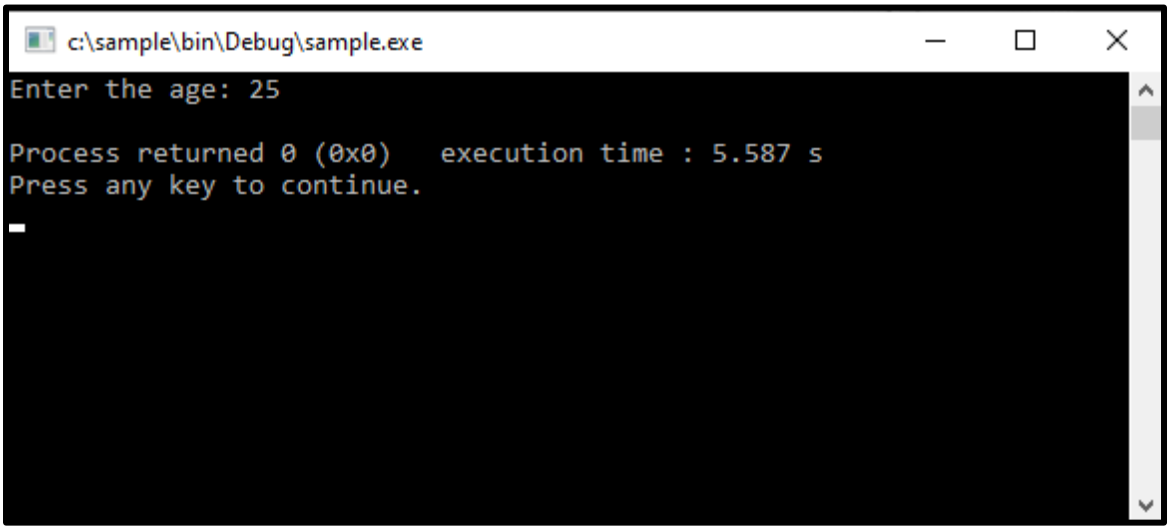
```
1   #include <iostream>
2   #include <cstdlib>
3
4   using namespace std;
5
6   int main()
7   {
8       int age;
9
10      cout<<"Enter the age: ";
11      cin>>age;
12
13      if (age < 10)
14      {
15          cout<<"Juan Dela Cruz"<<endl;
16          cout<<"His age is "<<age<<endl<<endl;
17      }
18
19      return EXIT_SUCCESS;
20  }
21
```

SAMPLE OUTPUT:



The output above shows that the name and age of a person is printed because the evaluation of the ***if*** statement is ***true*** since the input age ***5*** is less than ***10***.

ANOTHER SAMPLE OUTPUT:

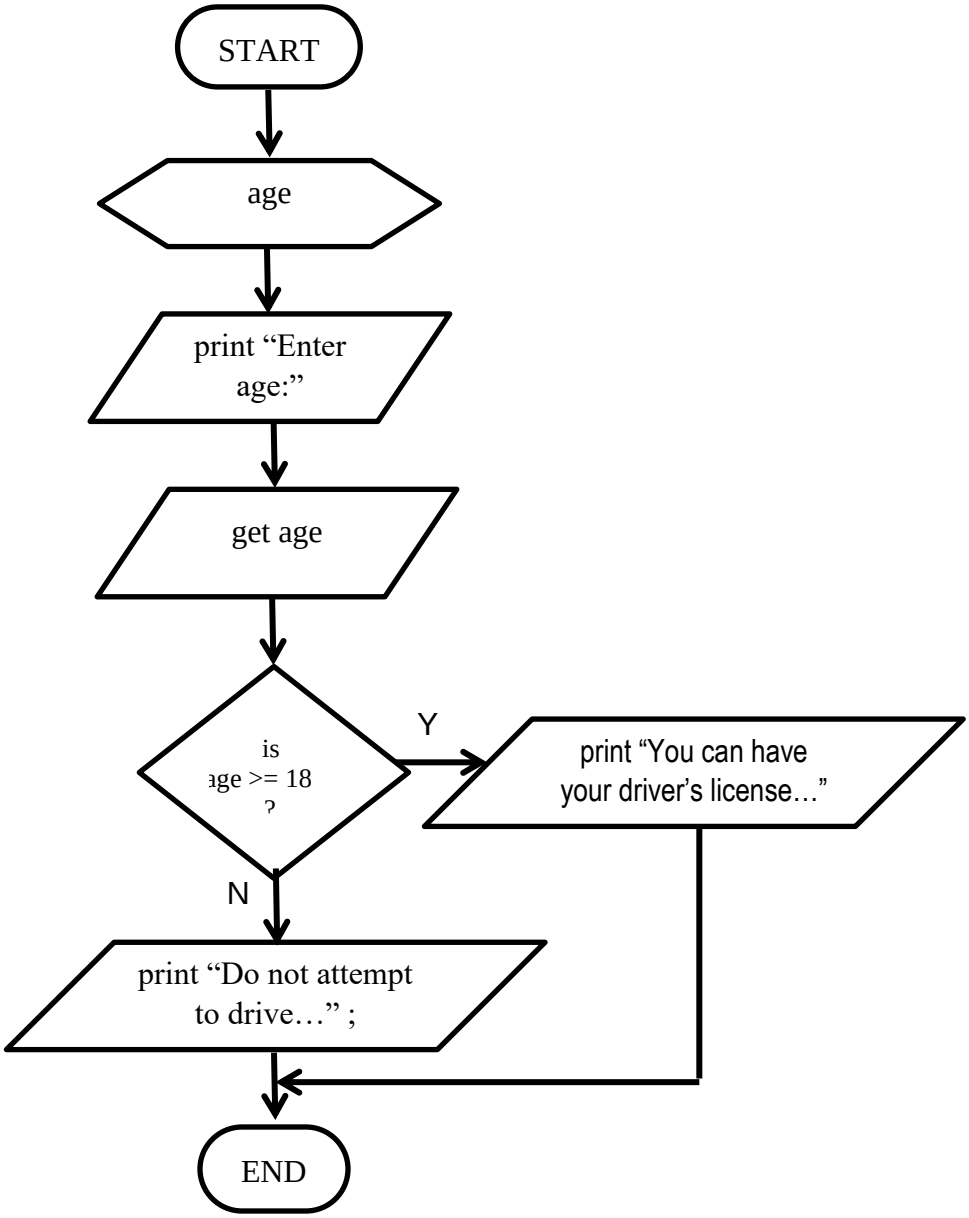


In this sample output, nothing is printed after the input of ***25*** since it does not satisfy the condition of the ***if*** statement.

D. THE if else STATEMENT

The ***if else statement*** is a selection construct that will execute a statement or a block of statements when the condition evaluates to true or another series of instructions when the result of the comparison is false.

The figure below is an example of the flowchart of an if else construct.



The C++ structure of a simple ***if else*** statement is as follows:

```
if (comparison expression)
{
        statement/s;
}
        else
        {
                statement/s;
        }
```

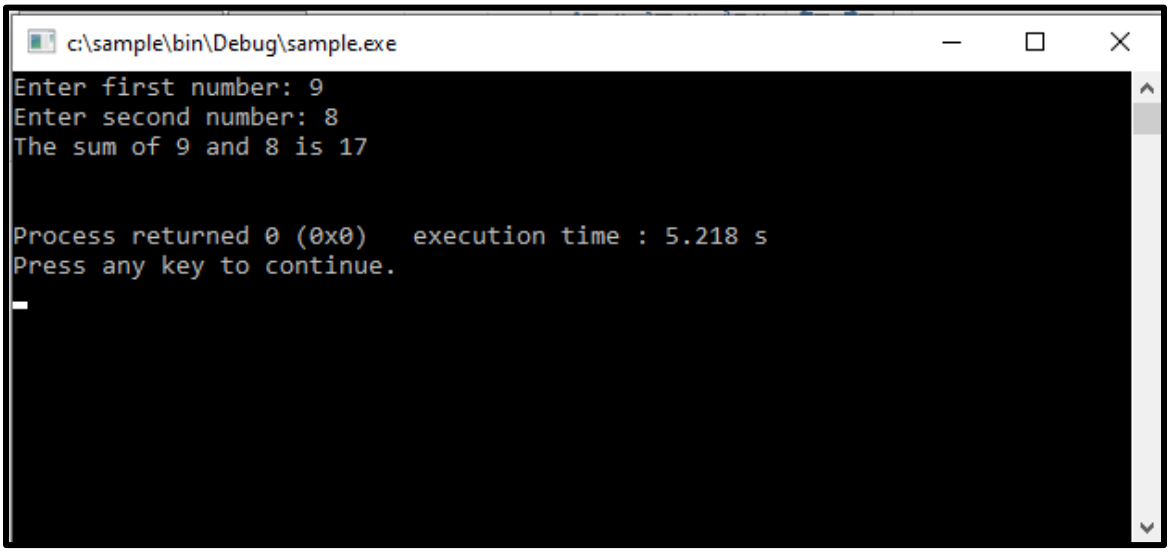
SAMPLE PROBLEM:

Two numbers will be input values from the user. Make a C++ program that will compute and display the sum of the numbers if the first number is in the range of 5-10. Otherwise, compute and then display the product of the two numbers.

SAMPLE PROGRAM:

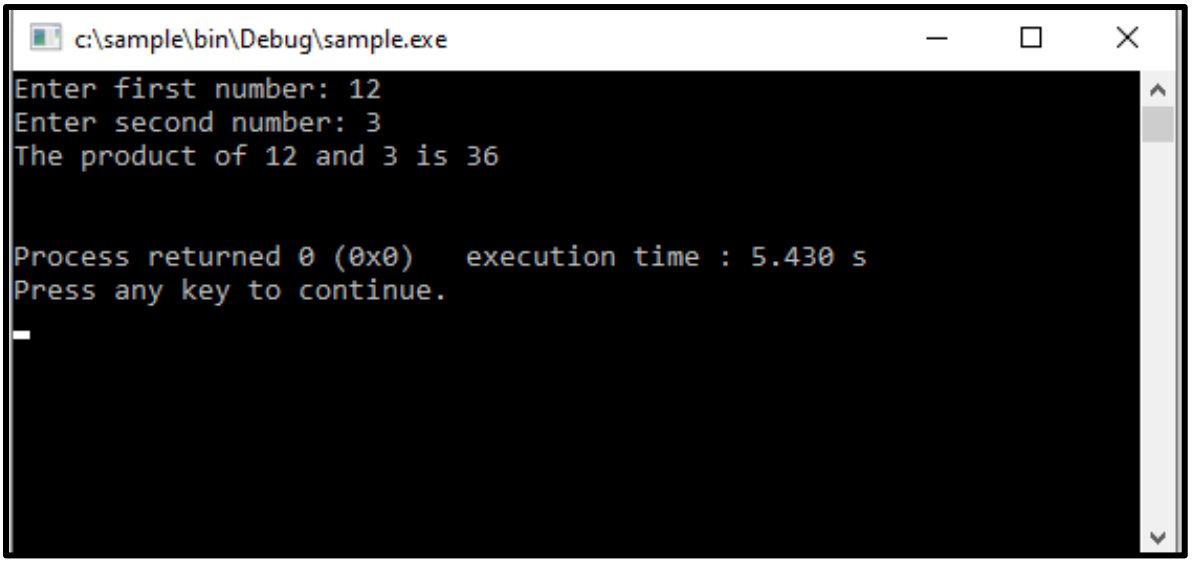
```
1   #include <iostream>
2   #include <cstdlib>
3
4   using namespace std;
5
6   int main()
7   {
8       int num1, num2, sum, prod;
9
10      cout<<"Enter first number: ";
11      cin>>num1;
12      cout<<"Enter second number: ";
13      cin>>num2;
14
15      if (num1 >= 5 && num1 <= 10)
16      {
17          sum = num1 + num2;
18
19          cout<<"The sum of "<<num1<<" and "<<num2<<" is "<<sum<<endl<<endl;
20      }
21          else
22          {
23              prod = num1 * num2;
24
25              cout<<"The product of "<<num1<<" and "<<num2<<" is "<<prod<<endl<<endl;
26          }
27
28      return EXIT_SUCCESS;
29  }
30
```

SAMPLE OUTPUT:



In this sample output, the input number ***9*** satisfies the condition of the ***if else*** statement, so, it computed and printed the ***sum*** of the two input numbers.

ANOTHER SAMPLE OUTPUT:



The above output shows that the input value is ***12*** and it resulted to a ***false*** value, so the program proceeded in executing the ***else*** part of the ***if else*** statement by computing and printing the ***product*** of the two input numbers.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 16

I. Evaluate the following expressions; where x = 30, y = 25, z = 20.

	Solution	Answer
1. (x + y) >= z	_____	_____
2. y < z	_____	_____
3. (x + 3) != y	_____	_____
4. (z - 12) > y	_____	_____
5. y > (z - 70)	_____	_____
6. (z <= y) || (y > x)	_____	_____
7. !(y > x)	_____	_____
8. (x < y) && (y > z)	_____	_____
9. (x == 2) || (y >= 5)	_____	_____
10. (z<x) && (!(z>y))	_____	_____

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 18

Make a C++ program that will input two numbers and compute, then print the difference of the numbers if the first number is more than the second number. If the second number is more than the first number or they are equal, the program should print the product of the numbers. Draw the flowchart first.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 19

Draw a flowchart then create a program that will output the area of a circle if the first input number is not equal to the second input number. Otherwise it should compute and display the area of a square.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 3

Open the code for *LabEx2.cpp* and modify it by adding syntax to apply If-Statements that will:

a. Ask the user to input a letter. The program will check if the input is S, it will display “Scholar”.

Sample Output 1:
Nueva Ecija University of Science and Technology
Sumacab Campus

Input Last Name: *Rizal*
Input First Name: *Jose*
Input Middle Initial: *P.*
Input Address: *Cabanatuan City*
Input Contact Number: *09151234567*
Input Birth Year: *1991*
Input a Letter: *S*

Full Name: Jose P. Rizal
Address: Cabanatuan City
Contact Number: 09151234567
Birth Year: 1991
Age: 30
Remarks: Scholar

Sample Output 2:
Nueva Ecija University of Science and Technology
Sumacab Campus

Input Last Name: *Rizal*
Input First Name: *Jose*
Input Middle Initial: *P.*
Input Address: *Cabanatuan City*
Input Contact Number: *09151234567*
Input Birth Year: *1991*
Input a Letter: *A*

Full Name: Jose P. Rizal
Address: Cabanatuan City
Contact Number: 09151234567
Birth Year: 1991
Age: 30

Save it as *LabEx3.cpp*

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 4

Open the code for *LabEx3.cpp* and modify it by adding syntax to apply If-Else Statements that will:

a. Ask the user to input the Sex. The program will identify if it is a Male or Female.

Sample Output 1:
Nueva Ecija University of Science and Technology
Sumacab Campus

Input Last Name: *Rizal*
Input First Name: *Jose*
Input Middle Initial: *P.*
Input Address: *Cabanatuan City*
Input Contact Number: *09151234567*
Input Birth Year: *1991*
Input a Letter: *S*
Input Sex: M

Full Name: Jose P. Rizal
Address: Cabanatuan City
Contact Number: 09151234567
Birth Year: 1991
Age: 30
Remarks: Scholar
Sex: Male

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Sample Output 2:
Nueva Ecija University of Science and Technology
Sumacab Campus

Input Last Name: *Rizal*
Input First Name: *Jose*
Input Middle Initial: *P.*
Input Address: *Cabanatuan City*
Input Contact Number: *09151234567*
Input Birth Year: *1991*
Input a Letter: *S*
Input Sex: F

Full Name: Jose P. Rizal
Address: Cabanatuan City
Contact Number: 09151234567
Birth Year: 1991
Age: 30
Remarks: Scholar
Sex: Female

Save it as *LabEx4.cpp*

E. THE nested if-else STATEMENT

If there are more than two alternatives or choices, you can use the ***nested if statement.***

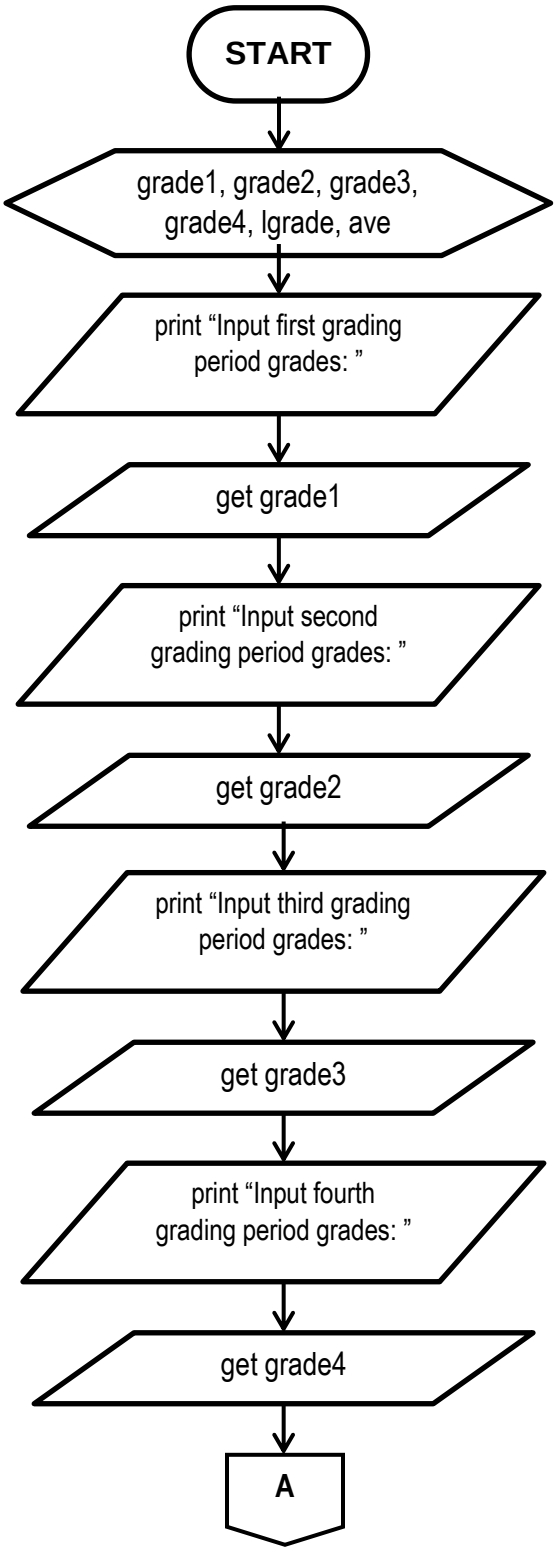
```
if (comparison expression)
{
        statement/s;
}
else
        if  (comparison expression)
        {
                statement/s;
        }
        else
                if  (comparison expression)
                {
                        statement/s;
                }
```

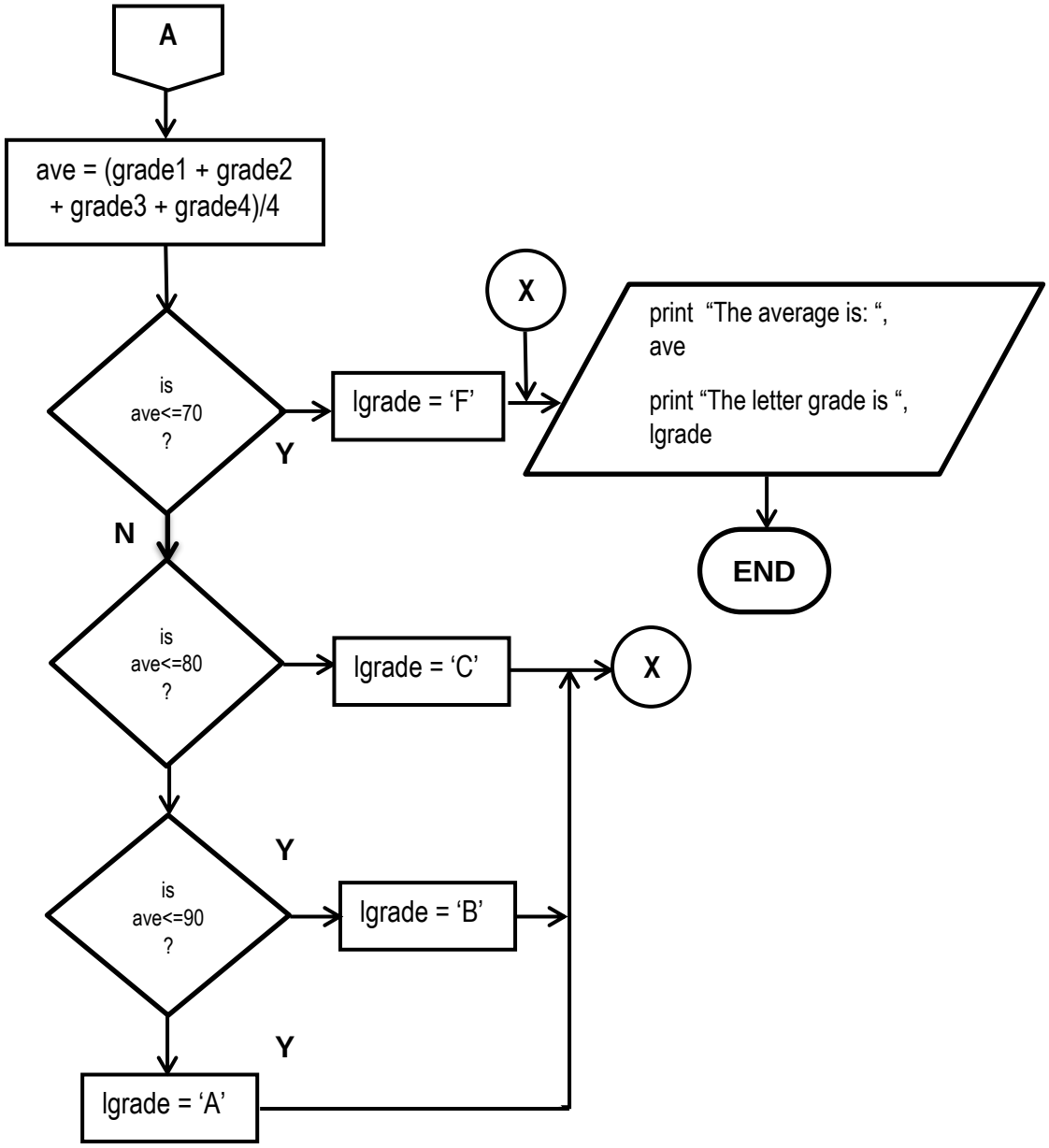
The depth of a nested if statement is dependent on the given problem.

SAMPLE PROBLEM:

Make a C++ program that will compute for the average of the grades of a high school student by getting the average of his grades from first grading period to the fourth, and display the letter grade using the following conversions: 91 – 100 = ‘A’, 81 – 90 = ‘B’, 71 – 80 = ‘C’, 70 and below = ‘F’. It should print the average of the grades of the student and his letter grade.

FLOWCHART:


START
grade1, grade2, grade3, grade4, lgrade, ave
print “Input first grading period grades: ”
get grade1
print “Input second grading period grades: ”
get grade2
print “Input third grading period grades: ”
get grade3
print “Input fourth grading period grades: ”
get grade4
A



ANOTHER SAMPLE PROBLEM:

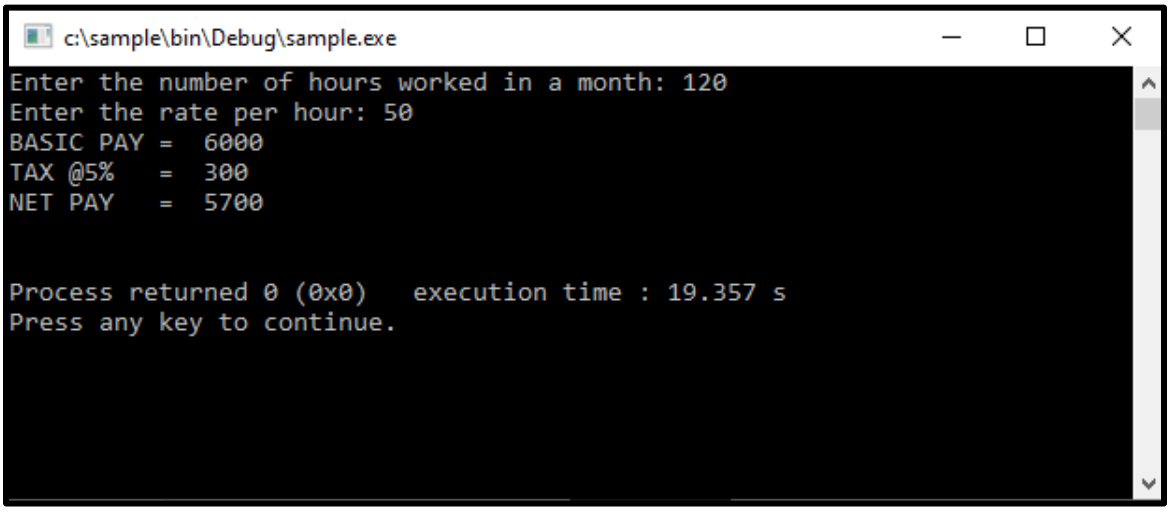
Write a C++ program that will ask for a number of hours that an employee had worked for a month and his or her rate per hour. Then compute and print the netpay based on the following:

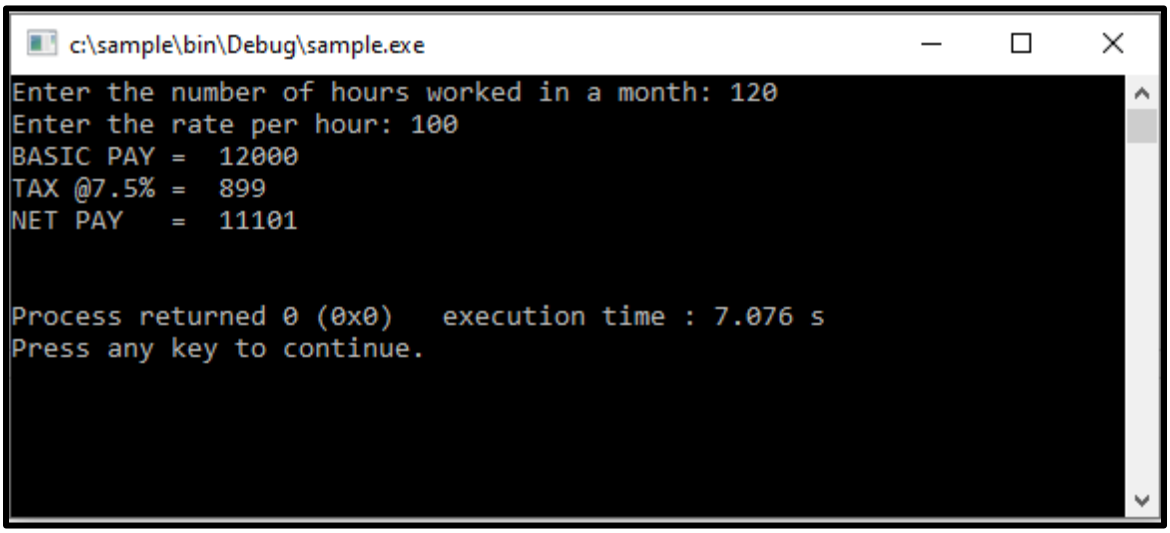
Basic Pay	**Tax**
10000 or less	5% of Basic Pay
10001 - 20000	7.5% of Basic Pay
More than 20000	10% of Basic Pay

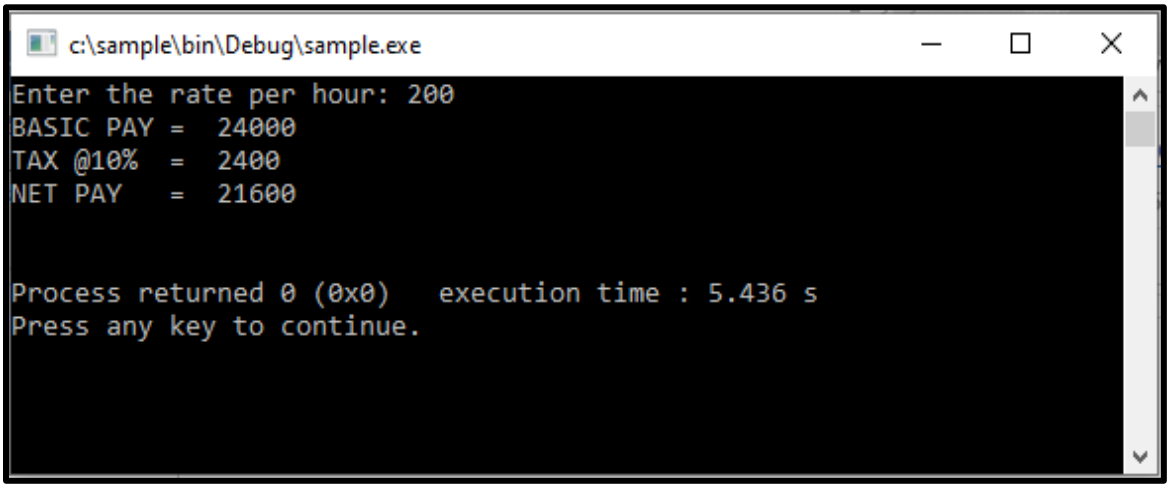
SAMPLE PROGRAM:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    int nhw, rph, bpay, tax, npay;

    cout<<"Enter the number of hours worked in a month: ";
    cin>>nhw;
    cout<<"Enter the rate per hour: ";
    cin>>rph;

    bpay = nhw *rph;

    cout<<"BASIC PAY =  "<<bpay<<endl;

    if (bpay <= 10000)
    {
       tax = bpay * 0.05;

       cout<<"TAX @5%   =  "<<tax<<endl;
    }
       else if (bpay <= 20000)
       {
           tax = bpay * 0.075;

           cout<<"TAX @7.5% =  "<<tax<<endl;
       }
           else
           {
               tax = bpay *0.1;

               cout<<"TAX @10%  =  "<<tax<<endl;
           }
    npay = bpay - tax;

    cout<<"NET PAY   =  "<<npay<<endl<<endl;

    return EXIT_SUCCESS;
}
```

SAMPLE OUTPUTS:







Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 20

Using the nested if else statement, create a c++ program that will compute for your electricity bill based on the following data:

KILOWATT USED	ELECTRIC BILL
20 or less	₱5.50/kwh
21 – 50	₱10.00/kwh
More than 50	₱15/kwh

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 5

Open the code for *LabEx4.cpp* and modify it by adding syntax to apply Nested If-Else Statements that will:

Input Attendance Score:
Input Seatworks Score:
Input Recitations Score:
Input Quizzes Score:
Input Laboratory Exercises Score:
Input Term Exam Score:

Compute for the following:
Attendance 10%
Seatworks 15%
Recitations 10%
Quizzes 20%
Laboratory Exercises 25%
Term Exam 20%

Compute for the total of the percentage.

Convert the total of the percentage using this range and print the corresponding grade:

Range	Grade
98-100	1.00
95-97	1.25
92-94	1.50
89-91	1.75
85-88	2.00
82-84	2.25
80-81	2.50
77-79	2.75
75-76	3.00
Below 75	5.00

Save it as *LabEx5.cpp*

F. THE switch STATEMENT

The ***switch statement*** is an alternative for if else statement. It looks for a match among the cases and executes the statements inside the match. If no match is found, it will execute the default statements.

```
switch (variable)
 {
    case value_1:
    {
              statement(s)_1;
           break;
    }
    case value_2:
    {
              statement(s)_2;
           break;
    }
   :
   :
    case value_n:
    {
              statement(s)_n;
           break;
    }

    default:
    {
              statement(s);
    }
}
```

SAMPLE PROBLEM 1:

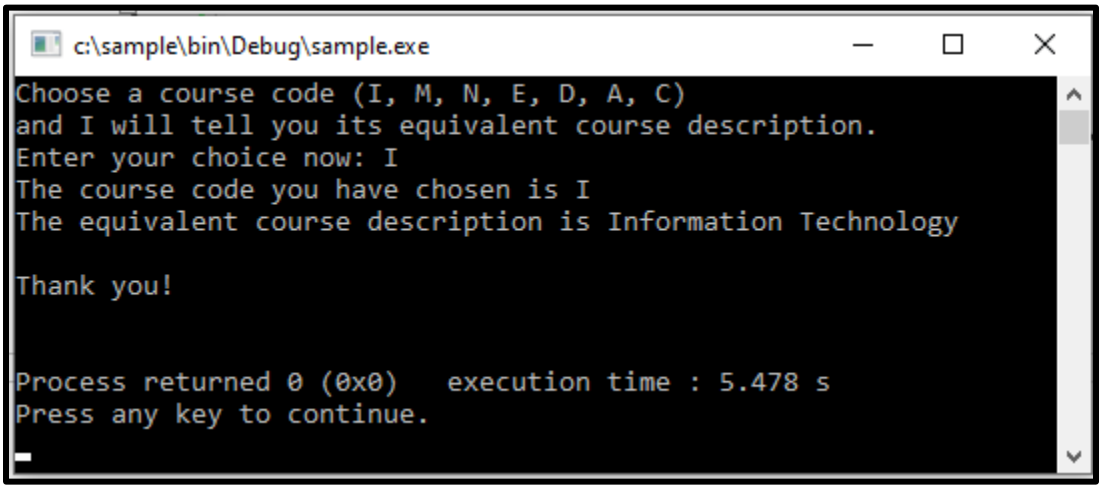
Write a c++ program that inputs the course code of a student then outputs his/her course based in the following data:

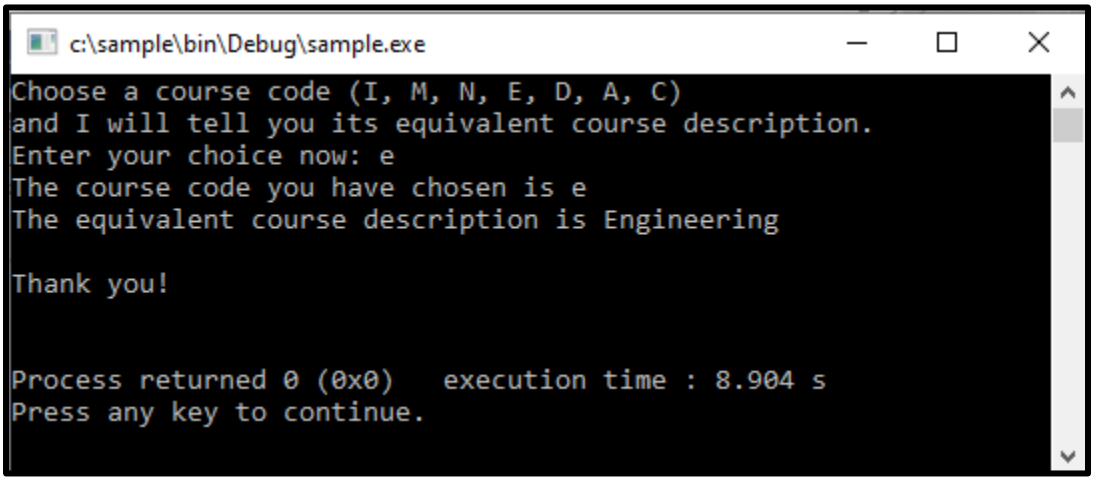
CODE	COURSE DESC
I or i	Infotech
M or m	Management
N or n	Nursing
E or e	Engineering
D or d	Education
A or a	Architecture
C or c	Criminology

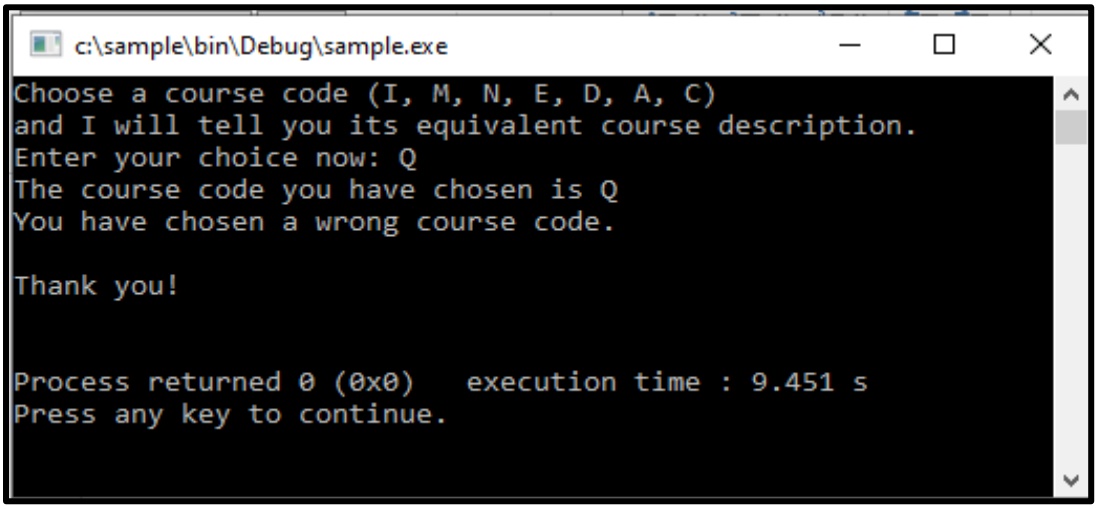
SAMPLE PROGRAM:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char cc;

    cout<<"Choose a course code (I, M, N, E, D, A, C) \n";
    cout<<"and I will tell you its equivalent course description.\n";
    cout<<"Enter your choice now: ";
    cin>>cc;

    cout<<"The course code you have chosen is "<<cc;

    switch (cc)
    {
        case 'I': case 'i':
            cout<<"\nThe equivalent course description is Information Technology\n\n";
            break;
        case 'M': case 'm':
            cout<<"\nThe equivalent course description is Management\n\n";
            break;
        case 'N': case 'n':
            cout<<"\nThe equivalent course description is Nursing\n\n";
            break;
        case 'E': case 'e':
            cout<<"\nThe equivalent course description is Engineering\n\n";
            break;
        case 'D': case 'd':
            cout<<"\nThe equivalent course description is Education\n\n";
            break;
        case 'A': case 'a':
            cout<<"\nThe equivalent course description is Architecture\n\n";
            break;
        case 'C': case 'c':
            cout<<"\nThe equivalent course description is Criminology\n\n";
            break;
        default:
            cout<<"\nYou have chosen a wrong course code.\n\n";
    }

    cout<<"Thank you!\n\n";

    return EXIT_SUCCESS;
}
```

SAMPLE OUTPUTS:







Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 21

Using the switch statement create a c++ program that compute your water bill based on the account classification. A residential building is charged ₱20.00 per cubic meter while a commercial building is charged ₱30.00 per cubic meter.It should first input the current and previous readings of the water consumption.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 22

Make a C++ program that will create a Q and A using the switch statement. Use the menu below for the question and the choices.

What do you call a 2D shape in geometry, having 4 sides and 4 corners. Its two sides meet at right angles?
a - RHOMBUS
b - PARALLELOGRAM
c - RECTANGLE
d - TRAPEZIUM

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Seatwork No. 23

Using switch statement, create a c++ program that provide the five mathematical operations as choices for computation for two input numbers.

Seat No.: _____ Rating: _____

Name: _____ Date: _____

Laboratory Exercise No. 6

Open the code for *LabEx5.cpp* and modify it by adding syntax to apply switch Statements that will:

Convert the grade using this range and print the corresponding interpretation:

Grade	Interpretation
1.00	With Highest Honor
1.25	With High Honor
1.50	With Honor
Below 1.50	Study Harder …

Save it as *LabEx6.cpp*

Activity: What Did You Learn About?

Post-Assessment Test

Congratulations on completing **CC101 – Computer Programming, Fundamentals**! Let's have a little quiz to assess how much have you learned in programming. There is no right or wrong answer. Please answer the questions honestly by marking your responses with a check.

Do You Already Know...	***Yes, I do!***	***Maybe but unsure***	***I still have no idea***
...what is programming?			
... how to draw flowcharts?			
... how to write codes in C++?			
...what are variables?			
...the different control structures?			
... if-else and switch statements?			

Leave a short comment on which topic you enjoyed the most and why?

_____

_____

_____

REFERENCES

Smith, Jo Ann. (2014). *C++ Programs to Accompany Programming Logic and Design*. Cengage Learning.

Leona, Rodibelle F. *Worktext for CC101*.

CICT CC101 – Computer Programming, Fundamentals Module

https://www.tutorialspoint.com/cplusplus/

http://www.cplusplus.com/doc/tutorial/

http://www.text-editor.org/

https://www.techopedia.com/definition/3912/compiler

https://en.wikipedia.org/wiki/GNU_Compiler_Collection

https://itunes.apple.com/app/cppcode-offline-c-c++-ide/id936694712

http://www.cppcode.info/

https://play.google.com/store/apps/details?id=org.zhiyuan.zhiyuan24

https://www.google.com/url?sa=t&source=web&rct=j&url=https://play.google.com/store/apps/details%3Fid%3Dname.antonsmirnov.android.cppdroid%26hl%3Den%26referrer%3Dutm_source%253Dgoogle%2526utm_medium%253Dorganic%2526utm_term%253Dcppdroid%26pcampaignid%3DAPPU_1_rbENX5jpCceUr7wPwuqzuAE&ved=2ahUKEwjY84Gj68zqAhVHyosBHUL1DBcQ5IQBMAB6BAgPEAM&usg=AOvVaw0hW7KjCxj0wxFIL1Tq-8ky&cshid=1594733019938

https://play.google.com/store/apps/details?id=ru.iiec.cxxdroid

http://texteditor.org/?fbclid=IwAR0phsV4QRGThctQObygLFgZAmCbHFA9F04y896a74Bm5RNm8zTzg8ccwJY

http://www.webopedia.com/TER M/s/source_code.html