Advanced Programming (C++)

BY

Dr. EMAD SAMI

http://www.bu.edu.eg/staff/emadattwa3

Course Chapters

1. Introduction
2. Variables and Constants
3. Expressions and Statements
4. Loops and Decisions
5. Functions
6. Arrays and Strings
7. Pointers
8. Miscellaneous

1. Introduction

Chapter Objectives:

1-1 Programming Languages

1-2 Setting up the Program

1-3 C-Free 4.0 IDE Preparation Steps

1-4 Program Execution and Testing

1-5 Types of Errors

1-6 My First Program and the Output

1-7 Basic Parts of C++ Program

1-8 Assignment (1)

1-1 Programming Languages

- Programming Languages are classified into High Level Languages (HLL) and Low Level Languages (LLL).
- HLL are two types: Machine Language and Assembly Language
- LLL are three types: Procedural, Structured and Object-Oriented
- *Machine Language*:

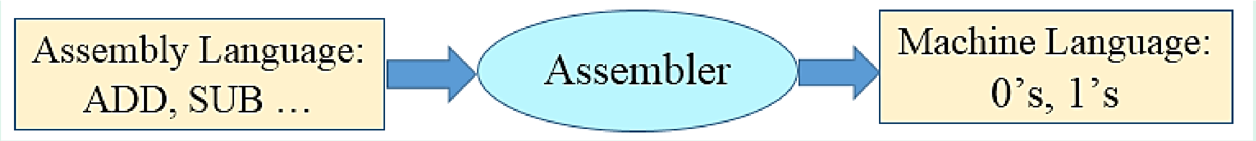
- It appears first and deals directly with the computer.
- The program was written by collecting some of Zeros (0's) and Ones (1's) to construct the code.

$$\begin{bmatrix} 01100111 \\ 01011011 \\ 10011010 \end{bmatrix} \\ \vdots$$

- *Assembly Language:*

- As group of 0's and 1's are very difficult to read and written, so we collect some of this code to give one symbol ("mnemonic") such as ADD, SUB, …etc.
- It is easy to deal with but it needs assembler to convert that symbol into Machine Language code of 0's and 1's.

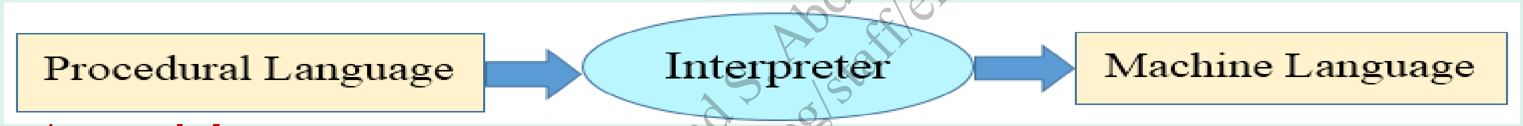
$$\begin{bmatrix} \left.\begin{matrix} 01011011 \\ 01100111 \end{matrix}\right\} \text{ADD} \\ \left.\begin{matrix} 01011011 \\ 10011010 \end{matrix}\right\} \text{SUB} \end{bmatrix} \\ \vdots$$



1-1 Programming Languages …

- *Procedural Language:*

- Each statement in the code tells the computer to do something. Get some input, add numbers, display output. The program is a list of instructions. For example languages: C, BASIC, COPOL, …etc.
- It needs an interpreter to convert the Procedural Code into Machine Language Code.



- *Structured Language:*

- When the code becomes larger, it is a must to divide the code into some functions. Each function has its purpose and executed when it is called from the main program. Examples of functions are; (Ex1: get a greater number from a list of numbers, Ex2: get factorial of a certain number). For example languages: new version of C, new version of BASIC, …etc.
- It needs an interpreter to convert the Structural Code into Machine Language Code.

1-1 Programming Languages …

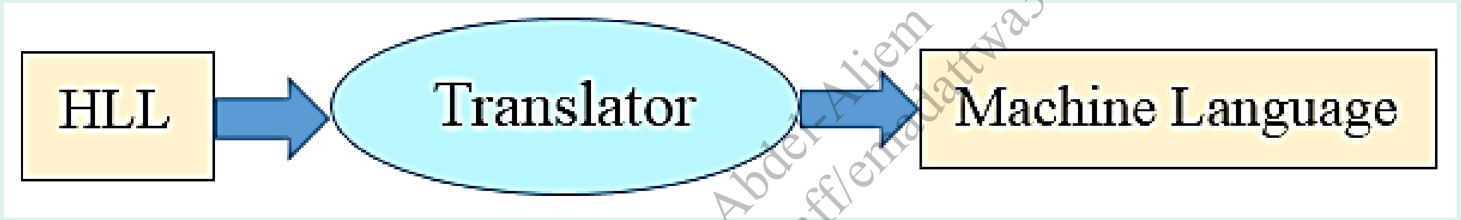
- *Object-Oriented Programming Language (OOP)*

- When the program becomes more and more large and complex, we make classes (each class contains many functions) to reduce the number of statements.
- Compiler translates the object-oriented code (the whole code) into an intermediary form to produce an Object file of 0's and 1's. (.o file)
- We can put the Object file at the necessary place.
- Also, you can build Objects that have known prosperities and then be able to use them in your future codes as you need.
- For example languages: C++, JAVA, …etc.

1-1 Programming Languages …

Compare between Translators?

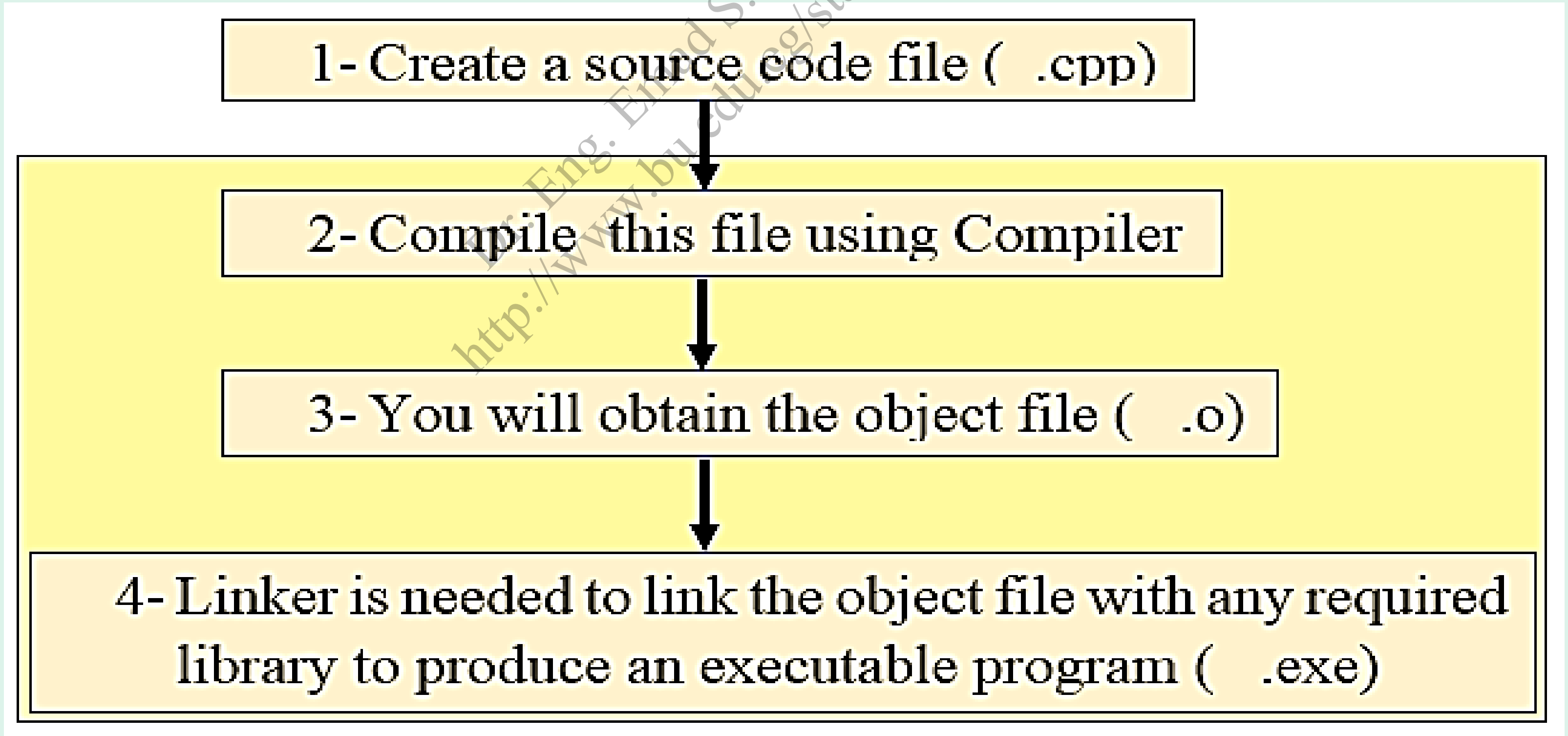
- Translator is Interpreter or Compiler.
- All programs written in HLL must be translated to the machine code (the language that computer understand and deal with).



- Interpreter: translates from BASIC, C to Machine code. It translates the program line by line, if error occurs it will stops until the programmer correct the error. It is slower than compiler but easier to debug and correcting.
- Compiler: translates from C++, BASCAL to Machine code. It translates the file once as whole, if errors occurs it will be shown after translation. So, the compiler is very fast.

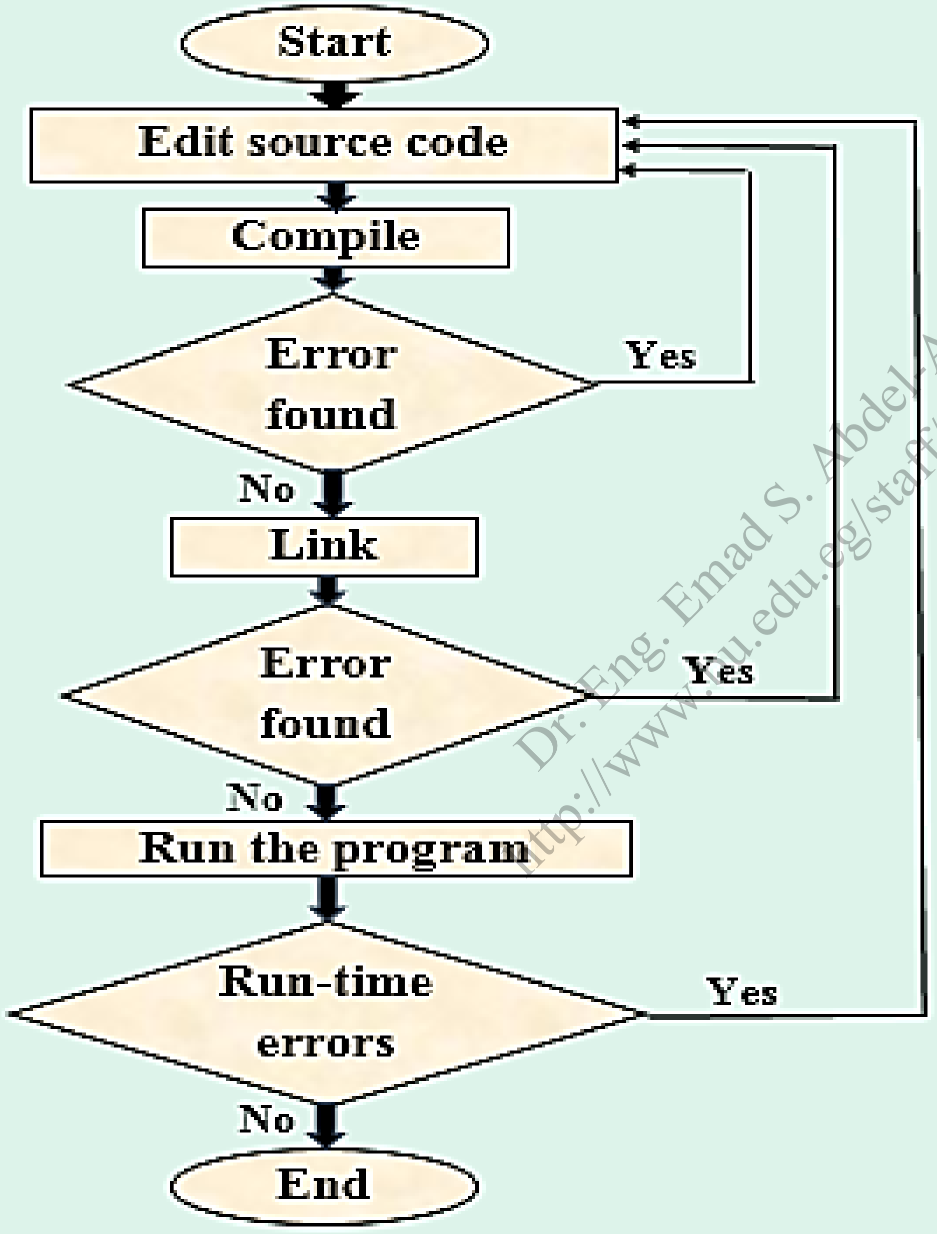
1-1 Programming Languages …

- Your source code file is not a program, and it can't be executed or run as a program (.exe) can.
- To turn on your source code, you must use a compiler, hence an object file (.o file) is produced.
- To turn on the (.o file), you must run your linker to obtain an executable file (.exe file)
- The steps to create an executable file (.exe) of C++ code:


1- Create a source code file (.cpp)
2- Compile this file using Compiler
3- You will obtain the object file (.o)
4- Linker is needed to link the object file with any required library to produce an executable program (.exe)

1-1 Programming Languages …

- The steps to create an executable file (.exe) with flow chart:



- If the program runs for the first time, it must have the cycle (**Edit**, **Compile**, **Link** and **Run**).
- If you have errors during run of the program you must fit it and rerun to reach the End step in the flow chart.

1-2 Setting up the Program

* For using C++ Programming Language on you PC; your PC must have specifications not less than the following:

1- Processor: Pentium 4

2-HDD: 80Gb

3-RAM: 512 Mb

- C++ Programming Language that will use in this course:

C-Free 4.0 IDE

1-2 Setting up the Program…

* For writing a source code using C++ Programming Language**:** you need**:**

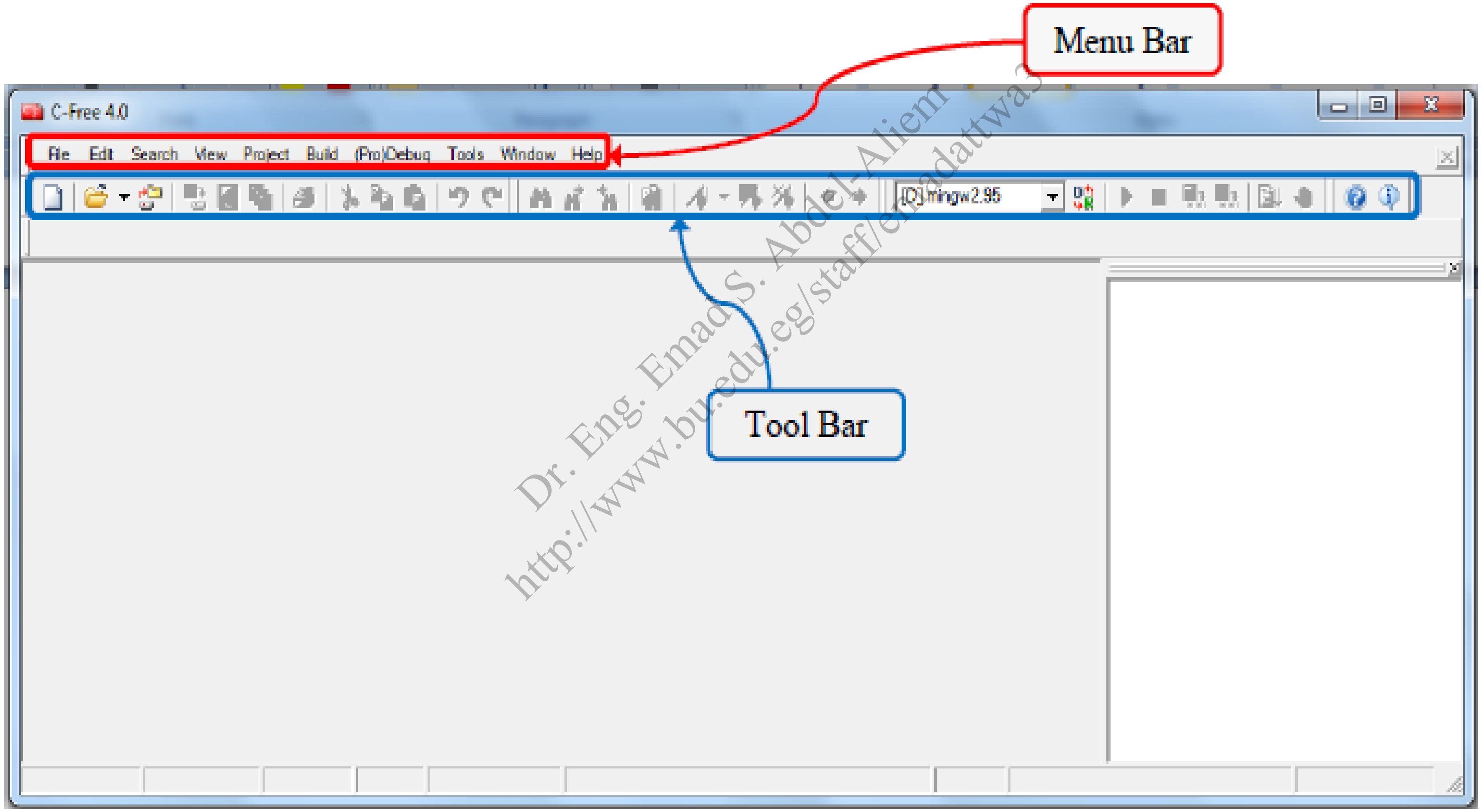
1- Text Editor: to write the source code; but save it to extension “ .cpp”
2- Compiler: to debug “i.e., finding and correcting errors” and convert the source code into executable file “.exe”, then run it.

Or

C-Free 4.0 software which is a professional C/C++ Integrated Development Environment (IDE) that: allow you to: edit, build, debug, and run programs freely.

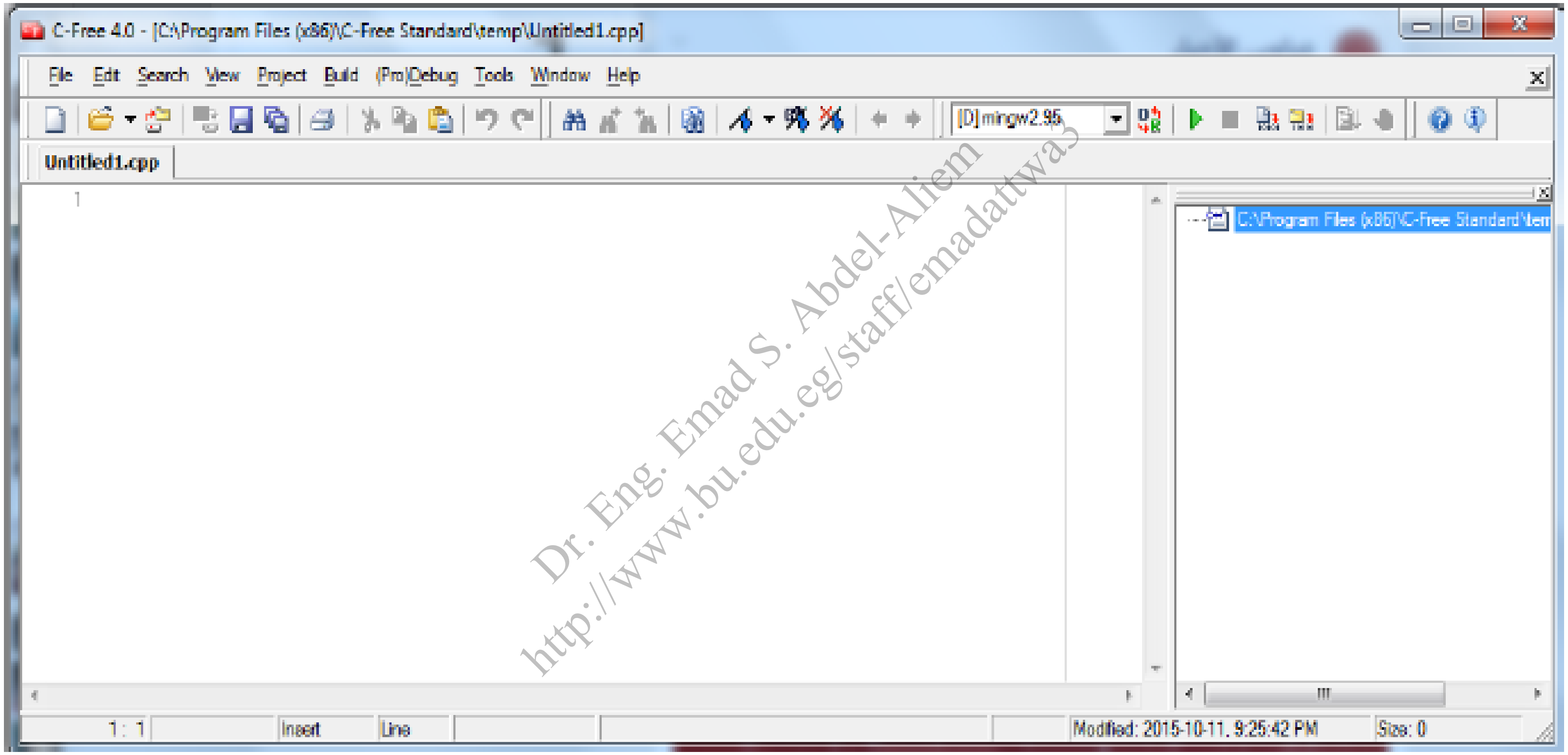
1-3 The C-Free 4.0 IDE Preparation Steps

1- Setup/Install C-Free 4.0 IDE on your HDD.
2- Run C-Free 4.0 IDE; you will see:


Menu Bar
C-Free 4.0
File Edit Search View Project Build (Pro)Debug Tools Window Help
[D] mingw2.95
Tool Bar

1-3 The C-Free 4.0 IDE Preparation Steps …

3- From Menu Bar, open **File>New** or **Ctrl+N**

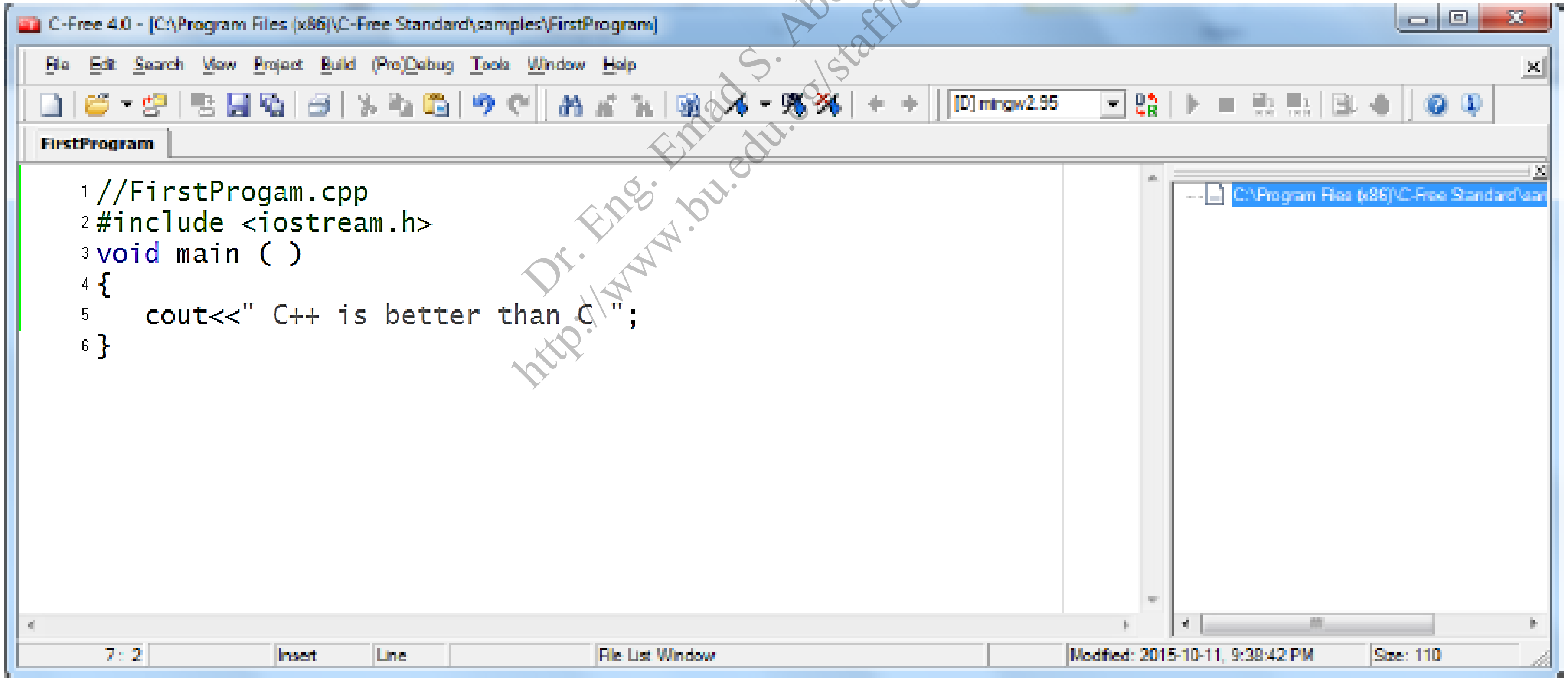

C-Free 4.0 - [C:\Program Files (x86)\C-Free Standard\temp\Untitled1.cpp]
File Edit Search View Project Build (Pro)Debug Tools Window Help
[D] mingw2.95
Untitled1.cpp
C:\Program Files (x86)\C-Free Standard\tem
1 : 1
Insert
Line
Modified: 2015-10-11, 9:25:42 PM
Size: 0

1-3 The C-Free 4.0 IDE Preparation Steps …

First C++ Program

1- Open C Program by double click on the icon at the desktop:

2- **File>New** … then edit the following program as shown


C-Free 4.0 - [C:\Program Files (x86)\C-Free Standard\samples\FirstProgram]
File Edit Search View Project Build (Pre)Debug Tools Window Help
[D] mingw2.95
FirstProgram
C:\Program Files (x86)\C-Free Standard\sam
7: 2
Insert
Line
File List Window
Modified: 2015-10-11, 9:38:42 PM
Size: 110

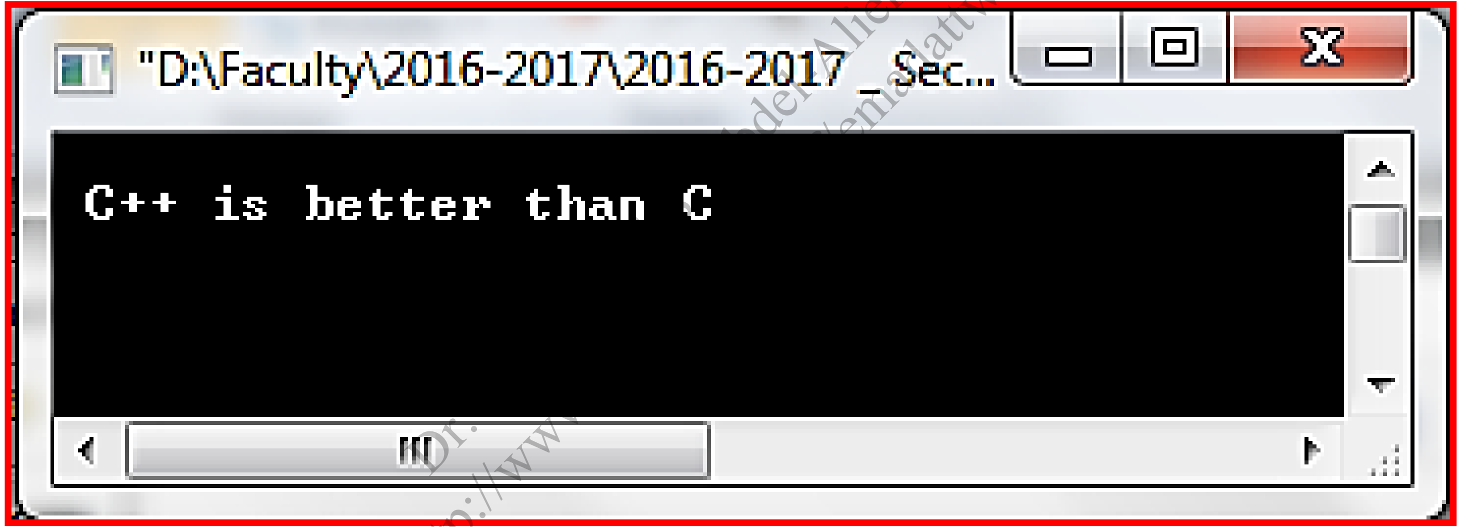
```
//FirstProgam.cpp
#include <iostream.h>
void main ( )
{
   cout<<" C++ is better than C ";
}
```

1-3 The C-Free 4.0 IDE Preparation Steps …

3- **Debug** or **F9** … to check if errors or not in the program

4- **File>Save As** … **FirstProgram.cpp**

5- **Run** or **F5** … run the program to appear the output



```
C++ is better than C
```

1.4 Program Execution and Testing

- Here are the four steps to run the program:

  a. *Editing* b. *Compiling*

  c. *Linking* d. *Executing* (*run*).

1.4 Program Execution and Testing …

a. Editing Step

- The program code is transferred from a paper form into electronic form by using any types of editors, e.g. *Notepad* or *Word processing*.
- Then save the edited file with a suitable name with extension (.cpp), because the program codes are written using the C++ programming language.
- The program file (.cpp file) at this step is called the source code.

1.4 Program Execution and Testing …

b. Compiling Step

- In this step the source code (.cpp file) is transferred from HLL (C++ Language) into machine language (Binary code) using a compiler as C-free 4.0.
- The compiler starts to check that the program code match the used programming language. If no errors found, the binary code is obtained. But, if the compiler finds any syntax error, it displays a message to the user to correct that error.
- ***Note that***: the binary code is saved automatically by the compiler in an object file has the same name entered by the user but with the extension (.o).

1.4 Program Execution and Testing …

c. Linking Step

- In this step, a program called the linker, starts to check the logical errors in the object file, if it does not find any; it links the object program with the run-time libraries functions which are required to execute the program, i.e., it copies the required functions from libraries into the user’s program to become a part of it.
- The program after linking with libraries functions is converted into an executable program with the same name entered by the user but with the extension (.exe).

1.4 Program Execution and Testing …

d. Execution Step

- The program in this step is ready to be run or debugged on the computer.

1-5 Types of Errors (*3 types*)

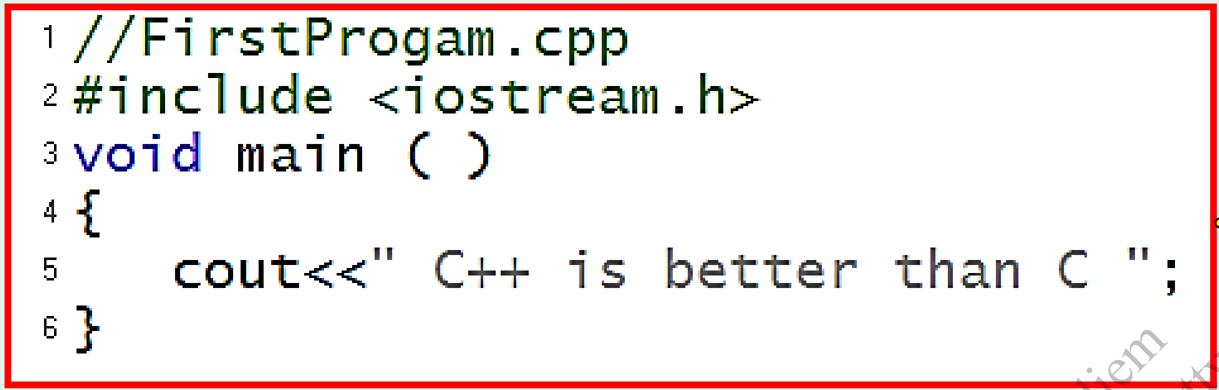
Syntax errors: errors that you have made in the form, or syntax of the language. For example, *spelling*. At running these errors are shown in a dialog box that appears and is highlighted in the program.

Execution errors: For example, at running if a number is divided by zero causes an execution error and the program execution stops. Press the Enter key to clear the dialog box. Then you must correct the error.

Logic errors: if the output of the program does not agree with what is expected, this is logic error.

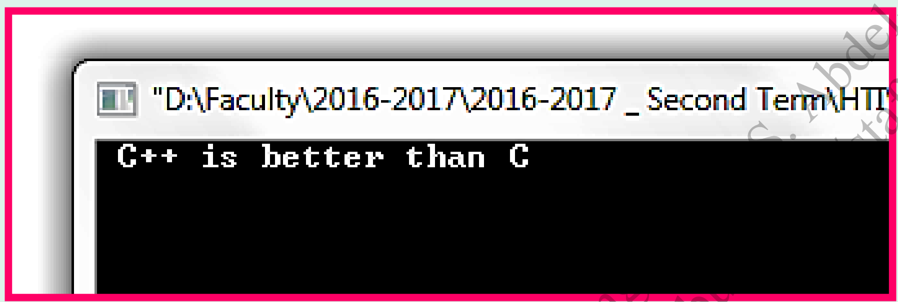
i.e., you expect a certain output but the running execute a different output. So, this depends on the programmer

1-6 My First Program and the Output



```
//FirstProgam.cpp
#include <iostream.h>
void main ( )
{
    cout<<" C++ is better than C ";
}
```

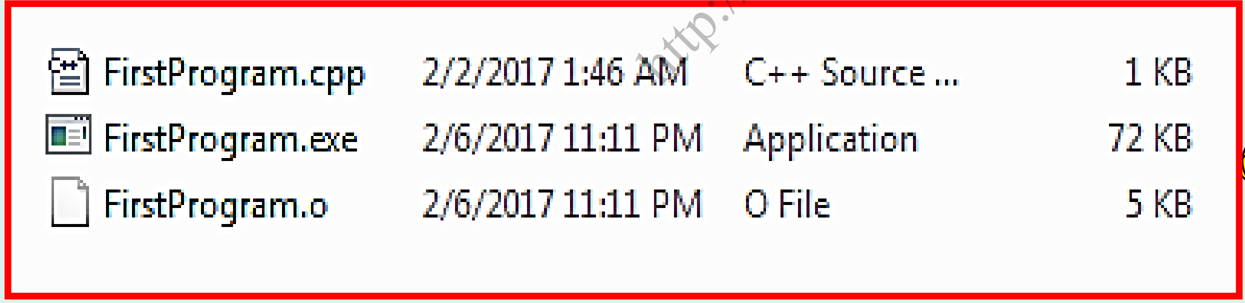
Source code



"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI

```
C++ is better than C
```

Output



FirstProgram.cpp	2/2/2017 1:46 AM	C++ Source ...	1 KB
FirstProgram.exe	2/6/2017 11:11 PM	Application	72 KB
FirstProgram.o	2/6/2017 11:11 PM	O File	5 KB

Files of the program:
___.cpp file,
___.exe file,
___.o file

1-7 Basic Parts of C++ Program

- The program format is as follows:

```
1 //comments to show  the program purpose and name
2 #include<iostream.h>
3 void main ( )
4 {
5         ----------------;
6         ----------------;
7         //comments as needed
8         ----------------;
9 }
```

Line 1: (**// comments**): The double slashes (//) are used to write after them the comments that show the purpose and the name of the program. Number of lines in the comment not affect on the program size. Also, you can use this form to write the comments (/*   */). The compiler ignores the comments when making the **object** file. The comments can be used inside the **main** function to explain what each function does.

1-7 Basic Parts of C++ Program …

```
1 //comments to show  the program purpose and name
2 #include<iostream.h>
3 void main ( )
4 {
5       ----------------;
6       ----------------;
7       //comments as needed
8       ----------------;
9 }
```

Line 1: (**// comments**): comments can be written in one of these forms:

```
//comments to show
//the program purpose and name
```

```
/*comments to show  the program purpose and name*/
```

```
/*comments to show
the program purpose and name*/
```

1-7 Basic Parts of C++ Program …

```
1 //comments to show  the program purpose and name
2 #include<iostream.h>
3 void main ( )
4 {
5       ----------------;
6       ----------------;
7       //comments as needed
8       ----------------;
9 }
```

Line 2: (**#include <   .h>**)

- It is called Preprocessor statement or Preprocessor directive
- It has the form (**#include <   .h>**).
- It includes a header file that contain functions needed in the **main**.
- It is placed and executed before **main** function begins.
- **<iostream.h>** the header file that contains **cin** for input characters from the keyboard and **cout** for print out characters on the screen.
- **<math.h>** the header file that contains mathematical functions such as **sqrt**, **pow(x,2)**, … etc.

1-7 Basic Parts of C++ Program …

```
1 //comments to show  the program purpose and name
2 #include<iostream.h>
3 void main ( )
4 {
5        -----------------;
6        -----------------;
7        //comments as needed
8        -----------------;
9 }
```

Line 3 up to Line 9: the function **main ( )**

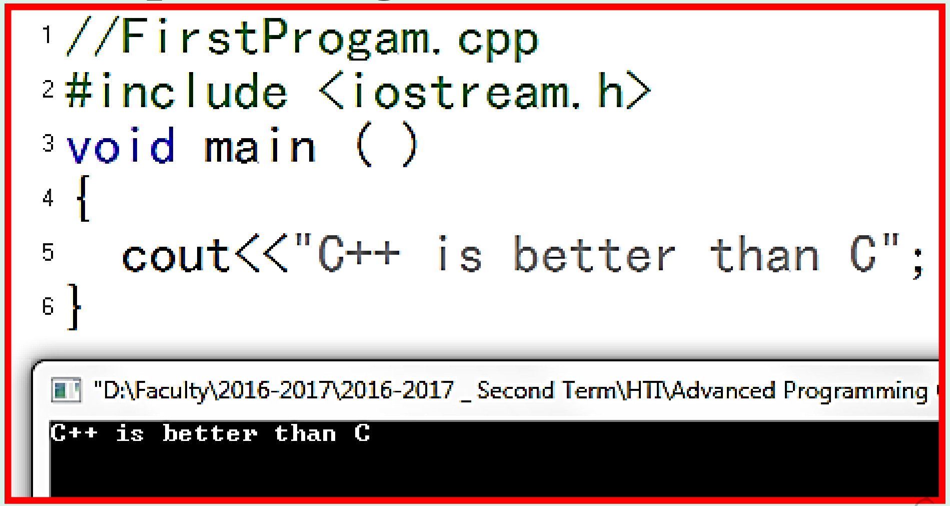
- Every program must have **main ( )** function.
- **main ( )** function starts with an opening brace (**{**) and ends with a closing brace (**}**). Between these braces the code is written.
- **main ( )** function states what the type of data that it will be return.
- return or non-return data will be explained in Chapter 5.

```
void main ( )
{
     ----------;
     ----------;
     //comments as needed
     ----------;
}
```

```
int main ( )
{
     ----------;
     ----------;
     //comments as needed
     ----------;
     return 0;
}
```

1-7 Basic Parts of C++ Program …

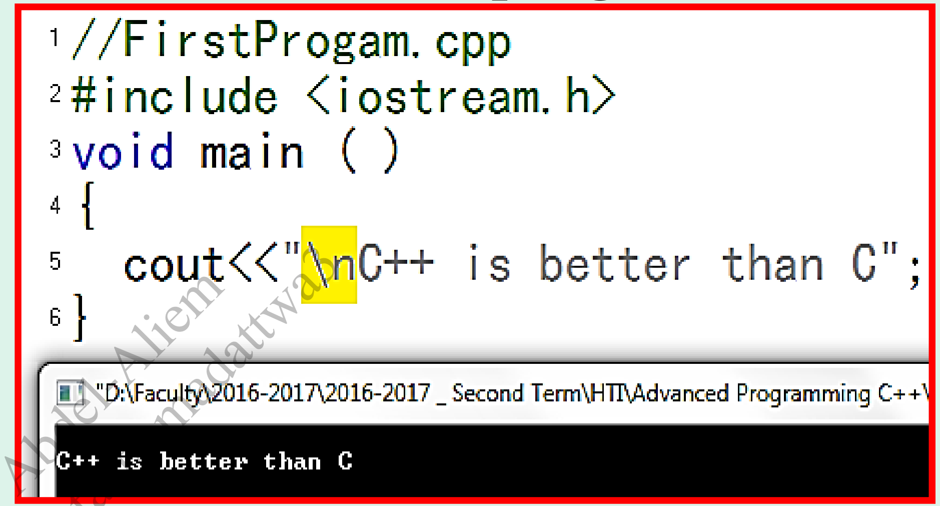
Output using **cout**: note the difference in the three programs:



```
//FirstProgam.cpp
#include <iostream.h>
void main ( )
{
  cout<<"C++ is better than C";
}
```

"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI\Advanced Programming

```
C++ is better than C
```

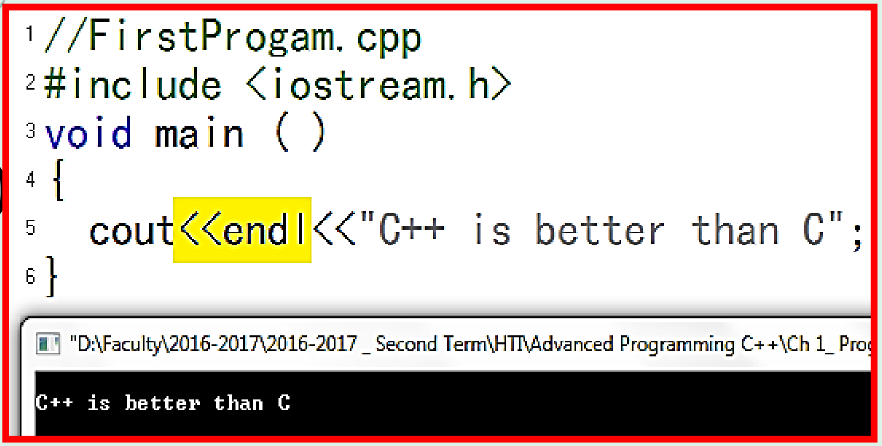


```
//FirstProgam.cpp
#include <iostream.h>
void main ( )
{
  cout<<"\nC++ is better than C";
}
```

"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI\Advanced Programming C++\

```

C++ is better than C
```



```
//FirstProgam.cpp
#include <iostream.h>
void main ( )
{
  cout<<endl<<"C++ is better than C";
}
```

"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI\Advanced Programming C++\Ch 1_ Prog

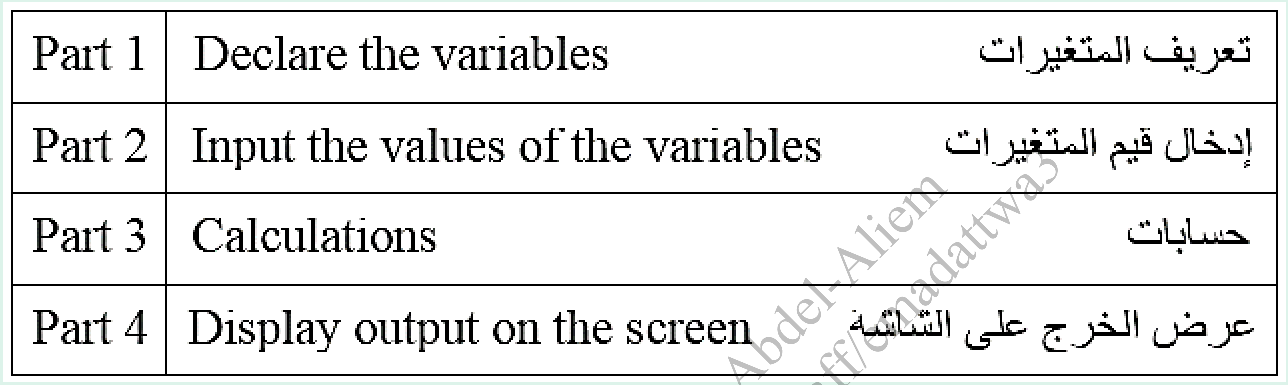
```

C++ is better than C
```

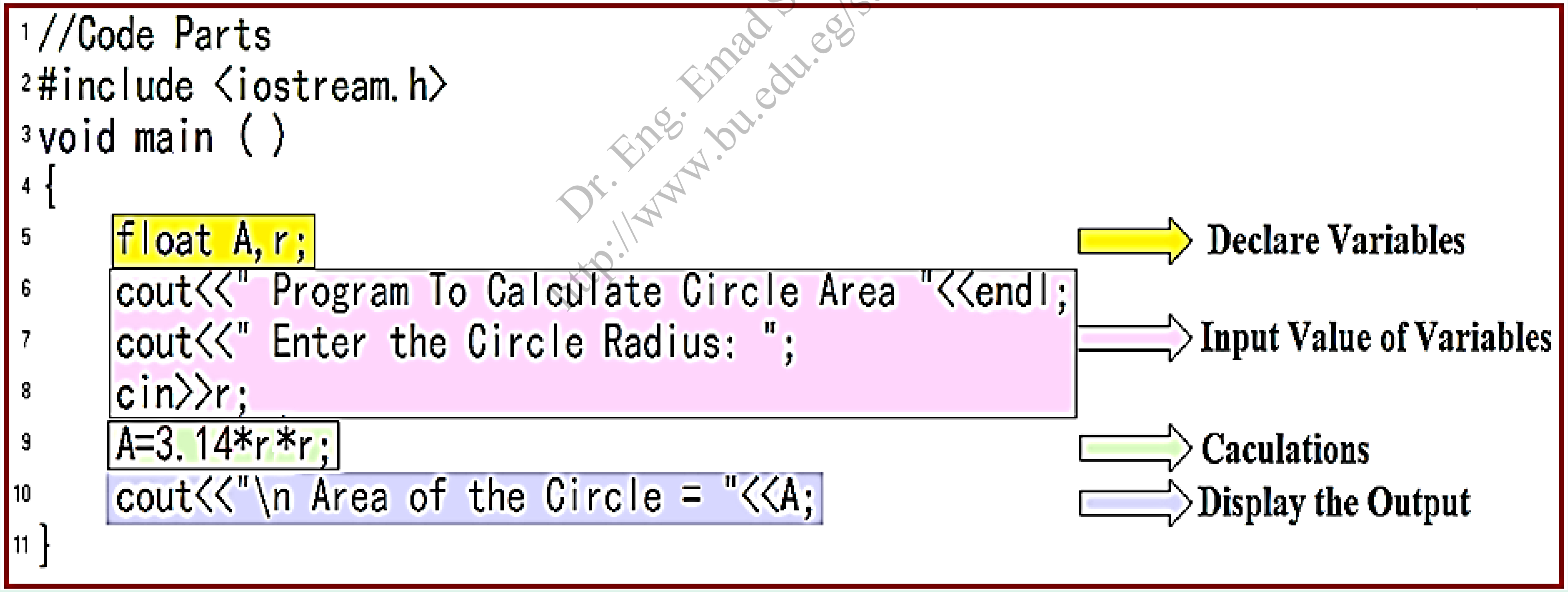
- **cout<<"　　　";**
  this statement causes the phrases between the quotation marks (" ") to be displayed on the screen.
- The symbol (**\n**) means start printing from a new line.
- Each statement inside **main** must ends with a semicolon (**;**).
- (**<<endl**) is a manipulator equivalent to (**\n**).

1-7 Basic Parts of C++ Program …

- The program code is written inside the **main** function and should contain the following parts:



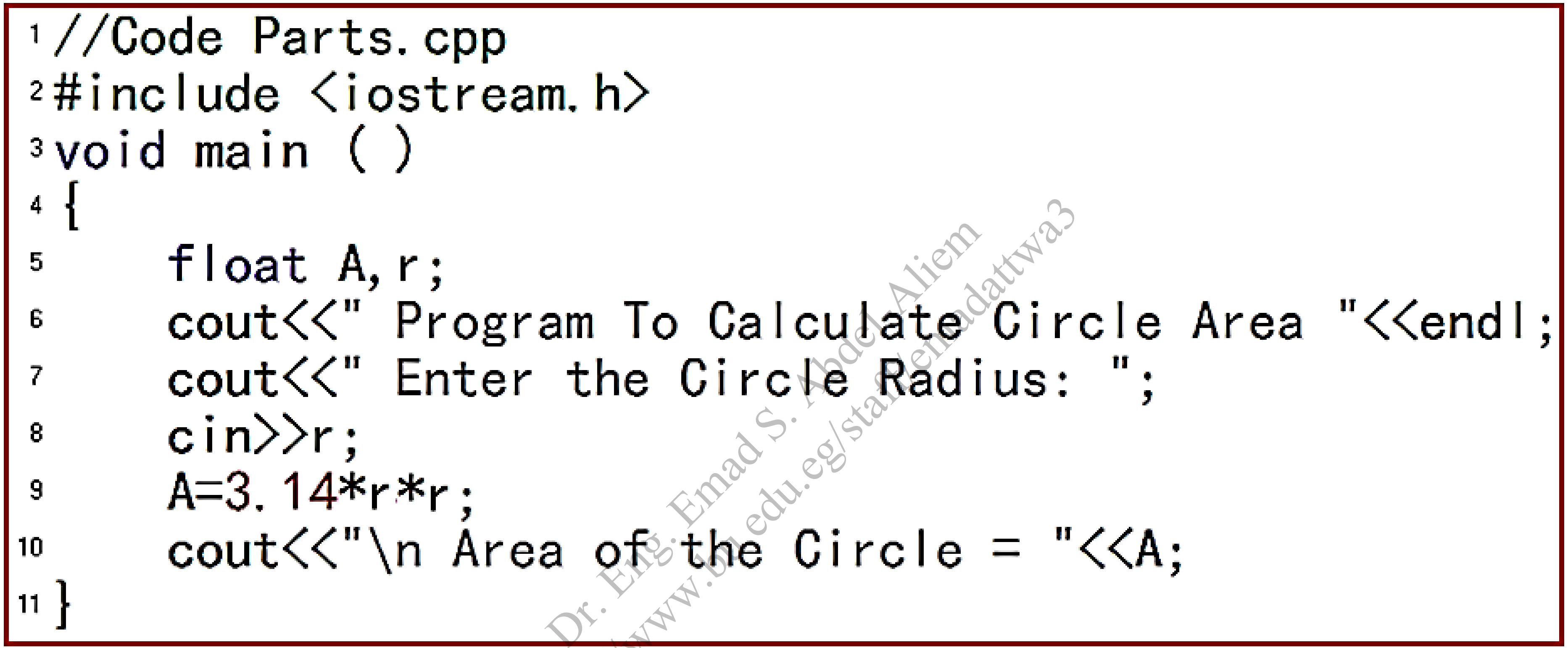
Part 1	Declare the variables	تعريف المتغيرات
Part 2	Input the values of the variables	إدخال قيم المتغيرات
Part 3	Calculations	حسابات
Part 4	Display output on the screen	عرض الخرج على الشاشة



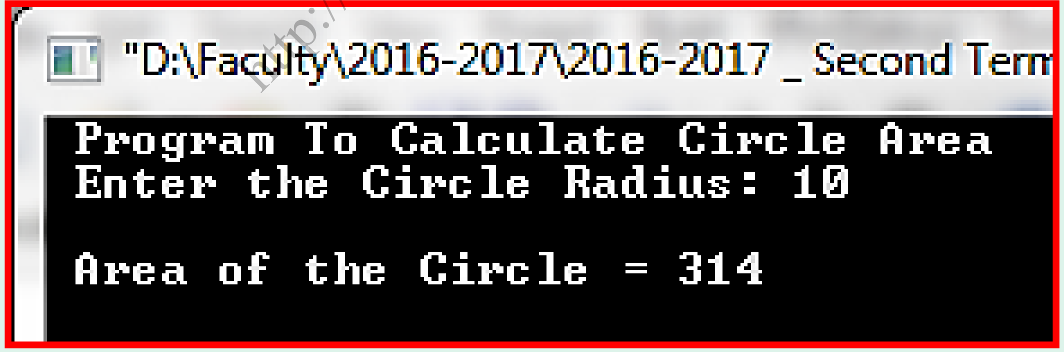
```
//Code Parts
#include <iostream.h>
void main ( )
{
      float A,r;
      cout<<" Program To Calculate Circle Area "<<endl;
      cout<<" Enter the Circle Radius: ";
      cin>>r;
      A=3.14*r*r;
      cout<<"\n Area of the Circle = "<<A;
}
```

1-7 Basic Parts of C++ Program …

- Here is the program and its output:



```
//Code Parts.cpp
#include <iostream.h>
void main ( )
{
	float A,r;
	cout<<" Program To Calculate Circle Area "<<endl;
	cout<<" Enter the Circle Radius: ";
	cin>>r;
	A=3.14*r*r;
	cout<<"\n Area of the Circle = "<<A;
}
```



"D:\Faculty\2016-2017\2016-2017 _ Second Term

```
Program To Calculate Circle Area
Enter the Circle Radius: 10

Area of the Circle = 314
```

1-8 Assignment (1)

2. Variables and Constants

Chapter Objectives:

2-1 What is the Variable in C++ Language?

2-2 Declare (Define) a Variable inside C++ Program

2-3 Rules (Constrains) for Naming a Variable

2-4 Integer Variable (**int**)

2-5 Character Variable (**char**)

2-6 Float Variable (**float**)

2-7 Naming constants and Notes

2-8 Variable Type (**long int**)

2-9 Table of Variable Type and Its Size

2-10 Automatic Data Conversion

2-11 Assignment (2)