Lab Work 1

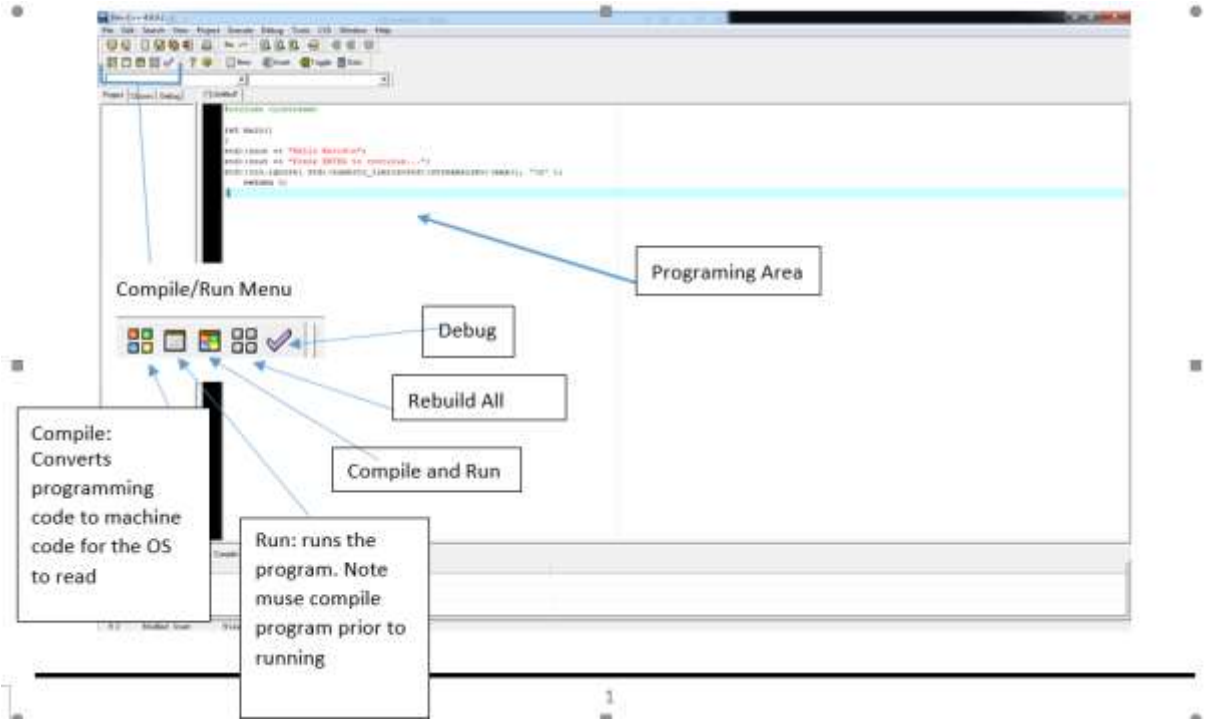
Learning Objectives

1. Walk through the DEV C++ compiler
2. Creating a new source file
3. Basic programming
4. Hello World.cpp
5. C++ input and output
6. Arithmetic operations
7. Formatting

1.1 Walkthrough the DEV C++ Compiler

A compiler takes the code that is written and converts it into machine code for input/output purposes. There are anumber of compilers in the market place. This is just one of many. It is a freeopen source software produced by

1. Load Dev C++ Software
2. Read/Close or Close through the Tips of the Day. Do Not Check the option to "Don't
   Display Tips at Startup"



1.2 Creating a new source file

- File > New > Source File (this will create an area where the program will be written)
- File → save as program1.cpp
- Program1.cpp is the file name and can be named accordingly

1.3 Basic Programming

You can write the codes below using **program1.cpp**
This is the basic structure of the codes. The one indicated in **RED.**
The one indicated in **YELLOW** are comments. Comments are added to clarify codes but they are never compiled and never appears in the output.
#include <iostream>//Calls library for Input/Output for the Operating Systemto configure

using namespace std;//simplifies some of the coding when asking for input or output

int main()/Starts the program. All programs will begin with thiscoding

{//brace represents the program parameter

return 0;//return 0; will send back a zero value to the start ofthe program, thus ending the program

}//the final brace will complete the program. Since we started with a right brace at the beginning of the program, we must end with the opposite brace (left facing) to formally close off the program; similar to what would occur in a algebraic function.

1.4 Hello World.cpp

To input data into a program you must first declare a variable (I.E Int, Char, String, Real, etc) in order to be able to record the user input. The variables should be declared at the beginning of the program below the first brace.
NOTE: theoretically a programmer can declare a variable in anywhere in the program, but this is seen as sloppy and against best practice.
Below there are added lines to the Hello World! Program. Create a new cpp file called HelloWorld.cpp and save it in your Week 1 folder. Add the additional lines in. We will be adding a string variable that will be asking your name.

Test.cpp

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5  char a;
6  string firstname;//declares variable string which
7  //is equal to a word. If a single letter is
8  //desired for entry the variable type would be a char.
9  cout << "Hello World" << endl;
10 cout << "What is your name?";
11 cin >> firstname;
12     // cin>> allows user to type in info and assign to a variable
13 cout << "Hello " << firstname << " my name is PC" << endl;
14 // provides text with embedded version ofvariable. endl;
15 //drops the next command to a line below. Try removing and see what happens.
16 cin >> a;
17
18 return 0;
19 }
20
```

1.5 C++ input and output

[*] Test.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
char a; string firstname; int age;
cout << "Hello World"
<< endl;
cout << "What is your name?";
cin >> firstname;
cout << "Hello " << firstname << " my name is PC" << endl;
//please ask the user for their age
if (age <= 40)
{

cout <<"You are a youngster" << endl;
}
else
return 0;
}

```

1.6 Arithmetic operations

Operator	Meaning	Type	Example
+	Addition	Binary	`total = cost + tax;`
-	Subtraction	Binary	`cost = total - tax;`
*	Multiplication	Binary	`tax = cost * rate;`
/	Division	Binary	`salePrice = original / 2;`
%	Modulus	Binary	`remainder = value % 3;`

Try opening a new cpp file. Write appropriate codes to test the operators listed in the tables above. Name this file as arithmetics.cpp in Week 1 folder.

Test.cpp

```cpp
// This program calculates the circumference of a circle.
#include <iostream>
using namespace std;

int main()
{
    // Constants
    const double PI = 3.14159;
    const double DIAMETER = 10.0;

    // Variable to hold the circumference
    double circumference;

    // Calculate the circumference.
    circumference = PI * DIAMETER;

    // Display the circumference.
    cout << "The circumference is: " << circumference << endl;
    return 0;
}

```

1.7 Formatting

Formatting output in C++, is important in the development of the output screen, which can be easily read and understood. C++ offers the programmer several input/output manipulators. Two of these (widely used) I/O manipulators are:

setw()
setprecision()

In order to use these manipulators, you must include the header file named iomanip.h. Here is an example, showing how to include this header file in your C++ program.

#include<iomanip.h>

The setw() Manipulator

In C++, the setw() manipulators sets the width of the field assigned for the output. It takes the size of the field (in number of characters) as parameter. Here is an example, this code fragment:

cout<<setw(6)<<"R";

generates the following output on the screen (each underscore represents a blank space).

_ _ _ _ _R

The setw() manipulator does not stick from one cout statement to the next. For example, if you want to right-justify three numbers within an 8-space field, you will need to repeat setw() for each value, as it shown below:

cout<<setw(8)<<22<<"\n";
cout<<setw(8)<<4444<<"\n";
cout<<setw(8)<<666666<<endl;

The output will be (each underscore represents a blank space):

_ _ _ _ _ _ 2 2
_ _ _ _ 4 4 4 4
_ _ 6 6 6 6 6 6

C++ Formatting Output Example

Here are some example program demonstrating, how to format the output screen in C++

```
/* C++ Formatting Output - The setw() Manipulator */

#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
  clrscr();
  int i, num;
  cout<<"Enter a number: ";
  cin>>num;
  cout<<"\nTable of "<<num<<" is:\n\n";
  for(i=1; i<=10; i++)
  {
    cout<<num<<setw(3)<<"*"<<setw(4)<<i<<setw(4)<<"="<<setw(4)<<num*i<<"\n";
  }
  getch();
}
```

c++ formatting output

Here another type of C++ program, also demonstrating, output formatting in C++

```
/* C++ Formatting Output - The setw() Manipulator */

#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
  clrscr();
  int i;
  long int num;
  cout<<"Enter a number: ";
  cin>>num;
  cout<<"\nMultiplying (by 5) and printing the number 10 times with 3 columns:\n";
  for(i=0; i<10; i++)
  {
    cout<<num<<setw(25)<<num<<setw(25)<<num<<"\n";
    num = num * 5;
  }
  getch();
```

}

The setprecision() manipulator

In C++, the setprecision() manipulator sets the total number of digits to be displayed when floating-point numbers are printed. Here is an example, this code fragment:

cout<<setprecision(5)<<123.456;
will print the following output to the screen (notice the rounding) :

123.46
The setprecision() manipulator can also be used to set the number of decimal places to be displayed. In order for setprecision() to accomplish this task, you will have to set an ios flag. The flag is set with the following statement :

cout.setf(ios::fixed);
Once the flag has been set, the number you pass to setprecision() is the number of decimal places you want displayed. The following code:

cout.setf(ios::fixed);
cout<<setprecision(5)<<12.345678;
generates the following output on the screen (notice no rounding):

12.34567
Additional IOS flags
In the statement:

cout.setf(ios::fixed);
"fixed" i.e., ios::fixed is referred to as a format option. Other possible format options can be one of the following :

Format Value	Meaning
left	left-justify the output
right	right-justify the output
showpoint	displays decimal point and trailing zeros for all floating point numbers, even if the decimal places are not needed
uppercase	display the "e" in E-notation as "E" rather than "e"
showpos	display a leading plus sign before positive values
scientific	display floating point numbers in scientific ("E") notation
fixed	display floating point numbers in normal notation - no trailing zeroes and no scientific notation

// display money
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout<<setprecision(2);
cout<<5.8;
Please note that all the subsequent couts retain the precision set with the last setprecision(). That means setprecision() is "sticky". Whatever precision you set, sticks with the cout device until such time as you change it with an additional setprecision() later in the program.