

Exercise 2a: Using Code::Blocks

Learning Outcomes:

The student should gain the following basic knowledge:

- Use Code::Blocks
- Edit / Compile programs
- Single step & watch variables

Installing Code::Blocks

Download the program from [eLearning@SP->ET0083->Learning Resources->Resources->CodeBlocks](#). Unzip and install the program. Use default setup during installation.

Using Code::Blocks

There are four steps to writing a program using CodeBlocks.

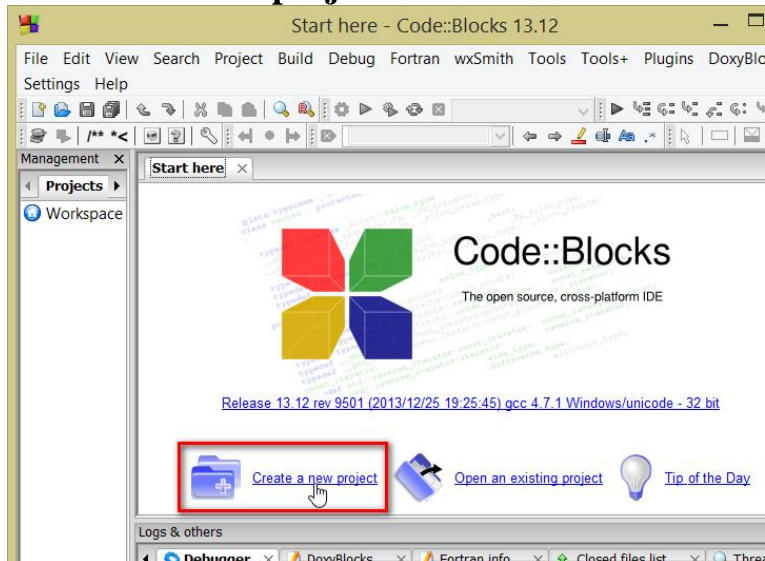
1. Create a new project : every program needs a project
2. Enter the C++ codes
3. Build project, (and if no errors)
4. Run the program

To Start CodeBlocks:

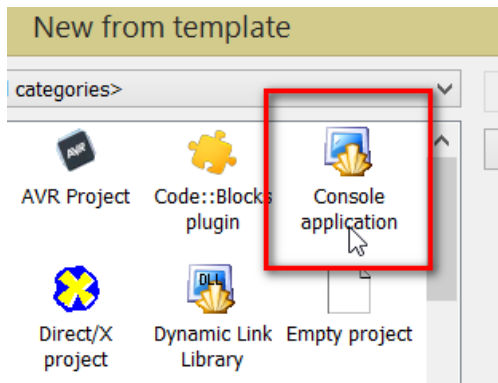


Look for the icon and double click it.

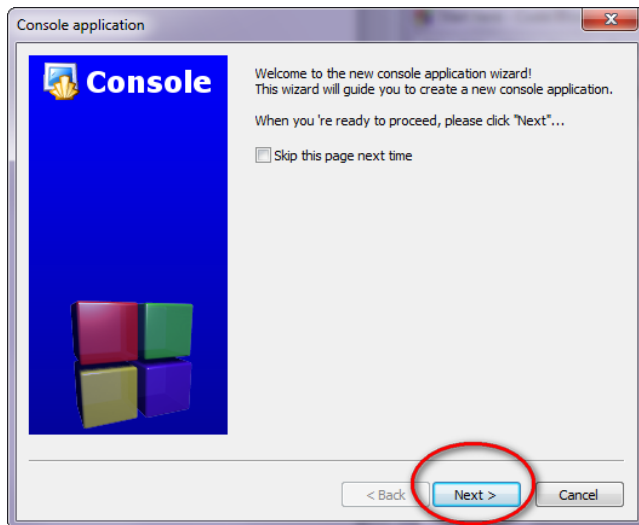
A. Create a new project



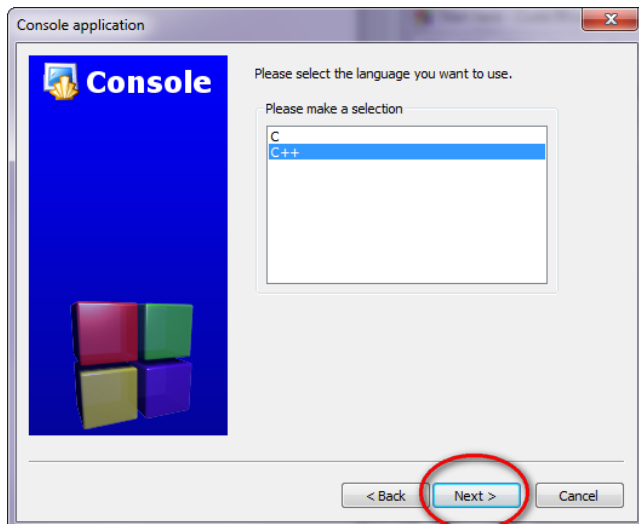
B. Select Console application and click the Go button



C. Click the Next> button



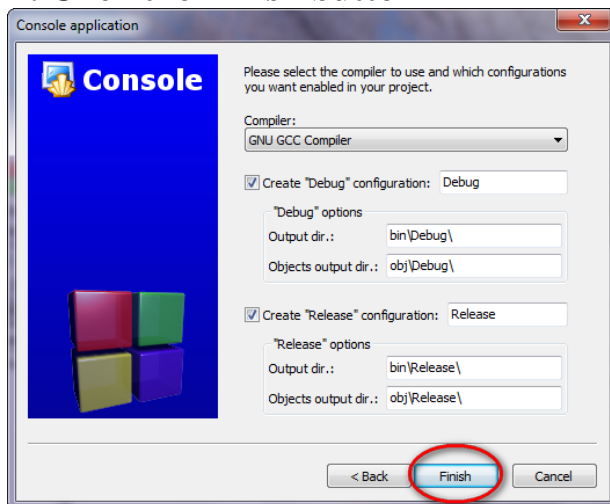
D. Click the Next> button



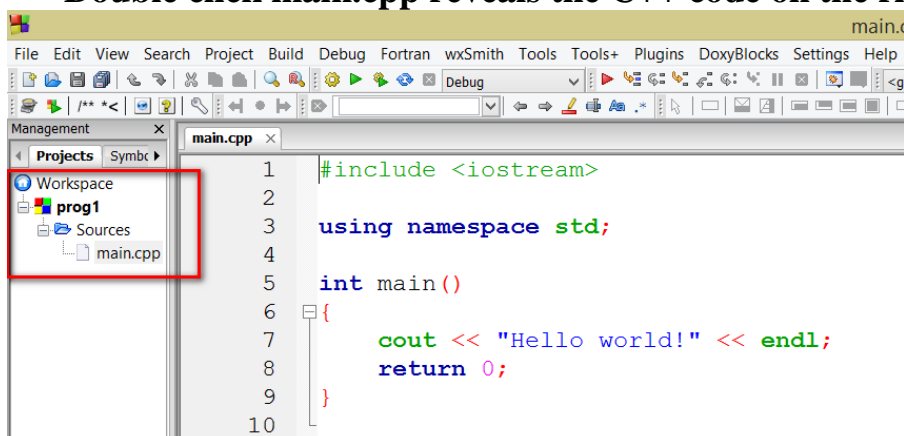
E. Enter the Project title "prog1". You have to select a folder where your project is to be stored in.



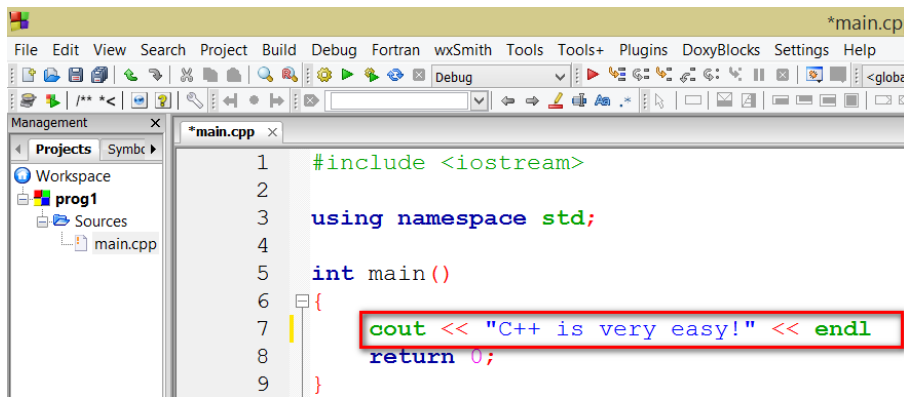
F. Click the Finish button



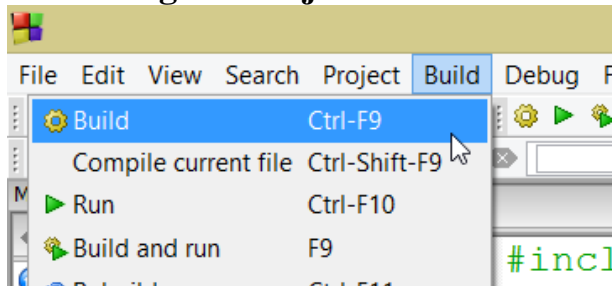
G. Double click Sources on the left panel and you should see main.cpp. Double click main.cpp reveals the C++ code on the right panel



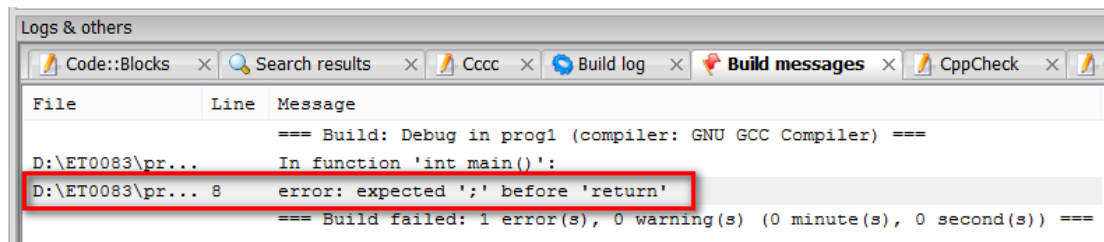
H. Modify the program as shown below



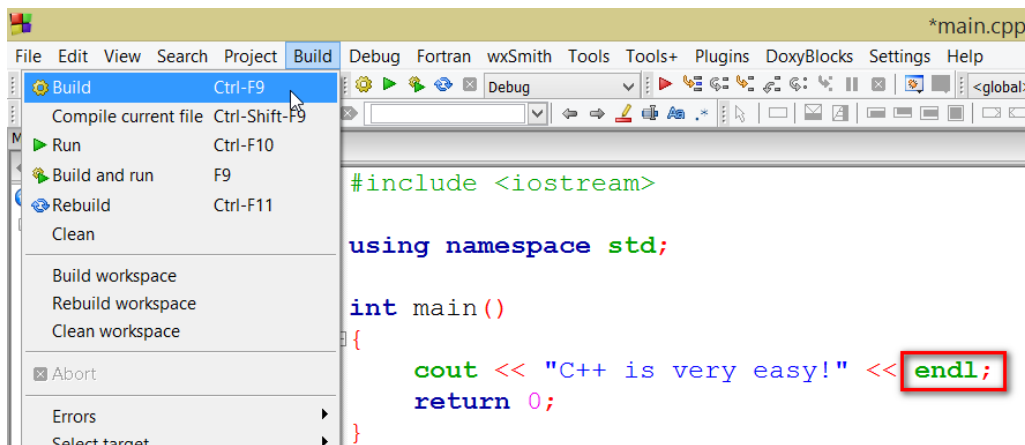
I. Building the Project



Before we can execute the program, we must build it first.

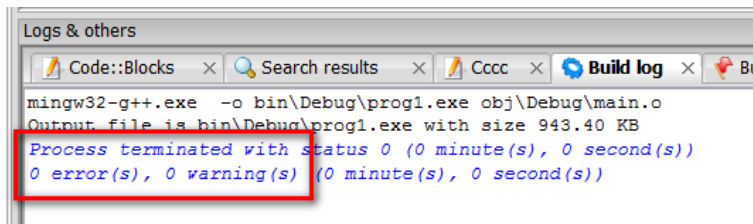


There should be 1 error (missing semicolon after the endl).

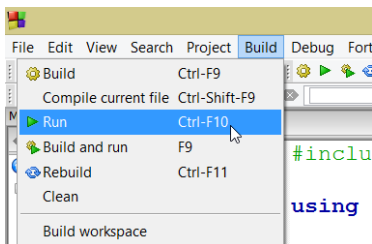


Place a semicolon after endl and build again.

You should have 0 errors.



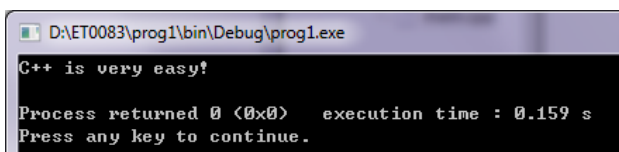
J. Run the program



Once there are no errors, we are ready to RUN or execute the program.

The result of this program is the statement,

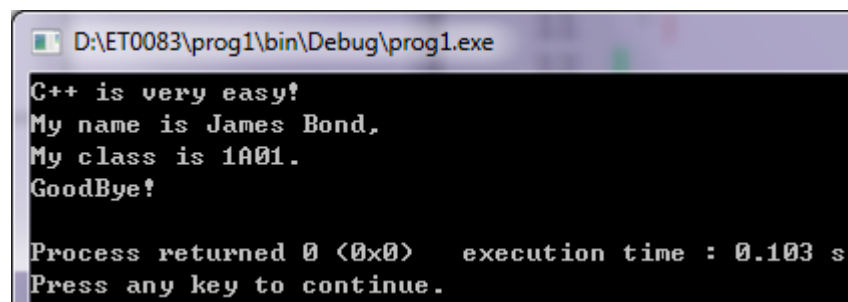
C++ is very easy!



“Process returned 0 (0x0) execution time: 0.159s. Press any key to continue.” is a prompt generated by the system.

Activity: Introduce yourself

Modify the program to produce the following result:



Edit / Compile Programs

Create a new project in CodeBlocks call “prog2” and type in the following codes:

```
#include <iostream>
using namespace std;

int main()
{
    int value;
    cout << "Enter a decimal value : ";
    cin >> value;
    cout << "The value entered is " << value ;
    return 0;
}
```

Build and run program to see the result.

Spelling Errors:

1. Spell the word "**decimal**" as “**desimal**”. Re-build your program. Are there any error messages? If not, why?
2. Spell the word "**cout**" as “**Cout**”. Re-build your program. Record the error messages.

Multiple Errors:

1. Make two errors:
 - (i) Spell one of the word “**value**” as “**values**” and
 - (ii) Delete any semicolon.

Re-build your program. Record the error messages.

Ambiguous errors

Remove the " before Enter in the following statement:

```
cout << "Enter a decimal value : ";
```

1. Look at the error messages. Note that one mistake had generated so many error messages.
2. Briefly explain the importance of the " you removed.

Single Step and Watch Variables

Run time errors and warnings

Some errors cannot be detected at all during compilation. Such errors affect the program only when the program is executed. These errors are called **run time errors** or **logical errors** and may result in error messages, wrong outputs or wrong operation of the computer.

Warnings do not normally stop the compiler thus they too may result in run time errors. The following activity explores run time errors.

Create a new project in CodeBlocks call “prog3”.

```
#include<iostream>
using namespace std;

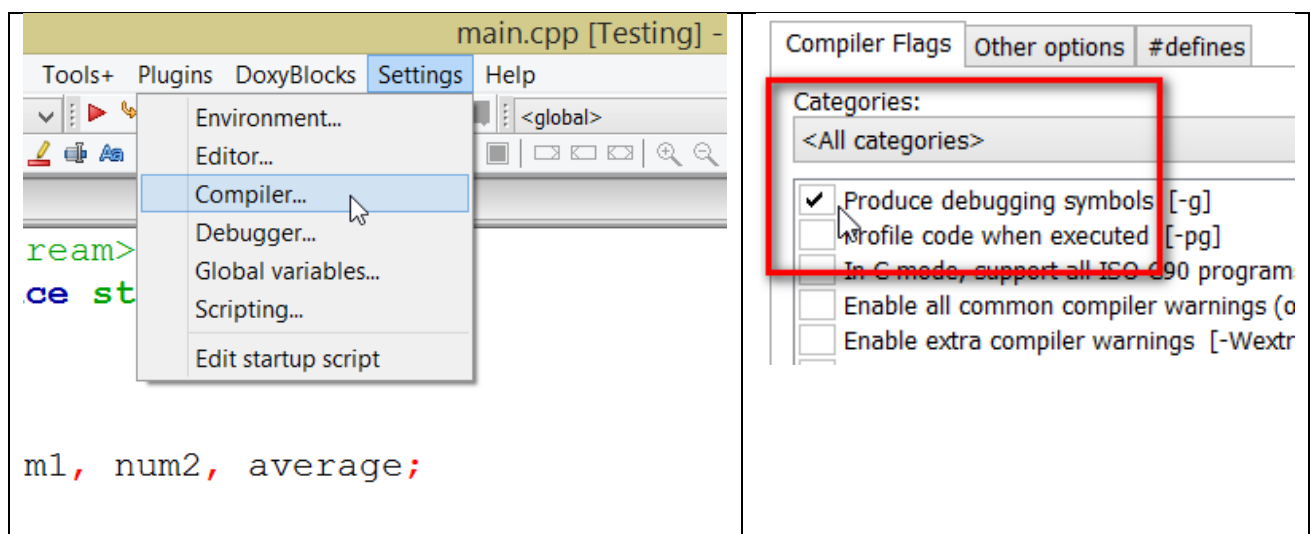
int main()
{
    double num1, num2, average;

    num1 = 10.0;
    num2 = 20.0;
    average = num1 + num2 / 2;
    cout << "Average value is " << average << "\n" ;
    return 0;
}
```

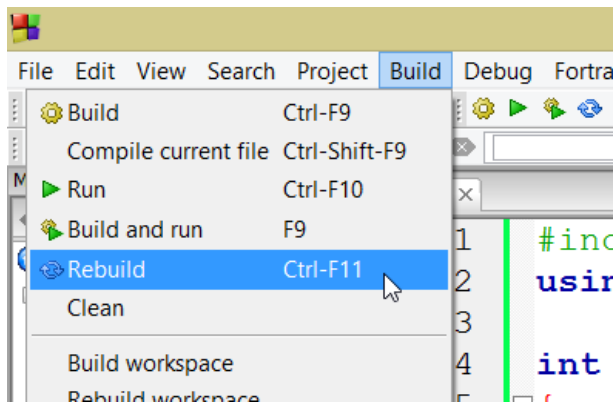
Build and run program to see the result.

The result of 20 is not correct, it should be 15.

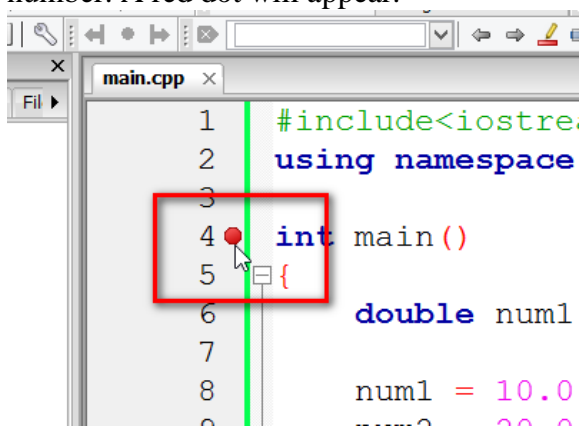
There is run time error in this program. Single stepping can be used to determine the location of the erroneous code statement(s). To debug a program by single stepping, the compiler options need to be changed.



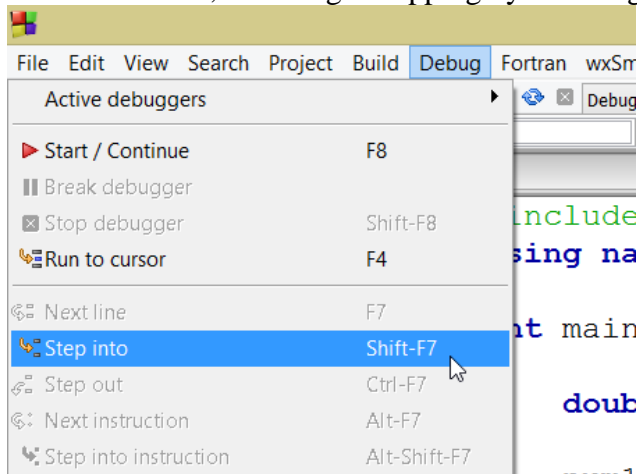
Then do a Build -> Re-Build



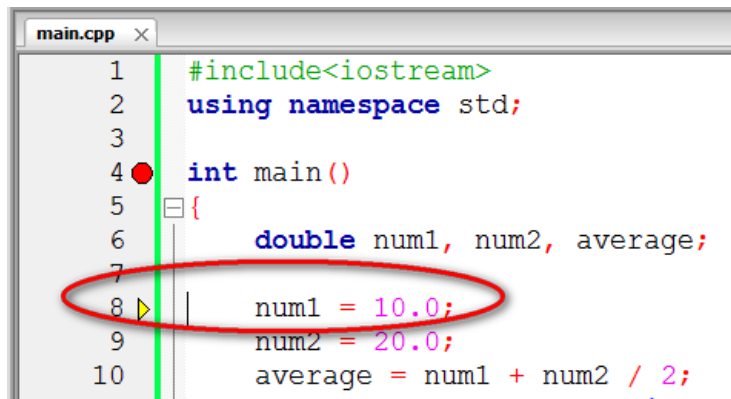
Before debugging, set a break point that the main function by clicking beside the line number. A red dot will appear.



Instead of RUN, start single stepping by selecting Debug->Step into or press **Shift F7**.

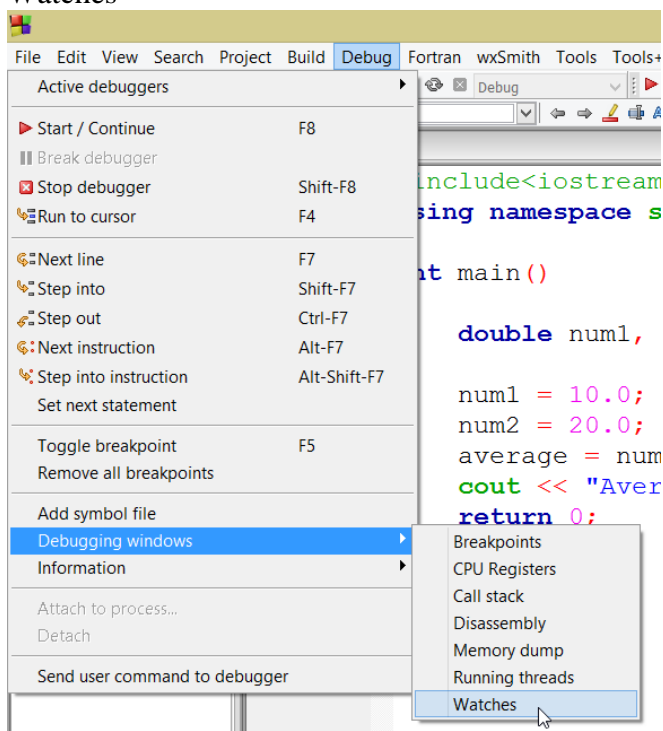


Press **F7** and a yellow triangle indicate that the program is at the starting position.

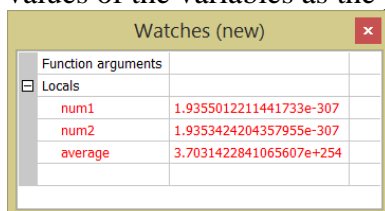


```
1 #include<iostream>
2 using namespace std;
3
4 int main()
5 {
6     double num1, num2, average;
7
8     num1 = 10.0;
9     num2 = 20.0;
10    average = num1 + num2 / 2;
```

If the Watch window is not enable, turn it on by selecting Debug → Debugging windows → Watches

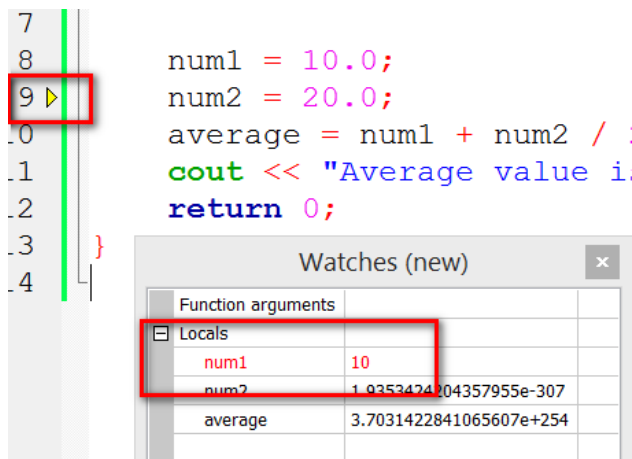


Click on Local Variables in the Watches window to display all the local Variables to see the values of the variables as the program is single stepping.

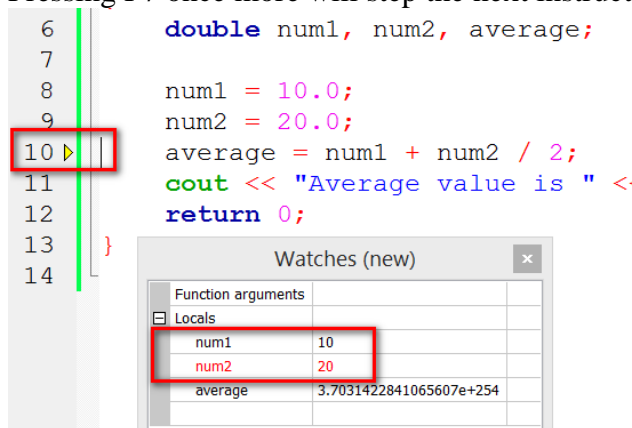


Watches (new)	
Function arguments	
Locals	
num1	1.9355012211441733e-307
num2	1.9353424204357955e-307
average	3.7031422841065607e+254

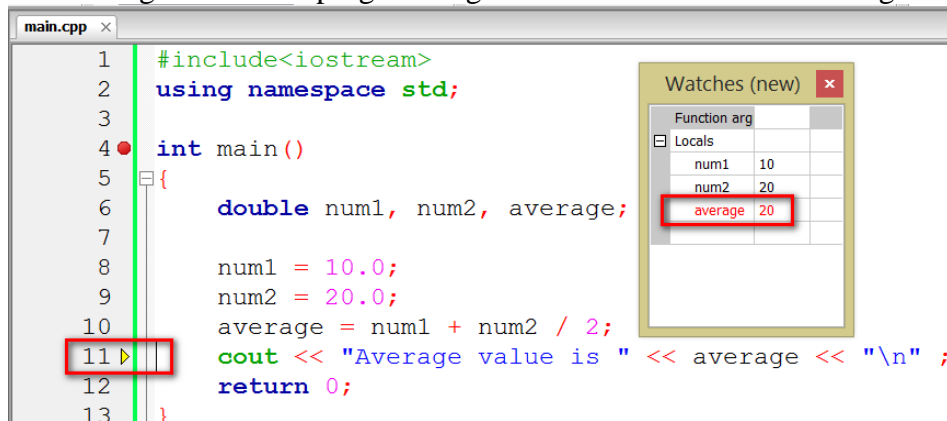
At the beginning of the program, the variables contain random values. Press F7 once and the statement at the yellow triangle will be executed. The yellow triangle will move to the next statement. Pressing F7 again will execute that statement and move the yellow triangle to the next line. At the same time, the affected variable will be updated whenever a statement is executed. Press F7 until the value of num1 in the Watches changes to 10.



Pressing F7 once more will step the next instruction.



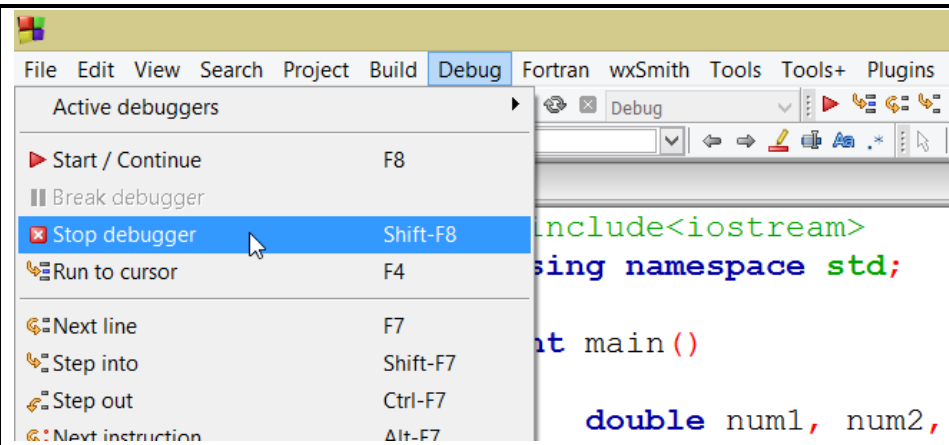
Press F7 again until the program to get the calculated result in average.



Correct error and re-do single stepping

Can you see where the problem is?

To stop the debugger,



Edit the program in order to get the correct result.

Exercise 2b: Input / Output Operations & Data Types

1. What are the characters that indicate the **begin** and **end** of instructions in a C++ Program?

Where is the error?

Create a new project and call it “prog4”

```
#include<iostream>
using namespace std;

int main()
{
    double radius, PI;
    int area;

    radius = 12.5;
    PI = 22 / 7;
    area = PI * radius * radius;
    cout << "Area of circle is " << area << "\n" ;
    return 0;
}
```

Build and run the program to see the result.

The correct value for the area is 491.071.

Where is/are the error(s)?

2. What is the difference between **variables** and **constants**?
Name two data items, which are best represented as constants and two that are best represented as variables.
3. How much memory (in bytes) is occupied in each of the following statements?

```
double number;
int count, index;
char yesno;
```

4. Which of the following variable declarations are correct? If a variable declaration is not correct, give the reason(s) and provide the correct variable declaration.
- a. `n=12;`
 - b. `char letter =;`
 - c. `int one = 5, two;`
 - d. `double x, y, z;`
5. Spot and correct the errors in the following code:

```
#include <iostream>
using namespace std;
#define PI=3.14;
int Main()
{
    int count;
    double colorVal

    cout << "Enter an integer number;
    cin << "count";
    colorVal = PI*count;
    cout << "\nThe color value is : "
        << colorVal;
    return 0;
}
```

6. In the following questions, apply the design & development process. Give both the first and second level pseudocode and flowcharts.

a. The volume of a cylinder is given by the following equation:

$$\text{volume} = \pi r^2 l$$

r is the radius
l is the length of the cylinder.

Write a program that will prompt the user to enter the radius and the length of the cylinder. The program will calculate and display the volume.

b. Enhance the above program such that it also calculates and displays the total surface area of the cylinder.

The total surface area of the cylinder is given by the following equation:

$$\text{total surface area} = 2\pi r^2 + 2\pi r l$$