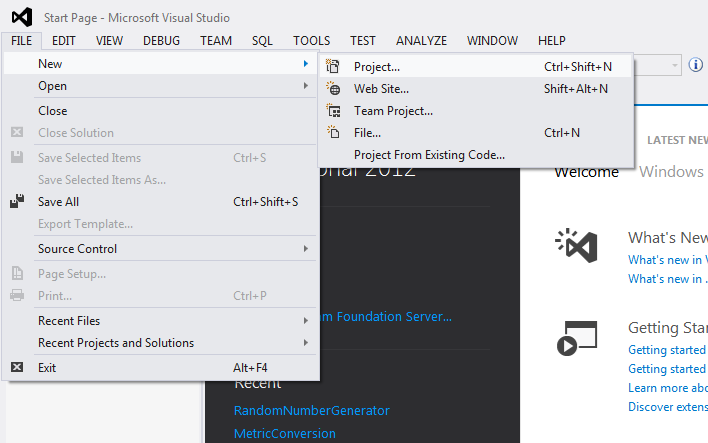
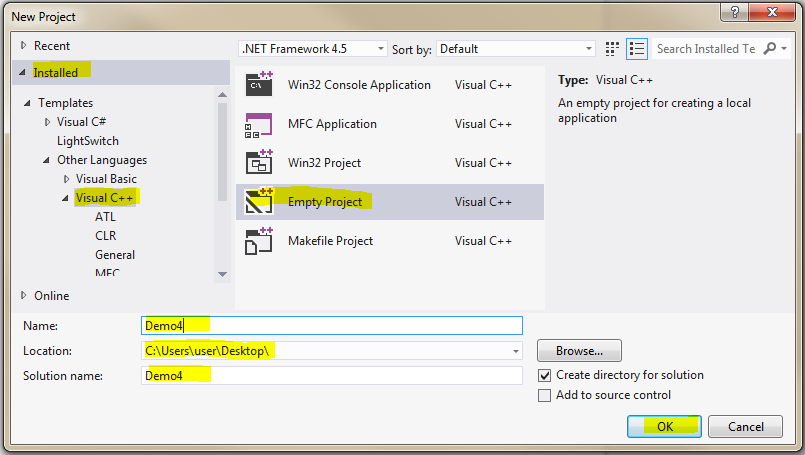
FORMATING OUTPUT and STRINGS Demo

Video is at <https://youtu.be/om9uVGWR44w>

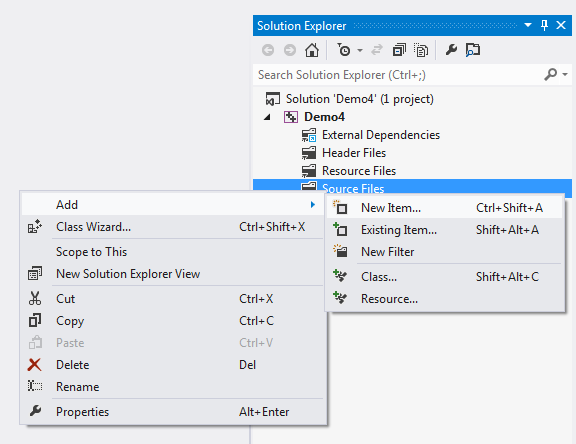
1. Start our program
   1. Start Visual Studio
   2. Create a new empty C++ project:



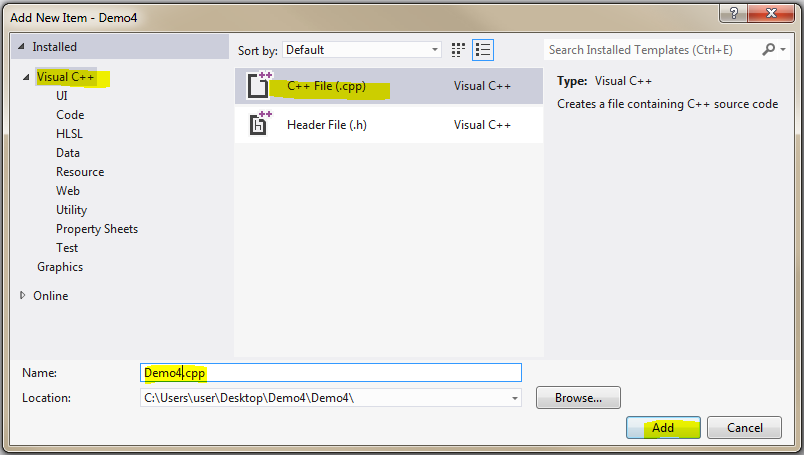
* 1. Call it “Demo4”, pick the desktop as the location and click OK.



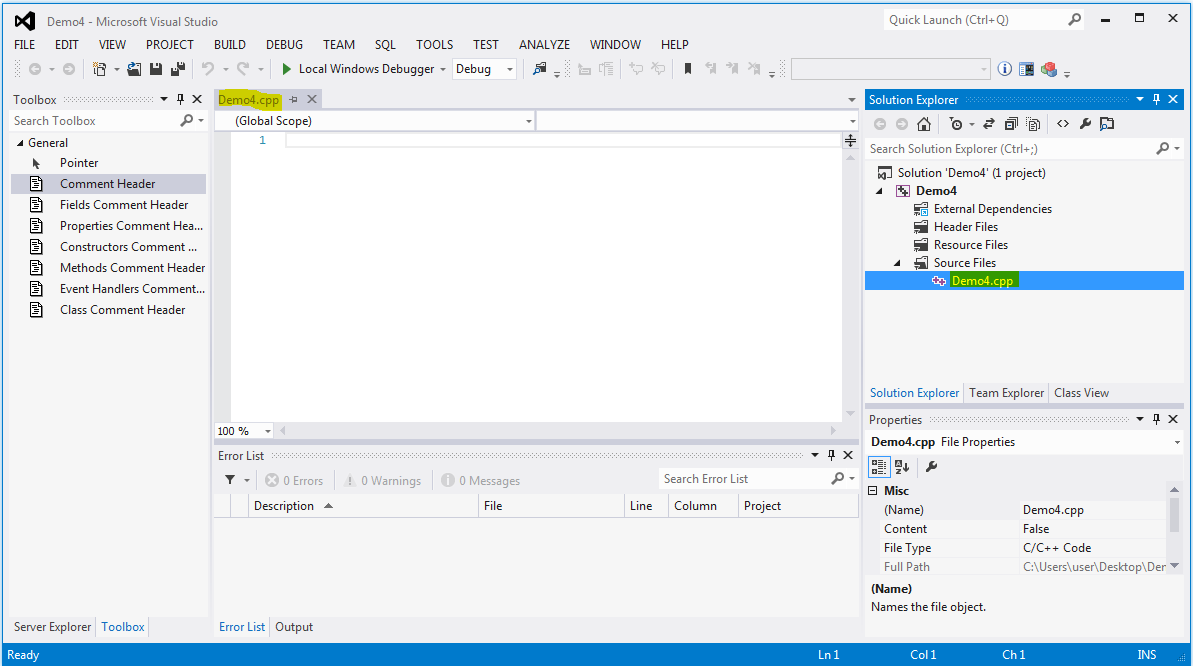
* 1. Click on Source Files and Add|New Item:



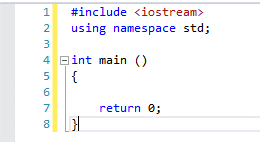
* 1. Select Visual C++, C++ File (.cpp), call it “Demo4”, then click “Add”:



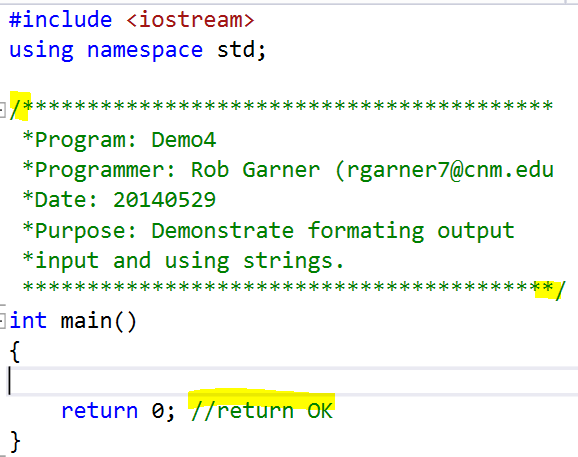
* 1. You should now have “DeclarationDemo.cpp” in the solution explorer and it should be open in the main window:



* 1. Enter the following code to start our program:

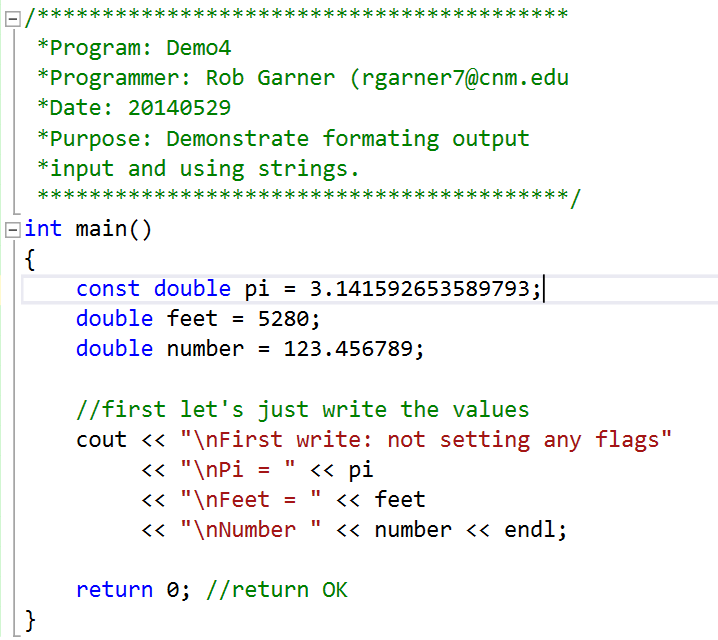


* 1. Take a look at this code and memorize it. We will be starting all of our programs this way. We #include the iostream library so we can send text to the screen and read text from the keyboard. We use namespace std so we can deconflict with other files we may create later. We then have to have a main function so the operating system will know where to start our program. Main will return an int of 0 so the operating system knows our program executed OK. Let’s add some comments:

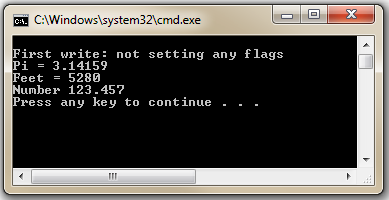


* 1. Comments can be either block comments that start with “/\*” and end with “\*/” or single line comments that just start with “//”. The picture above highlights the characters that designate the beginning and end of the block comment at the top and the beginning of the single line comment at the bottom. Single line comments don’t need anything specific at the end because everything to the end of the line is considered part of the comment.
  2. Comments do not affect execution of the program. They are just there to help you and, more importantly programmers who may have to modify your code in the future, to understand how your code works. You must comment you code!
  3. You should have a comment block at the start of the program with:
     1. Name of the program
     2. Your name and email address
     3. Date
     4. Purpose of program
  4. Points will be taken off of programs that are not adequately commented.

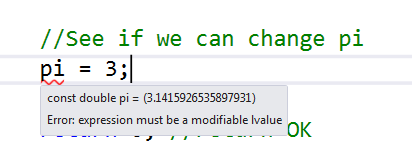
1. Work with IOManipulators
   1. Write some values:
      1. Add the following code:



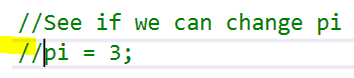
* + 1. Test it by clicking on Ctrl-F5. Compile if required.



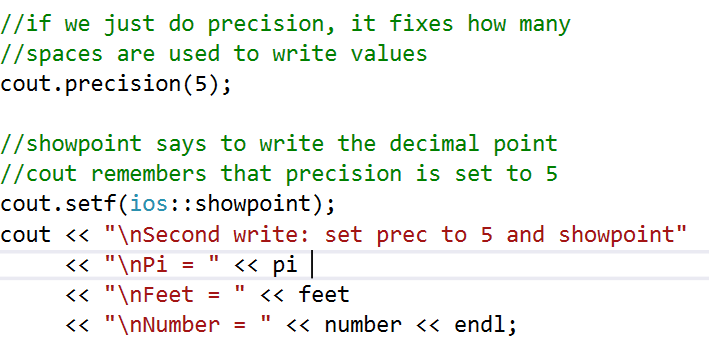
* 1. See if we can modify pi:
     1. Add the following code:



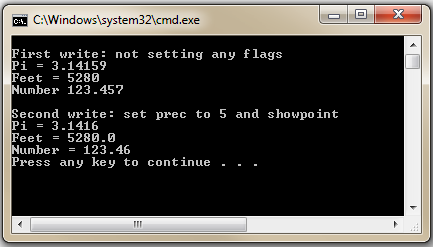
* + 1. Notice that we get an error! Since pi was declared as a “const” we can not change it later in the program the way we can change other variables.
    2. Comment out or delete the code in 12.1:



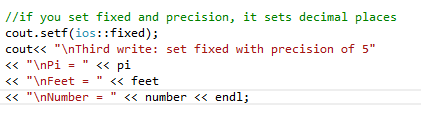
* 1. Set precision:
     1. Add the following code:



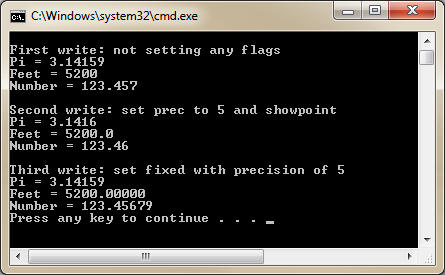
* + 1. Try it with Ctrl-F5



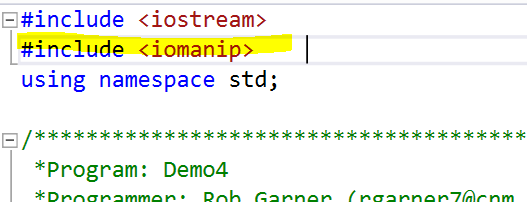
* + 1. Notice that cout.precision(5) and cout.setf(ios::showpoint) change the output so at least five points of precision are displayed.
  1. Using ios:fixed
     1. Add the following code:



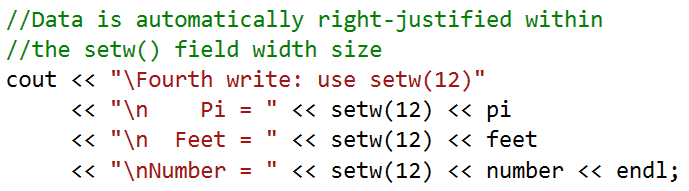
* + 1. Try it with Ctrl-F5



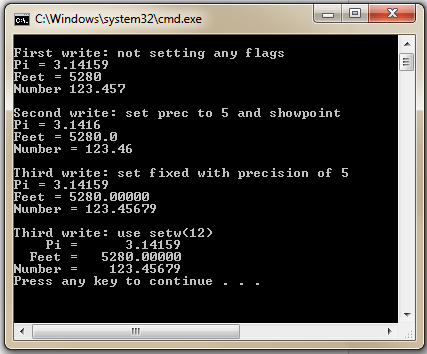
* 1. Using set(w)
     1. Add an include iomanip at the top of the program:



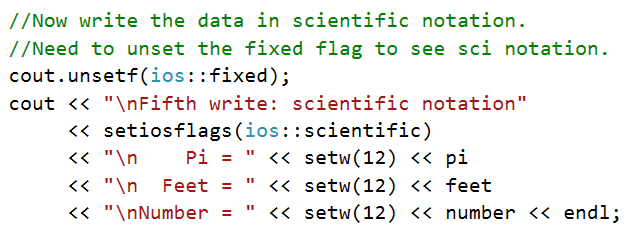
* + 1. Back in main add the following code:



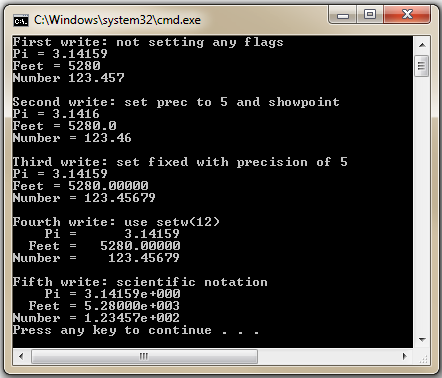
* + 1. Try it with Ctrl-F5



* + 1. Notice that setw(12) sets the number of characters used to display number to 12. Empty leading spaces are added in front of the number in order to make the number take up 12 spaces.
  1. Unsetf and scientific notation flag.
     1. Enter the following code:

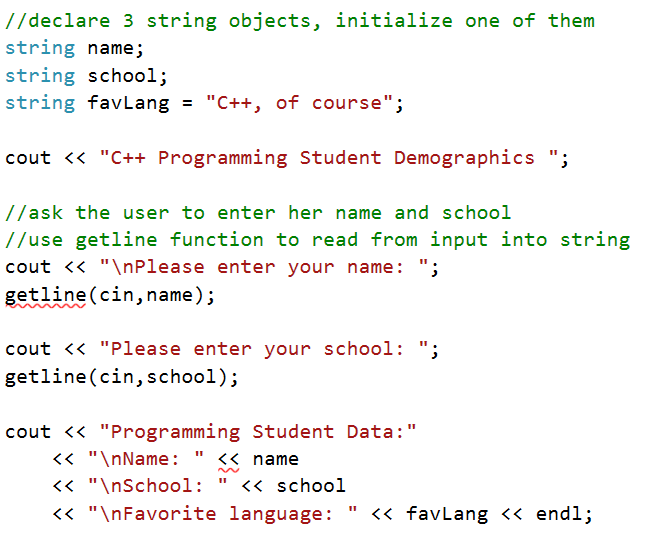


* + 1. Test it with Ctrl-F5:

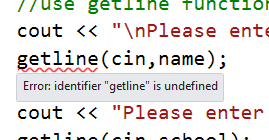


* + 1. Notice that the numbers are now displayed in scientific notation!

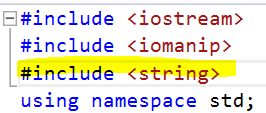
1. Intro to Strings
   1. Add the following code:



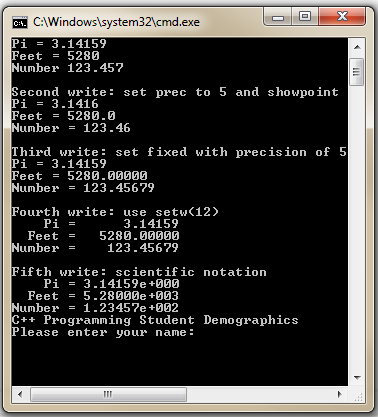
* 1. Notice that the variables are declared as strings. String is a new type that can store more than one character, in a sense a “string” of characters. The getline function is passed cin and a string variable as arguments using the syntax: getline (cin, <Variable Name>). In this case we coded it as getline(cin, name).
  2. Hover over your cursor over the red squigles of getline:



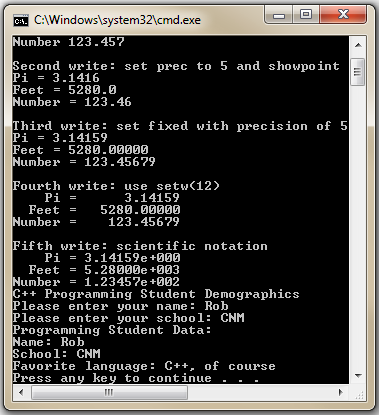
* 1. Notice that getline is currently “undefined”. This is because we are missing an include statement.
  2. At the top of the program add another include statement:



* 1. You should see the red squiggle disappear as well as the error.
  2. Try the program with Ctrl-F5:

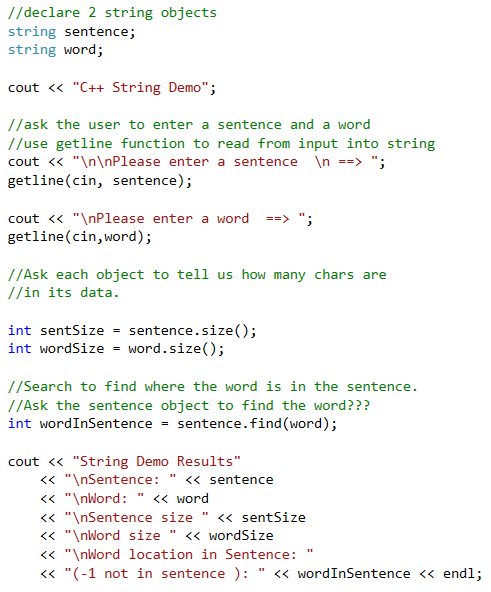


* 1. Notice the program now asks for user input. Answer the questions:



* 1. The program will take the user inputs and then display a message using those inputs. Notice that favLanguage is set within the program. You don’t get to pick your favorite language, the program picks it for you☺

1. Using string functions:
   1. Add the following code:



Adding .find(searchString) to the identifier of a string variable will return the position of searchString if it is located in that string.

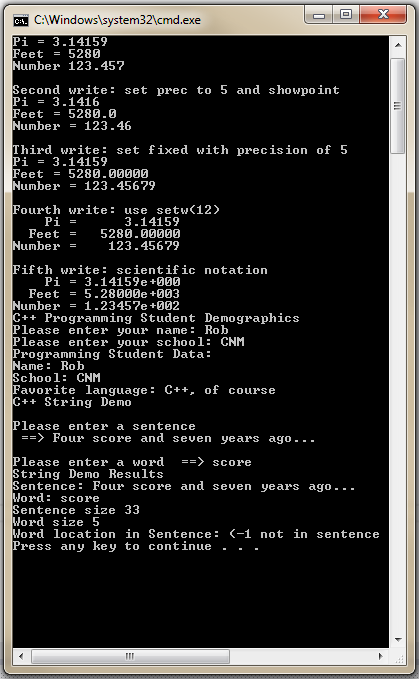
Use getline to fill sentence variable with user input.

Use getline to fill word variable with user input.

Adding .size() to the identifier of a string variable will return the size of sentence as an int.

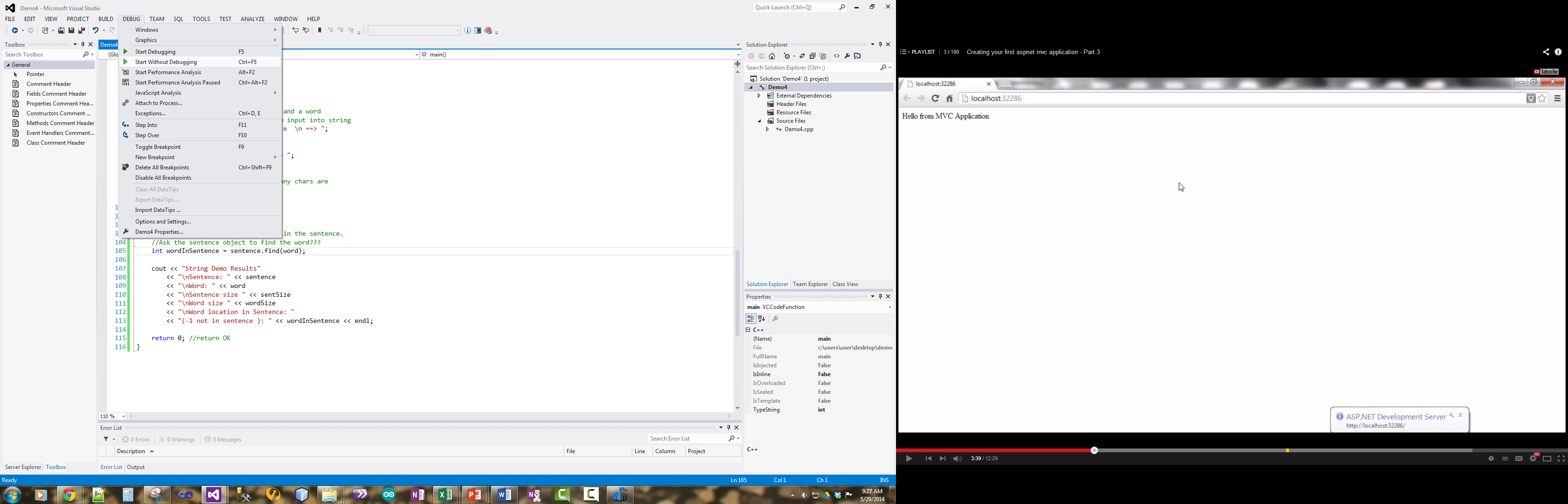
Declare some strings.

* 1. Notice how we use the dot operator “.” to access functions that are unique to the string class!
  2. Try it with ctrl-F5:

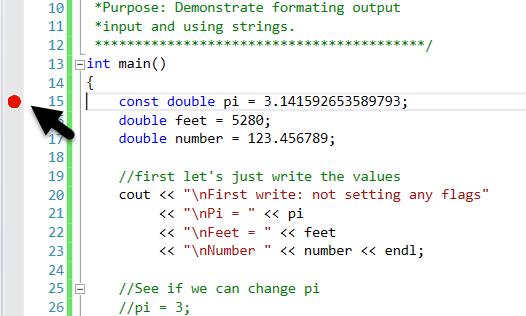


* 1. See how this code finds a word in a larger sentence.

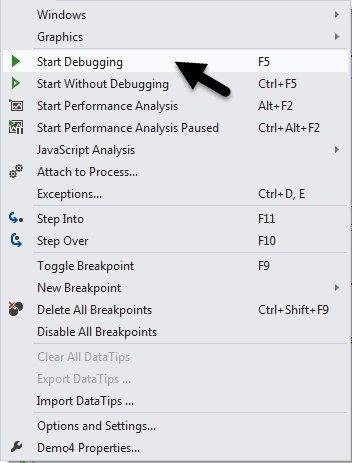
1. Using the debugger to understand our code.
   1. Up until now we have been running our program using Ctrl-F5 or DEBUT|Start Without Debugging off of the menu



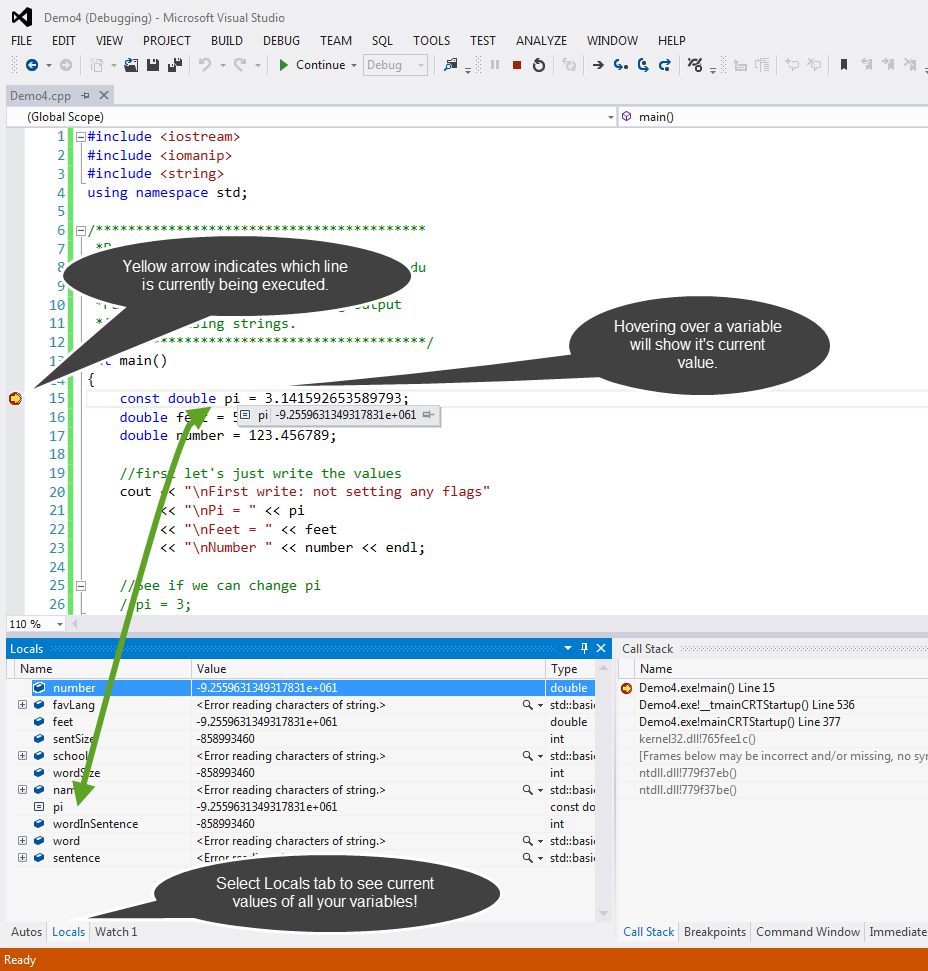
* 1. To use debugging with must first set a “break point”. A break point will halt execution at the line of code where it is set and allow us to “Step through” execution a line at a time.
  2. On the right side of your code is a gray bar. Click on it where pi is declared at the top of the program:



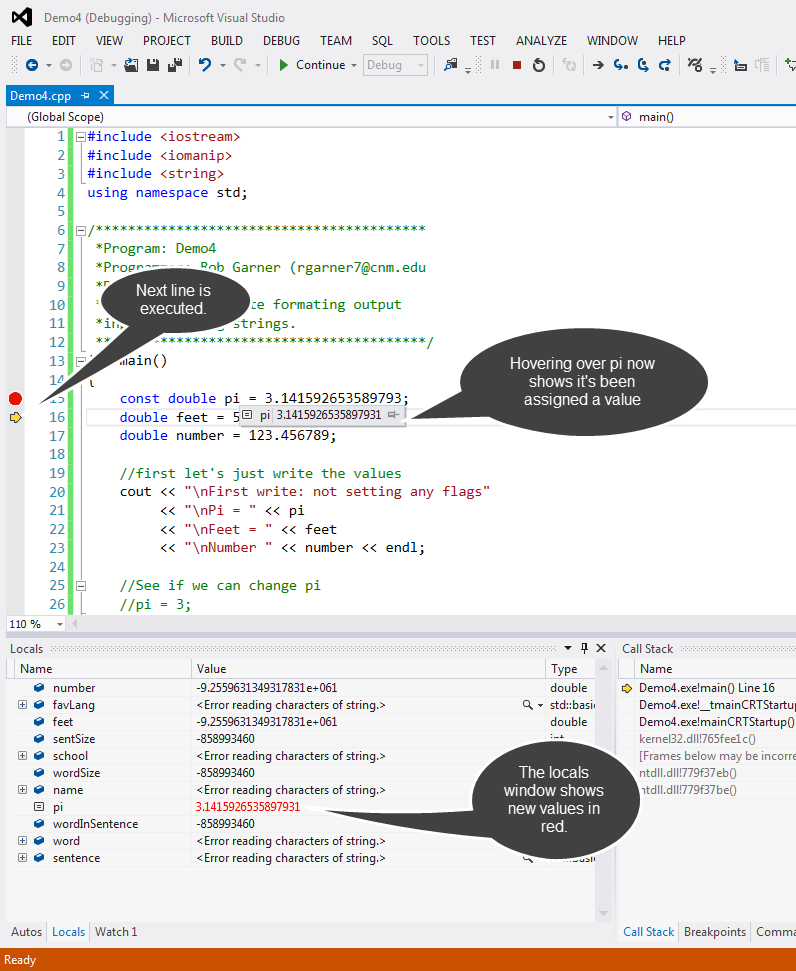
* 1. Next click on Start Debugging instead or press F5 (without Ctrl).



* 1. You will notice visual studio will change appearance. The status bar at the bottom will turn orange, a yellow arrow will appear over your break point and debug windows will appear. The figure bellow shows what some of those new items are:



* 1. Clicking again on the debug menu will give you the option to “Set Into”, “Step Over”, and “Step Out” of each line of code.
  2. (Optional explanatory material) Stepping Into a line of code will execute the next line of code and if that next line of code involves a call to another function or part of the program it will take you there. Stepping Over a line of code will also execute that next line of code but will not take you into another part of the program if it’s a function call. Stepping out will execute everything to the end of the current function and then stop when the program goes back to the calling function. Since we are in main this would run to the end of the program.
  3. Notice that there are short cut keys, F11 for stepping into, F10 for stepping over and Shift+F11 for Stepping out.
  4. Press F10 for stepping over. Notice that the next line of code executes.



* 1. Once you understand how debug is showing you the values in your code. Step through the rest of your program using F10. Run it several times until you understand how your code is setting and changing the variables in your program.
  2. When you are done debugging click the stop icon to stop debugging:

