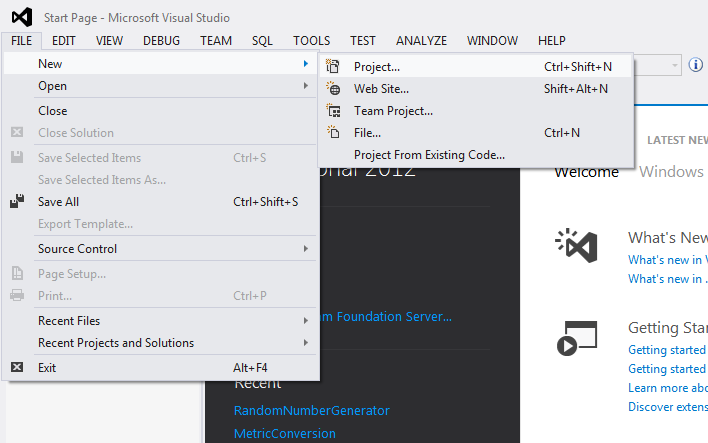
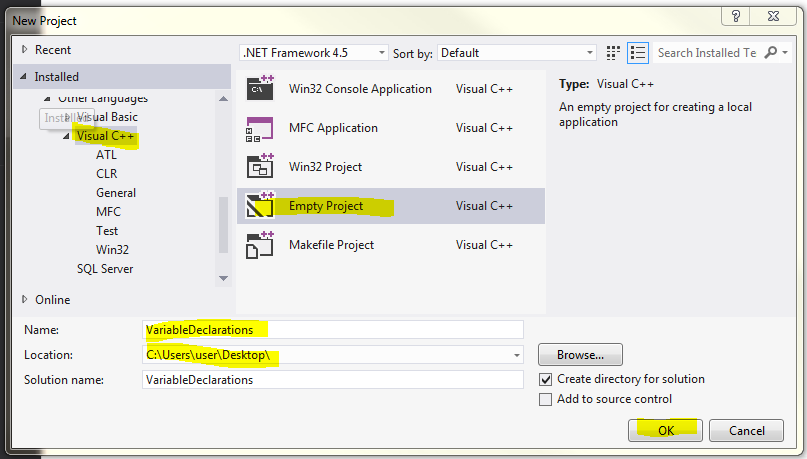
Variable Declaration Demo

Video at <https://youtu.be/xcPB5MvpAgs>

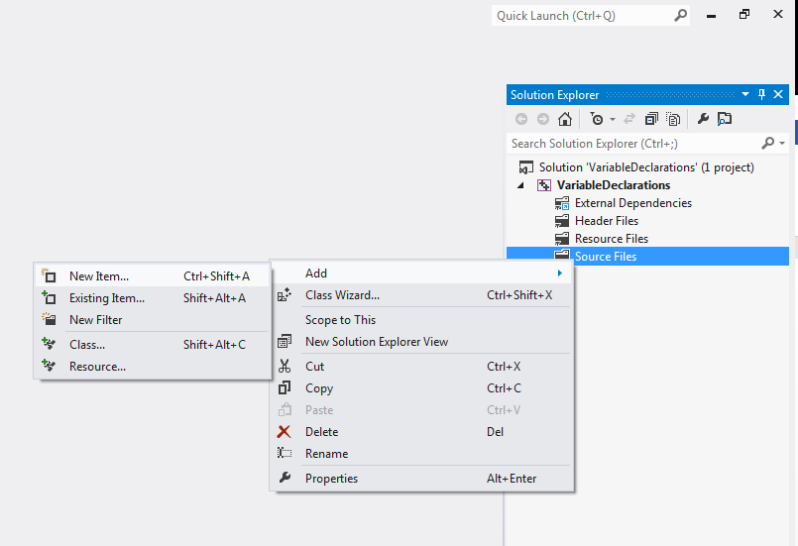
1. Start Visual Studio
2. Create a new empty C++ project:



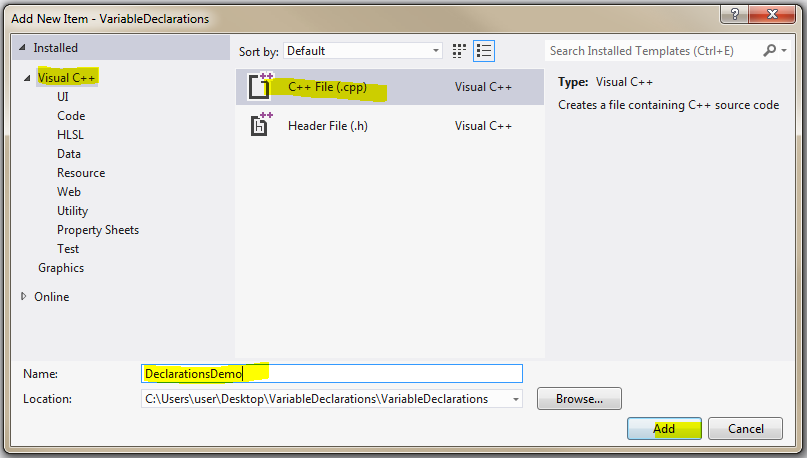
1. Call it “VariableDeclarations”, pick the desktop as the location and click OK.



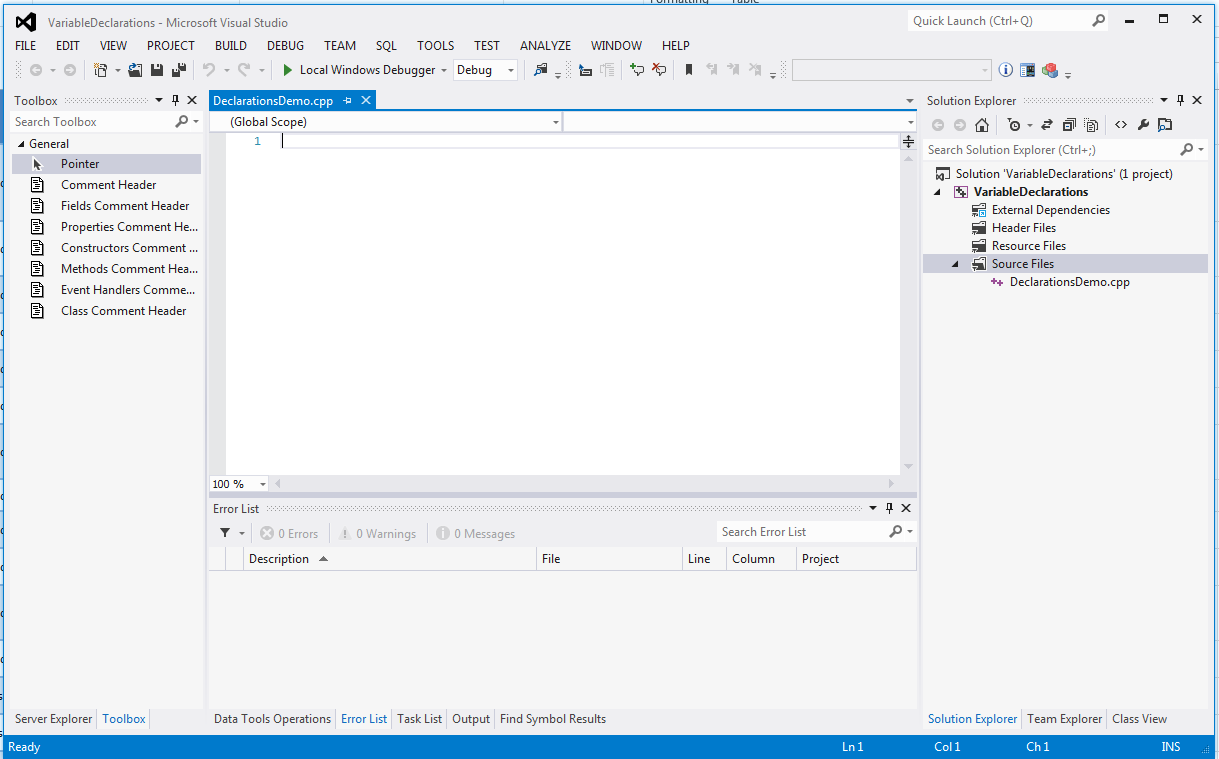
1. Click on Source Files and Add|New Item:



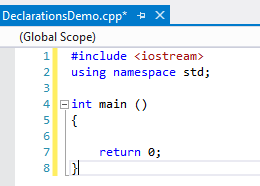
1. Select Visual C++, C++ File (.cpp), call it “DeclarationDemo”, then click “Add”:



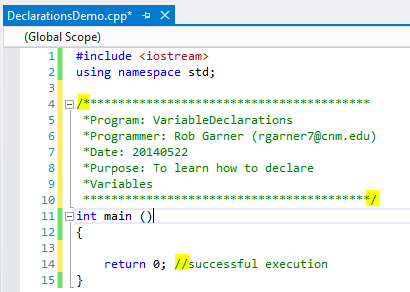
1. You should now have “DeclarationDemo.cpp” in the solution explorer and it should be open in the main window:



1. Enter the following code to start our program:

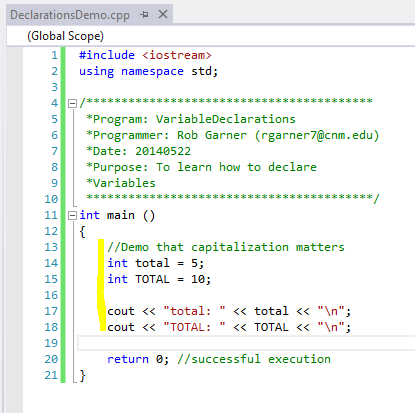


1. Take a look at this code and memorize it. We will be starting all of our programs this way. We #include the iostream library so we can send text to the screen and read text from the keyboard. We use namespace std so we can deconflict with other files we may create later. We then have to have a main 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.
2. Let’s add some comments:

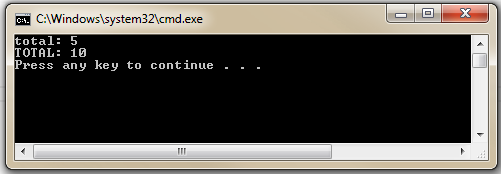


* 1. Comments can be either clock comments that start with “/\*” and end with “\*/” or single line comments that just start with “//”. The picture above highlights the characters that designate the beginning and end of the block comment at the top and the beginning of the single line comment at the bottom. 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.

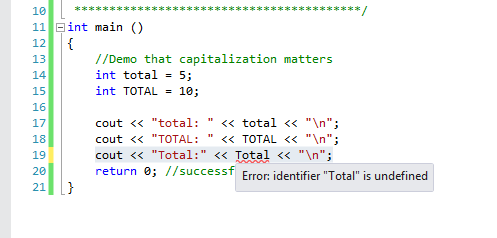
1. Capitalization matters!
   1. Add the highlighted code:



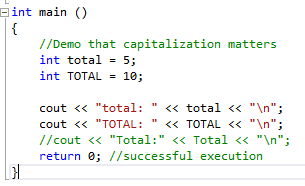
* 1. Run it with Ctrl-F5



* 1. Notice that total and TOTAL refer to two completely separate items. Realize if you change the capatilization of your identifiers the program won’t identify them as the same variable!
  2. Add another line to your code: cout << "Total:" << Total << "\n";
  3. Notice that if you hover over the red squiggle you get an error. The program does not know that Total is. We declared a TOTAL and a total but not a Total.

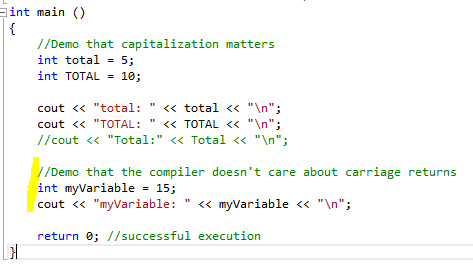


* 1. Comment out that line with “//”:

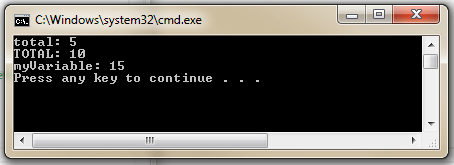


* 1. “Commenting out a line” is a technique used by programmers during debugging to disable something that doesn’t work yet but that they want to keep in place temporarily. By adding the “//” in front of the line it makes it a comment and allows our program to compile without that error.

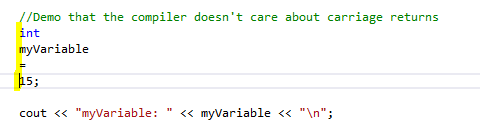
1. The compiler doesn’t care about carriage returns:
   1. Add the highlighted code:



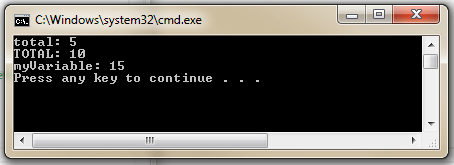
* 1. Run the program with Ctrl-F5. Say yes if it asks you to compile.



* 1. Notice that myVariable is successfully declared and displays when run.
  2. Now change the code to look like this:



* 1. Again run it with Ctrl-F5
  2. Notice you still get:

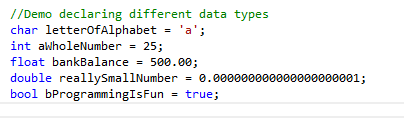


* 1. The compiler considers the carriage returns at the end of the line as “white space” and ignores them. From the compiler’s perspective the program looks something like this:

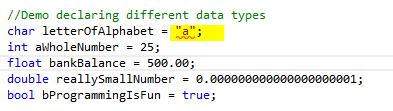
#include|<iostream>|using|namespace|stdint|main|()|{int|total|=|5;|int|TOTAL|=|10;|cout|<<|"total: "|<<|total|<<|"\n";|cout|<<|"TOTAL: "|<<|TOTAL|<<|"\n";||int|myVariable|=|15;|cout|<<|"myVariable: "|<<|myVariable|<<|"\n";|return|0;|

* 1. That would be hard for you to read. So use carriage returns, spaces and tabs to make you program clear and easy to understand.

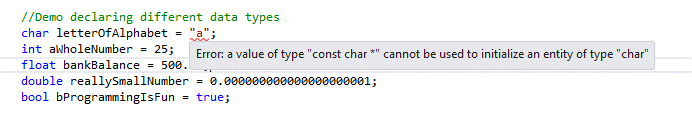
1. Declare some other variables:
   1. Add the following code:



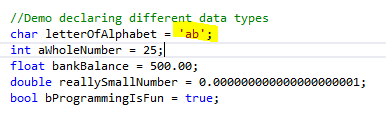
* 1. Notice that there are no errors.
  2. Now change the char definition:



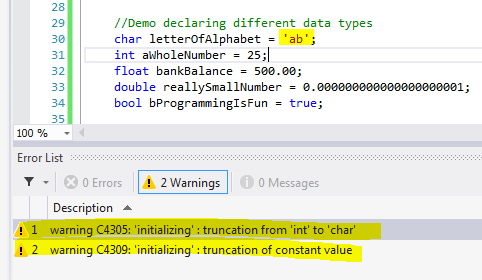
* 1. Notice you now get red squiggles indicating an error.
  2. If you hover over it:



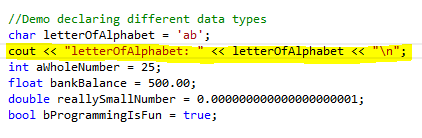
* 1. Double quotes are not the same as single quotes. A char requires one character between two single quotes.
  2. Change it to the following:



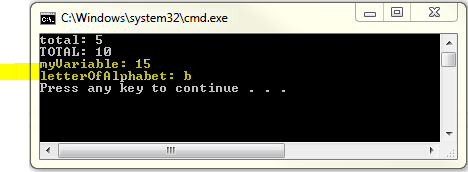
* 1. No errors appear but look at the “Error List” tab at the bottom of the window. Notice we get a warning. Your code will compile (you can try it with Build|Build Solution). However, letterOfAlphabet will only contain ‘b’.



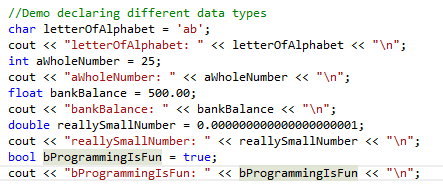
* 1. Add a cout line and try it:



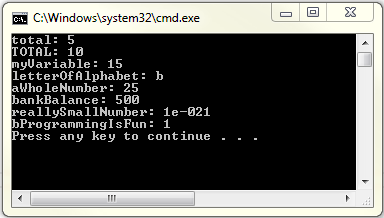
* 1. Run it with Ctrl-F5:



* 1. Notice that only the b character was retained! Only put one character between single quotes when assigning a value to a char.
  2. Add cout lines for the rest of the variables:

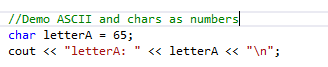


* 1. Run it with Ctrl-F5

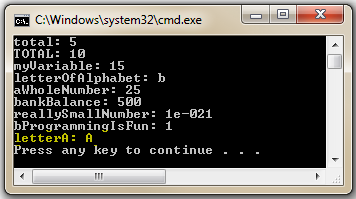


* 1. Notice that everything displays normally except:
  2. ReallySmallNumber is displayed in scientific notation. 1e-021 is the same as 0.000000000000000000001. To read scientific notation read it as “One times ten to the negative twenty first power” or .
  3. bProgrammingIsFun displays “true” as 1. If you change bProgrammingIsFun to “false” it will display 0.

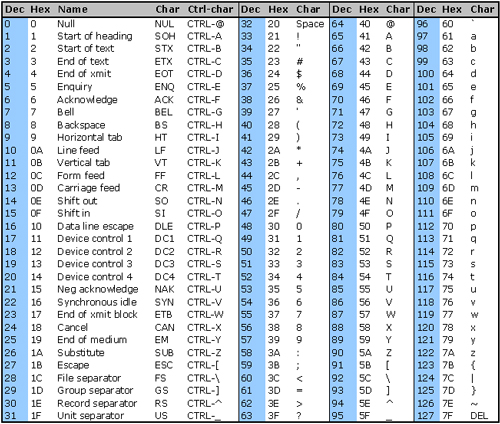
1. Demo ASCII and chars as numbers:
   1. Add the following code:



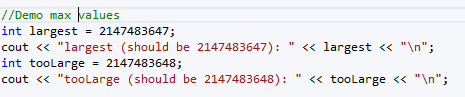
* 1. Run it with Ctrl-F5



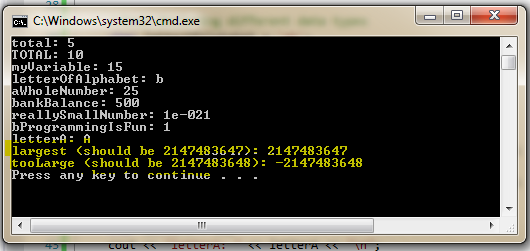
* 1. Notice that when we assign 65 to the char letterA it displays as an “A”. Why is this?
  2. Chars are actually numeric types that represent a number from -128 to 127. However when we send them to cout, C++ displays them as their ASCII equivalent character.
  3. The following ASCII chart shows what numbers correspond to what character:



1. Demo max values:
   1. Add the following code:

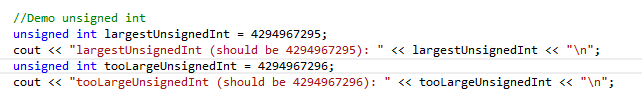


* 1. Test it with Ctrl-F5:

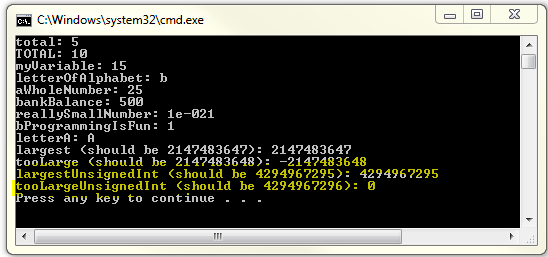


* 1. Notice that the int largest displays ok but tooLarge displays as a negative number!
  2. (optional material) 2147483647 is the largest possible number we can represent with an int and is 01111111 11111111 11111111 11111111 in binary. When we add 1 to 2147483647 to get 2147483648 we get 10000000 00000000 00000000 00000000. The left most bit is used as a sign bit for most numbers. The result is that when we add one more to the largest possible number the operating system treats it as a negative number and not what we expected.
  3. When declaring variables pay attention to what their maximum and minimum values might be and always test your code to make sure it won’t be a problem. We will learn how to do that later in the course.

1. Demo unsigned modifier:
   1. Add the following code:



* 1. Test it with Ctrl-F5:



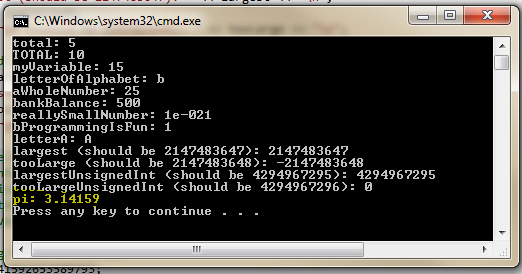
* 1. Notice tooLargeUnsignedInt become 0!
  2. (Optional) 4294967295 is 11111111 11111111 11111111 11111111 in binary. With unsigned numbers the most significant bit is not used as a sign bit and instead becomes part of the number (that’s why an unsigned int can store a number twice as large as the max positive value of a signed int). When we add one more to the number it would become: 1 00000000 00000000 00000000 00000000. However there is no space for the 1 at the far left so it is just thrown away and therefore 4294967296 becomes: 00000000 00000000 00000000 00000000 or 0.
  3. Bottom line: pay attention to max and min values for your declared types.
  4. Max and min values by type:



1. Demo Floating Point Truncation:
   1. Enter the following code:

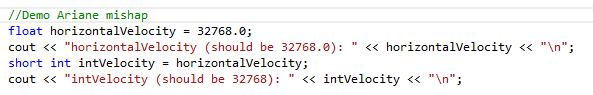


* 1. Test the code with ctrl-F5:

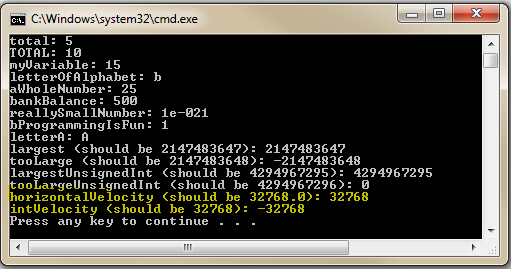


* 1. Notice that pi is truncaged to 3.14159

1. In our presentation we spoke about the Arian rocket problem:
   1. On June 4, 1996, an unmanned Ariane 5 rocket was launched by the European Space Agency and exploded 40 seconds later.
   2. Why? A programming error – a 64-bit floating point number relating the horizontal velocity of the rocket wrt the platform was converted into a 16 bit signed integer.
   3. The number was larger than 32,767, the largest integer that can be stored in a 16 bit number, and the conversion failed. So did the rocket.
   4. Lets demo this error:
      1. Add the following code:



* + 1. We declare a float to represent the horizontal velocity of the rocket. 32768 is one more than the largest value we can store in a 16 bit int.
    2. A short int is a 16 bit integer number. We declare intVelocity as a short int.
    3. Next we assign the value of horizontalVelocity into intVelocity. C++ will implicitly convert horizontalVelocity which is a float into a short int before it assigns it into intVelocity.
    4. When we run it we get:



* + 1. intVelocity is now the negative of what it should be! The rocket thinks it’s going the wrong way!
    2. We will talk about type conversions in more detail later.