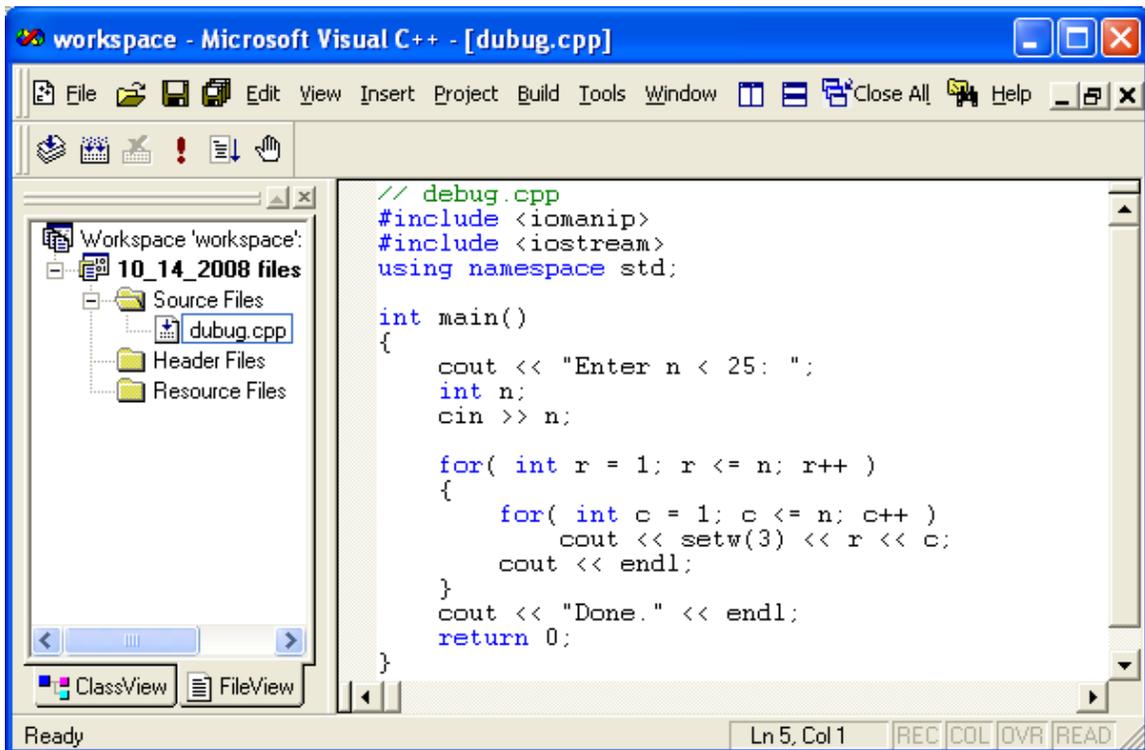


Create a Visual C++ Win32 Console project named 10\_14\_2008. Add to it a new C++ source code file named debug.cpp (you do not need to add a header file). Edit the source code file so it looks like this:



It behaves like the following when it is run: (output is shown in black; user input is shown in red):

```

Enter n < 25: 5
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
51 52 53 54 55
Done.

```

a. Modify the program so that instead of outputting a row number followed by a column number, it outputs their sum. It should behave as below:

```

Enter n < 25: 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
Done.

```

b. Modify your program so it behaves this way:

```

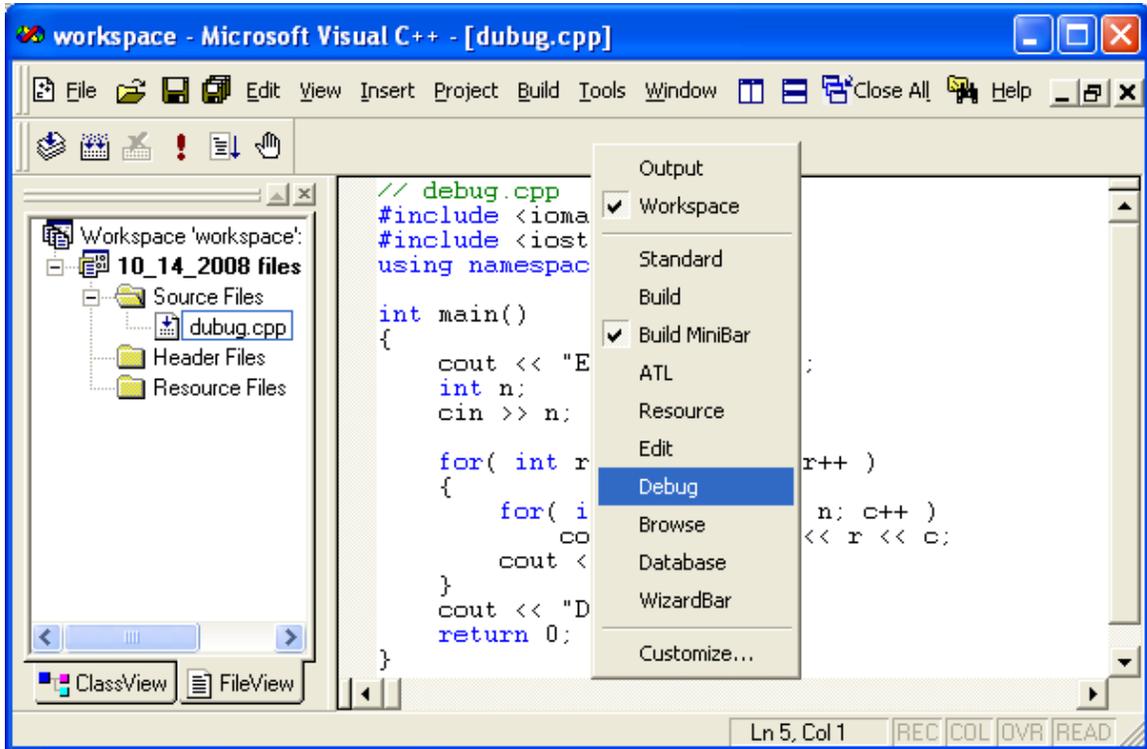
Enter n < 25: 5
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
Done.

```

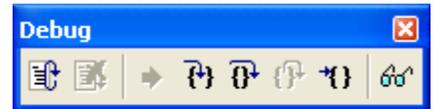
(This is called a *Hankel Matrix* by the way)

Now we'll get familiar with the debugger.

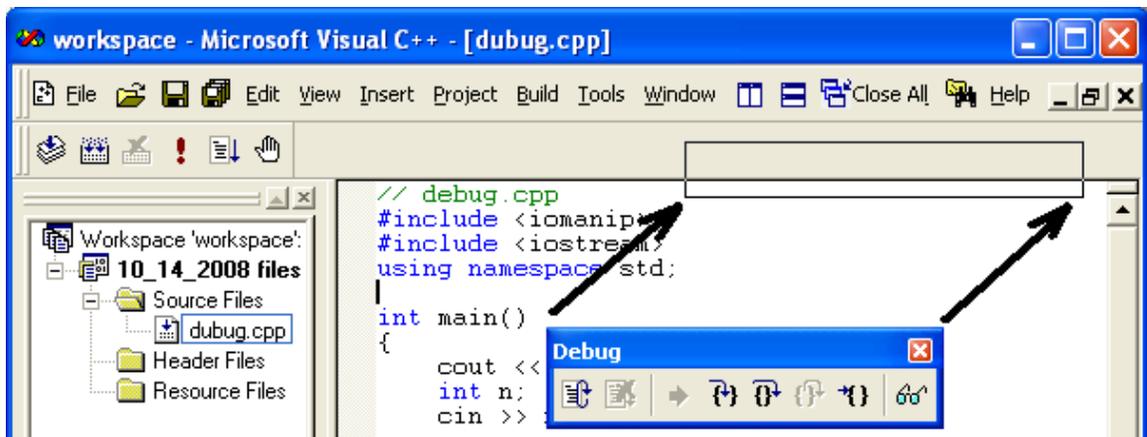
Right-click in an area of the menu bars that has no buttons: this will pop up a window listing the available toolbars. If it is not selected already, select the *Debug* toolbar:



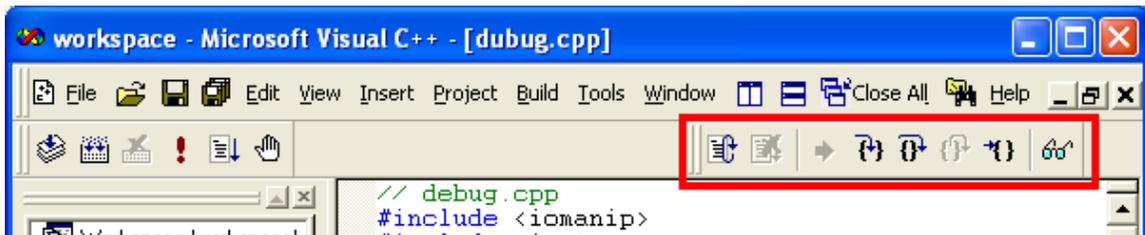
If it opens a window that looks something like this:



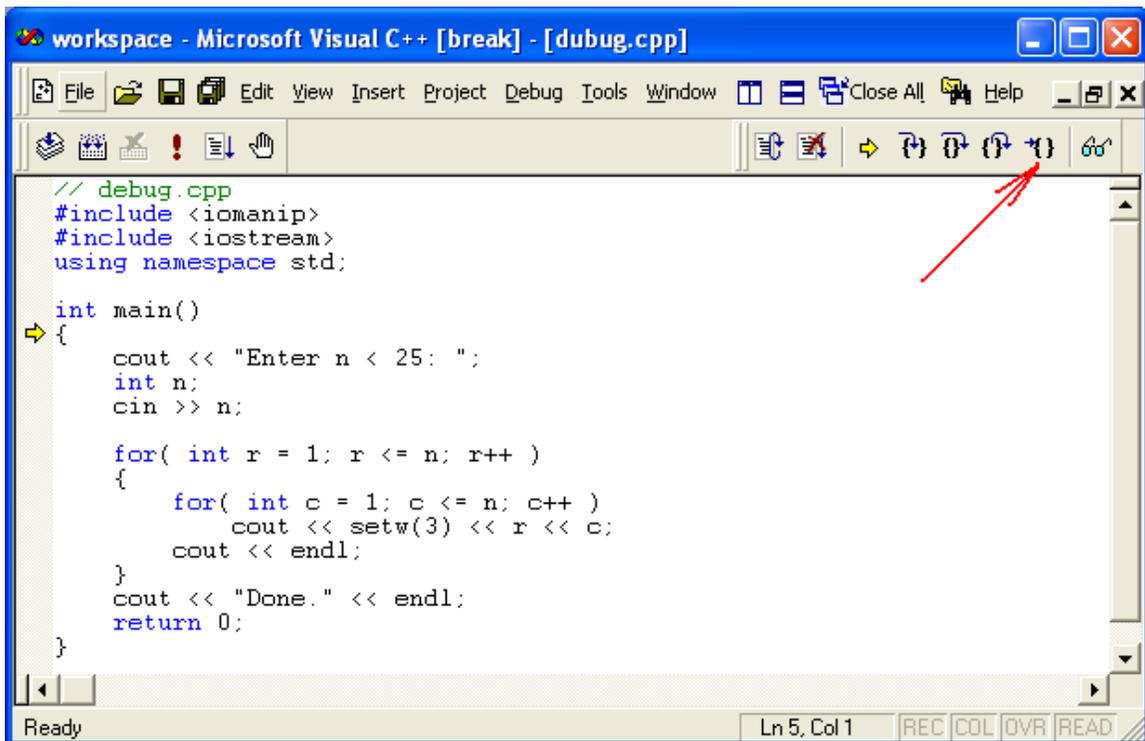
... drag it to the menu bar area:



Here's the Debug window docked on the menu bar:



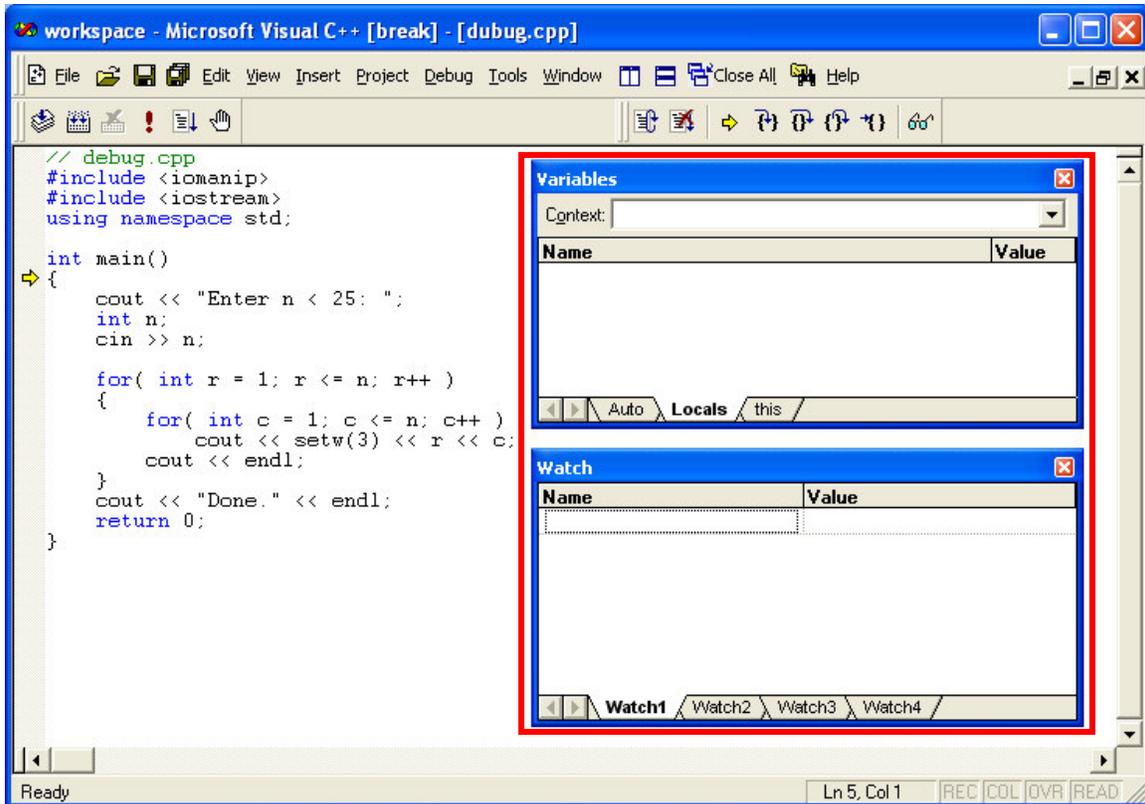
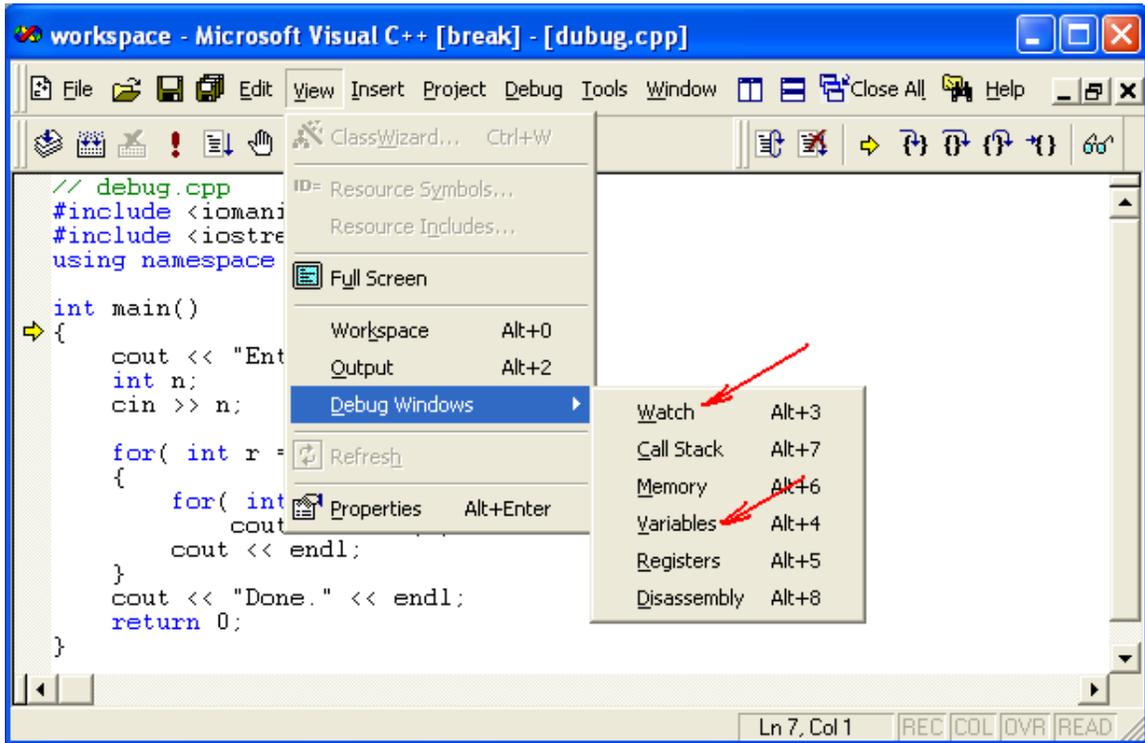
Put your cursor on the line of code with `int main()`, and click on the **Run to Cursor** icon.



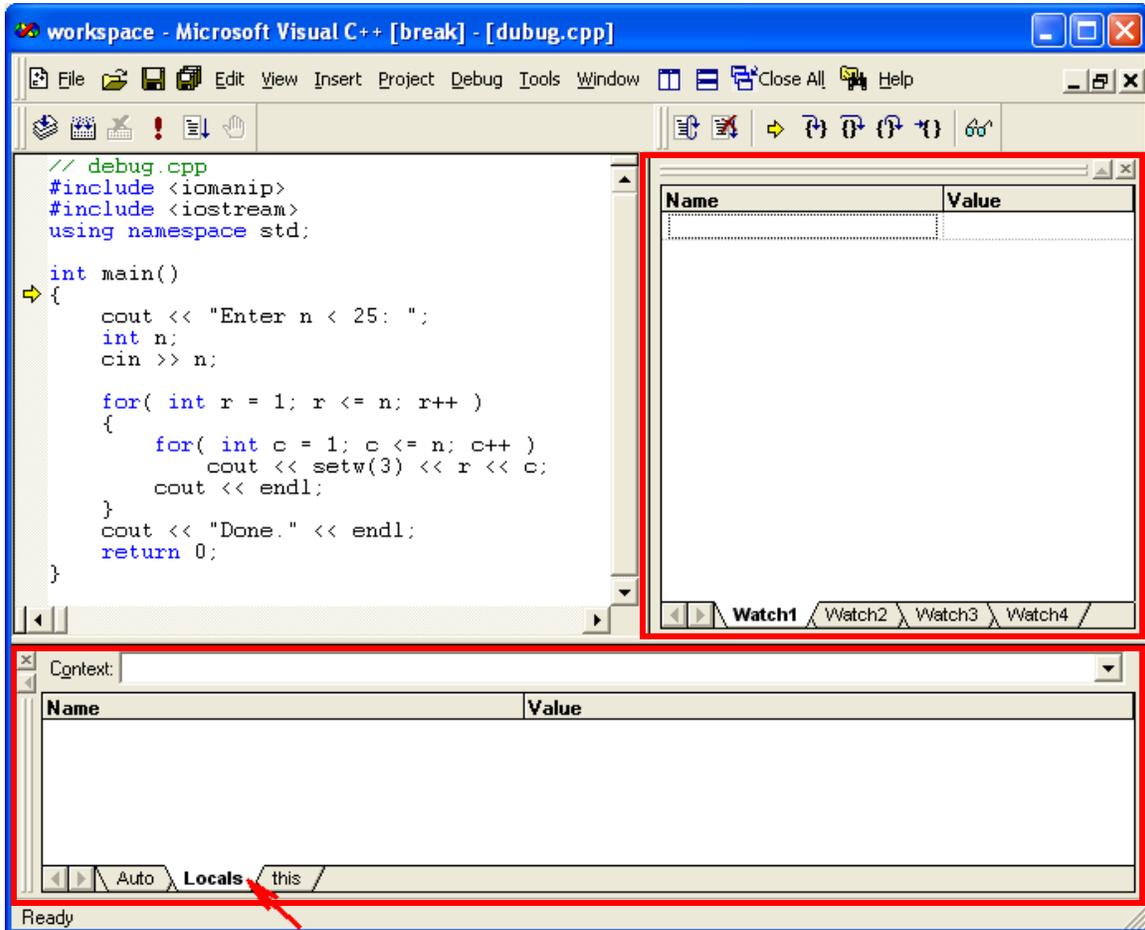
The program will compile and link if necessary, begin executing, then stop at the line where your cursor is. A yellow arrow indicates the next line to execute.

**Run to Cursor** is useful if you want your program to run to, then pause at a certain spot but you don't want to have to individually step through every single line of code by hand to get there.

From the View -> Debug Windows ► menu, open the following debugging windows: Watch, Variables



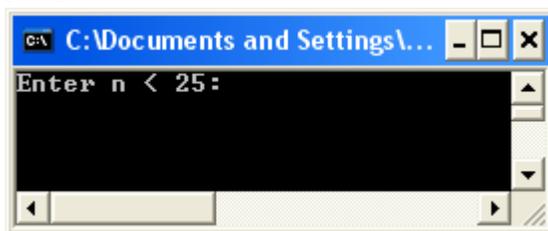
You can resize these windows, drag them around on the screen, and dock them in the Visual C++ main window if you like. Here I have the Variables window docked at the bottom, and the Watch window docked in the upper right:



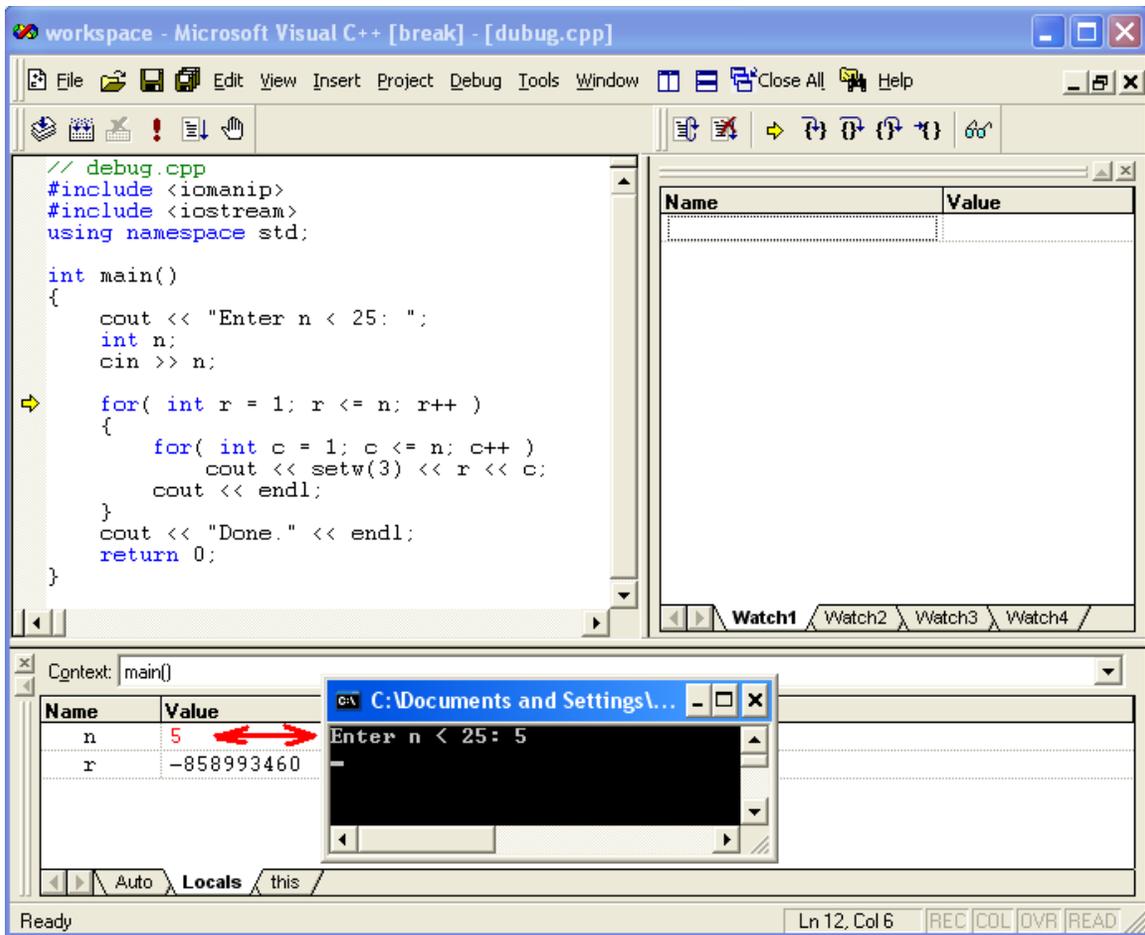
Make sure the **Locals** tab is selected in the Variables window.

Click the **Step Over** icon  until the yellow arrow goes past the variable declaration, **int n**. In the Variables window **n** will be displayed with its current value (if we had declared other variables, they would be shown here as well). Since we have not assigned anything to **n**, it will have a random value.

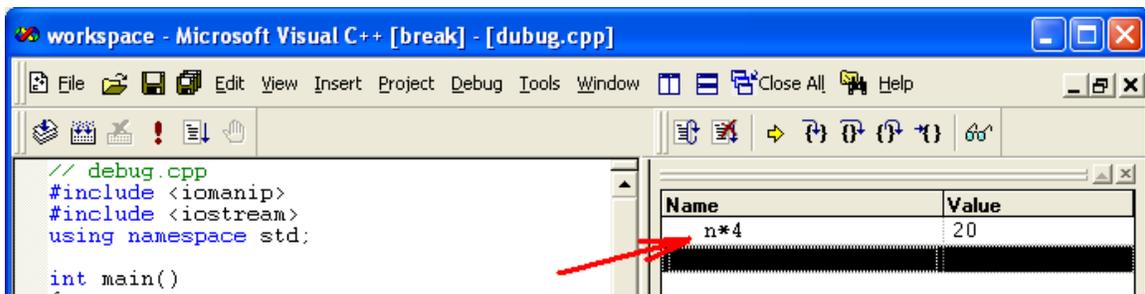
You can continue to click **Step Over**, but progress stops on the **cin** line. The program is waiting for you to input **n**. Click on your program's console window:



Type a value for **n** in the console window, and verify that it shows up in the **Locals** window.



Variables that come into scope automatically appear in the Locals window, and disappear when they go out of scope. It is also possible to manually add variables and expressions, but these will appear in the Watch window. Click on the **Watch1** tab, then click in the blank cell on the left-hand side of that window, and then enter an expression like **n\*4**. You should see 4 times the current value of **n** that you typed into the console window (i.e., your input, read by cin):

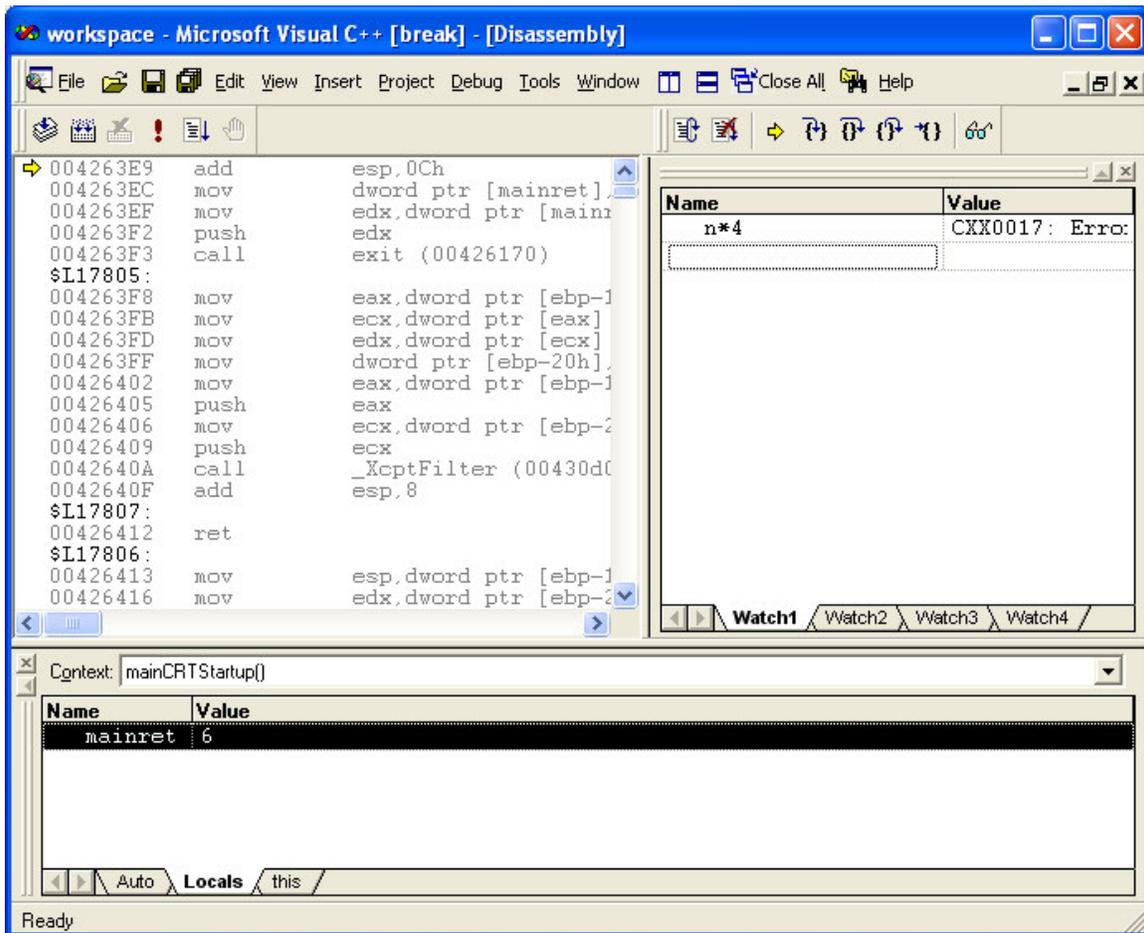


Proceed through your program using **Step Over**. Observe how the variables change as you proceed through the loops. Observe how they come into and go out of scope as you go.

Walking through your program in this manner is very slow. To move faster, we'll revisit the **Run to Cursor** button. Position the cursor to the line with the first curly brace following the first for. Click on the **Run to Cursor** button. What happened (check the **Program Window**)? Click on the **Run to Cursor** button several times, checking the output each time.

**Step Into**  and **Step Out**  are used to step into and out of functions other than main() (we don't have any in this example). If you accidentally end up in a screen full of code from one of the libraries, you can get back to your code by using the **Step Out** icon.

Try this: **Step Out** of main() by clicking on the **Step Out** icon. If a window opens that says "Find Source", select CANCEL to close it.



What is displayed is assembly language code that Windows uses to terminate your program. Close that window (use the small black X on the top right of the Visual C++ window). You should be back to your source code.

To stop debugging, click the **Stop Debugging** icon: 

For each of these programs, (1) compile and run it, (2) use the debugger to correct it, (3) Submit: A hardcopy listing of each corrected program.

**// Debugging practice program #1**

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

int main()
{
    cout << "Enter two Integers: ";
    int a, b;
    cin >> a >> b;

    if(a = b)
        cout << "They are Equal!\n";
    else if( a > b )
        cout << "The 1st is bigger!\n";
    else
        cout << "The 2nd is bigger!\n";

    cout << "Enter a Decimal to be Converted to Percent: ";
    double decimal;
    int percent;
    cin >> decimal;

    percent = static_cast<int>(decimal) * 100;
    cout << "That's " << percent << '%' << endl;

    return 0;
}
```

**// Debugging practice program #2**

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

void myswap( double& x, double& y );

int main()
{
    cout << "Enter two numbers and I'll swap them: ";
    double a, b;
    cin >> a >> b;

    cout << fixed << showpoint << setprecision(3);
    cout << "a was " << a << ", b was " << b << endl;
    myswap( a, b );
    cout << "a is " << a << ", b is " << b << endl;

    return 0;
}

void myswap( double& x, double& y )
{
    int temp;
    temp = x;
    y = x;
    x = temp;
}
```

```
// Debugging practice program #3
#include <iomanip>
#include <iostream>
using namespace std;

void squareIt( int n );

int main()
{
    cout << "Enter an integer n <= 20: ";
    int n;
    cin >> n;
    if( n < 20 )
        return 1;

    cout << fixed << showpoint << setprecision(3);
    cout << "The first " << n << " squares: " << endl;
    for( int i = 1; i < n; i++ )
    {
        squareIt( i );
        cout << setw(2) << i << endl;
    }

    return 0;
}

void squareIt( int n )
{
    n = n * n;
}
```