Break Points

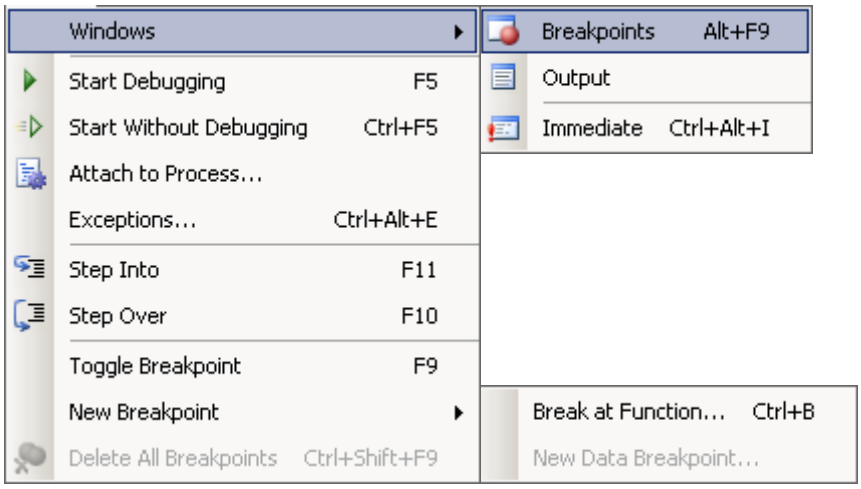
http://www.cs.uvm.edu/~upe/resources/debugging/visualStudioCDebug/
http://msdn.microsoft.com/en-us/library/5557y8b4(VS.80).aspx
http://msdn.microsoft.com/en-us/library/k80ex6de(VS.80).aspx

Breakpoints can be added by clicking in the gray margin next to a line of code or via the Debug Menu.

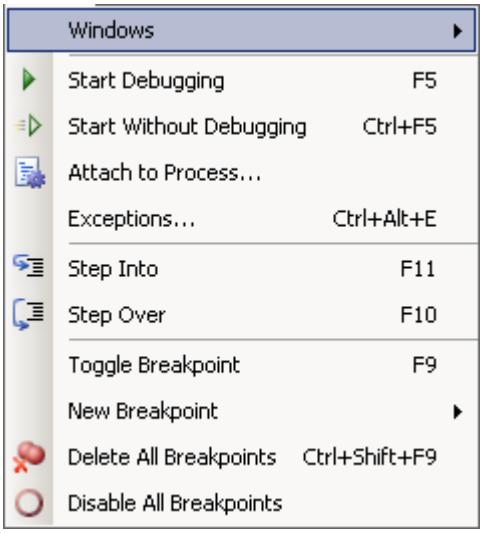
Breakpoints will be marked with symbols such as the following next to lines of code:

	Normal breakpoint. The solid glyph indicates that the breakpoint is enabled. The hollow glyph indicates that it is disabled.

If the code does not currently contain any breakpoints, the Debug Menu will show:

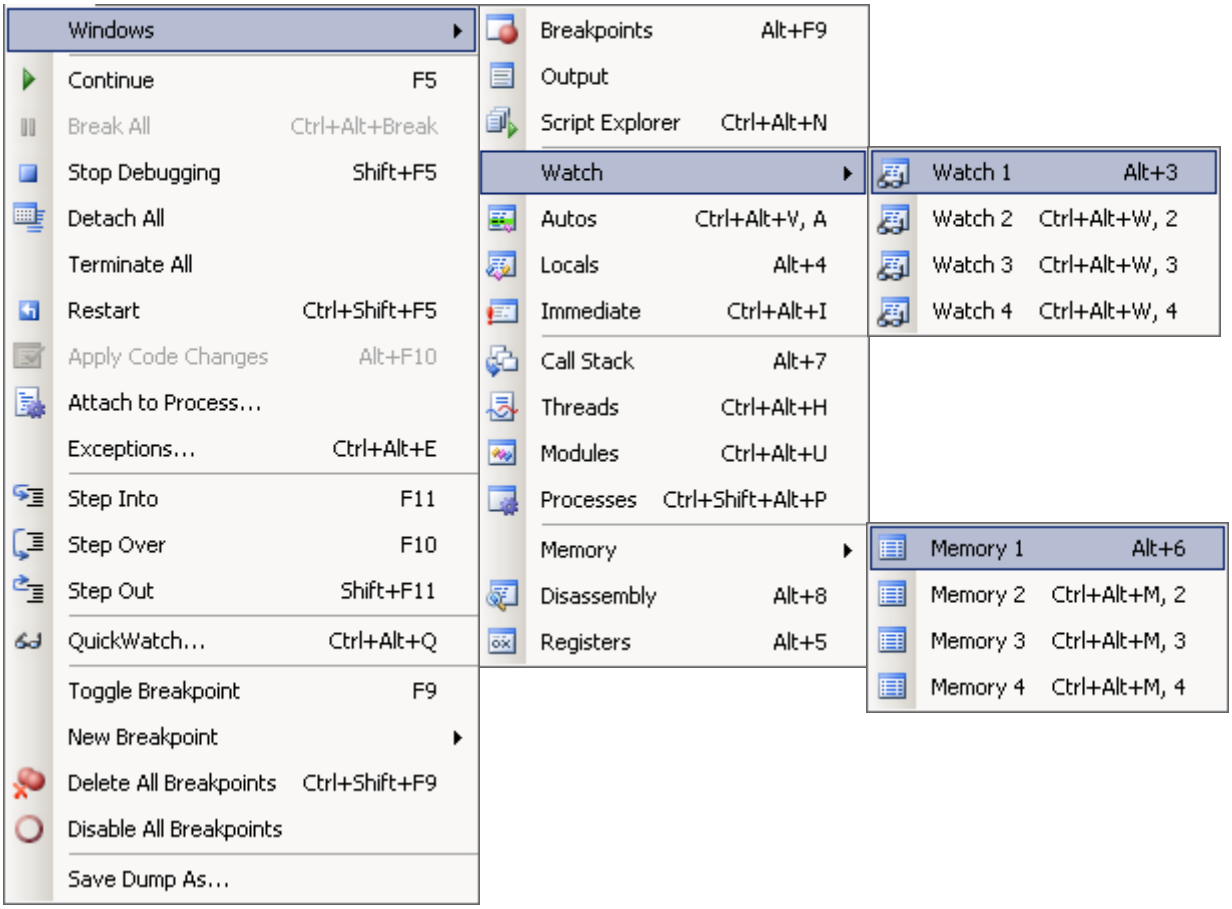


If the code contains at least one breakpoint, the Debug Menu contains one more option to "Disable All Breakpoints":



If the code contains at least one breakpoint and is running, the Debug Menu will contain many additional options for controlling the execution of the program:

(While the program is running, the Windows Submenu is the same whether the program contains breakpoints or not.)



After adding one or more breakpoints, press **F5** to start the program. The program will run until it gets to the first line with a breakpoint, however it will NOT actually execute the line with a breakpoint. To run the program and force C++ to ignore the break points, press **CTRL+F5** (Start Without Debugging).

A yellow arrow will indicate which line will be **executed NEXT.** 
It does **NOT** indicate a line which has just been executed:

To execute the line with the breakpoint and **continue until the next breakpoint**, press **F5** again.

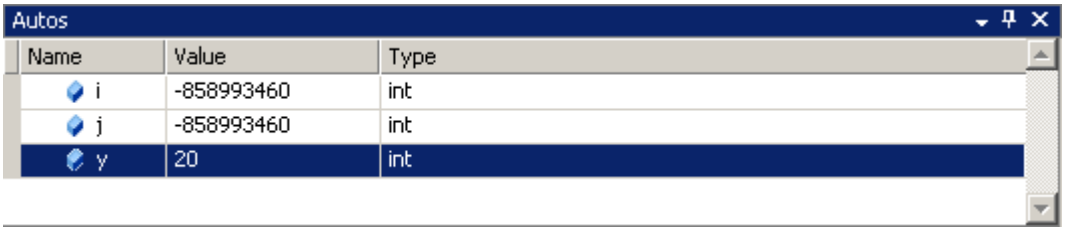
To execute the program **line-by-line**, whether it contains breakpoints or not, press **F10**.

To **stop** running the program, press **F7** or **SHIFT+F5**.

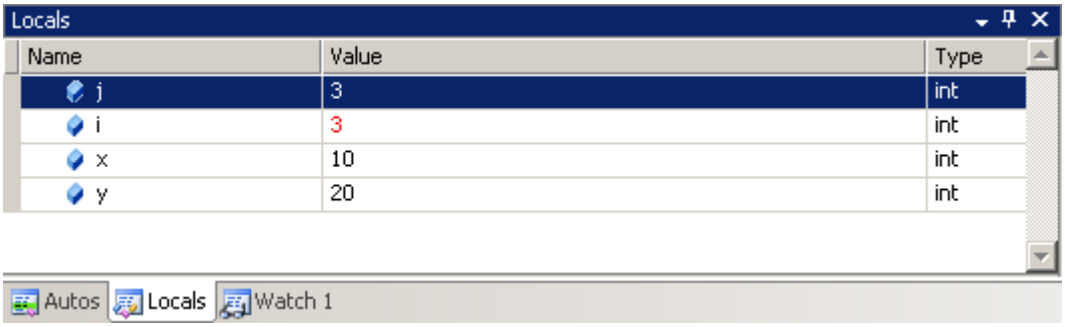
Consider the following code example:

```
int x = 10;
int y = 20;

for (int i = 1, j = 1; i <= 9; i++, j++)
{
   cout << "i: " << i << endl;
   cout << "j: " << j << endl << endl;
}

for (int r = 1, s = 1; r <= 9; r++, s++)
{
   cout << "r: " << r << endl;
   cout << "s: " << s << endl << endl;
}

cout << x << endl;
cout << y << endl;
```

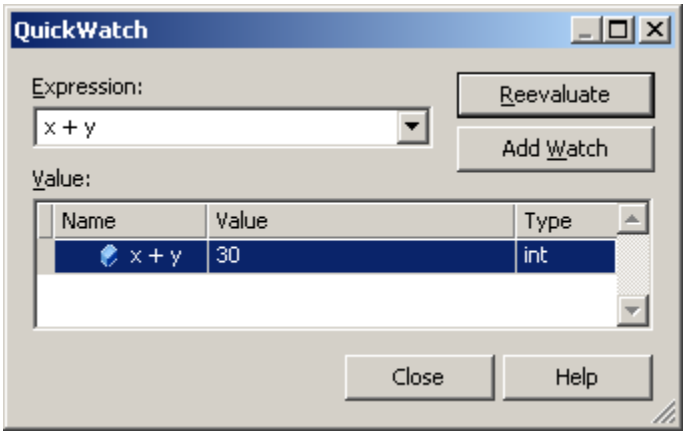
The Debug Menu…Windows…Autos window (CTRL+ALT+V, a) will display the value of the most recently accessed variables as the program runs. The list of displayed variables will change as the program runs.



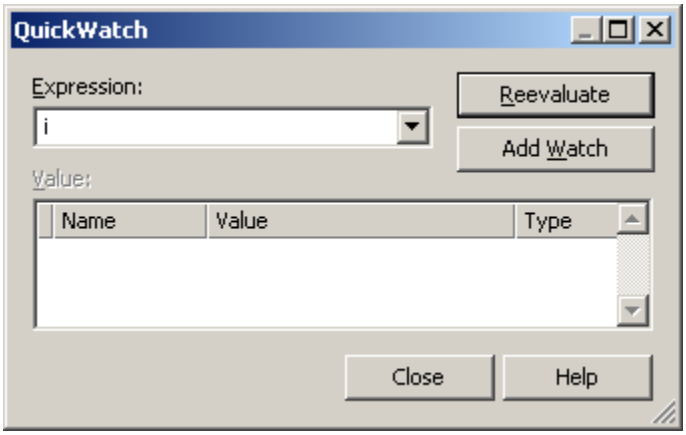
The Debug Menu…Windows…Locals window (ALT+4) will display the values of variables as they come into scope (local variables). Recently changed variable values are displayed in red.

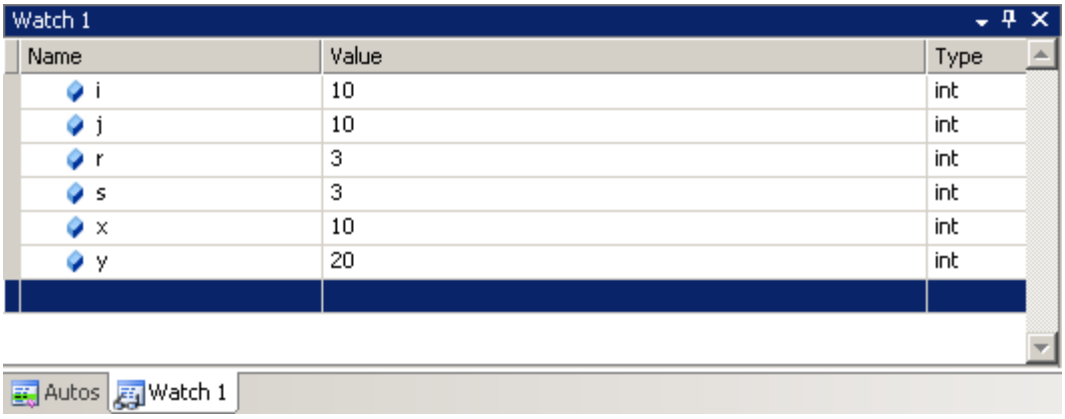


The QuickWatch debug tool can be used to evaluate an expression in the course of running a program.

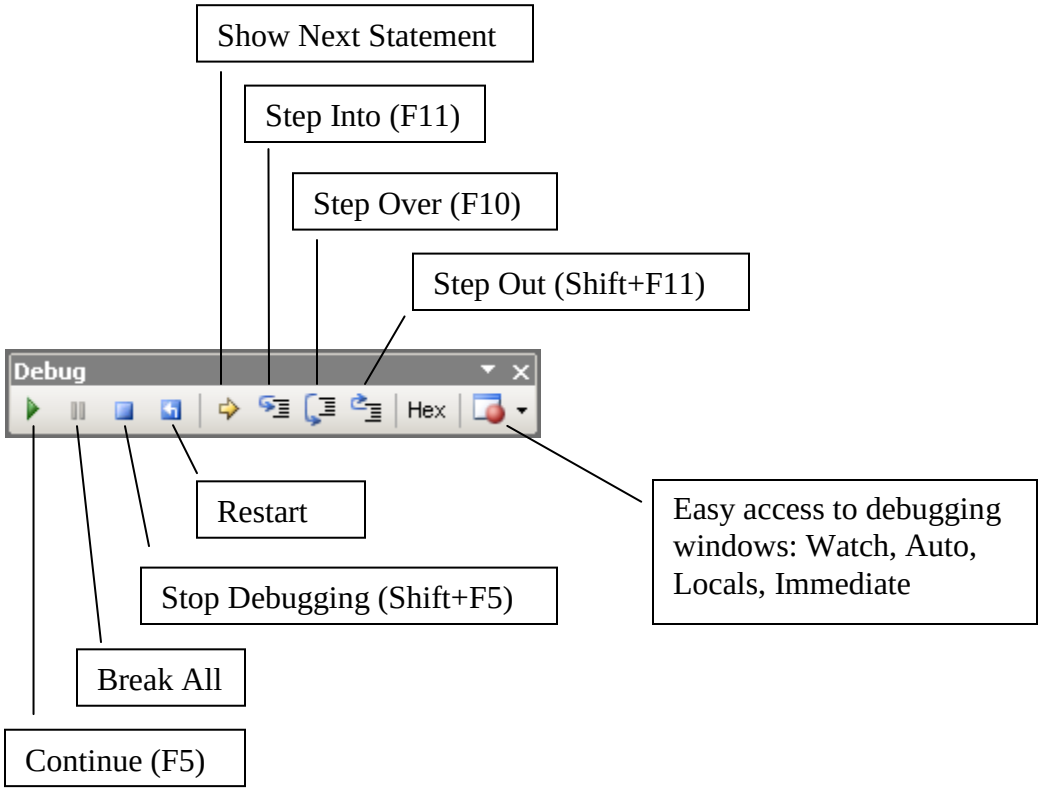


QuickWatch can also be used to create a constant list of variable values (or custom expressions) to watch. The variables and their values will be displayed in a window at the bottom of Visual Studio.





When a program is run in debug mode, the debug toolbar appears:

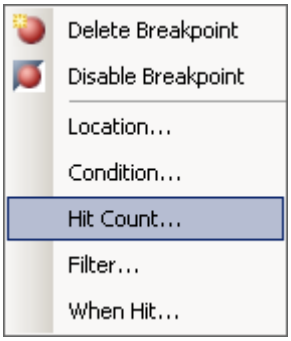


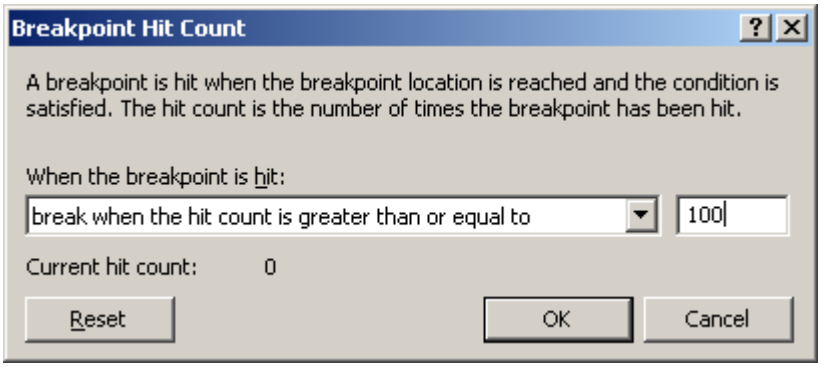
Continue (F5)	- Continue executing until the next breakpoint is encountered
Break All	-
Stop Debugging (Shift+F5)	- Immediately exits out of the program
Restart	- Restarts the program from the beginning
Step Over (F10)	- Entire statement is executed. If the statement includes a function call, the function call is fully executed, and the "Next Statement" indicator moves to the next statement.
Step Into (F11)	- Statement is executed. If the statement includes a function call, control moves inside the function, and the "Next Statement" indicator moves to the first statement inside the function.
Step Out (Shift+F11)	- If program control is in the middle of a called function, the rest of the function will be executed, and control will return to the next statement after the statement which called the function.

Conditional Breakpoints

If a break point is inside a loop, and the program only fails after 100 iterations, then breaking on the first 99 iterations is tedious and a waste time.

It is possible to instruct Visual Studio to break only after the break point has been encountered 99 times.





The icon for a conditional breakpoint looks like: