CIS540/543 Lab 7 Fall 2009

Objectives:

- Learn Debugging in Visual Studio

Introduction:

Errors can be syntax errors, semantic errors, or logic errors.

The most obvious type of error is the syntax error, which occurs when you write code in a manner not allowed by the rules of the language. Syntax errors are almost always caught by the compiler or interpreter.

Semantic errors are a more subtle type of error. A semantic error occurs when the syntax of your code is correct, but the semantics or meaning are not what you intended. Because the construction obeys the language rules, semantic errors are not caught by the compiler nor the interpreter. Compilers and interpreters concern themselves only with the structure of the code you write, not the meaning. A semantic error can cause your program to terminate abnormally, with or without an error message. In colloquial terms, it can cause your program to crash or hang.

Not all semantic errors manifest themselves in such an obvious fashion, however. A program can continue running after some semantic errors, but the internal state of the program will not be what you intended. Variables may not contain the correct data, or the program may continue down a path that is not what you intended. The eventual result will be incorrect output. These errors are called logic errors, because while the program does not crash, the logic that it executes is in error.

Once you have created your application and resolved the build errors you might need to correct logic errors that keep your application from running correctly. The only way to detect logic errors is by testing your program, manually or automatically. Unfortunately, while testing can show you that the output of your program is incorrect, it usually leaves you without a clue as to what part of your code actually caused the problem. This is where Debugging can be of use.

Debugging Basics:

Breakpoint

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point.

In C++, you can even make changes to the code while in break mode (a feature called Edit and Continue).

To insert a line breakpoint, click in the gray margin next to the line where you want to set the breakpoint.

Quickwatch

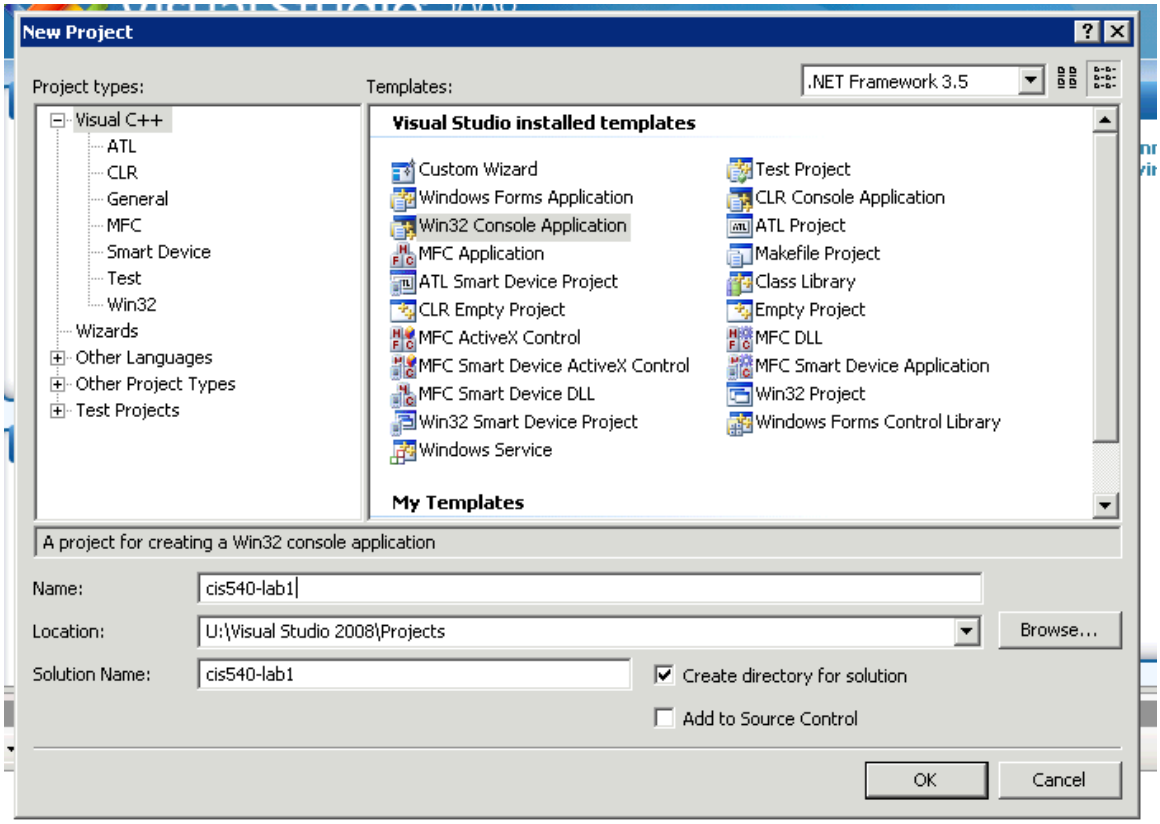
As the name implies, Quick-Watch provides a fast way to look at and evaluate variables and expressions.

In Visual Studio, you can get a quick look at a variable's value by placing the cursor over the variable. A small box called a Data-Tip will appear showing the value.

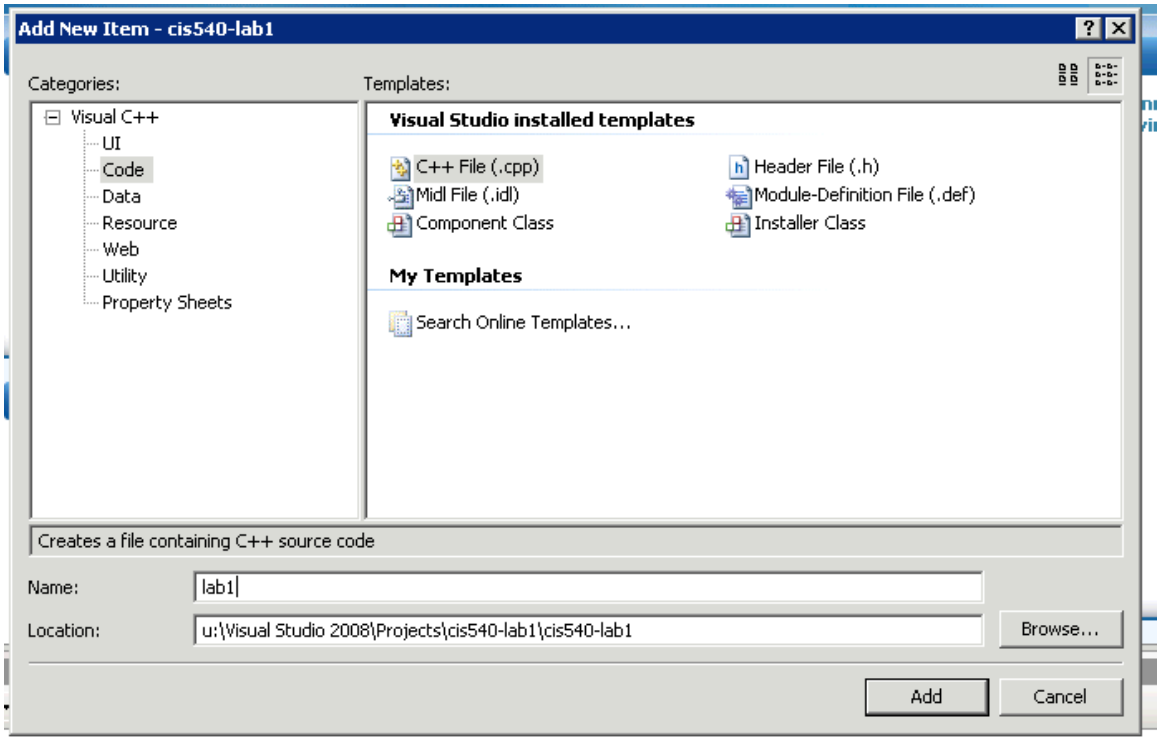
Exercise:

Create a simple C++ Console Application

Open *Microsoft Visual Studio 2008*, Click *File->New->Project: Visual C++, Win32 Console Application*. Choose a name and click *Next*. Under *Additional options*, check ***Empty project***. Click *Finish*.



Right click *Source Files* folder in your new project, choose *Add->New Item*. Select *Code: C++ File*. Choose a name and then click *Add*.



Exercise Code:

```
#include <iostream> // Provides cout
using namespace std;

class DebugThisClass {
      private:
            const static int MAX = 6;
            int count;
            int num[MAX];

      public:

            DebugThisClass(){
                  count = MAX;
            }

            ///Enter Values into the array.
            void loadArray() {
                  //you can step over this loop if you want,
                  //but watch how the values of variables change in
the quick watch window
                  for (count = MAX - 1; count >= 0; count--) {
                        cout << "Enter an integer: " ;
                        cin >> num[count];
                  }
                  //after one or two iterations you can step out of
this function if you want.
            }

            ///Display the array
            //Step into this function, see the values of variables
in the watch window
            void printArray() {
                  int m;

                  cout <<"The data in the array at the indices
shown in the left column is:" << endl;
                  for (m = 0; m < count; m++)
                        cout << m << ":  " << num[m] << endl;
            }
};//end of class


///Driver for the CPP program
int main() {
   DebugThisClass *test = new DebugThisClass();
   test->loadArray(); //insert a breakpoint here and step into
the function
   test->printArray(); //insert a breakpoint here and step into
the function
}
```

Click *Build->Build Solution*.

To run, click *Debug->Start Without Debugging*.

The program should output the contents of the array that holds all the values that you just entered.

You will not get the output. Debug the program using the Visual studio debugger and show the proper output to the TA.

You should be able to demonstrate that you are able to step into and step over a function.

How to Debug

Insert the breakpoints at places, where you find the comments, instructing you to do so in the program.

Use Step Into (F11) the code if you want to go inside of a function.

Use Step Over (F10) if you do not want to go inside of a function when stepping through a line that has a function call.

Use Step Out (Shift + F11) if you need to get out of a function that you have gotten into.

Add values to the Quickwatch window (Ctrl + Alt + Q), this opens a window, you can type in the variables whose values you want to watch. Then click 'add watch'. You should be able to see the variables and values at the bottom left window.