Debugging in Microsoft Visual Studio

You have created your application and resolved the build errors. You must now correct those logic errors that keep your application from running correctly. You can do this with the development environment’s integrated debugging functions. These allow you to stop at procedure locations, inspect memory and register values, change variables, observe message traffic, and get a close look at what your code does. Debugger allows you to examine the objects in your program at each step and interactively execute C++ statements one at a time.

Terminology

Breakpoint
A stopping point in your program. When you start your program it will continue until the first breakpoint it encounters. At this point, the debugger stops code execution and allows you to examine the state of any objects in your program.

Stepping
Executing your program, one statement at a time. There are two ways of stepping: “Step Into” (F11) and “Step Over” (F10)

Step Over
Execute the current statement as a single entity even if it may contain other statements (e.g. block statements).

Step Into
If the statement represents a collection of actions (such as another function), execution goes in to the first statement inside this function

Starting Debugger
Your project must be in the debug configuration, to be able to properly debug. To switch to the Debug or Release configuration on the Standard toolbar, click Debug or Release from the Solution Configurations list box.

Then hit F5 or select Debug → Start Debugging.
Basic Steps

Insert break point

A breakpoint is shown as a solid maroon circle. Either click inside the margin indicator bar (the grey margin on the left hand side of the code) next to the line of code at which you wish to break, or right click the line of code and select Insert breakpoint. Click on the break point to remove it. You can set as many break points as you want. When the program runs, it stops at any line that contains a break point (called break mode).

Begin Debugging

After setting breakpoints in the code editor, select Build → Build Solution to compile the code. Your program should successfully build, to be able to debug. Then select Debug → Start Debugging (or press F5) to begin the debugging process. During debugging, a Command Prompt window appears, allowing application interaction (input/output).
Examine Program Execution

Program execution suspends at the first breakpoint, and the IDE becomes the active window. The yellow arrow to the left of the line indicates that this line contains the next statement to execute.

```c++
    #include <iostream>

    int main() {
        int a = 10;
        int b = 20;
        int c = 30;

        // Program body
    }
```

Examine Data

In break mode, the variables can be examined using the Locals Window of the debugger. Select Debug → Windows → Locals to view these. You will notice when you move the mouse over or select simple expressions, Visual Studio will actually show you the current value of the variable or expression. Note that some variables will contain values even if you do not assign one to those. This occurs because C++ does not specify a default value for variables when they are created. One can change the value of a variable on the local window by clicking on the value field. By selecting Debug → Window → Watch one can evaluate arithmetic and Boolean expression on the watch window by typing the appropriate expressions with the names of the program variables. This particular window can also be used to see the value and type of any program variable by just typing it on the left column and pressing enter.

Executing one C++ statement at a time (also called single stepping)

Once you are in the break mode, you can execute once C++ statement at a time. This is a great way to trace the execution of your program. To single step, select Debug → Step Over (or press F10). This will execute the next statement and stop. Every time you press F10, it will execute the next statement. In later exercises, we will explore the other stepping commands (step into and step out). You can do these by using the toolbar that appears, as well.

Using the Continue Command to Resume Execution

To resume execution, select Debug → Continue (or press F5). The application executes until it stops at the next breakpoint. If there is no other break point, the program will execute to completion and the Command Prompt window will close. Note that if you set your break point at the middle of a statement that spans more than one line, the break point will move itself at the end of that statement. Also note that if you place your mouse pointer on a variable name, the value of the variable will be displayed in a Quick Info box. This is like peeking inside the computer at the value of one of your variables. It can help one to spot logic errors in the program.
**Disable a Break Point**
Right click a line of code on to which a breakpoint has been set and select Disable Breakpoint. The breakpoint will become a hollow maroon circle. Disabling rather than removing a breakpoint allows one to re-enable the breakpoint by clicking inside the hollow circle. This can also be done by right clicking the line marked by the hollow maroon circle and selecting Enable Breakpoint.

**Deleting a Breakpoint**
Click on the maroon circle or right click the line of code containing the breakpoint and select Delete Breakpoint.

**Conditional Break**
You can also break depending on a condition that you set, for example if a value of an expression is equal to zero. To do this, right click on the breakpoint and select condition and type in the condition using C++ syntax.