Debugging with Visual Studio

Introduction
In the software development life cycle, testing and defect fixing take more time than actually code writing. In general, debugging is a process of finding out defects in the program and fixing them. Defect fixing comes after the debugging, or you can say they are co-related. When you have some defects in your code, first of all you need to identify the root cause of the defect, which is called the debugging. When you have the root cause, you can fix the defect to make the program behavior as expected.

Now how to debug the code? Visual Studio IDE gives us a lot of tools to debug our application. Sometimes debugging activity takes a very long time to identify the root cause. But VS IDE provides a lot of handy tools which help to debug code in a better way. Debugger features include error listing, adding breakpoints, watch variables, visualize the program flow, control the flow of execution and many more. Many of them are very common for many developers and many are not. This quick tutorial will cover two common features of VS IDE for debugging: Breakpoint and Watch Window.

1. How to start
You can start debugging from the Debug menu of VS IDE. From the Debug Menu, you can select "Start Debugging" or just press F5 to start the program. If you have placed breakpoints in your code, then execution will begin automatically.

![Debug Menu](image)

2. Breakpoints
Breakpoint is used to notify debugger where and when to pause the execution of program. You can put a breakpoint in code by clicking on the side bar of code or by just pressing F9 at the front of the line. So before keeping a breakpoint, you should know what is going wrong in your code and where it has to be stopped. When the debugger reaches the breakpoint, you can check out what's going wrong within the code by using a different debugging tool.

2.1 Debugging with breakpoints
You have already set a breakpoint in your code where you want to pause the execution. And now start the program by pressing "F5". When the program reaches the breakpoint, execution will automatically pause. Now you have several options to check your code. After hitting the breakpoint, breakpoint line will show as yellow color which indicates that this is the line which will execute next.
Now you have several commands available in break mode, using which you can proceed for further debugging.

**STEP OVER**
After debugger hits the breakpoint, you may need to execute the code line by line. "Step Over" [ F10 ] command is used to execute the code line by line. This will execute the currently highlighted line and then pause. If you select F10 while a method call statement is highlighted, the execution will stop after the next line of the calling statement. Step Over will execute the entire method at a time.
STEP INTO
This is similar to Step Over. The only difference is, if the current highlighted section is any methods call, the debugger will go inside the method. Shortcut key for Step Into is "F11".

STEP OUT
This is related when you are debugging inside a method. If you press the Shift - F11 within the current method, then the execution will complete the execution of the method and will pause at the next statement from where it called.

CONTINUE
It's like run your application again. It will continue the program flow unless it reaches the next breakpoint. The shortcut key for continue is "F5".

SET NEXT STATEMENT
This is quite an interesting feature. Set Next Statement allows you to change the path of execution of program while debugging. If your program paused in a particular line and you want to change the execution path, go to the particular line, Right click on the line and select "Set Next Statement" from the context menu. You will see, execution comes to that line without executing the previous lines of code. This is quite useful when you found some line of code may causing breaking your application and you don't want to break at that time. Shortcut key for Set Next Statement is Ctrl + Shift + F10.
3. **Watch Windows**
You can say it is an investigation window. After breakpoint has been hit, the next thing you want to do is to investigate the current object and variables values. When you mouse hover on the variable, it shows the information as a data tip which you can expand, pin, import which I have already explained. There are various types of watch windows like Autos, Local, etc. Let's have a look into their details.

### 3.1 Locals
It automatically displays the list of variables within the scope of current methods. If your debugger currently hits a particular breakpoint and if you open the "Autos" window, it will show you the current scope object variable along with the value.

![Locals](image)

### 3.2 Autos
These variables are automatically detect by the VS debugger during the debugging. Visual Studio determines which objects or variables are important for the current code statement and based on that, it lists down the "Autos" variable. Shortcut key for the Autos Variable is "Ctrl + D + A".

![Autos](image)

### 3.3 Watch
Watch windows are used for adding variables as per requirement. It displays variables that you have added. You can add as many variables as you want into the watch window. To add
variables in the watch window, you need to “Right Click” on variable and then select “Add To Watch”. If you want to delete any variable from watch window, just right click on that variable and select "Delete Watch". From the debug window, you can also edit the variable value at run time.

Reference