**Debugging in Visual Studio**

**Introduction**

In the Visual Studio context, when you debug your app, it usually means that you are running the application with the debugger attached (that is, in debugger mode). When you do this, the debugger provides many ways to see what your code is doing while it runs.

You can step through your code and look at the values stored in variables, you can set watches on variables to see when values change, you can examine the execution path of your code, etc.

**Common Shortcuts for Debugging**

* **Start Debugging**: F5
* **Stop Debugging**: Shift + F5
* **Restart Debugging**: Ctrl + Shift + F5
* **Start Without Debugging**: Ctrl + F5

**Debug points**

Breakpoints are a useful feature when you know the line of code or the section of code that you want to examine in detail at runtime.

Breakpoints are one of the most important debugging techniques in your developer's toolbox. You set breakpoints wherever you want to pause debugger execution. For example, you may want to see the state of code variables or look at the call stack at a certain breakpoint.

To set a breakpoint in source code:

* Click in the far left margin next to a line of code.
* select the line and press **F9**
* select **Debug** > **Toggle Breakpoint**
* right-click and select **Breakpoint** > **Insert breakpoint**

**Breakpoints** window :

* select **Debug** > **Windows** > **Breakpoints**
* press **Ctrl**+**Alt**+**B**

**Types of breakpoints:**

**Standard Breakpoint**

* A regular breakpoint that pauses the program’s execution when it reaches a specific line of code.

**Conditional Breakpoint**

* This breakpoint only triggers when a specific condition is met.

**Temporary Breakpoint**

* A temporary breakpoint automatically removes itself once it's hit.

**Different Debug windows**

**Immediate Window**

This window lets you interact with the debugger by executing expressions, inspecting variables, and even testing methods while the application is paused.

To open it: Debug > Windows > Immediate.

**Watch Window**

The Watch Window lets you monitor the values of specific variables or expressions over time as your program runs.

To open it: Debug > Windows > Watch > Watch 1 (or 2, 3).

**Call Stack Window**

The Call Stack shows you the stack of function calls that led to the current execution point. It’s very useful for tracing how the program reached a particular state, especially when debugging deeper, recursive functions or trying to understand the flow of execution.

To open it: Debug > Windows > Call Stack.

**Locals Window**

This window automatically displays all the variables in the current scope (method or function) and their values as the program is paused at a breakpoint. It updates dynamically as you step through your code.

To open it: Debug > Windows > Locals.

**Autos Window**

The Autos window shows variables used around the current statement where the debugger is paused.

To open it: Debug > Windows > Autos.

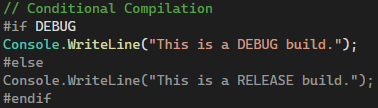
**Editing**

Modify code during debugging without restarting the application.

* Steps:
  + Pause execution at a breakpoint.
  + Edit the code directly in the editor.
  + Press F5 to continue execution with the updated code.

**Conditional Compilation**

Include or exclude code during compilation using preprocessor directives like #if, #else, and #endif:

****

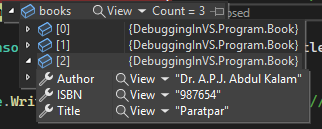
**Stepping Through Code**

Control the flow of code execution during debugging:

* Step Over (F10): Execute the current line and move to the next line.
* Step Into (F11): Move into the called method to debug it.
* Step Out (Shift + F11): Exit the current method and return to the caller.

**Data inspector**

The Data Inspector in Visual Studio 2022 is a debugging tool that helps developers view and interact with data during debugging sessions. It allows you to inspect variables, collections, objects, and data structures in a more user-friendly and detailed way than the standard debugging windows.

****