1. Introduction

Debugging refers to the process of discovering defects (bugs) in software and correcting them. This process is invoked when your code has successfully built, but does not perform according to the software specifications. The Microsoft Visual Studio integrated development environment (IDE) provides excellent capability for accurately and efficiently locating bugs within your code. In this tutorial, we cover the basic debugger functionality that will be needed for this course.

2. Before you start

Before you start a debugging session, build your code by pressing F7. Assuming your code built successfully, you can proceed to setting your breakpoints.

2. Setting Breakpoints

A breakpoint is used to notify the debugger where to pause the execution of your code. A breakpoint can be placed at any line of your code by left-clicking on the side bar of your code, or pressing F9 while at the front of the line where you want to break.

```c
int main(void)
{
    int i, j;  // loop control variables
    printf("\t\ti \t\tj \n");  // prints column labels
    for (i = 1; i < 4; ++i) // outer loop
    {
        printf("Outer\%4d\n", i);
        for (j = 0; j < i; ++j) // inner loop
        {
            printf("Inner\t\%9d\n", j);
        }
    }
    return (0);
}
```

Figure 1: Setting a breakpoint by left-clicking on the sidebar

Breakpoints can be set at any line of your code, but you will find it most useful to place breaks at
- the beginning of loops,
- if-then-else statements,
• function calls.

3. Running the Debugger

Once you have set various breakpoints within your code, you can initiate the debugging process by pressing **F5** or following the menu shown in Figure 2.

![Figure 2: Initiating the debugging process.](image-url)

Once in break mode, your code will pause at the first breakpoint. Here, you have several options for continuing the code execution:

- Press **F10** to do a “Step Over” operation. This command will execute the code line-by-line. Function calls will be executed as a single statement.
- Press **F11** to do a “Step Into” operation. This is a similar operation with a step over operation except in statements that involve function calls. For function calls, the code execution will be transferred within the function.
- Press **Shift + F11** to do a “Step Out” operation. This operation will complete the execution of a function that you had previously stepped into.
- Press **F5** to continue the execution to the next breakpoint.

4. Setting Conditional Breakpoints

Suppose you iterate through a large amount of data (e.g., a large for loop) and you want to debug a few of them. That is, you want to pause at a breakpoint only if some specific condition is met. Visual Studio Breakpoints allow you to set conditional breakpoints in which execution is paused only if the selected condition is met.
To set a conditional breakpoint, first set your breakpoint as in Section 2. Then **right-click** on the red breakpoint icon and select “Condition...”.

In the appearing context menu, set your breakpoint condition.

The code execution will stop at the conditional breakpoint when at the breakpoint line and the breakpoint condition is true.

**5. Setting Hit Count Breakpoints**

The Breakpoint hit count is used to pause the code execution when a breakpoint has been executed a fixed number of times. To set a hit count condition, **right-click** on the red breakpoint icon and select “Hit Count...”
The breakpoint hit count window provides you with the following options:

- Break always
- Break when the hit count is equal to a specified number
- Break when the hit count is a multiple of a specified number
- Break when the hit count is greater than or equal to a specified number

Select the appropriate option and press OK.

6. Watching variables

During the debugging process, you can watch the values of your declared data types using the local and watch window. The local window provides you only with a list of variables related to the scope of the function that currently execute. The watch window allows you to explicitly select the variables that you would like to watch.
To watch a variable, **right-click** on the variable name within your code and press “Add Watch”.

![Figure 7: Watching a variable of choice](image1)

Alternatively, you can watch a variable by simply typing its name on the rows within the watch window.

![Figure 8: The Watch window](image2)