Tutorial: VSCode Debugger

Debugging is an important skill for programmers which consists of finding and fixing errors or bugs in their codes. Visual Studio Code (VSCode) has a built-in debugging tool which helps the programmer to examine the code execution at selected statements or at each statement execution.

This document gives a brief description of the essential functionalities of the debugging tool in VSCode; for further details, explore further VSCode interface and study the relevant documentation, see reference.

To Launch the Debugger, select “Debug C/C++ File” to run your program.

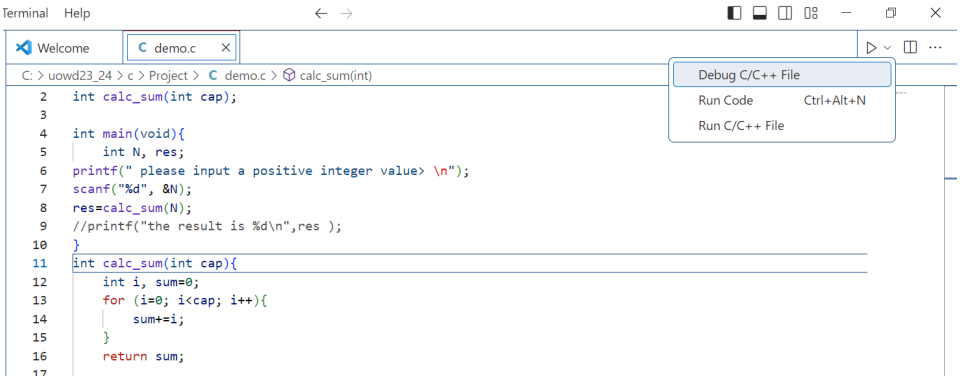

Figure 1

I. Breakpoint

Breakpoint is useful to **pause** the execution of a program at specific line; you can set many in your program. To set a breakpoint, you can simply click on the left margin of the code in the editor window, at left of the line number where you want the program execution to pause**;** you will see a red dot indicating the breakpoint (alternatively, right click at the same location and select “Add Breakpoint”). To delete a breakpoint, just click on it and the red point disappears.

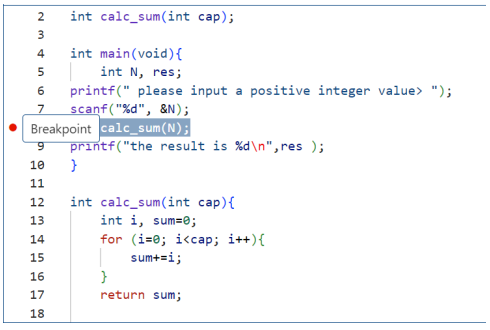

Figure 2

The debugger in VSCode supports **different types of breakpoints**. To access them, right click on your added breakpoint red dot and select "Edit Breakpoint", a drop-down list appears (see Figure 3), to choose between the following breakpoint types:

a. **Expression**: the corresponding breakpoint is hit, i.e., program execution pauses at the breakpoint, when the input expression evaluates to true. Example of expression in the above program: N==10

b. **Hit Count:** this is useful with loops when the developer wants to check the state of the program at a particular loop iterator value. The Breakpoint hits when the hit count condition is met, e.g., i==3

c. **Log Message:** to log a message when the breakpoint is hit. The log message will be displayed under the "Debug Console" tab in VSCode. You put the variable you would like to print its value between {}, e.g., log Message: N value is {N}

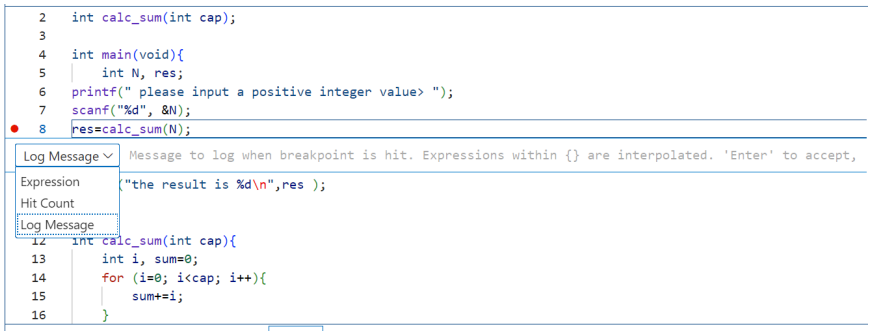

Figure 3

"Log Message" can also be added by right clicking at left of the line number of your selected program statement, then select "Add Logpoint". **"Log Message"** allows a developer to log the values of the variables (s)he would like to inspect during the development of the application without a need for **printf** statements. The log points can be removed once the implementation of the application requirements has been verified.

II. Tracing of Program execution

When you run your program using "Debug C/C++ file" (see Figure 1), the program execution will eventually pause at your breakpoint whose statement in the editor window gets highlighted in yellow; a **debug execution toolbar** appears at the top of the editor window, see Figure 4. The toolbar allows the developer

to choose how to carry on the execution of the program; **the values of the variables within the scope of the breakpoint statement can be accessed by hovering the computer mouse over them.**

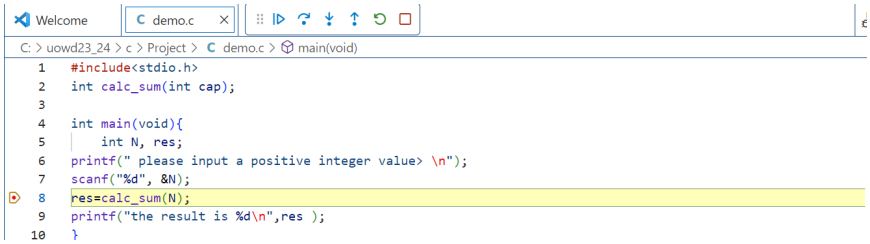

Figure 4.

Figure 5 gives the different debug modes of the program execution:

- **Step by step** : to execute the code line by line.
- **Step over:** will advance the debugger to the next statement, useful mainly to step over a function call execution, in order to skip tracing the execution of the function statements.
- **Step into:** this is the opposite of the Step over, used in order to trace the function code execution.
- **Step out:** to be used while inside a function code, to exit the function and get back to its caller

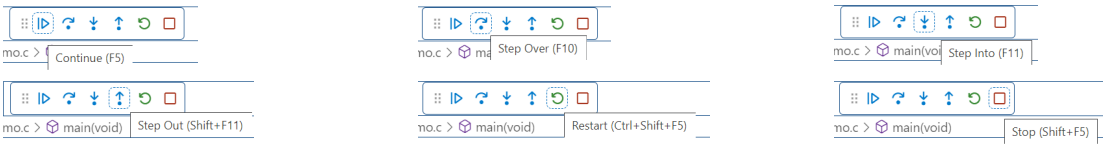

Figure 5: Debug execution toolbar

Furthermore, at the left side of the IDE, you can find three windows:

- **Variables window: i**t shows the local variables (Locals) within the scope of the current statement execution.
- **Watch window:** it allows to track the value of a variable within the scope of the current executed statement. To add/remove a watched variable, click on + / x icon as shown in Figure 6.

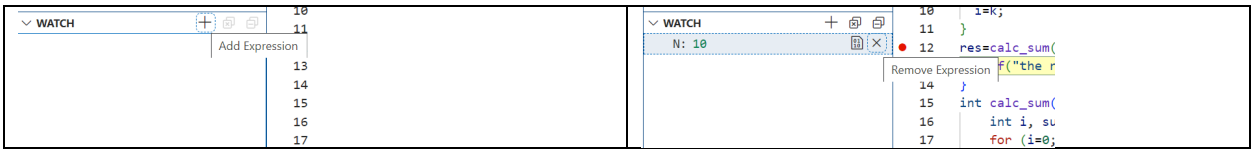

Figure 6

- **Call stack window** This window allows to see the execution path of your program in terms of function calls and returns. In fact, each time a method is invoked an entry is added to this window, which is later removed when the called function is exited.

Further reference:
https://code.visualstudio.com/docs/editor/debugging