National University of Singapore
School of Continuing and Lifelong Education

TIC1001: Introduction to Computing and Programming I
Semester I, 2020/2021

Lab 4
Using the Visual Studio Code Debugger

Introduction

Visual Studio Code has an introduced a visual debugger interface that allows you to examine the memory state of your program as you step through the execution, similar to C++ Tutor. This might be a useful tool to use during your practical exam. This lab will teach you how to run the debugger and debug a sample program.

Obtaining the files

1. Obtain the zip file for the lab from Coursemology. You will be given a similar zip file for the PE.
2. Extract the contents of the zip file to a directory, e.g. Desktop or Documents.
    - For Mac, double clicking the zip file will extract the contents to the current folder.
    - For Windows, double clicking will open a window to examine the zip file. You will have to drag the folder out to extract the contents.

   You should now see a folder called `lab04` containing at least two files: `lab04.cpp` and `lab04.code-workspace`.

Note for MacOS
In later versions of MacOS, the System Integrity Protection (SIP) is enabled, disallowing the VS Code debugger to communicate with the GDB debugger. SIP may have to be disabled if you wish to use the VS Code debugger.
To disable SIP:
1. Restart your Mac into Recovery mode, by holding down ⌘+R from when it restarts until you see the Apple icon and a progress bar.
2. From the **Utilities** menus, select **Terminal**.
3. At the terminal prompt, type `csrutil disable` exactly and press **Enter**. The terminal should display a message saying that SIP was disabled.
4. From the **Apple** menu, select **Restart**.

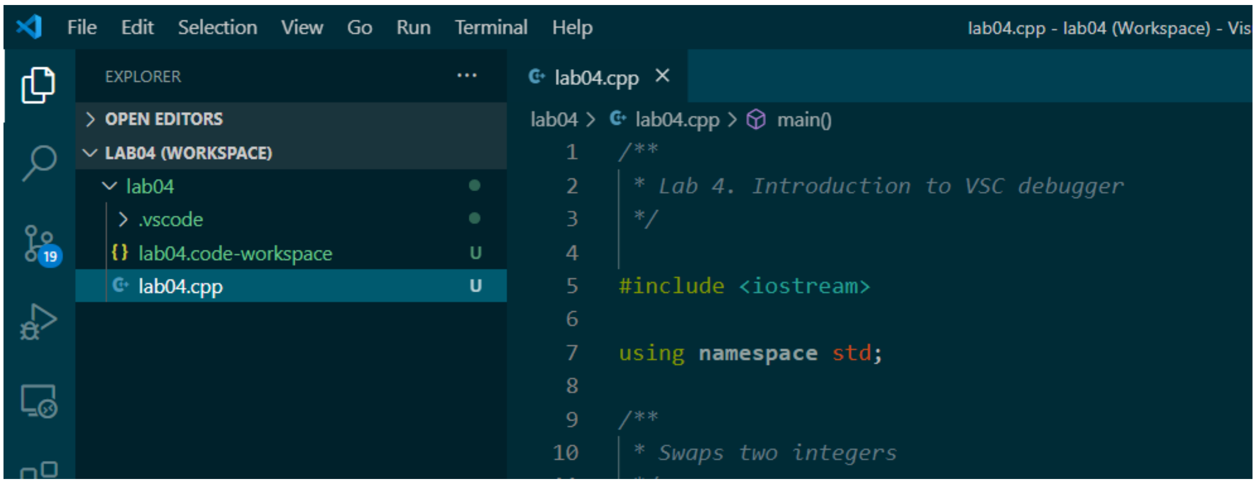
SIP can be re-enabled by following the same steps, but entering `csrutil enable` instead.

Starting Visual Studio Code

Normally you will open the file `lab04.cpp` using Visual Studio Code. But in order to use the debugger, you will instead open the file `lab04.code-workspace` using Visual Studio Code instead.

If double-clicking on the file does not work, simply select **File** from the menu in Visual Studio Code, and **Open Workspace...** and open the file.

You should see the `lab04.cpp` file open for editing in VS Code:

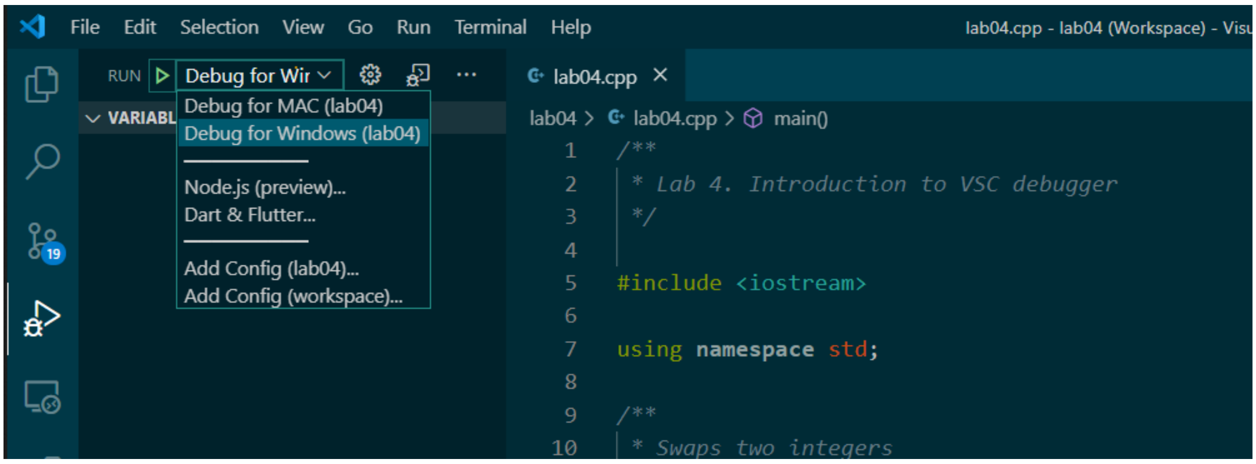

File Edit Selection View Go Run Terminal Help
lab04.cpp - lab04 (Workspace) - Vis
EXPLORER
OPEN EDITORS
LAB04 (WORKSPACE)
lab04
.vscode
lab04.code-workspace U
lab04.cpp U
19
lab04.cpp
lab04 > lab04.cpp > main()

```
/**
 * Lab 4. Introduction to VSC debugger
 */

#include <iostream>

using namespace std;

/**
 * Swaps two integers
```

If not, simple click on the filename on the explorer pane on the left. You may now work on the cpp file as usual.

Running the debugger

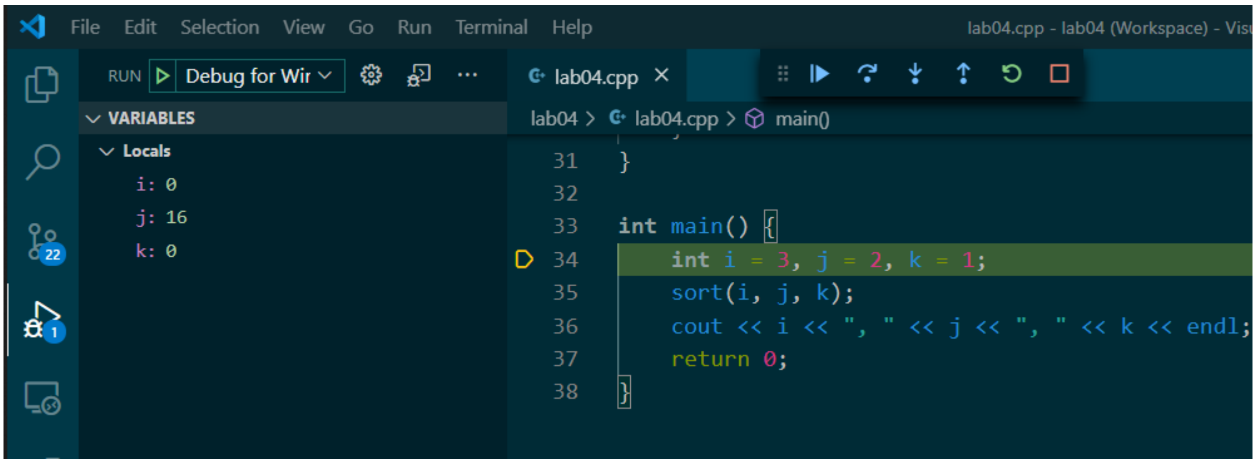
Switch the left pane to the debugger by clicking on the icon. At the top you should see a **Run** button, with two options: Debug for MAC and Debug for Windows.


File Edit Selection View Go Run Terminal Help
lab04.cpp - lab04 (Workspace) - Visu
RUN
Debug for Wir
VARIABL
Debug for MAC (lab04)
Debug for Windows (lab04)
Node.js (preview)...
Dart & Flutter...
Add Config (lab04)...
Add Config (workspace)...
19
lab04.cpp
lab04 > lab04.cpp > main()

```
/**
 * Lab 4. Introduction to VSC debugger
 */

#include <iostream>

using namespace std;

/**
 * Swaps two integers
```

Click on the appropriate option. These configurations should work if you had followed the setup given in Problem Set 0. Otherwise, you might have to modify them yourself.

Your code will be compiled and run, and the execution should be paused with the very first step highlighted.


File Edit Selection View Go Run Terminal Help
lab04.cpp - lab04 (Workspace) - Visu
RUN
Debug for Wir
lab04.cpp
lab04 > lab04.cpp > main()
VARIABLES
Locals

```
i: 0
j: 16
k: 0
```

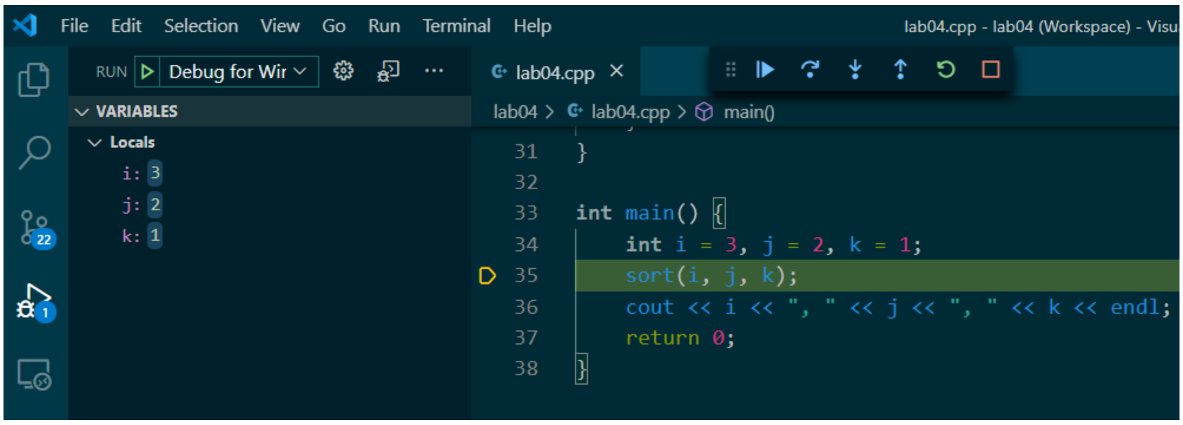
```
}

int main() {
    int i = 3, j = 2, k = 1;
    sort(i, j, k);
    cout << i << ", " << j << ", " << k << endl;
    return 0;
}
```

The debugging view

The left pane shows the memory state of the program at the current step. On the code pane, the current instruction to be executed is highlighted. Since the variables are not yet initialized, we see they contain random values in memory.

Somewhere at the top there is a controls bar (that can be repositioned by dragging). Mouse over the buttons will reveal that they are **Continue**, **Step Over (F10)**, **Step Into (F11)**, **Step Out (Shift+F11)**, **Restart** and **Stop**.

1. Click or press the **Step over** icon in the debugging control panel. This will advance the program execution by one step. You should see that the variables have been correctly initialized.

   
   File Edit Selection View Go Run Terminal Help
   lab04.cpp - lab04 (Workspace) - Visu
   RUN
   Debug for Wir
   lab04.cpp
   lab04 > lab04.cpp > main()
   VARIABLES
   Locals

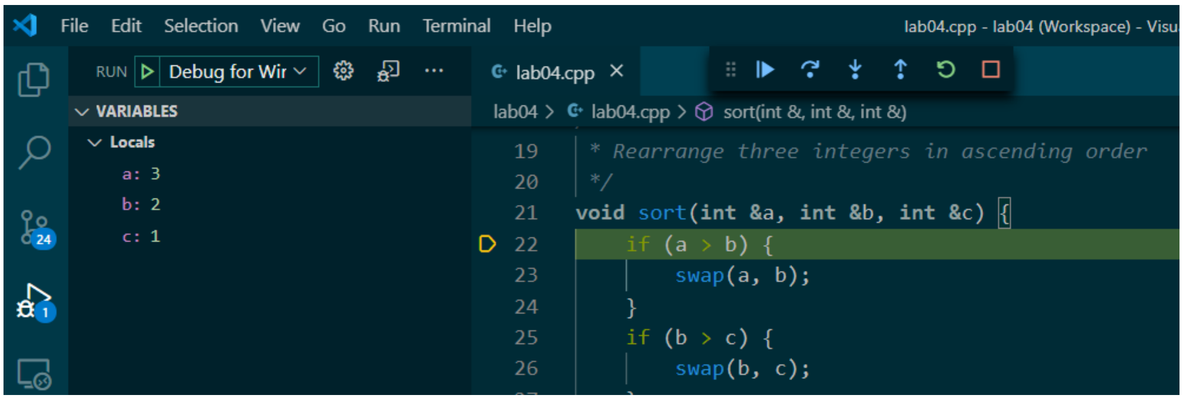
   ```
   i: 3
   j: 2
   k: 1
   ```

   ```
   }

   int main() {
       int i = 3, j = 2, k = 1;
       sort(i, j, k);
       cout << i << ", " << j << ", " << k << endl;
       return 0;
   }
   ```

2. Now the next line to be executed is function call to `sort`. If you were to click **Step over** button again, the execution will "step over" that line by performing the function call without pausing. Try it.

   You will notice that the variables did not change at all after the function call. Something must be wrong in the function.

3. So restart the debugger and come back to the previous step. This time, click or press **Step into** the function call.

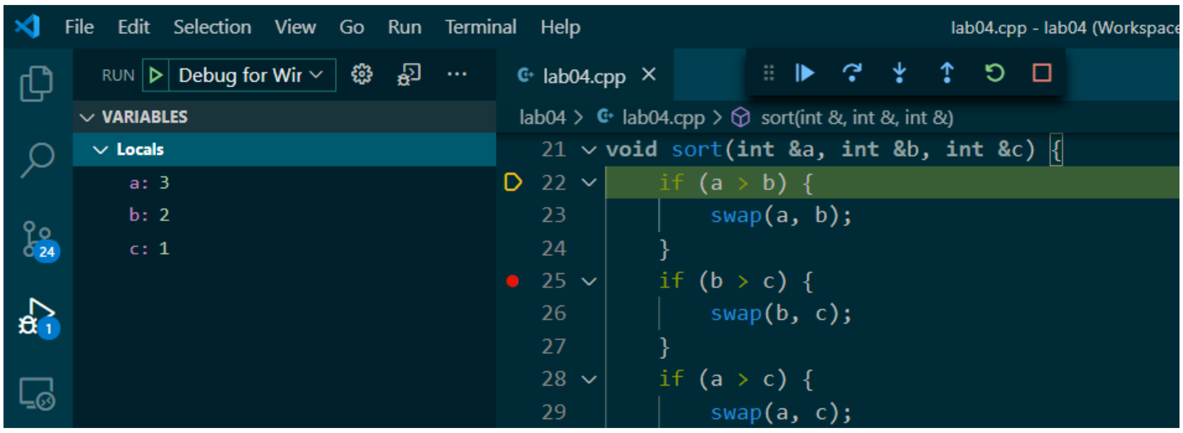

File Edit Selection View Go Run Terminal Help
lab04.cpp - lab04 (Workspace) - Visu
RUN Debug for Wir
lab04.cpp
lab04 > lab04.cpp > sort(int &, int &, int &)
VARIABLES
Locals

```
a: 3
b: 2
c: 1
```

```
 * Rearrange three integers in ascending order
 */
void sort(int &a, int &b, int &c) {
    if (a > b) {
        swap(a, b);
    }
    if (b > c) {
        swap(b, c);
```

Notice that the execution is now at the sort function and our variables are now replaced with the inputs `a`, `b` and `c`.

4. Now use the **Step over** or **Step into** buttons to control the execution of the program and view the memory state changes.
5. If you do not want to continue tracing through a function, or you stepped into a function in the C++ standard library, you can return using two ways. The first is to press **Step out** to continue execution until it steps out of the current function. The second is to put a breakpoint on a line of code and press **Continue**. The program will execute until it hits a breakpoint.
6. To put a breakpoint, click on the left of the line number of the line where you want the execution to pause. A red circle should appear, indicating a breakpoint has been created.


File Edit Selection View Go Run Terminal Help
lab04.cpp - lab04 (Workspace
RUN Debug for Wir
lab04.cpp
lab04 > lab04.cpp > sort(int &, int &, int &)
VARIABLES
Locals

```
a: 3
b: 2
c: 1
```

```
void sort(int &a, int &b, int &c) {
    if (a > b) {
        swap(a, b);
    }
    if (b > c) {
        swap(b, c);
    }
    if (a > c) {
        swap(a, c);
```

Fixing the code

There are some mistakes (bugs) in this code. Now try using the debugger to find what is wrong and fix it.