C/C++ Training

Workstation Setup Guide

DataBank IMX

January 12, 2024

DataBank
A KYOCERA GROUP COMPANY

Setting Up Your Workstation for C/C++ Development

Performing C/C++ development requires a few components to be installed on your workstation. You can, of course, install whatever IDE you prefer, but I will be teaching the class using Visual Studio Code (a popular open-source IDE) and other convenience tools that you may find useful. If you want to set up your development environment similar to mine, the instructions below will walk you through the setup process. If you're following these steps, it is best to do them in the order presented.

Table of Contents

Prerequisites

Prior to any of the steps listed below, please complete all tasks in the guide titled:

Setting Up Your Workstation for Development Training.pdf

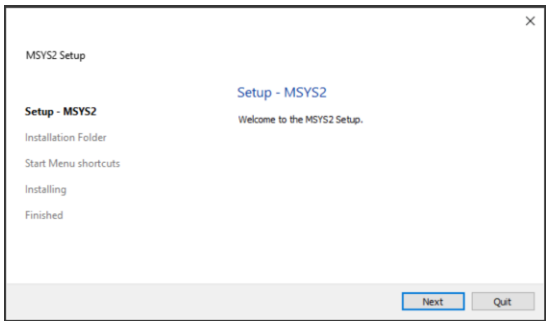
Install the MingW (MSYS) Components

Before we can write and execute C and C++, we need to install tools for compiling, IntelliSense, etc.

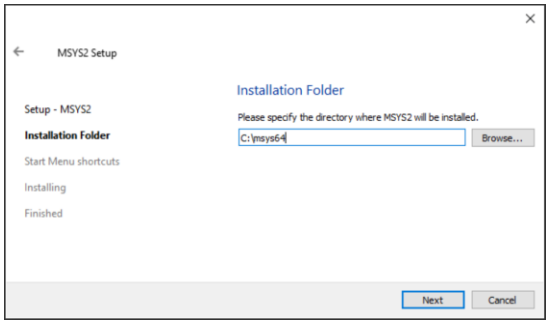
1. Download the MSYS2 Installer from the URL below:
   [GitHub Link to MSYS2 Installer]

2. Run the executable.

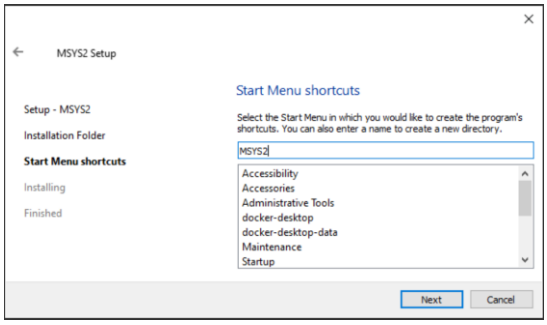
3. Click [Next]



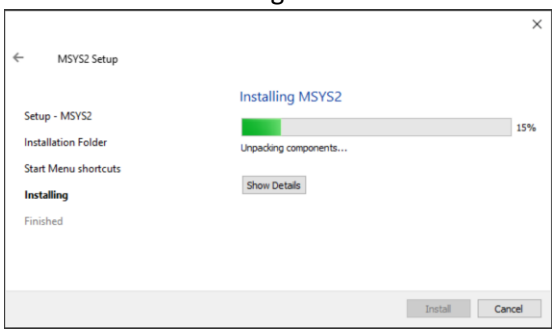
4. Leave the default path (C:\msys64) and click [Next]



5. Leave the default start menu option (MSYS2) and click [Next]



6. The installation will begin

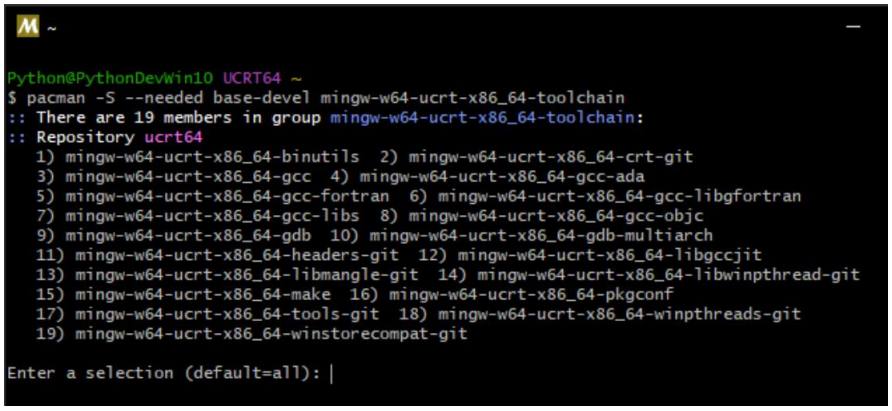


Note: This installation sometimes takes quite a while. It may appear to hang at 50%, but just be patient. It will complete.

7. When the installer completes, leave the "Run now" box checked and click [Finish]

8. You will be presented with the MingW terminal, where you will run the following command:
`pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain`

9. After the list pre-loads, select the default "all" option by pressing <ENTER>



10. When the download size calculates, enter `y` to proceed with installing all components

```
              mingw-w64-ucrt-x86_64-winstorecompa

Total Download Size:    165.03 MiB
Total Installed Size:  1046.89 MiB

:: Proceed with installation? [Y/n] y
```

11. A series of installations will begin

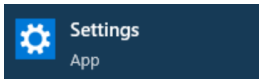
```
:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 mingw-w64-ucrt-x86_64-gcc-...     3.0 MiB  1116 KiB/s 00:23 [###----------------------------]  10%
 mingw-w64-ucrt-x86_64-gcc-...     6.3 MiB  1458 KiB/s 00:10 [#########----------------------]  30%
 mingw-w64-ucrt-x86_64-pyth...     6.7 MiB  1554 KiB/s 00:07 [###########--------------------]  36%
 mingw-w64-ucrt-x86_64-gcc-...     6.8 MiB  1573 KiB/s 00:04 [################---------------]  52%
 mingw-w64-ucrt-x86_64-gcc-...     6.7 MiB  1579 KiB/s 00:02 [##################-------------]  59%
 Total ( 0/58)                    29.5 MiB  7.14 MiB/s 00:18 [#####------------------------]  17%
```

12. Once all the installations complete, you will be returned to the default command prompt. At this point, you can close this terminal. We will not need it again for these setup steps.

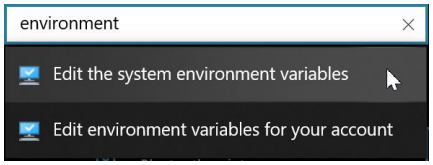
```
(57/58) installing mingw-w64-ucrt-x86_64-tools-git          [################################] 100%
(58/58) installing mingw-w64-ucrt-x86_64-winstorecompat-git [################################] 100%
:: Running post-transaction hooks...
(1/1) Updating the info directory file...

Python@PythonDevWin10 UCRT64 ~
$
```

Update Path Environment Variable

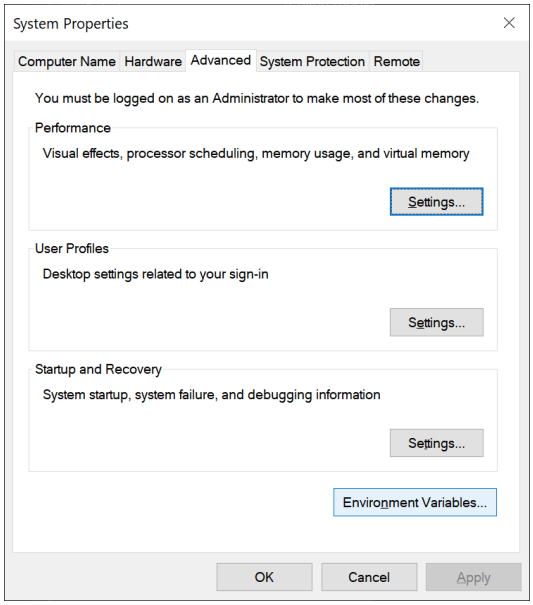
1. In Windows, click the start button, search for “Settings,” and click the Settings application



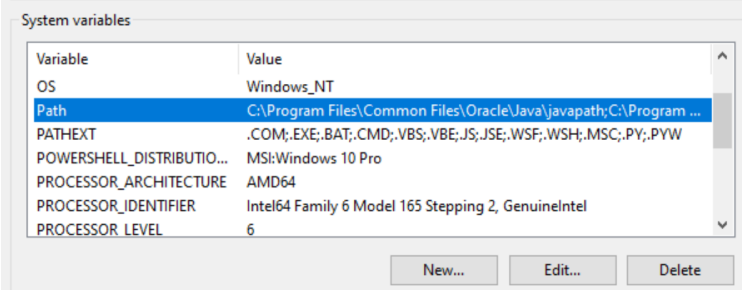
2. Search for “environment” and select the option to “Edit the system environment variables”



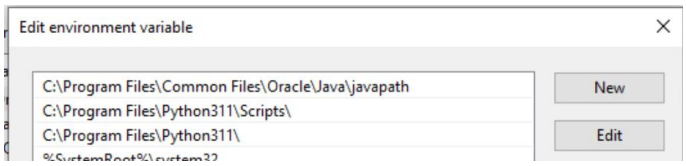
3. When the system properties dialog is presented, click the [Environment Variables] button



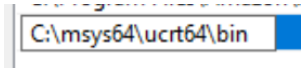
4. In the “System variables” pane, select “Path” and click [Edit]



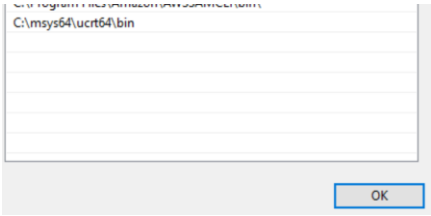
5. Click [New]



6. On the new line that appears, enter C:\msys64\ucrt64\bin and press <ENTER>



7. Click [OK] on all dialogs until you're back to the desktop



Test the MingW Installation

In a terminal, run each of the following commands to validate that the necessary C/C++ tools are installed properly:

1. `gcc --version`

```
Python  ~  19ms  gcc --version
gcc.exe (Rev6, Built by MSYS2 project) 13.1.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

2. `g++ --version`

```
Python  ~  25ms  g++ --version
g++.exe (Rev6, Built by MSYS2 project) 13.1.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

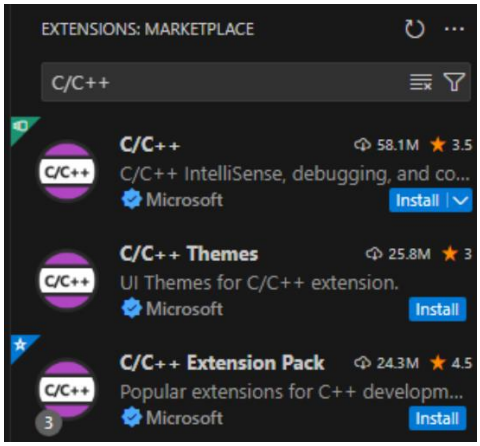
3. `gdb --version`

```
Python  ~  59ms  gdb --version
GNU gdb (GDB) 13.1
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Customize Visual Studio Code

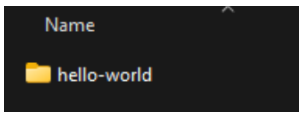
1. Click on the “extensions” icon on the sidebar

2. Search for “C/C++” and install the “C/C++” extension. Optionally, you can also install the “C/C++ Extension Pack,” but that is not required for this setup.

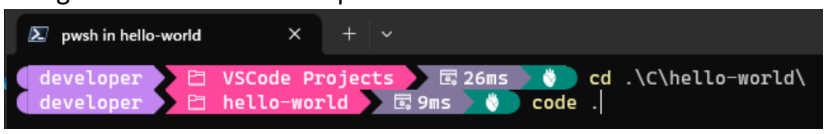


Create a Test Program in C

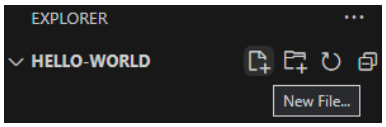
1. Create a folder called “C,” and in it, create a folder called “hello-world”



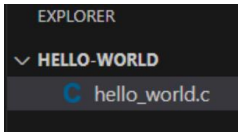
2. Navigate to the folder and open it with Visual Studio Code



3. Click on the “new file” icon in the explorer



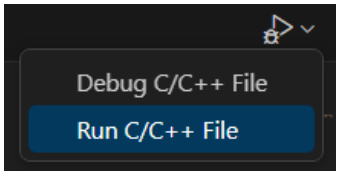
4. Title your file “hello_world.c”



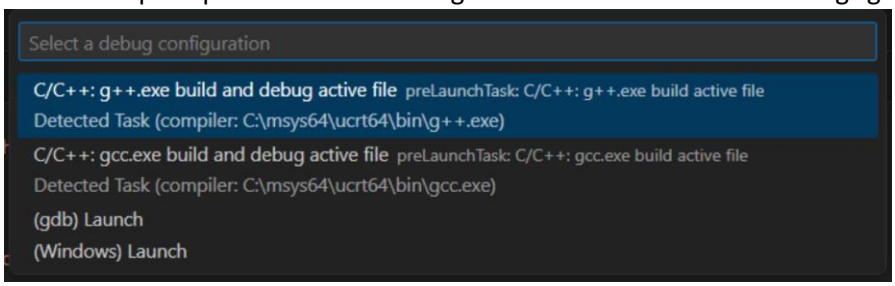
5. Enter the following code

```
#include <stdio.h>
int main() {
    printf("Hello World!");
    return 0;
}
```

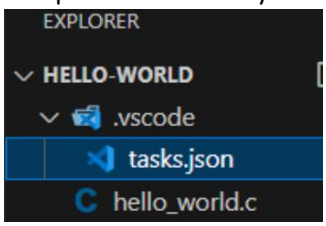
6. Click the drop-down arrow next to the “Run” icon on the toolbar, and select “Run C/C++ File”



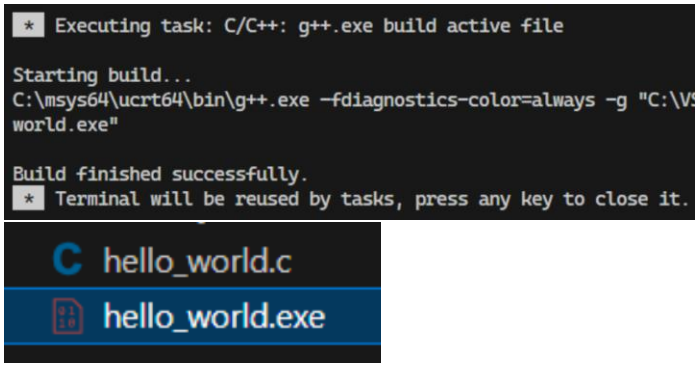
7. You will be prompted to select a configuration. Select the one including “g++.exe build and debug…”



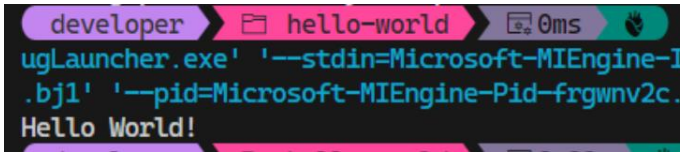
8. This will generate a file called “tasks.json,” which provides the configuration VSCode will use to compile and execute your C program.



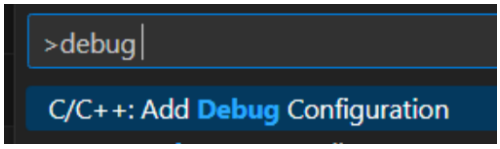
9. VS Code will compile hello_world.exe



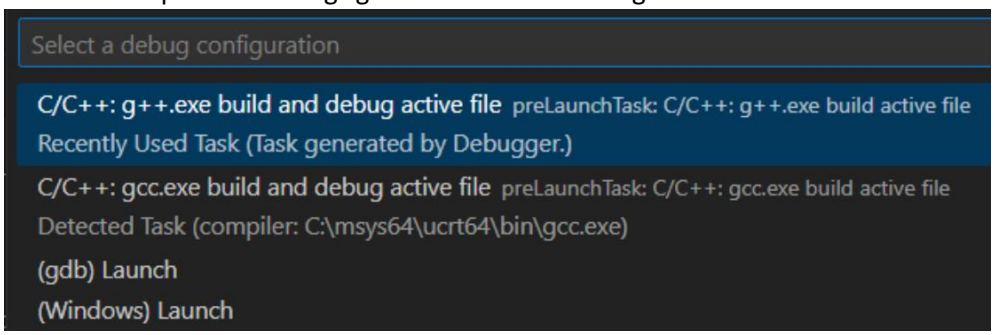
10. And execute it in the terminal



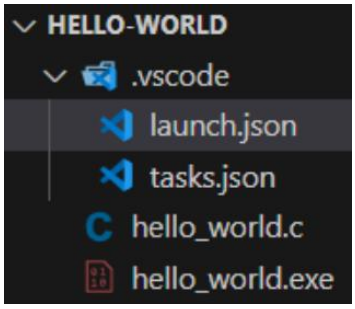
11. Access the menu by pressing CTRL+SHIFT+P. Seach for “debug” and select “C/C++ Add Debug Configuration”



12. Choose the option including “g++.exe build and debug…”

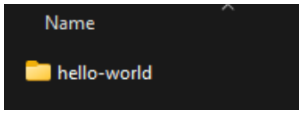


13. This will generate a file called “launch.json,” which will be used for “debug” as opposed to “run” executions.

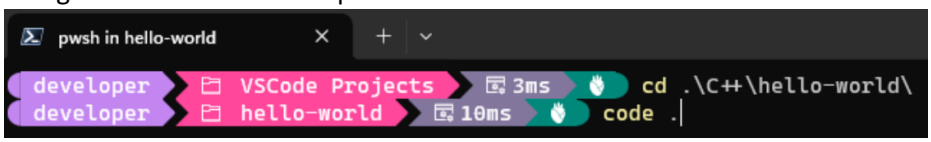


Create a Test Program in C++

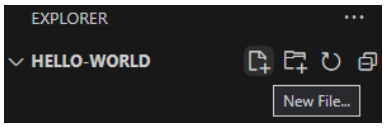
14. Create a folder called “C++,” and in it, create a folder called “hello-world”



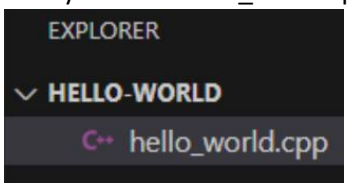
15. Navigate to the folder and open it with Visual Studio Code



16. Click on the “new file” icon in the explorer



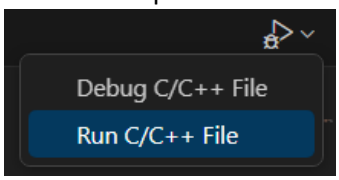
17. Title your file “hello_world.cpp”



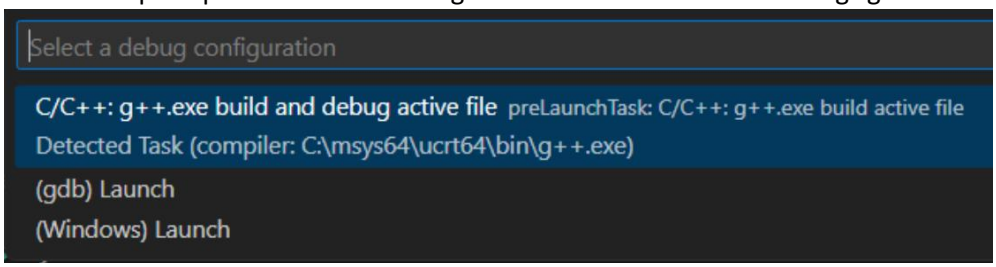
18. Enter the following code

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!";
    return 0;
}
```

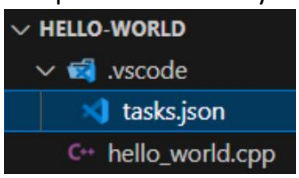
19. Click the drop-down arrow next to the “Run” icon on the toolbar, and select “Run C/C++ File”



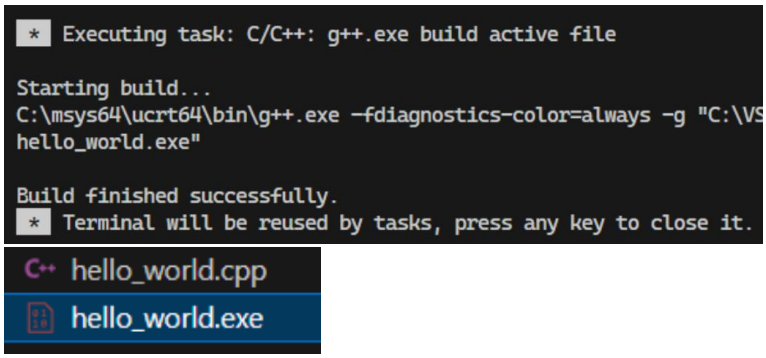
20. You will be prompted to select a configuration. Select the one including “g++.exe build and debug…”



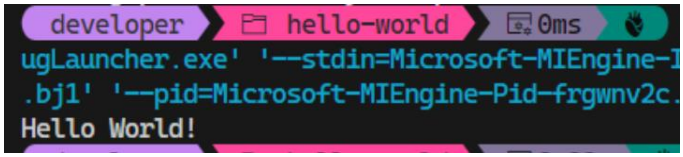
21. This will generate a file called “tasks.json,” which provides the configuration VSCode will use to compile and execute your C program.



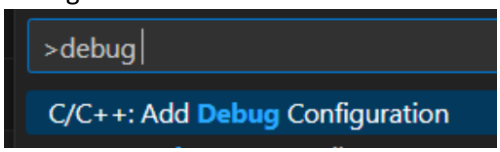
22. VS Code will compile hello_world.exe



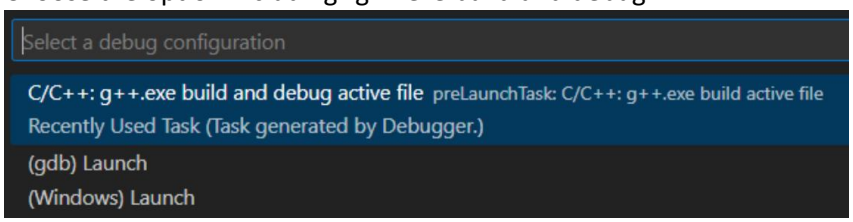
23. And execute it in the terminal



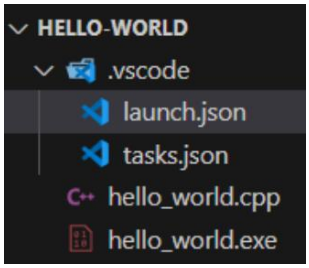
24. Access the menu by pressing CTRL+SHIFT+P. Seach for “debug” and select “C/C++ Add Debug Configuration”



25. Choose the option including “g++.exe build and debug…”



26. This will generate a file called “launch.json,” which will be used for “debug” as opposed to “run” executions.



Clone the C and/or C++ Training Repository

Finally, you'll need to clone a copy of the repository to work with.

I have two different locations where this each is stored:

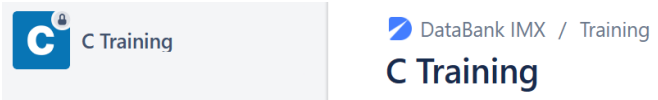
- C Training
  - Bitbucket: https://bitbucket.org/databankimx/c-training
  - GitHub: https://github.com/ZeroKlu/c-training
- C++ Training
  - Bitbucket: https://bitbucket.org/databankimx/c-plus-plus-training
  - GitHub: https://github.com/ZeroKlu/c-plus-plus

1. For access to either repository, email smclean@databankimx.com to request access.

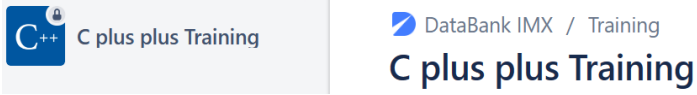
   Be sure to indicate whether you need access to Bitbucket or GitHub and provide the username you use on the selected source control system.

2. In a browser, navigate to the repository you selected and make sure you have access:

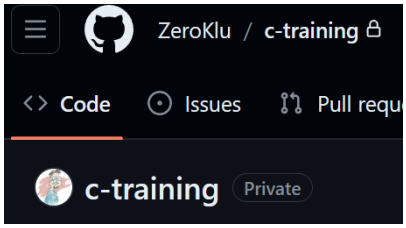
   a. In Bitbucket, you should see this:

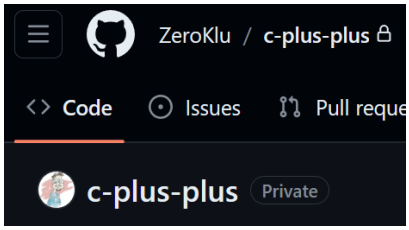
   or

   b. In GitHub, you should see this:

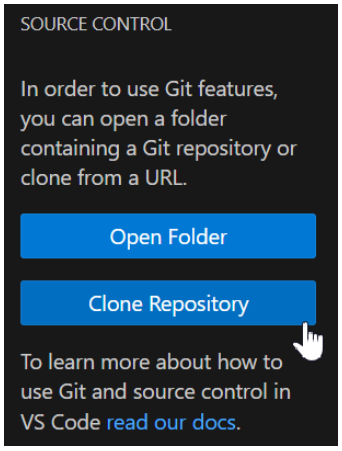
   or

3. Right-click the VS Code icon and select “New Window” to launch an empty instance of VS Code

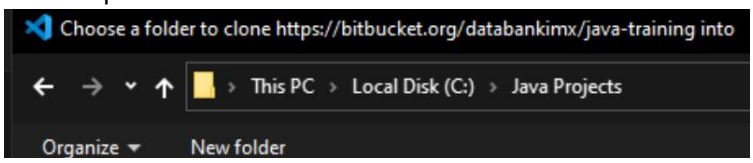
4. Click the Source Control icon on the sidebar

5. Click the button labeled “Clone Repository”

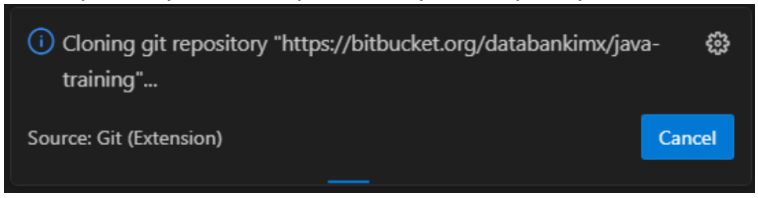


6. Enter the URL to the repository you selected in step 1

7. Select a path

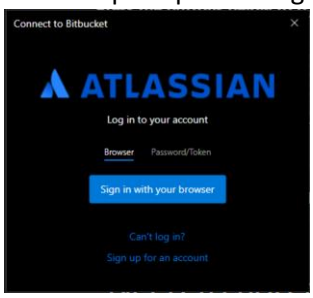


8. The repository will be copied locally to the path you selected

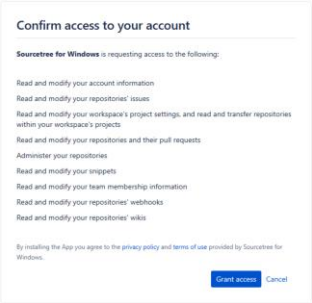


9. If you used the GitHub repository, skip to step 14

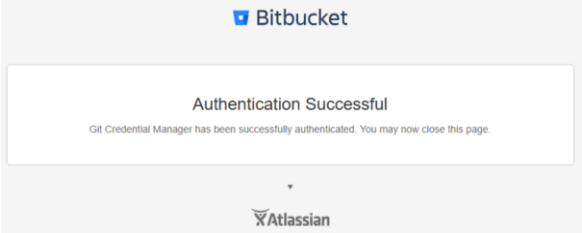
10. You'll be prompted to log in again



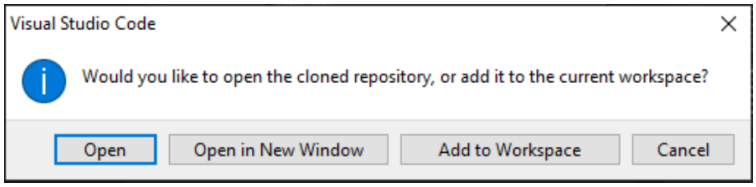
11. Grant access again in the web page that opens



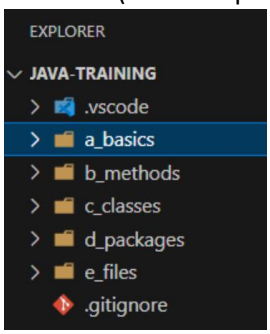
12. You'll see a success alert. After this, you can close the browser



13. When asked if you want to open the repository, click [Open]

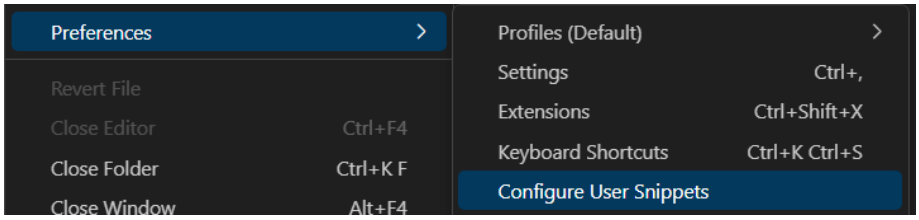


14. The repository will open, and you should see a number of folders containing sample code from the textbook (with samples and commentary from me).



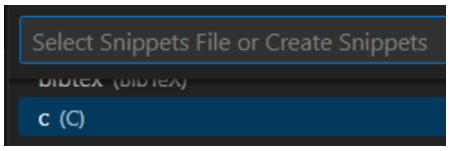
Import Snippets (optional)

1. In VS Code, under the “File” menu, click on “Preferences” > “Configure User Snippets”

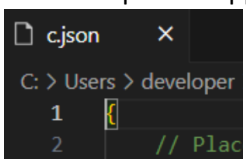


2. For C:

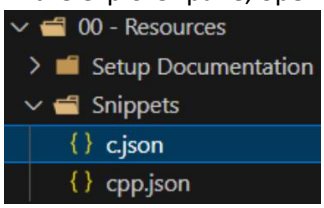
   a. From the languages list, select “c”



   b. This will open a snippets file called c.json



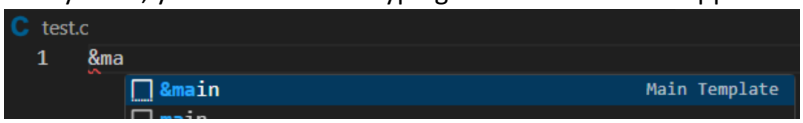
   c. In the explorer pane, open the provided file “c.json”



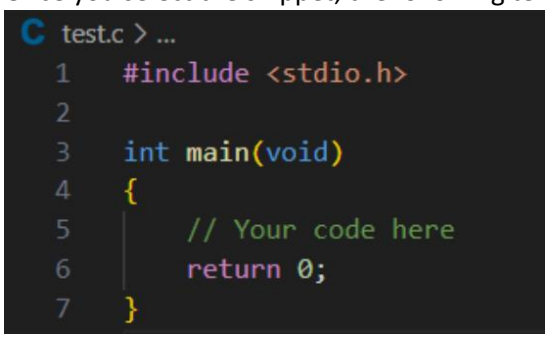
   d. Copy the content into the snippet file opened in step 3, then save the file.

```
{
    "Main Template": {
        "prefix": "&main",
        "body": [
            "#include <stdio.h>",
            "",
            "int main(void)",
            "{",
            "    // Your code here",
            "    return 0;",
            "}",
            ""
        ],
        "description": "Main Template"
    }
}
```

e. In any C file, you can now start typing “&main” and the snippet will be available.

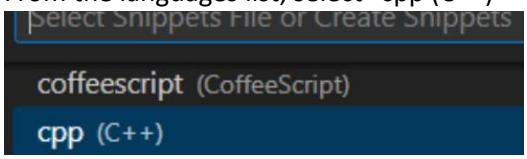


f. Once you select the snippet, the following template code will be added to your file

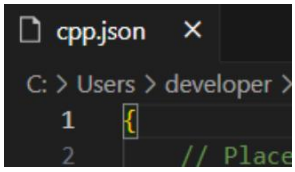


3. For C++:

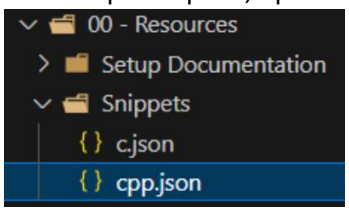
a. From the languages list, select “cpp (C++)”



b. This will open a snippets file called cpp.json



c. In the explorer pane, open the provided file “cpp.json”



d. Copy the content into the snippet file opened in step 3, then save the file.

```
cpp.json ×
C: > Users > developer > AppData > Roaming > Code > User >
1  {
2      "Default Main": {
3          "prefix": "&main",
4          "body": [
5              "#include <iostream>",
6              "using namespace std;",
7              "",
8              "int main()",
9              "{",
10             "    ",
11             "    return 0;",
12             "}"
13         ],
14         "description": ""
15     }
16 }
```

e. In any C++ file, you can now start typing “&main” and the snippet will be available.

```
test.cpp 2, U ×
test.cpp
1  &mai
   &main                    Default Main
```

f. Once you select the snippet, the following template code will be added to your file

```
test.cpp > main()
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6
7      return 0;
8  }
```

Congratulations! Your system is set up for C/C++ training.

Happy Coding!