DATA ANALYTICS REFERENCE DOCUMENT	
Document Title:	Setting up Cygwin
Document No.:	1570875139
Author(s):	Gerhard van der Linde, Rita Raher
Contributor(s):	

REVISION HISTORY

Revision	Details of Modification(s)	Reason for modification	Date	By
0	Draft release	Document the setup of cygwin	2019/10/12 10:12	Gerhard van der Linde

Cygwin debugger in VS Code

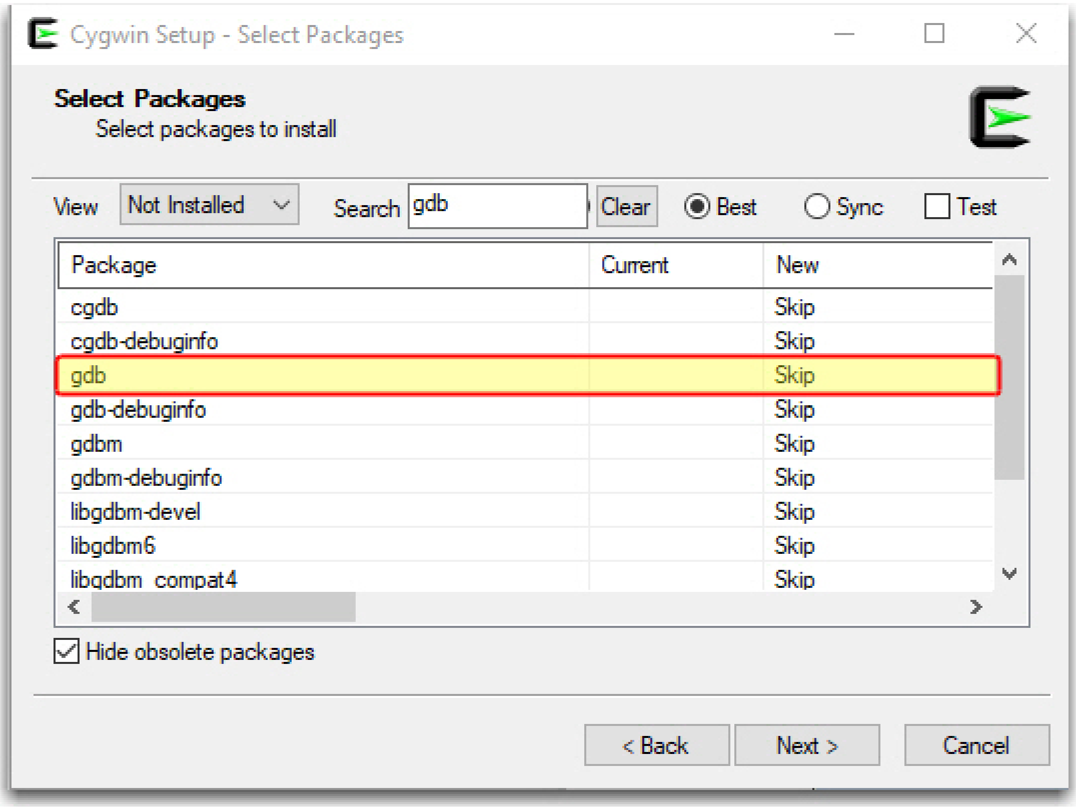
- Download and run the Cygwin installer from here https://cygwin.com/install.html
- Follow the installation instructions from here https://preshing.com/20141108/how-to-install-the-latest-gcc-on-windows/
- After installation building is required, **note** that this is done in a newly installed *Cygwin terminal*.
- The module in the course uses gcc instead of g++.

Adding the gcc debugger gdb

Install gdb

Rerun the cygwin installer(setup-x86_64.exe), located in downloads or moved to cywin folder in c:\cygwin64

Click through the installer and select gdb, you might have to use search to locate it.


Cygwin Setup - Select Packages

Select Packages
Select packages to install

View: Not Installed   Search: gdb   Clear   [✓ Best]   Sync   Test

Package	Current	New
cgdb		Skip
cgdb-debuginfo		Skip
gdb		Skip
gdb-debuginfo		Skip
gdbm		Skip
gdbm-debuginfo		Skip
libgdbm-devel		Skip
libgdbm6		Skip
libgdbm_compat4		Skip

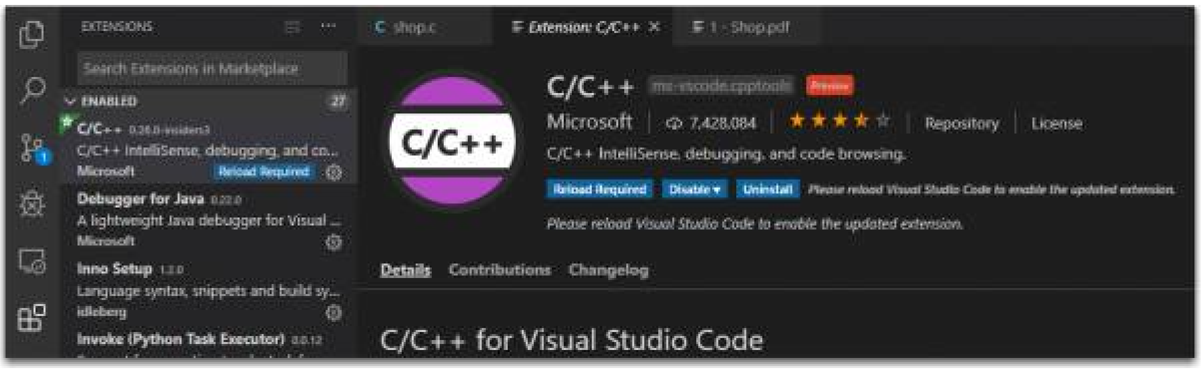
[✓] Hide obsolete packages

< Back   Next >   Cancel

Click skip to select gdb and click net to complete the installation.

The gdb.exe file will now appear in c:\cygwin64\bin folder.

No set up the debugger extension in VS Code if not already installed.



C/C++ 0.26.0-insiders3 — C/C++ IntelliSense, debugging, and co... — Microsoft — Reload Required

Debugger for Java 0.22.0 — A lightweight Java debugger for Visual ... — Microsoft

Inno Setup 1.2.0 — Language syntax, snippets and build sy... — idleberg

Invoke (Python Task Executor) 0.0.12

C/C++ ms-vscode.cpptools Preview

Microsoft | 7,428,084 | Repository | License

C/C++ IntelliSense, debugging, and code browsing.

Reload Required   Disable   Uninstall   Please reload Visual Studio Code to enable the updated extension.

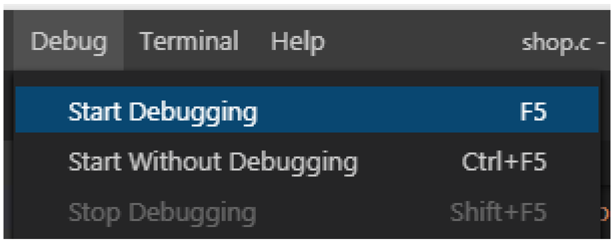
Please reload Visual Studio Code to enable the updated extension.

C/C++ for Visual Studio Code

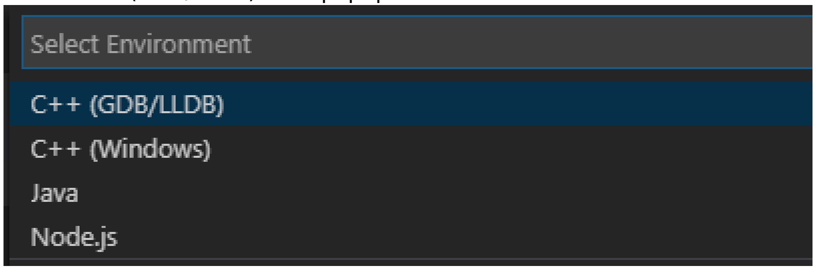
The debugger can now be configured in VS Code.

Step by step instructions to set up the debugger in VS Code

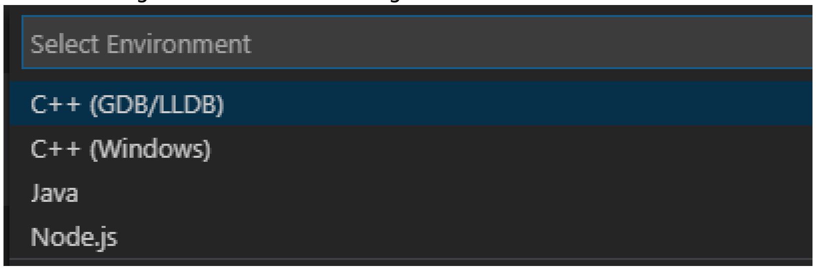
1. Open C file in VS Code
2. Set breakpoint in code
3. Press F5 or select Start debugging from Debug menu



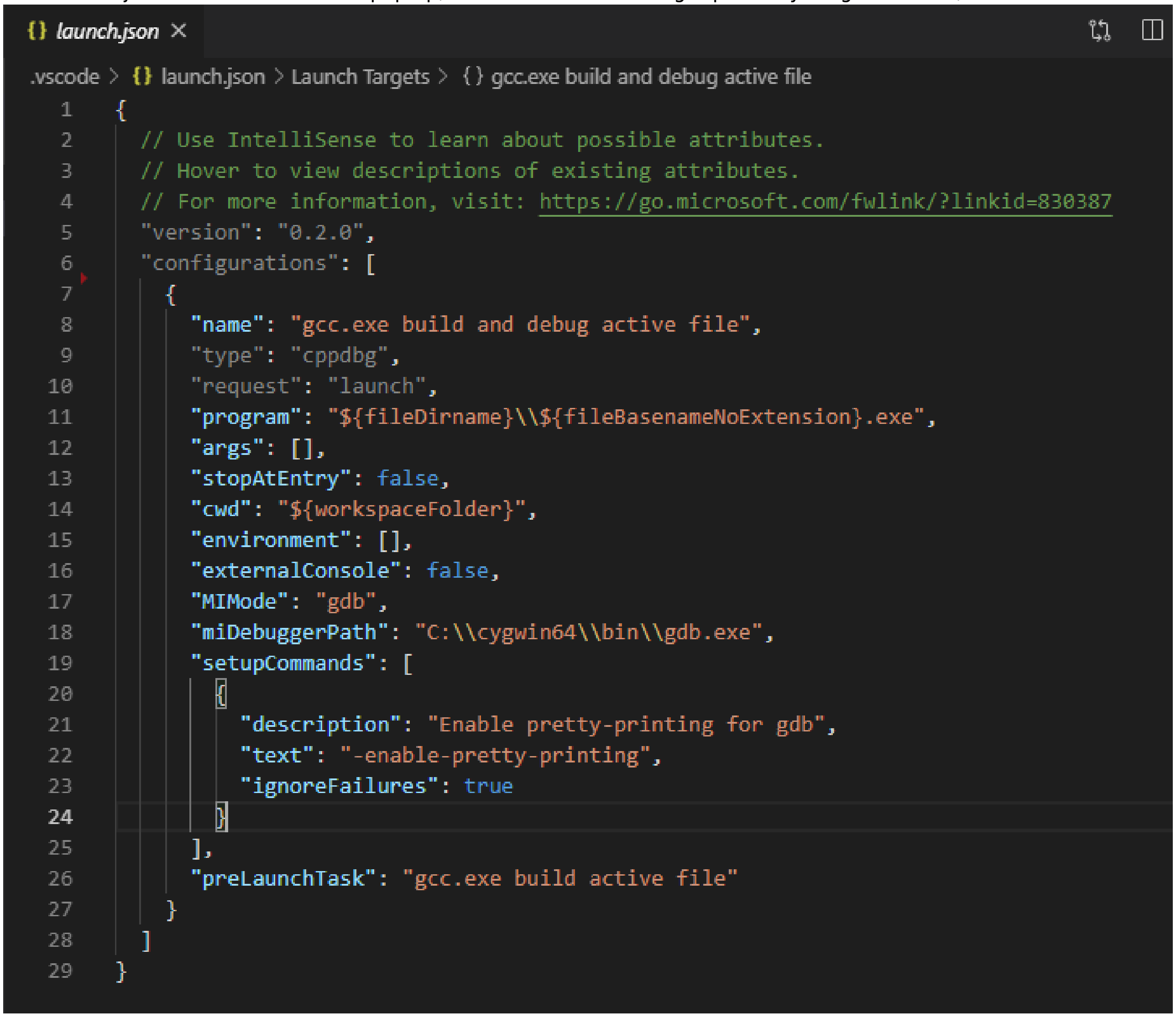
4. Select C++(GDB/LLDB) from popup



5. Then select gcc.exe build and debug active file

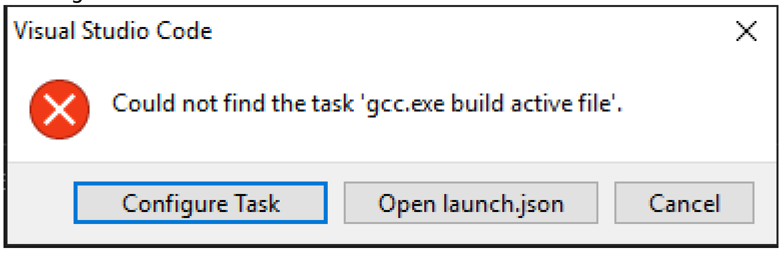


6. The Launch.json window should now pop up, see that this has the right path to your gdb.exe file, save and close.


launch.json
.vscode > launch.json > Launch Targets > { } gcc.exe build and debug active file

```json
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "gcc.exe build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "miDebuggerPath": "C:\\cygwin64\\bin\\gdb.exe",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "gcc.exe build active file"
    }
  ]
}
```

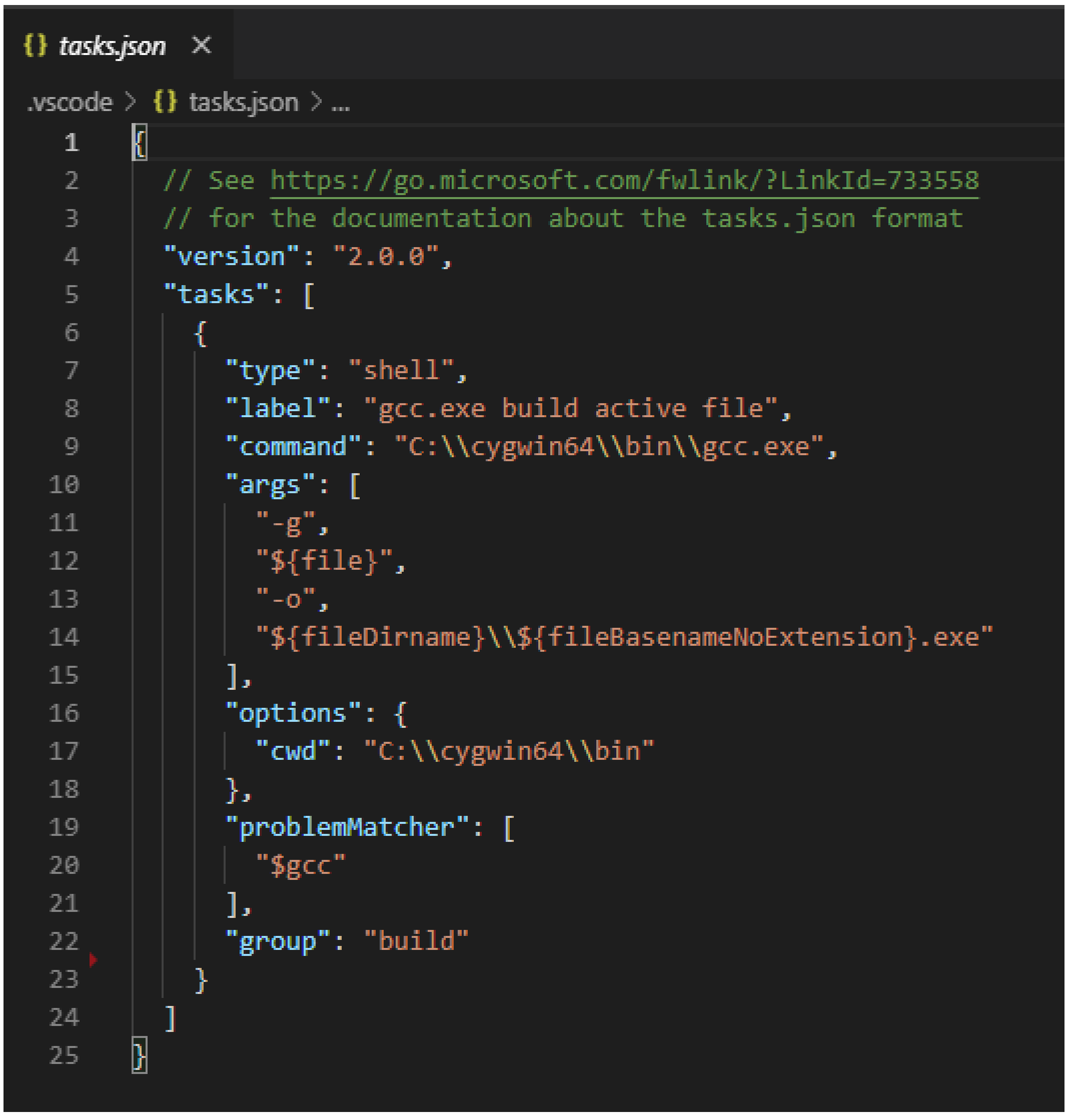
7. G back to the C file and F5 again, ow it should pop up a message about not being able to build the active file, click “Configure Task”.



Visual Studio Code

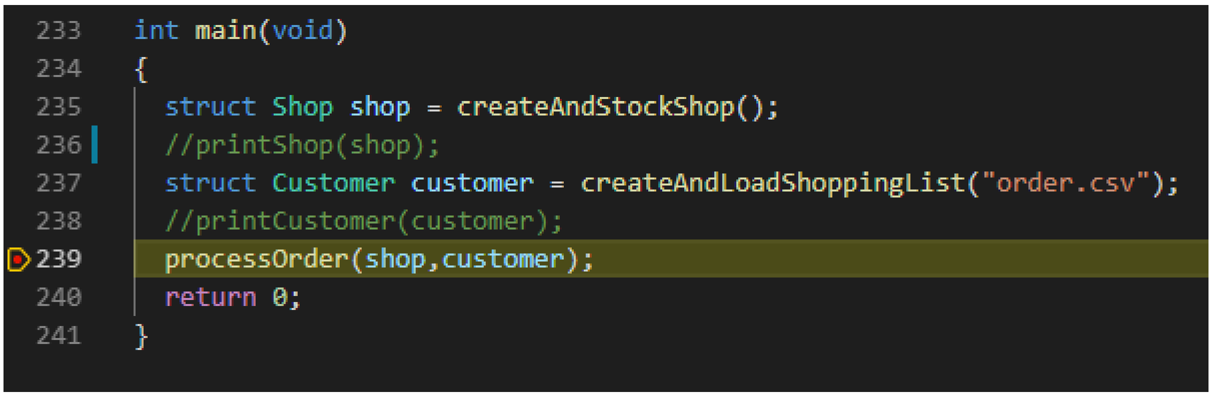
Could not find the task 'gcc.exe build active file'.

8. Select C/C++:gcc.exe build active file
9. Now the task,json file should pop up and confirm that “command” points to the right location where the gcc.exe file is located. Save and close



```
{
  // See https://go.microsoft.com/fwlink/?LinkId=733558
  // for the documentation about the tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "type": "shell",
      "label": "gcc.exe build active file",
      "command": "C:\\cygwin64\\bin\\gcc.exe",
      "args": [
        "-g",
        "${file}",
        "-o",
        "${fileDirname}\\${fileBasenameNoExtension}.exe"
      ],
      "options": {
        "cwd": "C:\\cygwin64\\bin"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": "build"
    }
  ]
}
```

10. Go back to the C file again and press F5.
11. Now everything should start up and the code should stop at the debugger breakpoint.



```c
int main(void)
{
  struct Shop shop = createAndStockShop();
  //printShop(shop);
  struct Customer customer = createAndLoadShoppingList("order.csv");
  //printCustomer(customer);
  processOrder(shop,customer);
  return 0;
}
```

Sample configuration files for Cygwin gdb to work in VS Code

launch.json

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [

    {
      "name": "gcc.exe build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "miDebuggerPath": "C:\\cygwin64\\bin\\gdb.exe",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "gcc.exe build active file"
    }
  ]
}
```

tasks.json

```
{
  // See https://go.microsoft.com/fwlink/?LinkId=733558
  // for the documentation about the tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "type": "shell",
      "label": "gcc.exe build active file",
      "command": "C:\\cygwin64\\bin\\gcc.exe",
      "args": [
        "-g",
        "${file}",
        "-o",
        "${fileDirname}\\${fileBasenameNoExtension}.exe"
      ],
      "options": {
        "cwd": "C:\\cygwin64\\bin"
      },
      "problemMatcher": [
        "$gcc"
```

```
            ],
            "group": "build"
          }
        ]
      }
```

Right click and save the two json configuration files above into a subfolder named `.vscode`. This folder should be in the same folder where your c files that you want to debug resides.

Troubleshooting

- https://github.com/microsoft/vscode-cpptools/issues/2778

```
"logging": { "engineLogging": true }
```

- http://cs.baylor.edu/~donahoo/tools/gdb/tutorial.html

UTF-8 Workaround

* https://github.com/microsoft/vscode-cpptools/issues/1527

So the issue is caused in gb and the extra escaped character returned on line three in the VS Code debugger window as shown below.

```
1: (2308) ->(gdb)
1: (2317) <-1001-gdb-set target-async on
1: (2318) ->&"\357\273\2771001-gdb-set target-async on\n"
1: (2319) ->&"Undefined command: \"\".  Try \"help\".\n"
1: (2320) ->^error,msg="Undefined command: \"\".  Try \"help\"."
1: (2320) ->(gdb)
```

As a workaround to fix this add a "gdb-with-chcp.cmd" within to your project (eg, "c:\cywin64\bin\gdp-with-chcp.cmd"):

```
@:: gdb-with-chcp.cmd
@chcp 1257 >NUL 2>&1 && @"gdb.exe" %*
```

This sets the codepage for GDB and fixes the problem cause above when running the PC set to UTF-8 codepage.

Amend the current json code launching the debugger to point to the cmd file just created.

launch.josn

```
        // use "PATHTO/gdb-with-chcp.cmd" as "miDebuggerPath" within "launch.json"
        // eg
        // ...
        "miDebuggerPath": "${workspaceFolder}/dbin/gdb-with-chcp.cmd",
        // ...
```

This resolved my gcc debugger issues.