CE103 Algorithms and Programming I

Development Environments and Algorithm Basics

Author: Asst. Prof. Dr. Uğur CORUH

Contents

List of Figures

List of Tables

0.1 CE103 Algorithms and Programming I

0.2 Week-2

0.2.0.1 Fall Semester, 2021-2022

Download DOC[1], SLIDE[2], PPTX[3]

[1] ce103-week-2-setup.tr.md_doc.pdf
[2] ce103-week-2-setup.tr.md_slide.pdf
[3] ce103-week-2-setup.tr.md_slide.pptx

0.3 Algorithm Basics

0.4 Flowgorithm

http://www.flowgorithm.org/

0.5 Pseudocode

Pseudocode - Wikipedia[4]

Pseudocode Examples[5]

How to write a Pseudo Code? - GeeksforGeeks[6]

0.6 Introduction to Analysis of Algorithms

In this course we will learn how to code with several development environments and next term we will see analysis of algorithms in details.

This topic is covered in the following link : CE100 Introduction to Analysis of Algorithms[7]

0.7 Programming Environment Setup and Configuration

Programming life is not about only learning how to code. Mostly you need to use several code development environments and you need to learn how to use them efficiently.

0.8 C / C++ Environment and Development

0.8.0.1 DevCpp (Install / Compile / Run / Debug) (1)

Download DevC++ IDE from following link

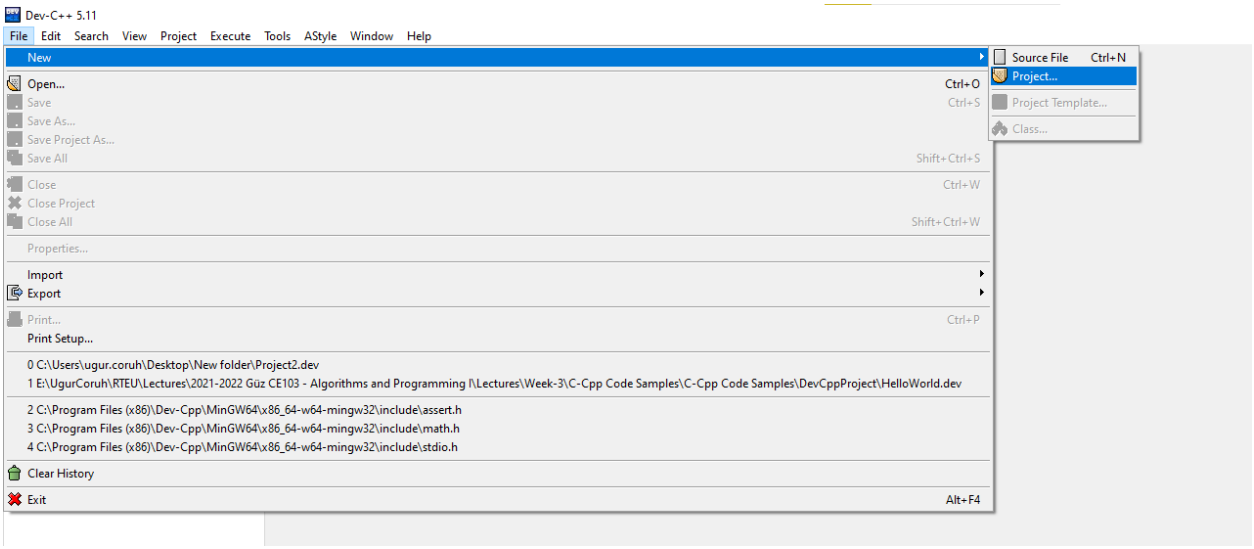
https://www.bloodshed.net/

0.8.0.2 DevCpp (Install / Compile / Run / Debug) (2)

Open DevC++ IDE for C Project Generation

Open File->New->Project

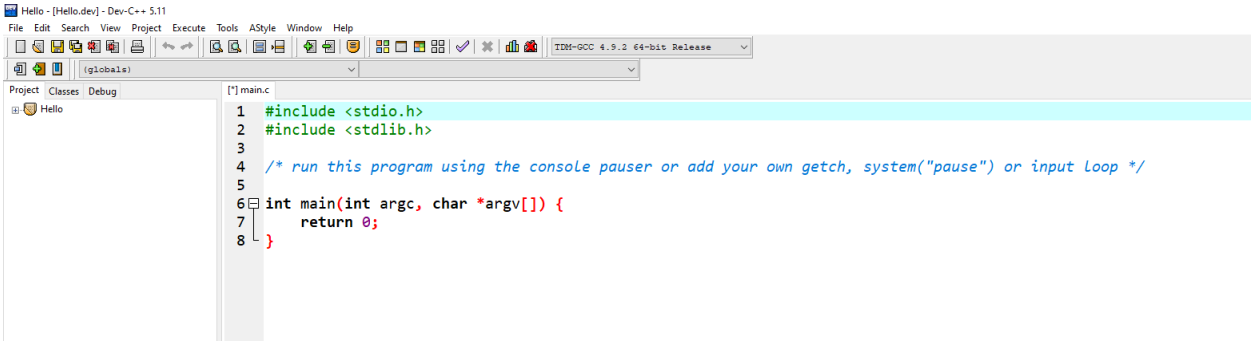
[4] https://en.wikipedia.org/wiki/Pseudocode
[5] https://www.unf.edu/~broggio/cop2221/2221pseu.htm
[6] https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/
[7] https://ucoruh.github.io/ce100-algorithms-and-programming-II/week-1/ce100-week-1-intro/



0.8.0.3 DevCpp (Install / Compile / Run / Debug) (3) Select **Console Application** from **Basic** tab and with **C Project** Option and write a project name such as "**Hello**" then press OK

Select a folder and save **Hello.dev** project file.

0.8.0.4 DevCpp (Install / Compile / Run / Debug) (4) You will see a sample main with empty body



0.8.0.5 DevCpp (Install / Compile / Run / Debug) (5)

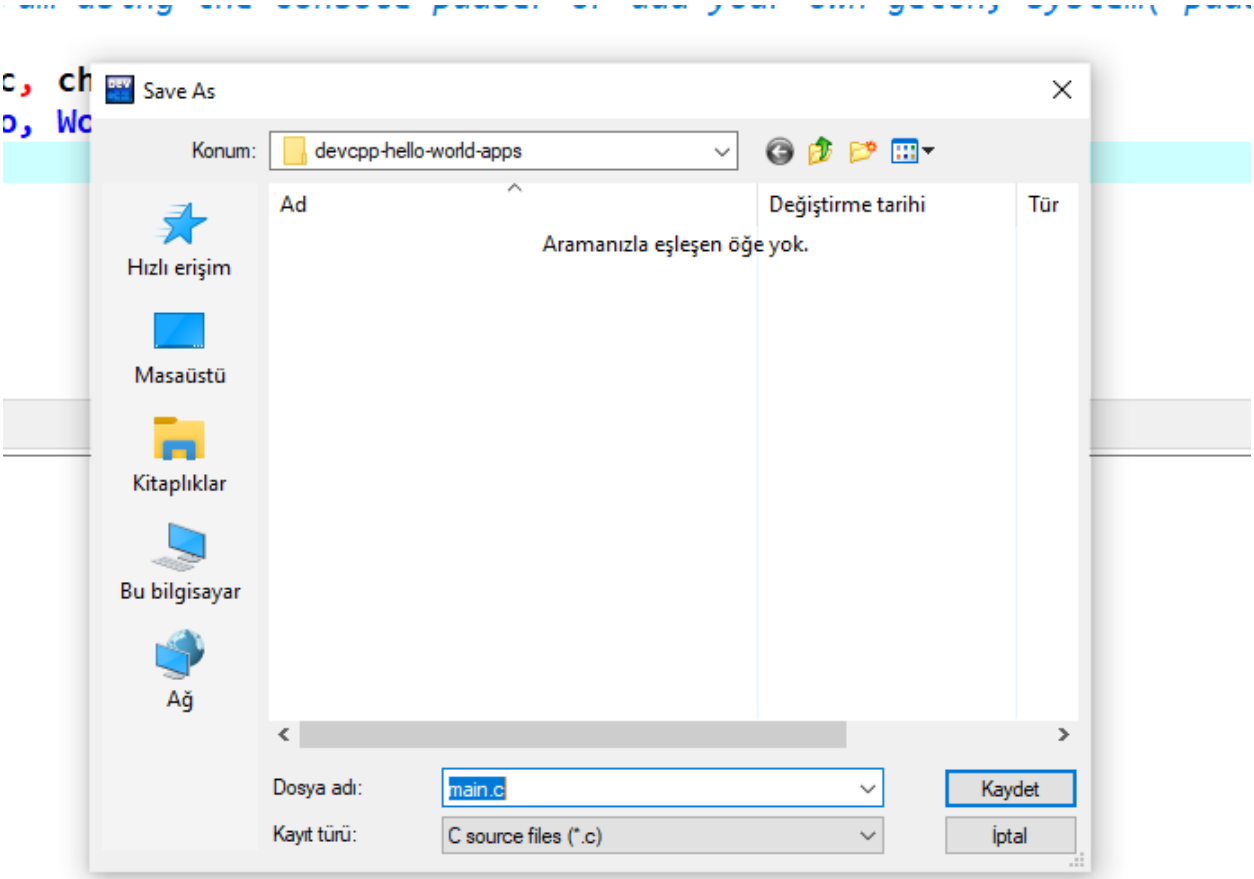
```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your own getch, s,ystem("pause") or input loop */
int main(int argc, char *argv[]) {
    retAdd 0;
}
```

0.8.0.6 DevCpp (Install / Compile / Run / Debug) (6) add the following line in main function. This will write "Hello, World!" on the screen and then wait a keypress to exit from application

```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main(int argc, char *argv[]) {
    printf("Hello, World!");
    getchar();
    return 0;
}
```

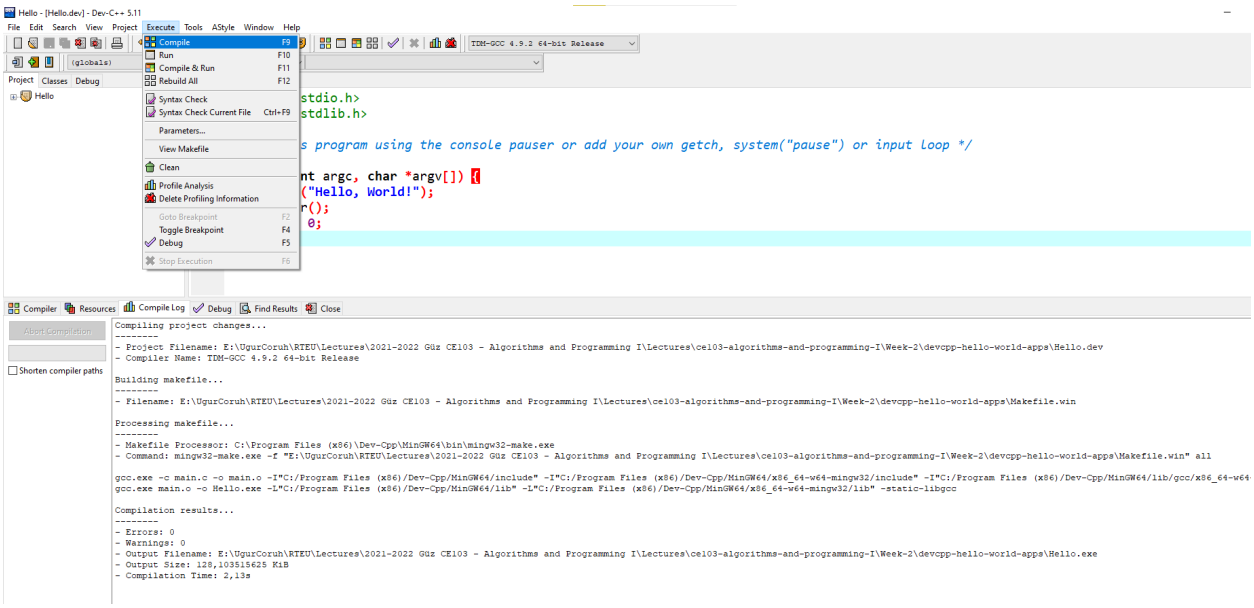
0.8.0.7 DevCpp (Install / Compile / Run / Debug) (7)

Then save the file



0.8.0.8 DevCpp (Install / Compile / Run / Debug) (8)

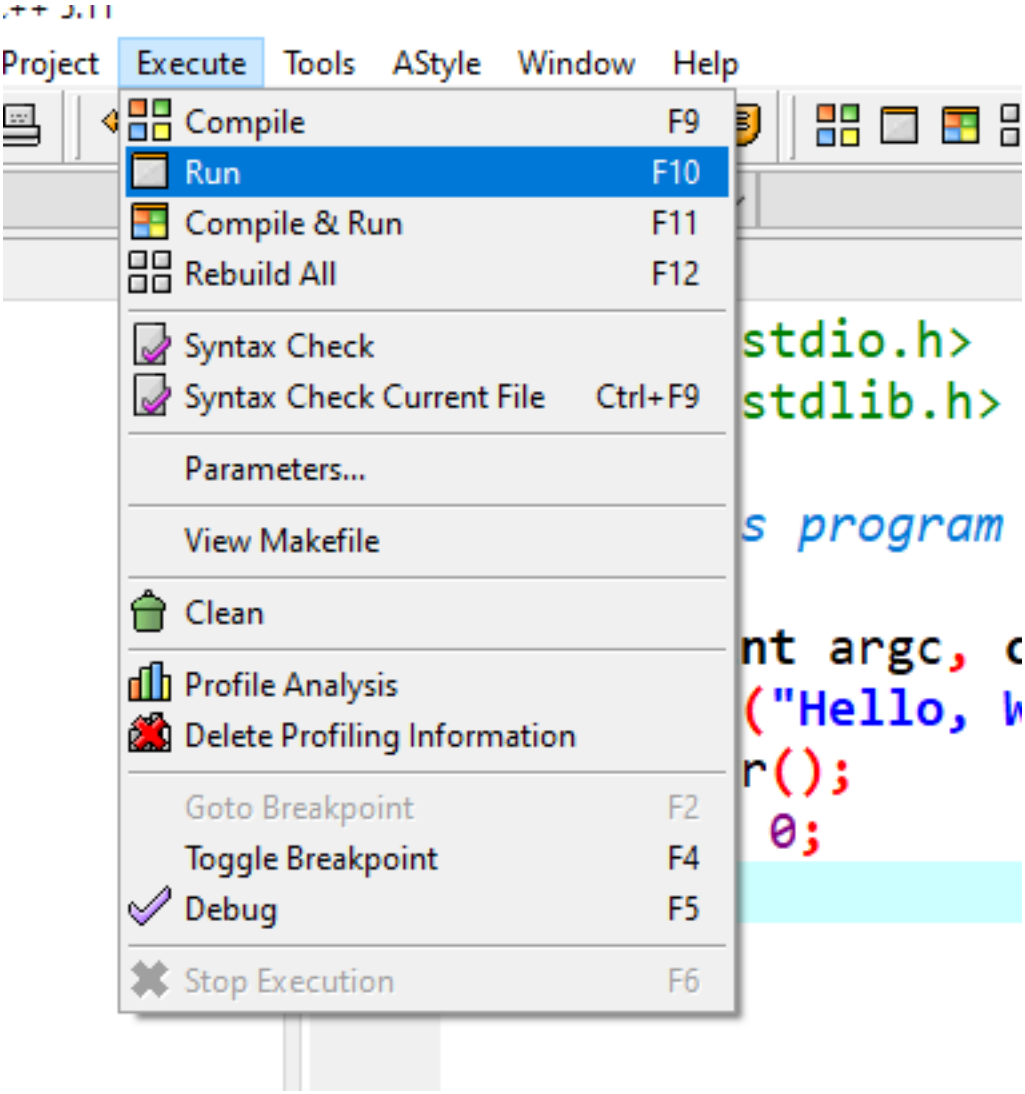
Use from menu *Execute->Compile F5* to generate Hello.exe



0.8.0.9 DevCpp (Install / Compile / Run / Debug) (9) You can find the generated Hello.exe path from Compile.log as follow. Check the Output Filename

```
Compiling project changes...
--------
- Project Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectu
- Compiler Name: TDM-GCC 4.9.2 64-bit Release

Building makefile...
--------
- Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce10

Processing makefile...
--------
- Makefile Processor: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\mingw32-make.exe
- Command: mingw32-make.exe -f "E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programm

gcc.exe -c main.c -o main.o -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/include" -I"C:/Program Files (x86

gcc.exe main.o -o Hello.exe -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib" -L"C:/Program Files (x86)/De


Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectur
- Output Size: 128,103515625 KiB
- Compilation Time: 2,13s
```

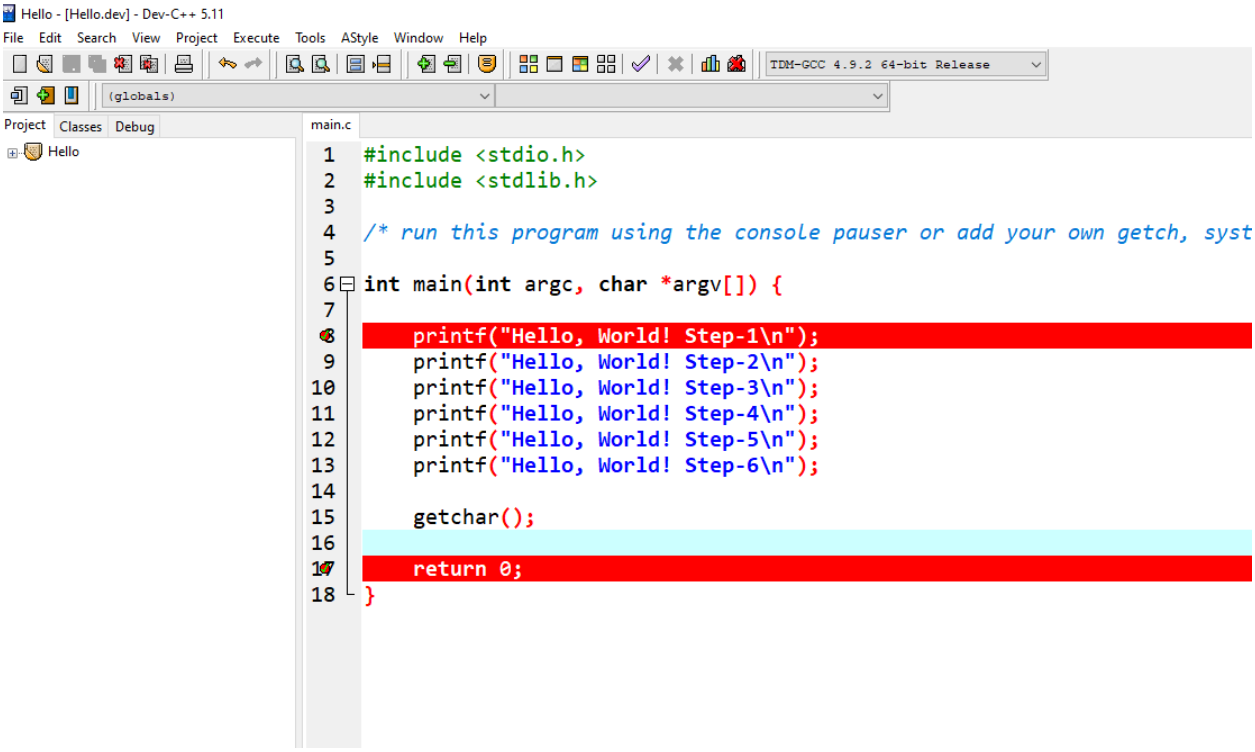
0.8.0.10 DevCpp (Install / Compile / Run / Debug) (10) Then you can run with Execute->Run F10 or Directly Compile&Run F11



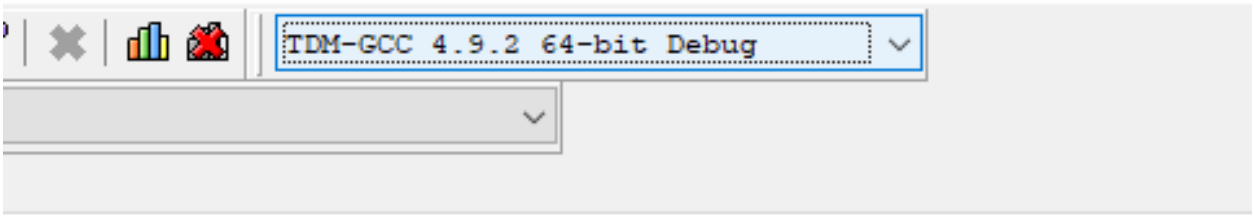
0.8.0.11 DevCpp (Install / Compile / Run / Debug) (11) for debugging operations, just change the code and add more statements as follow

```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your getch, system(",pause") or input loop */

int main(int argc, char *argv[]) {

    printf("Hello, World! Step-1\n");
    printf("Hello, World! Step-2\n");
    printf("Hello, World! Step-3\n");
    printf("Hello, World! Step-4\n");
    printf("Hello, World! Step-5\n");
    printf("Hello, World! Step-6\n");

    getchar();

    return 0;
}
```

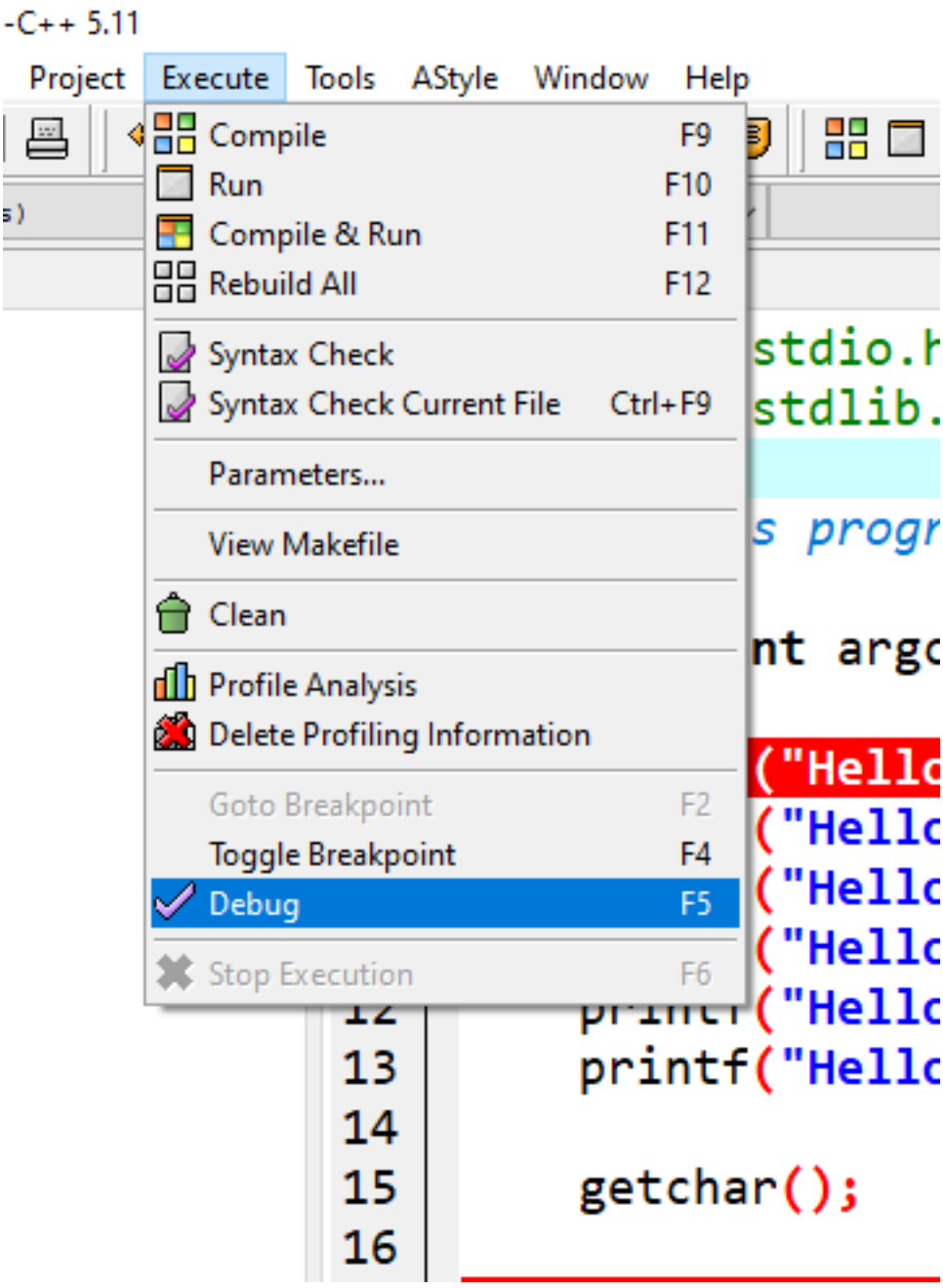
0.8.0.12 DevCpp (Install / Compile / Run / Debug) (12) Click on line numbers and add breakpoints for debugger. This red point will be debugger stop points



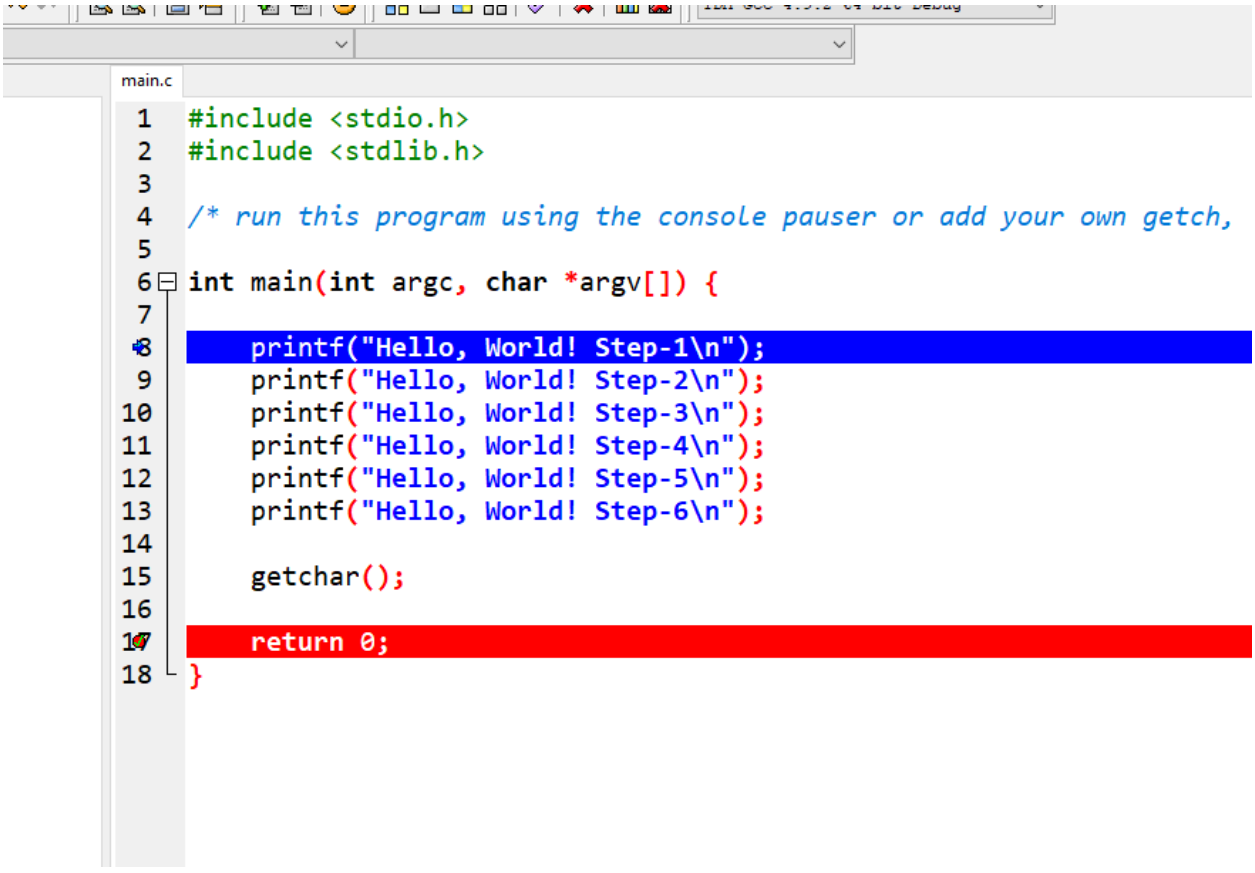
0.8.0.13 DevCpp (Install / Compile / Run / Debug) (13) In the ,menu section select compiler with debug option



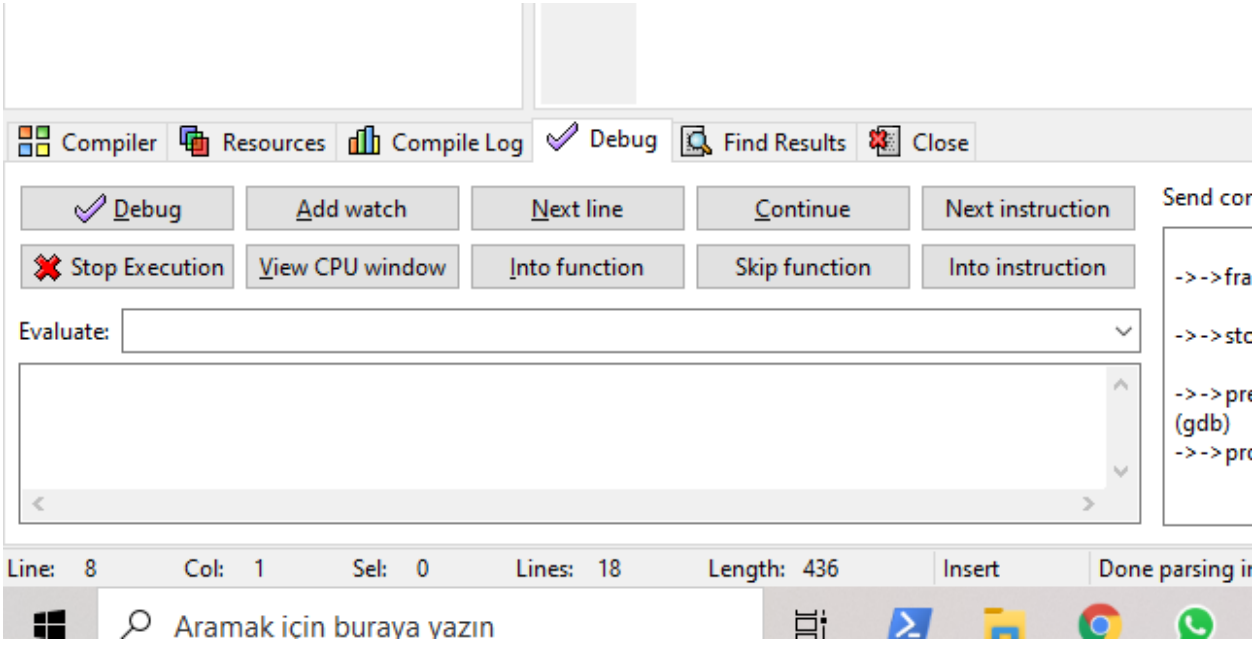
0.8.0.14 DevCpp (Install / Compile / Run / Debug) (14) Compile application with debug setting and in Execute Section use Debug F5 to start debugging



0.8.0.15 DevCpp (Install / Compile / Run / Debug) (15) Debugger will stop at breakpoint at the debug point (blue line)



0.8.0.16 DevCpp (Install / Compile / Run / Debug) (16) Moving to next statement can be done via control buttons or shortcuts



0.8.0.17 DevCpp (Install / Compile / Run / Debug) (17) Press F8 to step-by-step continue

Then go to Project Options - Compiler - Linker and set Generate debugging information to "yes", and make sure you are not using any optimization options (they're not good for debug mode). Also check the Parameters tab, make sure you don't have any optimization options (like -O2 or -O3, but -O0 is ok because it means no optimization) or strip option (-s).

0.8.0.18 DevCpp (Install / Compile / Run / Debug) (18) After that, do a full rebuild (Ctrl-F11), then set breakpoint(s) where you want the debugger to stop (otherwise it will just run the program). To set a breakpoint on a line, just click on the gutter (the gray band on the left), or press Ctrl-F5.

0.8.0.19 DevCpp (Install / Compile / Run / Debug) (19) Now you are ready to launch the debugger, by pressing F8 or clicking the debug button. If everything goes well, the program will start, and then stop at the first breakpoint. Then you can step through the code, entering function calls, by pressing Shift-F7 or the "step into" button, or stepping over the function calls, by pressing F7 or the "next step" button. You can press Ctrl-F7 or the "continue" button to continue execution till the next breakpoint. At any time, you can add or remove breakpoints.

0.8.0.20 DevCpp (Install / Compile / Run / Debug) (20) When the program stopped at a breakpoint and you are stepping through the code, you can display the values of various variables in your program by putting your mouse over them, or you can display variables and expressions by pressing F4 or the "add watch" button and typing the expression.

0.8.0.21 DevCpp (Install / Compile / Run / Debug) (21) How do I debug using Dev-C++[8]

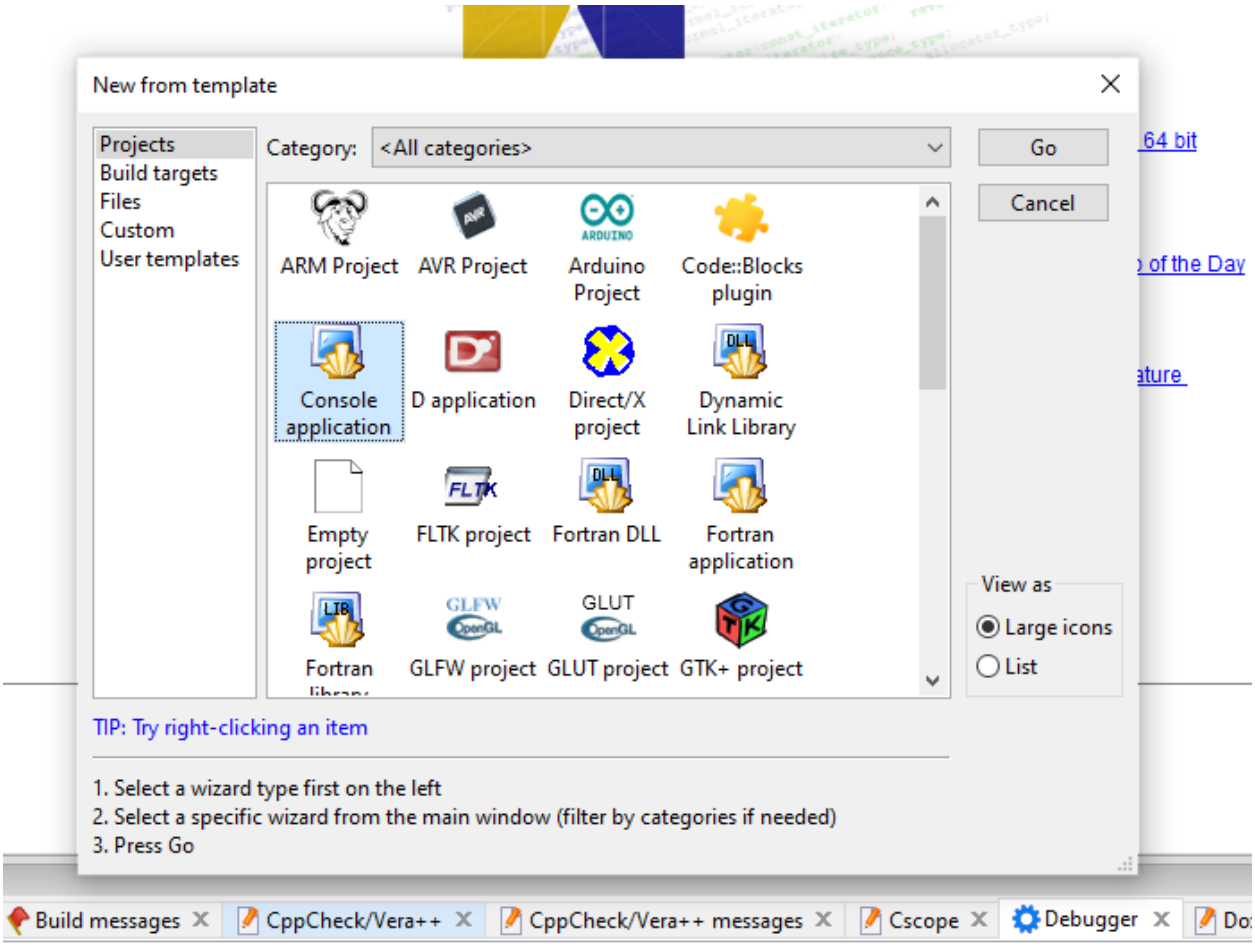
0.8.0.22 Code Blocks (Install / Compile / Run / Debug) (1) Download Code Blocks from the following link

Binary releases - Code::Blocks[9]

0.8.0.23 Code Blocks (Install / Compile / Run / Debug) (2) Open Code Blocks and

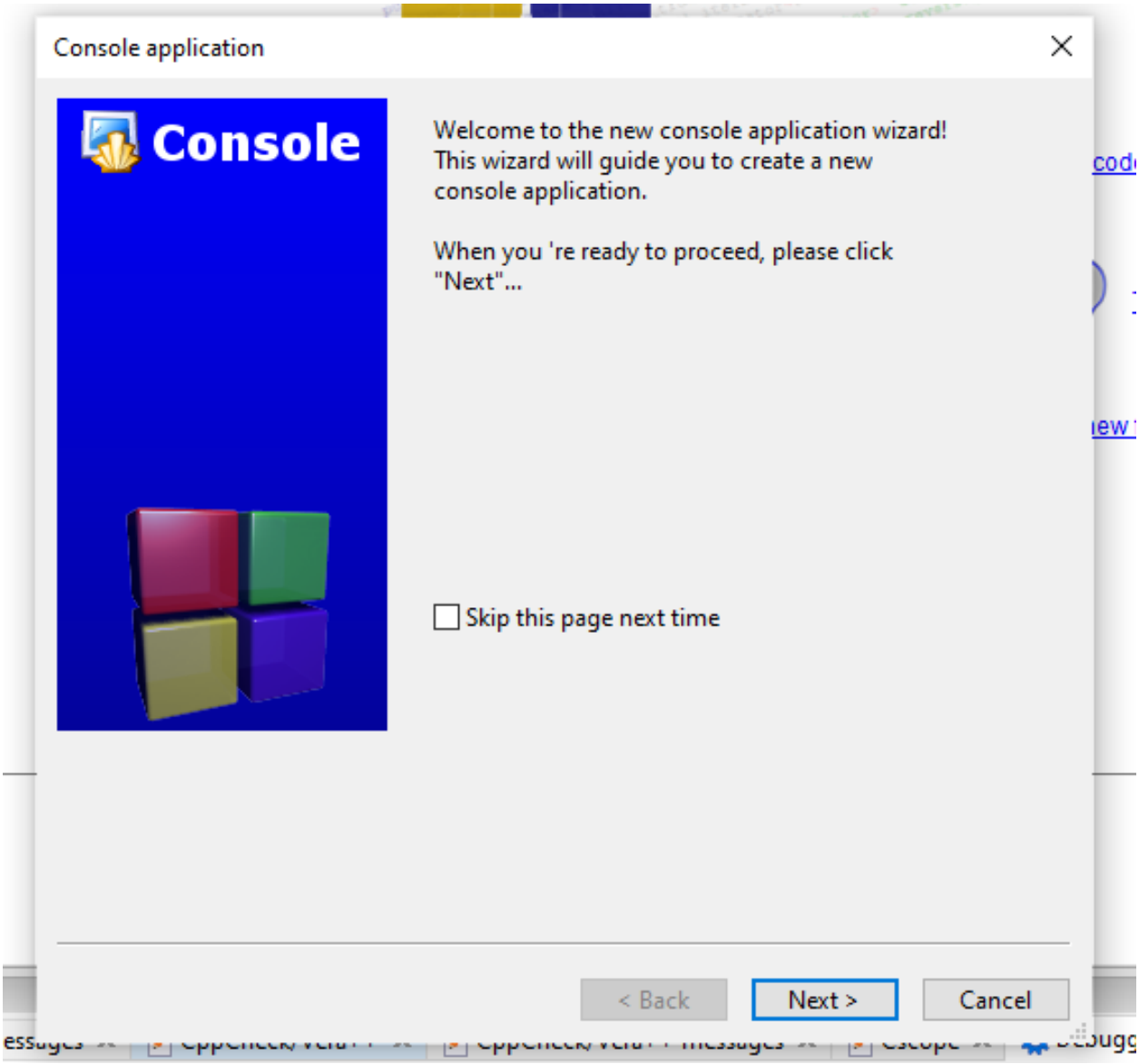
Select File->New->Project

[8] http://eilat.sci.brooklyn.cuny.edu/cis1_5/HowToDebug.htm
[9] https://www.codeblocks.org/downloads/binaries/

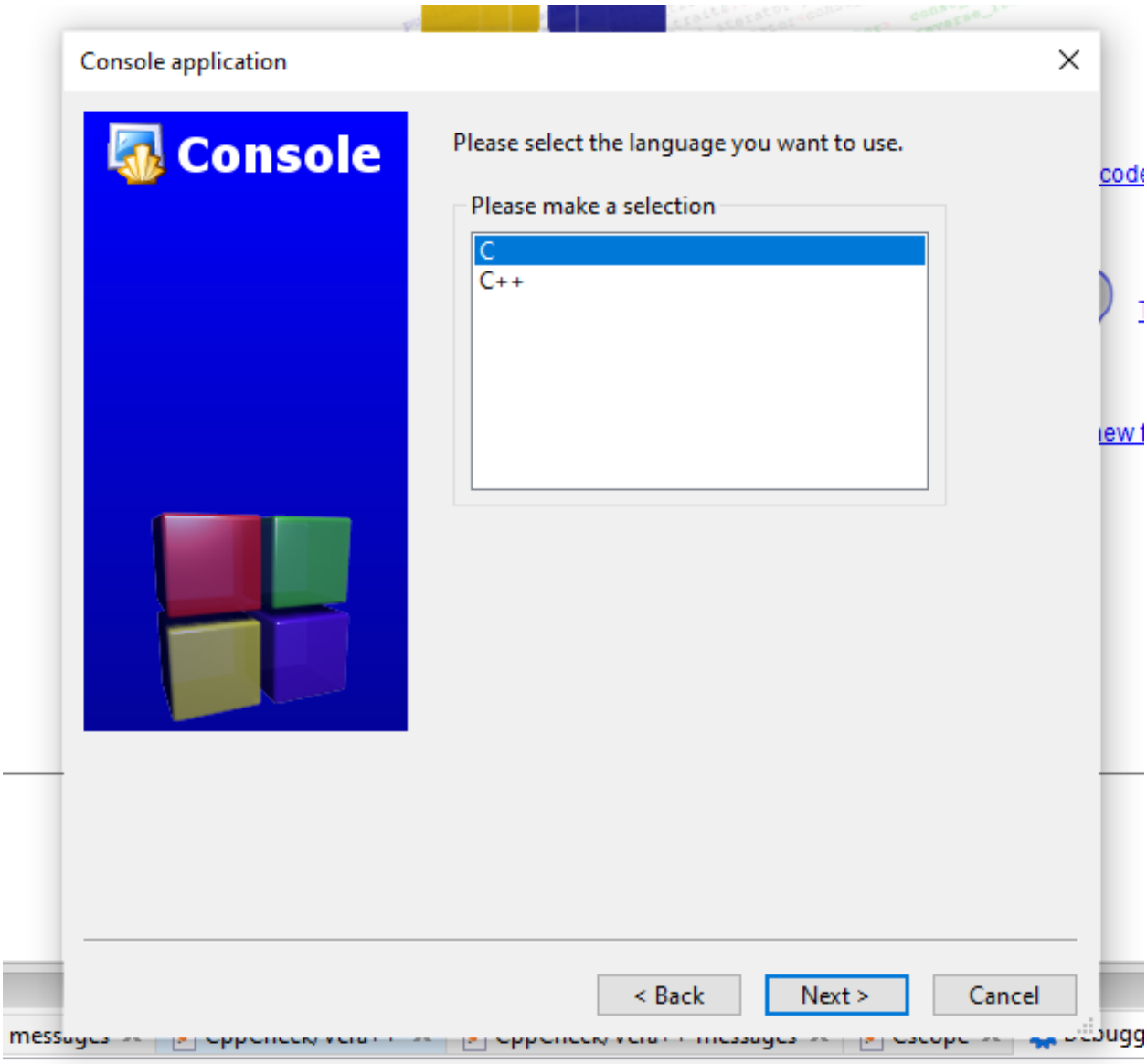


0.8.0.24 Code Blocks (Install / Compile / Run / Debug) (3) Select Console Application

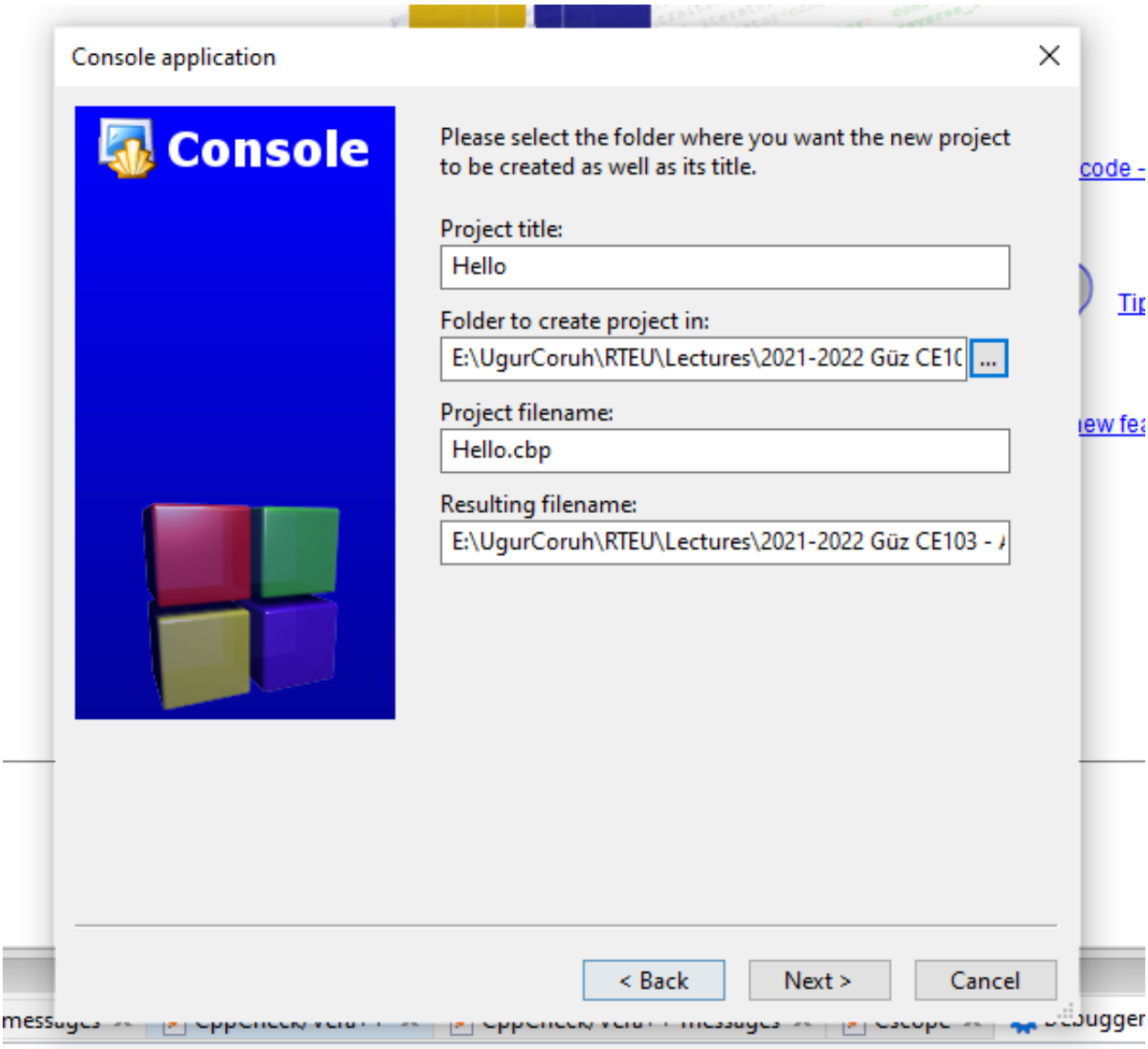
Click Next from Opening Window



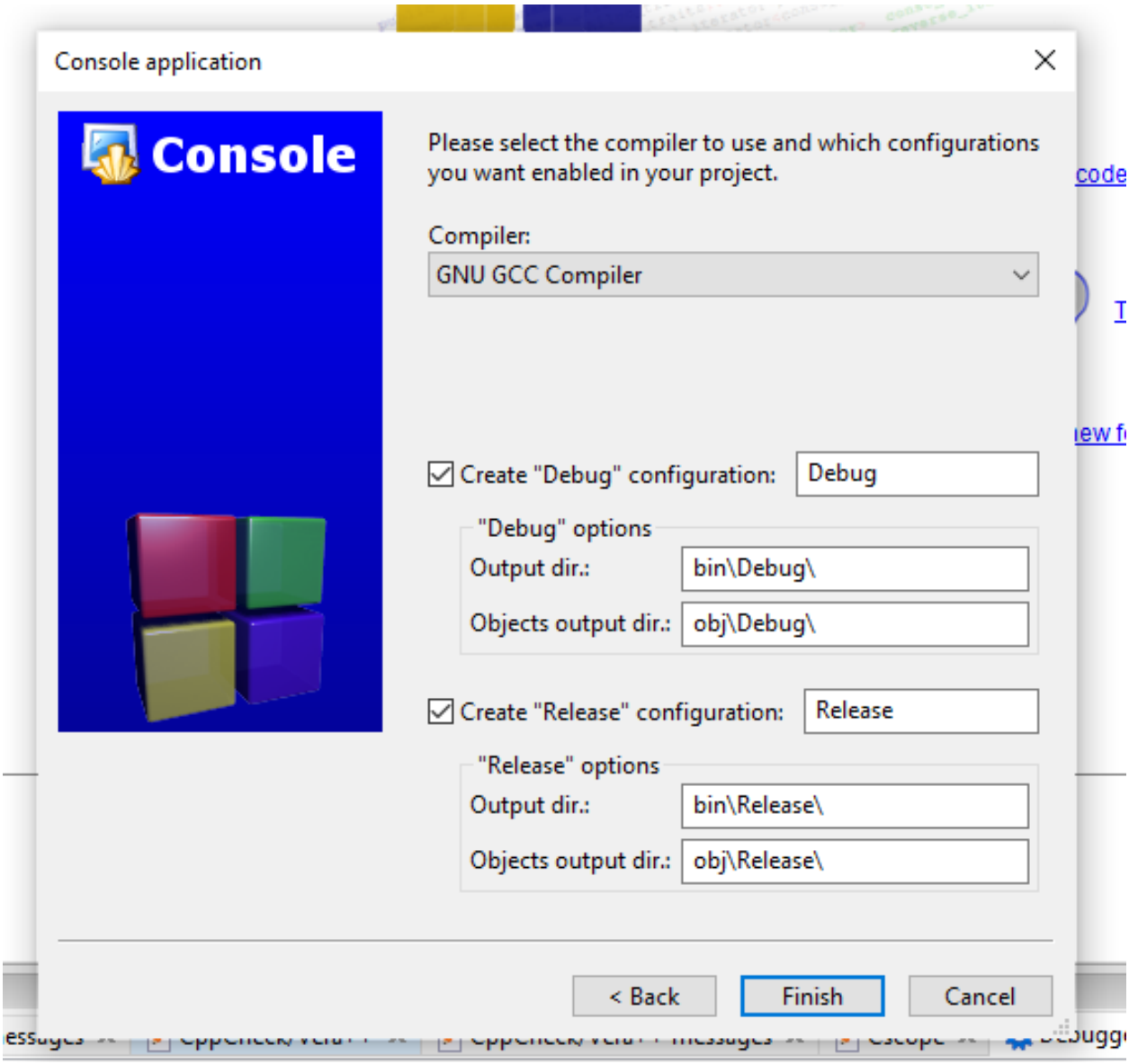
0.8.0.25 Code Blocks (Install / Compile / Run / Debug) (4) Select C for Sample Project



0.8.0.26 Code Blocks (Install / Compile / Run / Debug) (5) Write a project name and title also set a project folder



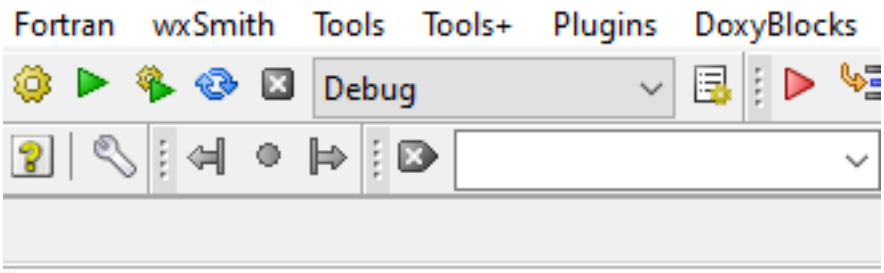
0.8.0.27 Code Blocks (Install / Compile / Run / Debug) (6) Select complier for this project we selected GCC but you can select C compliers from list. Set Debug and Release executable output folders.



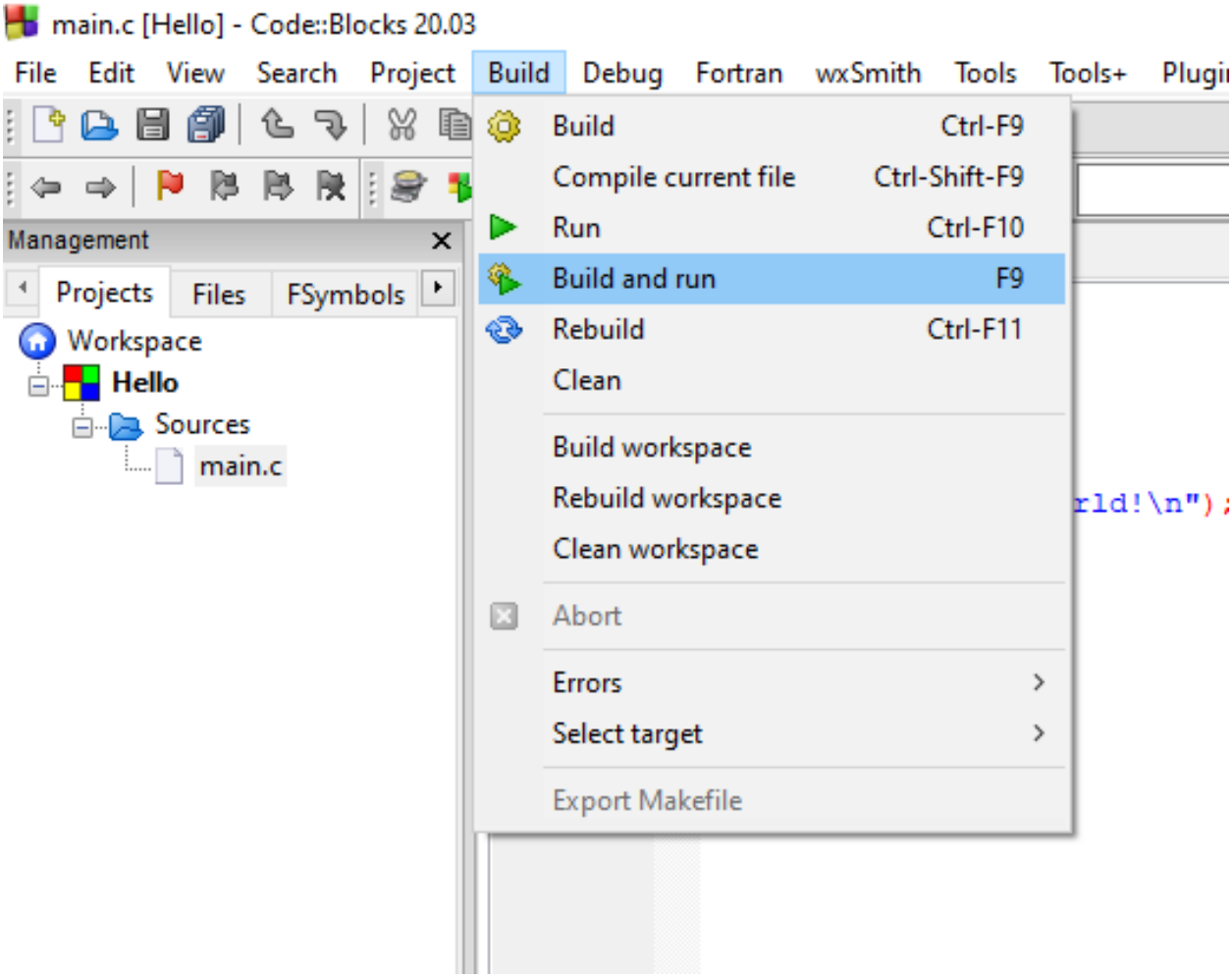
0.8.0.28 Code Blocks (Install / Compile / Run / Debug) (7) After this wizard you will have the following code

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

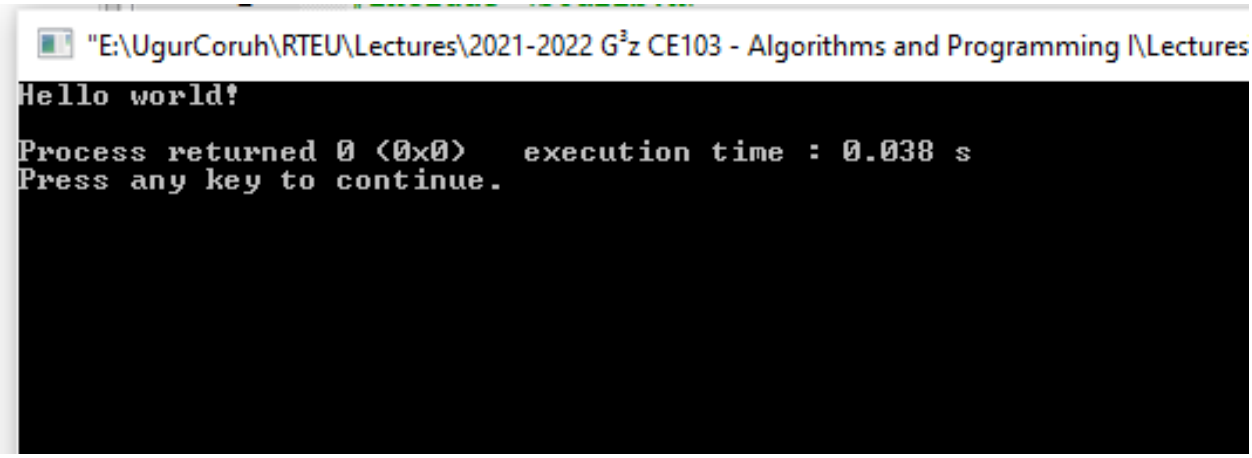
0.8.0.29 Code Blocks (Install / Compile / Run / Debug) (8) Select Debug Build from menu



0.8.0.30 Code Blocks (Install / Compile / Run / Debug) (9) Run with Build and Run F9



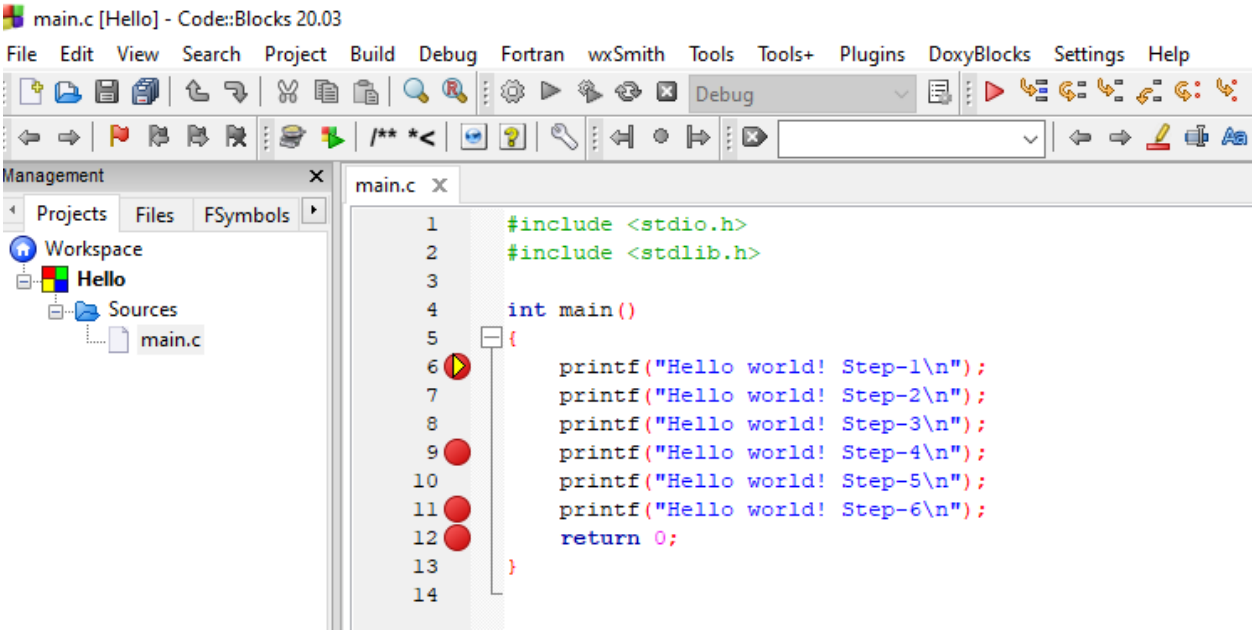
0.8.0.31 Code Blocks (Install / Compile / Run / Debug) (10) You should see the following output



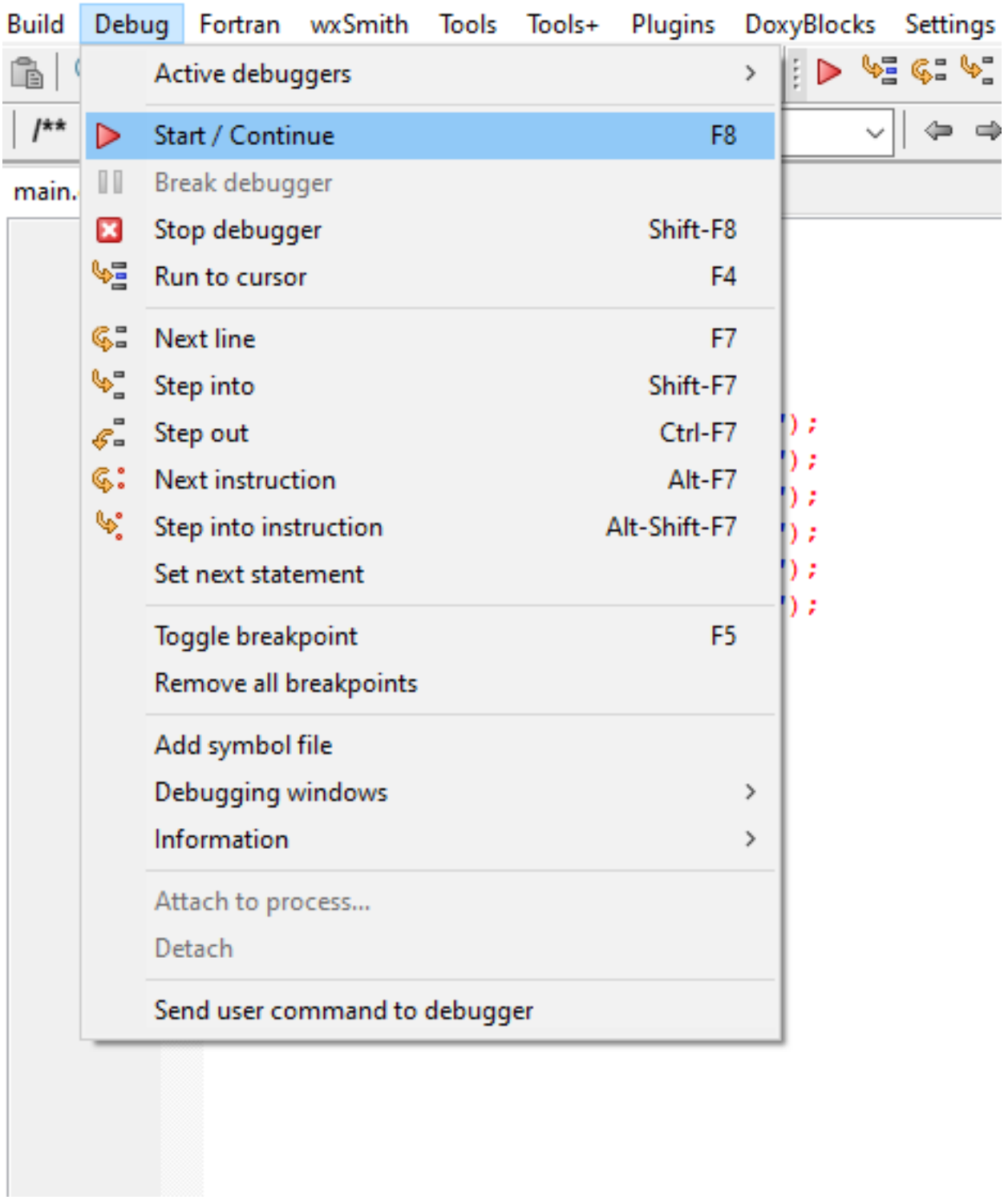
0.8.0.32 Code Blocks (Install / Compile / Run / Debug) (11) Add the following lines to your source code for debugging

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world! Step-1\n");
    printf("Hello world! Step-2\n");
    printf("Hello world! Step-3\n");
    printf("Hello world! Step-4\n");
    printf("Hello world! Step-5\n");
    printf("Hello world! Step-6\n");
    return 0;
}
```

0.8.0.33 Code Blocks (Install / Compile / Run / Debug) (12) and add break points with F5 or mouse click



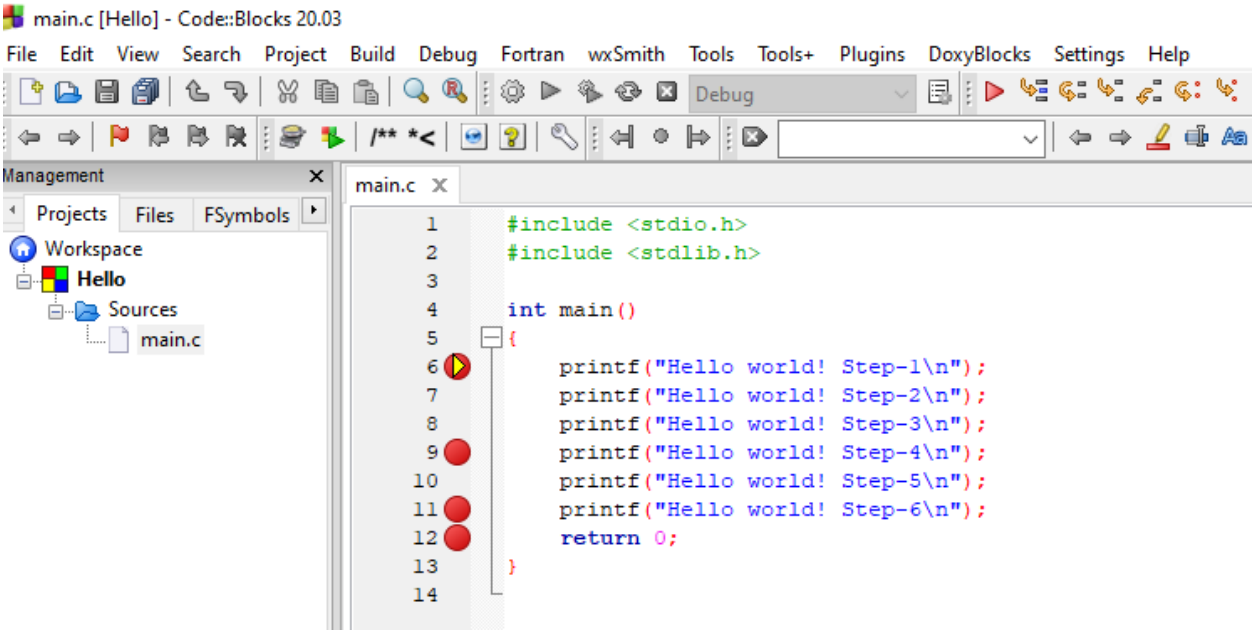
0.8.0.34 Code Blocks (Install / Compile / Run / Debug) (13) select Debug->Start/Continue to start debugger



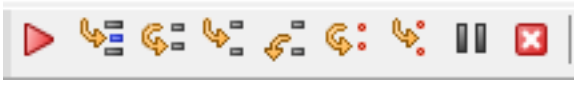
0.8.0.35 Code Blocks (Install / Compile / Run / Debug) (14) If you see the following error this is related with long or turkish character including path. Just move project to a shorter path and try again

```
Setting breakpoints
Debugger name and version: GNU gdb (GDB) 8.1
Starting the debuggee failed: No executable specified, use `target exec'.
Debugger finished with status 0
```

0.8.0.36 Code Blocks (Install / Compile / Run / Debug) (15) You will see the following yellow pointer for debugger



0.8.0.37 Code Blocks (Install / Compile / Run / Debug) (16) You can use the following menu or shortcuts for step-by-step debugging.



0.8.0.38 GCC/G++ Complier (MinGW) / Clang-cl (LLVM) (1) Download and install MinGW or LLVM compiler (if you downloded then skip this step)

MinGW installer (clang)

Download MinGW-w64 - for 32 and 64 bit Windows from SourceForge.net[10]

LLVM installer (gcc / g++)

Download LLVM releases[11]

Also use the following notes

https://llvm.org/devmtg/2014-04/PDFs/Talks/clang-cl.pdf

[10] https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download

[11] https://releases.llvm.org/

0.8.0.39 GCC/G++ Complier (MinGW) / Clang-cl (LLVM) (2) Open a console with "cmd" and test the following commands if commands are not recognized then set the system environment variable add gcc and g++ exe paths to path variable (add to both system and user path variable)

```
gcc --version
```

```
g++ --version
```

```
C:\Users\ugur.coruh>gcc --version
gcc (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 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.
```

```
clang --version
```

0.8.0.40 GCC/G++ Complier (MinGW) / Clang-cl (LLVM) (3) for gcc.exe, g++.exe and gdb.exe

```
C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw64\bin
```

for clang.exe , lldb.exe

```
C:\Program Files\LLVM\bin
```

This folder paths changes according to your setup

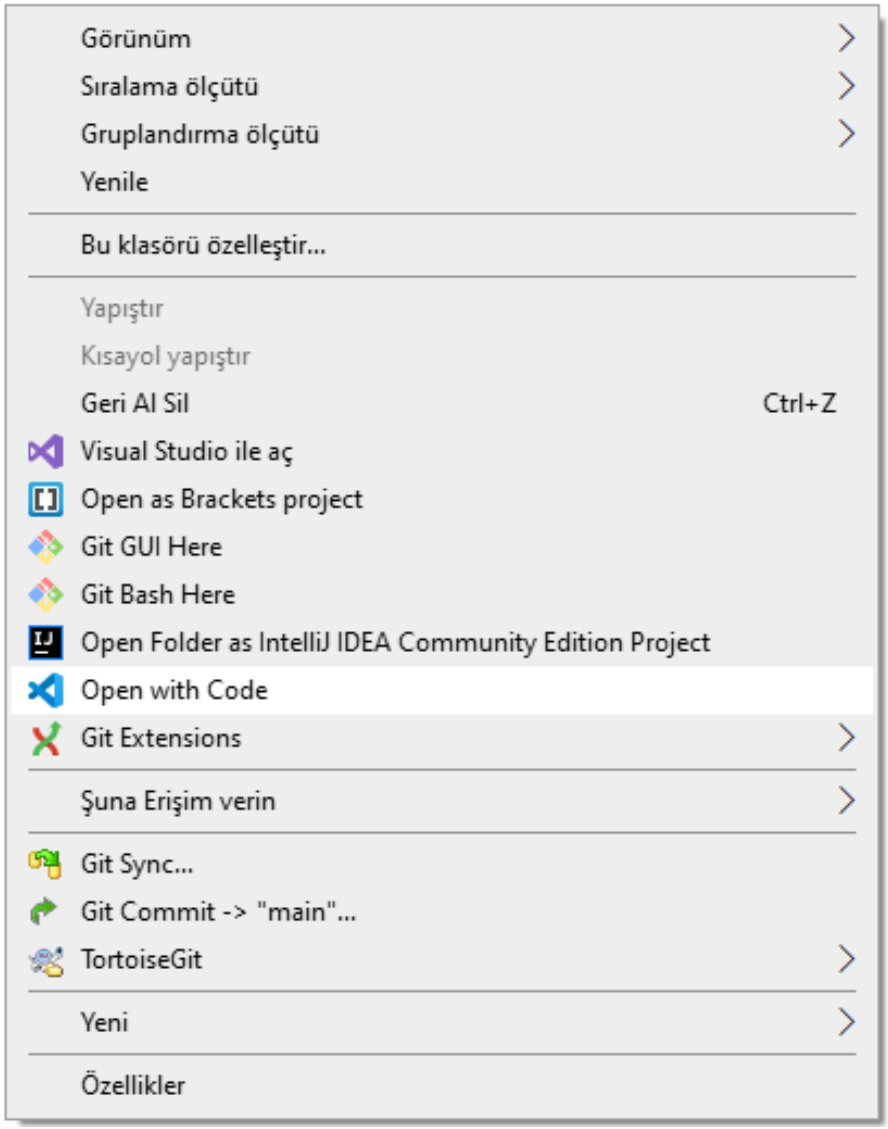
0.8.0.41 VSCode (Install / Compile / Run / Debug) (1) Download Visual Studio Code from the following link

Download Visual Studio Code - Mac, Linux, Windows[12]

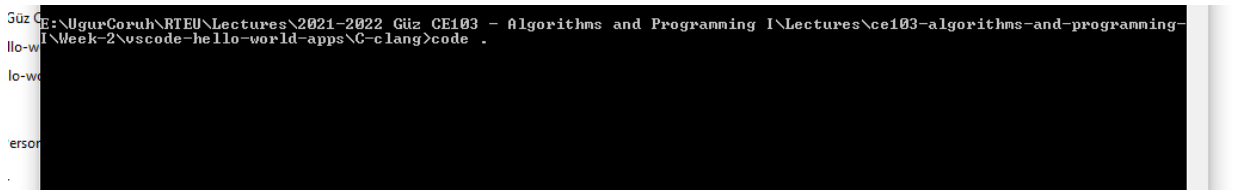
0.8.0.42 VSCode (Install / Compile / Run / Debug) (2) In this sample you will find MinGW and LLVM compiler combinations for C and C++

Create a folder and enter this folder then open this folder with vscode by right click

[12] https://code.visualstudio.com/download



0.8.0.43 VSCode (Install / Compile / Run / Debug) (3) or enter the folder via console

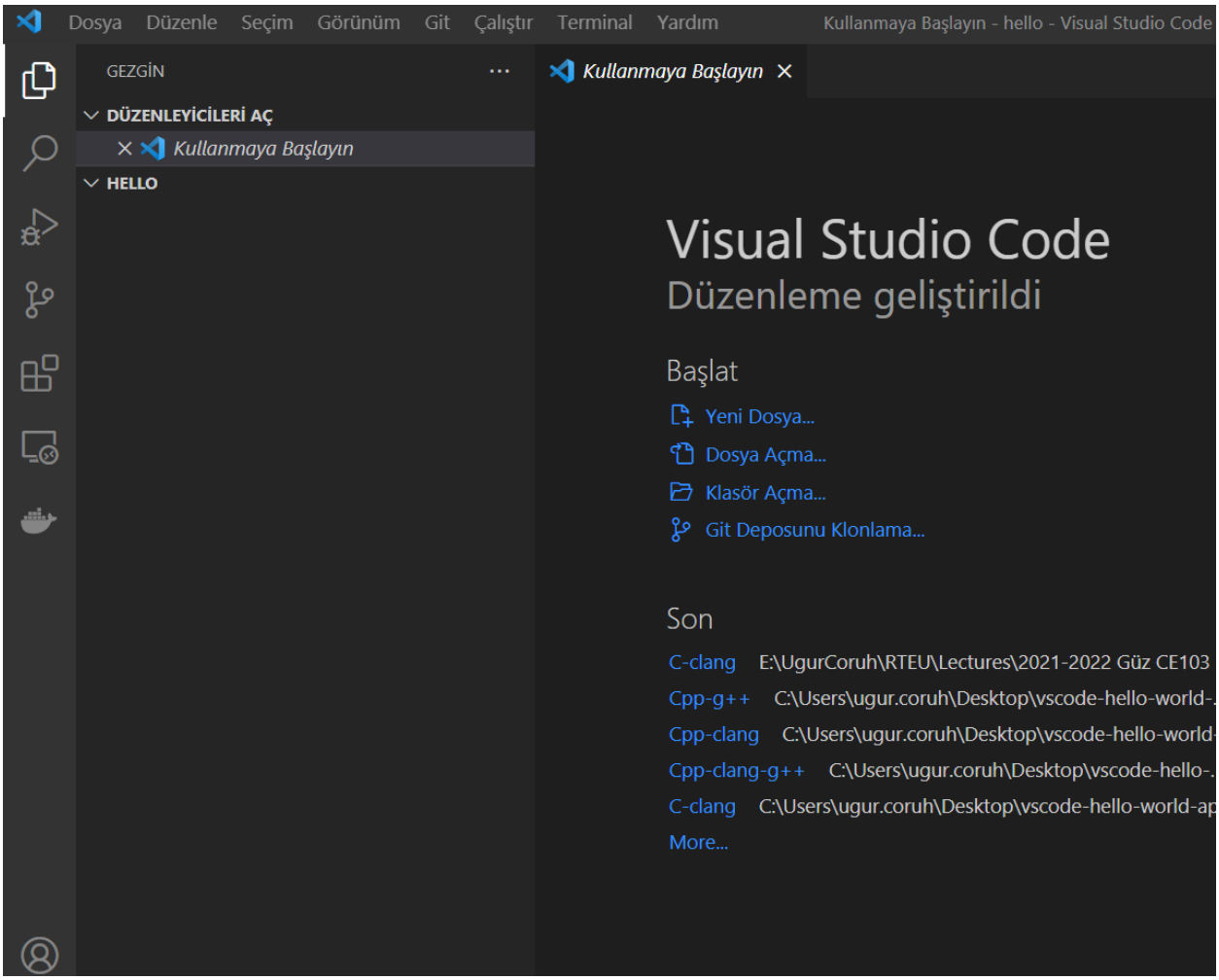


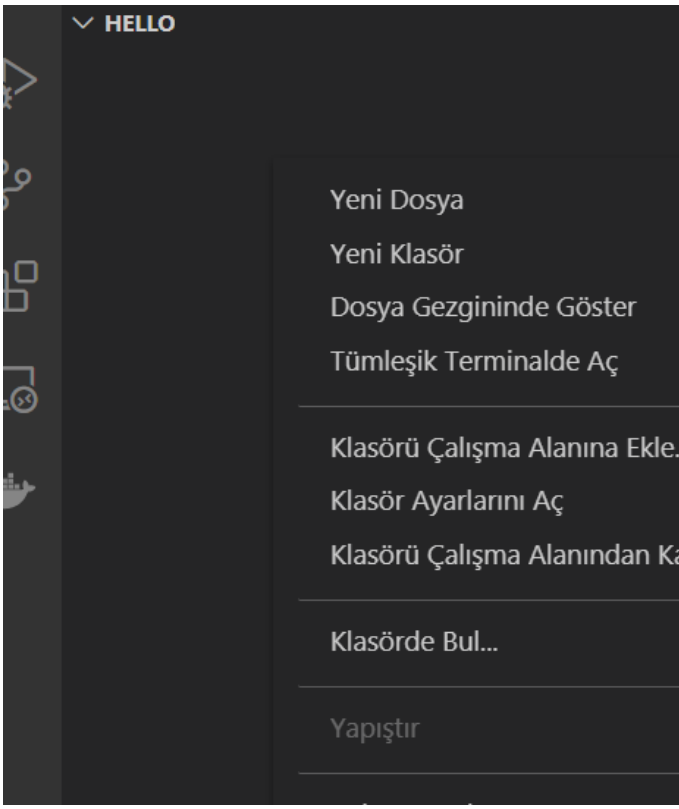
write

```
code .
```

0.8.0.44 VSCode (Install / Compile / Run / Debug) (4) This will open vscode for current folder . dot present current folder.

You will see a empty folder in the right window


Dosya Düzenle Seçim Görünüm Git Çalıştır Terminal Yardım
Kullanmaya Başlayın - hello - Visual Studio Code
GEZGİN
DÜZENLEYİCİLERİ AÇ
Kullanmaya Başlayın
HELLO
Kullanmaya Başlayın
Visual Studio Code
Düzenleme geliştirildi
Başlat
Yeni Dosya...
Dosya Açma...
Klasör Açma...
Git Deposunu Klonlama...
Son
C-clang E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103
Cpp-g++ C:\Users\ugur.coruh\Desktop\vscode-hello-world-
Cpp-clang C:\Users\ugur.coruh\Desktop\vscode-hello-world
Cpp-clang-g++ C:\Users\ugur.coruh\Desktop\vscode-hello-
C-clang C:\Users\ugur.coruh\Desktop\vscode-hello-world-ap
More...



0.8.0.45 VSCode (Install / Compile / Run / Debug) (5)

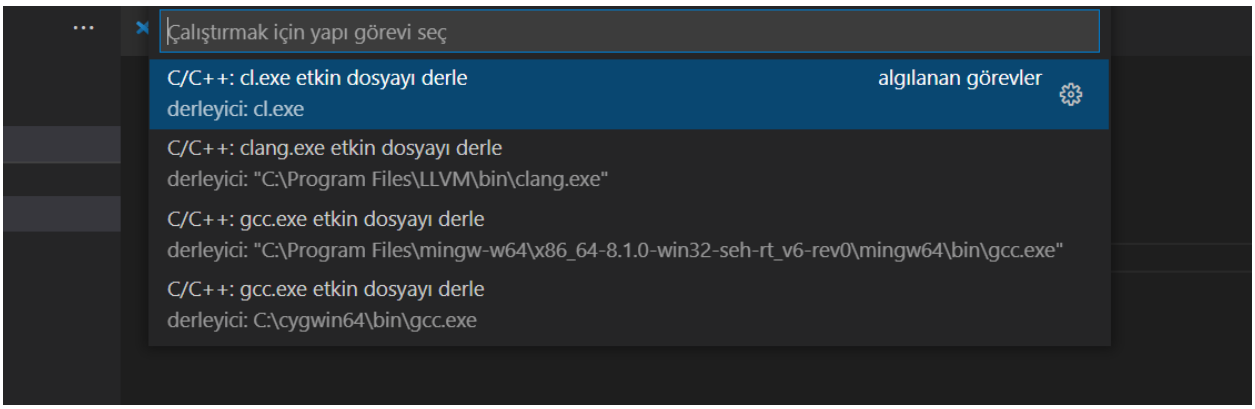
0.8.0.46 VSCode (Install / Compile / Run / Debug) (6)

Create a hello.c file and write following content

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

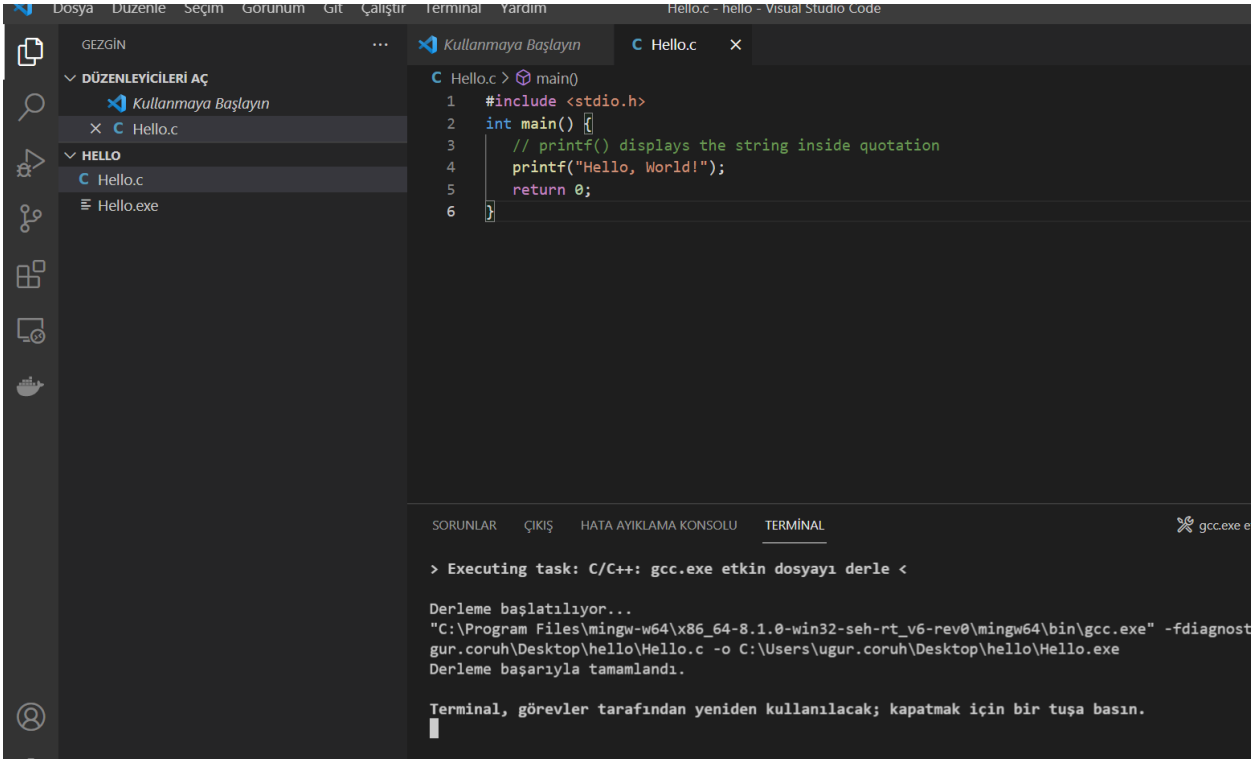
0.8.0.47 VSCode (Install / Compile / Run / Debug) (7)

use CTRL+SHIFT+B (you should be on source code section) to build file

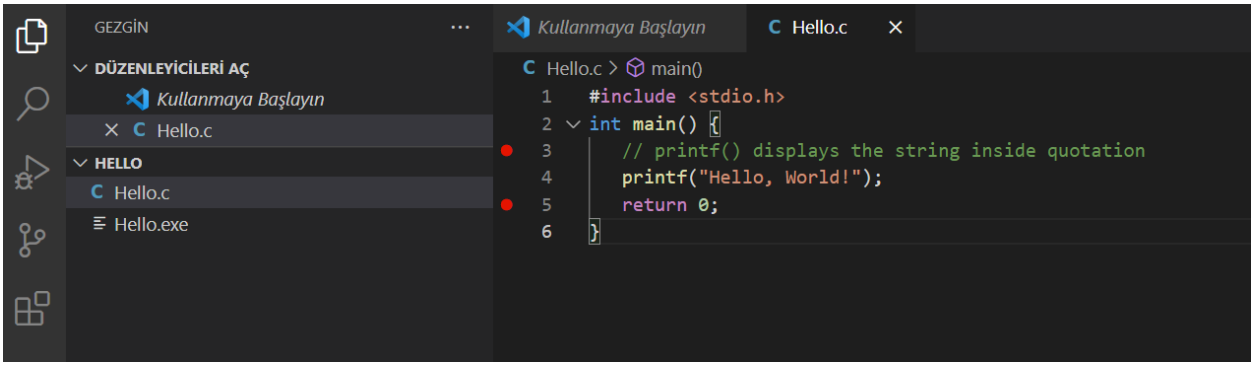


0.8.0.48 VSCode (Install / Compile / Run / Debug) (8) Select GCC or CLANG for this sample we can use GCC

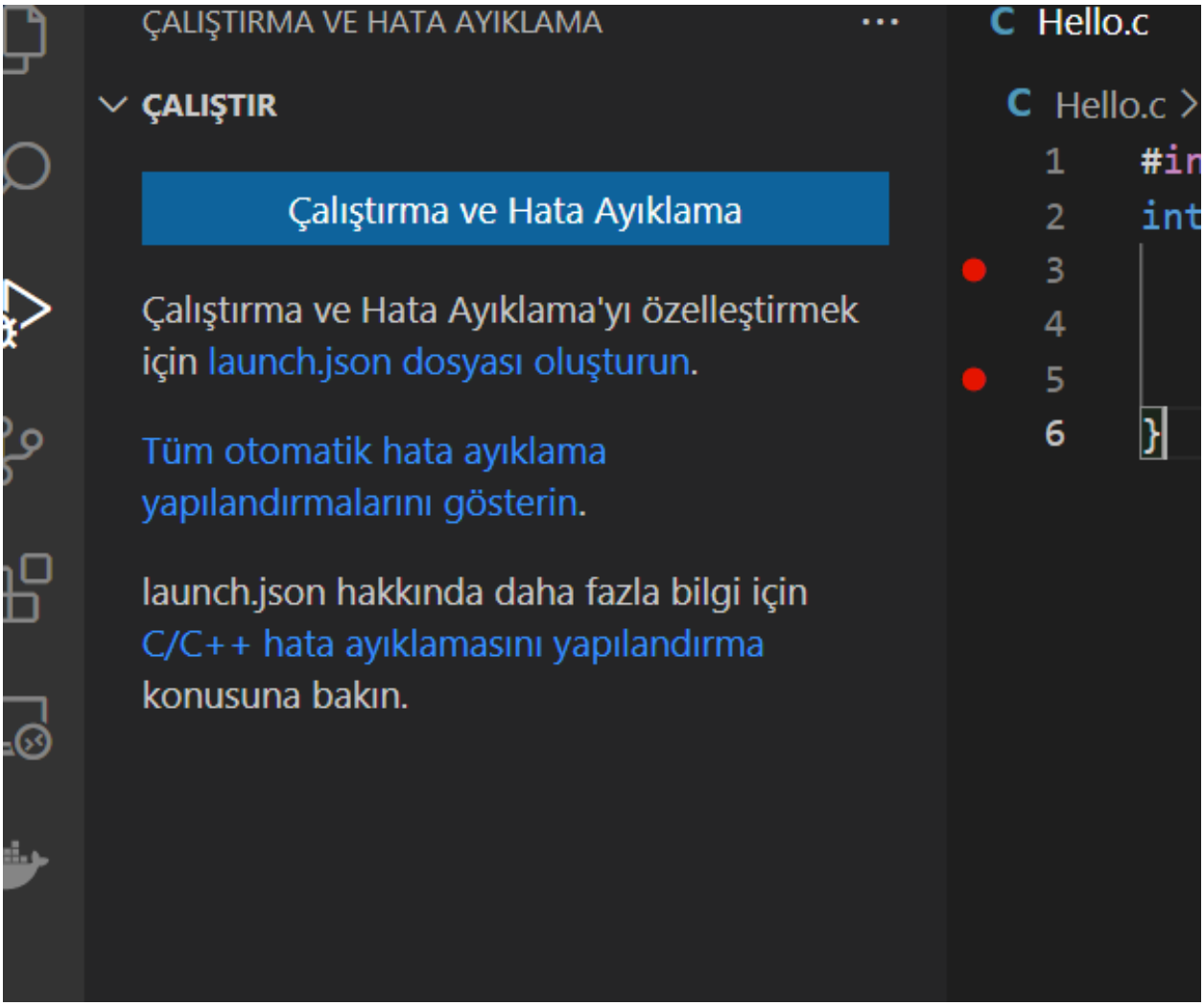
You will see output generated Hello.exe



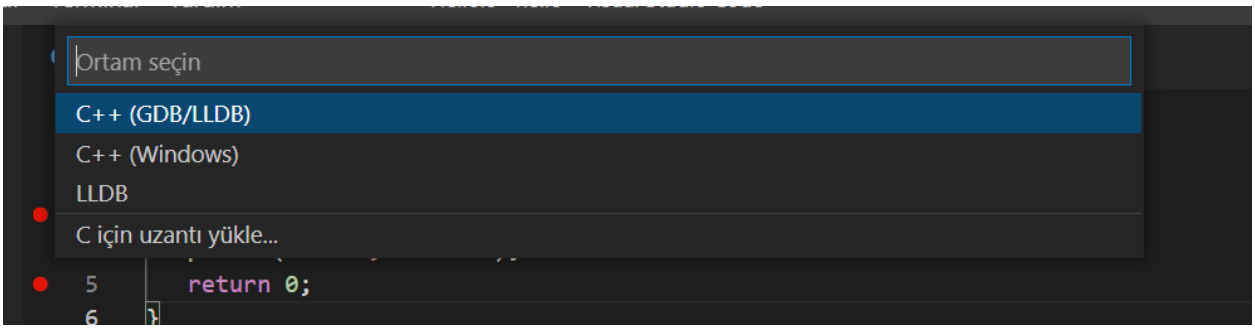
0.8.0.49 VSCode (Install / Compile / Run / Debug) (9) for debugging just put breakpoint and build again



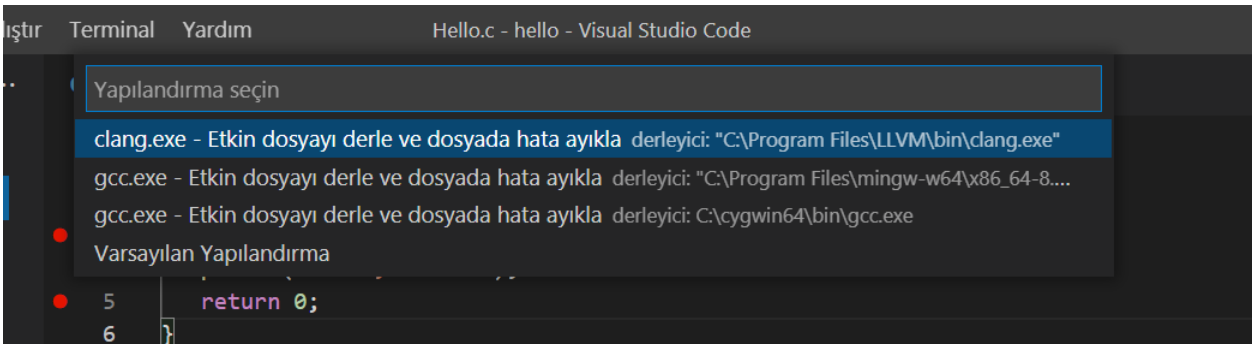
0.8.0.50 VSCode (Install / Compile / Run / Debug) (10) after build for debug press CTRL+SHIFT+D (you should be on source code section)and in the right window select create launch.json



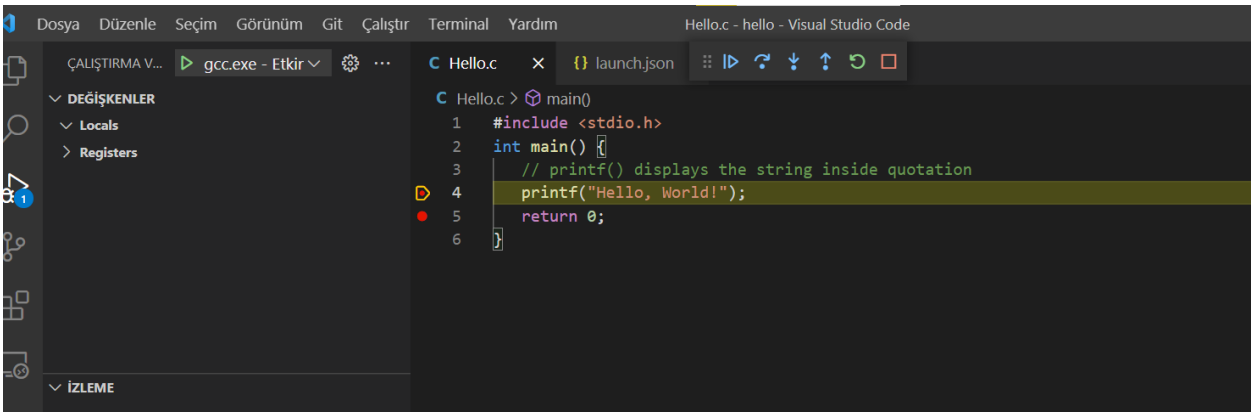
0.8.0.51 VSCode (Install / Compile / Run / Debug) (11) from opening window select C++ GDB/LLDB



0.8.0.52 VSCode (Install / Compile / Run / Debug) (12) from next opening menu select mingw-w64 gcc.exe

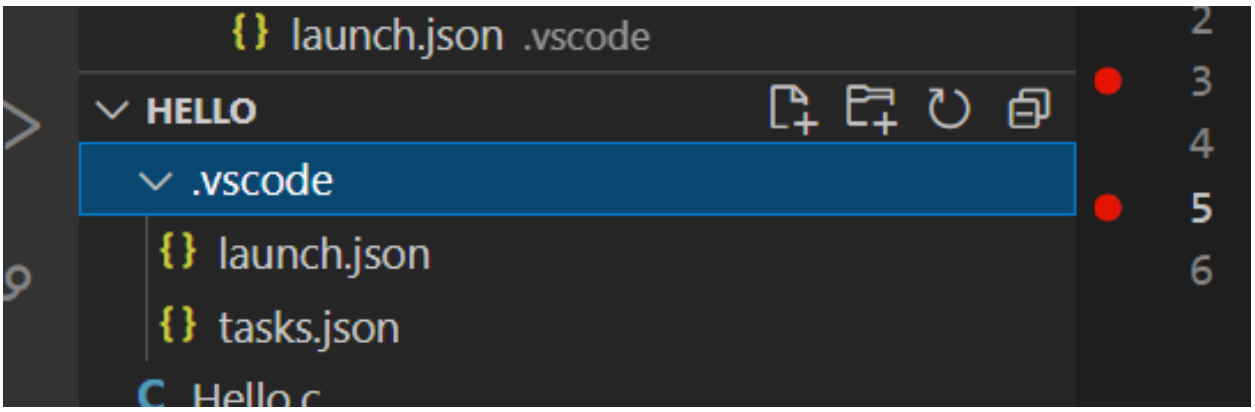


0.8.0.53 VSCode (Install / Compile / Run / Debug) (13) this will run debugger and you will see debug points activated



0.8.0.54 VSCode (Install / Compile / Run / Debug) (14) then you can step-by-step debug your code.

the following task.json and launch.json automatically generated with your selections



0.8.0.55 VSCode (Install / Compile / Run / Debug) (15) launch.json

```
{
    // Olası öznitelikler hakkında bilgi edinmek için IntelliSense kullanın.
    // Mevcut özniteliklerin açıklamalarını görüntülemek için üzerine gelin.
    // Daha fazla bilgi için şu adresi ziyaret edin: https://go.microsoft.com/fwlink/?linkid=830387
```

```
    "version": "0.2.0",
    "configurations": [
        {
            "name": "gcc.exe - Etkin dosyayı derle ve dosyada hata ayıkla",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "C:\\Program Files\\mingw-w64\\x86_64-8.1.0-win32-seh-rt_v6-rev0\\mingw64\
            "setupCommands": [
                {
                    "description": "gdb için düzgün yazdırmayı etkinleştir",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "C/C++: gcc.exe etkin dosyayı derle"
        }
    ]
}
```

0.8.0.56 VSCode (Install / Compile / Run / Debug) (16) task.json

```
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: gcc.exe etkin dosyayı derle",
            "command": "C:\\Program Files\\mingw-w64\\x86_64-8.1.0-win32-seh-rt_v6-rev0\\mingw64\\bin\\g
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Hata Ayıklayıcısı tarafından oluşturulan görev."
        }
    ],
    "version": "2.0.0"
}
```

0.8.0.57 VSCode (Install / Compile / Run / Debug) (17) you can do the same thing for other compilers and c++ source codes. LLVM do not support debug on vscode now.

for C++ VsCode you can check the following links

for Windows

https://code.visualstudio.com/docs/cpp/config-mingw

for Linux

https://code.visualstudio.com/docs/cpp/config-linux

for WSL

https://code.visualstudio.com/docs/cpp/config-wsl

0.8.0.58 VSCode (Install / Compile / Run / Debug) (18) in the launch file if you start debugging with F5

(you can select debugger with CTRL+SHIFT+P and then writing Debug and Selecting Configure Debugger Option)

0.8.0.59 VSCode (Install / Compile / Run / Debug) (19) following line will be your debugging application path

if you start debugging with F5 in Hello.c file this will set `<Hello.c base path>/Hello.exe`

0.8.0.60 VSCode (Install / Compile / Run / Debug) (20) You should set this correct for both LLVM and GCC configuration in launch.json

```
 "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
```

Also you should set your installed debugger paths

for GCC

```
"miDebuggerPath": "C:\\Program Files\\mingw-w64\\x86_64-8.1.0-win32-seh-rt_v6-rev0\\mingw64\\bin\\gdb.e
```

for LLVM

```
"miDebuggerPath": "C:\\Program Files\\LLVM\\bin\\lldb.exe",
```

for more details please check the sample source codes.

0.8.1 Visual Studio Community Edition (Install / Compile / Run / Debug)

//TODO//

0.8.1.1 Notepad++ (Install / Compile) (1) Please download Notepad++ from the following link

Downloads | Notepad++[13]

0.8.1.2 Notepad++ (Install / Compile) (2) Download and install MinGW or LLVM compiler (if you downloded then skip this step)

MinGW installer (clang)

Download MinGW-w64 - for 32 and 64 bit Windows from SourceForge.net[14]

LLVM installer (gcc / g++)

Download LLVM releases[15]

Also use the following notes

https://llvm.org/devmtg/2014-04/PDFs/Talks/clang-cl.pdf

0.8.1.3 Notepad++ (Install / Compile) (3) Open a console with "cmd" and test the following commands if commands are not recognized then set the system environment variable add gcc and g++ exe paths to path variable (add to both system and user path variable)

```
gcc --version

g++ --version
```

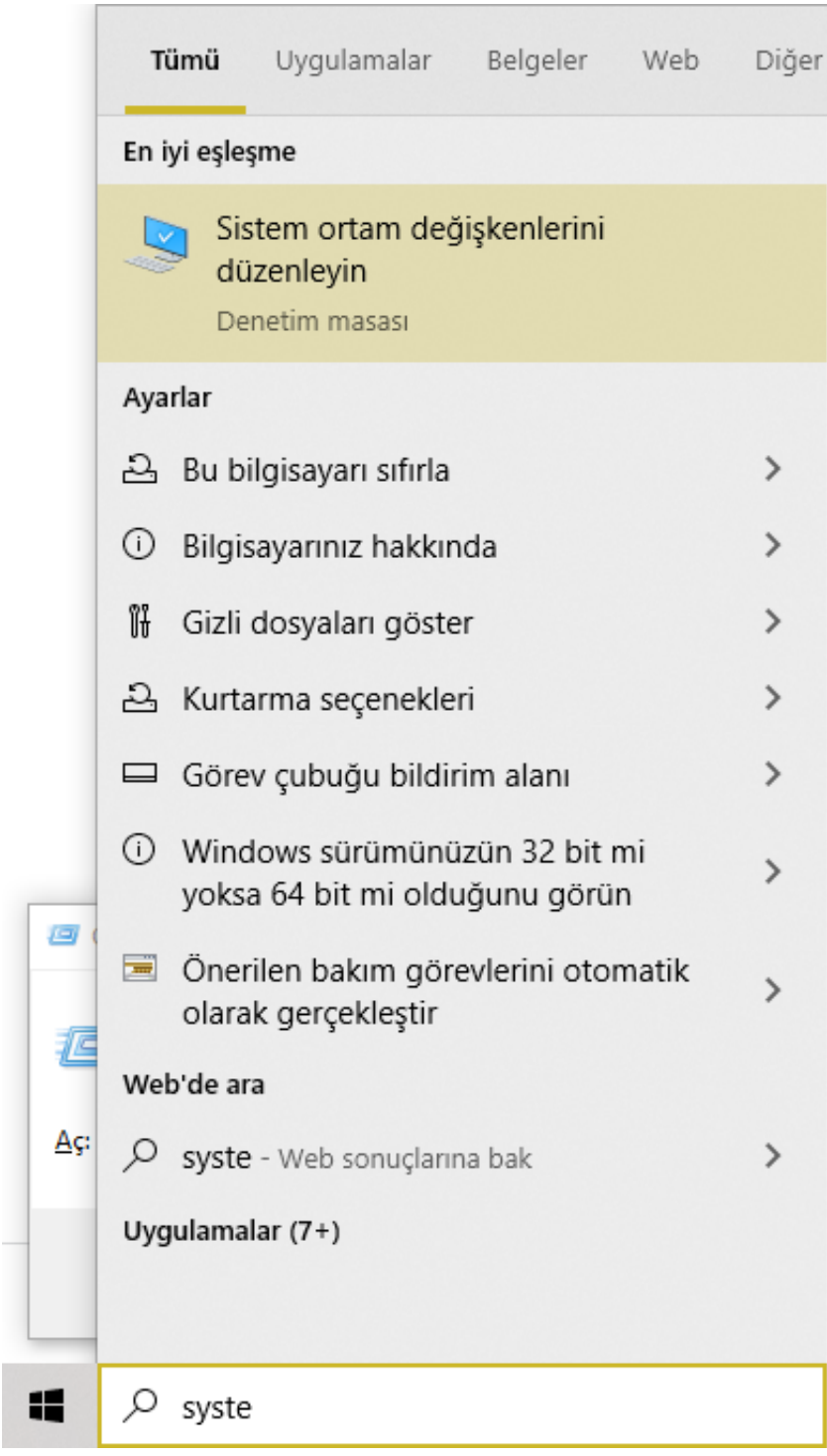
```
C:\Users\ugur.coruh>gcc --version
gcc (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

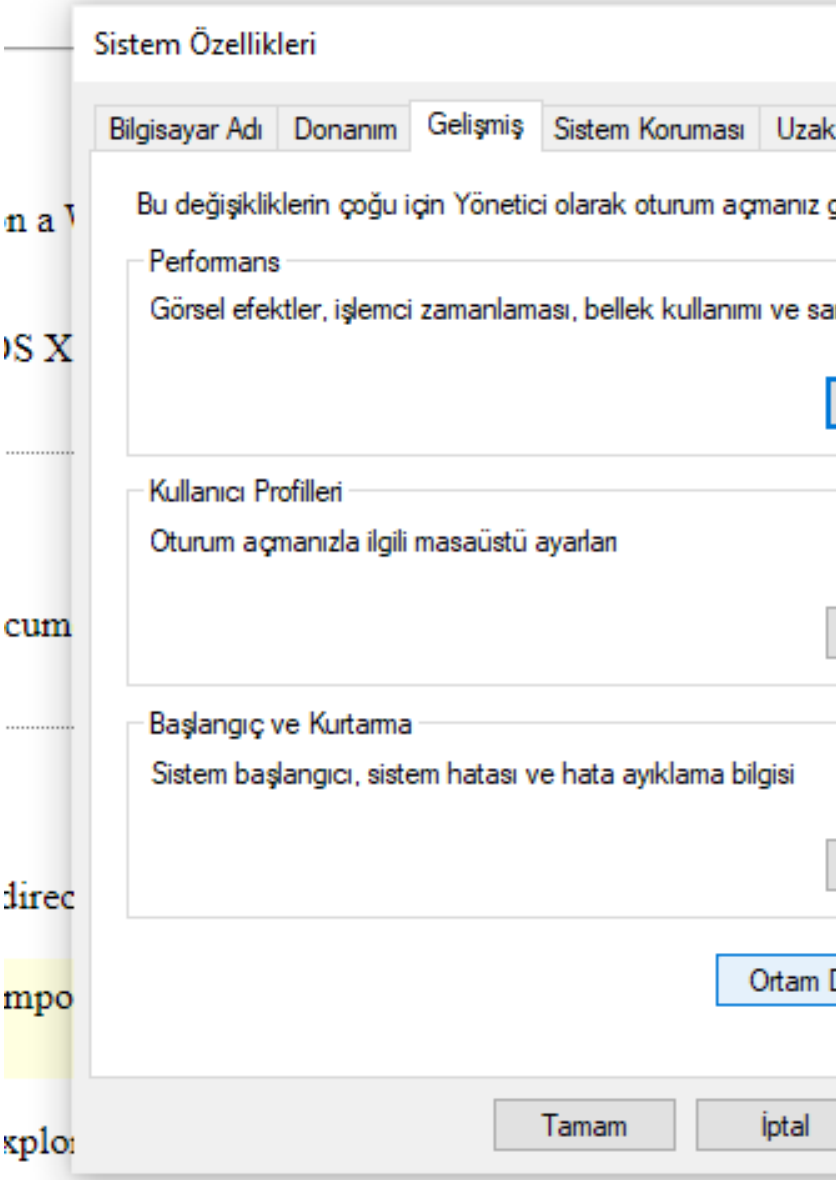
[13] https://notepad-plus-plus.org/downloads/

[14] https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download

[15] https://releases.llvm.org/

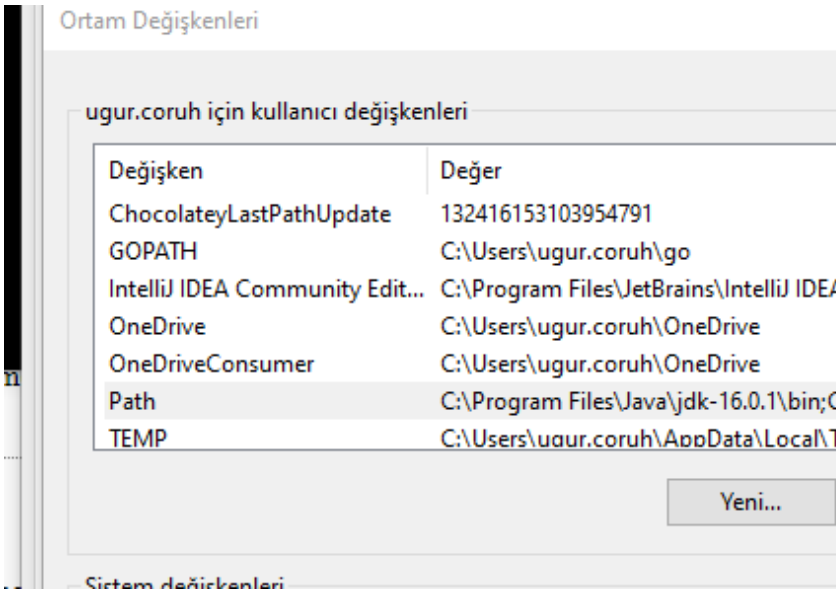
0.8.1.4 Notepad++ (Install / Compile) (4)



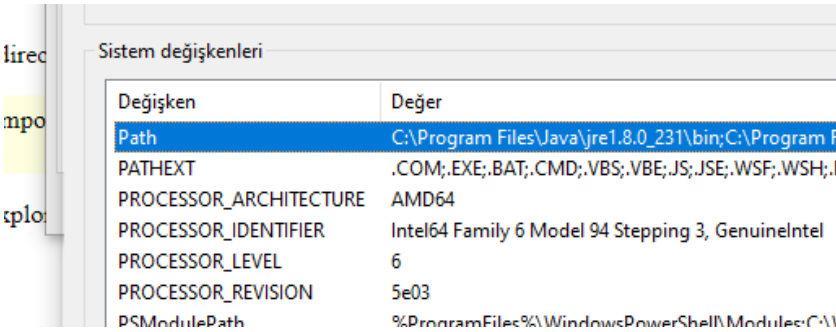


0.8.1.5 Notepad++ (Install / Compile) (5)

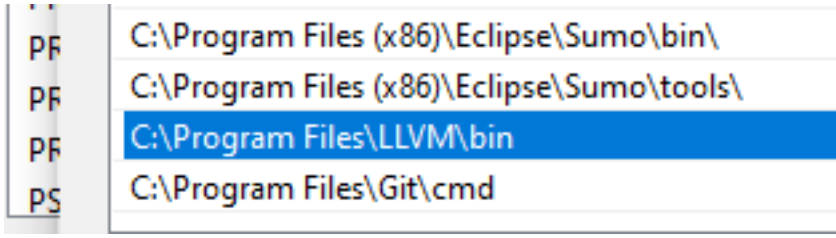
0.8.1.6 Notepad++ (Install / Compile) (6)



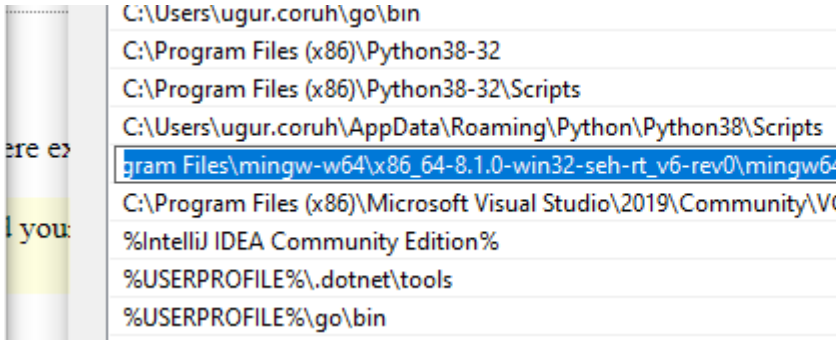
0.8.1.7 Notepad++ (Install / Compile) (7)



0.8.1.8 Notepad++ (Install / Compile) (8)



0.8.1.9 Notepad++ (Install / Compile) (9)



0.8.1.10 Notepad++ (Install / Compile) (10) for gcc.exe, g++.exe and gdb.exe

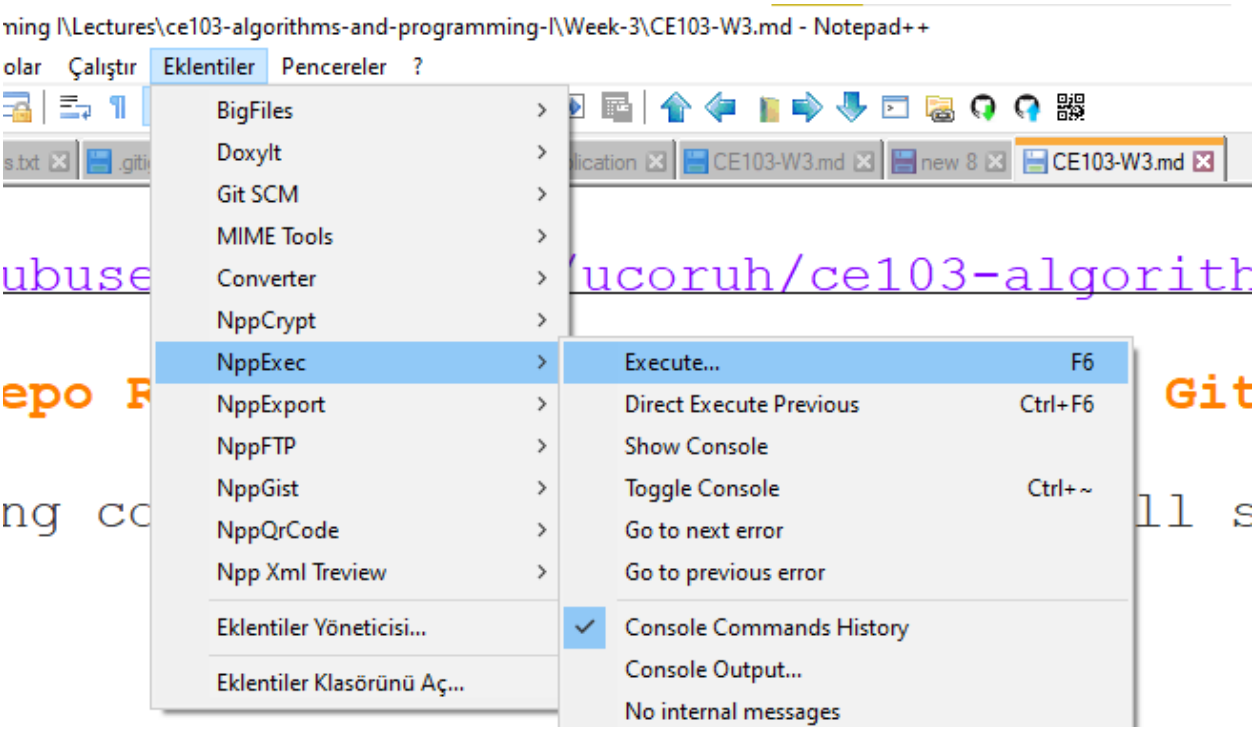
```
C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw64\bin
```

0.8.1.11 Notepad++ (Install / Compile) (11) for clang.exe , lldb.exe

```
C:\Program Files\LLVM\bin
```

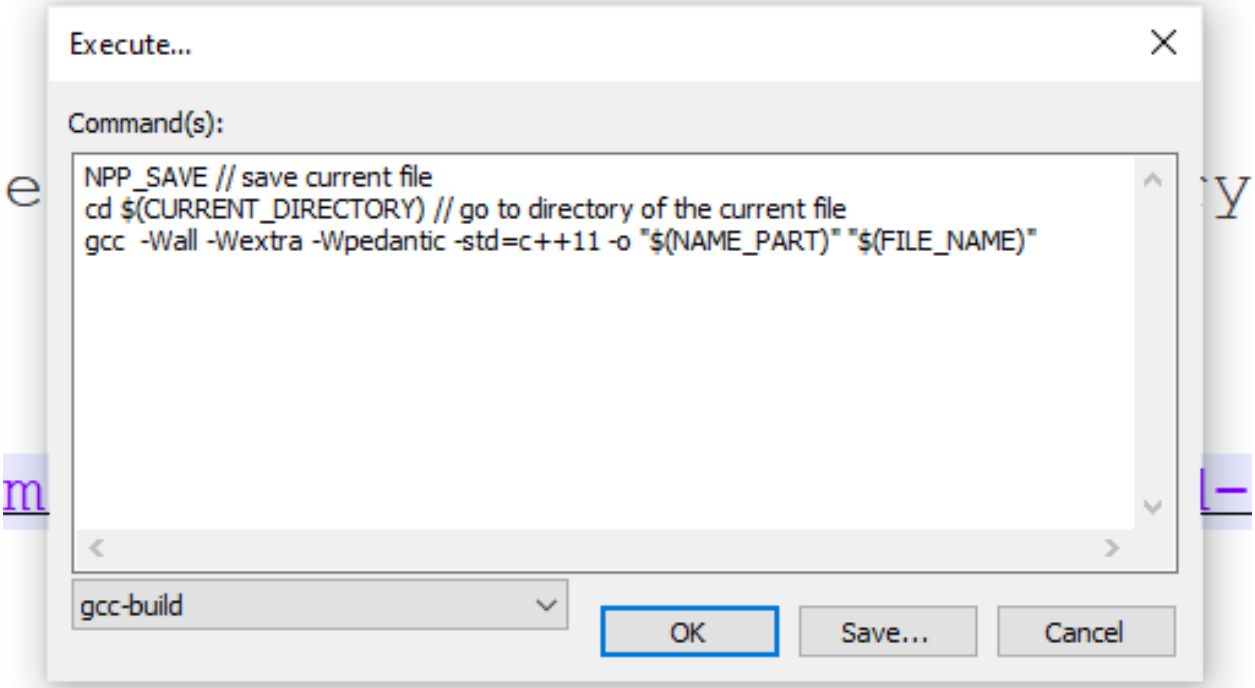
0.8.1.12 Notepad++ (Install / Compile) (12) This folder paths changes according to your setup

Open NppExec extension (install from extension manager if not exist)



0.8.1.13 Notepad++ (Install / Compile) (13) write the following commands in box

```
NPP_SAVE // save current file
cd $(CURRENT_DIRECTORY) // go to directory of the current file
gcc  -Wall -Wextra -Wpedantic -std=c++11 -o "$(NAME_PART)" "$(FILE_NAME)"
```



0.8.1.14 Notepad++ (Install / Compile) (14) save script as gcc-build and for more information check the following link

How To Setup Notepad for Writing C++ Programs[16]

You can modify or add multiple scripts for another tasks.

0.8.1.15 Vi/Vim (C/C++) //TODO//

[16] http://www.edparrish.net/common/npp4c.html

0.8.1.16 Eclipse (C/C++) //TODO//

0.8.1.17 Netbeans (C/C++) //TODO//

0.8.1.18 Turbo C++/C //TODO//

0.8.1.19 Cmake (C++/C) (1) **CMake** (http://www.cmake.org/) is a program which generates the **Makefiles** used by **Make**.

0.8.1.20 Cmake (C++/C) (2) Why use **CMake** ?

- Eases **Make** use
  - but the same way of thinking
  - generate the **Makefile**
- Separate the compilation from the sources
- Multi-platfoms
- Very flexible

0.8.1.21 Cmake (C++/C) (3)

- Check if the libraries/programs are available on your system
- File generator (**configure_file**)
- Calling programs or scripts (**doxygen**)
- One of the new standards

0.8.1.22 Cmake (C++/C) (4) (Download and Install) use the following link for download

Download | CMake[17]

0.8.1.23 Cmake (C++/C) (5) (WSL and Linux Environment) Hello world with CMake[18]

---

[17] https://cmake.org/download/
[18] https://lappweb.in2p3.fr/~paubert/ASTERICS_HPC/2-2-100.html

0.8.1.24 Cmake (C++/C) (6) (Windows Environment) main.c

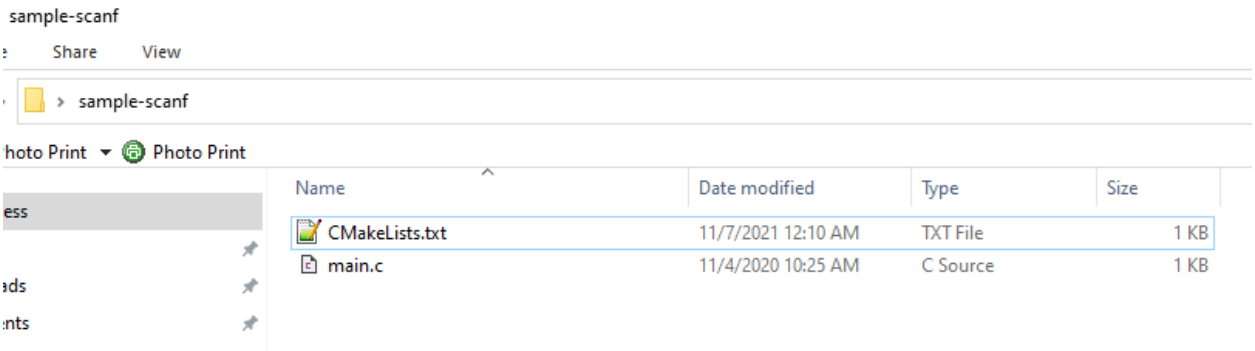
```c
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7.2)
project(scanf-sample)
add_executable(scanf-sample main.c)
```

0.8.1.25 Cmake (C++/C) (7) (Windows Environment)

put main.c and CMakeLists.txt file in sample-scanf folder and from command line



run the following cmake command with dot (.) to create solution file for c project

```
C:\Users\ugur.coruh\Desktop\sample-scanf>cmake .
```

0.8.1.26 Cmake (C++/C) (8) (Windows Environment)

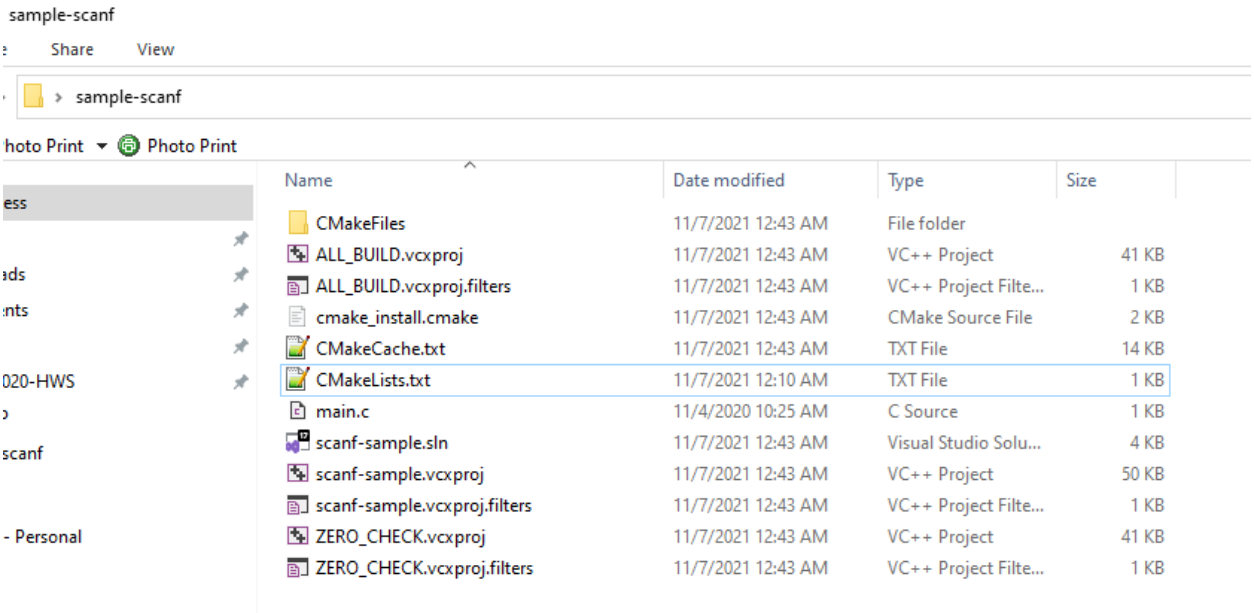
I have Visual Studio 2022 Community Edition Installed on My Computer, for these reason build tools are selected for visual studio environment and the following outputs are generated

```
C:\Users\ugur.coruh\Desktop\sample-scanf>cmake .
-- Building for: Visual Studio 17 2022
-- Selecting Windows SDK version 10.0.22000.0 to target Windows 10.0.19043.
-- The C compiler identification is MSVC 19.30.30704.0
-- The CXX compiler identification is MSVC 19.30.30704.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSV
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
```

```
-- Generating done
-- Build files have been written to: C:/Users/ugur.coruh/Desktop/sample-scanf

C:\Users\ugur.coruh\Desktop\sample-scanf>
```

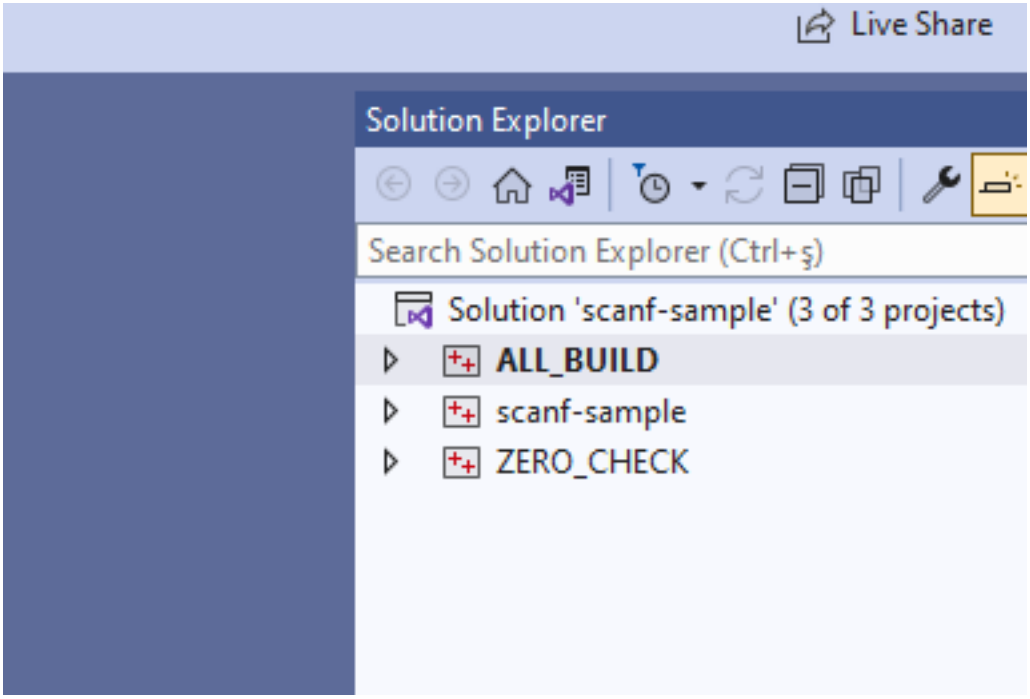
0.8.1.27 Cmake (C++/C) (9) (Windows Environment)

also following files are generated



0.8.1.28 Cmake (C++/C) (10) (Windows Environment)

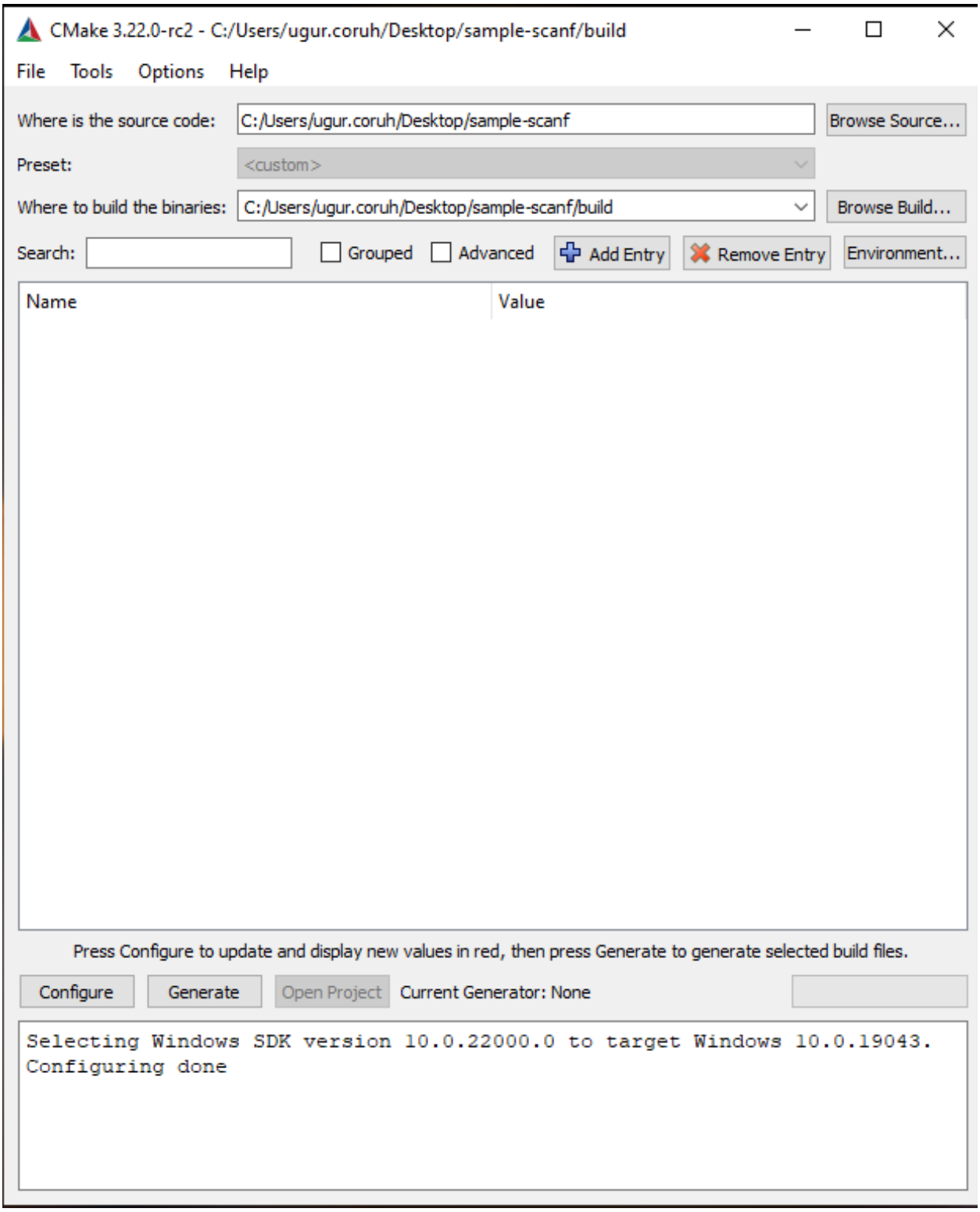
if we open scanf-sample.sln file we will have automated generated project files



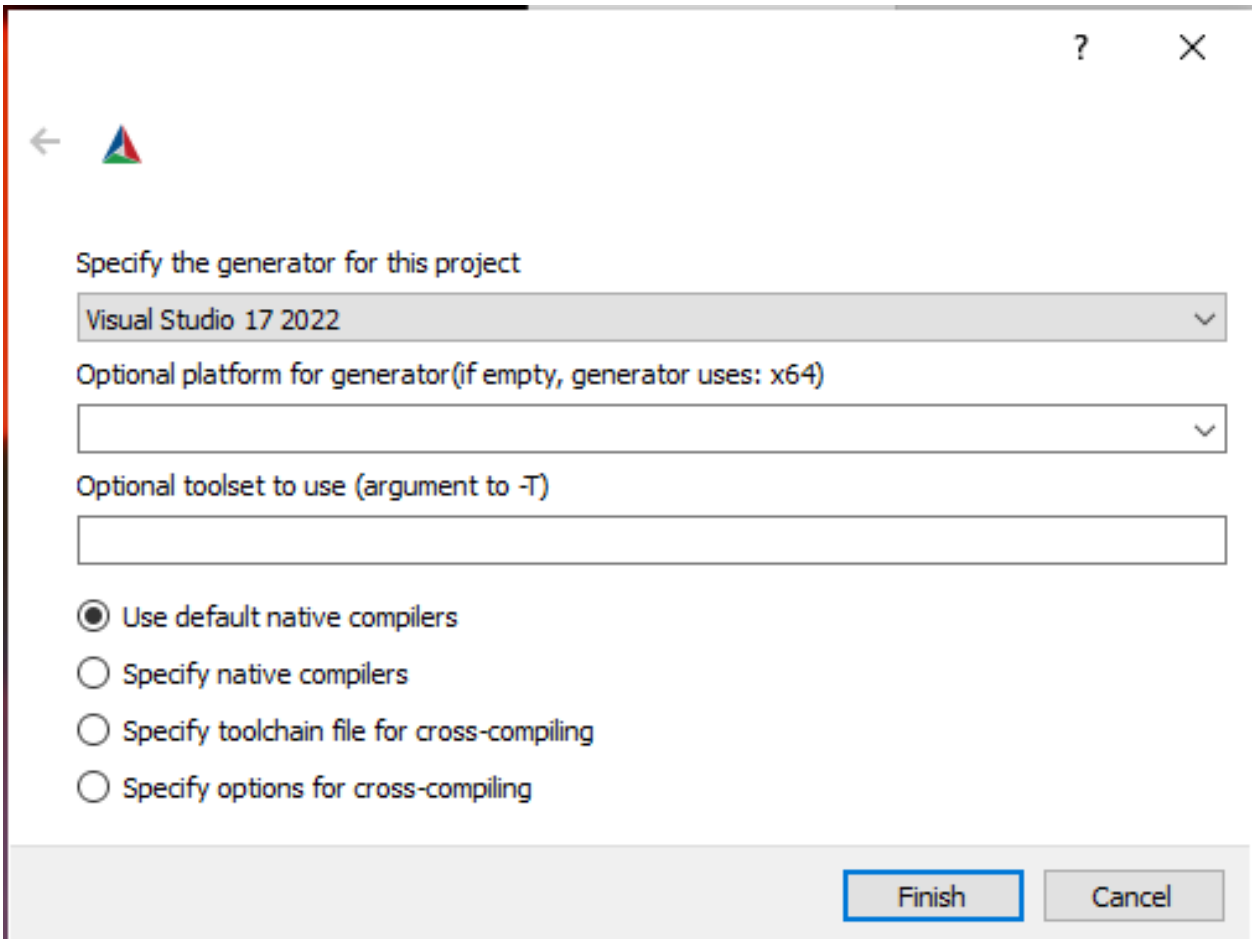
0.8.1.29 Cmake (C++/C) (11) (Windows Environment) you can make scanf-sample with startup project with right click and then run on visual studio.

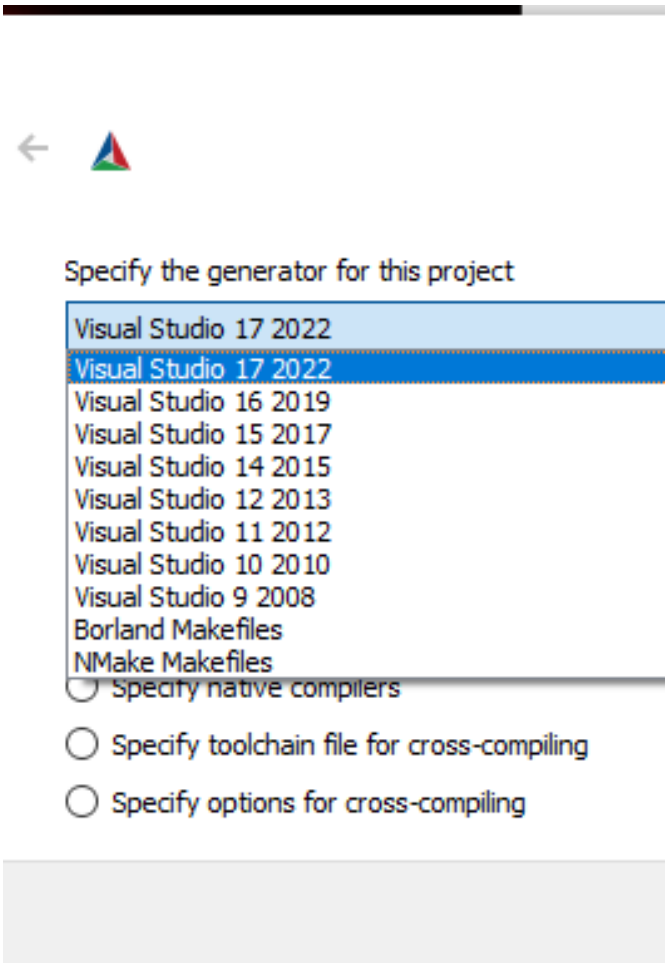
if you want to configure for another build tool you can use Cmake-GUI installed with setup on your computer

0.8.1.30 Cmake (C++/C) (12) (Windows Environment) Open GUI and Select *File-> Delete Cache*



0.8.1.31 Cmake (C++/C) (13) (Windows Environment) then you can click "Configure" to select build tool


Specify the generator for this project
Visual Studio 17 2022
Optional platform for generator(if empty, generator uses: x64)
Optional toolset to use (argument to -T)
Use default native compilers
Specify native compilers
Specify toolchain file for cross-compiling
Specify options for cross-compiling
Finish
Cancel



0.8.1.32 Cmake (C++/C) (14) (Windows Environment)

0.8.1.33 Cmake (C++/C) (15) (Windows Environment) if you click "Configure" twice it will generate the visual studio solution in build folder

for more detailed examples that include also docker and travis-ci sample you can check the following repo

GitHub - ttroy50/cmake-examples: Useful CMake Examples[19]

0.8.1.34 Make (1) Sample

hello.c

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

0.8.1.35 Make (2) Makefile

```
# This is the default target, which will be built when
# you invoke make
```

[19] https://github.com/ttroy50/cmake-examples

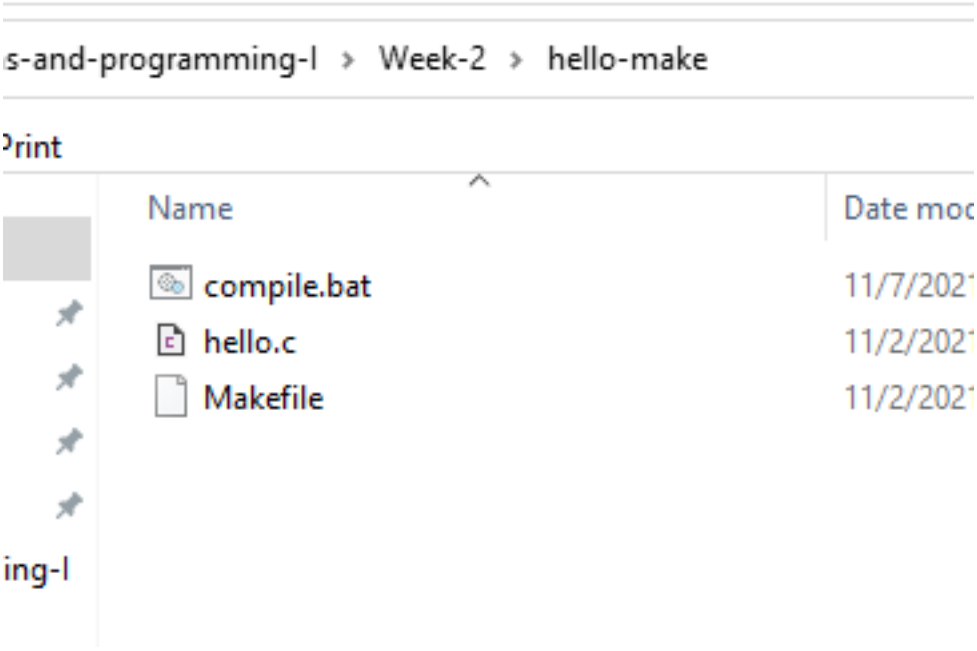
```
.PHONY: all
all: hello

# This rule tells make how to build hello from hello.cpp
hello: hello.c
    g++ -o hello hello.c

# This rule tells make to copy hello to the binaries subdirectory,
# creating it if necessary
.PHONY: install
install:
    mkdir -p binaries
    cp -p hello binaries

# This rule tells make to delete hello and hello.o
.PHONY: clean
clean:
    rm -f hello
```

0.8.1.36 Make (3) compile.bat

`make all .`

will create hello.exe

check hello-make sample



---

1 JAVA Environment and Development

//TODO//

1.0.1 JDK and JRE Setup

//TODO//

1.0.2 System Environments and Paths for Java

//TODO//

1.0.3 Netbeans (Java)

//TODO//

1.0.4 Eclipse (Java)

//TODO//

1.0.5 Intellij Idea (Jet Brains) (Java)

//TODO//

1.0.6 VSCode (Java)

//TODO//

1.0.7 Notepad++ (Java)

//TODO//

1.0.8 Cmake (Java)

ASTERICS_HPC[20]

---

[20] https://lappweb.in2p3.fr/~paubert/ASTERICS_HPC/outline.html

2 C# Environment and Development

2.0.1 Visual Studio Community Edition (C#)

//TODO//

2.0.2 Notepad++ (C#)

//TODO//

2.0.3 Cmake (C#)

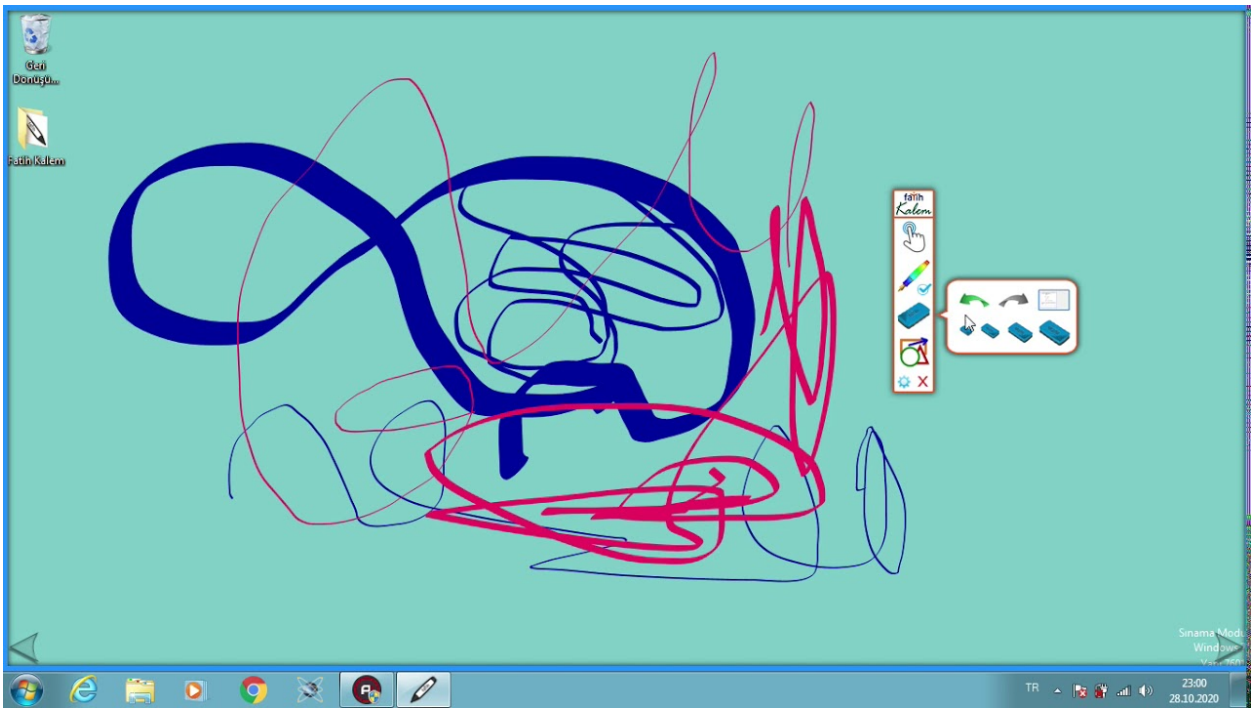
Outline[21]

2.0.4 Common Tools and Platforms

2.0.5 Fatih Kalem

https://cdnvideo.eba.gov.tr/fatihkalem/fatihkalem_portable.zip

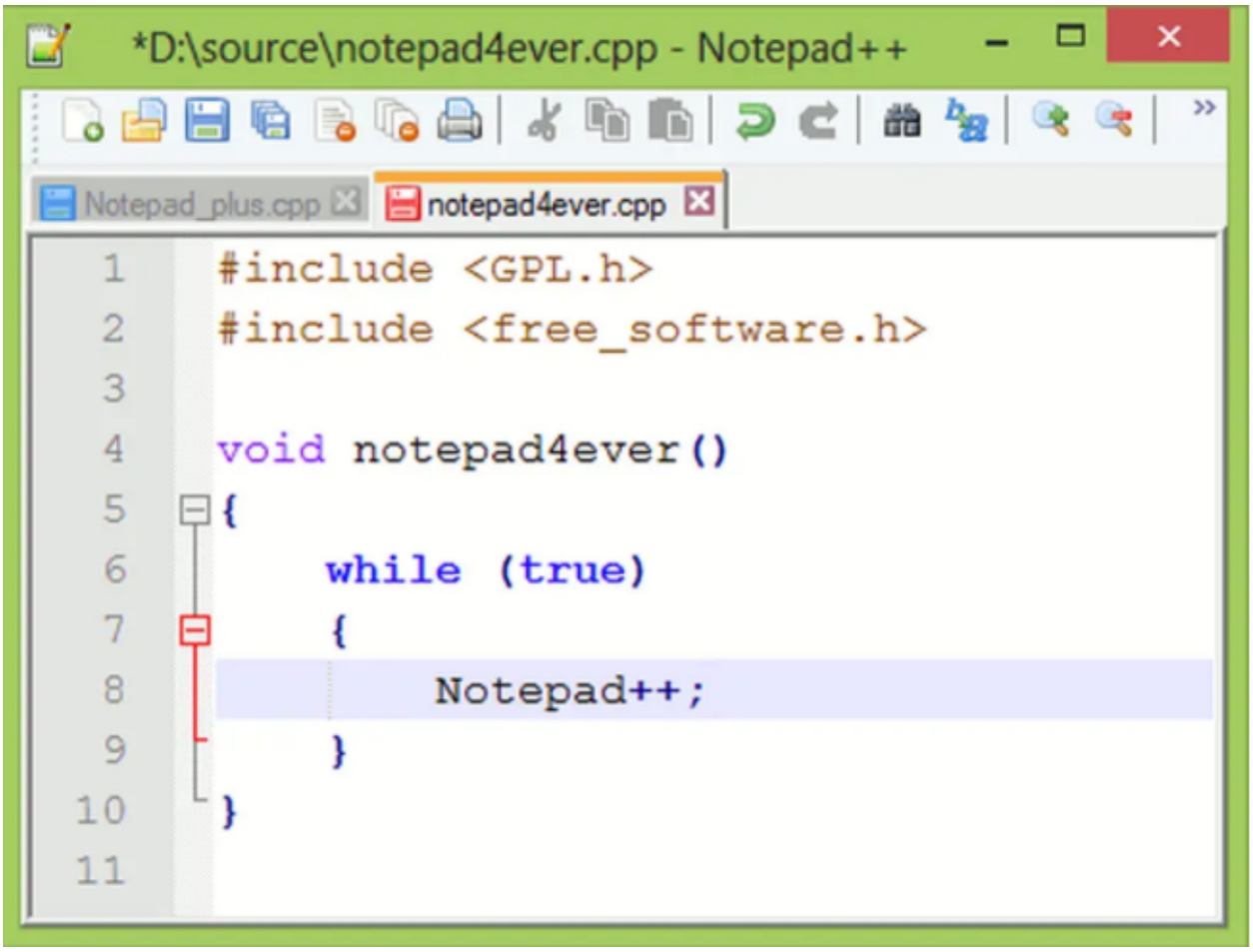
https://cdnvideo.eba.gov.tr/fatihkalem/fatihkalem_setup.exe



2.0.6 Notepad++ (Notepad for Source Code)

Downloads | Notepad++[22]

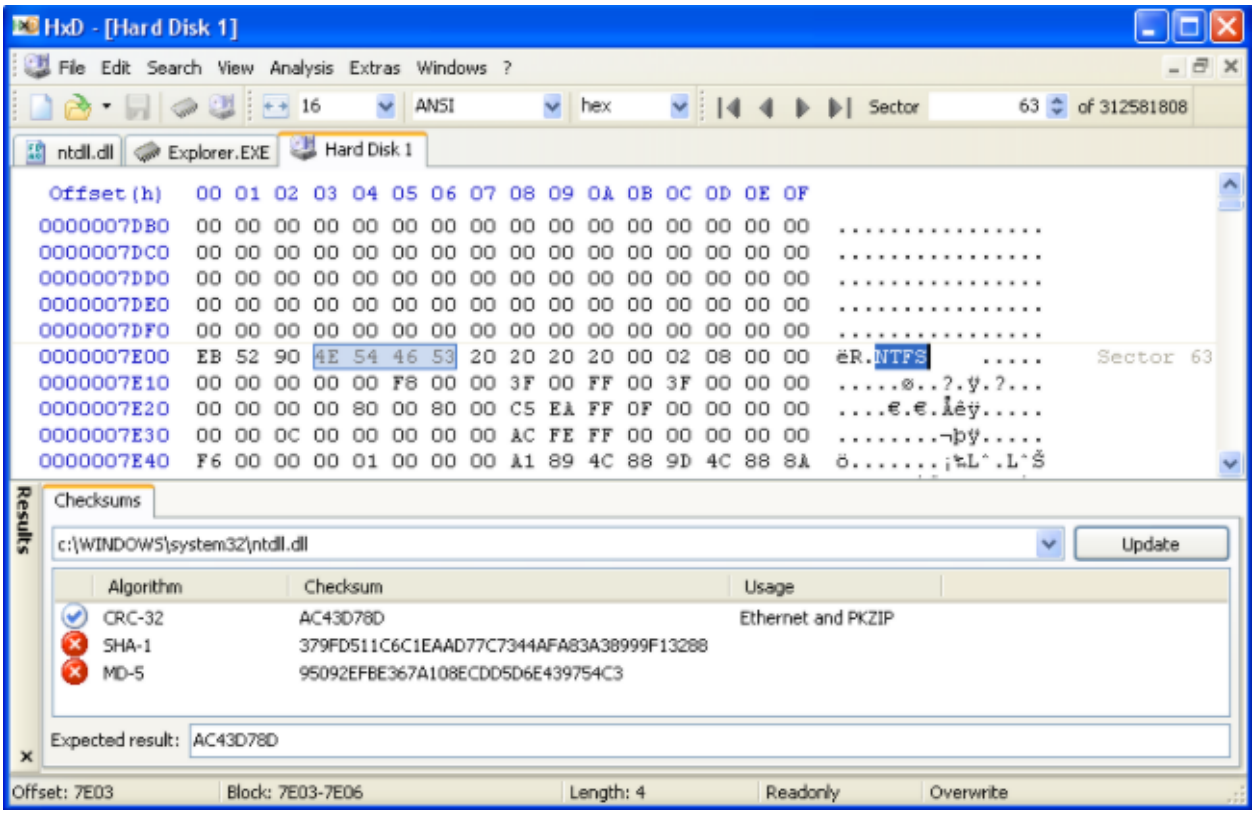
[21] https://lappweb.in2p3.fr/~paubert/ASTERICS_HPC/outline.html
[22] https://notepad-plus-plus.org/downloads/



2.0.7 HxD (Hex Editor)

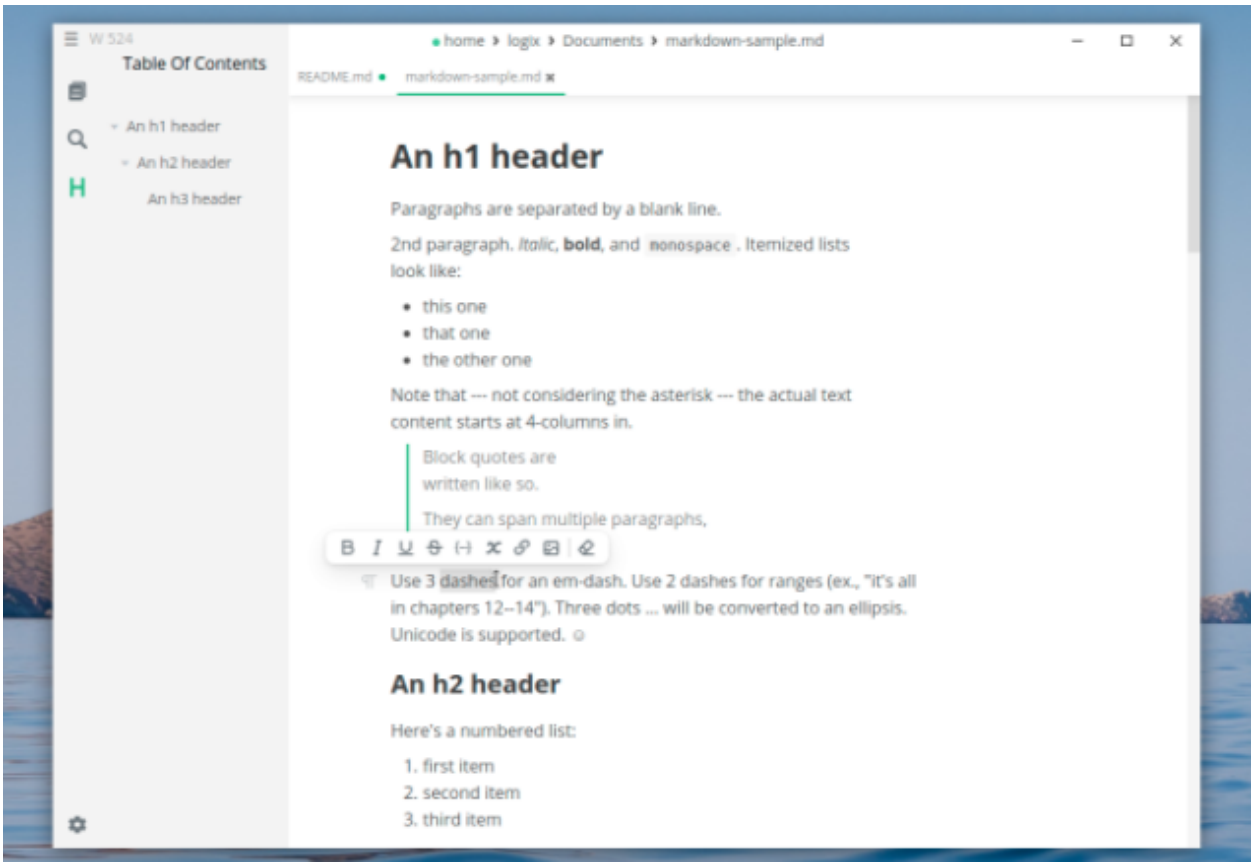
HxD - Freeware Hex Editor and Disk Editor | mh-nexus[23]

[23] https://mh-nexus.de/en/hxd/



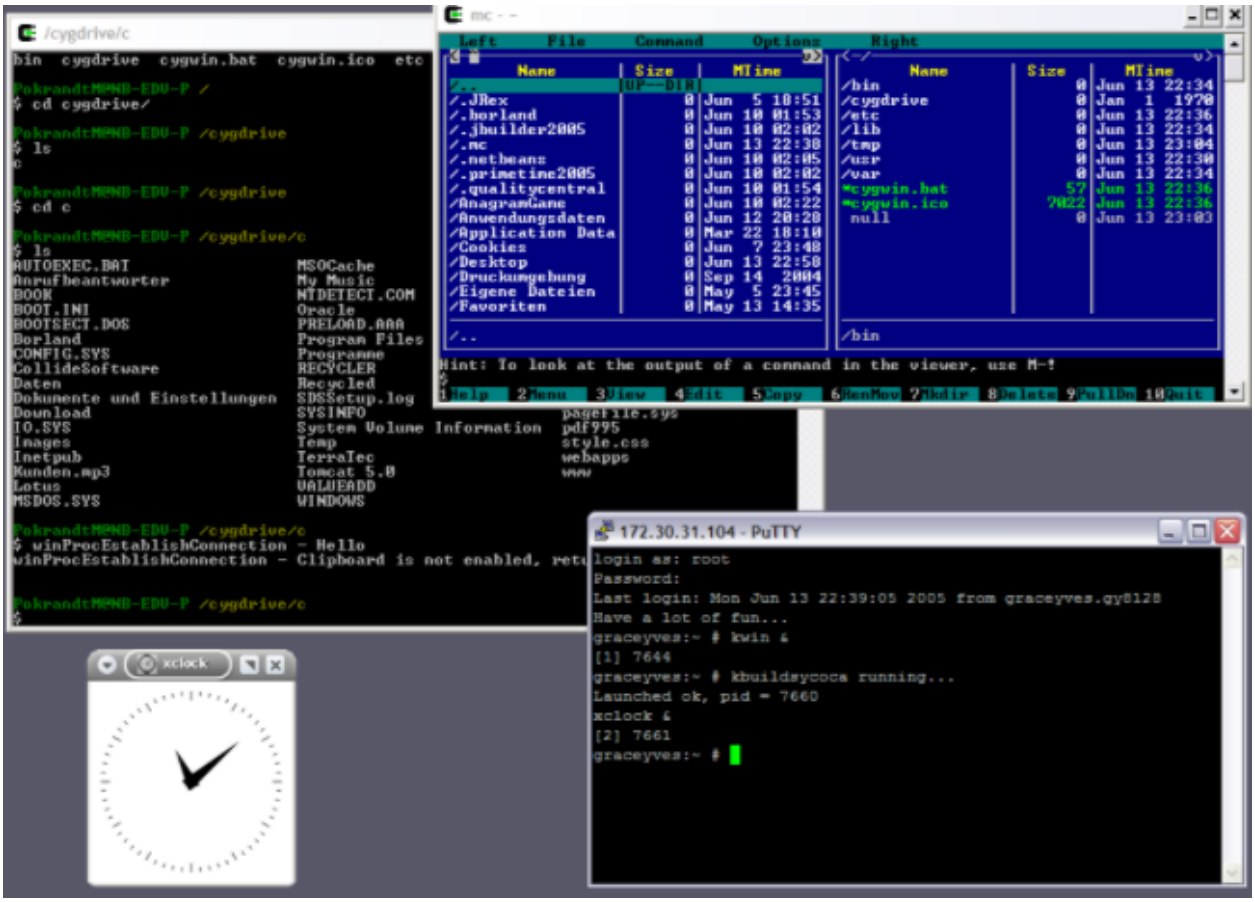
2.0.8 Marktext (Markdown Syntax Editor)

https://marktext.app/



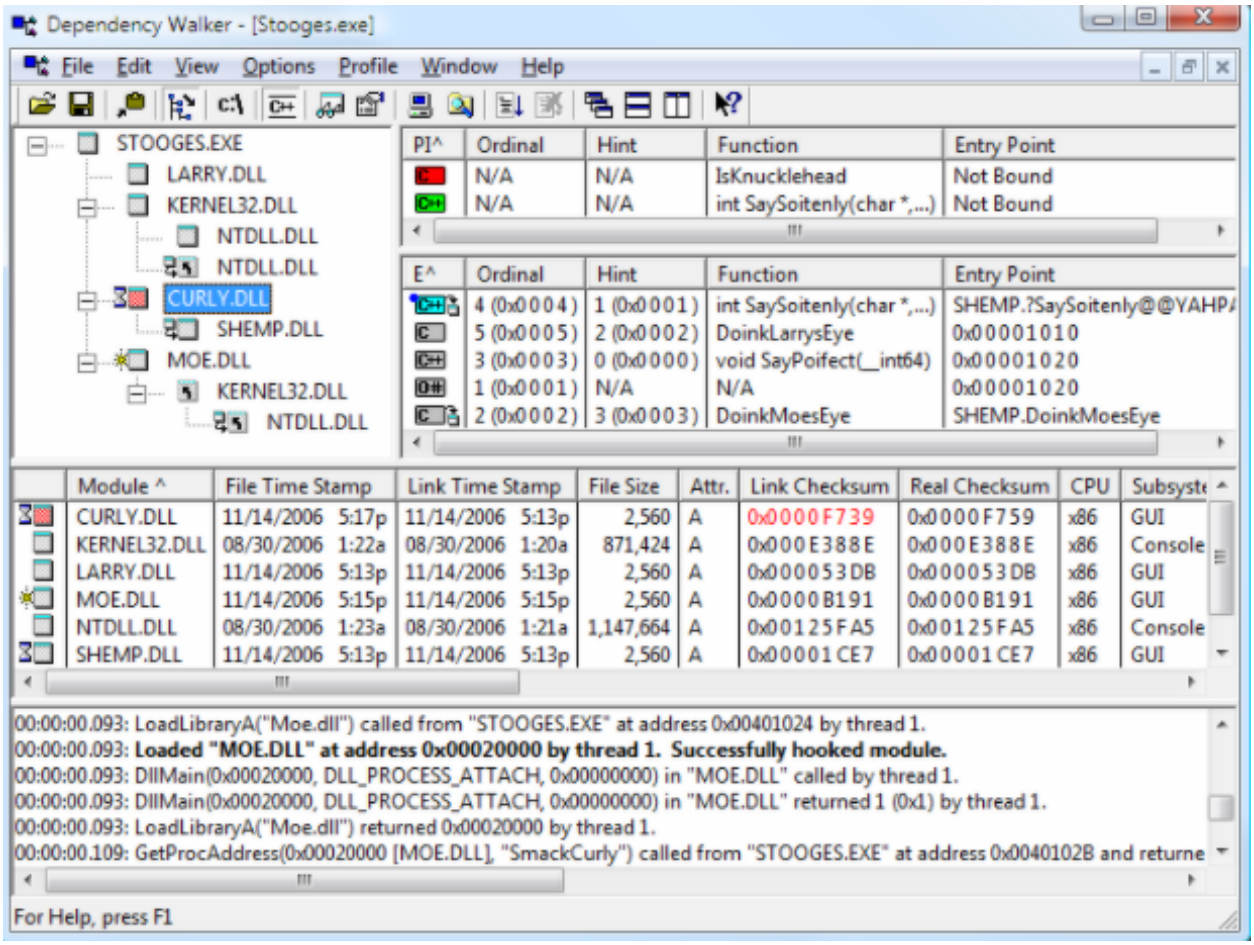
2.0.9 Cygwin (Linux environment for Windows)

https://www.cygwin.com/



2.0.10 Dependency Walker (32-bit or 64-bit Windows module dependency checker)

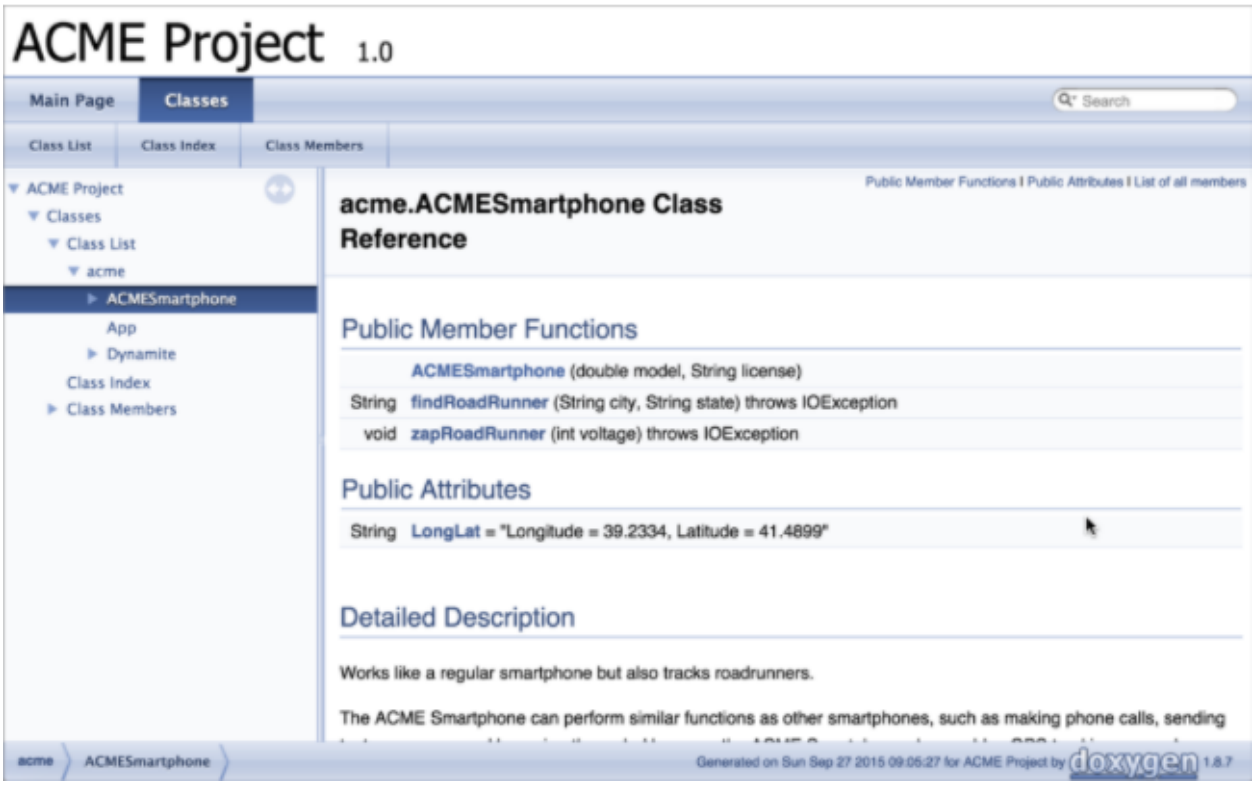
https://www.dependencywalker.com/



2.0.11 Doxygen (Code Documentation)

Doxygen: Doxygen[24]

[24] https://www.doxygen.nl/index.html



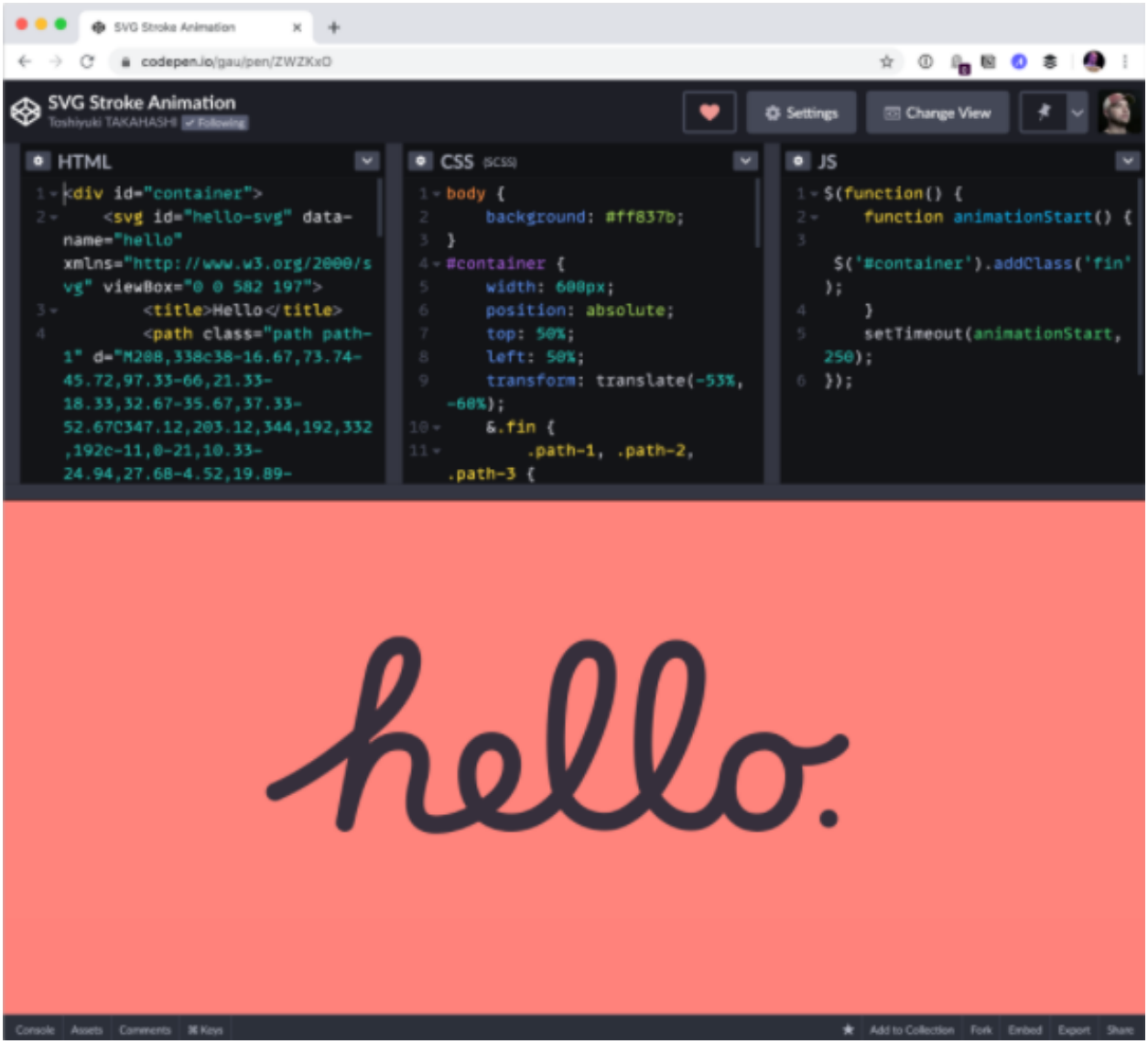
2.0.12 Sonarlint (Code Quality and Code Security Extension)

sonarlint

https://www.sonarlint.org/

2.0.13 Codepen.io (online code sharing)

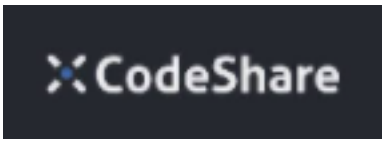
https://codepen.io/



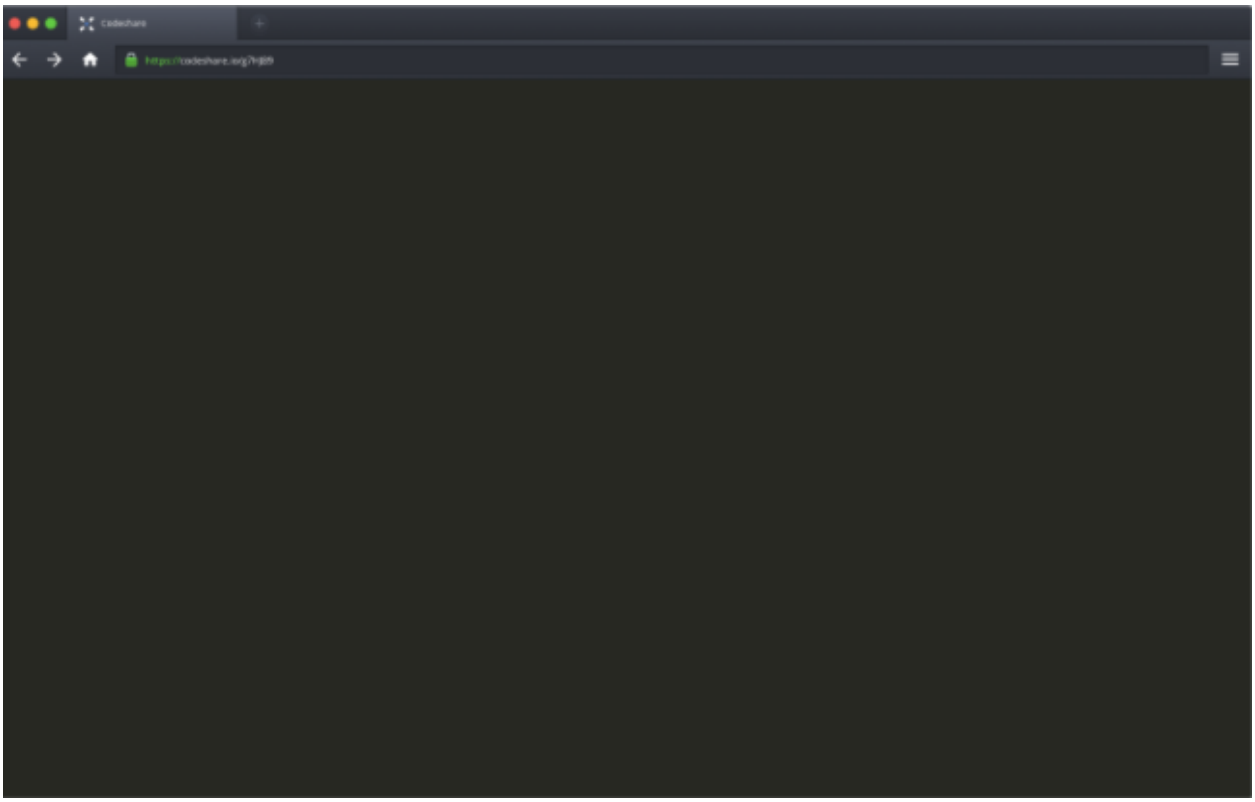
Credit Card Sample

https://codepen.io/quinlo/pen/YONMEa

2.0.14 Codeshare.io (real time code sharing)



https://codeshare.io/



2.0.15 Codebeautify.org (online data conversion tools)

https://codebeautify.org/

2.0.16 AsciiFlow.com (ASCII drawing tool)

//TODO//

2.0.17 Freemind (opensource mindmap application)

//TODO//

2.0.18 Wireflow (user flow designer)

//TODO//

2.0.19 PlantUML (software designer)

//TODO//

2.0.20 Drawio (drawing tool)

//TODO//

2.0.21 Putty (Remote Connection)

//TODO//

2.0.22 MobaXterm (Remote Connection)

//TODO//

2.0.23 Teamviewer (Remote Connection)

//TODO//

2.0.24 Paletton.com (Color Chooser)

//TODO//

2.1

2.1.1 Understand (Static Code Analysis)

//TODO//

2.1.2 JD Project (Java Decompiler)

//TODO//

2.1.3 Cutter (Multi-Platform Reverse Engineering Tool)

//TODO//

2.1.4 IDA Pro / Freeware (Native Reverse Engineering Tool)

//TODO//

2.1.5 Code Visualization (Python, C , C++ , Java)

https://pythontutor.com/

//TODO//

2.1.6 Assembly of C Code

https://godbolt.org/

//TODO//

2.1.7 Mobile Device Screen Sharing for Demo

GitHub - Genymobile/scrcpy: Display and control your Android device[25]

2.1.8 Travis-CI

- Travis.yml

//TODO//

[25] https://github.com/Genymobile/scrcpy

2.1.9 Jenkins

//TODO//

2.1.10 Valgrind

//TODO//

2.1.11 Docker

- `https://www.youtube.com/watch?v=nBwJm0onzeo&ab_channel=GaryExplains Dockerfile`
  - DockerHub
  - Docker Compose Yaml

- Dockerrun.aws.json (AWS)

//TODO//

2.1.12 Nuget Packages

//TODO//

2.1.13 Vim for Windows

- vim/vim-wim32-installer (windows vim installer)

//TODO//

2.1.14 SCV Cryptomanager

//TODO//

2.1.15 Addario CryptoBench

//TODO//

2.1.16 Raymond's MD5 & SHA Checksum Utility

//TODO//

2.1.17 SlavaSoft HashCalc

//TODO//

2.1.18 Portable PGP

//TODO//

2.1.19 Online Programming Envoriments

i. Hackerrank

ii. CS50 Sandbox

iii. Programiz C Online Complier

//TODO//


ToDo