

CE103 Algorithms and Programming I

Week-2

Fall Semester, 2021-2022

Download [DOC](#), [SLIDE](#), [PPTX](#)



Algorithm Basics

Flowgorithm

<http://www.flowgorithm.org/>

Pseudocode

[Pseudocode - Wikipedia](#)

[Pseudocode Examples](#)

[How to write a Pseudo Code? - GeeksforGeeks](#)

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](#)

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.

C / C++ Environment and Development

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

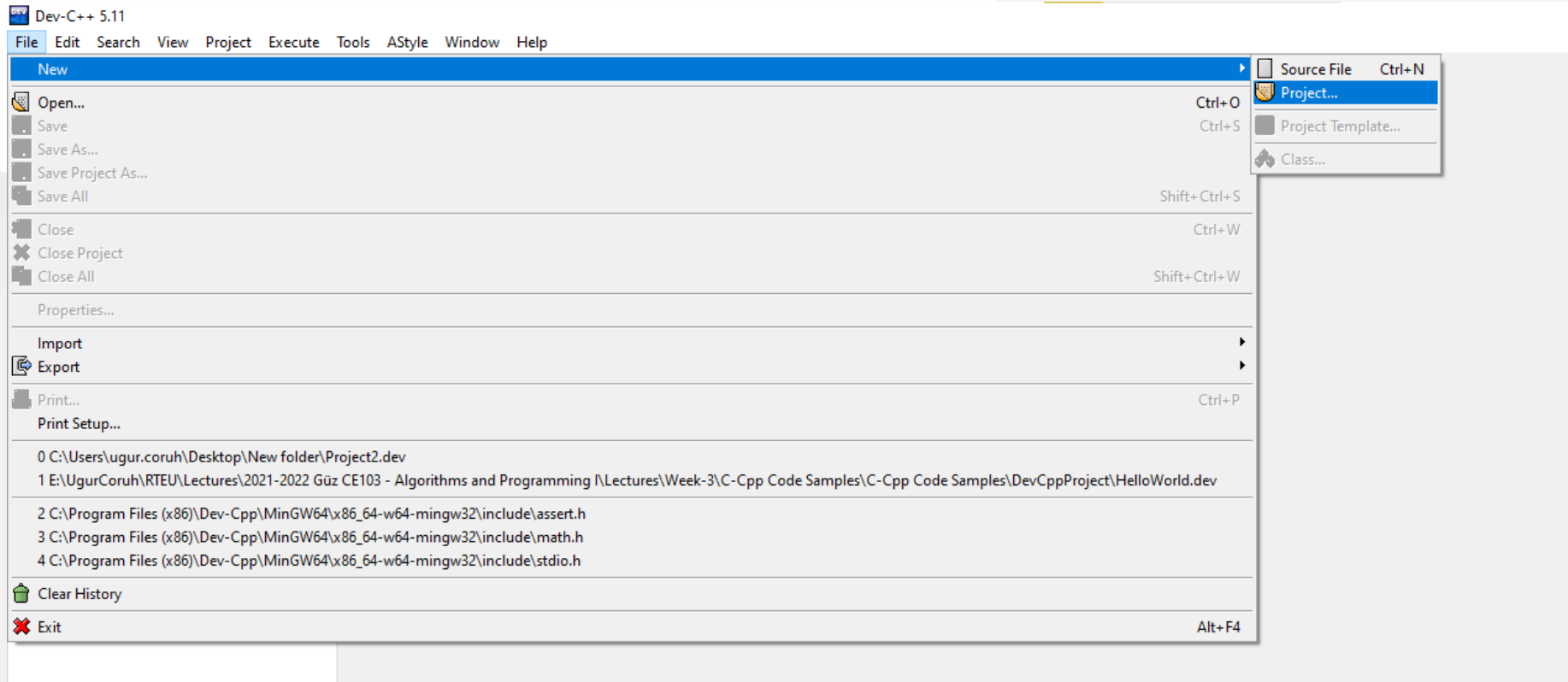
Download DevC++ IDE from following link

<https://www.bloodshed.net/>

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

Open DevC++ IDE for C Project Generation

Open File -> New -> Project



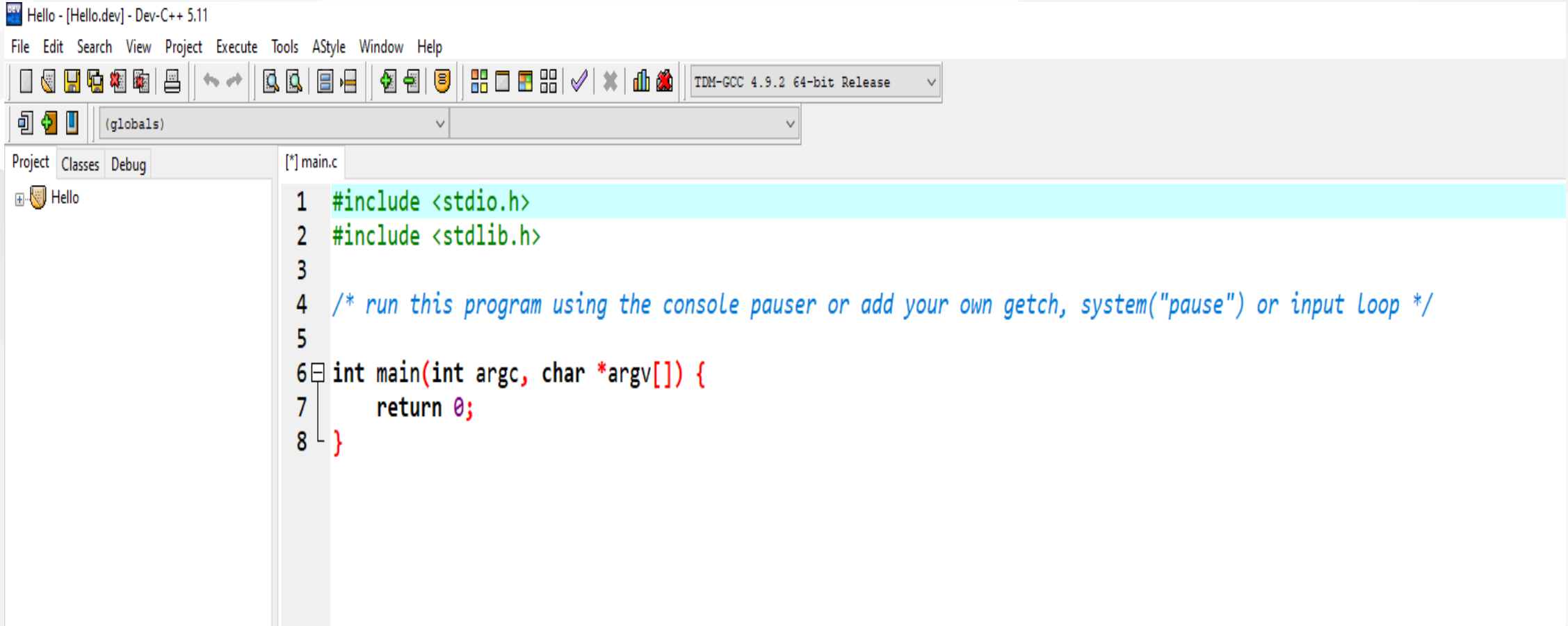
DevCpp (Install / Compile / Run / Debug) (3)

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.

DevCpp (Install / Compile / Run / Debug) (4)

You will see a sample main with empty body



The screenshot shows the Dev-C++ IDE interface. The title bar reads "Hello - [Hello.dev] - Dev-C++ 5.11". The menu bar includes File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, and Help. The toolbar contains various icons for file operations and execution. The compiler is set to "TDM-GCC 4.9.2 64-bit Release". The project explorer on the left shows a project named "Hello". The main editor window displays the following C++ code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* run this program using the console pauser or add your own getch, system("pause") or input loop */
5
6 int main(int argc, char *argv[]) {
7     return 0;
8 }
```

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

```
#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[]) {
    retAdd 0;
}
```

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

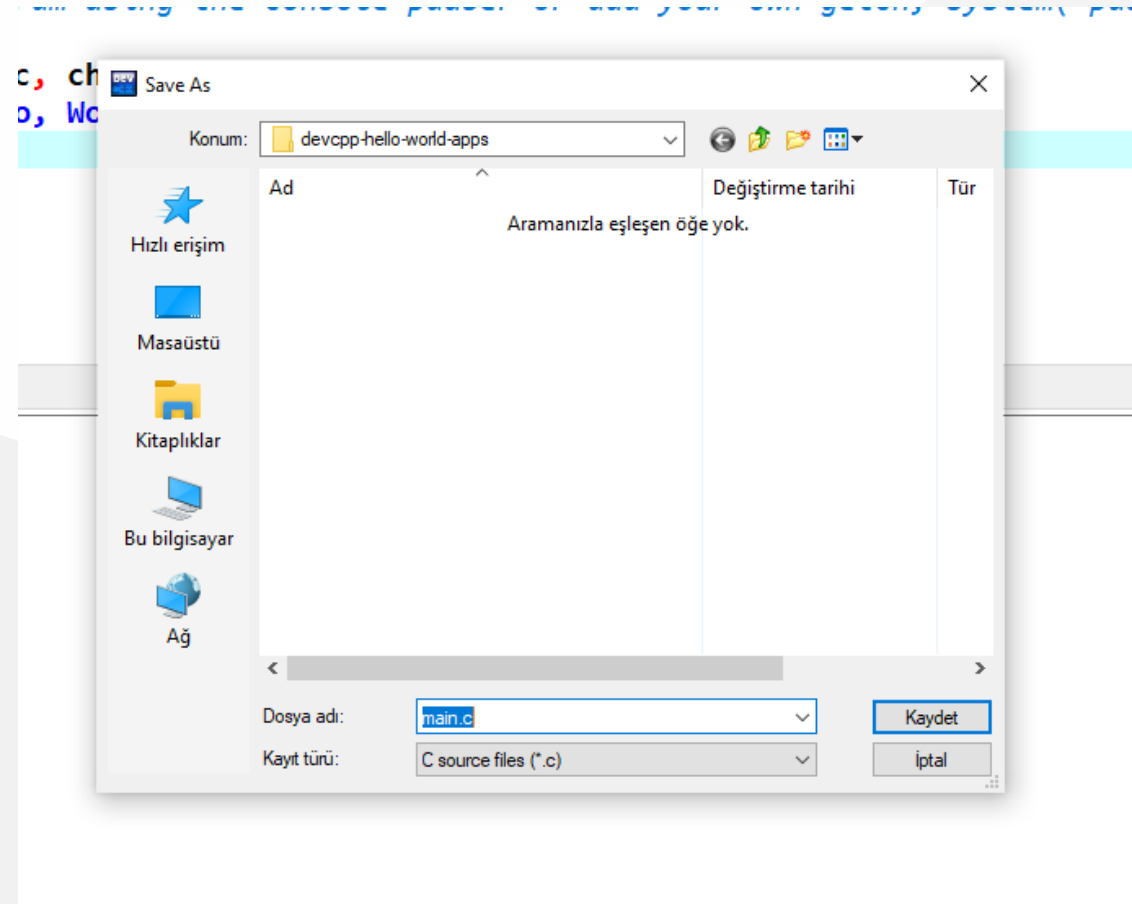
```
#include <stdio.h>
#include <stdlib.h>

/* 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!");
    getch();
    return 0;
}
```

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

Then save the file



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

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

The screenshot displays the Dev-C++ 5.11 interface. The 'Execute' menu is open, and 'Compile F5' is highlighted. The code editor contains the following C++ code:

```

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

/*
 * This program uses the console pauser or add your own getch, system("pause") or input Loop */

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

```

The console window shows the following output:

```

Compiling project changes...
-----
- Project Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Hello.dev
- 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\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Makefile.win

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 Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Makefile.win" all

gcc.exe -c main.o -o main.o -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/include" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/x86_64-w64-mingw32/include" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/lib64" -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib" -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/x86_64-w64-mingw32/lib" -static-libgcc

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Hello.exe
- Output Size: 128,103515625 KiB
- Compilation Time: 2,13s

```

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\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Hello.dev
- 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\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Makefile.win

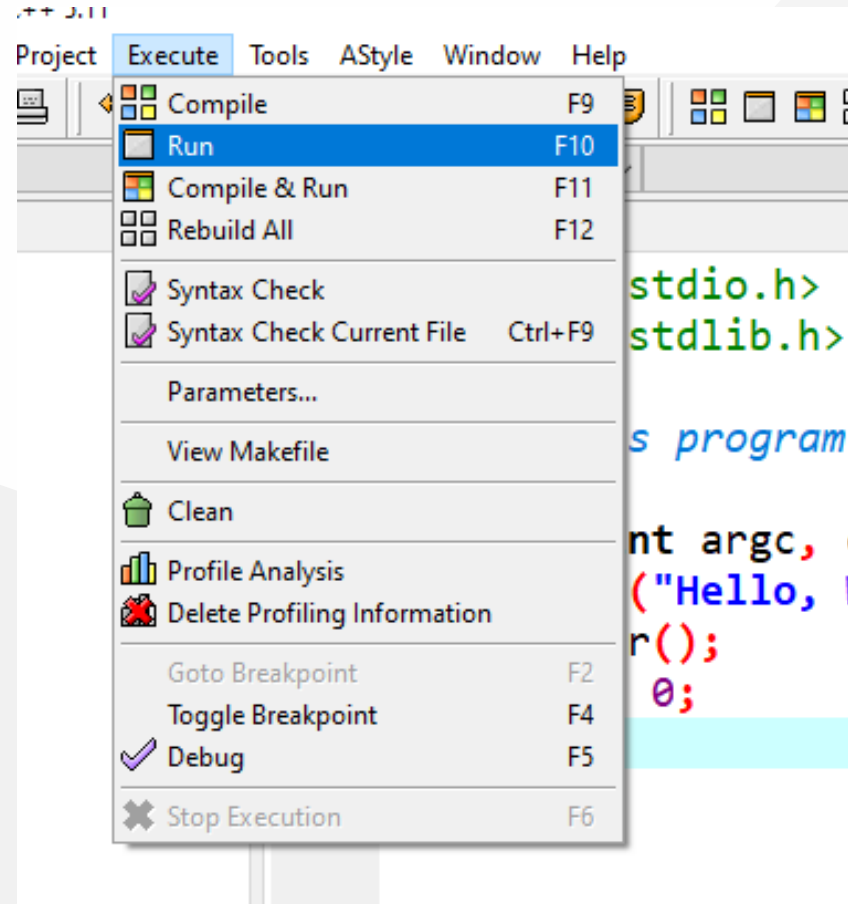
Processing makefile...
-----
- Makefile Processor: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\mingw32-make.exe
- Command: mingw32-make.exe -f "E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Makefile.win" all

gcc.exe -c main.c -o main.o -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/include" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/x86_64-w64-mingw32/include" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/4.9.2/include"
gcc.exe main.o -o Hello.exe -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib" -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/x86_64-w64-mingw32/lib" -static-libgcc

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Hello.exe
- Output Size: 128,103515625 KiB
- Compilation Time: 2,13s
```


DevCpp (Install / Compile / Run / Debug) (10)

Then you can run with Execute->Run F10 or Directly Compile&Run F11



DevCpp (Install / Compile / Run / Debug) (11)

for debugging operations, just change the code and add more statements as follow

```
#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[]) {

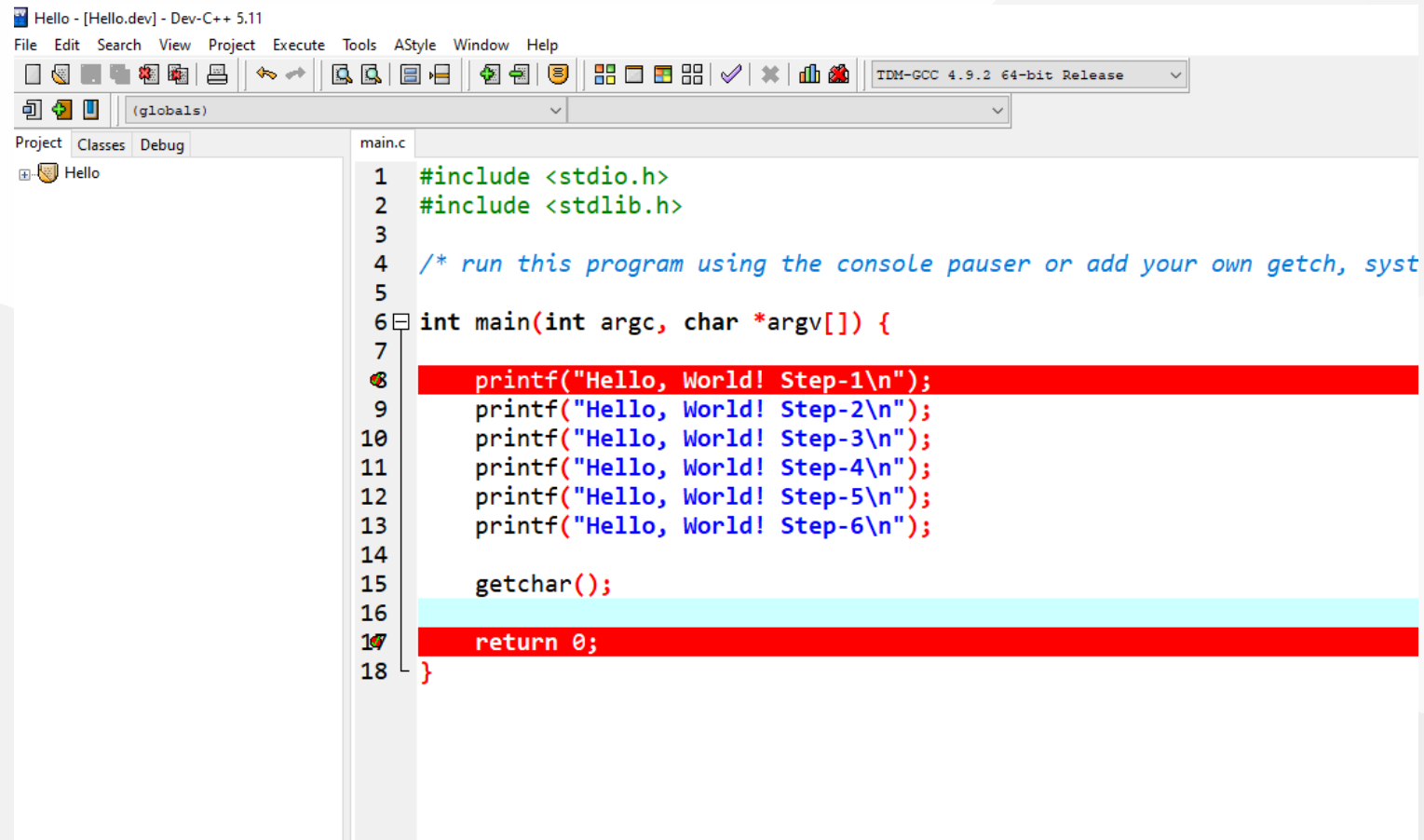
    printf("Hello, World! Step-1\n");
    printf("Hello, World! Step-2\n");
    printf("Hello, World! Step-3\n");
    printf("Hello, World! Step-4\n");
    printf("Hello, World! Step-5\n");
    printf("Hello, World! Step-6\n");

    getchar();

    return 0;
}
```

DevCpp (Install / Compile / Run / Debug) (12)

Click on line numbers and add breakpoints for debugger. This red point will be debugger stop points



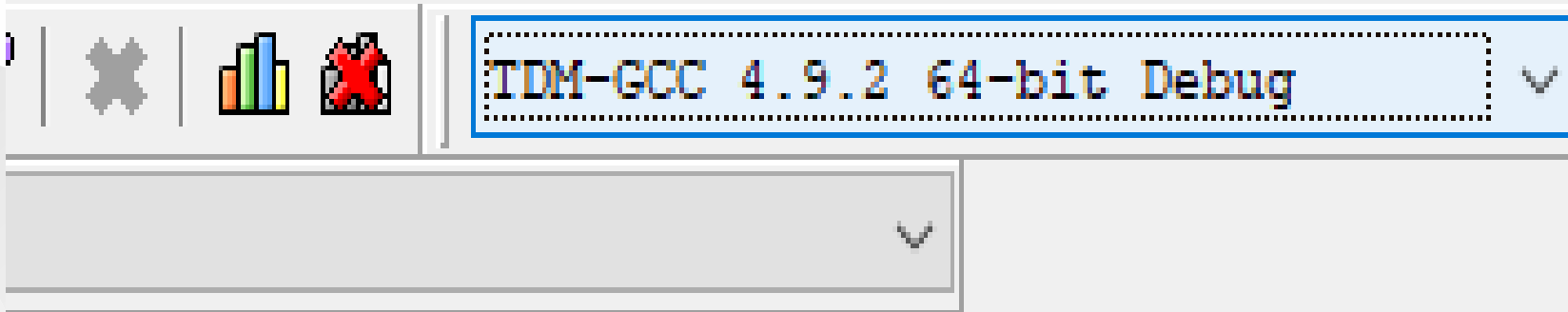
The screenshot shows the Dev-C++ IDE interface. The main window displays a C program named 'main.c' with the following code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* run this program using the console pauser or add your own getch, syst
5
6 int main(int argc, char *argv[]) {
7
8     printf("Hello, World! Step-1\n");
9     printf("Hello, World! Step-2\n");
10    printf("Hello, World! Step-3\n");
11    printf("Hello, World! Step-4\n");
12    printf("Hello, World! Step-5\n");
13    printf("Hello, World! Step-6\n");
14
15    getchar();
16
17    return 0;
18 }
```

Red vertical markers (breakpoints) are placed on the line numbers 8, 17, and 18. The line numbers 8 and 17 are highlighted in red, and the line number 18 is highlighted in light blue. The IDE title bar shows 'Hello - [Hello.dev] - Dev-C++ 5.11' and the compiler is set to 'TDM-GCC 4.9.2 64-bit Release'.

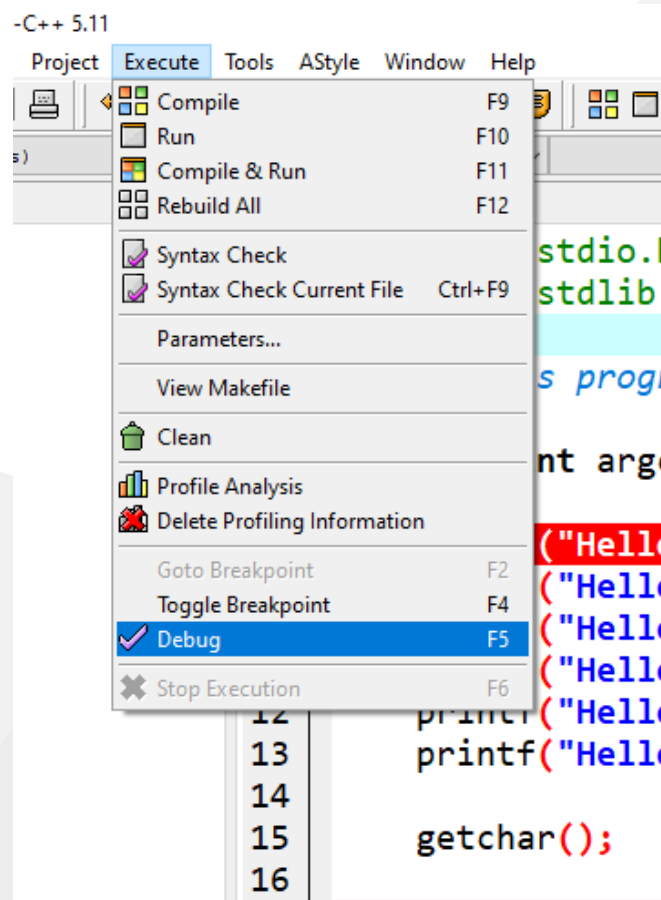
DevCpp (Install / Compile / Run / Debug) (13)

In the ,menu section select compiler with debug option



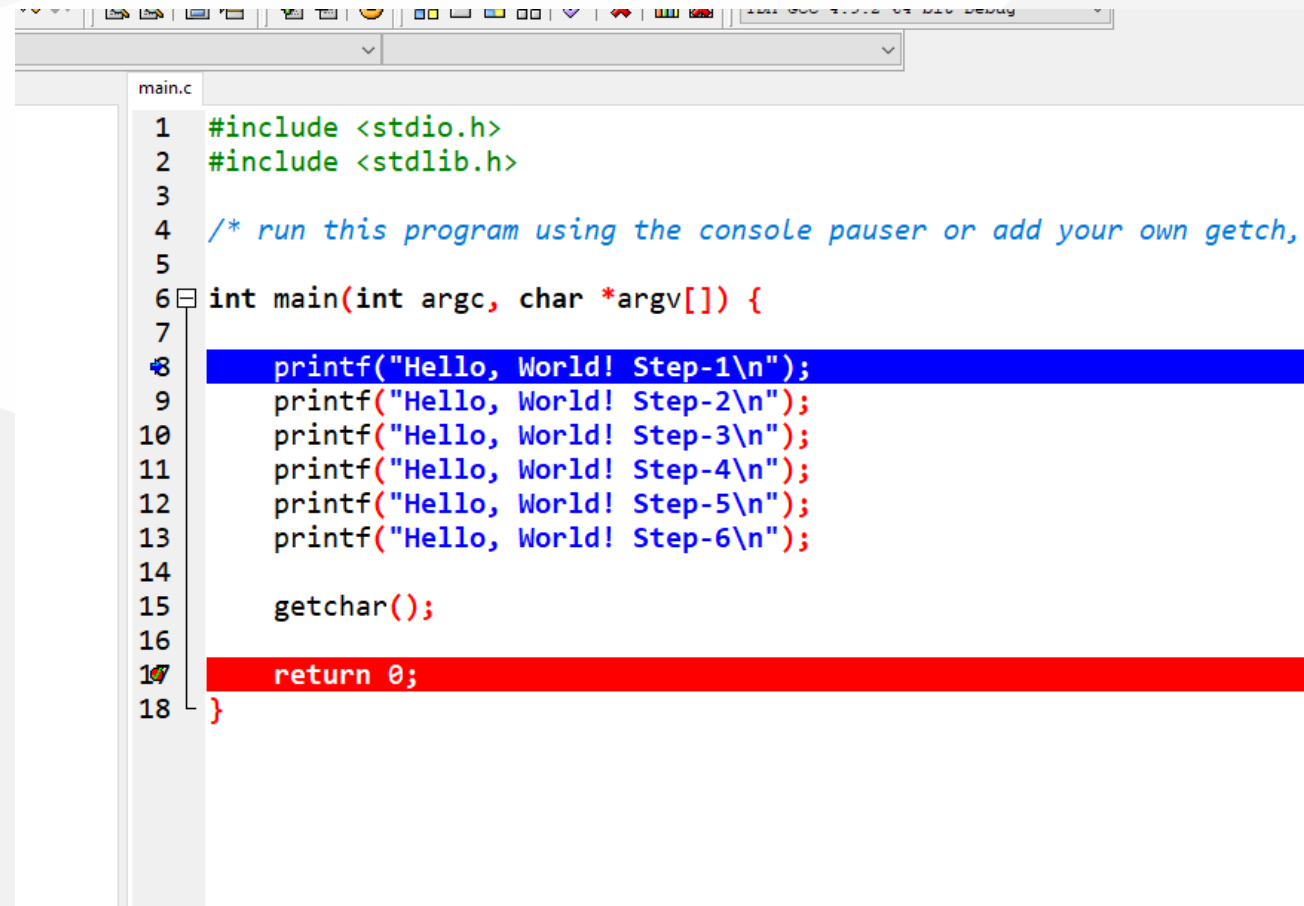
DevCpp (Install / Compile / Run / Debug) (14)

Compile application with debug setting and in Execute Section use Debug F5 to start debugging



DevCpp (Install / Compile / Run / Debug) (15)

Debugger will stop at breakpoint at the debug point (blue line)



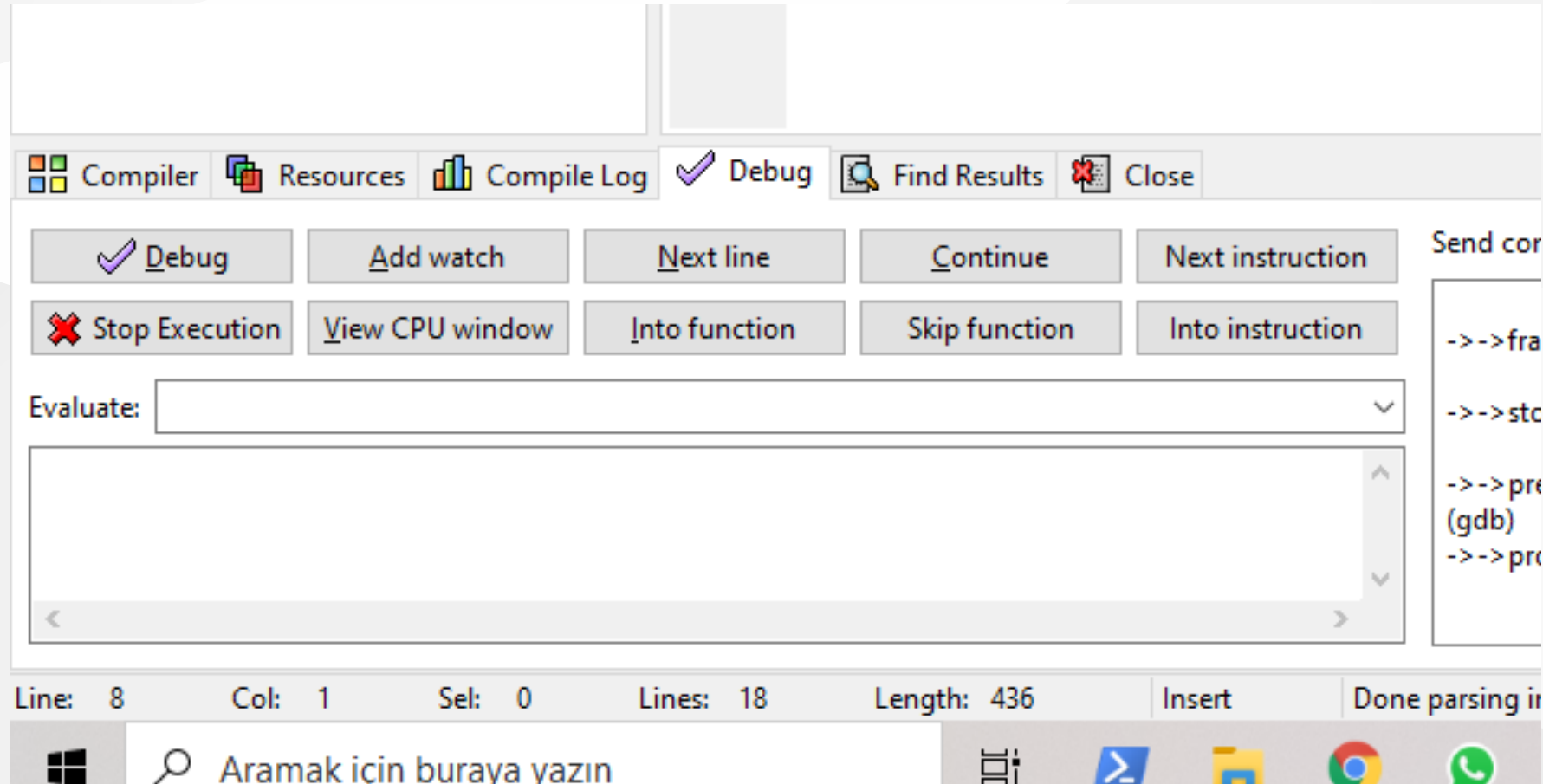
The screenshot shows the DevCpp IDE interface. The main window displays a C program named 'main.c'. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* run this program using the console pauser or add your own getch,
5
6 int main(int argc, char *argv[]) {
7
8 printf("Hello, World! Step-1\n");
9 printf("Hello, World! Step-2\n");
10 printf("Hello, World! Step-3\n");
11 printf("Hello, World! Step-4\n");
12 printf("Hello, World! Step-5\n");
13 printf("Hello, World! Step-6\n");
14
15 getchar();
16
17 return 0;
18 }
```

Line 8 is highlighted with a blue background, indicating a breakpoint. Line 17 is highlighted with a red background, indicating the end of the program. The IDE's toolbar and menu bar are visible at the top.

DevCpp (Install / Compile / Run / Debug) (16)

Moving to next statement can be done via control buttons or shortcuts



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).

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.

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.

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.

DevCpp (Install / Compile / Run / Debug) (21)

How do I debug using Dev-C++

Code Blocks (Install / Compile / Run / Debug) (1)

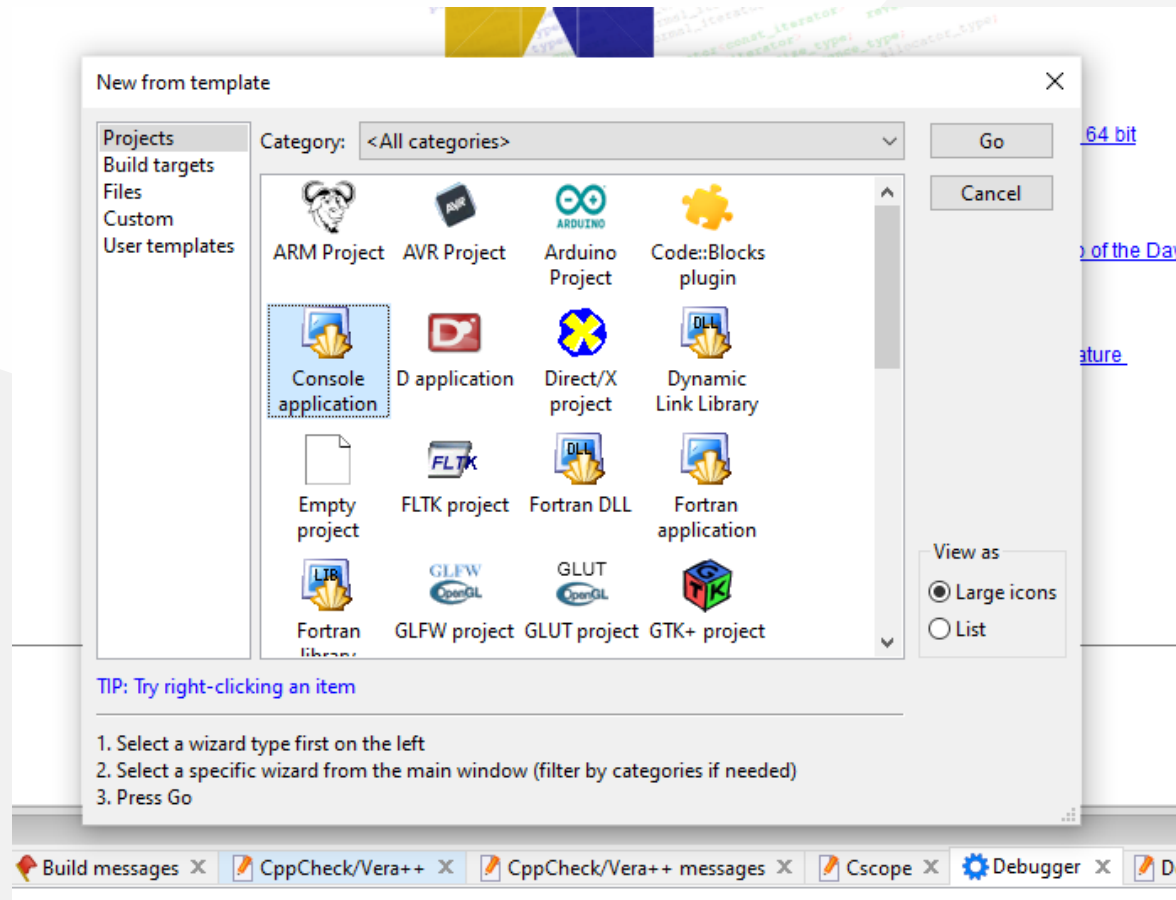
Download Code Blocks from the following link

[Binary releases - Code::Blocks](#)

Code Blocks (Install / Compile / Run / Debug) (2)

Open Code Blocks and

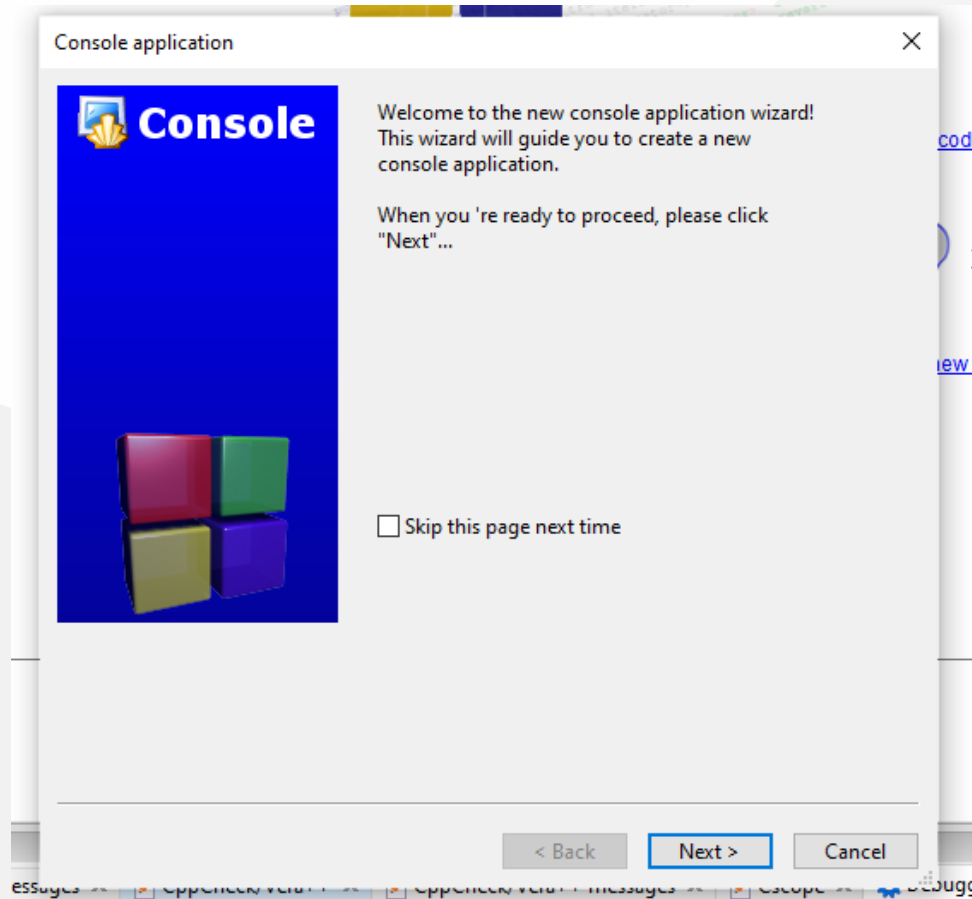
Select File->New->Project



Code Blocks (Install / Compile / Run / Debug) (3)

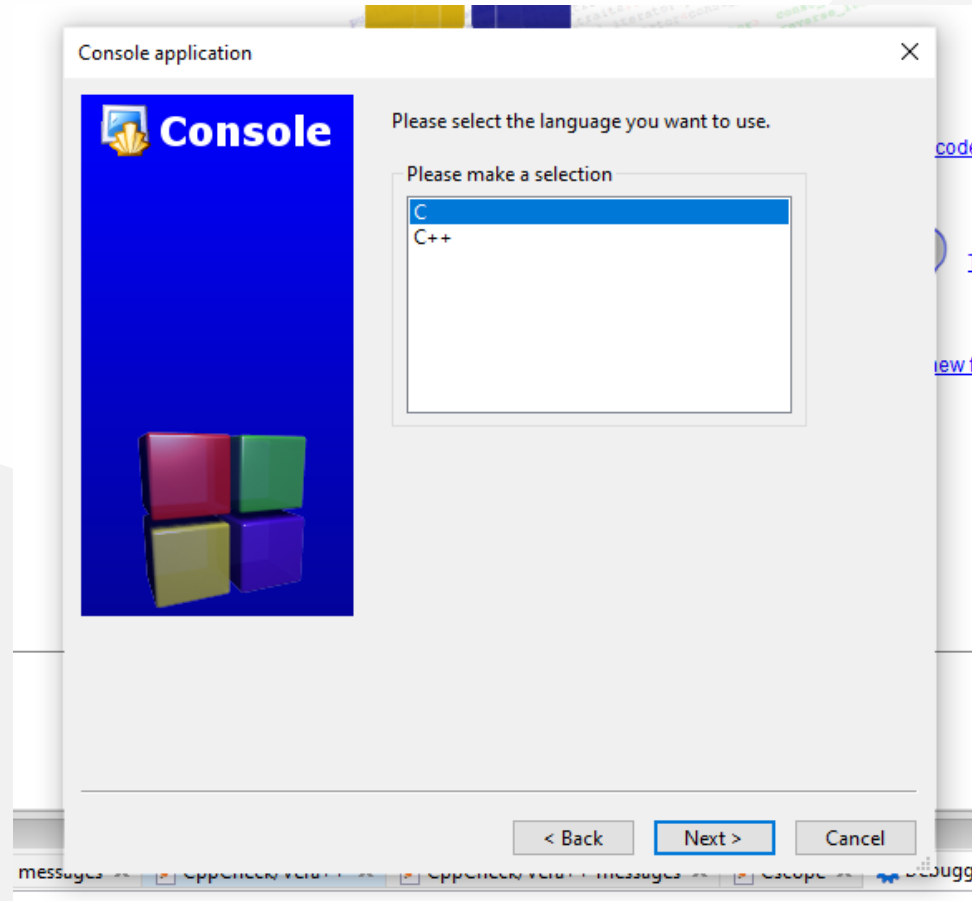
Select Console Application

Click Next from Opening Window



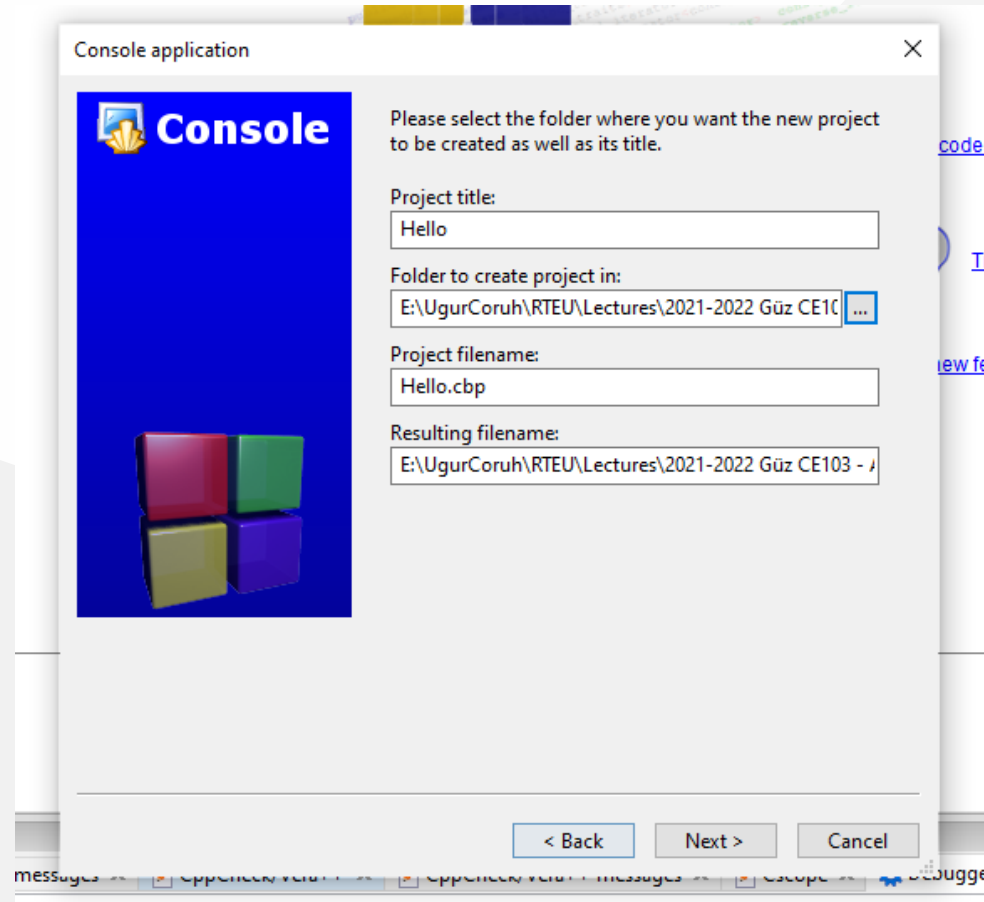
Code Blocks (Install / Compile / Run / Debug) (4)

Select C for Sample Project



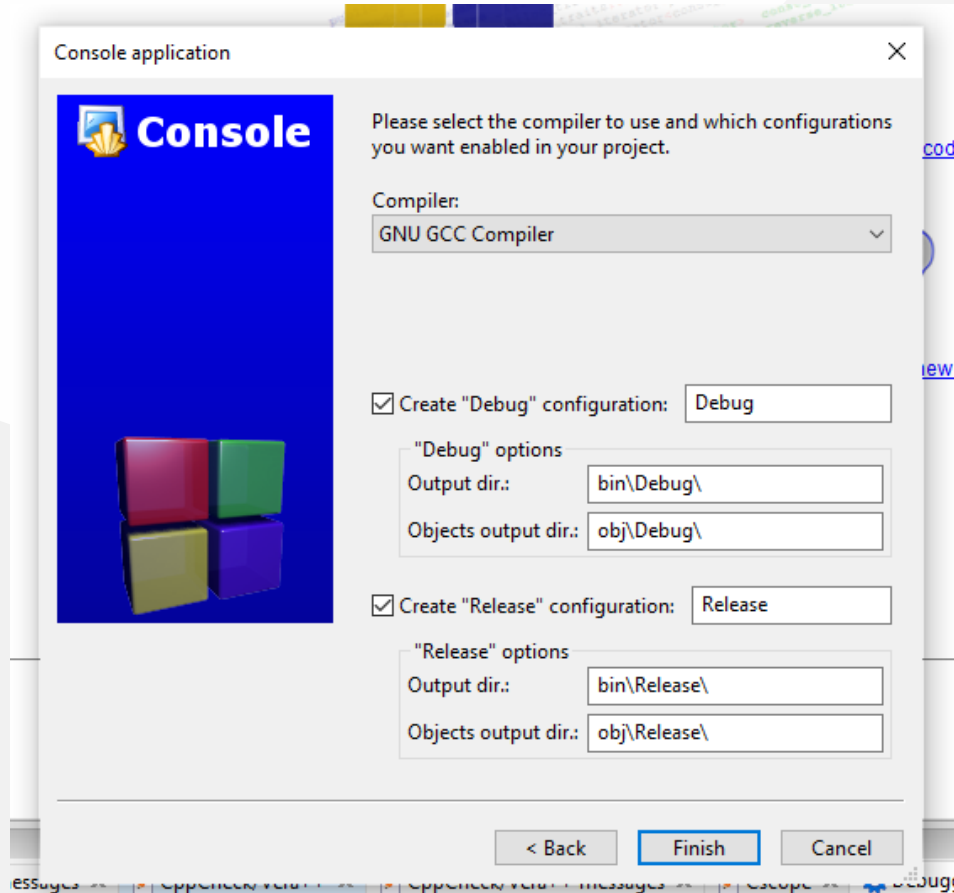
Code Blocks (Install / Compile / Run / Debug) (5)

Write a project name and title also set a project folder



Code Blocks (Install / Compile / Run / Debug) (6)

Select compiler for this project we selected GCC but you can select C compilers from list. Set Debug and Release executable output folders.



Code Blocks (Install / Compile / Run / Debug) (7)

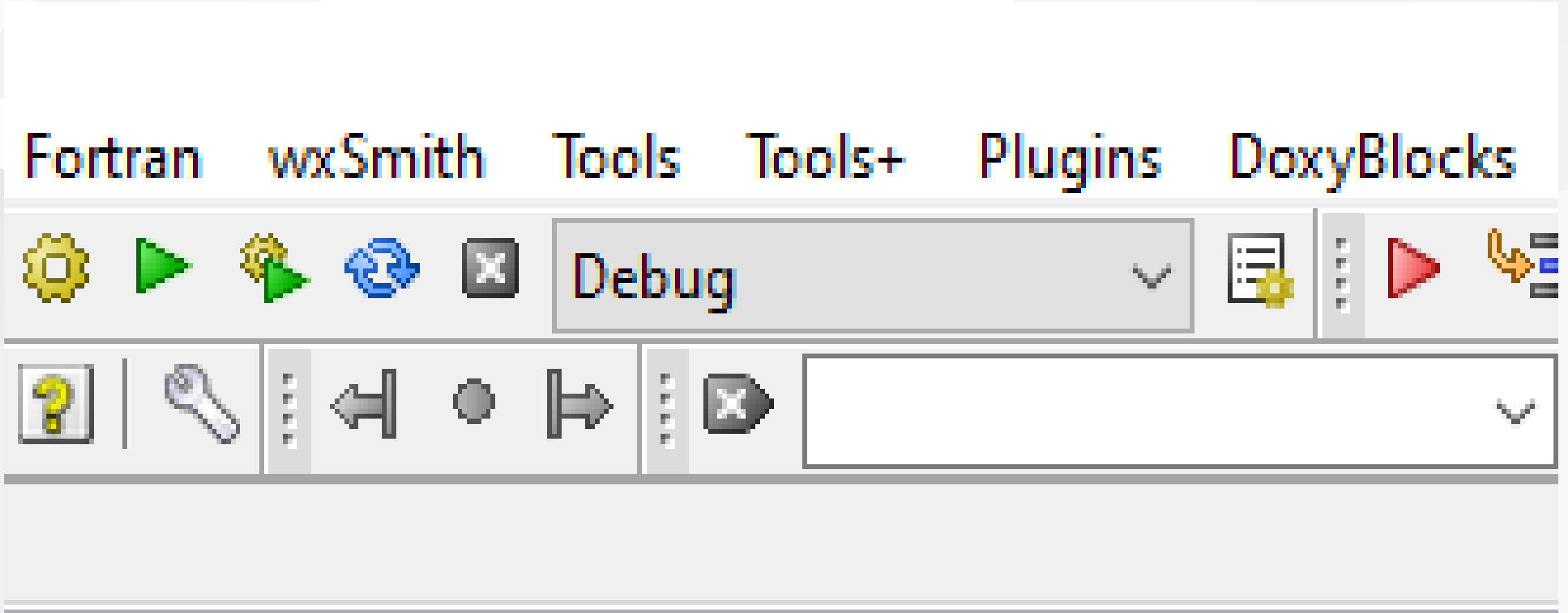
After this wizard you will have the following code

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

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

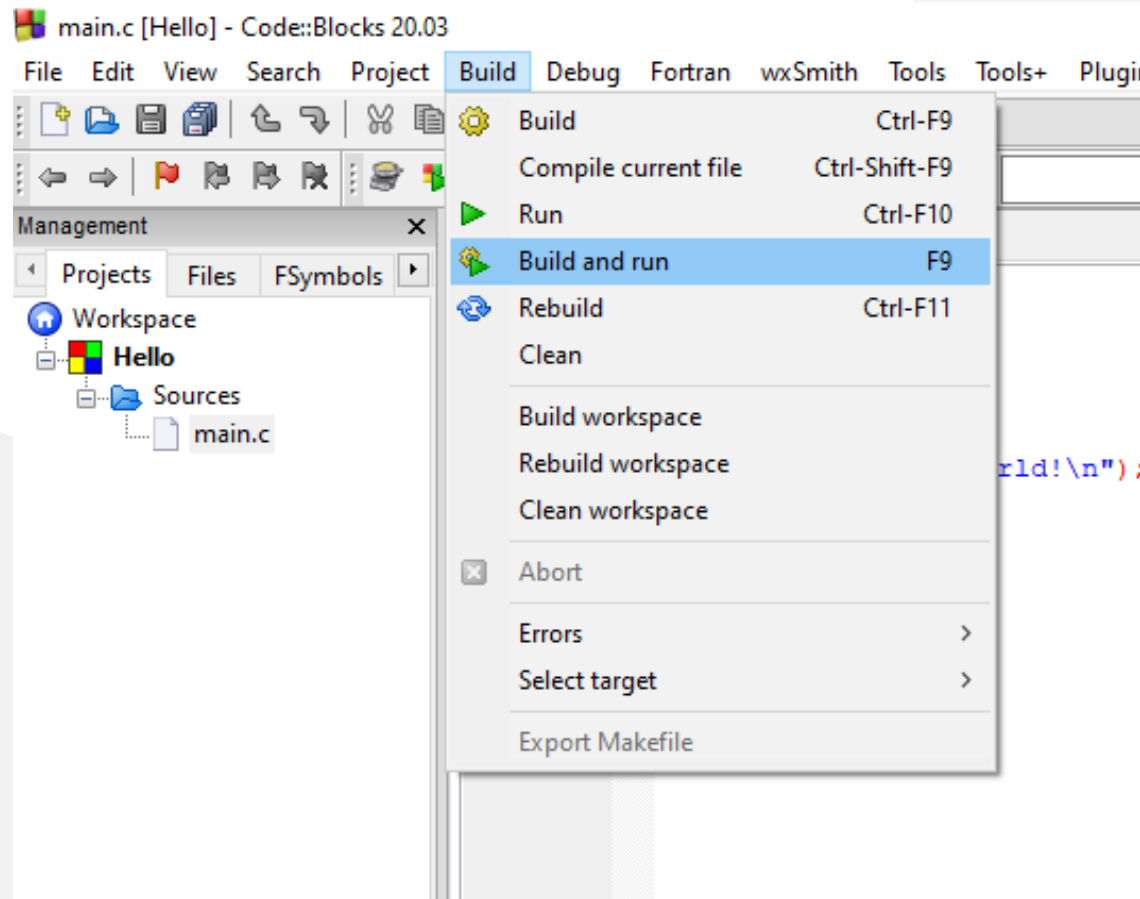
Code Blocks (Install / Compile / Run / Debug) (8)

Select Debug Build from menu



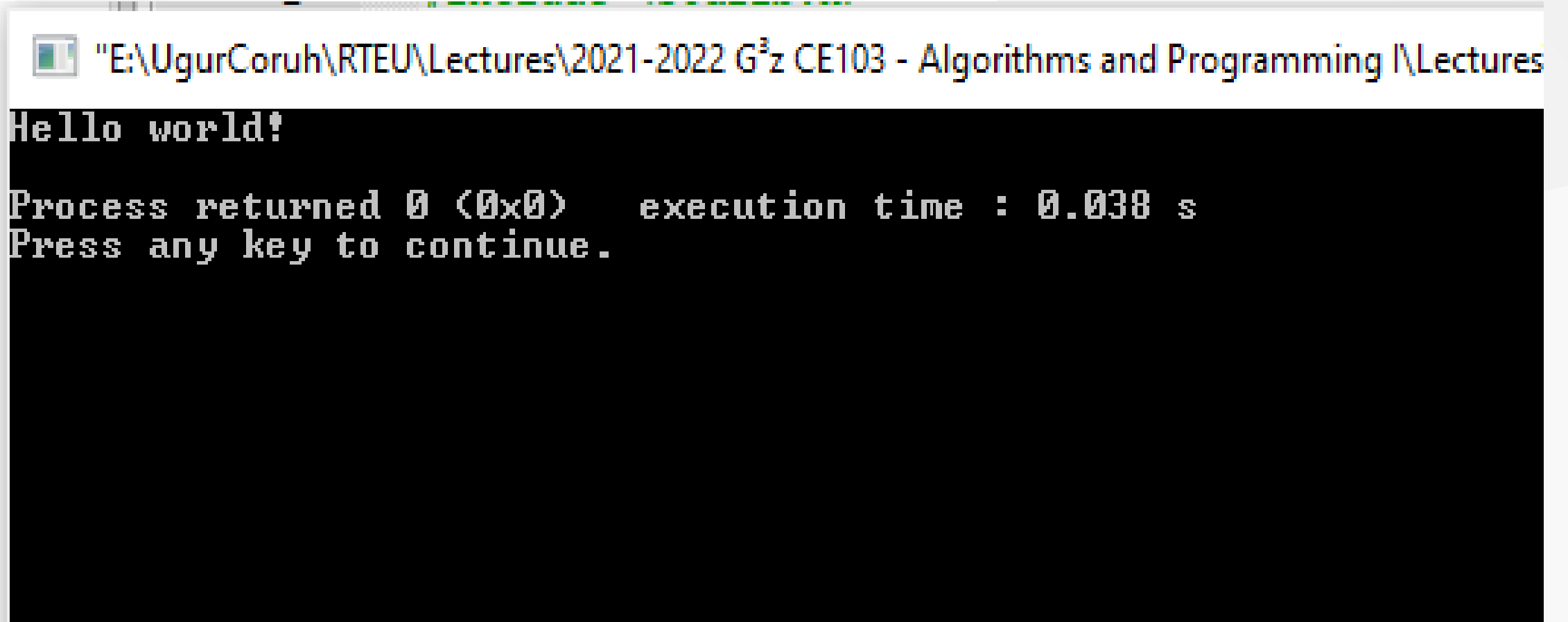
Code Blocks (Install / Compile / Run / Debug) (9)

Run with Build and Run F9



Code Blocks (Install / Compile / Run / Debug) (10)

You should see the following output



The screenshot shows a Code Blocks IDE window with the following content:

```
"E:\UgurCoruh\RTEU\Lectures\2021-2022 G3z CE103 - Algorithms and Programming I\Lectures  
Hello world!  
Process returned 0 (0x0)   execution time : 0.038 s  
Press any key to continue.
```

Code Blocks (Install / Compile / Run / Debug) (11)

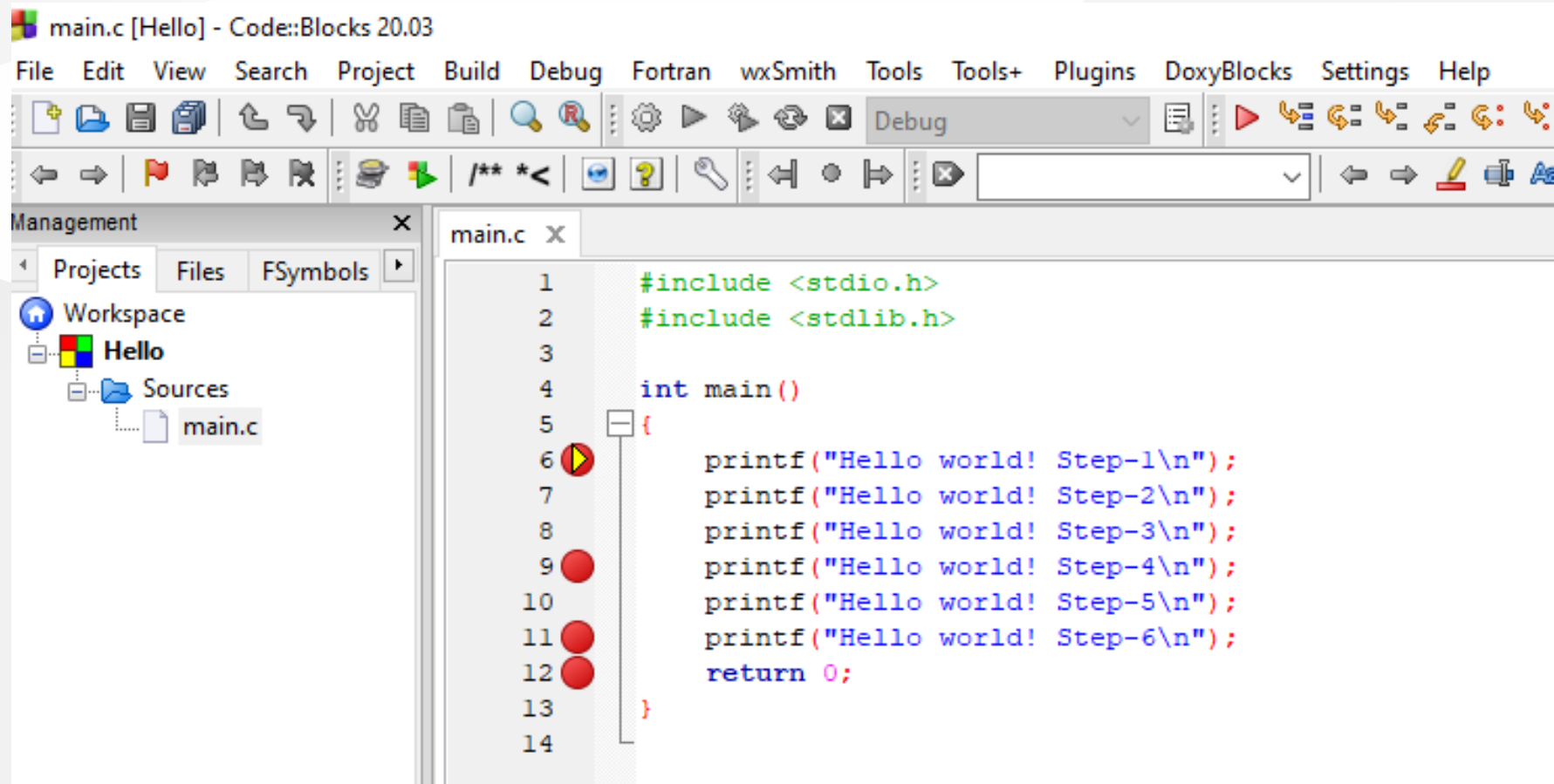
Add the following lines to your source code for debugging

```
#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;
}
```

Code Blocks (Install / Compile / Run / Debug) (12)

and add break points with F5 or mouse click



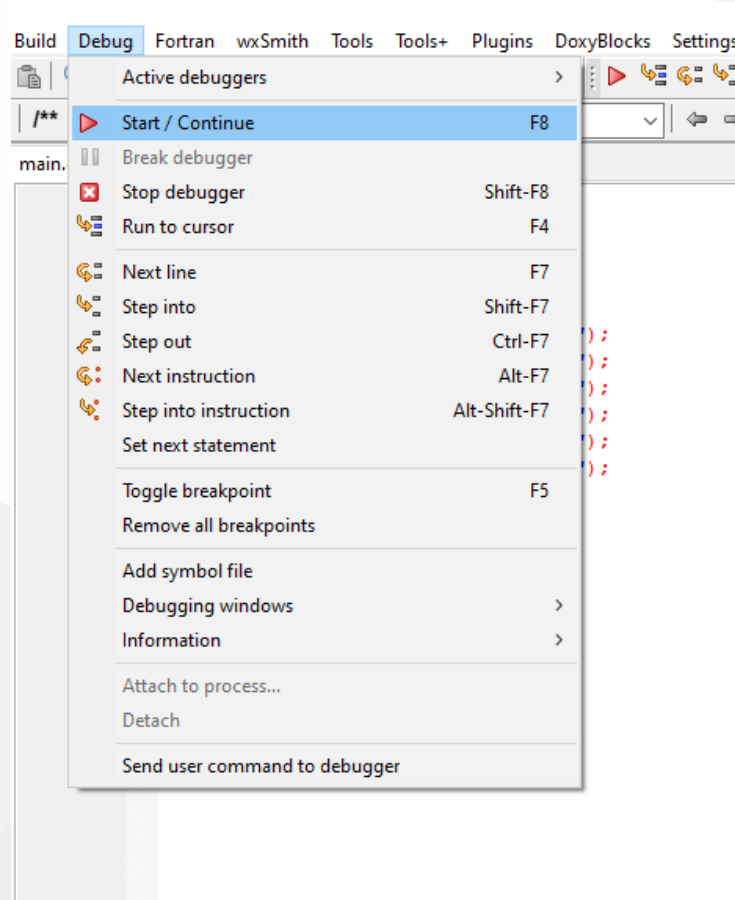
The screenshot shows the Code::Blocks IDE interface. The main window displays a C program named `main.c` with the following code:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world! Step-1\n");
7      printf("Hello world! Step-2\n");
8      printf("Hello world! Step-3\n");
9      printf("Hello world! Step-4\n");
10     printf("Hello world! Step-5\n");
11     printf("Hello world! Step-6\n");
12     return 0;
13 }
14
```

The code is displayed in a window titled `main.c`. The left sidebar shows the project structure with a workspace named `Hello` containing a `Sources` folder and the `main.c` file. The `Debug` menu is open, and the `Breakpoints` sub-menu is visible. A yellow play button icon is visible on line 6, indicating a breakpoint has been set. Red circular icons are visible on lines 9, 10, 11, and 12, indicating additional breakpoints.

Code Blocks (Install / Compile / Run / Debug) (13)

select Debug->Start/Continue to start debugger



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
```

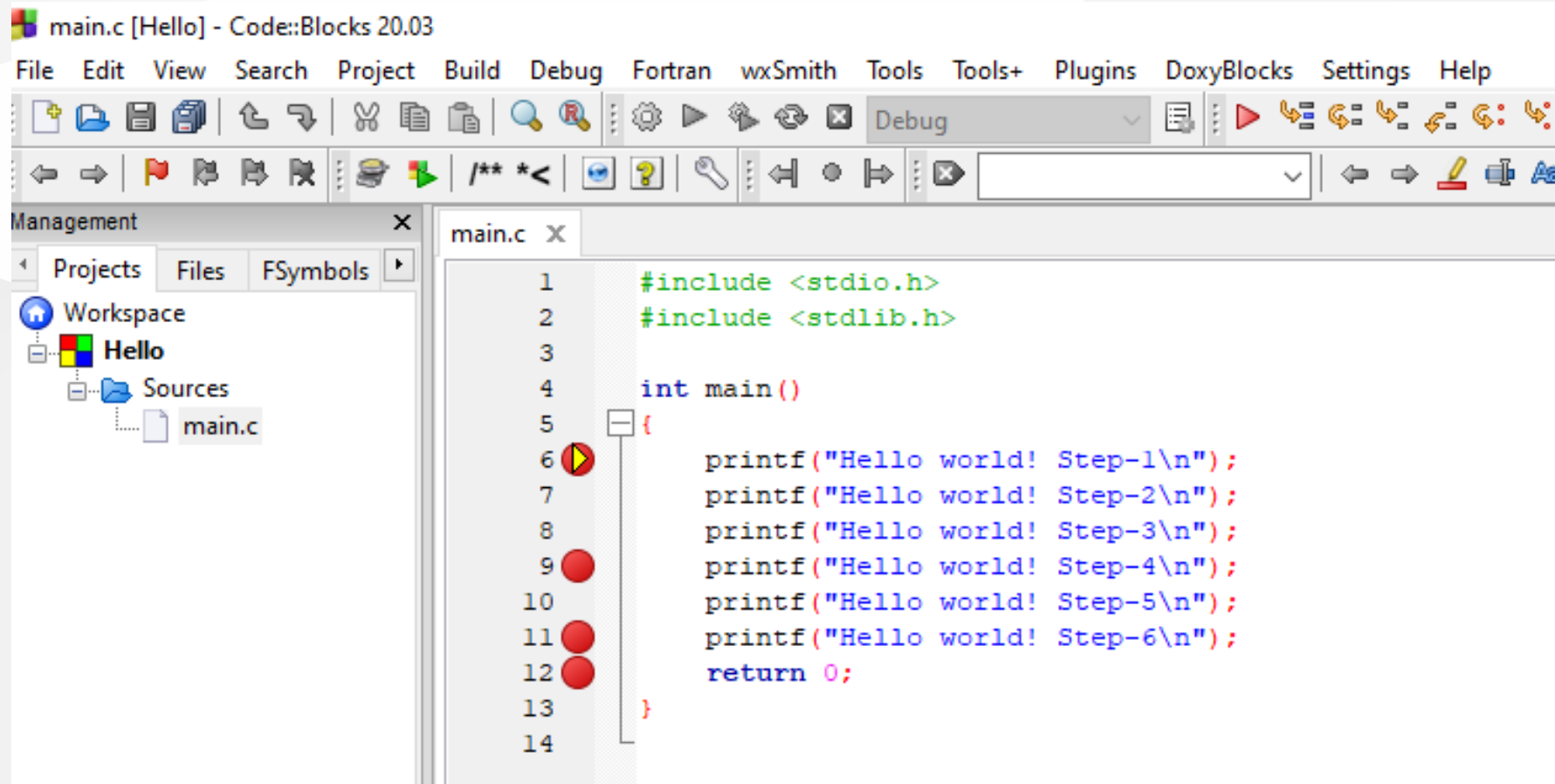
```
Debugger name and version: GNU gdb (GDB) 8.1
```

```
Starting the debuggee failed: No executable specified, use `target exec`.
```

```
Debugger finished with status 0
```

Code Blocks (Install / Compile / Run / Debug) (15)

You will see the following yellow pointer for debugger



The screenshot shows the Code::Blocks IDE interface. The main window displays a C program named 'main.c' with the following code:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6  printf("Hello world! Step-1\n");
7  printf("Hello world! Step-2\n");
8  printf("Hello world! Step-3\n");
9  printf("Hello world! Step-4\n");
10 printf("Hello world! Step-5\n");
11 printf("Hello world! Step-6\n");
12 return 0;
13 }
14
```

The code is displayed in a window titled 'main.c'. The left sidebar shows a project named 'Hello' with a subfolder 'Sources' containing 'main.c'. The top menu bar includes 'File', 'Edit', 'View', 'Search', 'Project', 'Build', 'Debug', 'Fortran', 'wxSmith', 'Tools', 'Tools+', 'Plugins', 'DoxyBlocks', 'Settings', and 'Help'. The 'Debug' menu is currently selected, and a yellow pointer is visible on line 6 of the code.

Code Blocks (Install / Compile / Run / Debug) (16)

You can use the following menu or shortcuts for step-by-step debugging.



GCC/G++ Compiler (MinGW) / Clang-cl (LLVM) (1)

Download and install MinGW or LLVM compiler (if you downloaded then skip this step)

MinGW installer (clang)

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

LLVM installer (gcc / g++)

[Download LLVM releases](#)

Also use the following notes

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

GCC/G++ Compiler (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
```

GCC/G++ Compiler (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

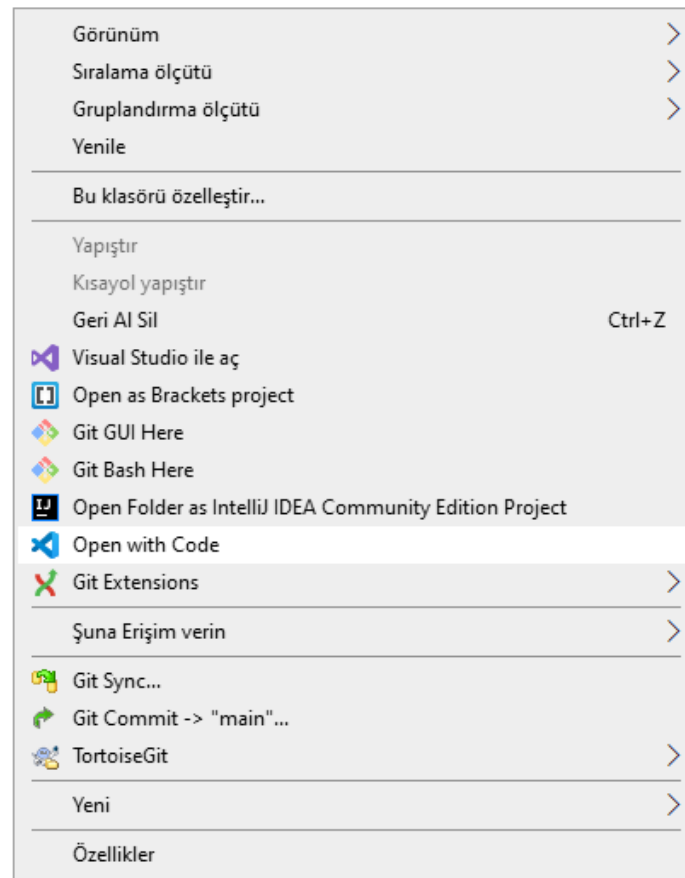
VSCode (Install / Compile / Run / Debug) (1)

Download Visual Studio Code from the following link

[Download Visual Studio Code - Mac, Linux, Windows](#)

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



VSCode (Install / Compile / Run / Debug) (3)

or enter the folder via console

```
Güz C E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-  
I\Week-2\vscode-hello-world-apps\C-clang>code .
```

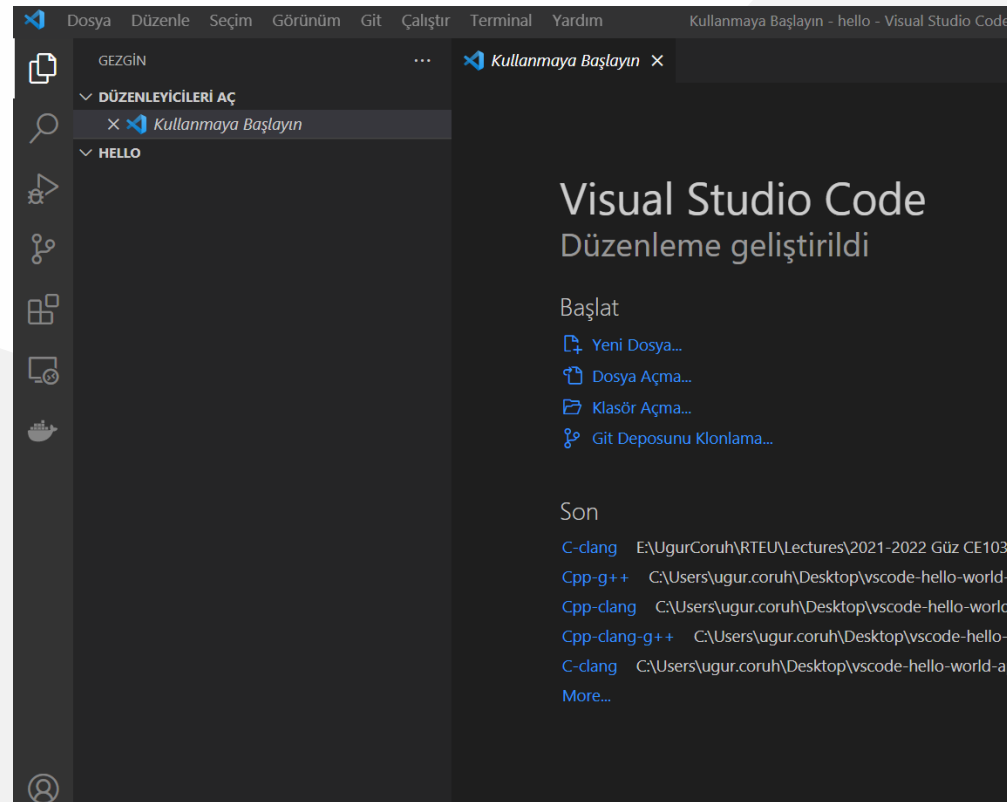
write

```
code .
```

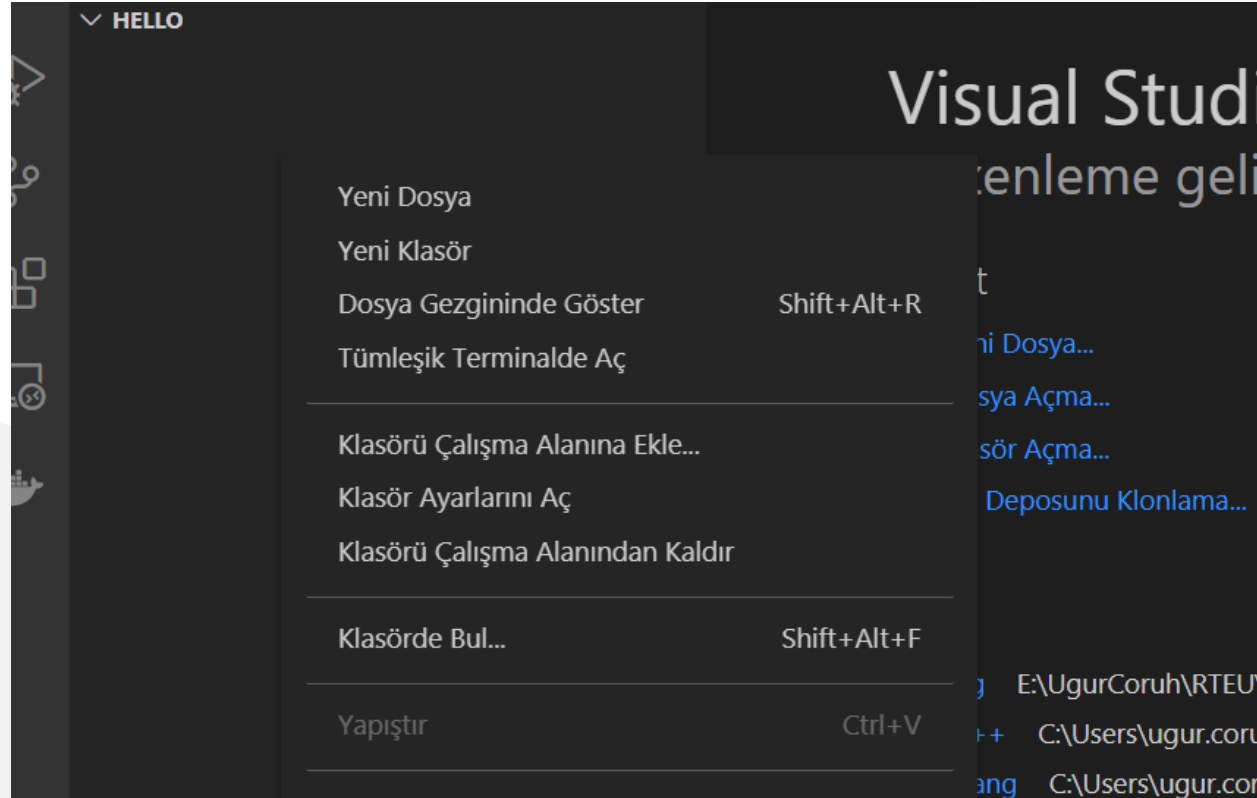
VSCode (Install / Compile / Run / Debug) (4)

This will open vscode for current folder . dot present current folder.

You will see a empty folder in the right window



VSCoDe (Install / Compile / Run / Debug) (5)



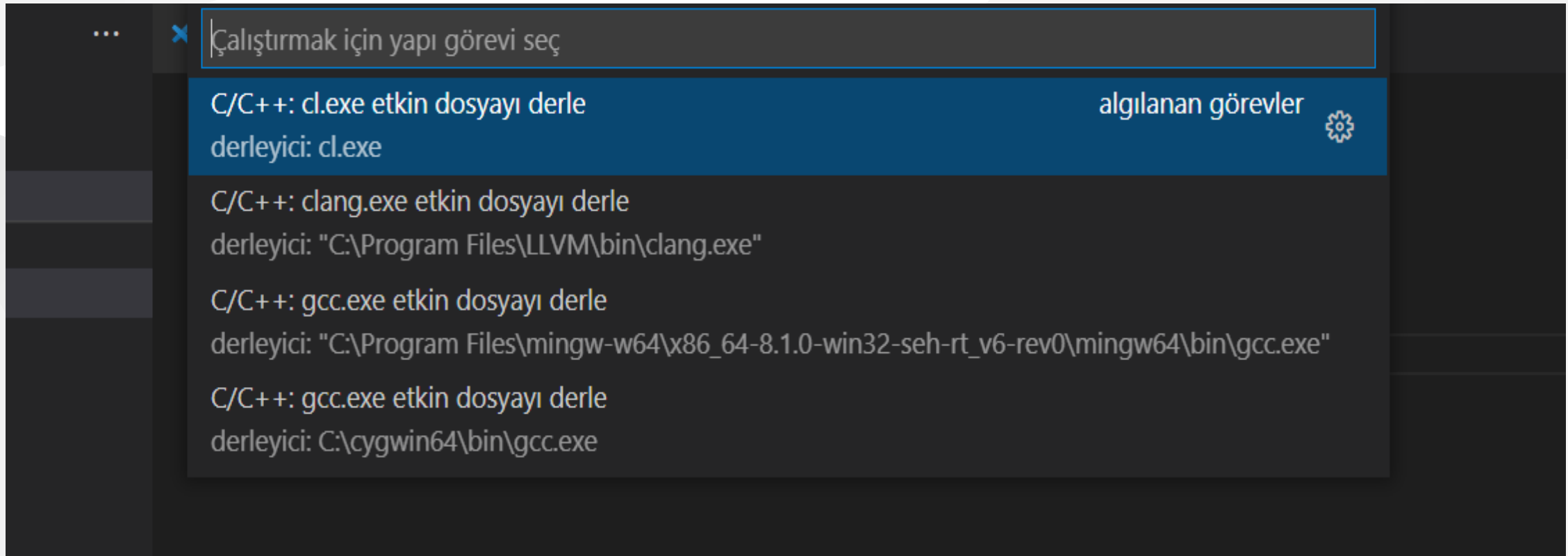
VSCoDe (Install / Compile / Run / Debug) (6)

Create a hello.c file and write following content

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

VSCoDe (Install / Compile / Run / Debug) (7)

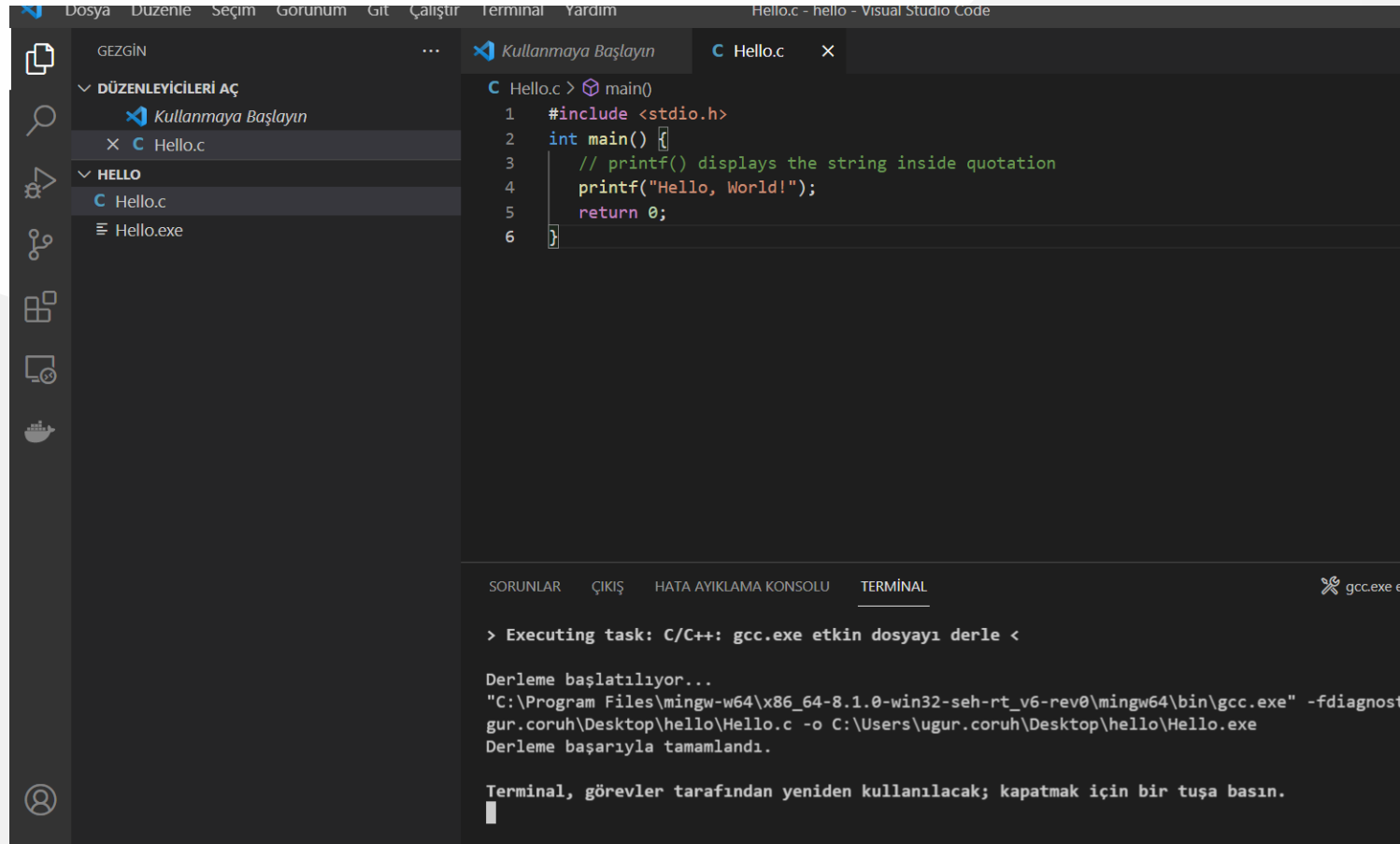
use CTRL+SHIFT+B (you should be on source code section) to build file



VSCoDe (Install / Compile / Run / Debug) (8)

Select GCC or CLANG for this sample we can use GCC

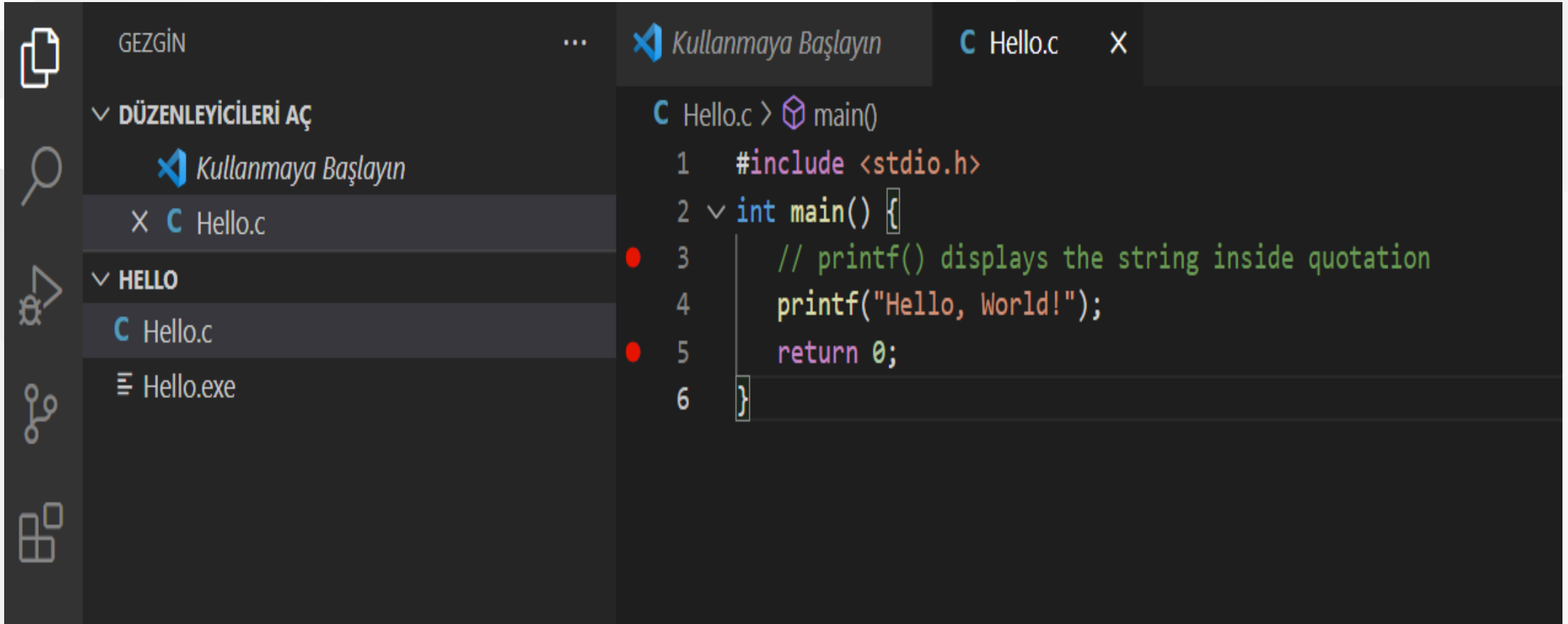
You will see output generated Hello.exe



```
gözetim
Düzenle Seçim Görünüm Git Çalıştır Terminal Yardım Hello.c - hello - Visual Studio Code
Kullanmaya Başlayın C Hello.c x
DÜZENLEYİCİLERİ AÇ
Kullanmaya Başlayın
Hello.c
HELLO
Hello.c
Hello.exe
C Hello.c > main()
1 #include <stdio.h>
2 int main() {
3 // printf() displays the string inside quotation
4 printf("Hello, World!");
5 return 0;
6 }
SORUNLAR ÇIKIŞ HATA AYIKLAMA KONSOLU TERMINAL gcc.exe e
> Executing task: C/C++: gcc.exe etkin dosyayı derle <
Derleme başlatılıyor...
"C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw64\bin\gcc.exe" -fdiagnost
gur.coruh\Desktop\hello\Hello.c -o C:\Users\ugur.coruh\Desktop\hello\Hello.exe
Derleme başarıyla tamamlandı.
Terminal, görevler tarafından yeniden kullanılacak; kapatmak için bir tuşa basın.
```

VSCoDe (Install / Compile / Run / Debug) (9)

for debugging just put breakpoint and build again

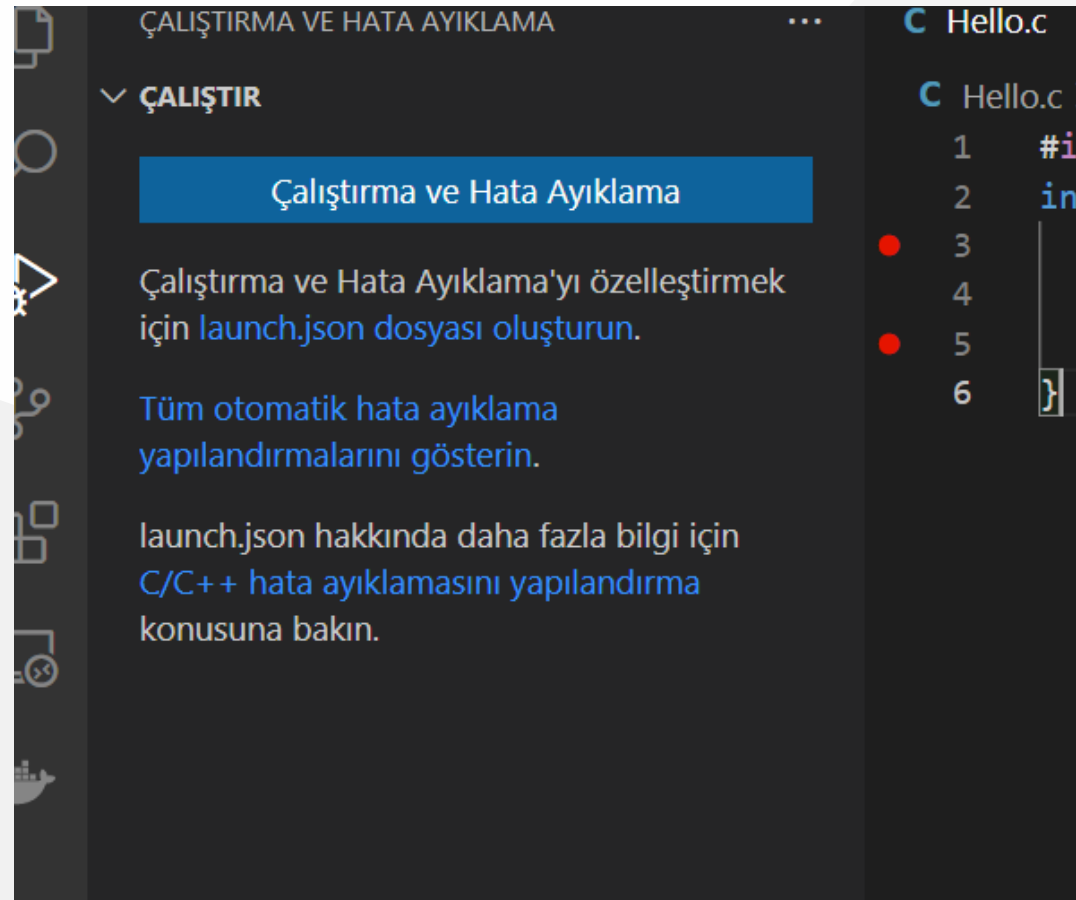


The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'GEZGIN' with a folder 'DÜZENLEYİCİLERİ AÇ' containing a file 'Hello.c'. Below it is a folder 'HELLO' containing 'Hello.c' and 'Hello.exe'. The main editor window shows the code for 'Hello.c' with a breakpoint (red dot) on line 3. The code is as follows:

```
C Hello.c > main()
1  #include <stdio.h>
2  int main() {
3      // printf() displays the string inside quotation
4      printf("Hello, World!");
5      return 0;
6  }
```

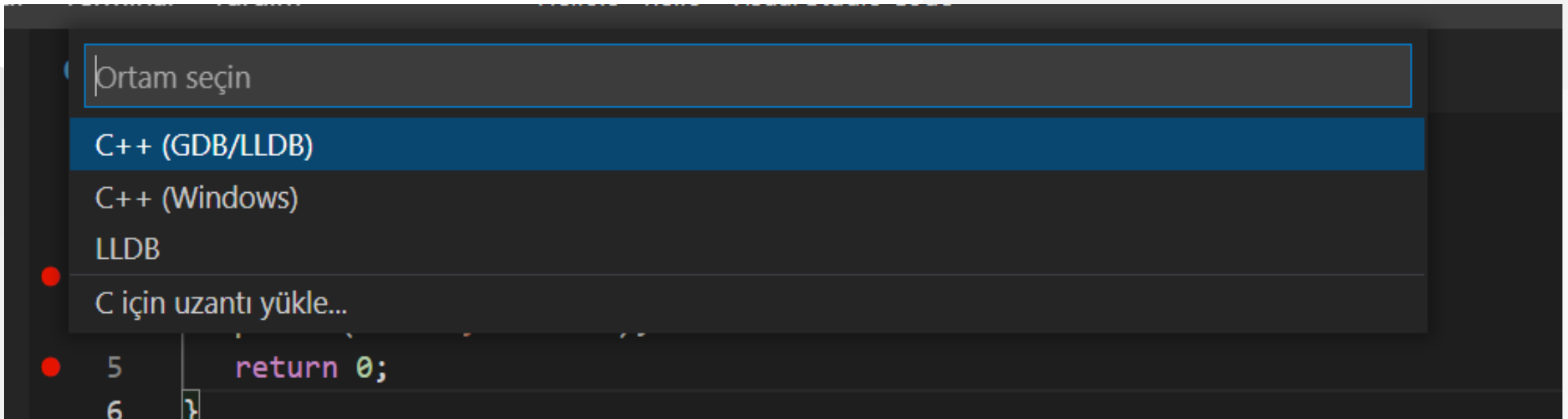

VSCode (Install / Compile / Run / Debug) (10)

after build for debug press CTRL+SHIFT+D (you should be on source code section)and in the right window select create launch.json



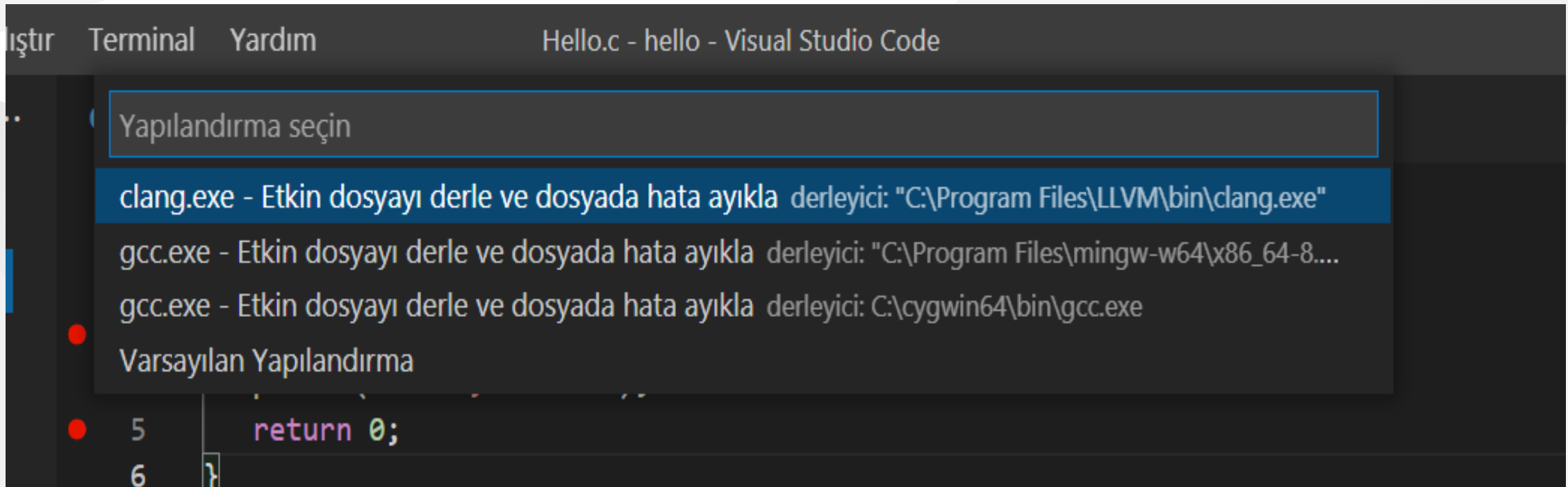
VSCoDe (Install / Compile / Run / Debug) (11)

from opening window select C++ GDB/LLDB



VSCoDe (Install / Compile / Run / Debug) (12)

from next opening menu select mingw-w64 gcc.exe



The screenshot shows the Visual Studio Code interface with the 'Build and Run' menu open. The menu items are:

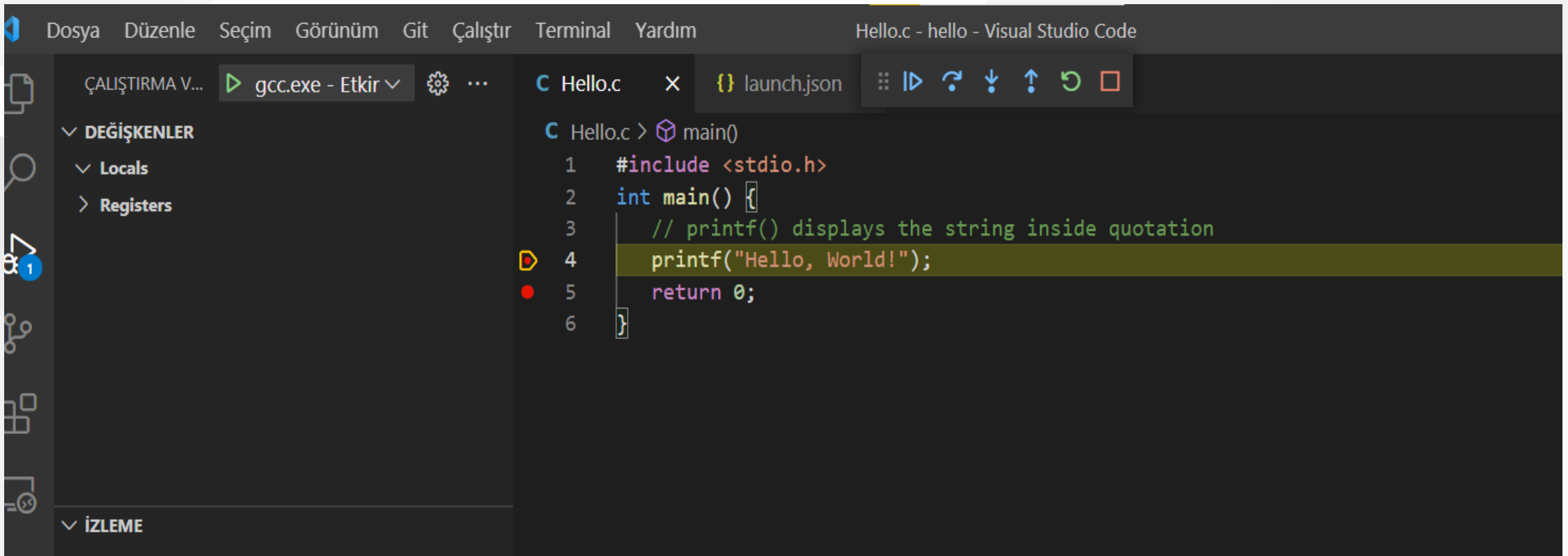
- Yapılandırma seçin
- clang.exe - Etkin dosyayı derle ve dosyada hata ayıkla derleyici: "C:\Program Files\LLVM\bin\clang.exe"
- gcc.exe - Etkin dosyayı derle ve dosyada hata ayıkla derleyici: "C:\Program Files\mingw-w64\x86_64-8...
- gcc.exe - Etkin dosyayı derle ve dosyada hata ayıkla derleyici: C:\cygwin64\bin\gcc.exe
- Varsayılan Yapılandırma

The 'gcc.exe' option is highlighted in blue. Below the menu, the code editor shows the following code:

```
5     return 0;  
6 }
```

VSCoDe (Install / Compile / Run / Debug) (13)

this will run debugger and you will see debug points activated



```
Dosya  Düzenle  Seçim  Görünüm  Git  Çalıştır  Terminal  Yardım  Hello.c - hello - Visual Studio Code

ÇALIŞTIRMA V...  gcc.exe - Etkir  ⚙️  ...  Hello.c  ×  {} launch.json  ⌵  ⏪  ⏩  ⏴  ⏵  ⏴  ⏵  ⏴  ⏵  ⏴  ⏵

DEĞİŞKENLER
  Locals
  Registers

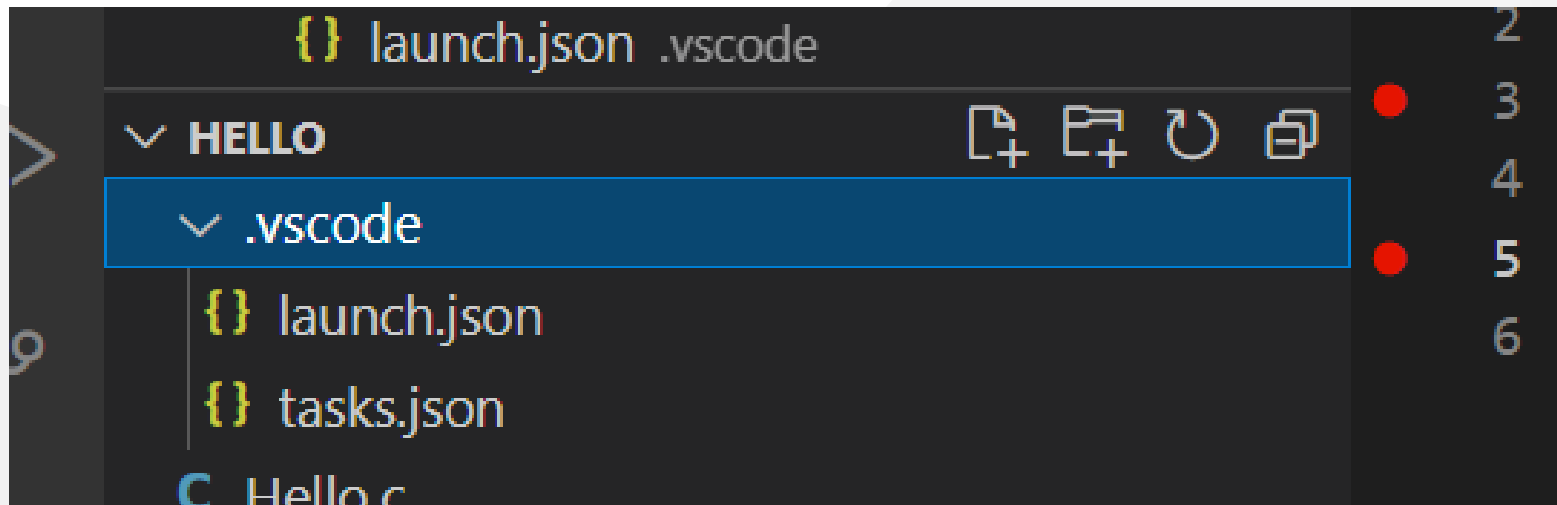
İZLEME

C Hello.c > main()
1  #include <stdio.h>
2  int main() {
3      // printf() displays the string inside quotation
4      printf("Hello, World!");
5      return 0;
6  }
```

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



VSCode (Install / Compile / Run / Debug) (15)

launch.json

```
{
  // Olası öznelikler hakkında bilgi edinmek için IntelliSense kullanın.
  // Mevcut özneliklerin 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\\bin\\gdb.exe",
      "setupCommands": [
        {
          "description": "gdb için düzgün yazdırmayı etkinleştir",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "C/C++: gcc.exe etkin dosyayı derle"
    }
  ]
}
```

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\\gcc.exe",
      "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"
}
```

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>

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)

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`

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.exe",
```

for LLVM

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

for more details please check the sample source codes.

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

//TODO//



Notepad++ (Install / Compile) (1)

Please download Notepad++ from the following link

[Downloads | Notepad++](#)

Notepad++ (Install / Compile) (2)

Download and install MinGW or LLVM compiler (if you downloaded then skip this step)

MinGW installer (clang)

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

LLVM installer (gcc / g++)

[Download LLVM releases](#)

Also use the following notes

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

Notepad++ (Install / Compile) (3)

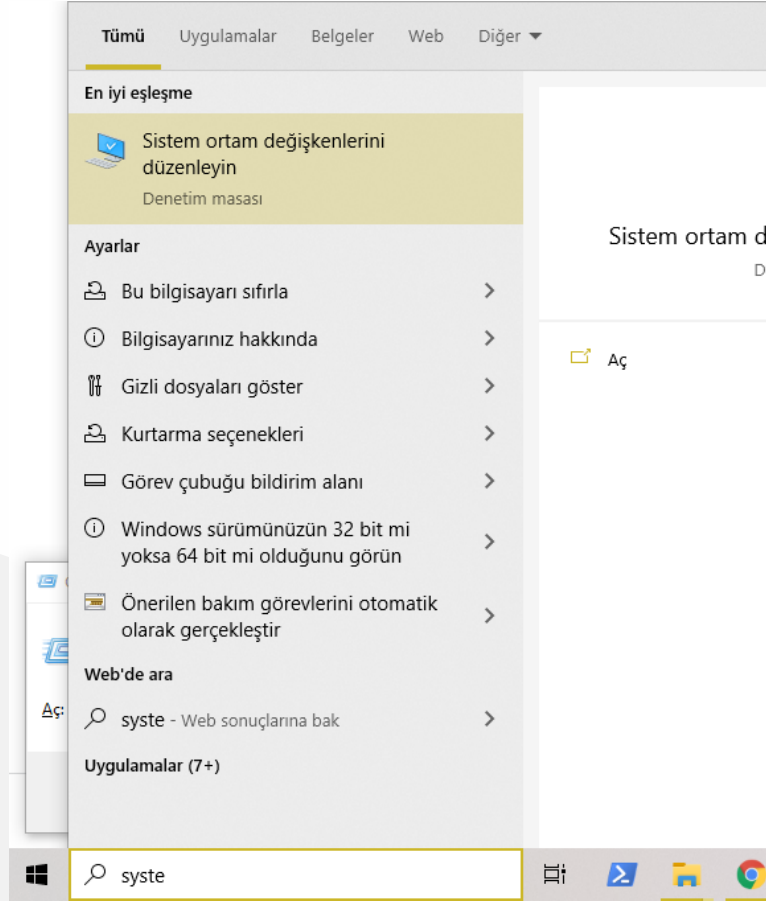
Open a console with "cmd" and test the following commands if commands are not recognized then set the system environment variable add gcc and g++ exe paths to path variable (add to both system and user path variable)

```
gcc --version
```

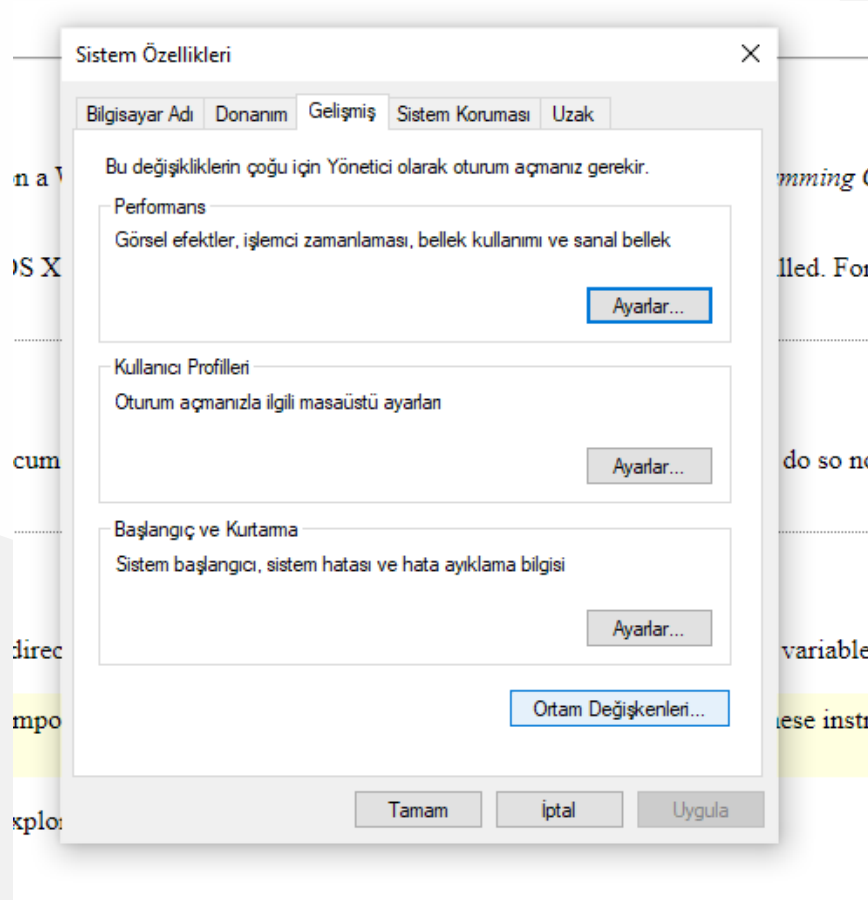
```
g++ --version
```

```
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.
```

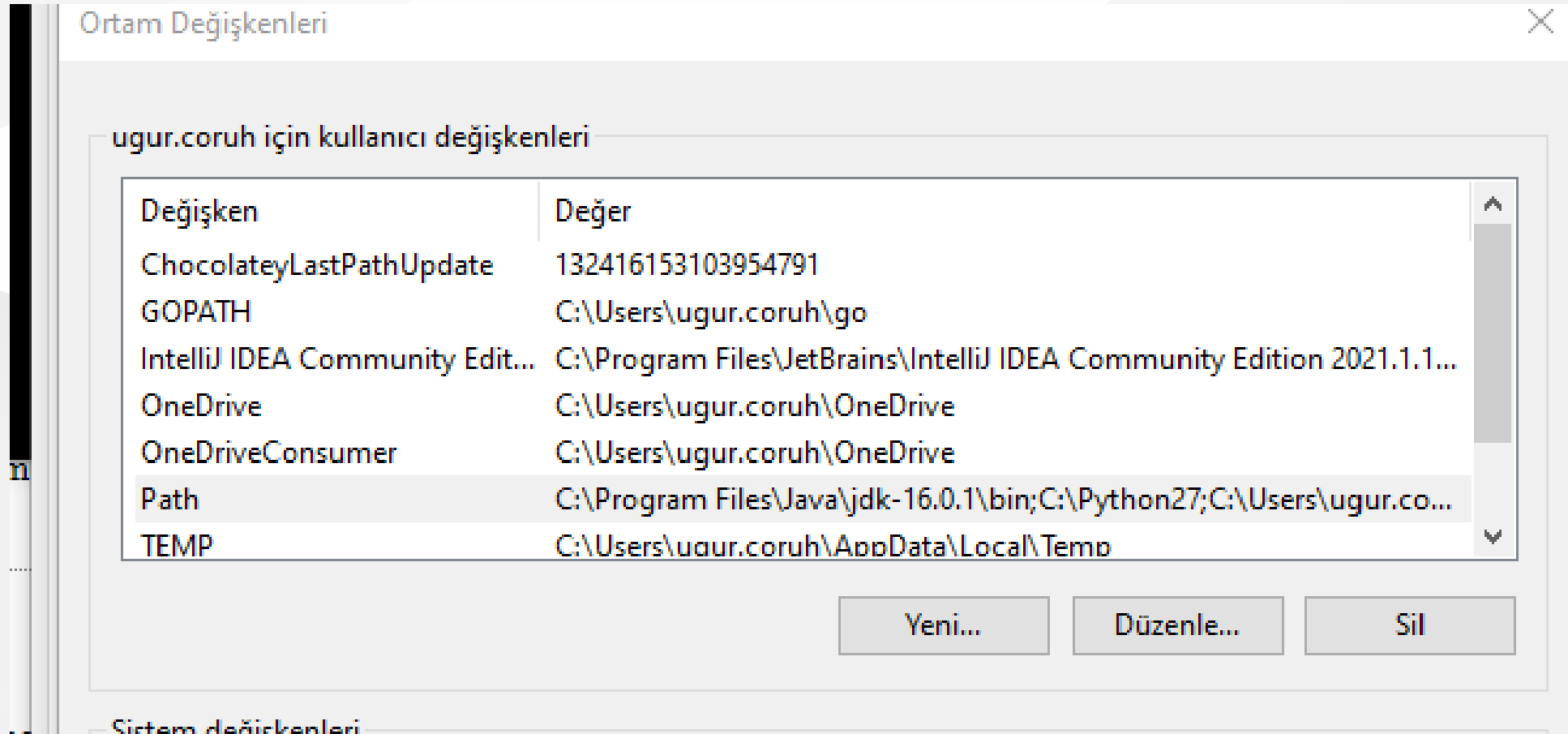
Notepad++ (Install / Compile) (4)



Notepad++ (Install / Compile) (5)



Notepad++ (Install / Compile) (6)



Notepad++ (Install / Compile) (7)

lirec

mpo

splo

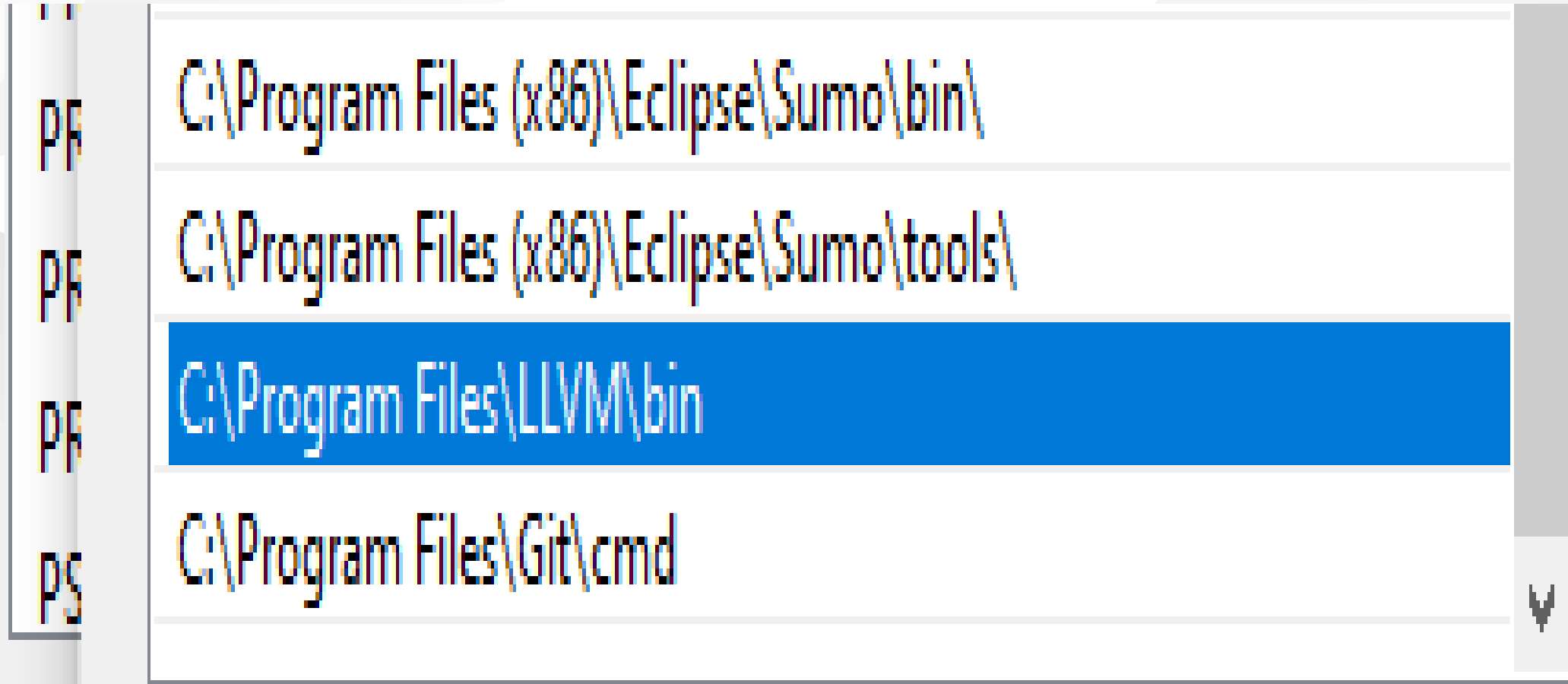
Sistem değişkenleri

Değişken	Değer
Path	C:\Program Files\Java\jre1.8.0_231\bin;C:\Program Files\Java\jdk-1...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 94 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	5e03
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\WINDOWS\cvs

ere executable cod

l your computer w

Notepad++ (Install / Compile) (8)



Notepad++ (Install / Compile) (9)

```
C:\Users\ugur.coruh\go\bin
C:\Program Files (x86)\Python38-32
C:\Program Files (x86)\Python38-32\Scripts
C:\Users\ugur.coruh\AppData\Roaming\Python\Python38\Scripts
C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw64\bin
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\...
%IntelliJ IDEA Community Edition%
%USERPROFILE%\dotnet\tools
%USERPROFILE%\go\bin
```

Yukarı Taşı

Aşağı Taşı

Metni düzenle...

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
```

Notepad++ (Install / Compile) (11)

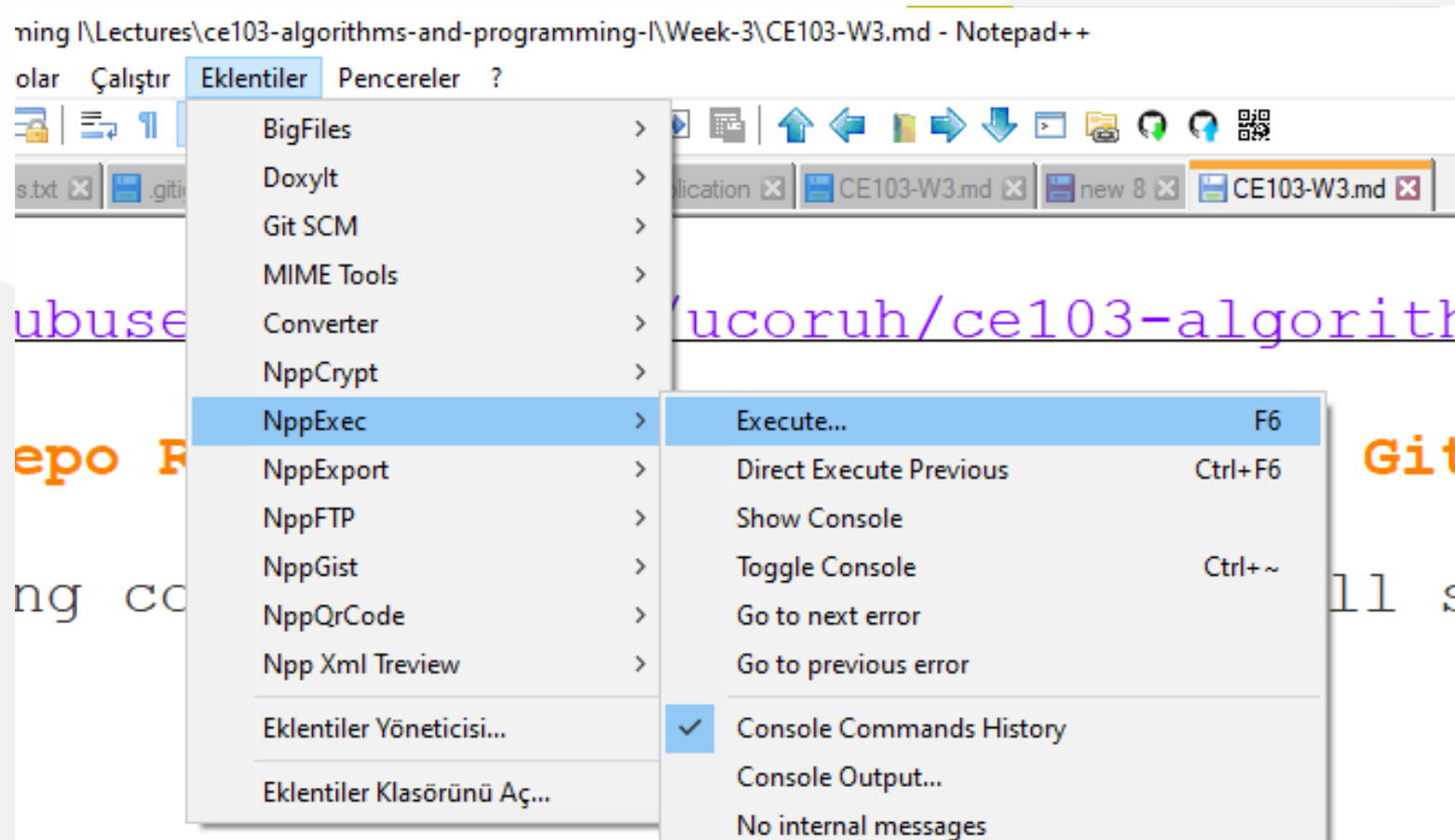
for clang.exe , lldb.exe

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

Notepad++ (Install / Compile) (12)

This folder paths changes according to your setup

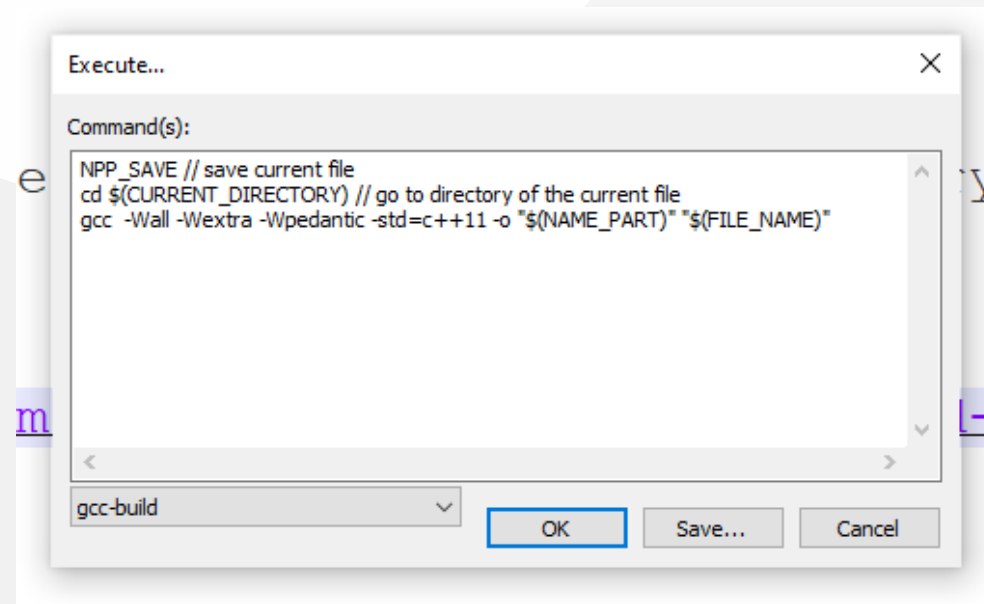
Open NppExec extension (install from extension manager if not exist)



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)"
```



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](#)

You can modify or add multiple scripts for another tasks.

Vi/Vim (C/C++)

//TODO//



Eclipse (C/C++)

```
//TODO//
```



Netbeans (C/C++)

```
//TODO//
```



Turbo C++/C

`//TODO//`



Cmake (C++/C) (1)

CMake (<http://www.cmake.org/>) is a program which generates the **Makefiles** used by **Make**.

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-platforms
- Very flexible

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

Cmake (C++/C) (4) (Download and Install)

use the following link for download

[Download | CMake](#)

Cmake (C++/C) (5) (WSL and Linux Environment)

Hello world with CMake

Cmake (C++/C) (6) (Windows Environment)

main.c

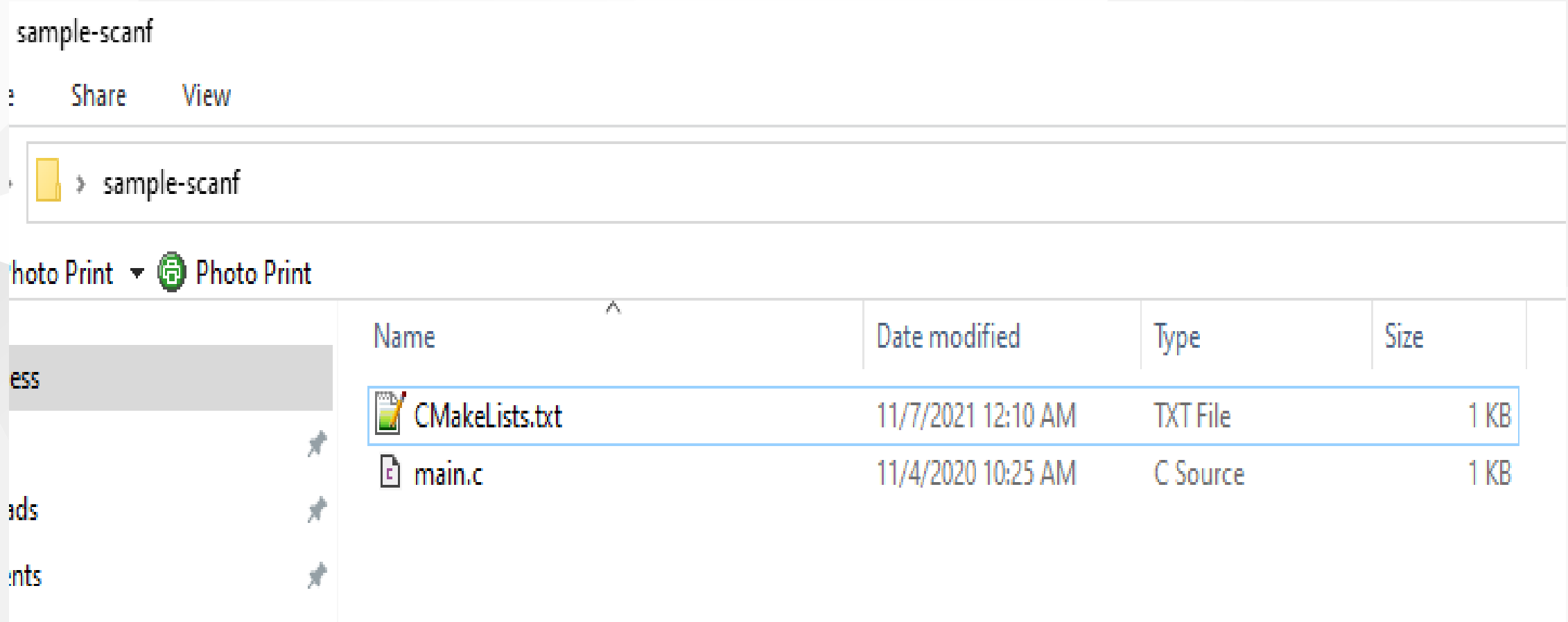
```
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

CMakeLists.txt

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

CE103 Algorithms and Programming I Cmake (C++/C) (7) (Windows Environment)

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



sample-scanf

Share View

sample-scanf

Photo Print Photo Print

Name	Date modified	Type	Size
CMakeLists.txt	11/7/2021 12:10 AM	TXT File	1 KB
main.c	11/4/2020 10:25 AM	C Source	1 KB

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

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

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/14.30.30704/bin/Hostx64/x64/cl.exe - skipped
-- 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/MSVC/14.30.30704/bin/Hostx64/x64/cl.exe - skipped
-- 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>
```

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

also following files are generated

sample-scanf

Share View

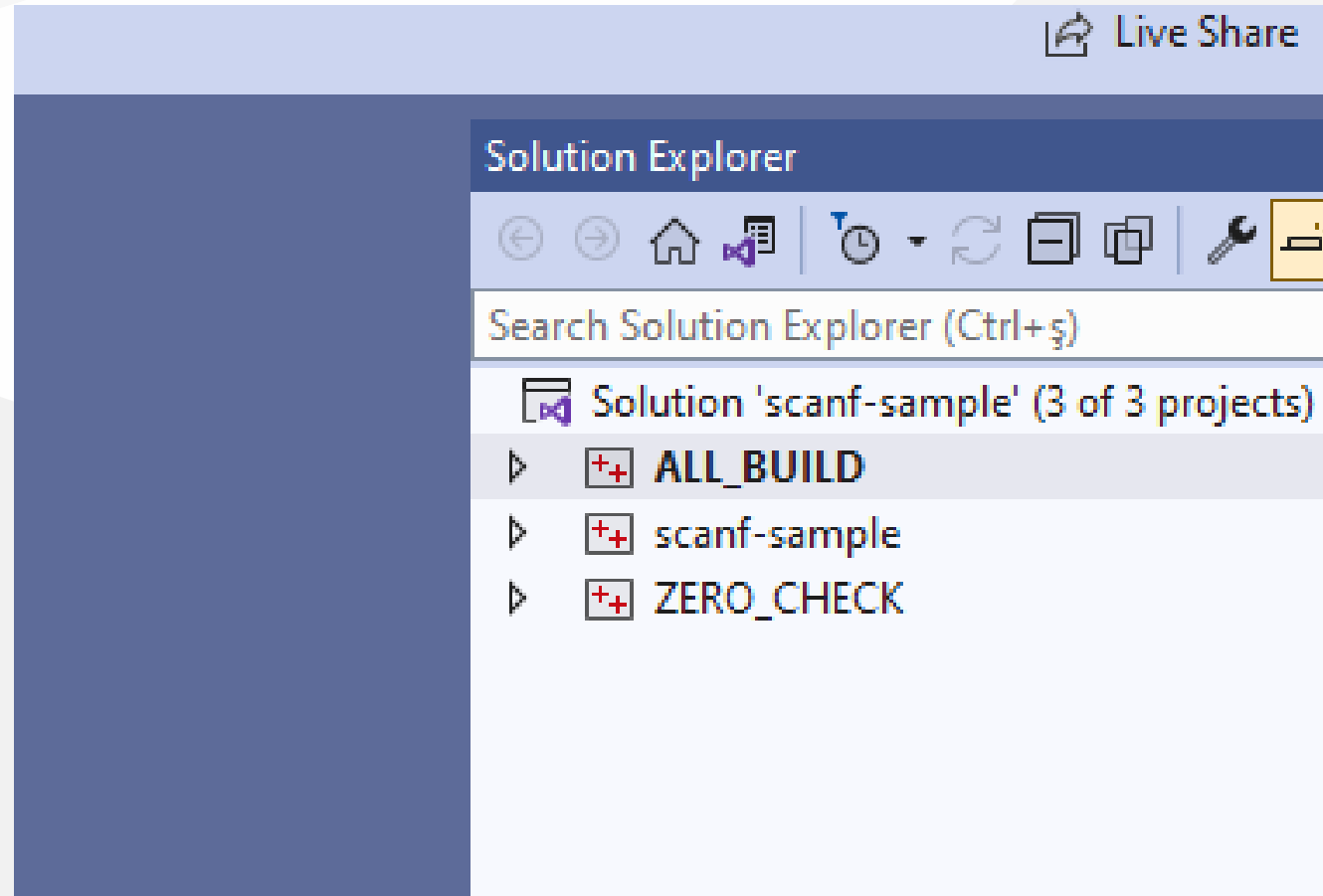
sample-scanf

Photo Print Photo Print

Name	Date modified	Type	Size
CMakeFiles	11/7/2021 12:43 AM	File folder	
ALL_BUILD.vcxproj	11/7/2021 12:43 AM	VC++ Project	41 KB
ALL_BUILD.vcxproj.filters	11/7/2021 12:43 AM	VC++ Project Filte...	1 KB
cmake_install.cmake	11/7/2021 12:43 AM	CMake Source File	2 KB
CMakeCache.txt	11/7/2021 12:43 AM	TXT File	14 KB
CMakeLists.txt	11/7/2021 12:10 AM	TXT File	1 KB
main.c	11/4/2020 10:25 AM	C Source	1 KB
scanf-sample.sln	11/7/2021 12:43 AM	Visual Studio Solu...	4 KB
scanf-sample.vcxproj	11/7/2021 12:43 AM	VC++ Project	50 KB
scanf-sample.vcxproj.filters	11/7/2021 12:43 AM	VC++ Project Filte...	1 KB
ZERO_CHECK.vcxproj	11/7/2021 12:43 AM	VC++ Project	41 KB
ZERO_CHECK.vcxproj.filters	11/7/2021 12:43 AM	VC++ Project Filte...	1 KB

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

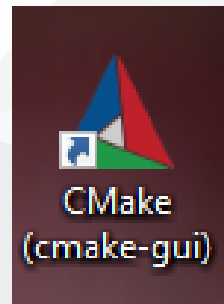
if we open scanf-sample.sln file we will have automated generated project files



Cmake (C++/C) (11) (Windows Environment)

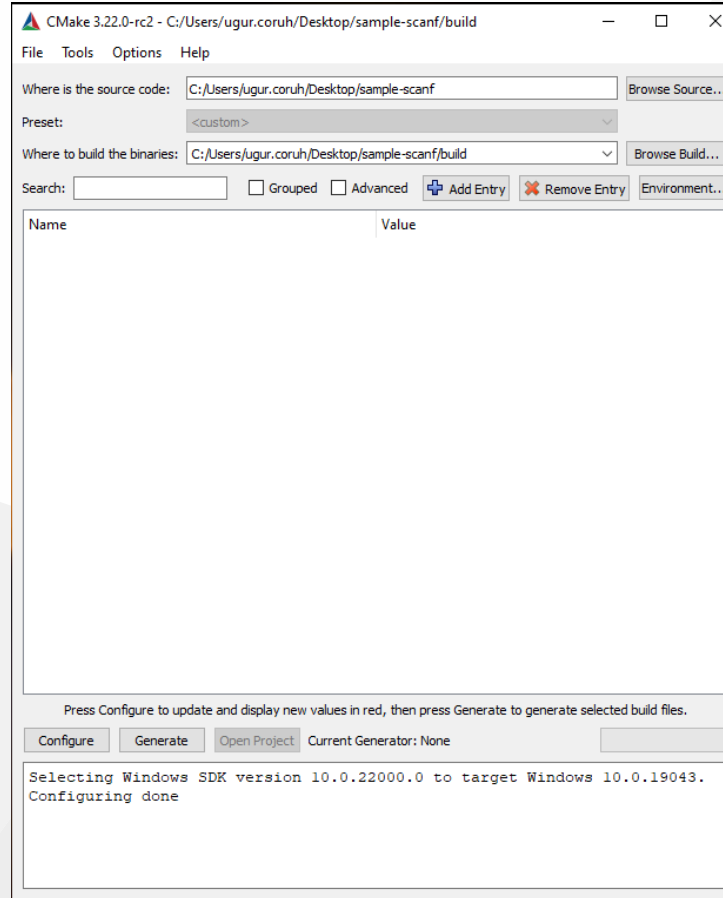
you can make scanf-sample with startup project with right click and then run on visual studio.

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



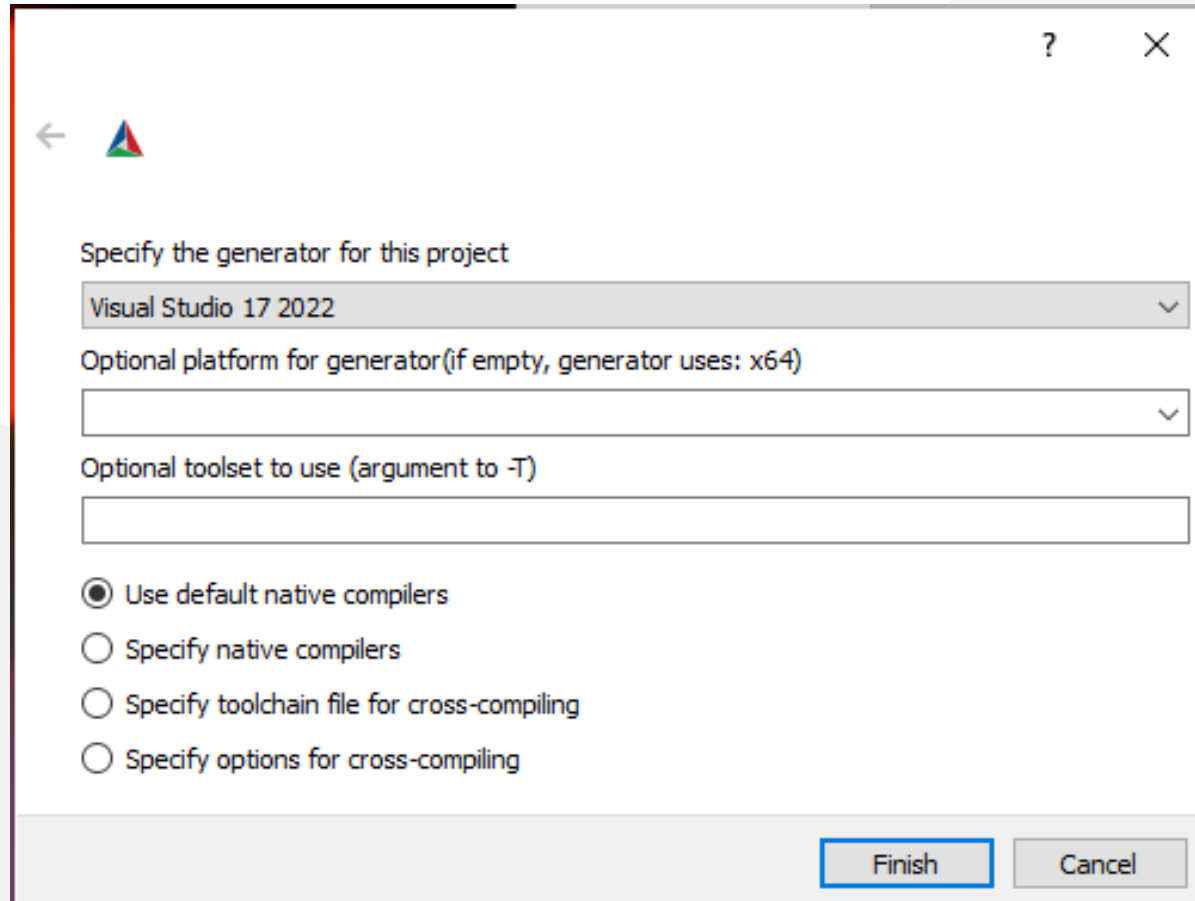
Cmake (C++/C) (12) (Windows Environment)

Open GUI and Select *File-> Delete Cache*

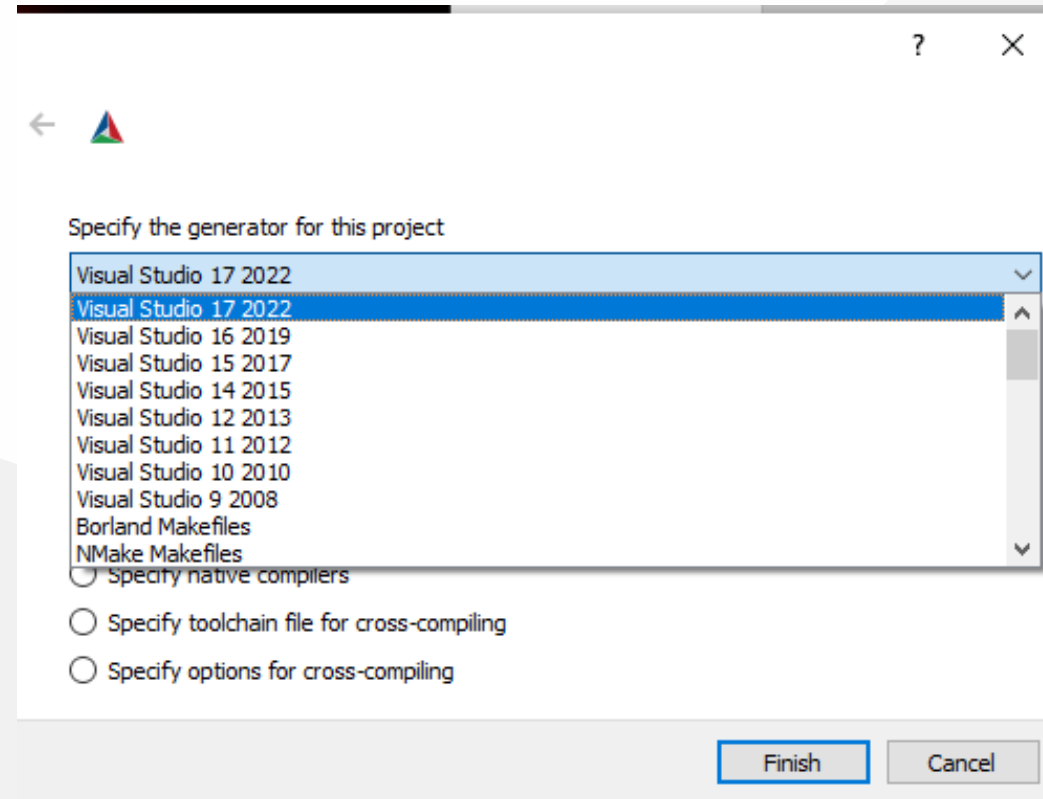


Cmake (C++/C) (13) (Windows Environment)

then you can click "Configure" to select build tool



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



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](#)

Make (1)

Sample

hello.c

```
#include <stdio.h>

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

Make (2)

Makefile

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

Make (3)

compile.bat









```
make all .
```

will create hello.exe

check hello-make sample

s-and-programming-I > Week-2 > hello-make

Print

	Name	Date modified
	 compile.bat	11/7/2021
	 hello.c	11/2/2021
	 Makefile	11/2/2021
		
		

ing-I

JAVA Environment and Development

```
//TODO//
```



JDK and JRE Setup

//TODO//



System Environments and Paths for Java

//TODO//



Netbeans (Java)

```
//TODO//
```



Eclipse (Java)

```
//TODO//
```



Intelij Idea (Jet Brains) (Java)

```
//TODO//
```



VSCode (Java)

```
//TODO//
```



Notepad++ (Java)

```
//TODO//
```



Cmake (Java)

ASTERICS_HPC

C# Environment and Development

Visual Studio Community Edition (C#)

```
//TODO//
```



Notepad++ (C#)

```
//TODO//
```



Cmake (C#)

Outline

Common Tools and Platforms

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

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



Notepad++ (Notepad for Source Code)

[Downloads | Notepad++](#)

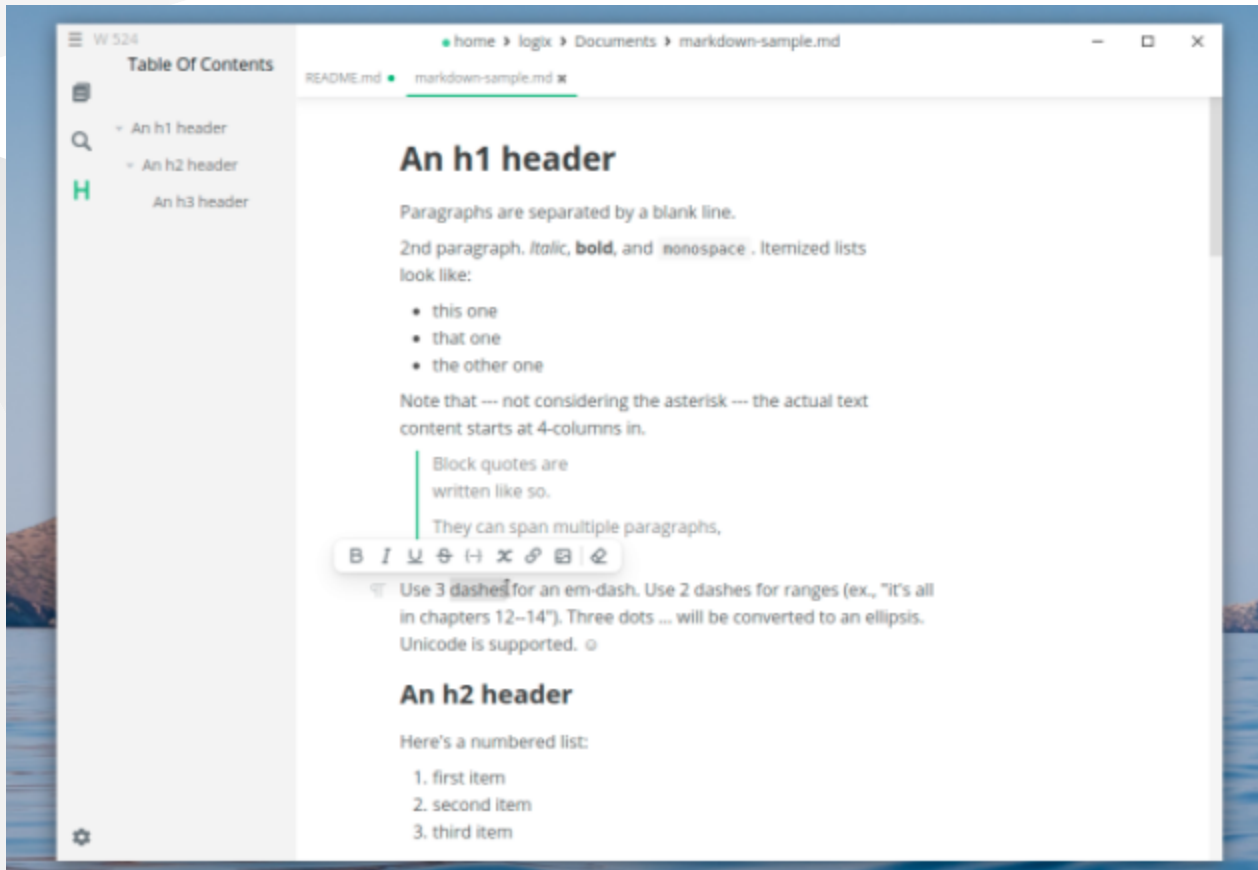
HxD (Hex Editor)

HxD - Freeware Hex Editor and Disk Editor | [mh-nexus](#)

Marktext (Markdown Syntax Editor)



<https://marktext.app/>



Cygwin (Linux environment for Windows)



<https://www.cygwin.com/>

The screenshot displays a Windows desktop with several open windows:

- Terminal Window (Left):** Shows the Cygwin shell prompt. The user navigates to the Cygwin installation directory and lists files.


```

/bin cygdrive cygwin.bat cygwin.ico etc
Pakrandt@MANN-EDU-P /
$ cd cygdrive/
Pakrandt@MANN-EDU-P /cygdrive
$ ls
Pakrandt@MANN-EDU-P /cygdrive
$ cd c
Pakrandt@MANN-EDU-P /cygdrive/c
$ ls
AUTOEXEC.BAT      MSOCache
Anrufbeantworter My Music
BOOK              MIDETECT.COM
BOOT.INI          Oracle
BOOTSECT.DOS     PRELOAD.AAA
Dorland           Program Files
CONFIG.SYS        Programme
CollideSoftware  RECYCLER
Daten            Recycled
Dokumente und Einstellungen SDBSetup.log
Download         SYSTEMPO
IO.SYS           System Volume Information
Images           Temp
Inetpub          TerraTec
Kunden.mp3       Tomcat 5.0
Lotus            VALUADD
MSDOS.SYS        WINDOWS
            
```
- File Explorer (Middle):** Shows the contents of the Cygwin directory.

Name	Size	MTime
..	UP-DIR	
.JBex	0	Jun 5 10:51
.horland	0	Jun 10 01:53
.jbuilder2005	0	Jun 10 02:02
.nc	0	Jun 13 22:30
.netbeans	0	Jun 10 02:05
.prinetinc2005	0	Jun 10 02:02
.qualitycentral	0	Jun 10 01:54
.AnagramGame	0	Jun 10 02:22
.Anwendungsdaten	0	Jun 12 20:28
.Application Data	0	Mar 22 18:10
.Cookies	0	Jun 7 23:40
.Desktop	0	Jun 13 22:50
.Druckumgebung	0	Sep 14 2004
.Eigene Dateien	0	May 5 23:45
.Favoriten	0	May 13 14:35
..		
- Terminal Window (Right):** Shows the output of the 'ls' command.

Name	Size	MTime
/bin	0	Jun 13 22:34
/cygdrive	0	Jan 1 1970
/etc	0	Jun 13 22:36
/lib	0	Jun 13 22:34
/tmp	0	Jun 13 23:04
/usr	0	Jun 13 22:30
/var	0	Jun 13 22:34
*cygwin.bat	57	Jun 13 22:36
*cygwin.ico	7022	Jun 13 22:36
null	0	Jun 13 23:03
/bin		
- Terminal Window (Bottom Right):** Shows a PuTTY session where the user logs in as root and runs 'kwin' and 'kbuildsysoca' commands.


```

login as: root
Password:
Last login: Mon Jun 13 22:39:05 2005 from graceyves.gy0128
Have a lot of fun...
graceyves:~ # kwin &
[1] 7644
graceyves:~ # kbuildsysoca running...
Launched ok, pid = 7660
kclock &
[2] 7661
graceyves:~ #
            
```
- xClock Window (Bottom Left):** A small utility window showing a clock face with a checkmark.

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



<https://www.dependencywalker.com/>

Dependency Walker - [Stooges.exe]

File Edit View Options Profile Window Help

STOOGES.EXE

- LARRY.DLL
- KERNEL32.DLL
- NTDLL.DLL
- NTDLL.DLL
- CURLY.DLL
- SHEMP.DLL
- MOE.DLL
- KERNEL32.DLL
- NTDLL.DLL

PI^	Ordinal	Hint	Function	Entry Point
■	N/A	N/A	IsKnucklehead	Not Bound
■	N/A	N/A	int SaySoitenly(char *,...)	Not Bound

E^	Ordinal	Hint	Function	Entry Point
■	4 (0x0004)	1 (0x0001)	int SaySoitenly(char *,...)	SHEMP.?SaySoitenly@@YAHP/
■	5 (0x0005)	2 (0x0002)	DoinkLarrysEye	0x00001010
■	3 (0x0003)	0 (0x0000)	void SayPoifect(_int64)	0x00001020
■	1 (0x0001)	N/A	N/A	0x00001020
■	2 (0x0002)	3 (0x0003)	DoinkMoesEye	SHEMP.DoinkMoesEye

Module ^	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum	Real Checksum	CPU	Subsystem
CURLY.DLL	11/14/2006 5:17p	11/14/2006 5:13p	2,560	A	0x0000F739	0x0000F759	x86	GUI
KERNEL32.DLL	08/30/2006 1:22a	08/30/2006 1:20a	871,424	A	0x000E388E	0x000E388E	x86	Console
LARRY.DLL	11/14/2006 5:13p	11/14/2006 5:13p	2,560	A	0x000053DB	0x000053DB	x86	GUI
MOE.DLL	11/14/2006 5:15p	11/14/2006 5:15p	2,560	A	0x0000B191	0x0000B191	x86	GUI
NTDLL.DLL	08/30/2006 1:23a	08/30/2006 1:21a	1,147,664	A	0x00125FA5	0x00125FA5	x86	Console
SHEMP.DLL	11/14/2006 5:13p	11/14/2006 5:13p	2,560	A	0x00001CE7	0x00001CE7	x86	GUI

00:00:00.093: LoadLibraryA("Moe.dll") called from "STOOGES.EXE" at address 0x00401024 by thread 1.
 00:00:00.093: Loaded "MOE.DLL" at address 0x00020000 by thread 1. Successfully hooked module.
 00:00:00.093:DllMain(0x00020000, DLL_PROCESS_ATTACH, 0x00000000) in "MOE.DLL" called by thread 1.
 00:00:00.093: DllMain(0x00020000, DLL_PROCESS_ATTACH, 0x00000000) in "MOE.DLL" returned 1 (0x1) by thread 1.
 00:00:00.093: LoadLibraryA("Moe.dll") returned 0x00020000 by thread 1.
 00:00:00.109: GetProcAddress(0x00020000 [MOE.DLL], "SmackCurly") called from "STOOGES.EXE" at address 0x0040102B and returne

For Help, press F1

Doxygen (Code Documentation)



Doxygen: Doxygen

The screenshot displays the Doxygen-generated documentation for the ACME Project 1.0. The main heading is "ACME Project 1.0". Below it, there are navigation tabs for "Main Page" and "Classes". A search bar is located in the top right corner. The left sidebar shows a tree view of the project structure, with "ACME Project" expanded to show "Classes", "Class List", and "acme". Under "acme", "ACMESmartphone" is selected. The main content area shows the "acme.ACMESmartphone Class Reference". It includes sections for "Public Member Functions" and "Public Attributes".

acme.ACMESmartphone Class Reference

Public Member Functions

- `ACMESmartphone` (double model, String license)
- String `findRoadRunner` (String city, String state) throws IOException
- void `zapRoadRunner` (int voltage) throws IOException

Public Attributes

- String `LongLat` = "Longitude = 39.2334, Latitude = 41.4899"

Detailed Description

Works like a regular smartphone but also tracks roadrunners.

The ACME Smartphone can perform similar functions as other smartphones, such as making phone calls, sending

Generated on Sun Sep 27 2015 09:05:27 for ACME Project by doxygen 1.8.7

Sonarlint (Code Quality and Code Security Extension)



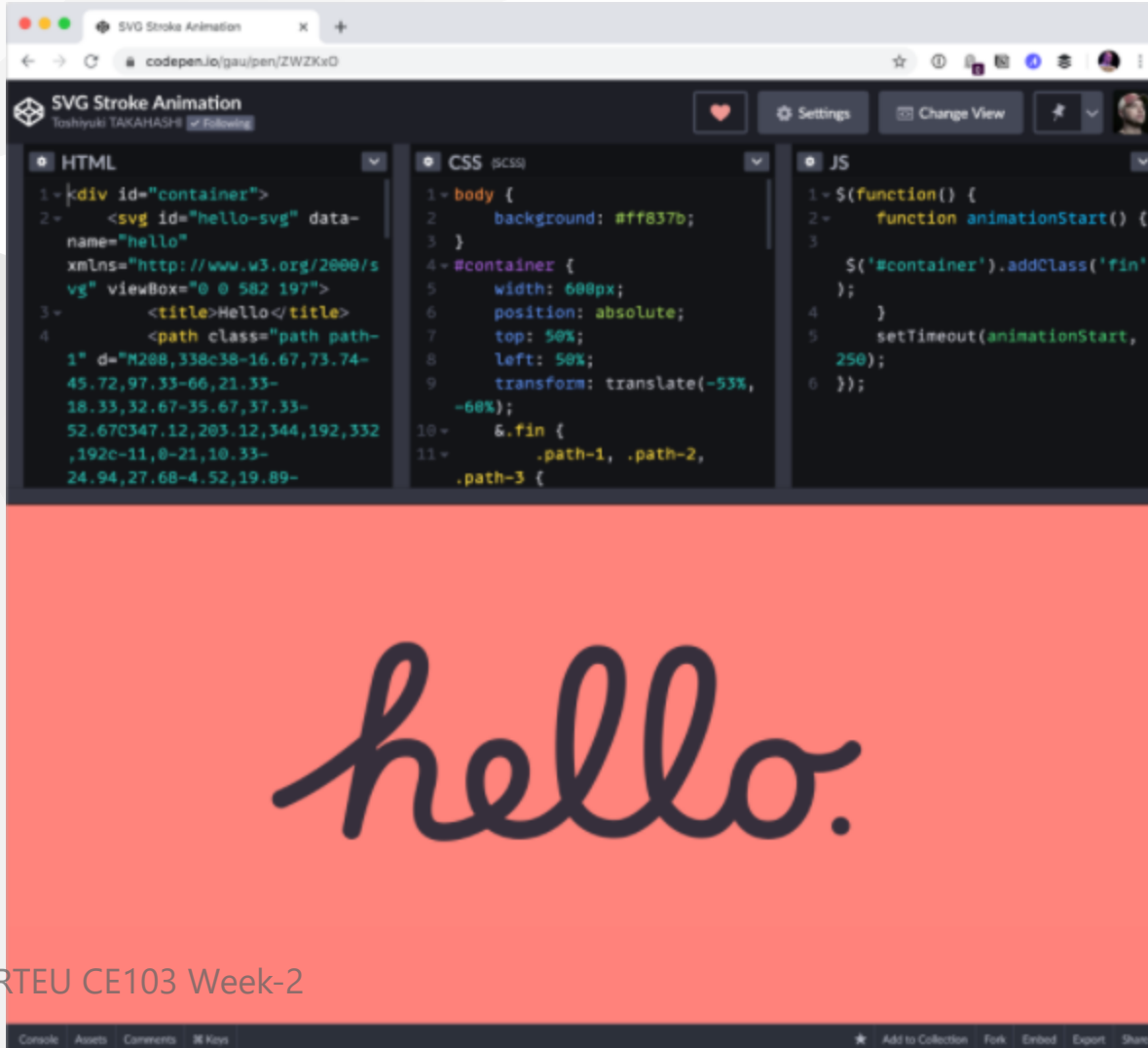
<https://www.sonarlint.org/>

Codepen.io (online code sharing)

CE103 Algorithms and Programming I



<https://codepen.io/>



The screenshot shows a Codepen.io project titled "SVG Stroke Animation" by Toshiyuki TAKAHASHI. The project is displayed in a dark-themed editor with three panels: HTML, CSS, and JS. The HTML panel contains a single path for an SVG stroke animation. The CSS panel defines the container's background color as #ff837b and sets the text to be absolute, centered, and animated. The JS panel contains a function that adds a class to the container and triggers the animation. Below the code panels, a preview of the word "hello." is shown in a dark blue, cursive font on a light red background.

```
HTML
1- <div id="container">
2-   <svg id="hello-svg" data-
   name="hello"
   xmlns="http://www.w3.org/2000/s
   vg" viewBox="0 0 582 197">
3-   <title>Hello</title>
4-   <path class="path path-
   1" d="M288,338c38-16.67,73.74-
   45.72,97.33-66,21.33-
   18.33,32.67-35.67,37.33-
   52.67c347.12,203.12,344,192,332
   ,192c-11,0-21,10.33-
   24.94,27.68-4.52,19.89-
```

```
CSS (SCSS)
1- body {
2-   background: #ff837b;
3- }
4- #container {
5-   width: 600px;
6-   position: absolute;
7-   top: 50%;
8-   left: 50%;
9-   transform: translate(-53%,
   -60%);
10-  &.fin {
11-    .path-1, .path-2,
    .path-3 {
```

```
JS
1- $(function() {
2-   function animationStart() {
3-     $('#container').addClass('fin'
   );
4-   }
5-   setTimeout(animationStart,
   250);
6- });
```

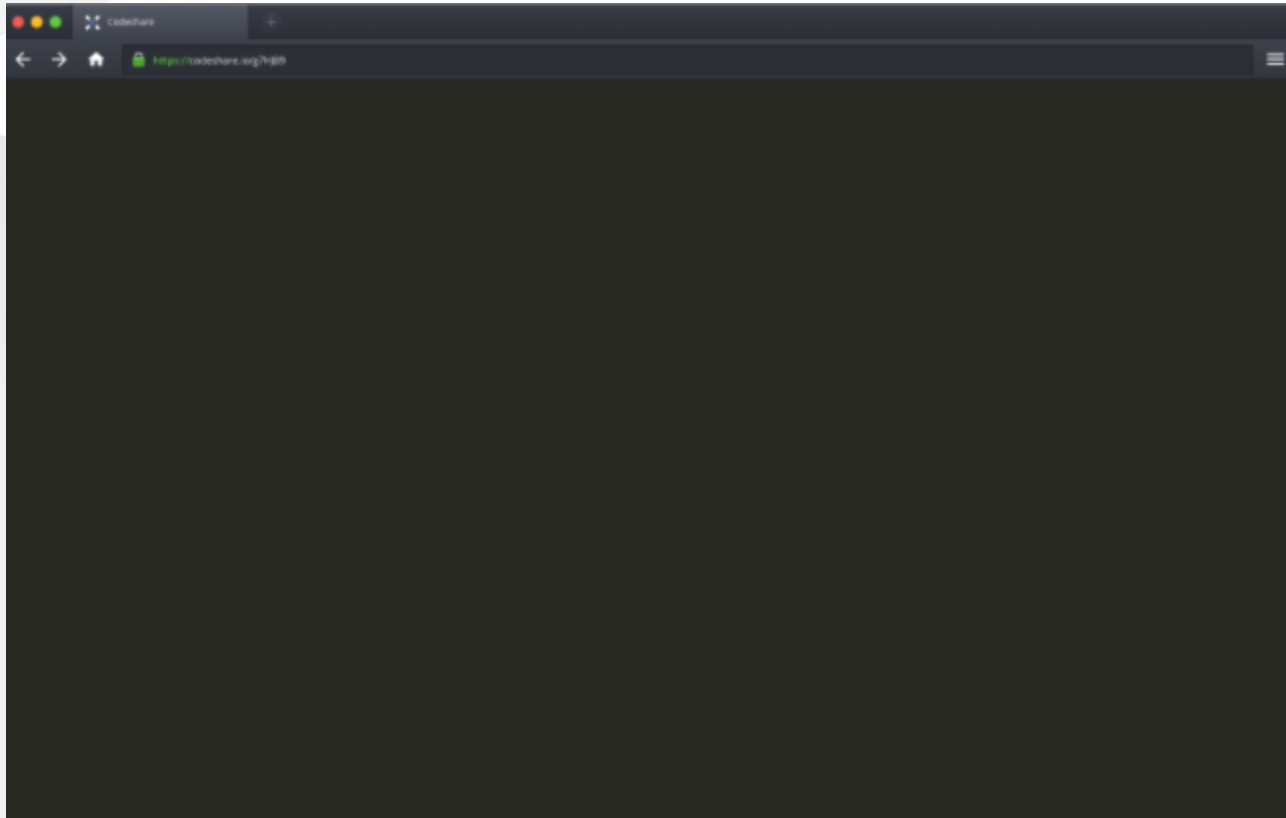

Credit Card Sample

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

Codeshare.io (real time code sharing)



<https://codeshare.io/>



Codebeautify.org (online data conversion tools)



<https://codebeautify.org/>

AsciiFlow.com (ASCII drawing tool)

```
//TODO//
```



Freemind (opensource mindmap application)

//TODO//



Wireflow (user flow designer)

//TODO//



PlantUML (software designer)

```
//TODO//
```



Drawio (drawing tool)

```
//TODO//
```



Putty (Remote Connection)

```
//TODO//
```



MobaXterm (Remote Connection)

//TODO//



Teamviewer (Remote Connection)

//TODO//



```
//TODO//
```



Understand (Static Code Analysis)

```
//TODO//
```



JD Project (Java Decompiler)

```
//TODO//
```



Cutter (Multi-Platform Reverse Engineering Tool)

```
//TODO//
```



IDA Pro / Freeware (Native Reverse Engineering Tool)

//TODO//



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

<https://pythontutor.com/>

```
//TODO//
```



Assembly of C Code

<https://godbolt.org/>

```
//TODO//
```



Mobile Device Screen Sharing for Demo

[GitHub - Genymobile/scrcpy: Display and control your Android device](#)

Travis-CI

- Travis.yml

//TODO//



Jenkins

```
//TODO//
```



Valgrind

//TODO//



Docker

- https://www.youtube.com/watch?v=nBwJm0onzeo&ab_channel=GaryExplains Dockerfile

- DockerHub
- Docker Compose Yaml
- Dockerrun.aws.json (AWS)

//TODO//



Nuget Packages

```
//TODO//
```



Vim for Windows

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

//TODO//



SCV Cryptomanager

//TODO//



Addario CryptoBench

//TODO//



Raymond's MD5 & SHA Checksum Utility

//TODO//



SlavaSoft HashCalc

//TODO//



Portable PGP

```
//TODO//
```



Online Programming Environments

i. Hackerrank

ii. CS50 Sandbox

iii. Programiz C Online Compiler

//TODO//

