

CE103 Algorithms and Programming I

Week-2

Fall Semester, 2021-2022

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



Algorithm Basics

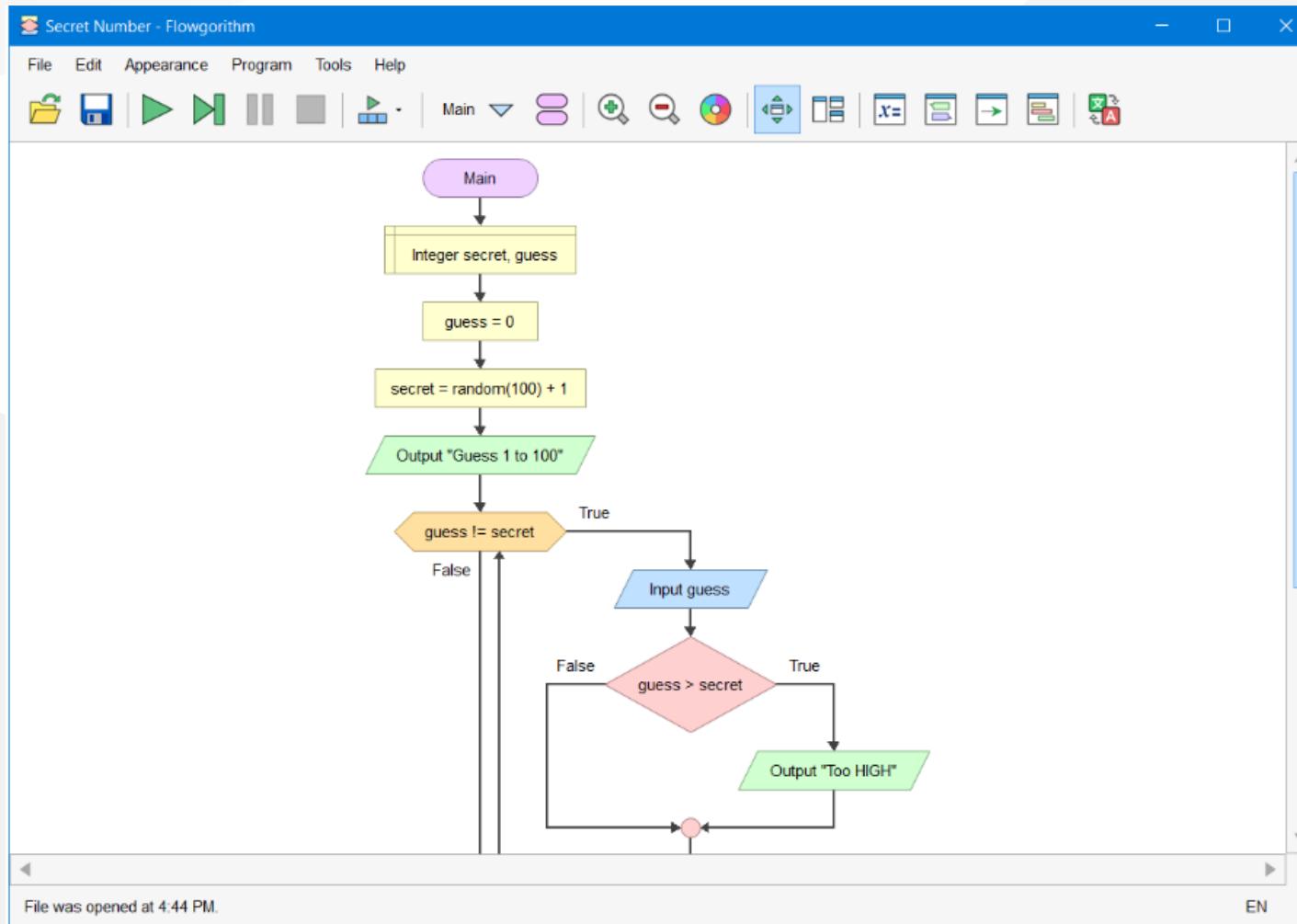


Flowgorithm (1)

- <http://www.flowgorithm.org/>
- Flowgorithm - Documentation
- <https://github.com/timoteoponce/flowgorithm-examples>

Flowgorithm (2)

- Main Window

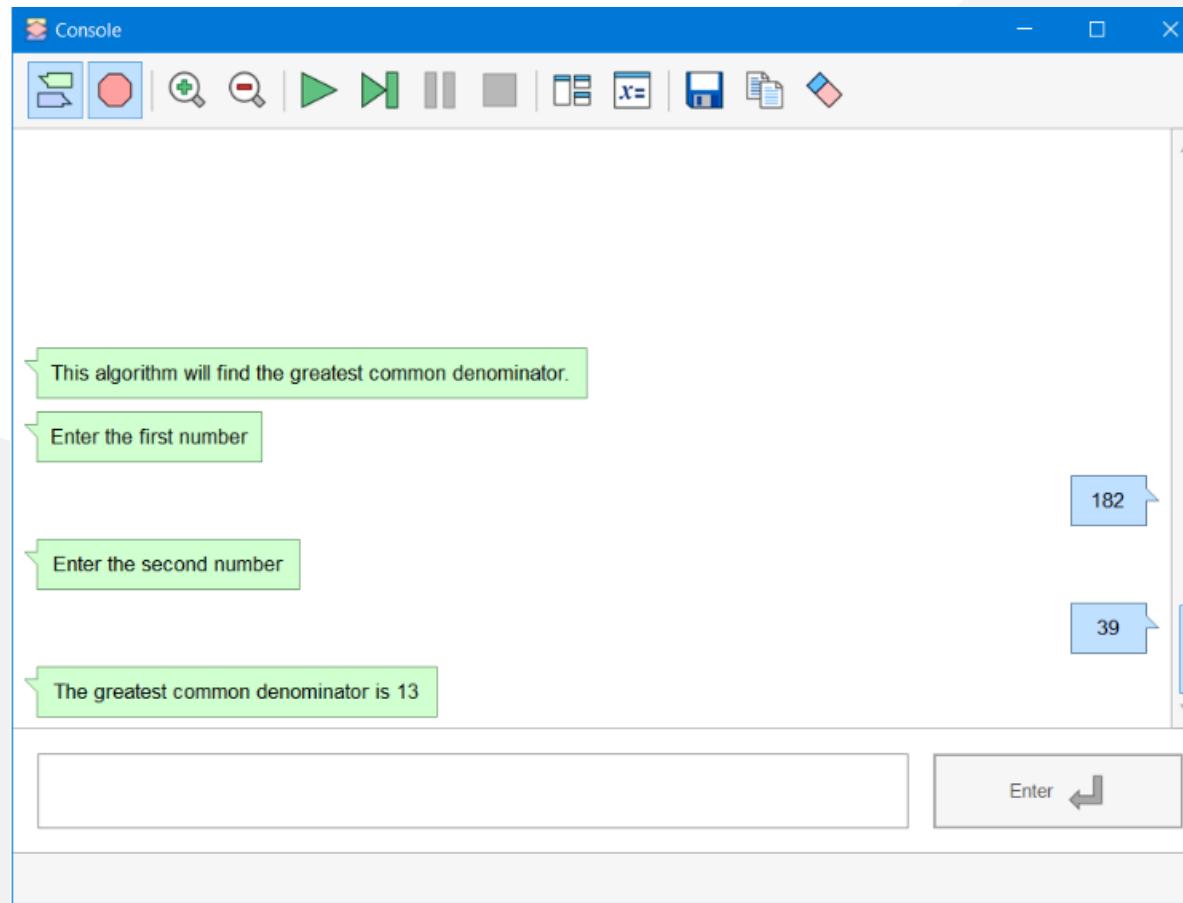


Flowgorithm (3)

- Console Window
 - The classic method to interact with the computer is to use the "Console". Flowgorithm attempts to make it look like a typical instant messenger window. The "chat bubbles" are color coded to match the Input and Output shapes used in the flowchart. If you don't want to use the chat bubbles, you can also toggle between them and the classical plain text.

Flowgorithm (4)

- Console Window

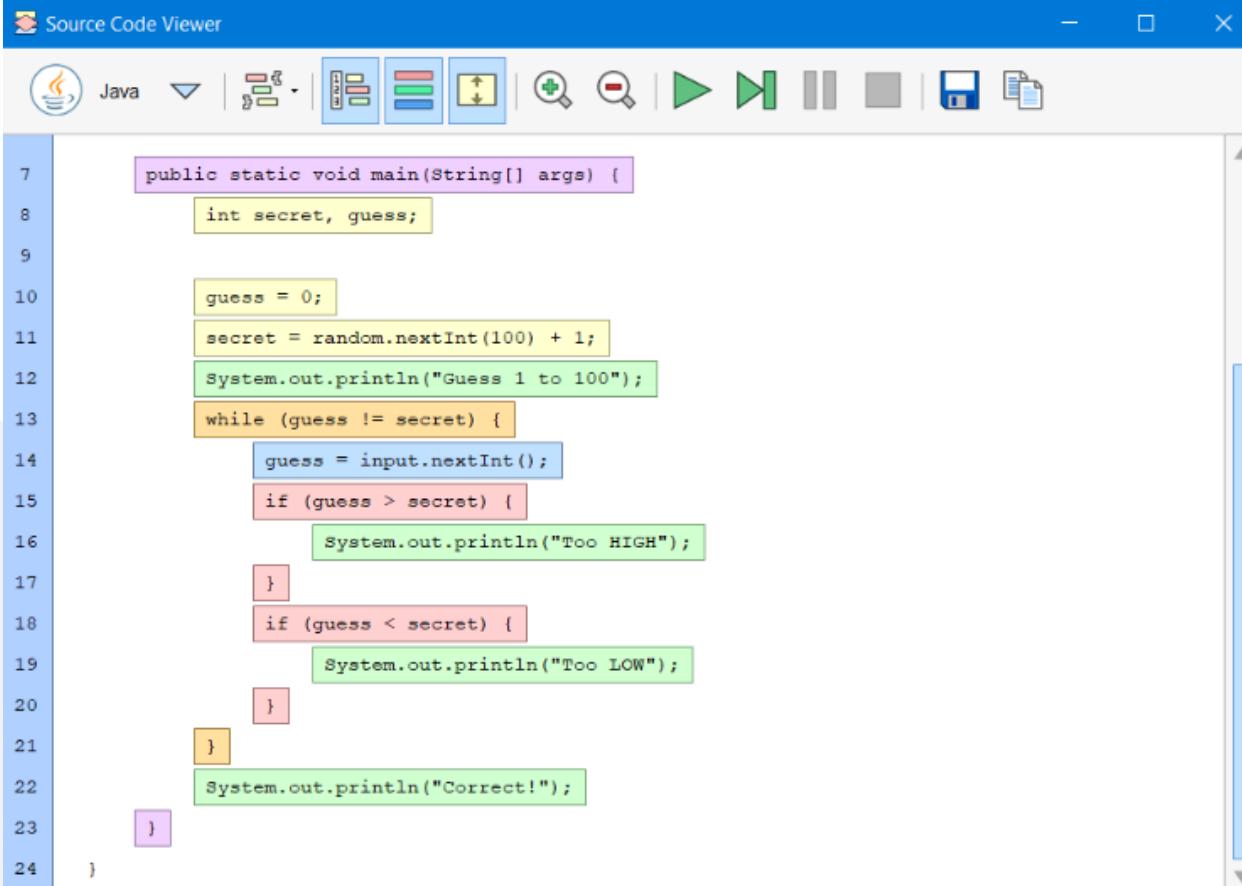


Flowgorithm (5)

- Source Code Viewer Window
 - The Source Code Viewer can convert your flowchart to several major programming languages. So, if you planning to learn a high-level language, then this feature should help you along the way.

Flowgorithm (6)

- Source Code Viewer Window



The screenshot shows a 'Source Code Viewer' window for Java. The window title is 'Source Code Viewer'. The toolbar includes icons for Java, file operations, and code navigation. The code editor displays the following Java code:

```
public static void main(String[] args) {
    int secret, guess;

    guess = 0;
    secret = random.nextInt(100) + 1;
    System.out.println("Guess 1 to 100");
    while (guess != secret) {
        guess = input.nextInt();
        if (guess > secret) {
            System.out.println("Too HIGH");
        }
        if (guess < secret) {
            System.out.println("Too LOW");
        }
    }
    System.out.println("Correct!");
}
```

Flowgorithm (7)

- Variable Watch Window
 - The variable watch window is used to keep track of how your variables are changing as your program executes. Each variable is color-coded based on its data type. At a glance, you can tell exactly what type of data is being stored - and catch where you may want to use a different data type.

Flowgorithm (8)

- Variable Watch Window

The image displays two side-by-side screenshots of the Flowgorithm Variable Watch window, illustrating the state of variables during program execution.

Screenshot 1 (Left): Shows variable values at the "Main" level. The variables and their values are:

- year: 1947
- ratio: 1.618
- name: Sacramento State
- done: true

Screenshot 2 (Right): Shows variable values at a more detailed level, likely a loop iteration. The variables and their values are:

- year: 1947
- squares:

0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
- ratio: 1.618

Flowgorithm (8)

- More Resources for Flowgorithm
 - [Flowgorithm Tutorial - TestingDocs.com](#)

Pseudocode (1)

- Algorithm design language
 - [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 an analysis of algorithms in detail.
- 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

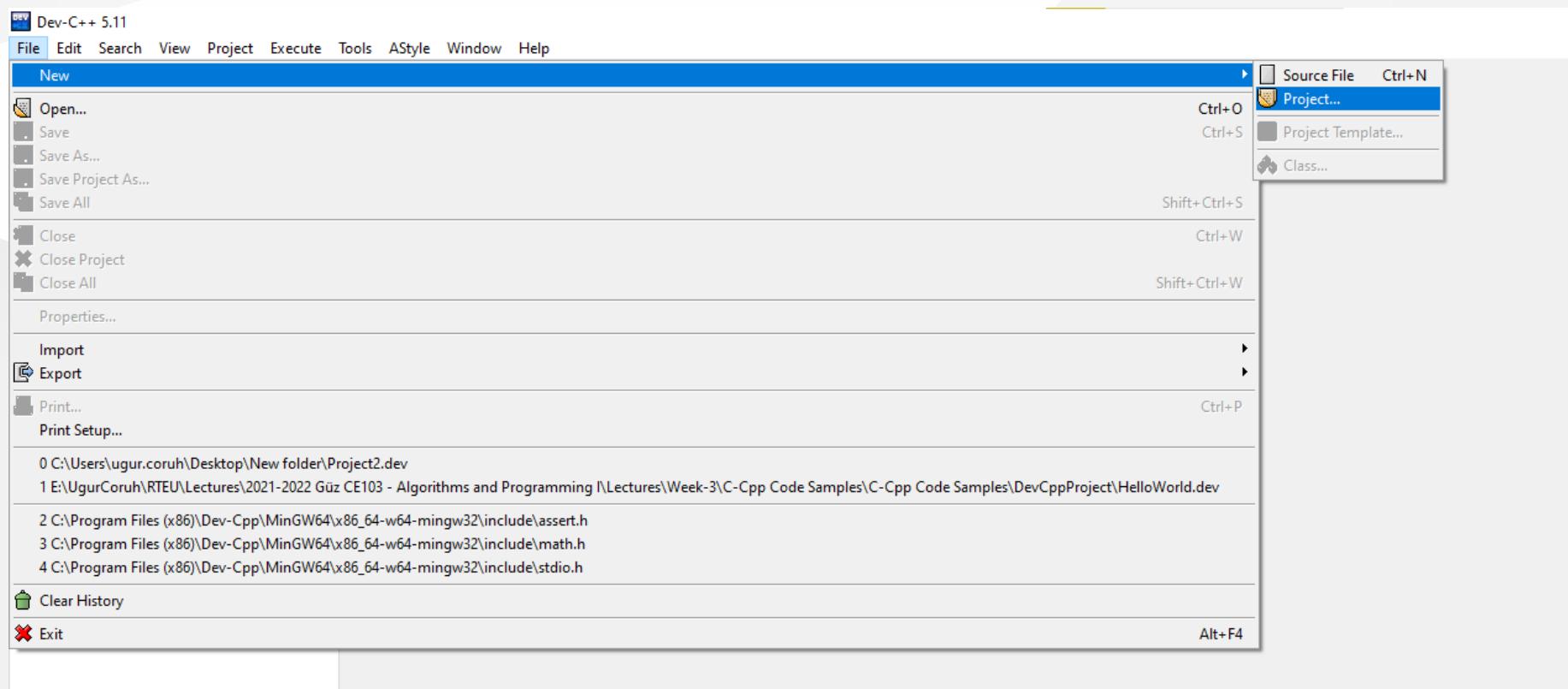
DevCpp (Install / Compile / Run / Debug) (1)

- Download DevC++ IDE from the following link
 - <https://www.bloodshed.net/>



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

- Open DevC++ IDE for C Project Generation **Open File->New->Project**



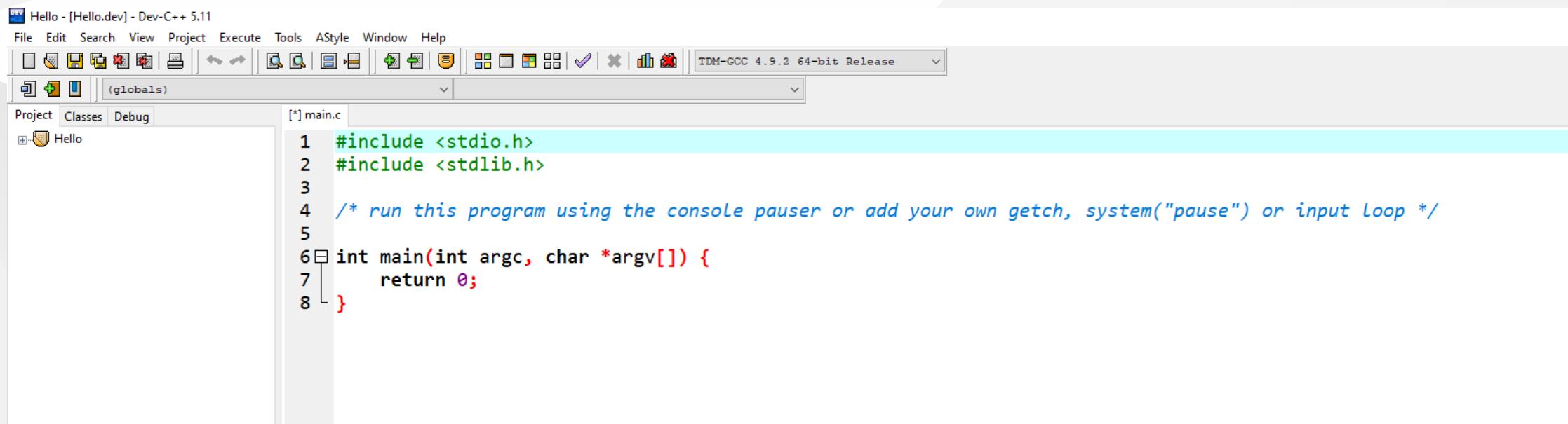
DevCpp (Install / Compile / Run / Debug) (3)

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 an 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 like Open, Save, and Build. The status bar indicates "TDM-GCC 4.9.2 64-bit Release". The left sidebar shows a project named "Hello" under the "Project" tab. The main editor window displays the "main.c" source 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 {
8     return 0;
}
```

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 the main function. This will write "Hello, World!" on the screen and then wait for a keypress to exit from the application

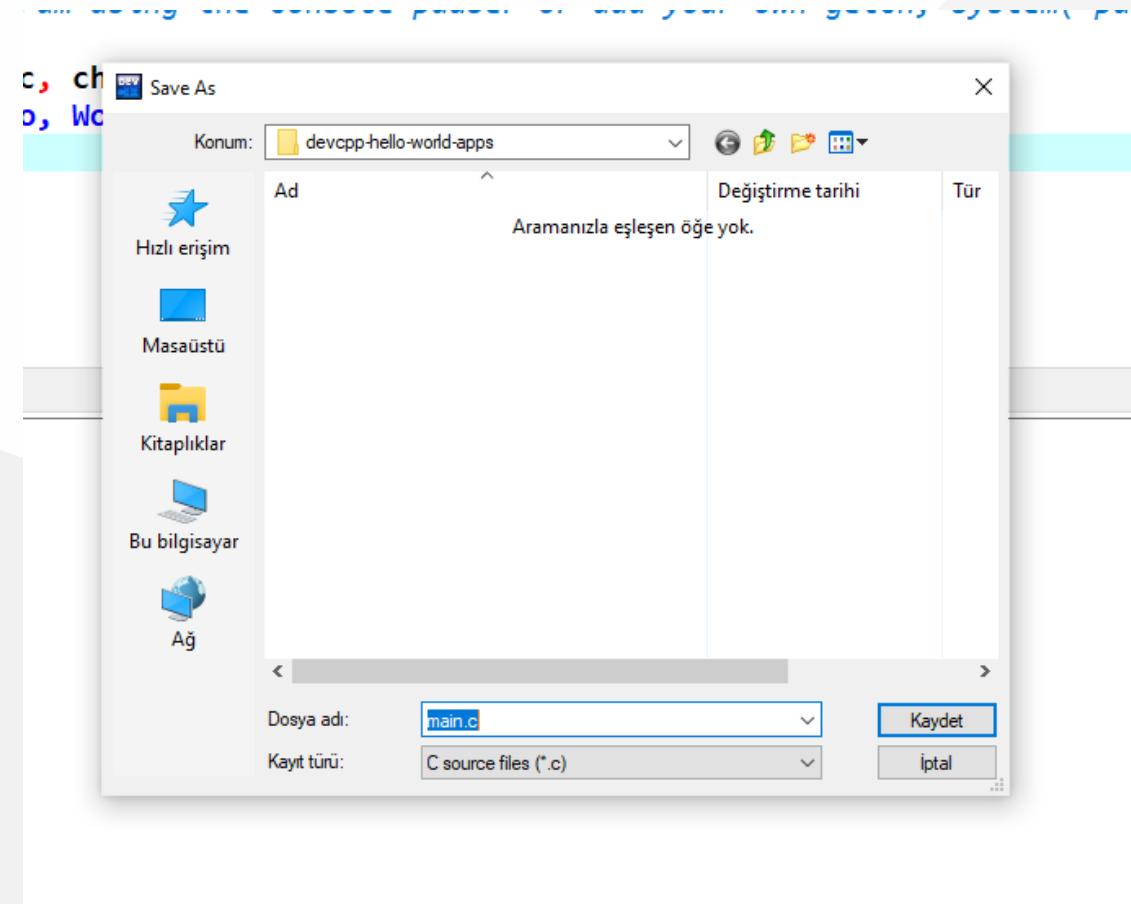
```
#include <stdio.h>
#include <stdlib.h>

/* 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!");
    getchar();
    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 shows the Dev-C++ IDE interface. The menu bar is visible at the top, and the main window displays the source code for a C program named 'Hello'. The code prints "Hello, World!" to the console. Below the code editor is a 'Compiler Log' window showing the compilation process. The log output includes the project filename, compiler name (TDM-GCC 4.9.2 64-bit Release), command-line arguments, and the resulting executable file 'Hello.exe'. The log also shows the number of errors and warnings, and the compilation time.

```
stdio.h>
stdlib.h>

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

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

Compiler Log
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/c++" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/4.9.2/include/c++/backward" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/4.9.2/include/c++/bits" -I"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/4.9.2/include/c++/support"
gcc.exe main.o -o Hello.exe -L"C:/Program Files (x86)/Dev-Cpp/MinGW64/lib" -LC:/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

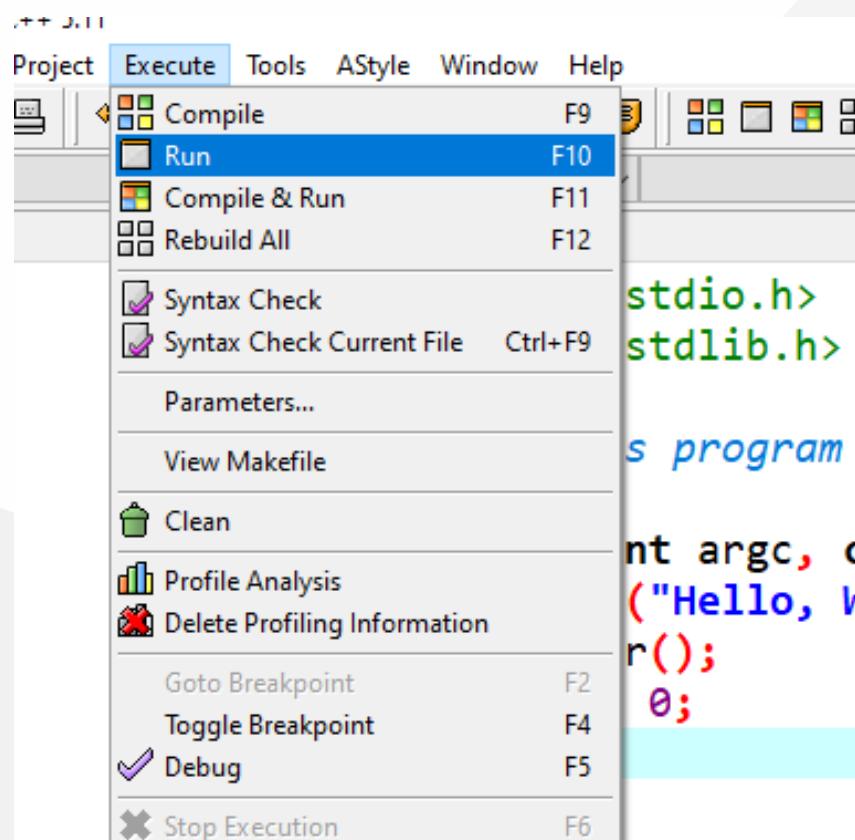
Processing makefile...
-----
- Makefile Processor: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\mingw32-make.exe
- Command: mingw32-make.exe -f "E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce103-algorithms-and-programming-I\Week-2\devcpp-hello-world-apps\Makefile.win"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

The screenshot shows the Dev-C++ IDE interface with a project named "Hello". The main window displays the code for "main.c". A red vertical bar highlights the line numbers 8 and 17, indicating where breakpoints have been set. The code itself is a "Hello, World!" program with additional comments and step printf statements.

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

/* run this program using the console pauser or add your own getch, system("pause") or exit(0) */

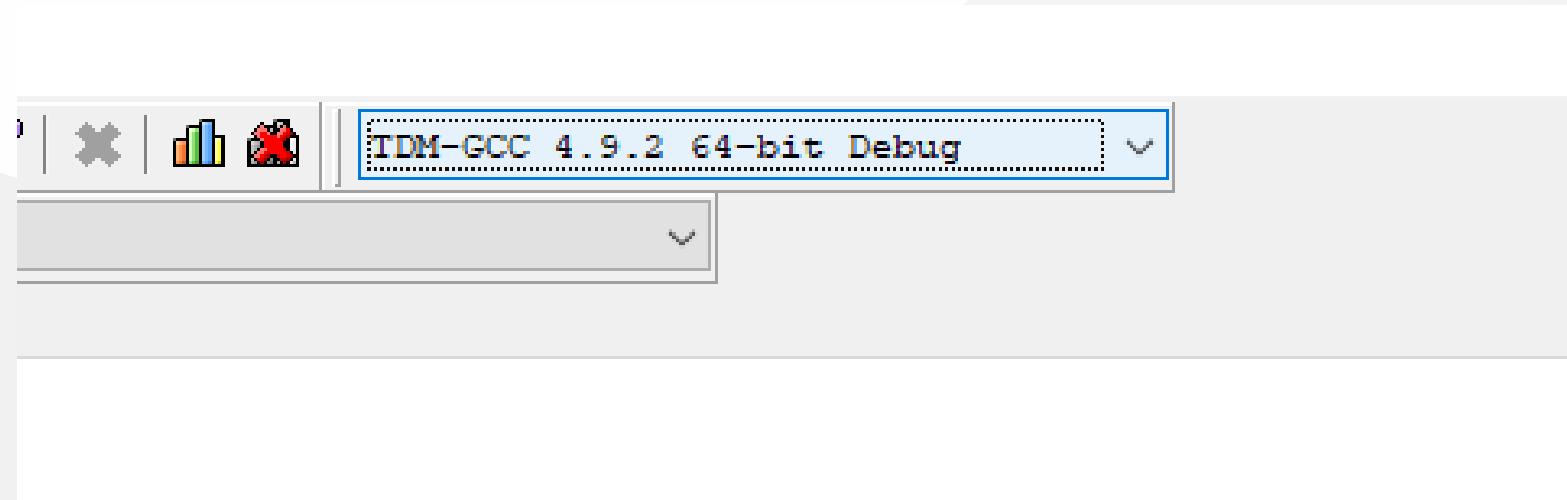
int main(int argc, char *argv[]) {
    printf("Hello, World! Step-1\n");
    printf("Hello, World! Step-2\n");
    printf("Hello, World! Step-3\n");
    printf("Hello, World! Step-4\n");
    printf("Hello, World! Step-5\n");
    printf("Hello, World! Step-6\n");

    getchar();

    return 0;
}
```

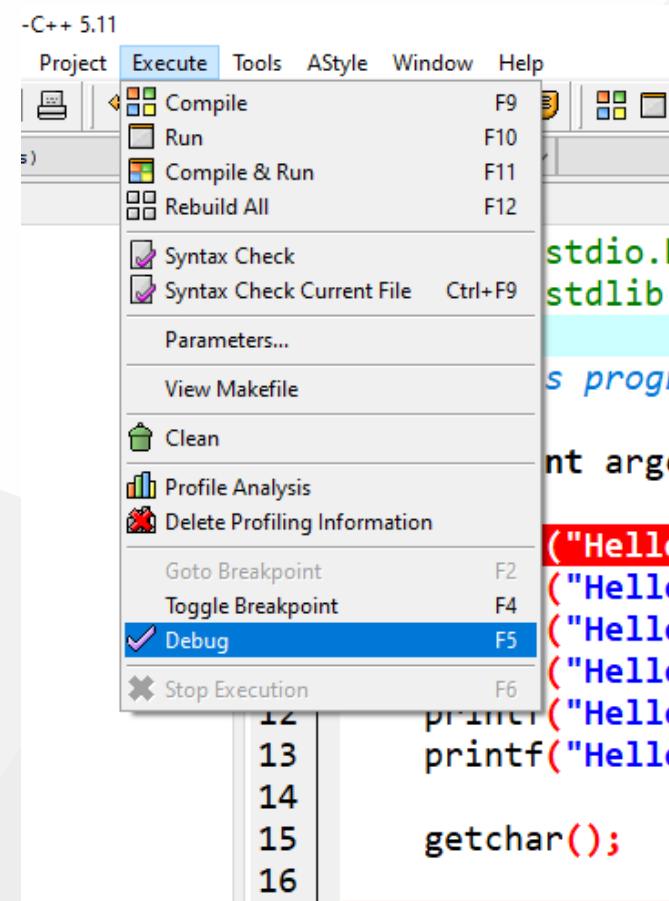
DevCpp (Install / Compile / Run / Debug) (13)

- In the menu section, select the compiler with debug option



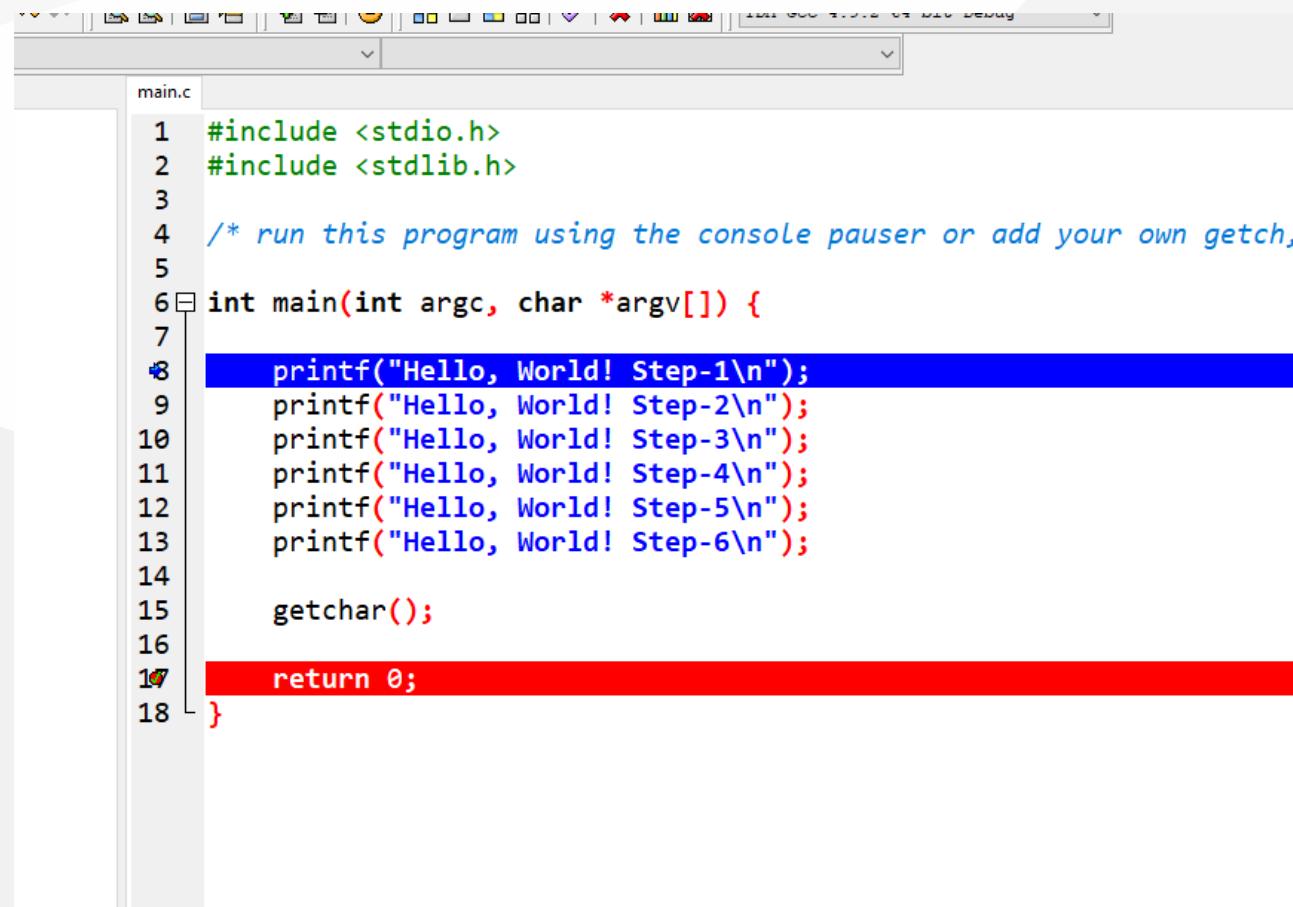
DevCpp (Install / Compile / Run / Debug) (14)

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



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

- The debugger will stop at the breakpoint at the debug point (blue line)

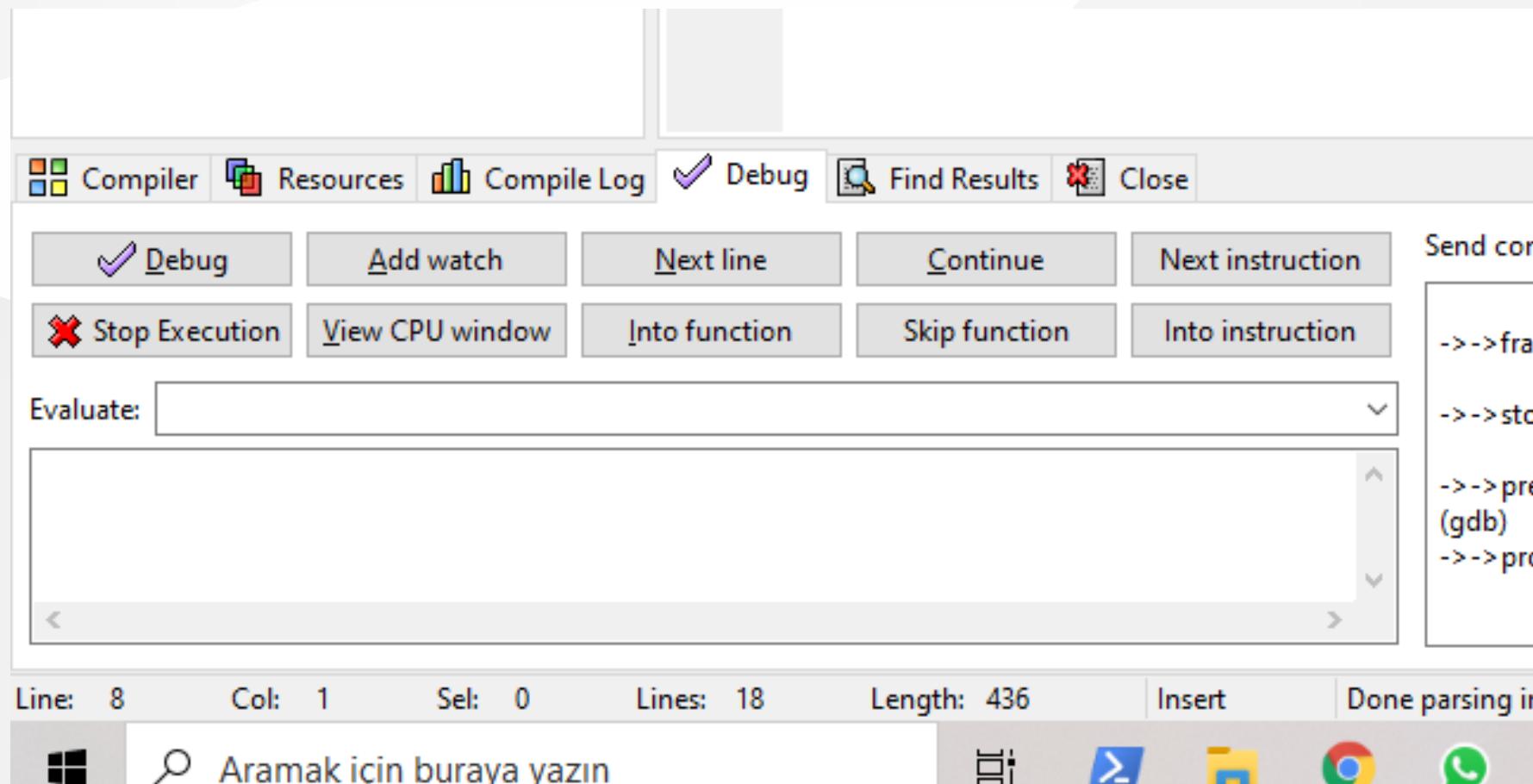


A screenshot of the DevCpp IDE interface. The main window shows a code editor with a file named "main.c". The code is a "Hello, World!" program with six printf statements and a getchar call. A breakpoint is set at the first printf statement (line 8), indicated by a blue horizontal bar under the line. The line numbers are on the left, and the code itself is in the center. The IDE has a standard Windows-style toolbar at the top.

```
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 }
```

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

- Moving to the 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, and 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 a 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

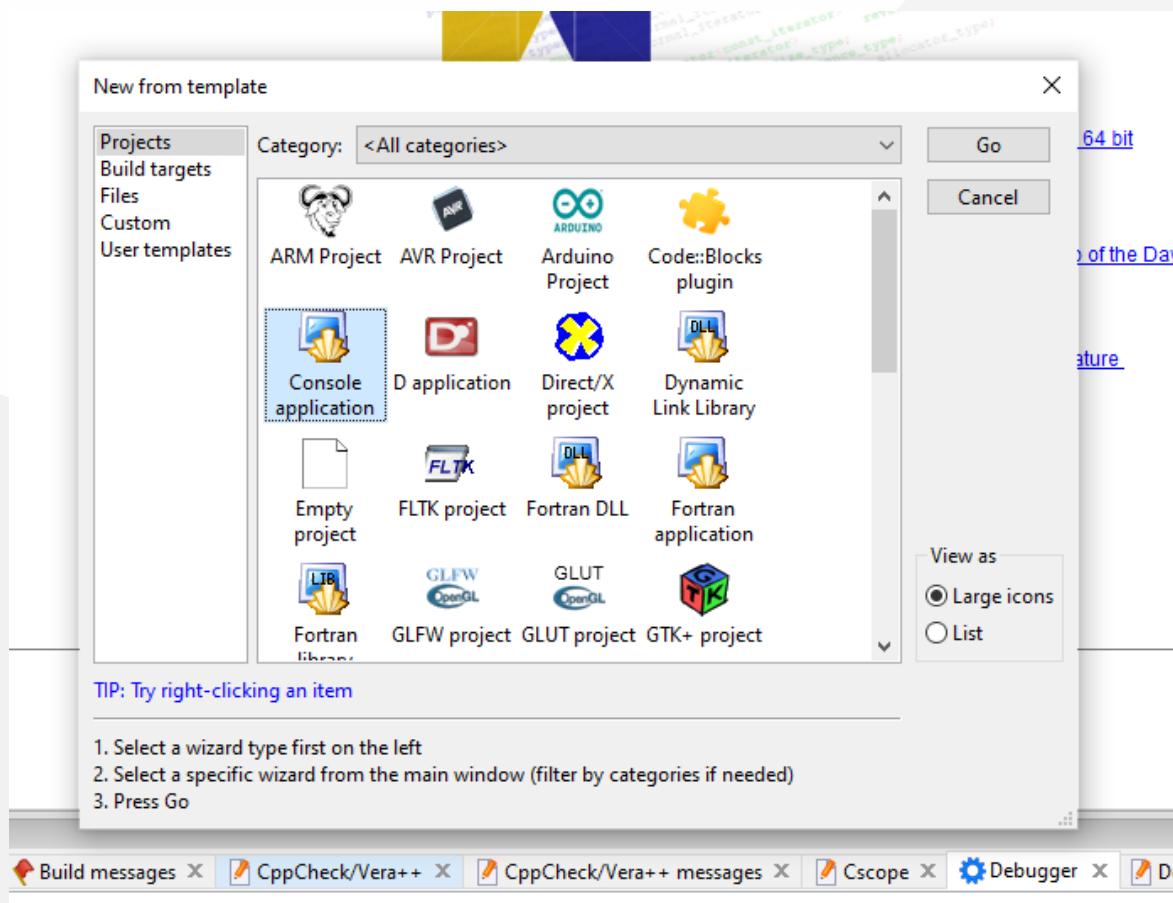
[Binary releases - Code::Blocks](#)



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

Open Code Blocks and

Select **File->New->Project**



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

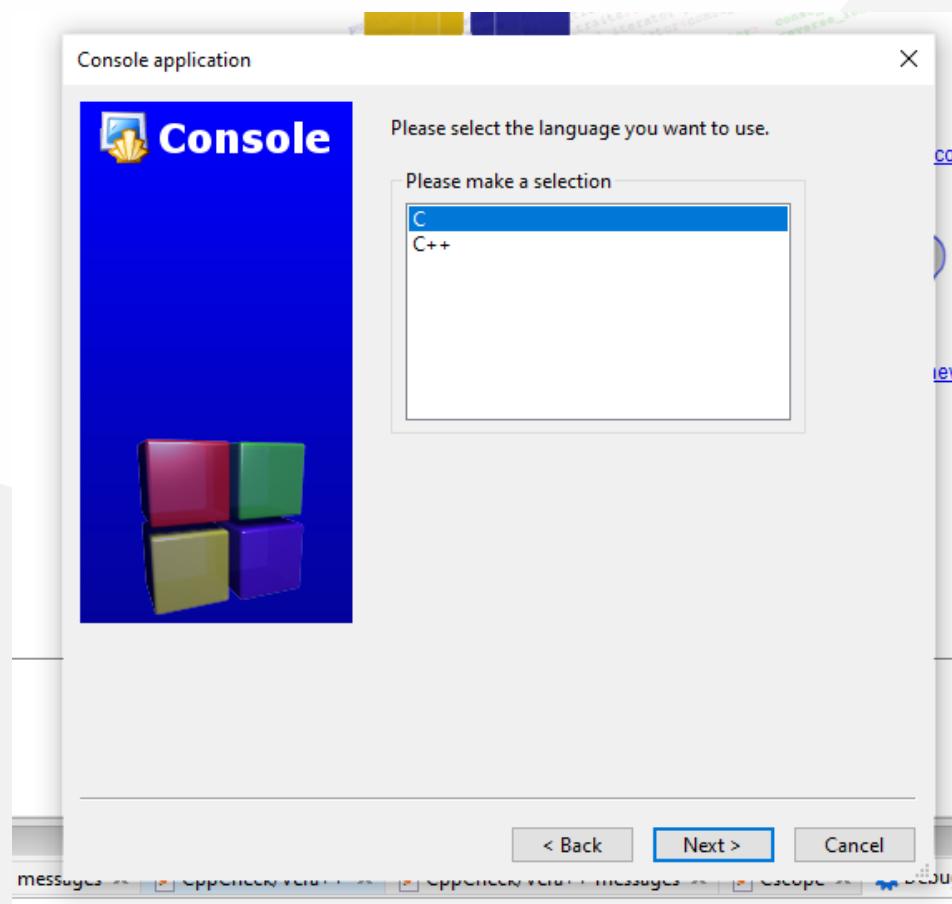
Select **Console Application**

Click **Next** from Opening Window



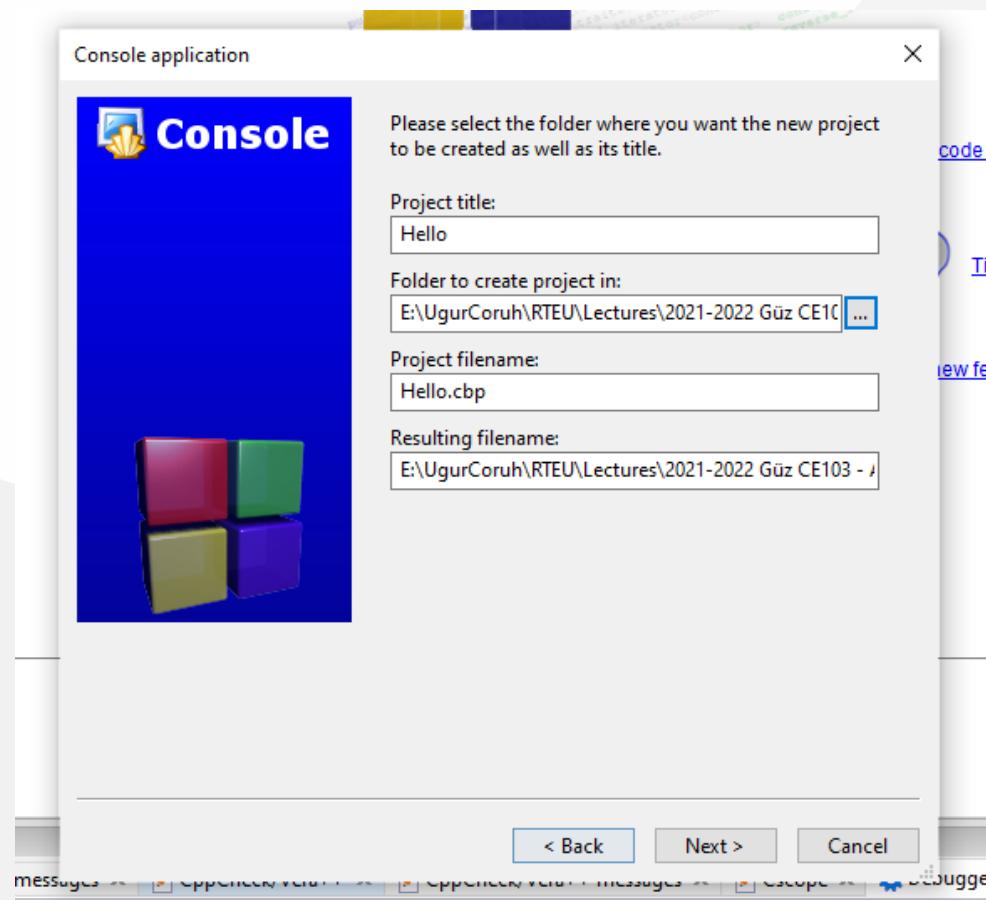
Code Blocks (Install / Compile / Run / Debug) (4)

Select **C** for Sample Project



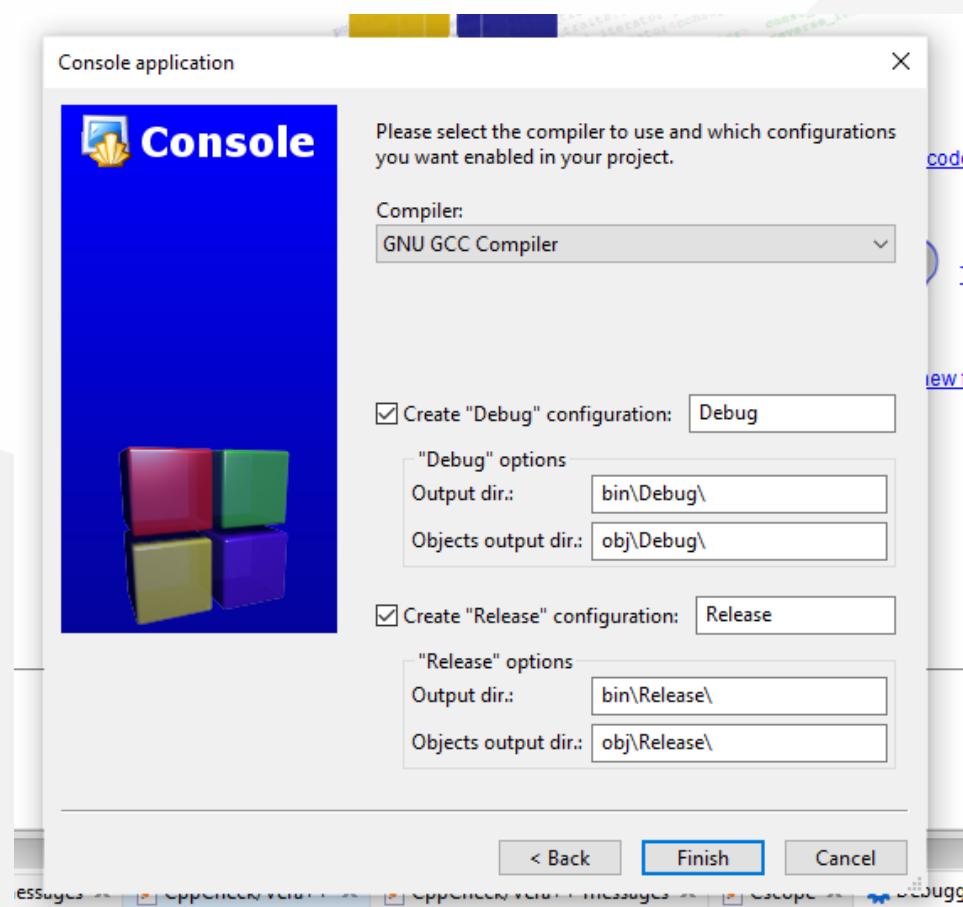
Code Blocks (Install / Compile / Run / Debug) (5)

Write a project name and title also set a project folder



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

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



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

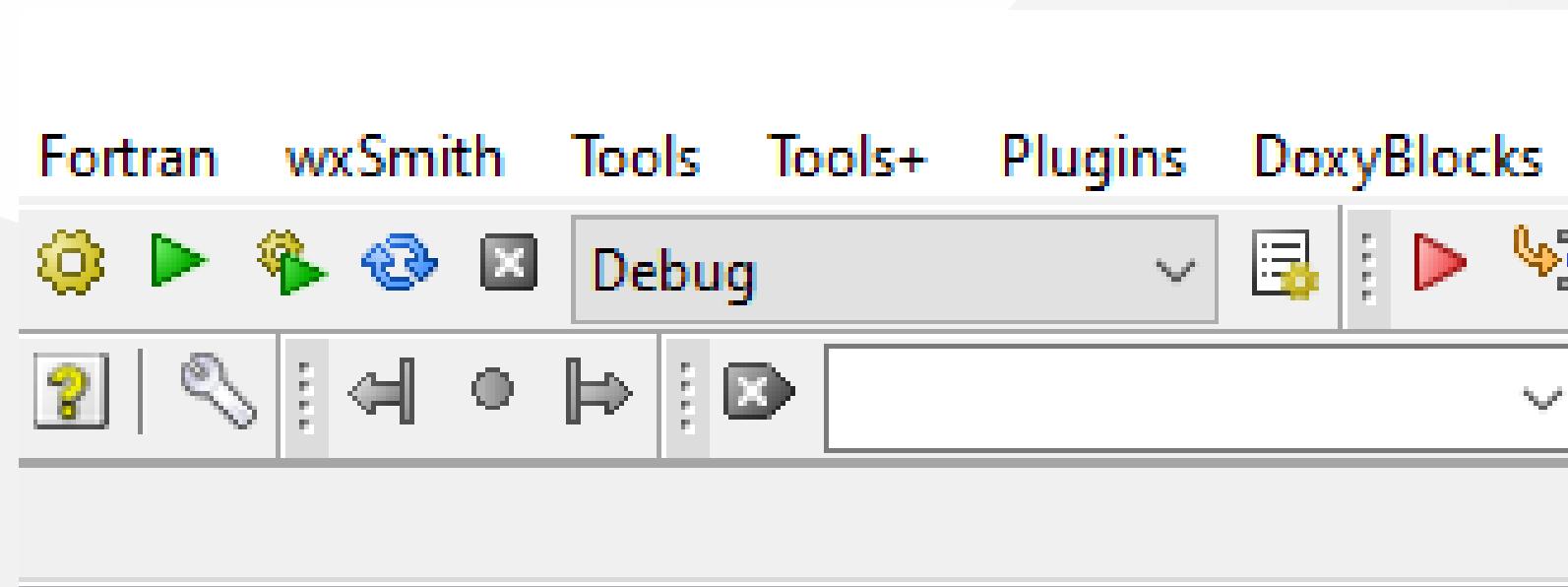
- After this wizard, you will have the following code

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

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

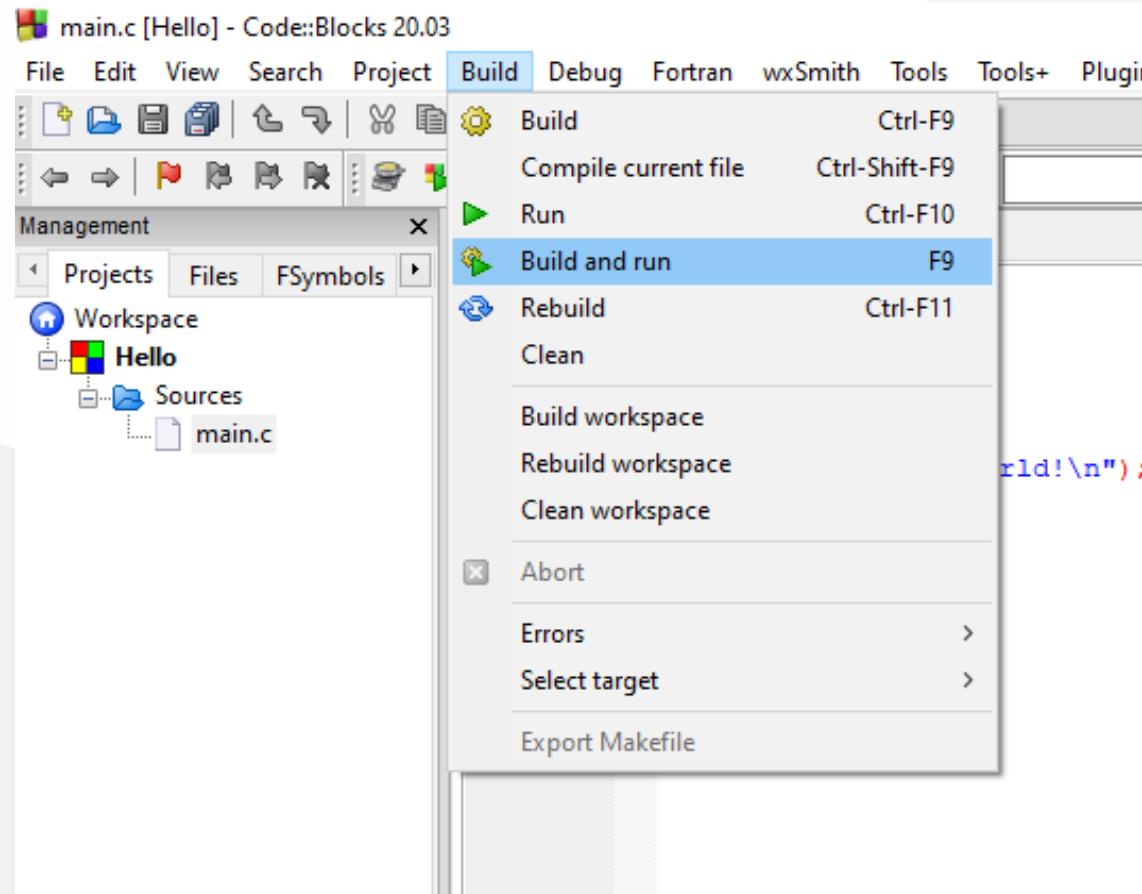
Code Blocks (Install / Compile / Run / Debug) (8)

Select **Debug** Build from the menu



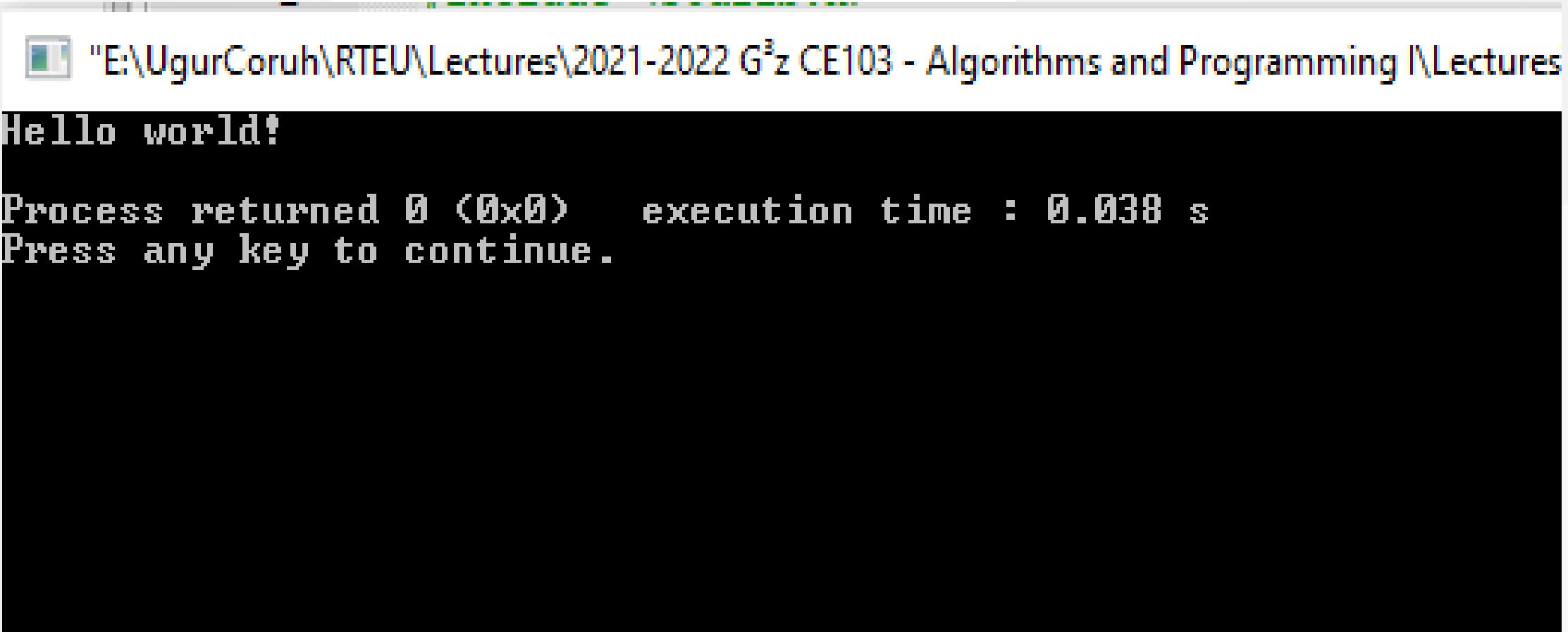
Code Blocks (Install / Compile / Run / Debug) (9)

Run with Build and Run F9



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

- You should see the following output



The screenshot shows a terminal window with the following text:

```
"E:\UgurCoruh\RTEU\Lessons\2021-2022 G3 CE103 - Algorithms and Programming\Lessons"
Hello world!
Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

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

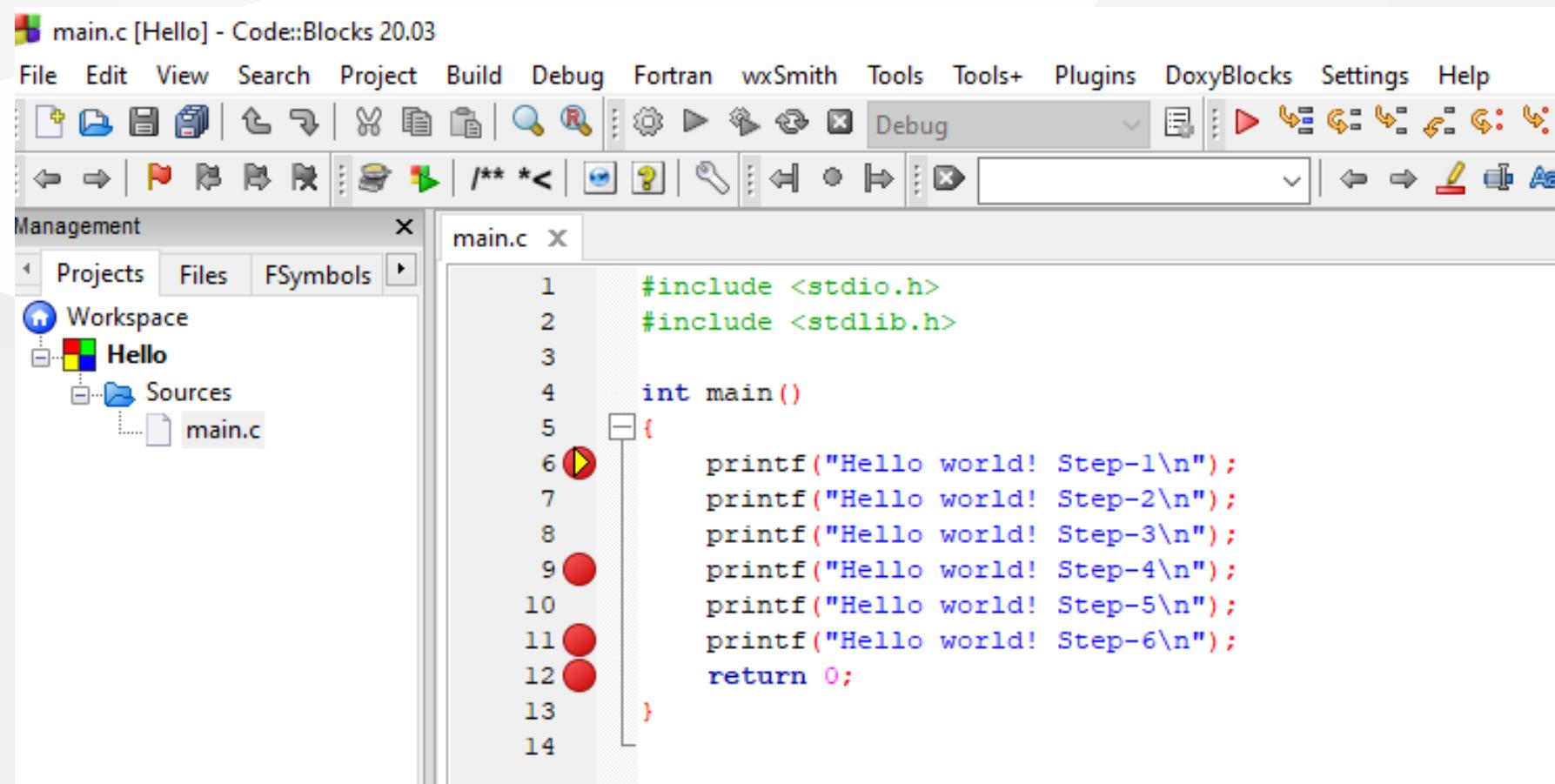
- Add the following lines to your source code for debugging

```
#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 breakpoints with F5 or mouse click



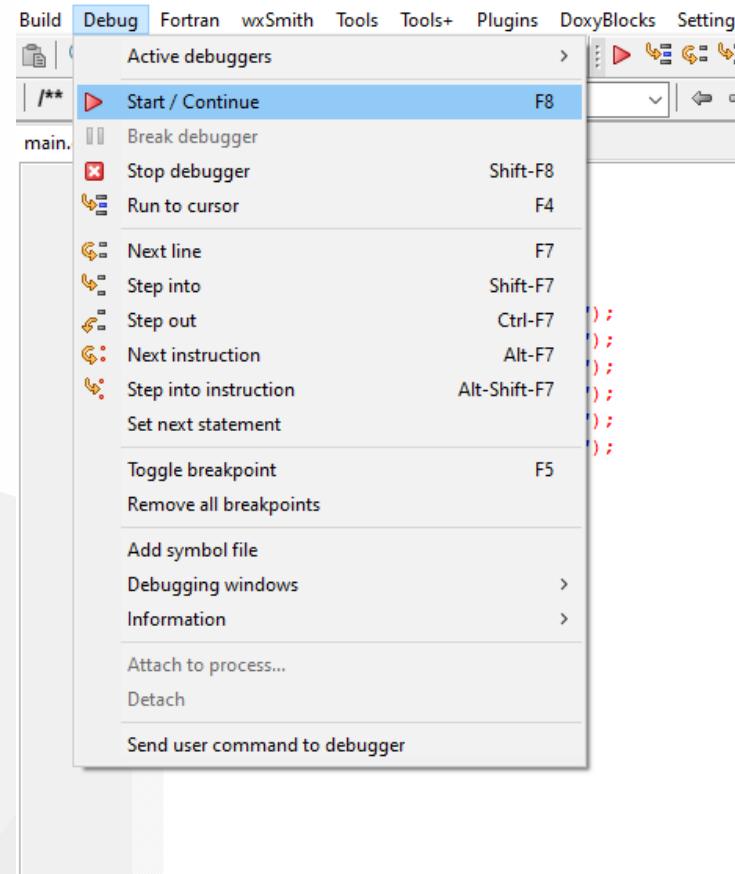
The screenshot shows the Code::Blocks 20.03 IDE interface. The title bar reads "main.c [Hello] - Code::Blocks 20.03". The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The toolbar has various icons for file operations like Open, Save, and Build. The "Debug" button is highlighted in the toolbar. The left sidebar is titled "Management" and shows the "Projects" tab selected, displaying a workspace named "Hello" which contains a "Sources" folder with a file named "main.c". The main editor window shows the code for "main.c":

```
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
```

A yellow circular breakpoint icon is placed on line 6, and three red circular icons are placed on lines 9, 10, and 11. The gutter on the left side of the code editor also features small colored circles corresponding to these 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 to long or turkish characters including the path. Just move the project to a shorter path and try again

```
Setting breakpoints
```

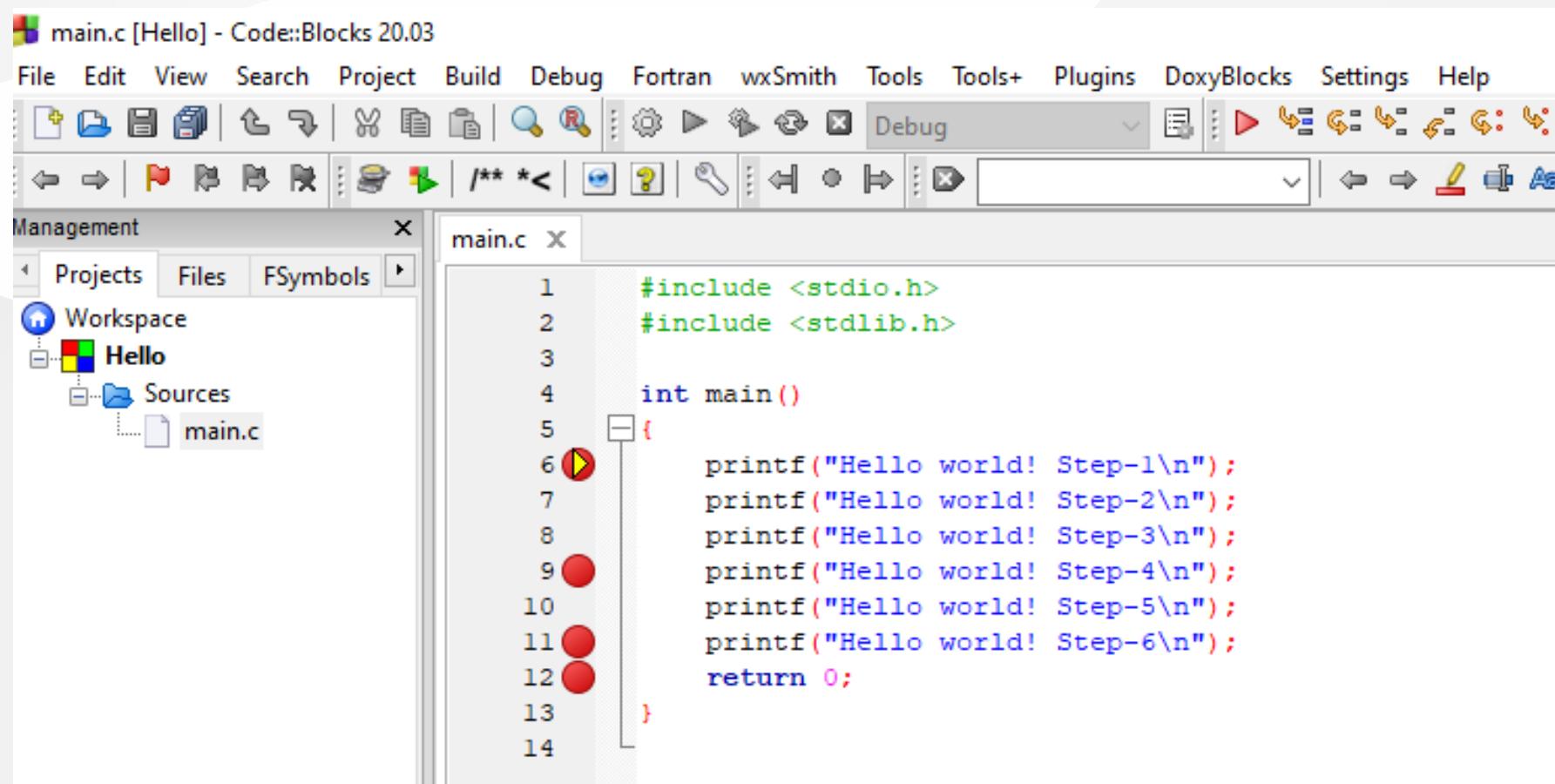
```
Debugger name and version: GNU gdb (GDB) 8.1
```

```
Starting the debugger 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 the debugger



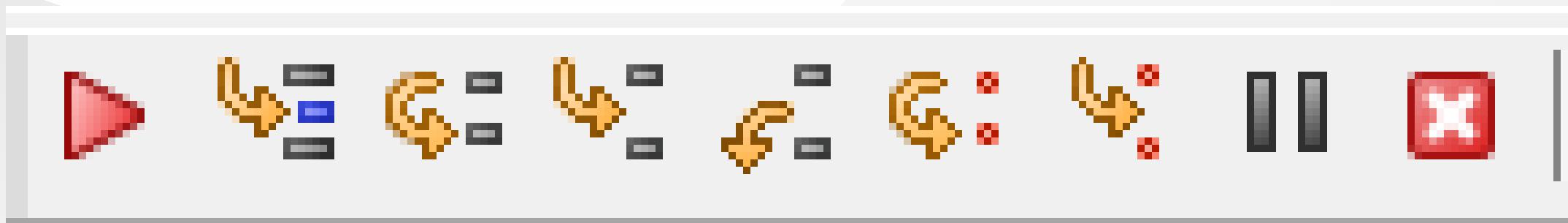
The screenshot shows the Code::Blocks 20.03 IDE interface. The title bar reads "main.c [Hello] - Code::Blocks 20.03". The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The toolbar has various icons for file operations like Open, Save, and Build. A "Management" dock window is open, showing the "Projects" tab with a workspace named "Hello" containing a source file "main.c". The main code editor window displays the following C 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 }
13
14 return 0;
```

A yellow circular debugger pointer is positioned over the opening brace of the main function at line 6. To the right of the code editor, there is a vertical column of red circular markers at lines 9, 10, 11, and 12, likely indicating breakpoints or error markers.

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

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



GCC/G++ Complier (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](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20-%20mingw-w64/branches/stable/)
- If you have a problem try `Github`` builds
 - <https://github.com/nixman/mingw-builds-binaries/releases>
 - https://github.com/nixman/mingw-builds-binaries/releases/download/12.2.0-rt_v10-rev0/x86_64-12.2.0-release-win32-seh-rt_v10-rev0.7z
- 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++ 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 to add gcc and g++ executable paths to the 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++ 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 path changes according to your setup

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

Download Visual Studio Code from the following link

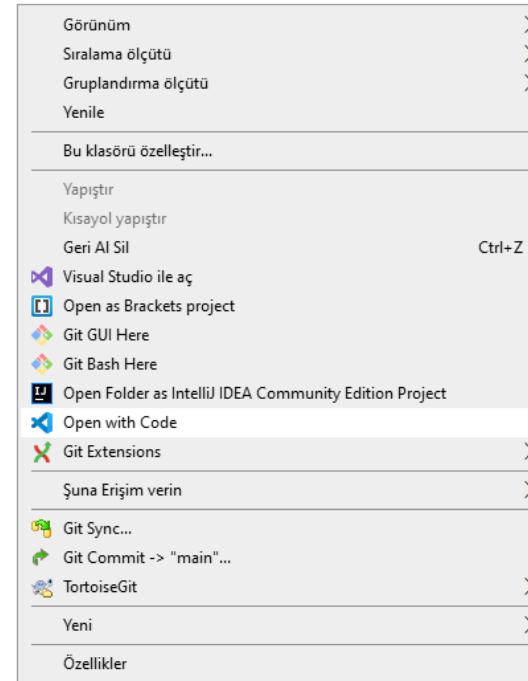
[Download Visual Studio Code - Mac, Linux, Windows](#)



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 CE103 - Algorithms and Programming I\\Lectures\\ce103-algorithms-and-programming-I\\Week-2\\vscode-hello-world-apps\\C-clang>code .
```

```
lo-w
```

```
lo-w
```

```
erson
```

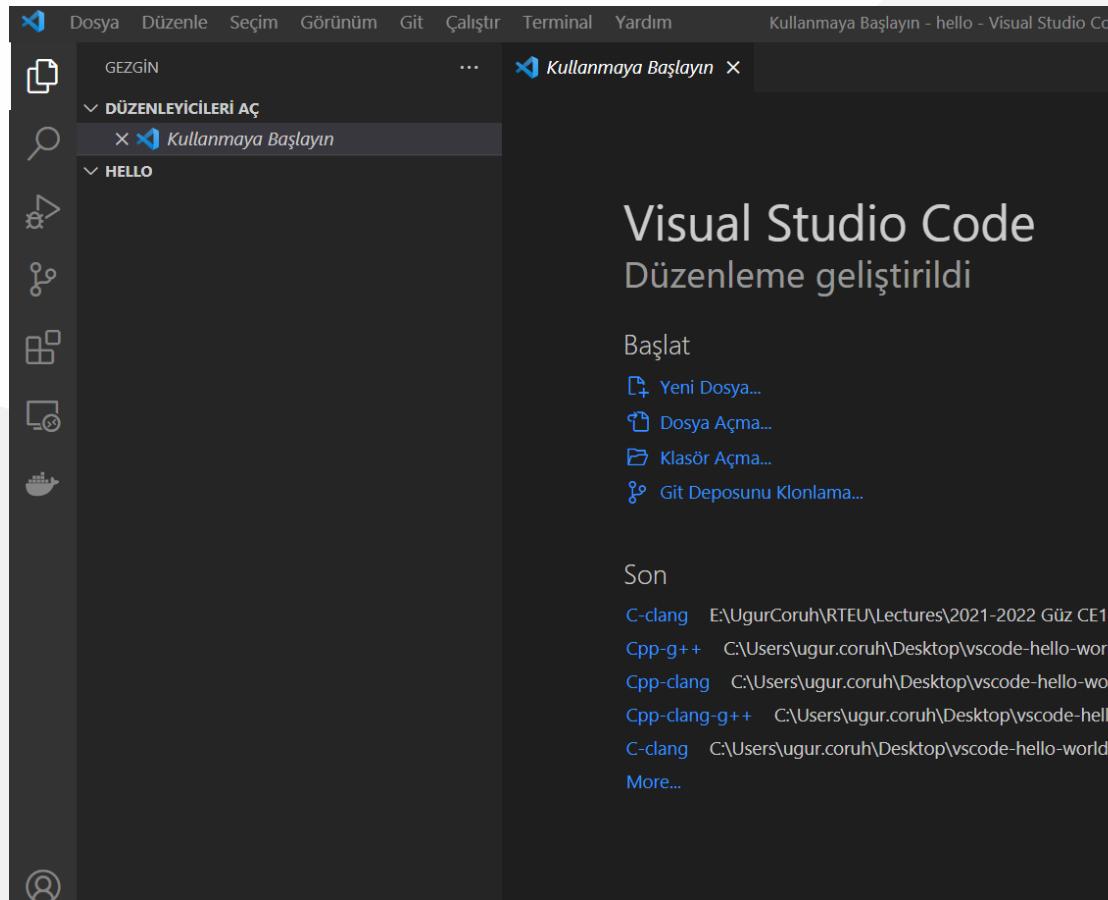
```
.
```

write

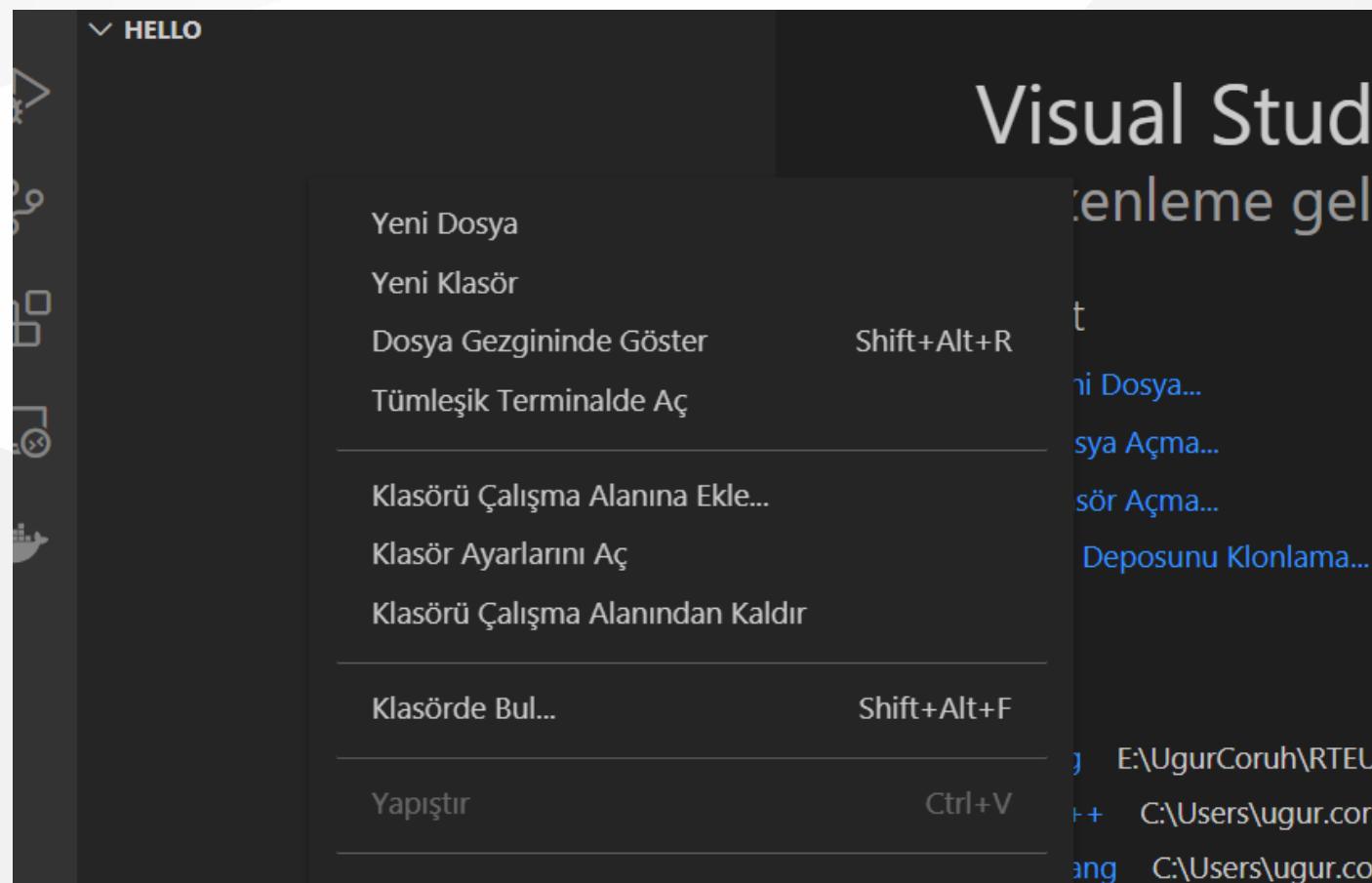
```
code .
```

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

- This will open vscode for the current folder, (.) dot present current folder.
- You will see an empty folder in the right window



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



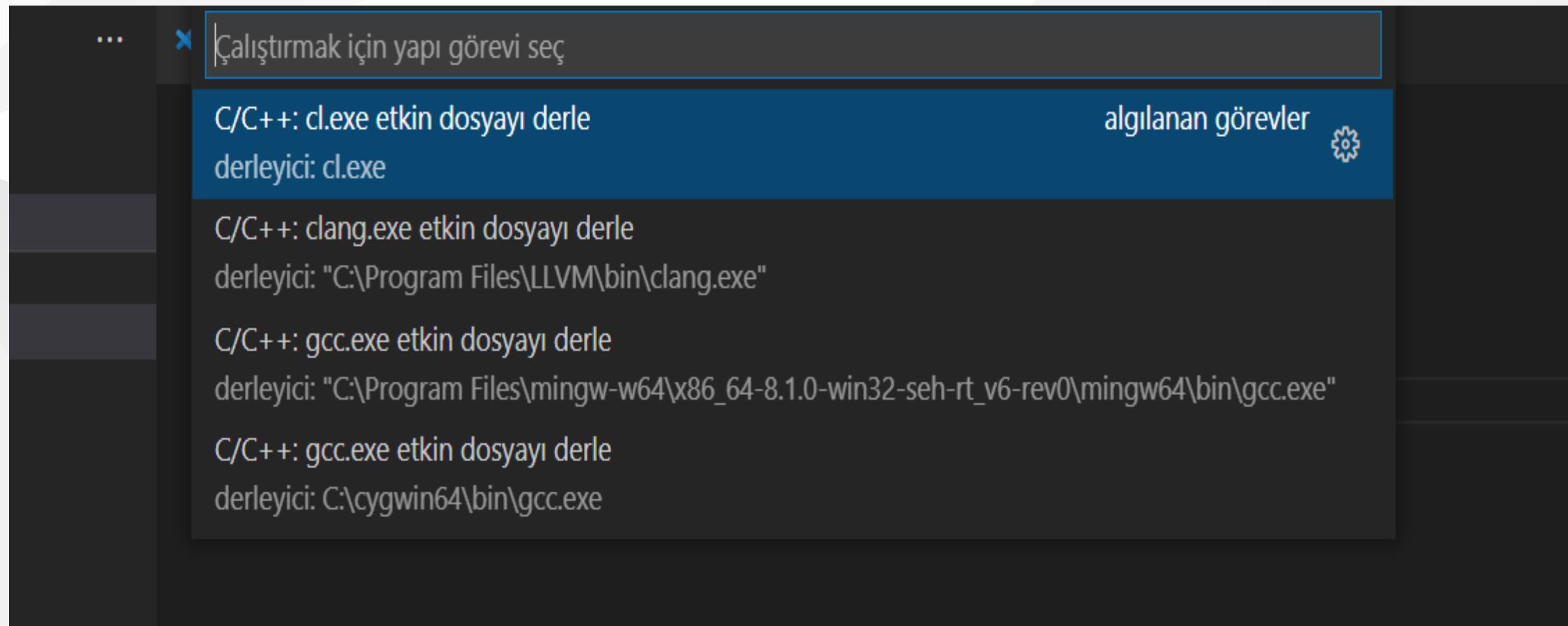
VSCode (Install / Compile / Run / Debug) (6)

- Create a `hello.c` file and write the following content

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

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

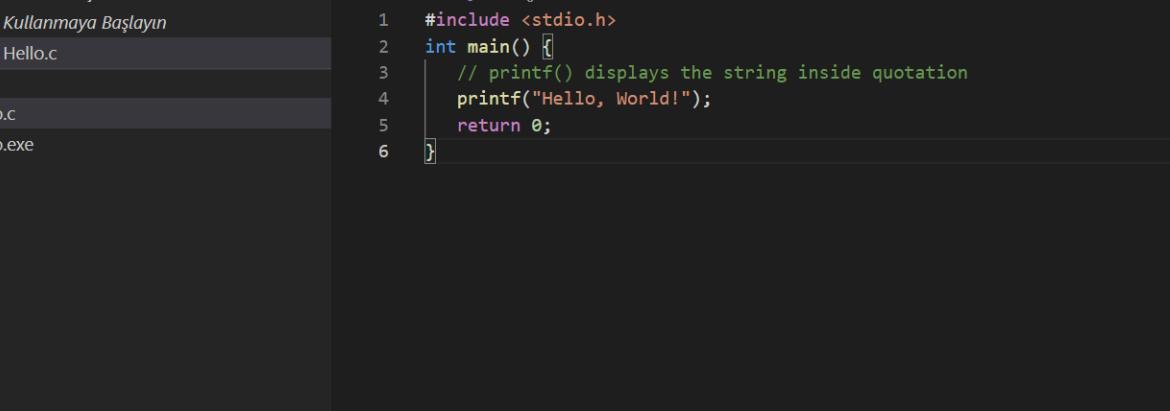
use **CTRL+SHIFT+B** (you should be on the source code section) to build a file



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

Select **GCC** or **CLANG** for this sample we can use **GCC**

You will see the output generated `Hello.exe`



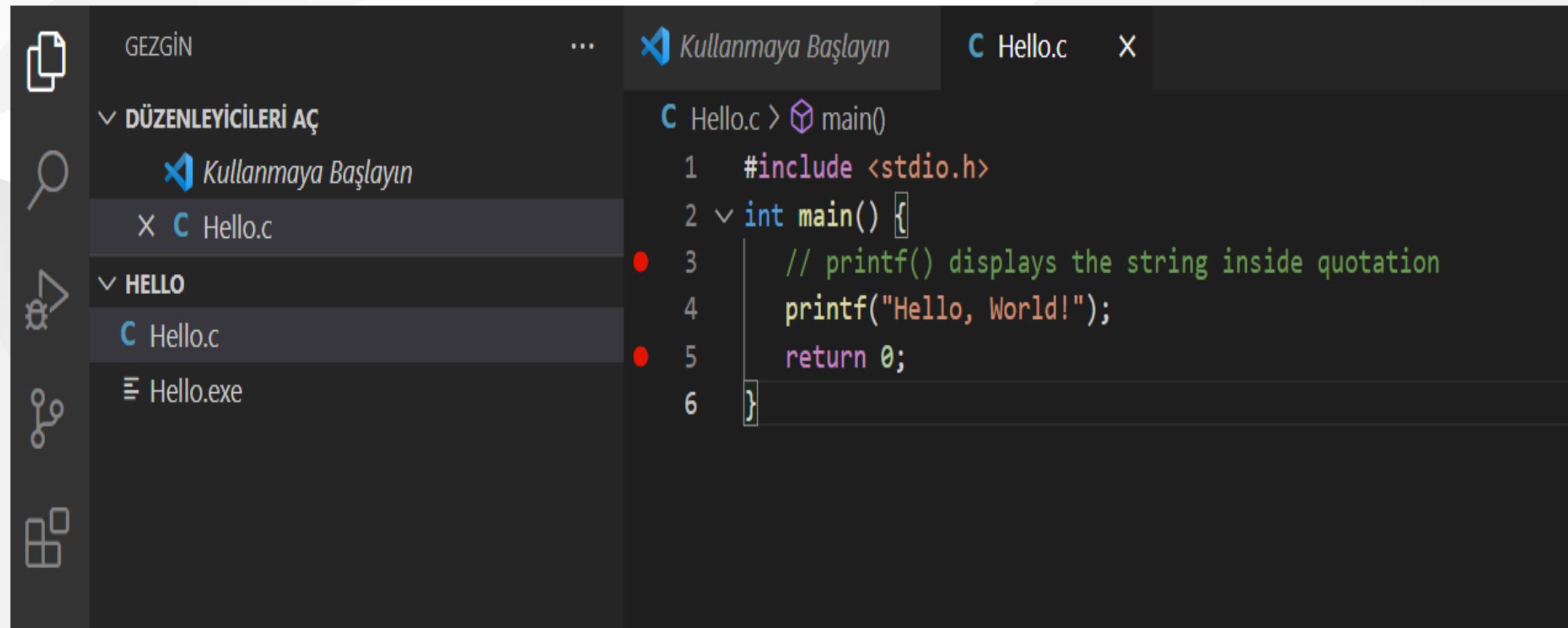
The screenshot shows the Visual Studio Code interface with the following details:

- Left Sidebar:** Shows a tree view with sections like "GEZİN", "DÜZENLEYİCİLERİ AÇ", and "HELLO". Under "HELLO", there is a file "Hello.c" and an executable "Hello.exe".
- Central Area:** A code editor window titled "Hello.c - hello - Visual Studio Code" displays the following C code:

```
#include <stdio.h>
int main() {
    // printf() displays the string inside quotation
    printf("Hello, World!");
    return 0;
}
```
- Bottom Navigation Bar:** Includes tabs for "SORUNLAR", "ÇIKIŞ", "HATA AYIKLAMA KONSOLU", and "TERMINAL".
- Bottom Status Bar:** Shows the message "Executing task: C/C++: gcc.exe etkin dosyayı derle <" followed by the output of the compilation process.

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

for debugging just put a breakpoint and build again



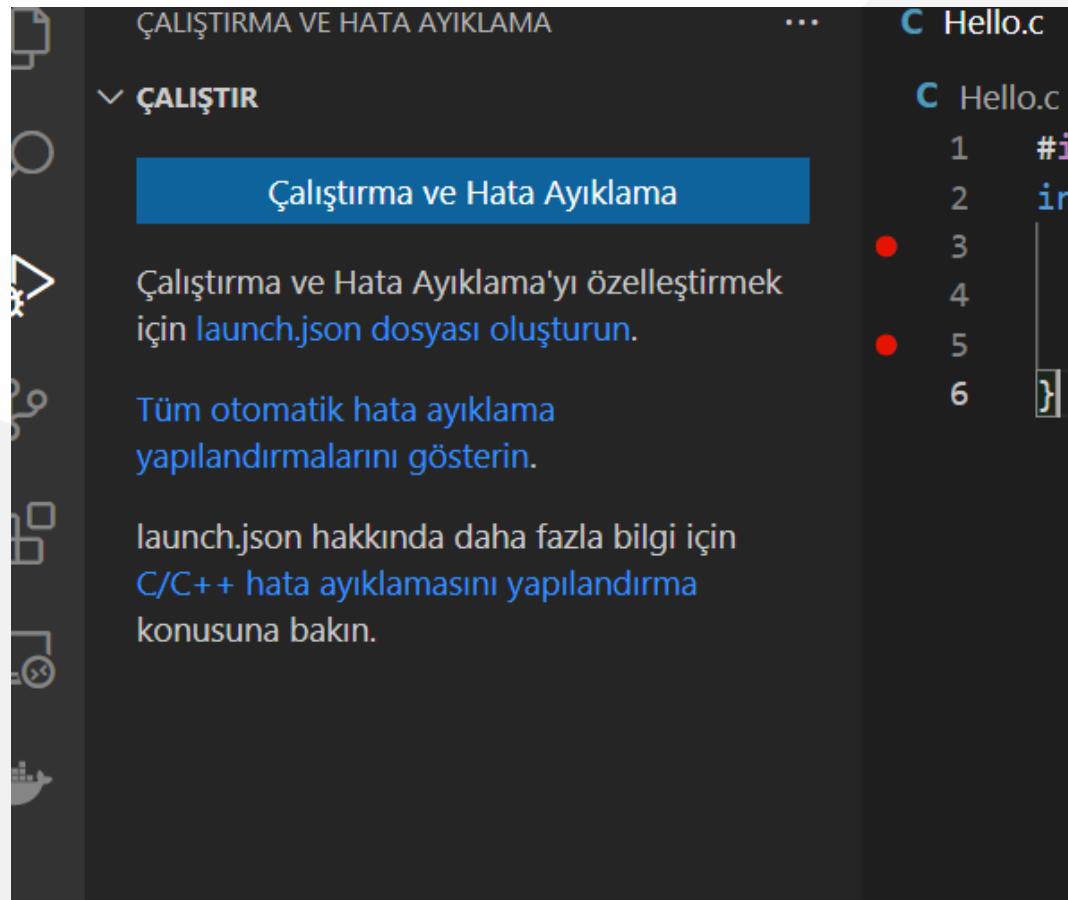
The screenshot shows the Visual Studio Code (VSCode) interface. On the left is the sidebar with icons for Explorer, Search, Problems, and Terminal. The main area shows a dark-themed code editor with a C file named "Hello.c". The code contains the following:

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

Two red circular markers with a dot inside are placed on the second line of the code, indicating breakpoints. The status bar at the bottom of the code editor shows the path "Hello.c > main()".

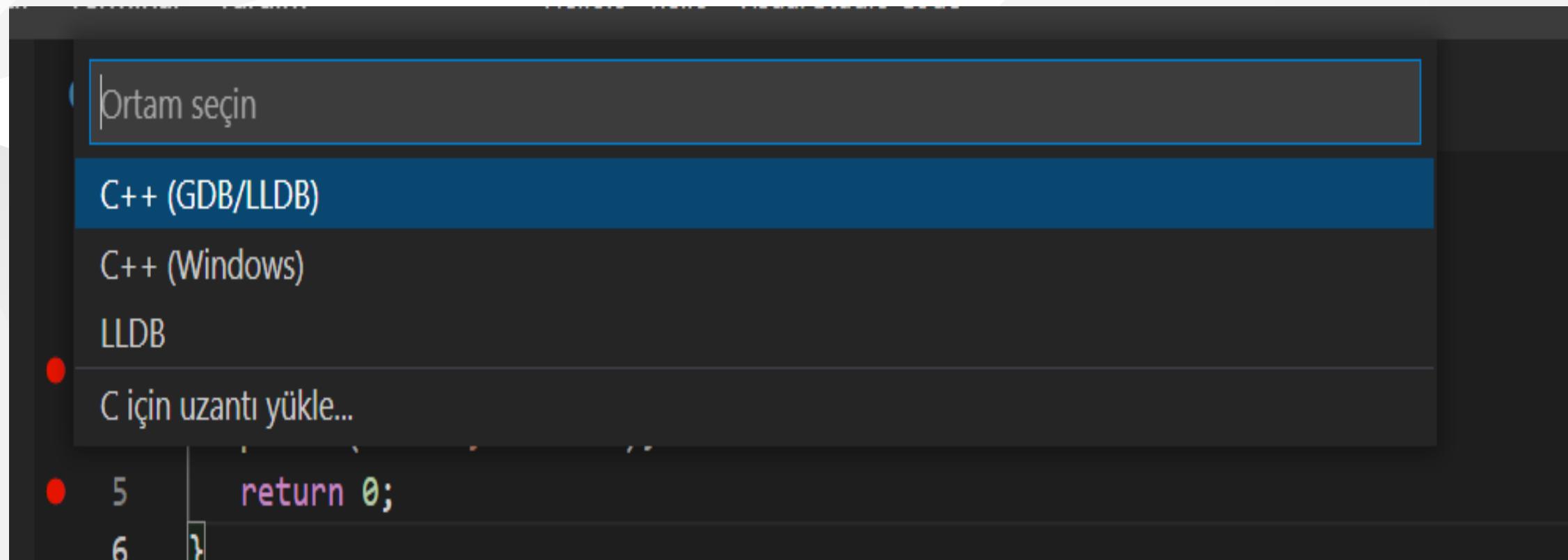
VSCode (Install / Compile / Run / Debug) (10)

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



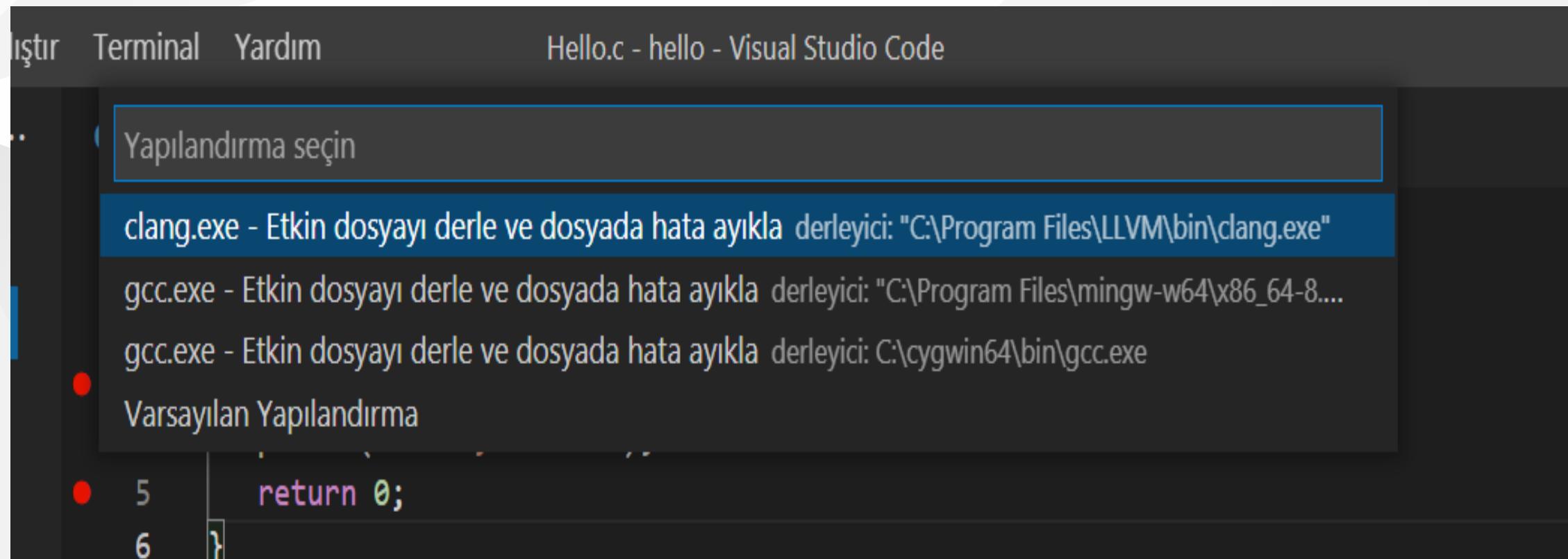
VSCode (Install / Compile / Run / Debug) (11)

- from opening, window select C++ GDB/LLDB



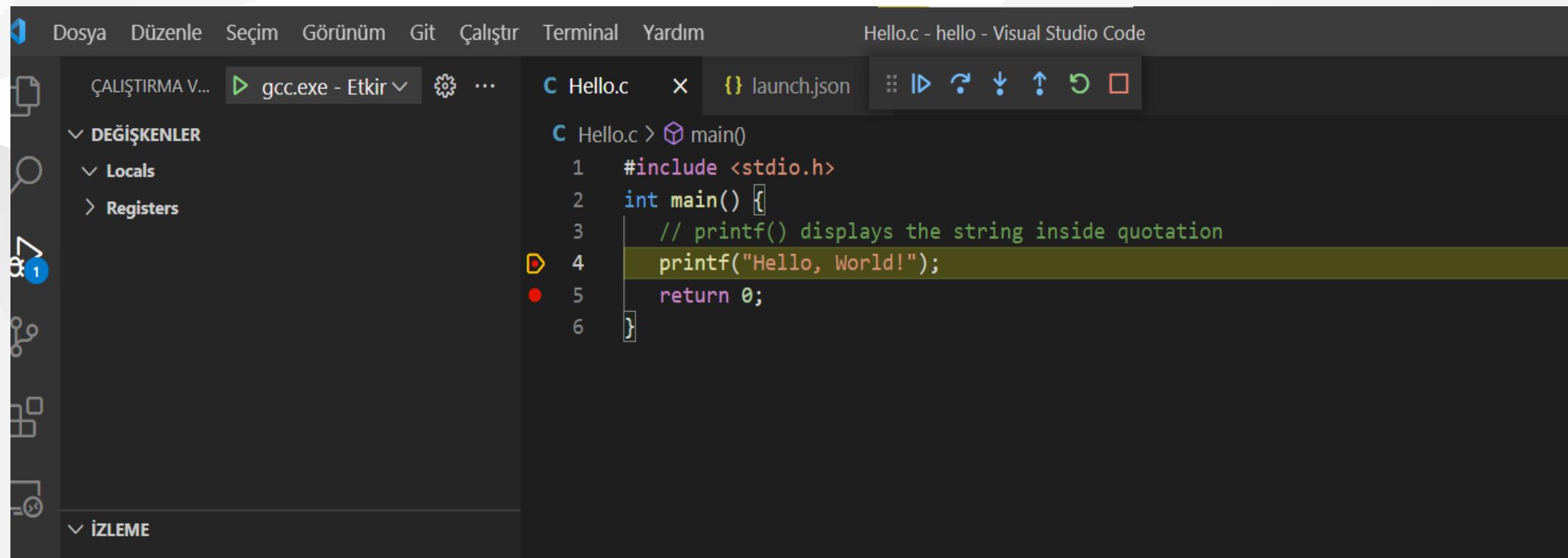
VSCode (Install / Compile / Run / Debug) (12)

- from the next opening, menu select mingw-w64 gcc.exe



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

this will run the debugger and you will see debug points activated



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "ÇALIŞTIRMA V..." and a "DEĞİŞKENLER" (Variables) section expanded, containing "Locals" and "Registers".
- Editor:** Displays the "Hello.c" file content:

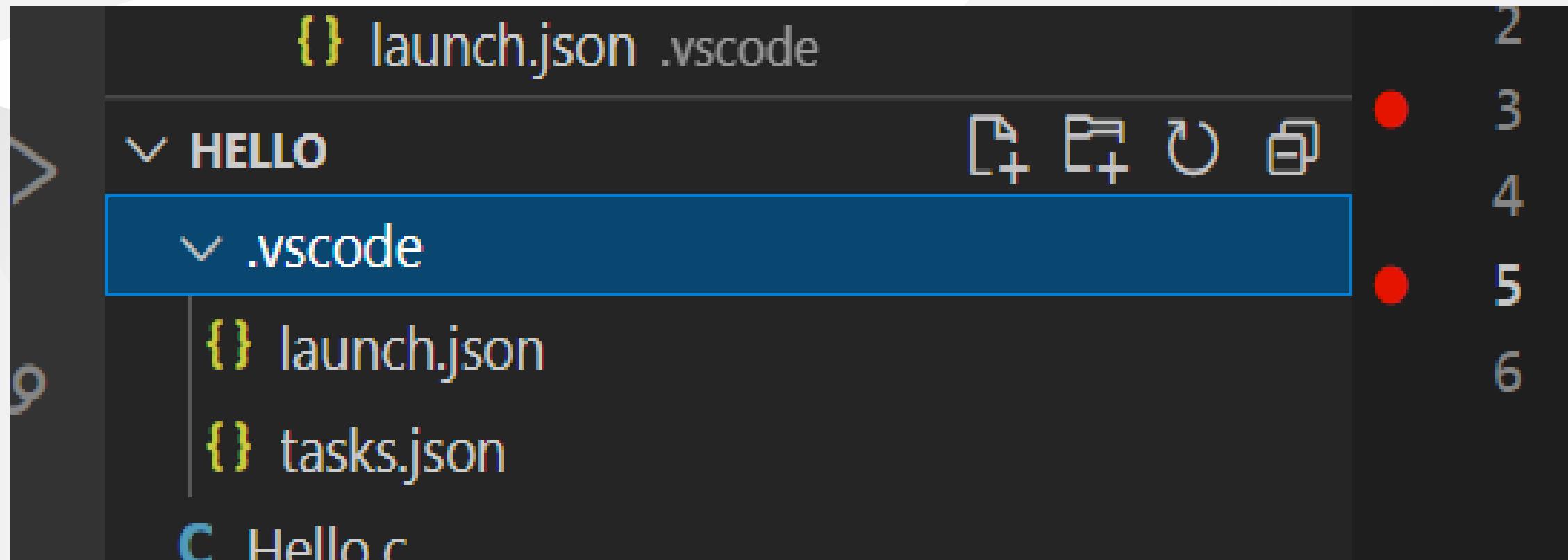
```
1 #include <stdio.h>
2 int main() {
3     // printf() displays the string inside quotation
4     printf("Hello, World!");
5     return 0;
6 }
```

A yellow rectangular selection highlights the entire body of the `main()` function.
- Terminal:** Shows the command "gcc.exe - Etkir" and the file path "Hello.c".
- Launch Configuration:** The "launch.json" file is shown in the editor.
- Debug Sidebar:** Shows a list of breakpoints: a yellow circle at line 4 and a red circle at line 5.

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 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\\bin\\gdb.exe",  
            "setupCommands": [  
                {  
                    "description": "gdb için düzgün yazdırmayı etkinleştir",  
                    "text": "-enable-pretty-printing",  
                    "ignoreFailures": true  
                }  
            ],  
            "preLaunchTask": "C/C++: gcc.exe etkin dosyayı derle"  
        }  
    ]  
}
```

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 does not support debugging 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 write Debug and Selecting Configure Debugger Option)

VSCode (Install / Compile / Run / Debug) (19)

- the 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 Code Extension List (1)

My Extension List

- Listing Installed Extensions

```
PS C:\Users\ugur.coruh\Desktop> code --list-extensions | % { "code --install-extension $_" }
```

Following topic can help you

[How can you export the Visual Studio Code extension list? - Stack Overflow](#)

Visual Studio Code Extension List (2)

```
code --install-extension 2gua.rainbow-brackets
code --install-extension aaron-bond.better-comments
code --install-extension abusaidm.html-snippets
code --install-extension ACharLuk.easy-cpp-projects
code --install-extension chris-noring.node-snippets
code --install-extension cschlosser.doxdocgen
code --install-extension csholmq.excel-to-markdown-table
code --install-extension DaChuiOpenSource.FreeMind
code --install-extension dannysteenman.cloudformation-yaml-snippets
code --install-extension Dart-Code.dart-code
code --install-extension Dart-Code.flutter
code --install-extension digized.umple
code --install-extension DotJoshJohnson.xml
code --install-extension DougFinke.vscode-pandoc
code --install-extension dzhavat.bracket-pair-toggler
code --install-extension esbenp.prettier-vscode
code --install-extension formulahendry.dotnet
code --install-extension franneck94.c-cpp-runner
code --install-extension gcc.
```

Visual Studio Code Extension List (3)

```
vscode-plugin-billionbottle
code --install-extension geeklearningio.graphviz-markdown-preview
code --install-extension geyao.html-snippets
code --install-extension GitHub.copilot-nightly
code --install-extension GrapeCity.gc-excelviewer
code --install-extension Ionide.Ionide-fsharp
code --install-extension ionut-botizan.vscode-cypher-ql
code --install-extension ipedrazas.kubernetes-snippets
code --install-extension JakeWilson.vscode-picture
code --install-extension James-Yu.latex-workshop
code --install-extension JasonMejane.base64viewer
code --install-extension jasonnutter.search-node-modules
code --install-extension jebbs.plantuml
code --install-extension jeff-hykin.better-cpp-syntax
code --install-extension Katacoda.vscode
code --install-extension KenDomino.Antlrvsix-vscode
code --install-extension l7ssha.tag-inserter
code --install-extension lolkush.quickstart
code --install-extension marp-team.marp-vscode
code --install-extension mindaro-dev.file-downloader
code --install-extension mindaro.mindaro
code --install-extension ms-azuretools.vscode-docker
code --install-extension MS-CEINTL.vscode-language-pack-tr
```

Visual Studio Code Extension List (4)

```
code --install-extension ms-dotnettools.csharp
code --install-extension ms-dotnettools.dotnet-interactive-vscode
code --install-extension ms-dotnettools.vscode-dotnet-pack
code --install-extension ms-dotnettools.vscode-dotnet-runtime
code --install-extension ms-kubernetes-tools.vscode-aks-tools
code --install-extension ms-kubernetes-tools.vscode-kubernetes-tools
code --install-extension ms-python.python
code --install-extension ms-python.vscode-pylance
code --install-extension ms-toolsai.jupyter
code --install-extension ms-toolsai.jupyter-keymap
code --install-extension ms-toolsai.jupyter-renderers
code --install-extension ms-vscode-remote.remote-containers
code --install-extension ms-vscode-remote.remote-ssh
code --install-extension ms-vscode-remote.remote-ssh-edit
code --install-extension ms-vscode-remote.remote-wsl
```

Visual Studio Code Extension List (5)

```
code --install-extension ms-vscode.azure-account
code --install-extension ms-vscode.brackets-keybindings
code --install-extension ms-vscode.brackets-pack
code --install-extension ms-vscode.cmake-tools
code --install-extension ms-vscode.cpptools
code --install-extension ms-vscode.cpptools-extension-pack
code --install-extension ms-vscode.cpptools-themes
code --install-extension ms-vscode.live-server
code --install-extension ms-vsliveshare.vsliveshare
code --install-extension oleg-shilo.cs-script
code --install-extension PascalReitermann93.vscode-yaml-sort
```

Visual Studio Code Extension List (6)

```
code --install-extension Pivotal.vscode-boot-dev-pack
code --install-extension Pivotal.vscode-concourse
code --install-extension Pivotal.vscode-manifest-yaml
code --install-extension Pivotal.vscode-spring-boot
code --install-extension PKief.material-icon-theme
code --install-extension platformio.platformio-ide
code --install-extension pranaygp.vscode-css-peek
code --install-extension redhat.fabric8-analytics
code --install-extension redhat.java
code --install-extension redhat.vscode-commons
code --install-extension redhat.vscode-xml
code --install-extension redhat.vscode-yaml
code --install-extension ritwickdey.LiveServer
code --install-extension sidthesloth.html5-boilerplate
code --install-extension TaodongWu.ejs-snippets
code --install-extension tht13.python
code --install-extension tomoki1207.pdf
code --install-extension twxs.cmake
code --install-extension vadimcn.vscode-lldb
```

Visual Studio Code Extension List (7)

```
code --install-extension VisualStudioExptTeam.intellicode-api-usage-examples
code --install-extension VisualStudioExptTeam.vscodeintellicode
code --install-extension vscjava.vscode-java-debug
code --install-extension vscjava.vscode-java-dependency
code --install-extension vscjava.vscode-java-pack
code --install-extension vscjava.vscode-java-test
code --install-extension vscjava.vscode-maven
code --install-extension vscjava.vscode-spring-boot-dashboard
code --install-extension vscjava.vscode-spring-initializr
code --install-extension walkme.HTML5-extension-pack
code --install-extension webfreak.debug
code --install-extension well-ar.plantuml
code --install-extension wildboar.asn1
code --install-extension Zignd.html-css-class-completion
```

Visual Studio Community Edition (Install / Compile / Run / Debug) (1)

- Download and install [visual Studio Community Edition](#)
- Select All Development Environments from Installer.

Ücretsiz Geliştirici Yazılımları ve Hizmetleri - Visual Studio



Visual Studio Community

Windows | Sürüm 17.2

Windows kullanan .NET ve C++ geliştiricileri için en iyi kapsamlı IDE. Yazılım geliştirmenin her aşamasını yükseltip geliştiren bir dizi güzel araçlar ve özelliklerle tam olarak paketlenmiştir.

[Daha fazla bilgi >](#)

[Ücretsiz indirin](#)



Mac için Visual Studio

macOS | Sürüm 17.

macOS'e özgü .NET geliştiricileri için kapsamlı bir IDE. Web, bulut, mobil ve oyun geliştirme için en üst düzey desteği içerir.

[Daha fazla bilgi >](#)

Sunun hakkında daha fazla bilgi edinin:
[lisansınız etkinleştiriliyor](#)

[Ücretsiz indirin](#)



Visual Studio Code

Windows, macOS ve Linux | Sürüm 1.68

Windows, macOS ve Linux üzerinde çalışan tek başına bir kaynak kod düzenleyicisi. JavaScript ve web geliştiricileri için hemen hemen her programlama dilini destekleyecek uzantılara sahip en iyi seçim.

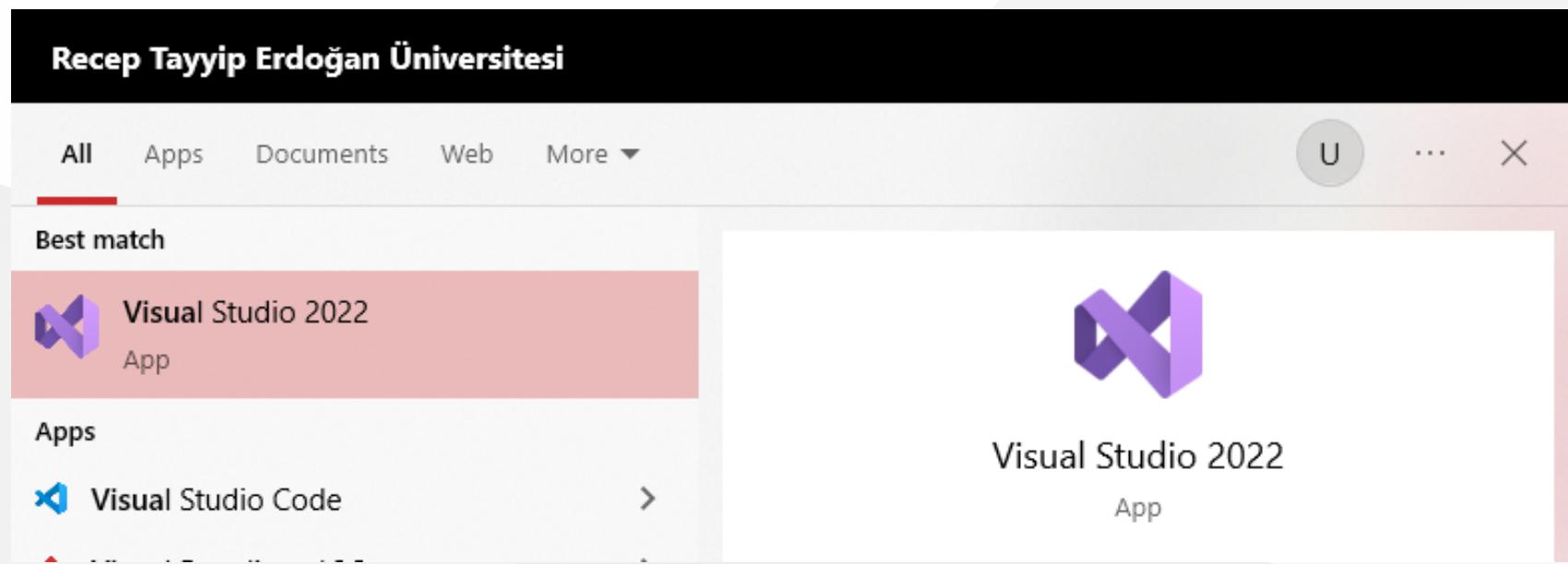
[Daha fazla bilgi >](#)

Visual Studio Code'u kullanarak [lisans & gizlilik bildirimini](#)

[Ücretsiz indirin](#)

Visual Studio Community Edition (Install / Compile / Run / Debug) (2)

- After installation open `Visual Studio 2022` from the menu.



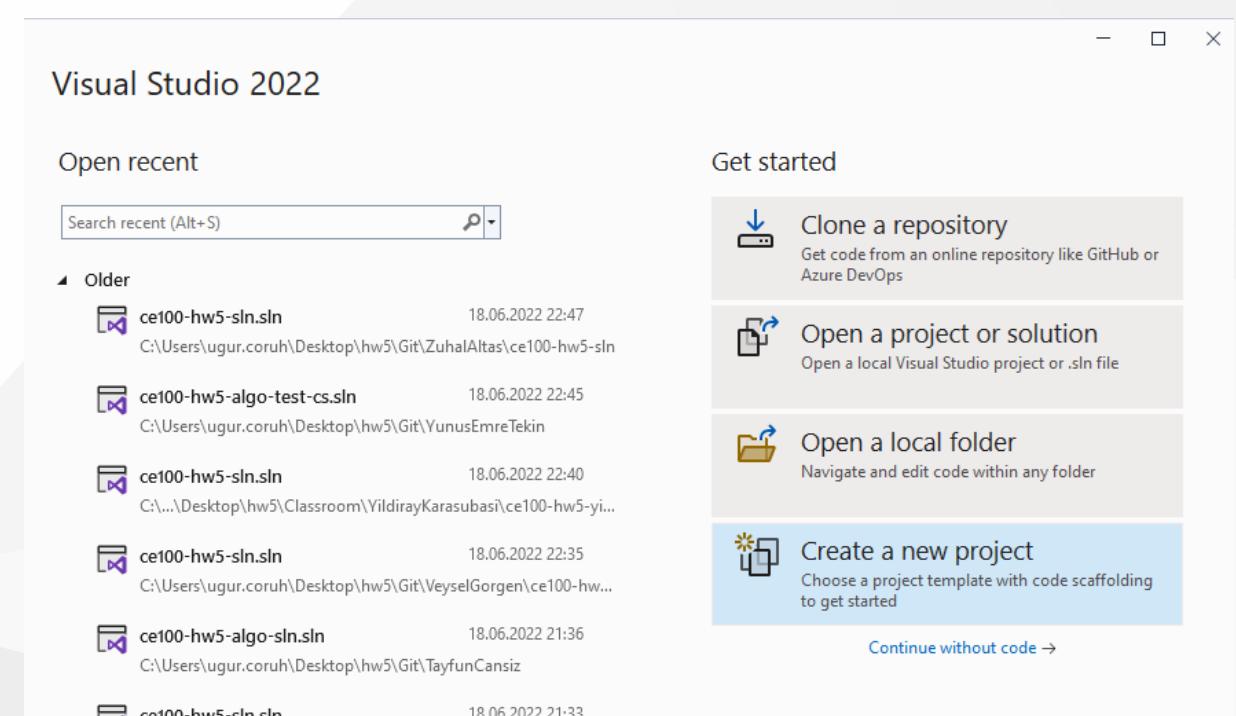
Visual Studio Community Edition (Install / Compile / Run / Debug) (3)

- The application will start...



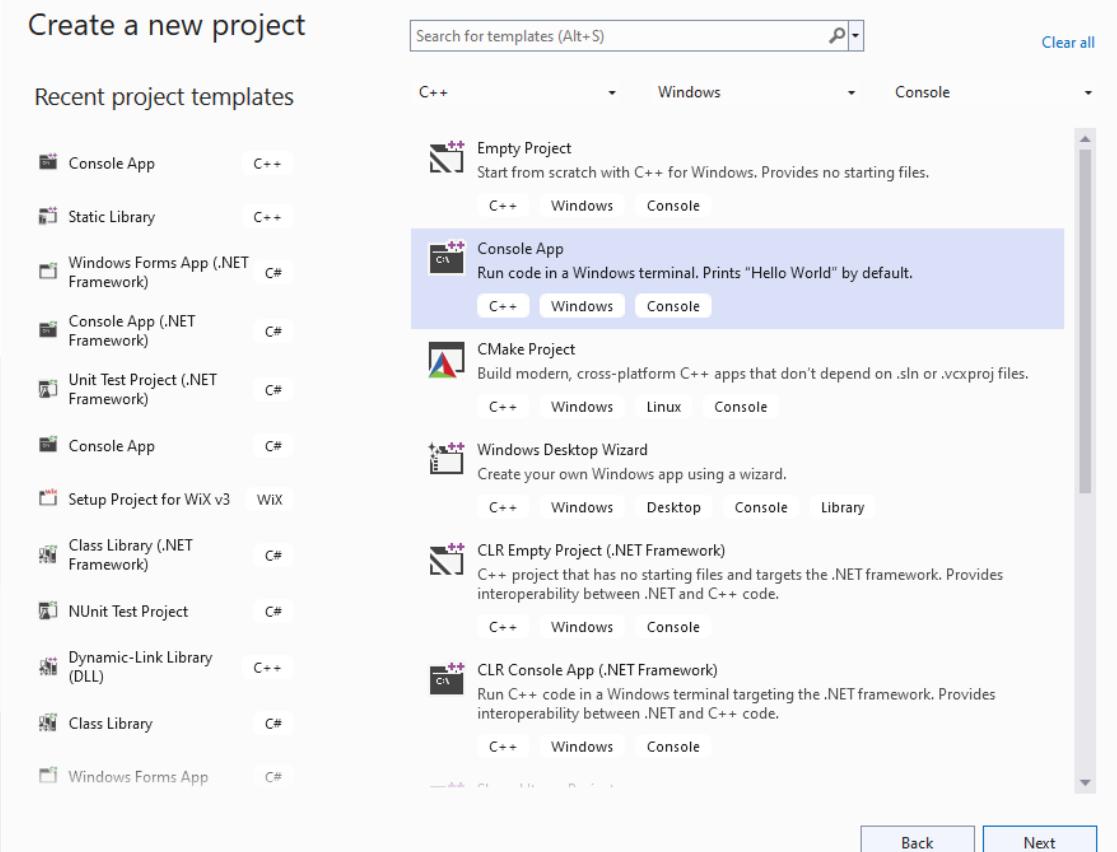
Visual Studio Community Edition (Install / Compile / Run / Debug) (4)

- From Opening Window Select **Create a new project**



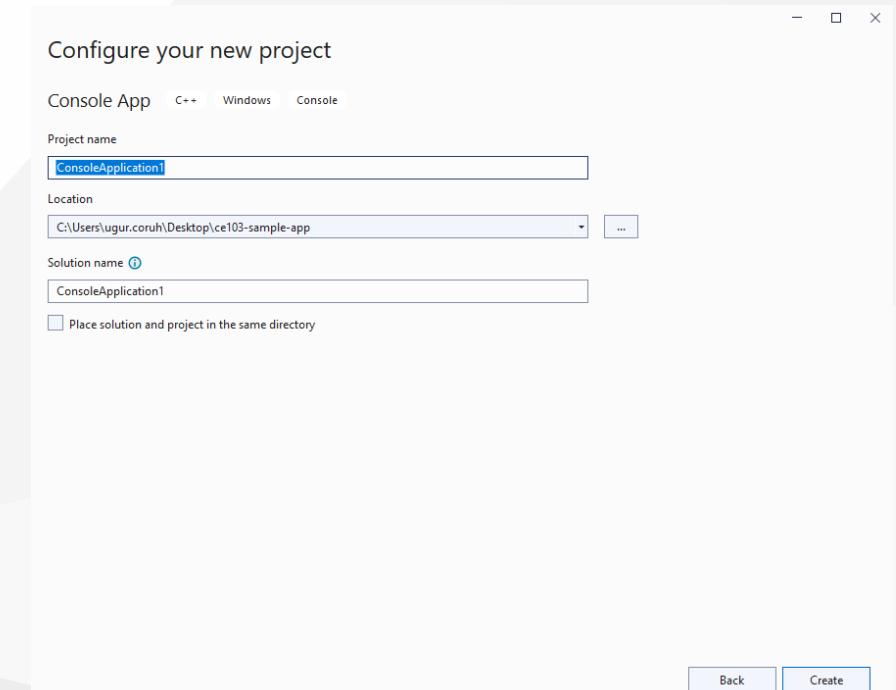
Visual Studio Community Edition (Install / Compile / Run / Debug) (5)

- There will be several options, you can review them.
- Select Windows , C++ , Console Application from Combobox.
- Select Console Application



Visual Studio Community Edition (Install / Compile / Run / Debug) (6)

- Give a solution and project name.
- Select save location



Visual Studio Community Edition (Install / Compile / Run / Debug) (7)

- You will have C++ basic Hello World application.

The screenshot shows the Visual Studio Community Edition interface. The main window displays the code for a C++ console application named "ConsoleApplication1". The code contains a simple "Hello World" print statement. The Solution Explorer on the right shows the project structure with one source file, "ConsoleApplication1.cpp". The Output window at the bottom indicates that there are no issues found and shows a message about GitHub Copilot access.

```
// ConsoleApplication1.cpp : This file contains all the // C++ code for your application. // // It is not recommended to change this file by hand. #include <iostream> int main() { std::cout << "Hello World!\n"; } // Run program: Ctrl + F5 or Debug > Start // Debug program: F5 or Debug > Start Debug // Tips for Getting Started: // 1. Use the Solution Explorer window to manage your files. // 2. Use the Team Explorer window to connect to source control. // 3. Use the Output window to see build output and other messages.
```

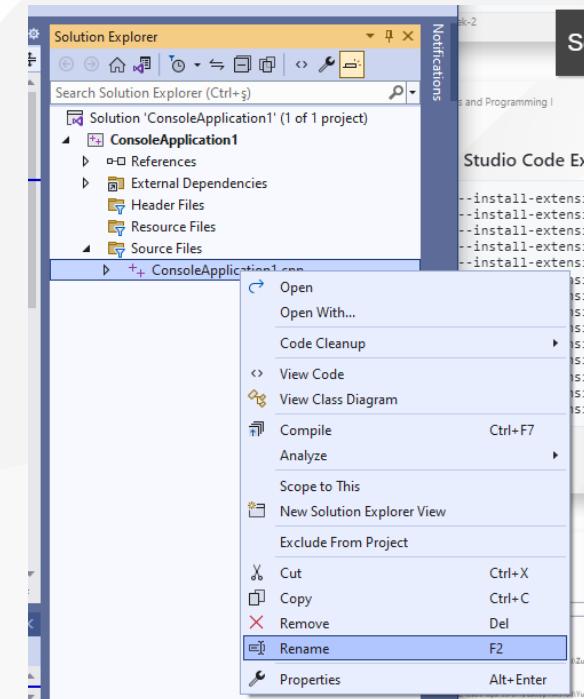
Visual Studio Community Edition (Install / Compile / Run / Debug) (8)

- You will have C++ basic Hello World application.

```
// ConsoleApplication1.cpp : This file contains the 'main' function. Program execution begins and ends there.  
//  
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello World!\n";  
}  
  
// Run program: Ctrl + F5 or Debug > Start Without Debugging menu  
// Debug program: F5 or Debug > Start Debugging menu  
  
// Tips for Getting Started:  
// 1. Use the Solution Explorer window to add/manage files  
// 2. Use the Team Explorer window to connect to source control  
// 3. Use the Output window to see build output and other messages  
// 4. Use the Error List window to view errors  
// 5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project  
// 6. In the future, to open this project again, go to File > Open > Project and select the .sln file
```

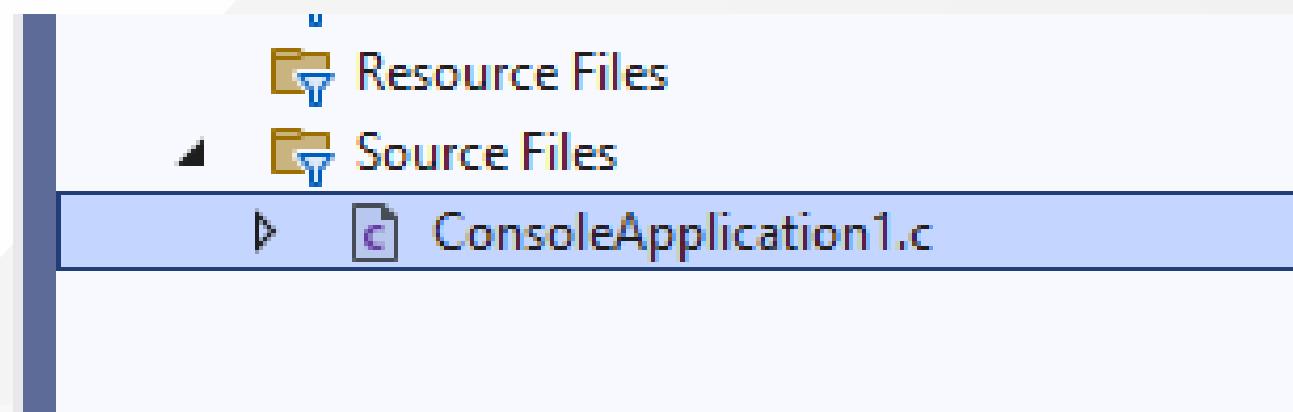
Visual Studio Community Edition (Install / Compile / Run / Debug) (9)

- We need to rename the file extension to `c` from `cpp`



Visual Studio Community Edition (Install / Compile / Run / Debug) (10)

- If you compile the source C compiler will be used.



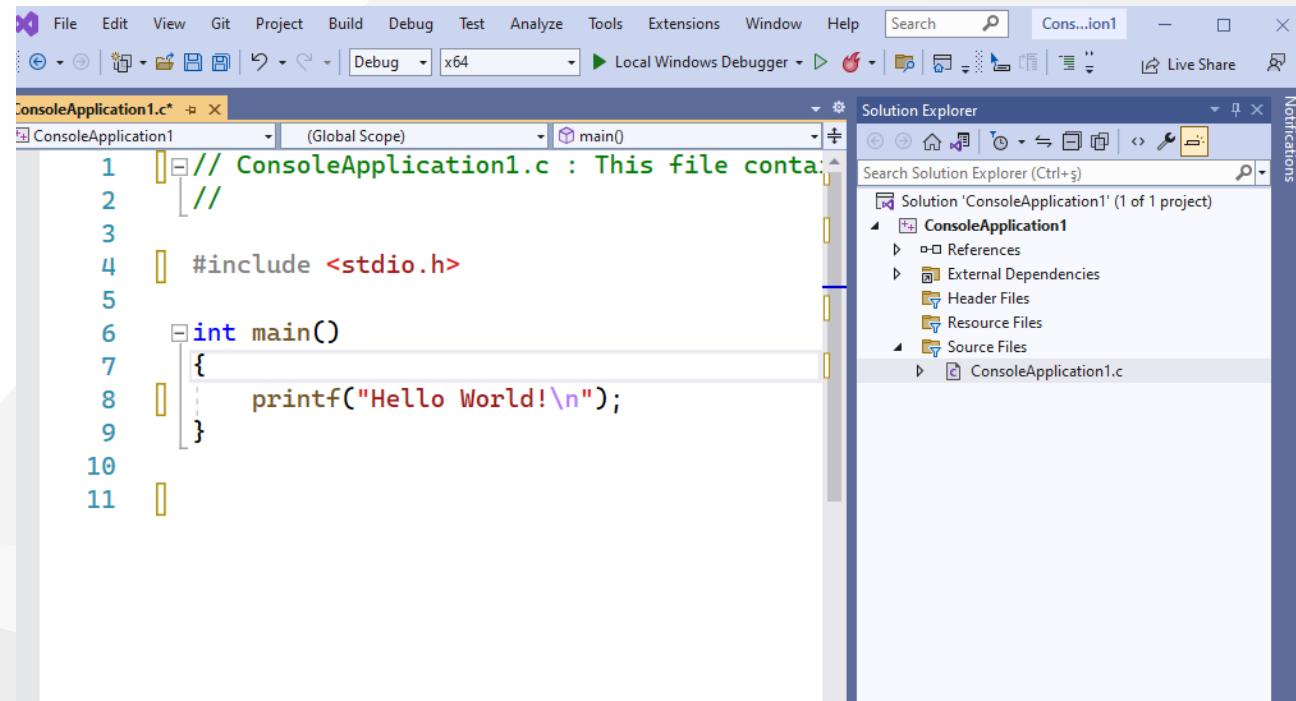
Visual Studio Community Edition (Install / Compile / Run / Debug) (11)

- We need to update our source for C as follows

```
// ConsoleApplication1.c : This file contains the 'main' function. Program execution begins and ends there.  
//  
#include <stdio.h>  
  
int main(){  
    printf("Hello World!\n");  
}
```

Visual Studio Community Edition (Install / Compile / Run / Debug) (12)

- We need to update our source for  as follows



The screenshot shows the Visual Studio Community Edition interface. The main window displays the code for `ConsoleApplication1.c`. The code is as follows:

```
// ConsoleApplication1.c : This file contains all the
// C-level specific source code for a console application
// See ConsoleApplication1.h for the C++/CLI wrapper headers

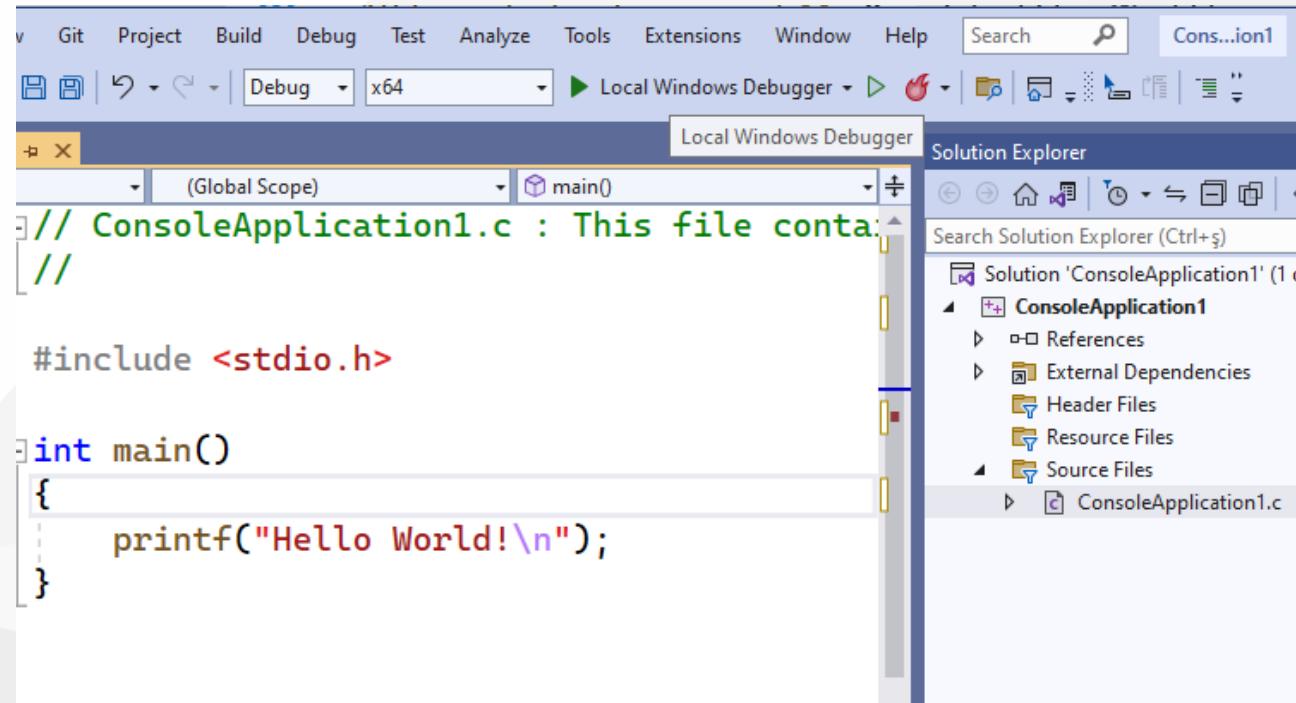
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
}
```

The Solution Explorer on the right shows the project structure for `ConsoleApplication1`, including files for References, External Dependencies, Header Files, Resource Files, and Source Files.

Visual Studio Community Edition (Install / Compile / Run / Debug) (13)

- Put a breakpoint by clicking the following location. Breakpoints will be stop points for our debugging operations.



A screenshot of the Visual Studio Community Edition interface. The menu bar includes File, Git, Project, Build, Debug (set to Debug), Test, Analyze, Tools, Extensions, Window, Help, Search, and a placeholder 'Cons...ion1'. The toolbar includes icons for file operations like Open, Save, and Print, along with Local Windows Debugger, Task List, and Solution Explorer. The status bar shows 'Local Windows Debugger' and other details. The main window displays the code for 'ConsoleApplication1.c':

```
// ConsoleApplication1.c : This file contains all the functions
// that are called from the main() function

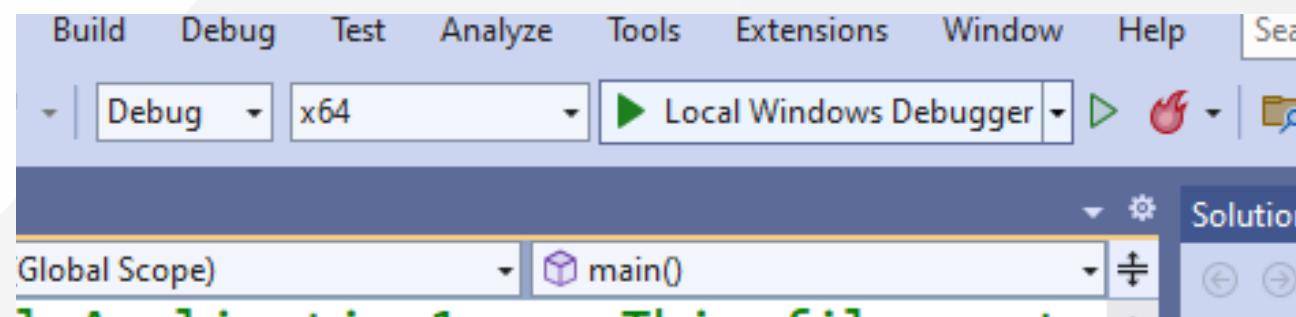
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
}
```

The Solution Explorer on the right shows the project structure for 'ConsoleApplication1' with files like 'ConsoleApplication1.cs' and 'ConsoleApplication1.csproj'.

Visual Studio Community Edition (Install / Compile / Run / Debug) (14)

- Then select Debug configuration and according to your operating system select x64 or x86 configuration and click Local Windows Debugger



Visual Studio Community Edition (Install / Compile / Run / Debug) (15)

- Update your source code as follow

```
// ConsoleApplication1.c : This file contains the 'main' function. Program execution begins and ends there.  
//  
#include <stdio.h>  
  
int sum(int input1, int input2);  
  
int main(){  
    int number = 5;  
    printf("Before Increment : %d\n",number);  
    number = sum(number, 5);  
    printf("After Increment : %d\n", number);  
}  
  
int sum(int input1, int input2){  
    return input1 + input2;  
}
```

Visual Studio Community Edition (Install / Compile / Run / Debug) (16)

- Put the following breakpoints and run the debugger. On number, the variable pins the variable to see its value in real-time.

The screenshot shows the Visual Studio Community Edition interface with a C program named `ConsoleApplication1.c`. The code defines a `main()` function that increments a variable `number` and prints its value before and after the increment. Breakpoints are set at the start of `main()`, the assignment `number = sum(number, 5);`, and the final `printf` statement. A variable pin is applied to the `number` variable, which is highlighted in red and has a tooltip showing its current value of 5. The `Output` window shows the command `Debug` selected. The status bar at the bottom indicates the application is `Ready`.

```
// ConsoleApplication1.c : This file contains the 'main' function.
// Please edit and save to build this project.

#include <stdio.h>

int sum(int input1, int input2);

int main()
{
    int number = 5;
    printf("Before Increment : %d\n", number);

    number = sum(number, 5);

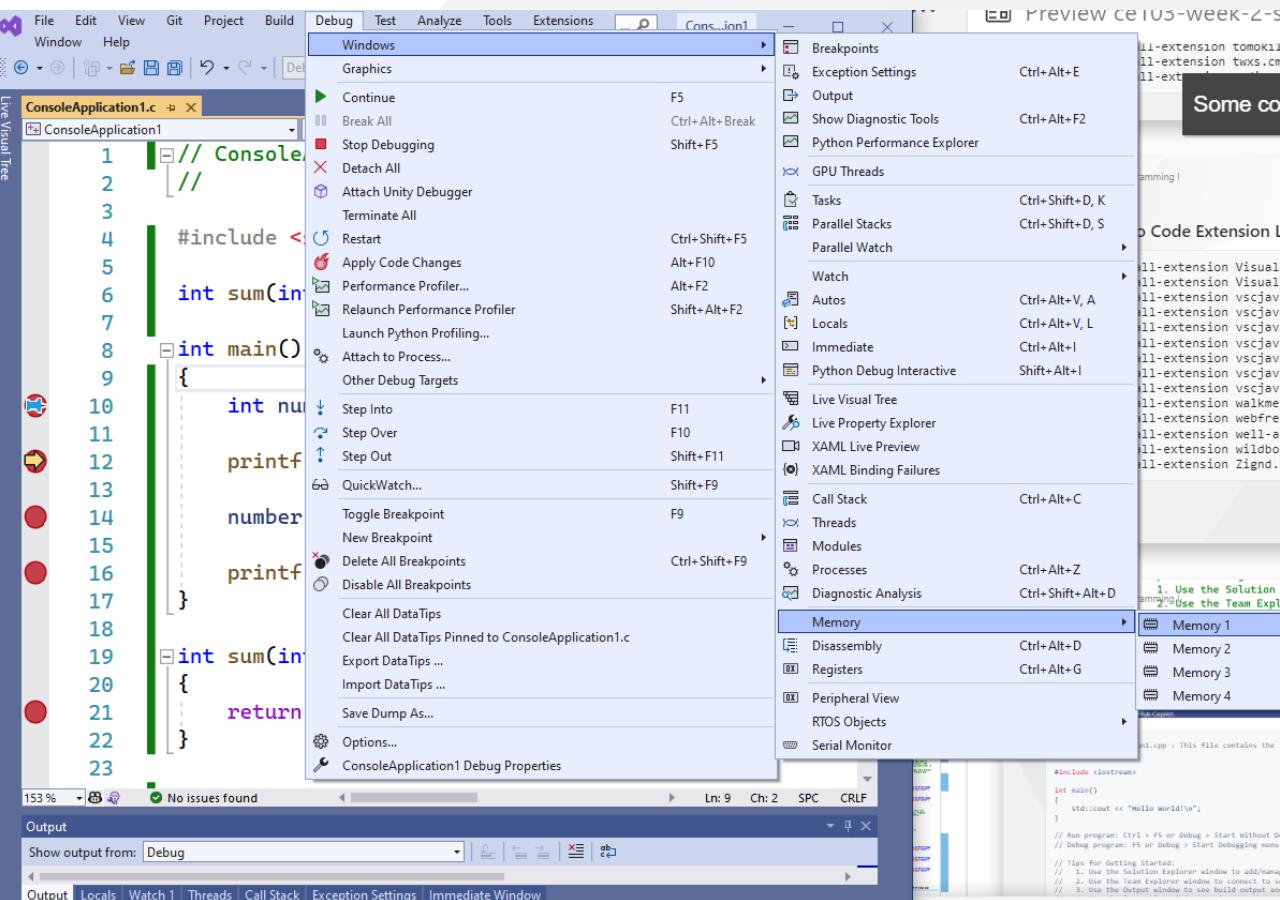
    printf("After Increment : %d\n", number);

    return 0;
}

int sum(int input1, int input2)
{
    return input1 + input2;
}
```

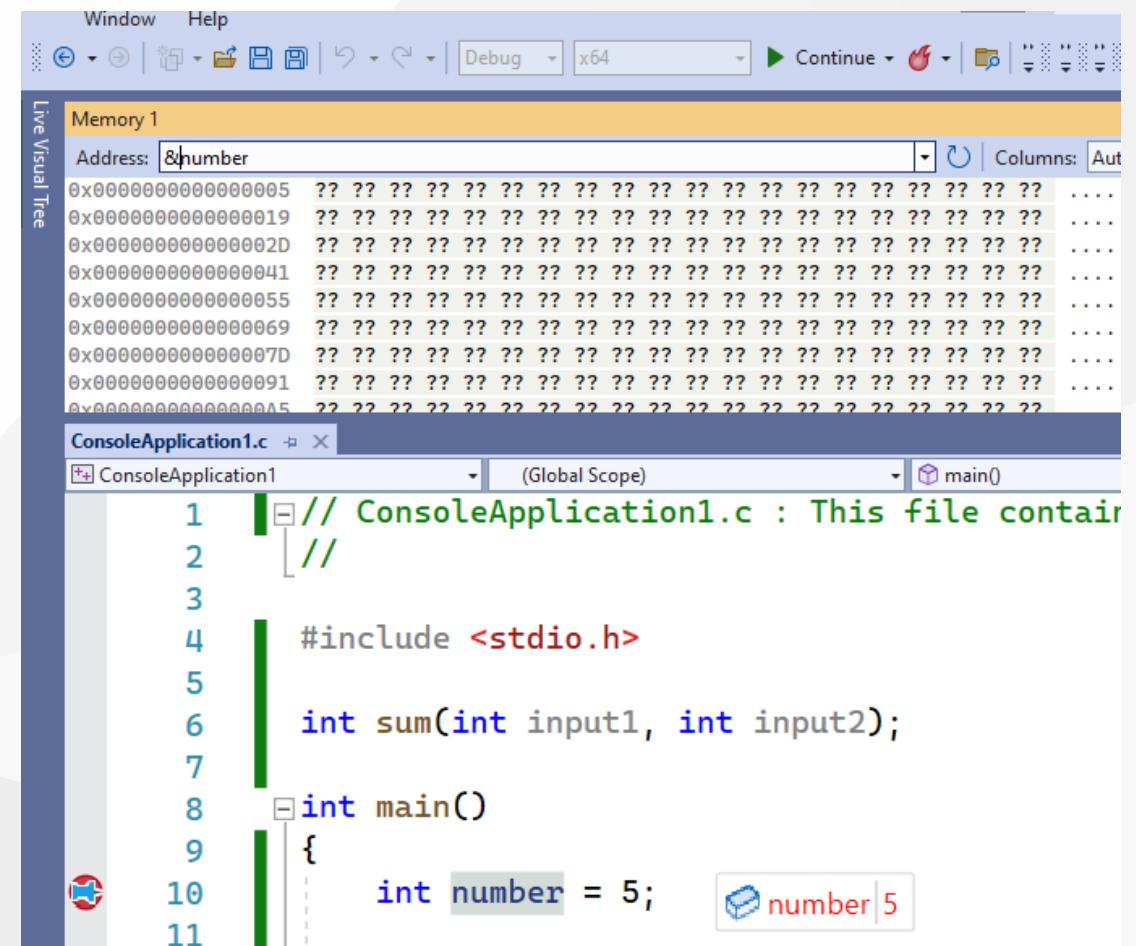
Visual Studio Community Edition (Install / Compile / Run / Debug) (17)

- Open Debug->Windows->Memory->Memory1 to see value in memory



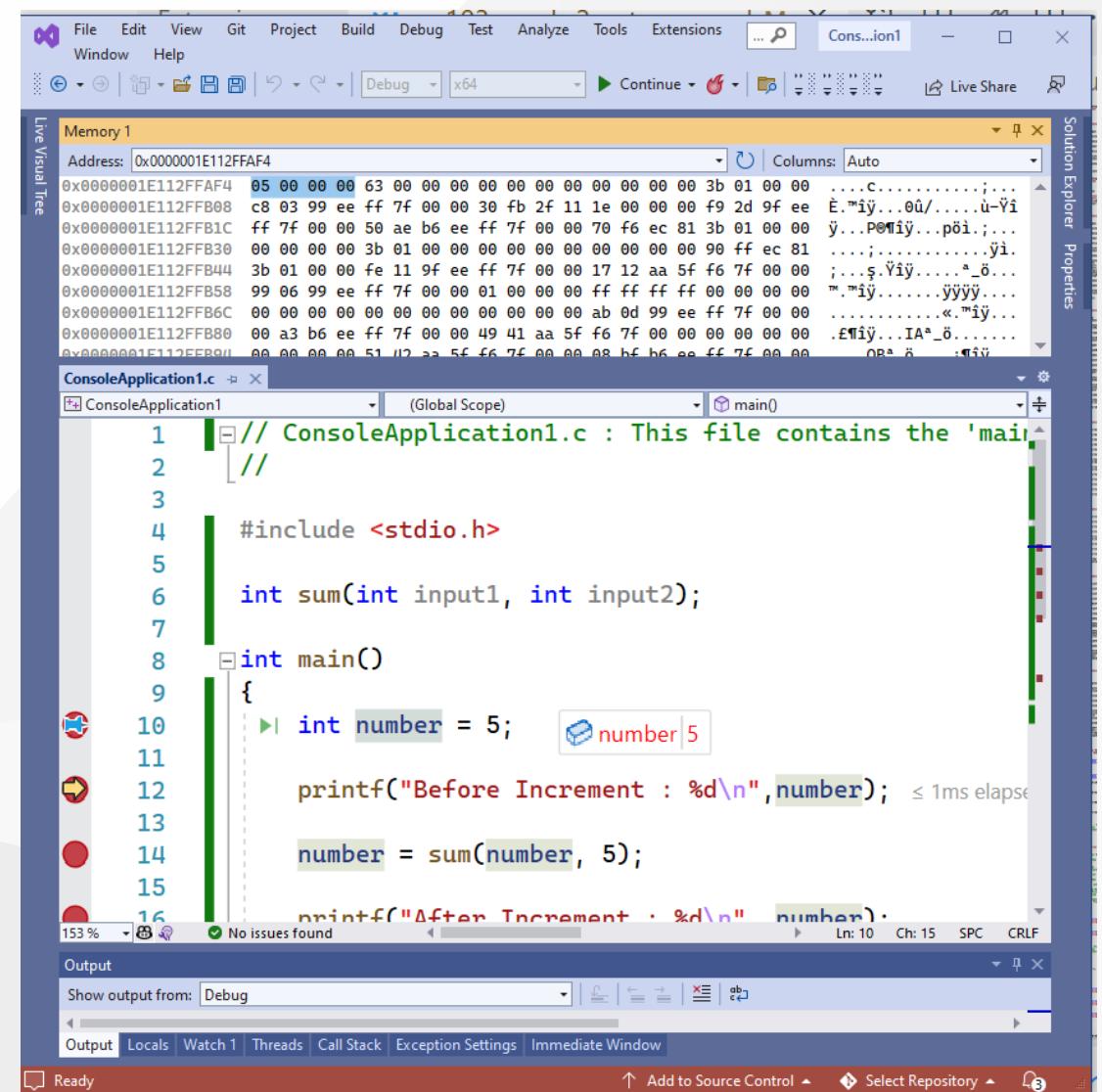
Visual Studio Community Edition (Install / Compile / Run / Debug) (18)

- In the memory window copy variable name (number) with address operator (&) and then (&number) press enter.



Visual Studio Community Edition (Install / Compile / Run / Debug) (19)

- You can see its value in memory with hexadecimal form (05 00 00 00)



Visual Studio Community Edition (Install / Compile / Run / Debug) (20)

- If you change value with pinned control your memory value and your current value will be updated. 20 in hexadecimal 0x14 (integer is 4 bytes length for this reason memory shows 14 00 00 00)

The screenshot shows the Visual Studio IDE. At the top, there is a code editor window displaying the following C code:

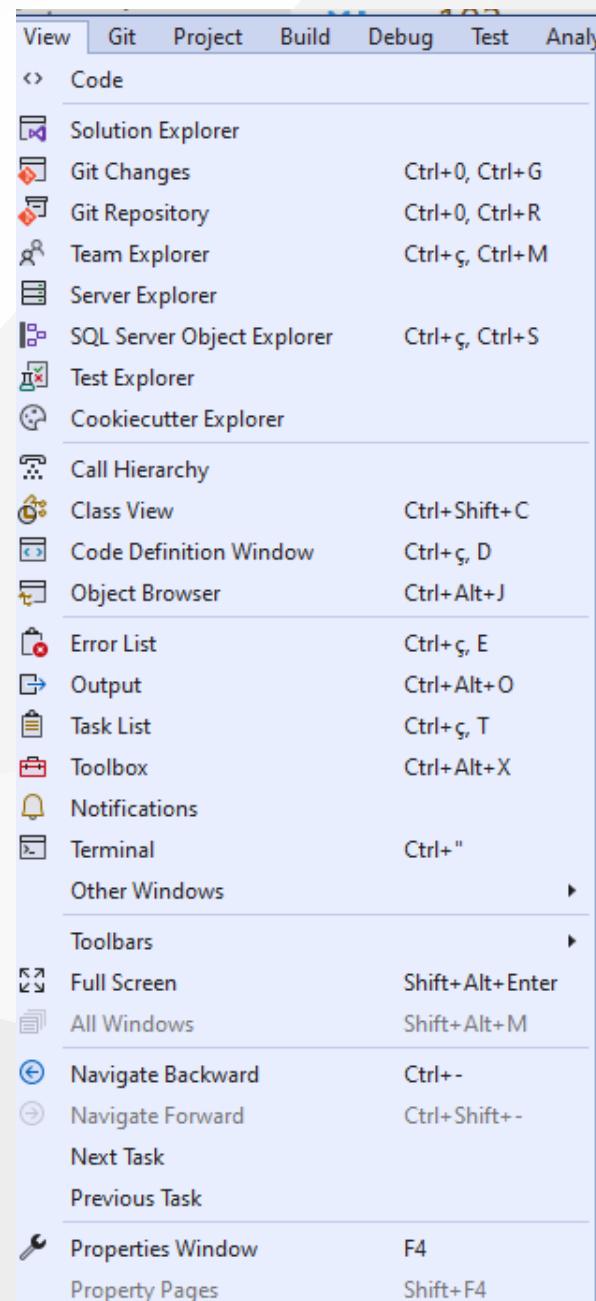
```
int main()
{
    int number = 5; number 20
    printf("Before Increment : %d\n", number);
}
```

Below the code editor is the 'Live Visual Tree' window, specifically the 'Memory 1' tab. It shows a memory dump starting at address 0x0000001E112FFAF4. The first four bytes of memory are highlighted in red, corresponding to the value 20 (hex 14). The memory dump table looks like this:

Address	Value
0x0000001E112FFAF4	14 00 00 00
0x0000001E112FFB08	c8 03 99 ee ff 7f 00
0x0000001E112FFB1C	ff 7f 00 00 50 ae b6
0x0000001E112FFB30	00 00 00 00 3b 01 00
0x0000001E112FFB44	3b 01 00 00 fe 11 9f

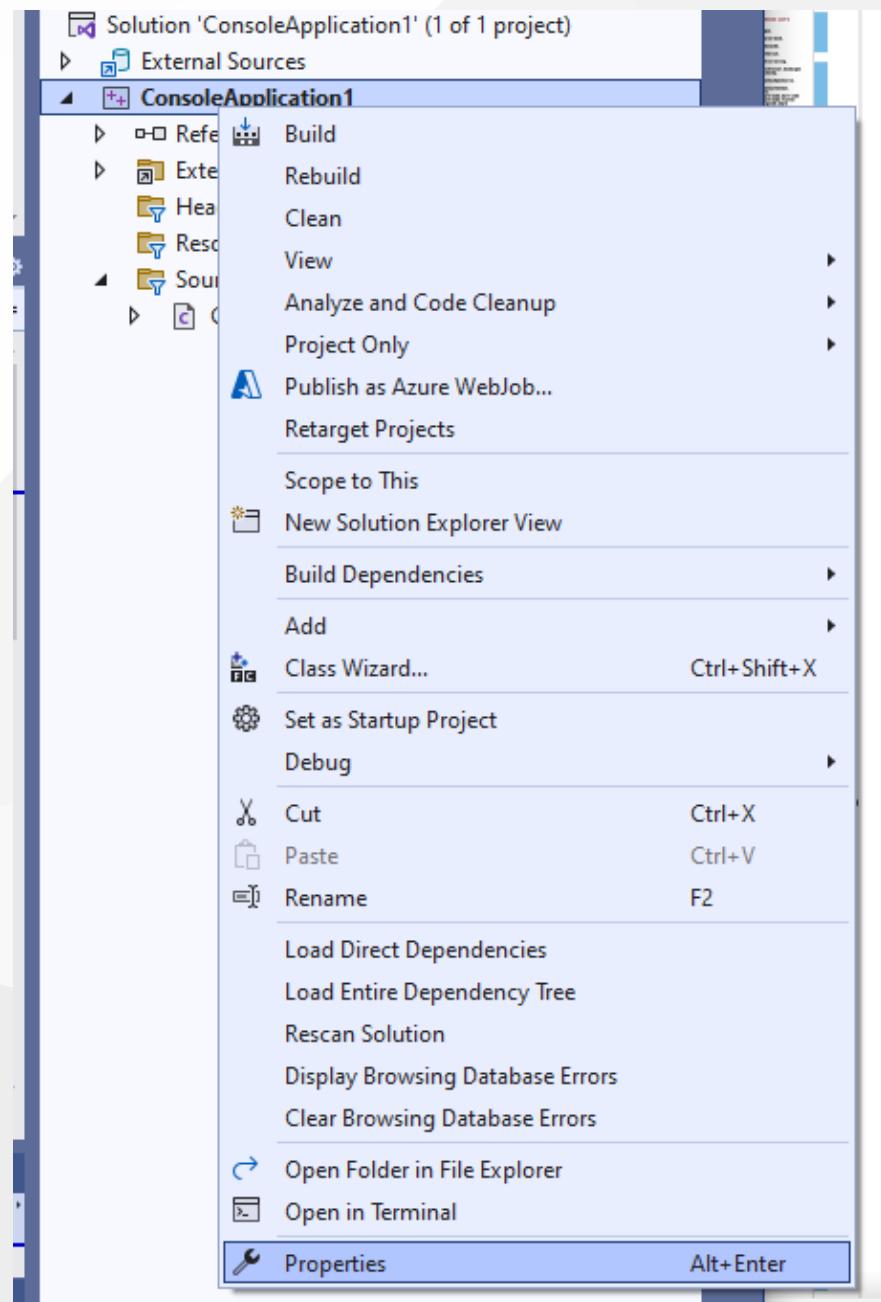
Visual Studio Community Edition (Install / Compile / Run / Debug) (21)

- If you close some windows such as solution explorer, and properties windows you can open them from the View menu.



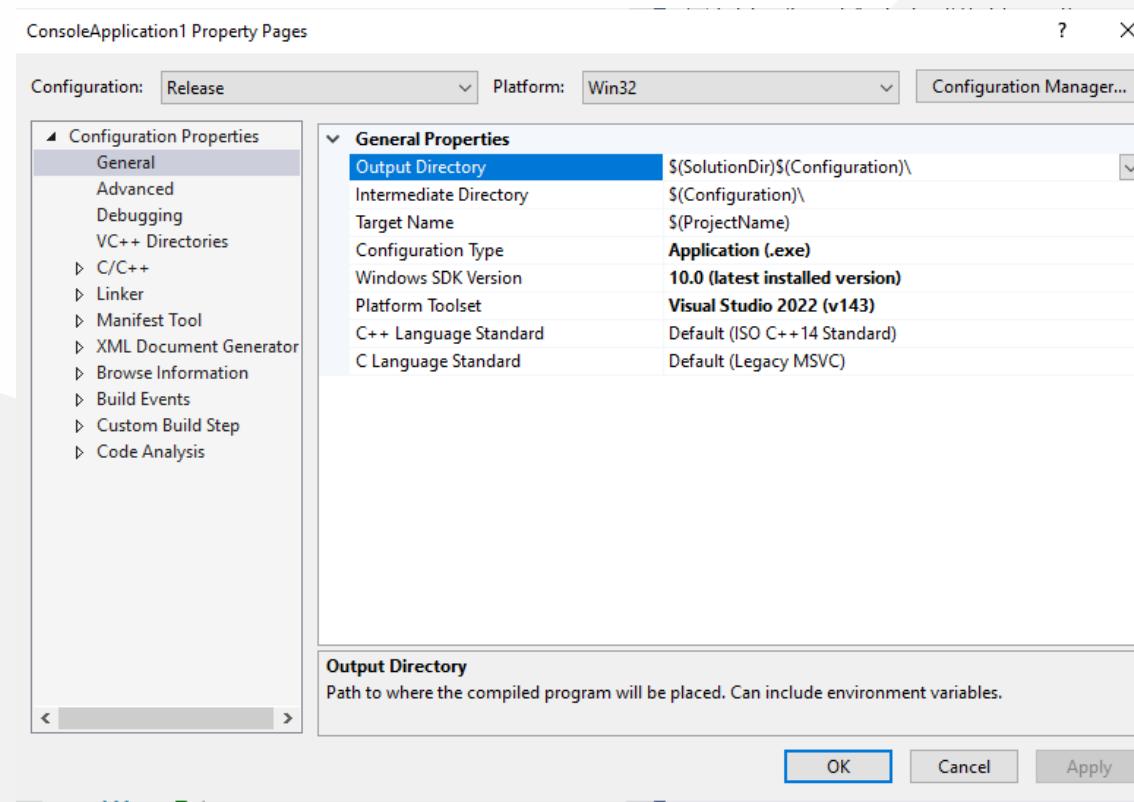
Visual Studio Community Edition (Install / Compile / Run / Debug) (22)

- Solution and Projects have several configurations for each setup such as Release - x86, Release-x64, Debug-x86, and Debug-x64 you need to configure all of them for your requirements. You can access configurations by right-clicking to project and then selecting properties.

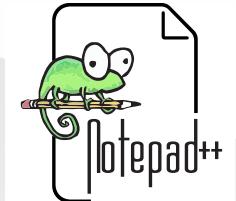


Visual Studio Community Edition (Install / Compile / Run / Debug) (23)

- Project properties has several settings



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 (gcc / g++)

- A complete runtime environment for gcc
 - <https://sourceforge.net/projects/mingw-w64/>
 - <https://sourceforge.net/projects/mingw-w64/files/latest/download>
- w64devkit is a portable C and C++ development kit for x64 (and x86) Windows.
 - <https://www.mingw-w64.org/downloads/#w64devkit>
- Also, see the following notes
 - <https://www.hanshq.net/building-gcc.html>

Notepad++ (Install / Compile) (3)

LLVM installer (clang)

- Download
 - <https://releases.llvm.org/>
- Also, use the following notes
 - <https://llvm.org/devmtg/2014-04/PDFs/Talks/clang-cl.pdf>
 - <https://www.hanshq.net/clang-plugin-example.html>

Notepad++ (Install / Compile) (4)

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

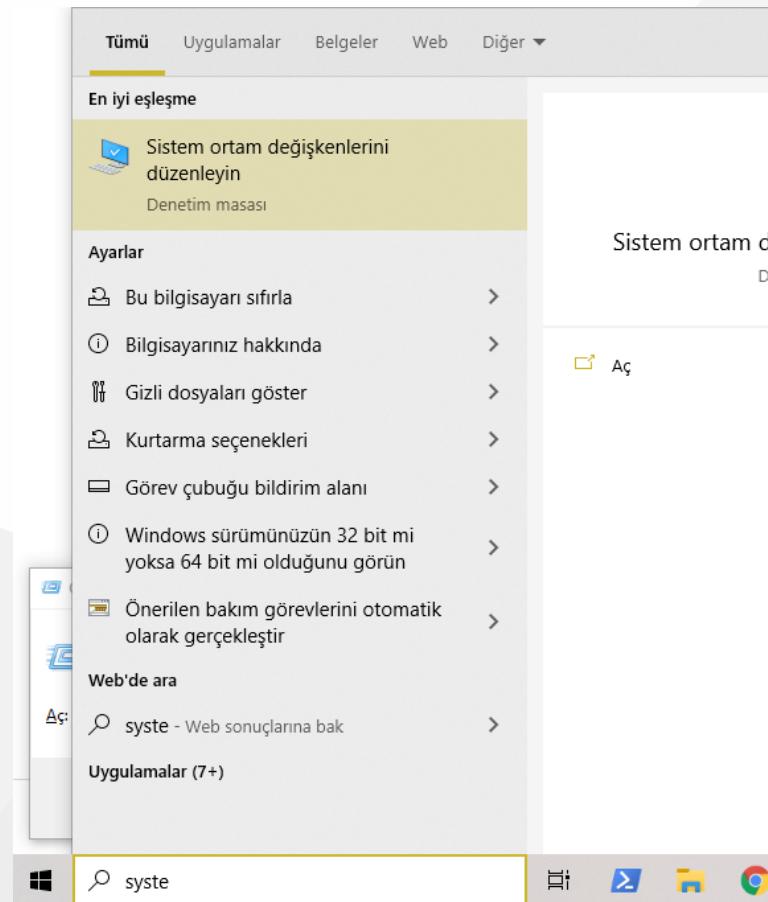
```
gcc --version
```

```
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.
```

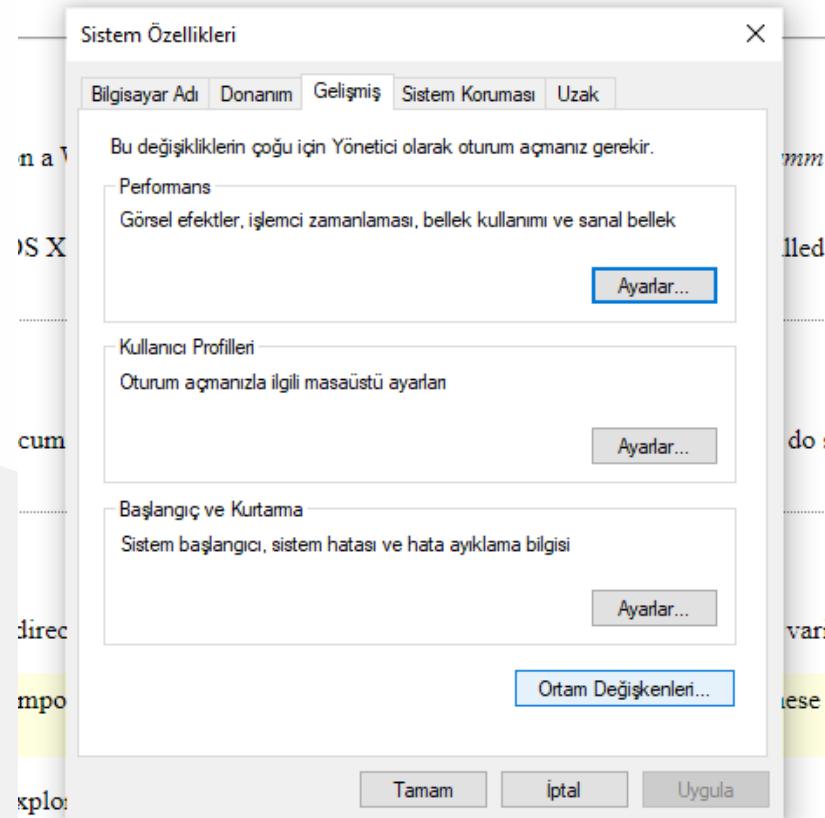
Notepad++ (Install / Compile) (5)

- Open system environments to update path variable for gcc/g++ and clang



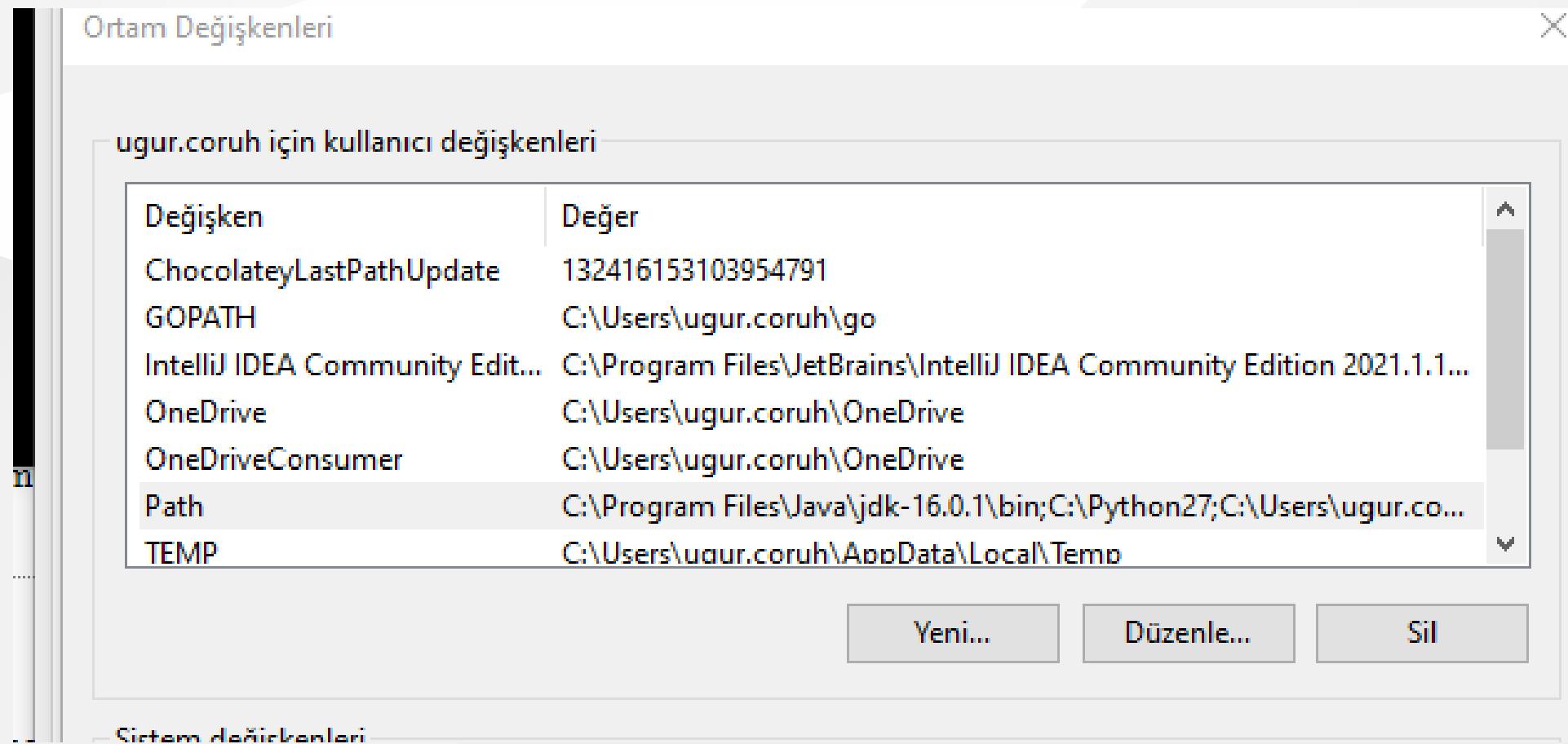
NotePad++ (Install / Compile) (6)

- Open "Environment Variables"



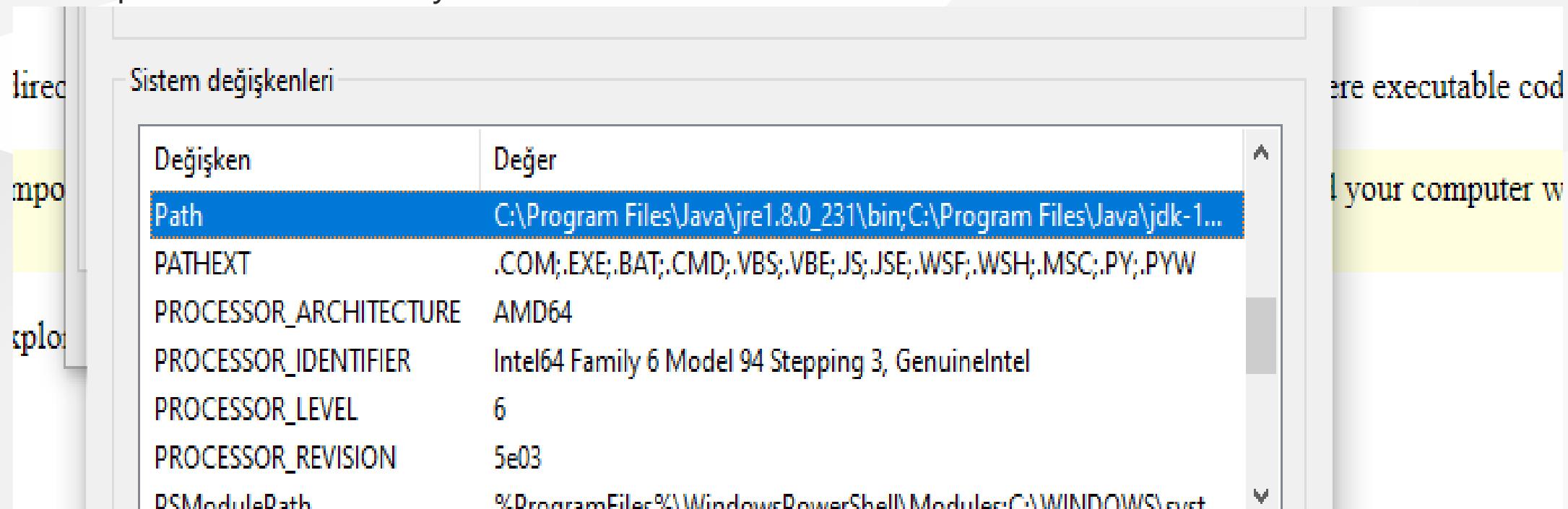
Notepad++ (Install / Compile) (7)

- Select path variable from user section.



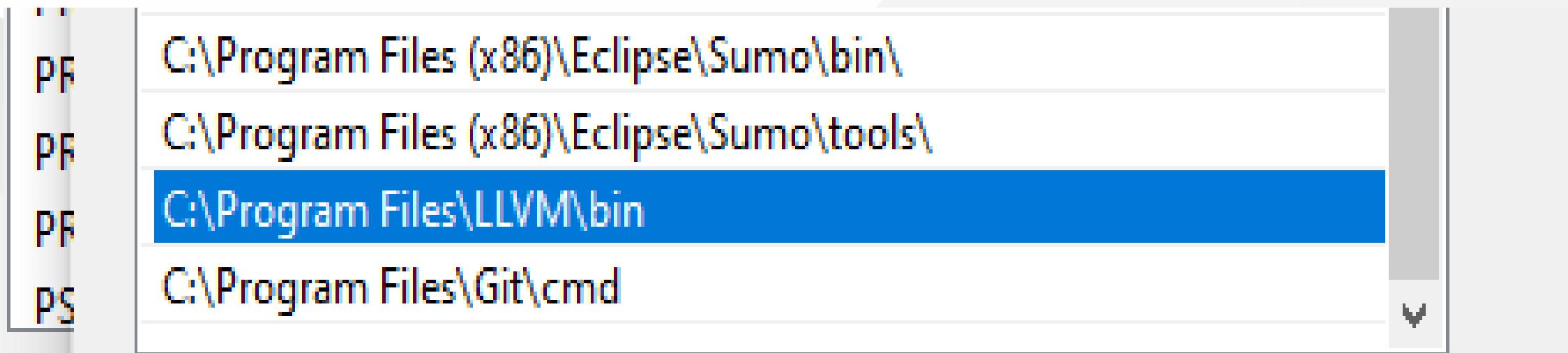
Notepad++ (Install / Compile) (8)

- Select path variable from system section.



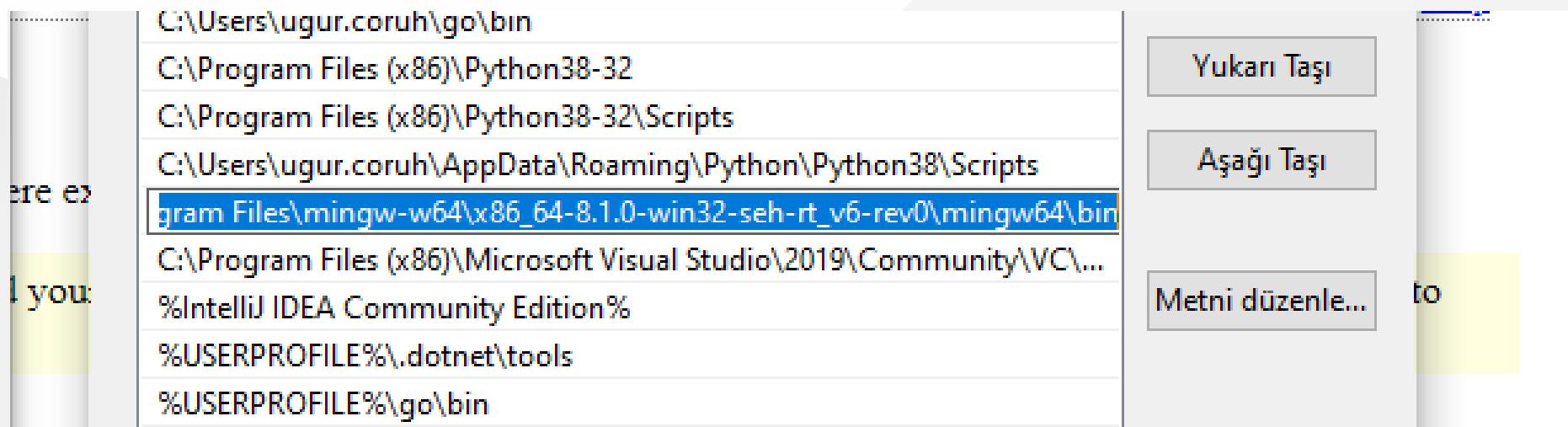
Notepad++ (Install / Compile) (9)

- Update variables add `MinGW` and `LLVM` to path `gcc.exe` `g++.exe` `clang.exe` will be in bin folders. Then we can run this commands from command line.



Notepad++ (Install / Compile) (9)

- Update variables add MinGW and LLVM to path gcc.exe g++.exe clang.exe will be in bin folders. Then we can run this commands from command line.



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
```

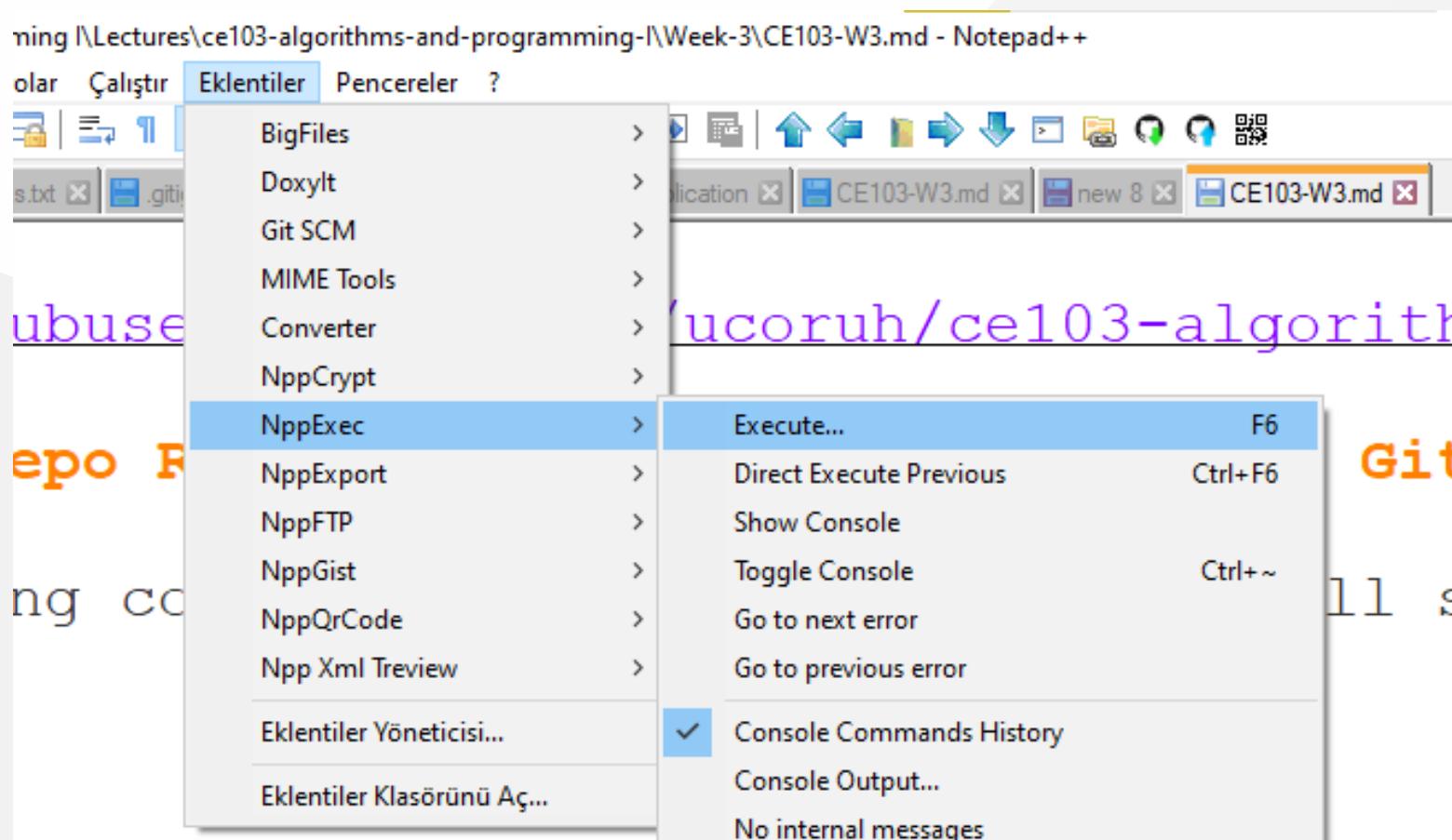
Notepad++ (Install / Compile) (11)

- for `clang.exe` , `lldb.exe` we will use the following path

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

Notepad++ (Install / Compile) (12)

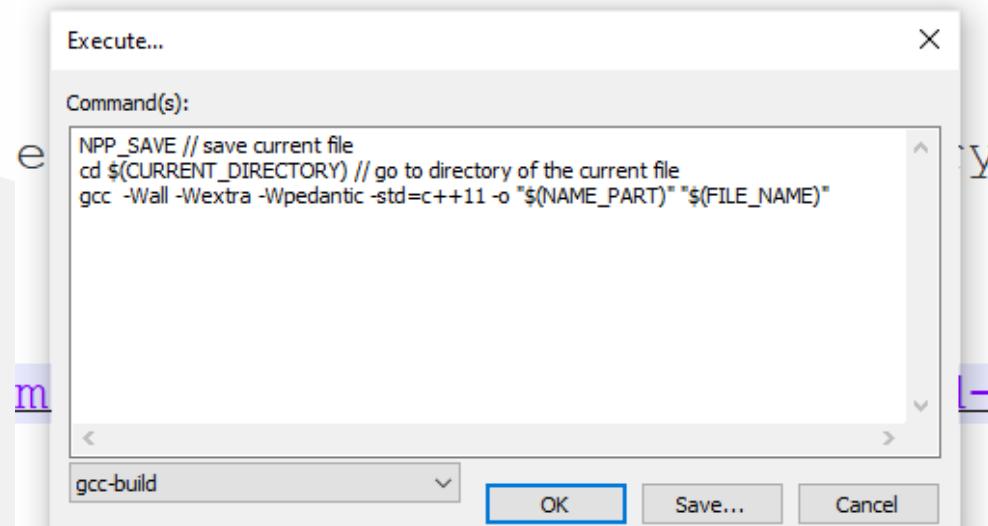
- This folder paths changes according to your setup
- Open `NppExec` extension (install from extension manager if not exist)



NotePad++ (Install / Compile) (13)

- Write the following commands in the 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 the script as `gcc-build` and for more information check the following link
- You can modify or add multiple scripts for another task.

MSYS2

- Software Distribution and Building Platform for Windows

<https://www.msys2.org/>



Vi/Vim (C/C++) for Windows (1)

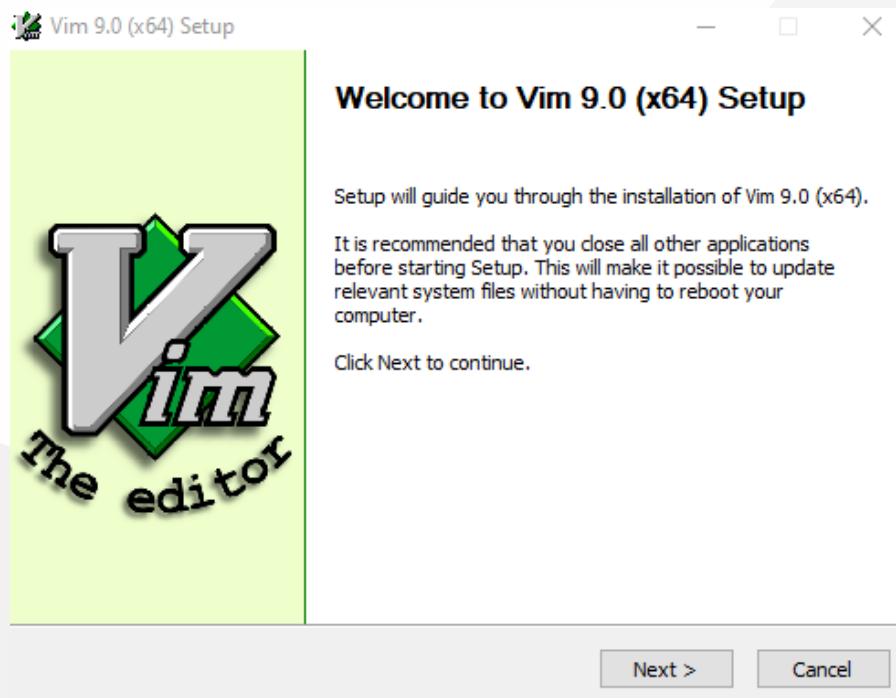
- Vim is a command-line editor for programming
- Use the following links to download Vim for Windows
 - <https://github.com/vim/vim-win32-installer/releases>
 - download : vim online

Vi/Vim (C/C++) for Windows (2)

- Run setup to install the application on your computer.

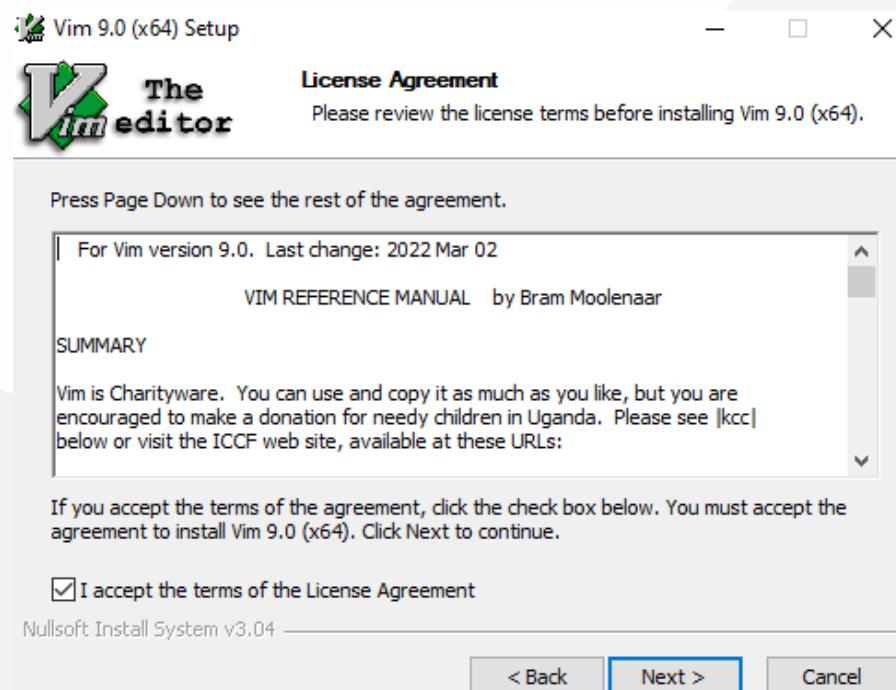


Vi/Vim (C/C++) for Windows (3)



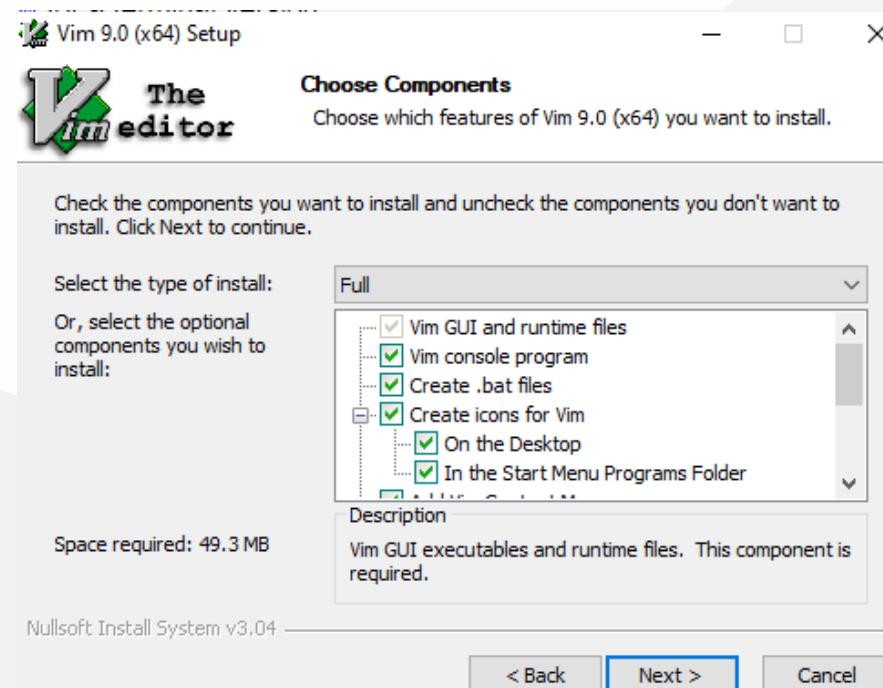
Vi/Vim (C/C++) for Windows (4)

- Installation steps.



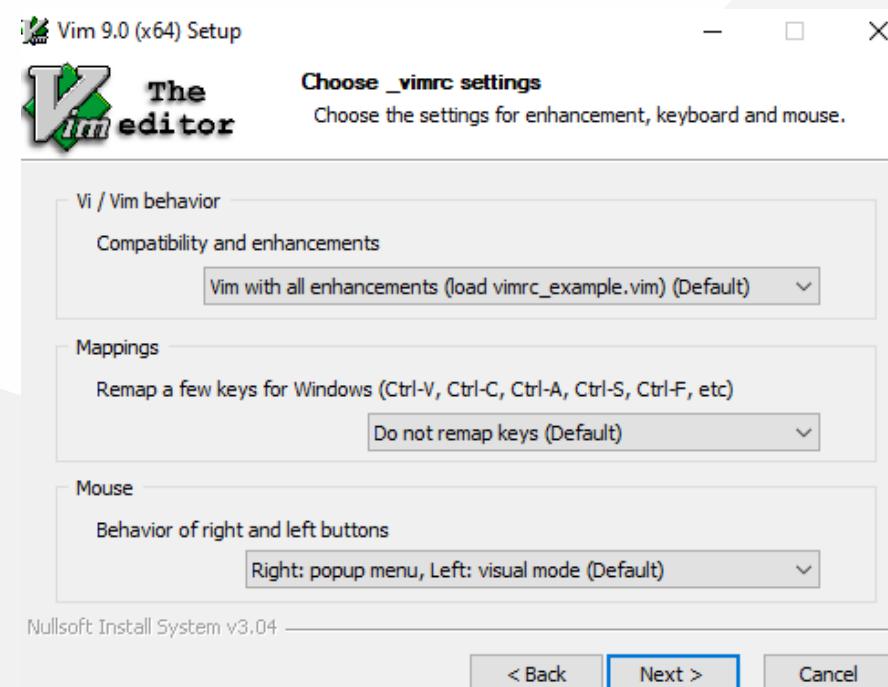
Vi/Vim (C/C++) for Windows (5)

- Installation steps.



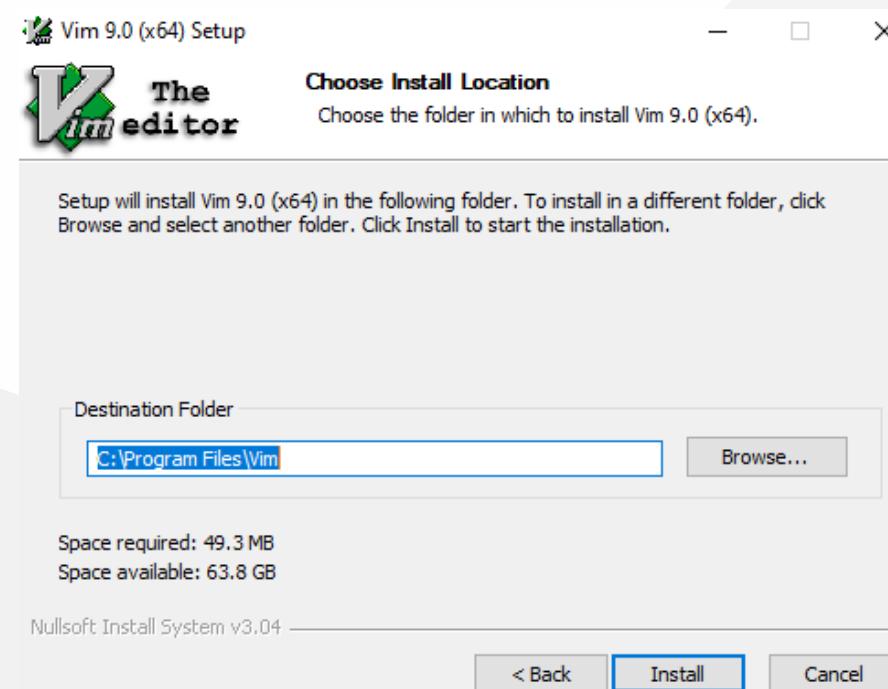
Vi/Vim (C/C++) for Windows (6)

- Installation steps.



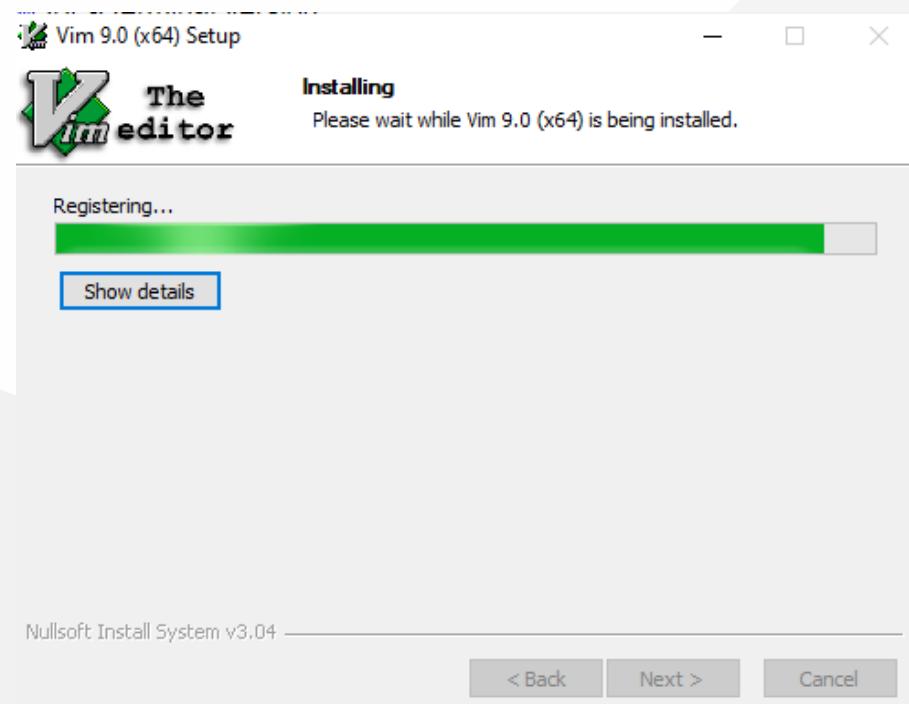
Vi/Vim (C/C++) for Windows (7)

- Installation steps.



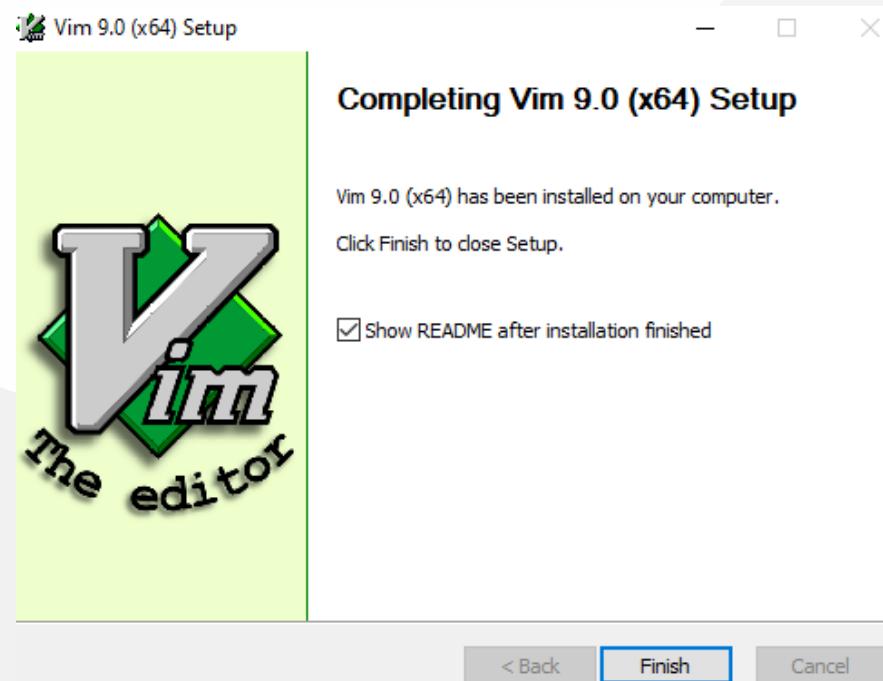
Vi/Vim (C/C++) for Windows (8)

- Installation steps.



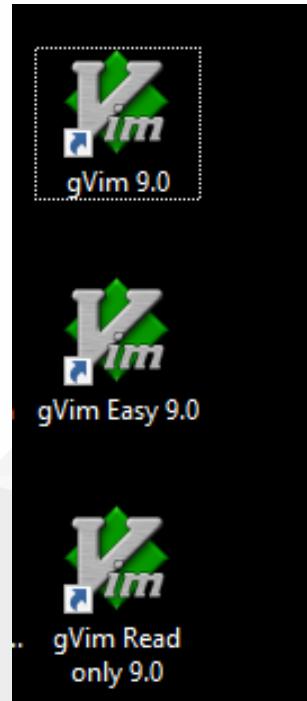
Vi/Vim (C/C++) for Windows (9)

- Installation steps.



Vi/Vim (C/C++) for Windows (10)

- Generated shortcuts on your desktop



Vi/Vim (C/C++) for Windows (11)

- Run `vim hello.c` on your command-line to open a c file with vim editor.

```
WINDOWS\system32\cmd.exe
soft Windows [Version 10.0.19044.2006]
icrosoft Corporation. All rights reserved.

ers\ugur.coruh>cd Desktop
ers\ugur.coruh\Desktop>mkdir vim-sample-project
ers\ugur.coruh\Desktop>cd vim-sample-project
ers\ugur.coruh\Desktop\vim-sample-project>dir
ome in drive C is Windows
ome Serial Number is 8C3C-8F8C

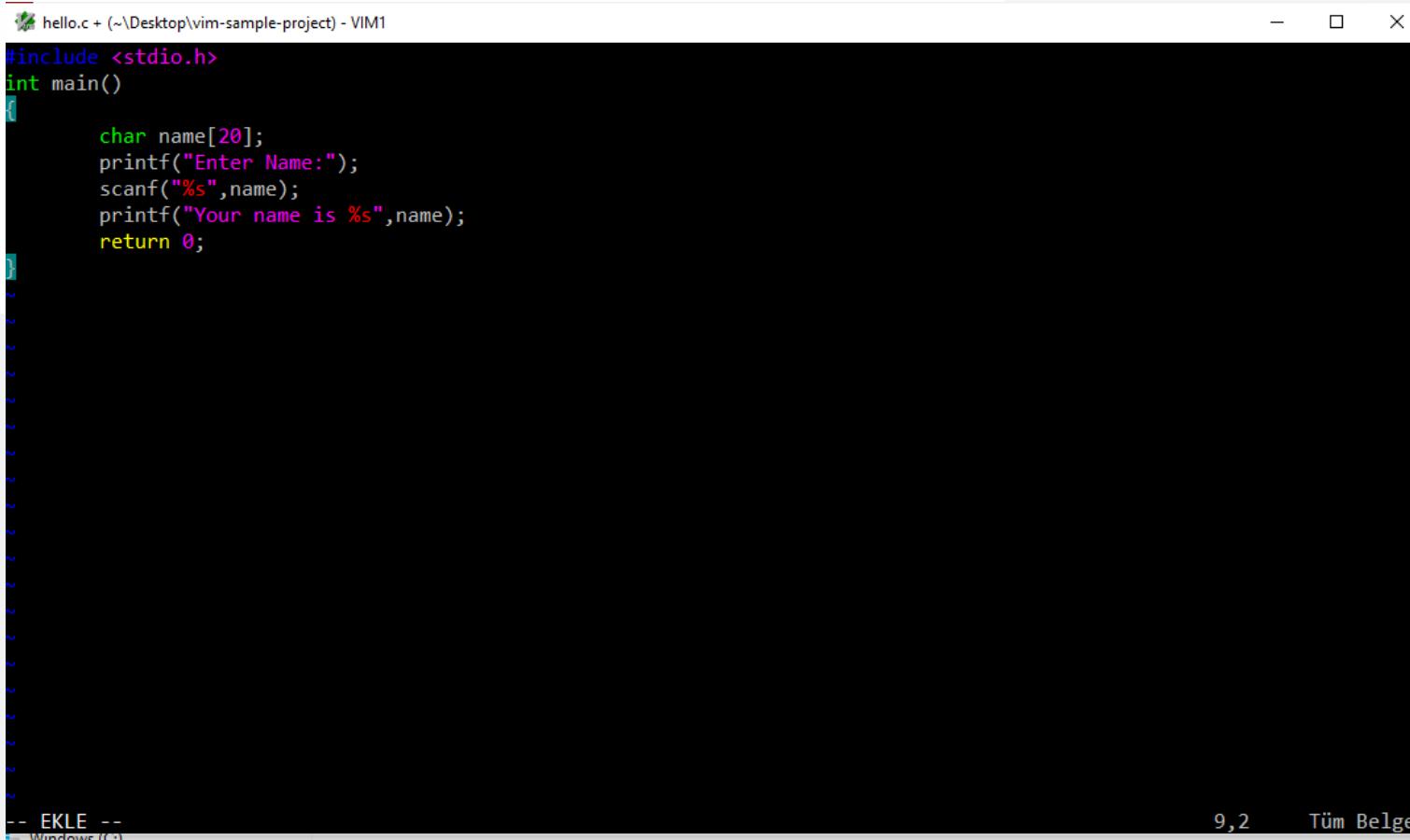
tory of C:\Users\ugur.coruh\Desktop\vim-sample-project

.2022 15:36    <DIR>    .
.2022 15:36    <DIR>    ..
          0 File(s)            0 bytes
          2 Dir(s)  68.409.643.008 bytes free

ers\ugur.coruh\Desktop\vim-sample-project>vim hello.c
```

Vi/Vim (C/C++) for Windows (12)

- You will have the following editor.
- Use INSERT to change edit options.



A screenshot of a Vim window titled "hello.c + (~\Desktop\vim-sample-project) - VIM1". The window contains the following C code:

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

The status bar at the bottom shows "9,2" and "Tüm Belge".

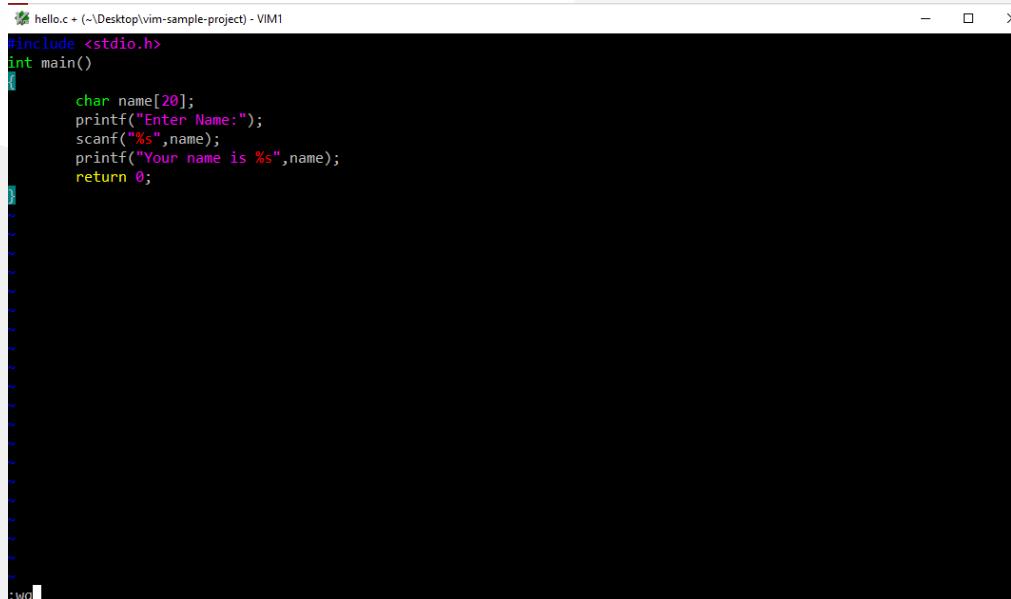
Vi/Vim (C/C++) for Windows (13)

- Sample source code

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

Vi/Vim (C/C++) for Windows (14)

- Write source code
- Press the `Esc` button to enter command mode
- Then type `:wq`. It will save the file and exit from Vim
 - w: write
 - q: quit



A screenshot of a Vim window titled "hello.c + (~\Desktop\vim-sample-project) - VIM1". The window displays the following C code:

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

The status bar at the bottom shows the command `:wd`.

Vi/Vim (C/C++) for Windows (15)

- compile source code with `gcc`
- link the objects and
- run executable

```
C:\Users\ugur.coruh\Desktop\vim-sample-project>gcc -c hello.c -o hello.o
```

```
C:\Users\ugur.coruh\Desktop\vim-sample-project>gcc hello.o -o hello.exe
```

```
C:\Users\ugur.coruh\Desktop\vim-sample-project>hello.exe
```

```
Enter Name:Ugur Coruh
```

```
Your name is Ugur
```

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

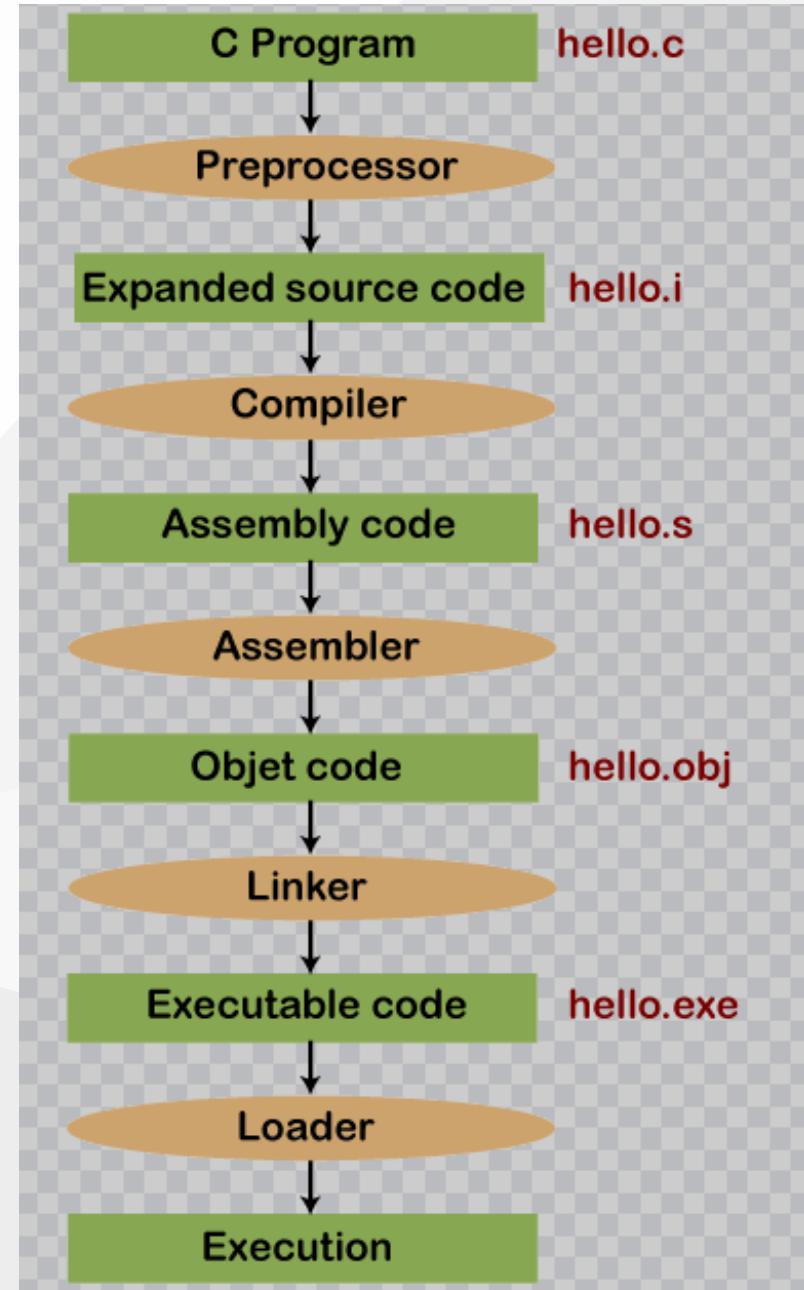
Vi/Vim (C/C++) for Windows (17)

- In the folder, you can find your executable. `hello.exe`

Name	Date modified
<code>.hello.c.swp</code>	1.10.2022 15:40
<code>.hello.c.un~</code>	1.10.2022 15:47
<code>a.exe</code>	1.10.2022 15:48
<code>hello.c</code>	1.10.2022 15:47
<code>hello.exe</code>	1.10.2022 15:49
<code>hello.o</code>	1.10.2022 15:49

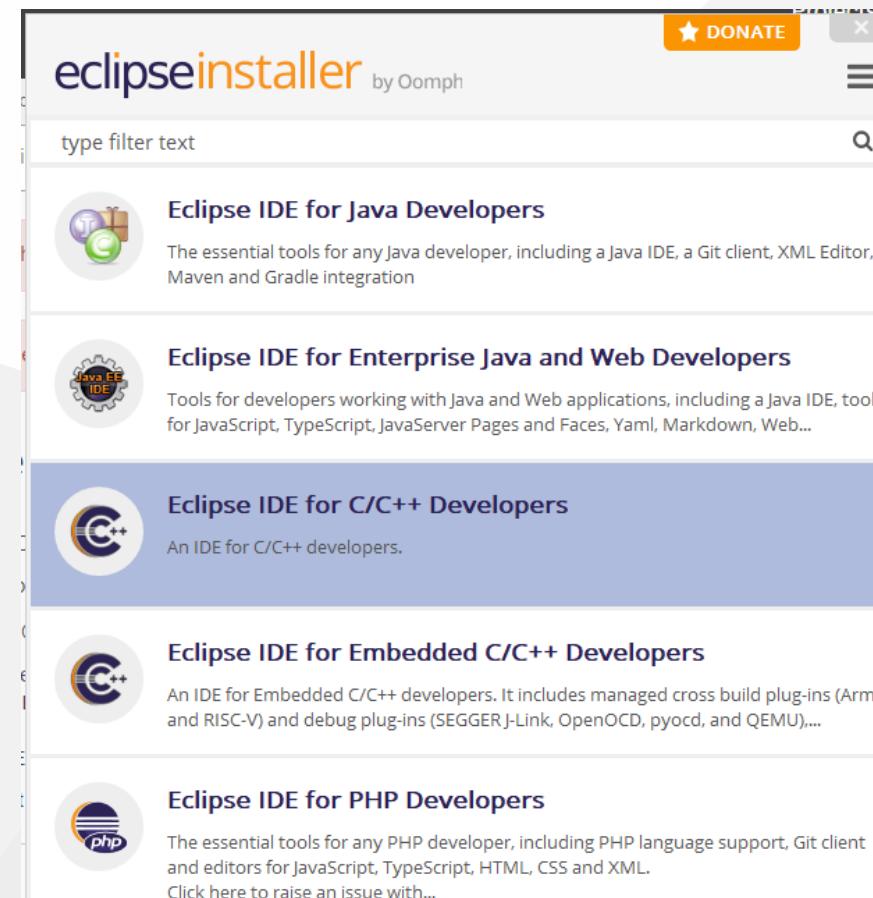
Vi/Vim (C/C++) for Windows (16)

- compile, link and execute flow will be as follow;



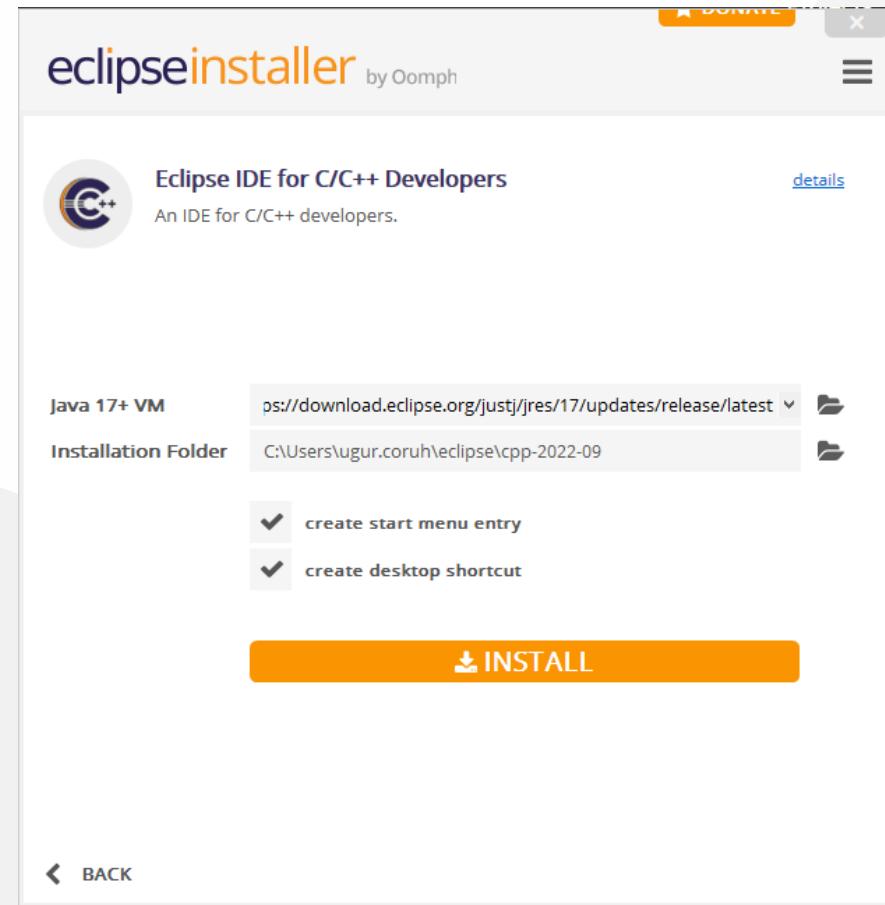
Eclipse (C/C++) - Compile Only / Debugging Has Problem (1)

- Download and install Eclipse IDE from the following link
 - [Eclipse IDE for C/C++ Developers | Eclipse Packages](#)
- Run Installer
- Select **Eclipse IDE for C/C++ Developers**



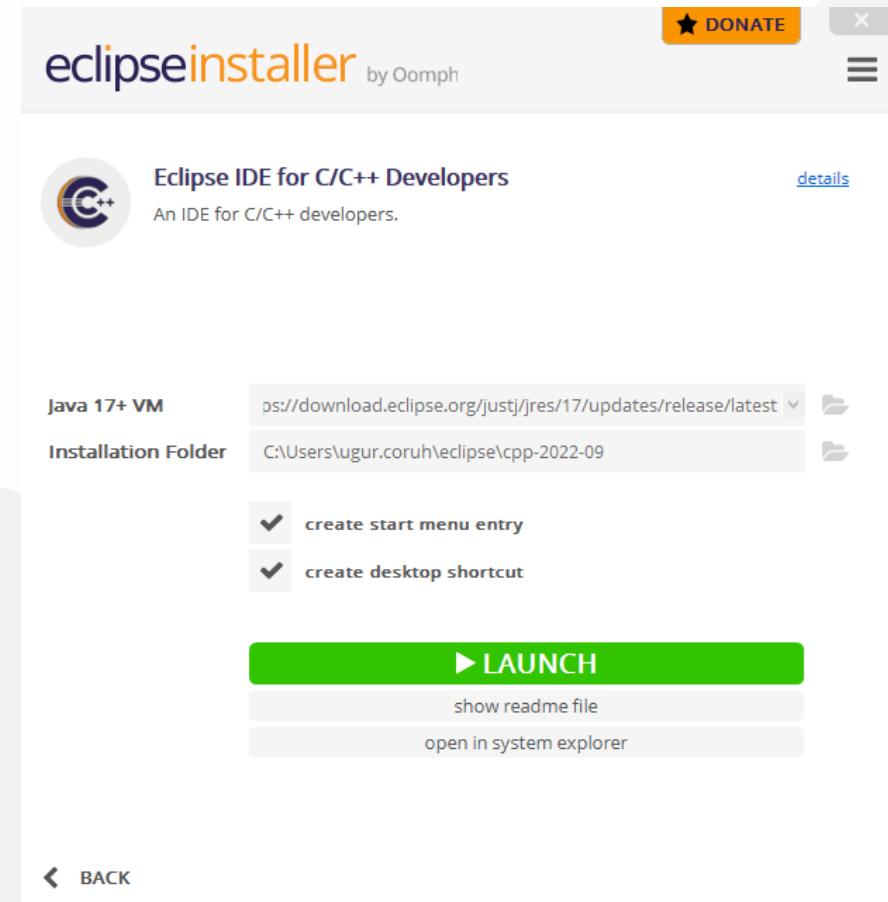
Eclipse (C/C++) - Compile Only / Debugging Has Problem (2)

- Select Java Version and Installation Path



Eclipse (C/C++) - Compile Only / Debugging Has Problem (3)

- After installation you can LAUNCH eclipse IDE

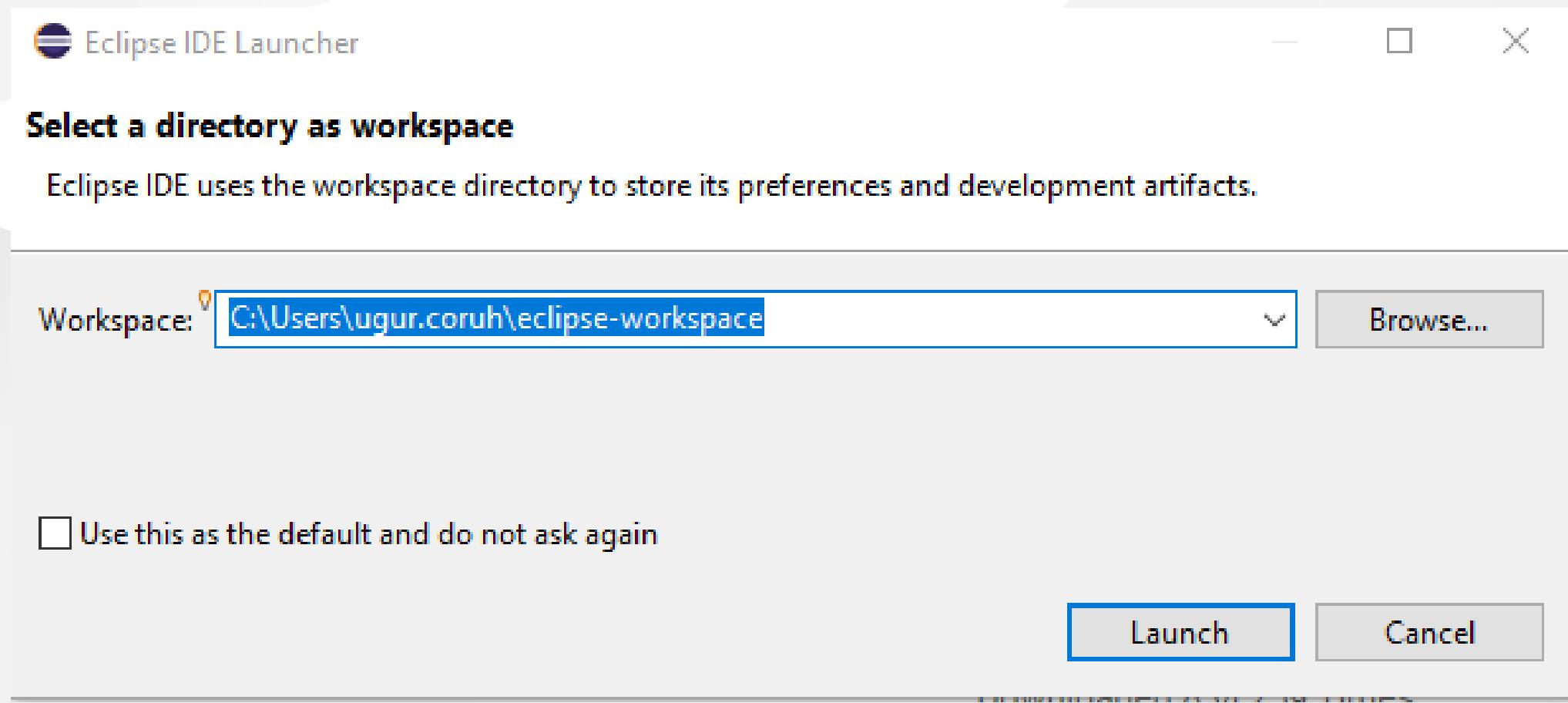


Eclipse (C/C++) - Compile Only / Debugging Has Problem (4)



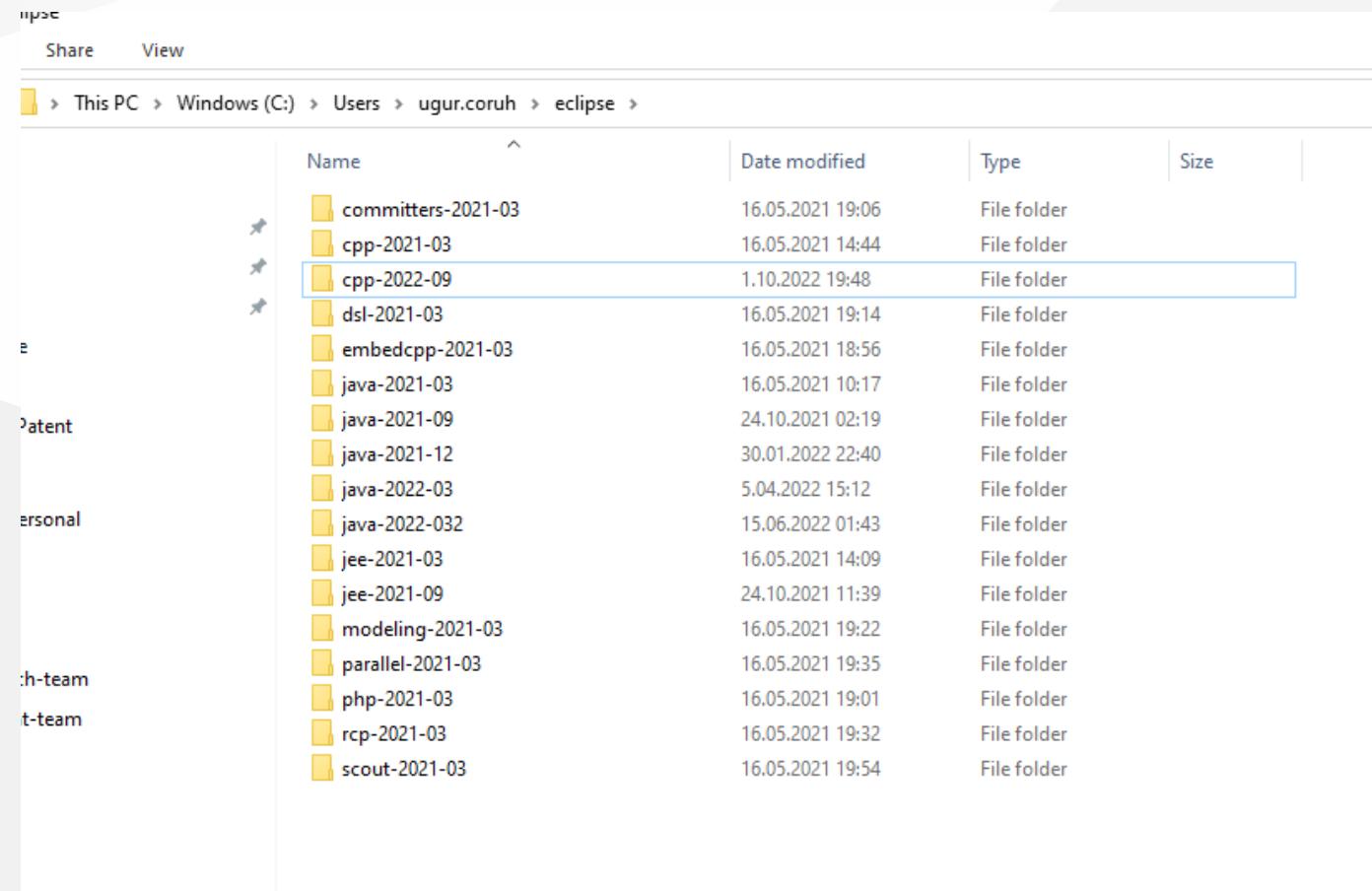
Eclipse (C/C++) - Compile Only / Debugging Has Problem (5)

- Select a workspace that your project will be saved



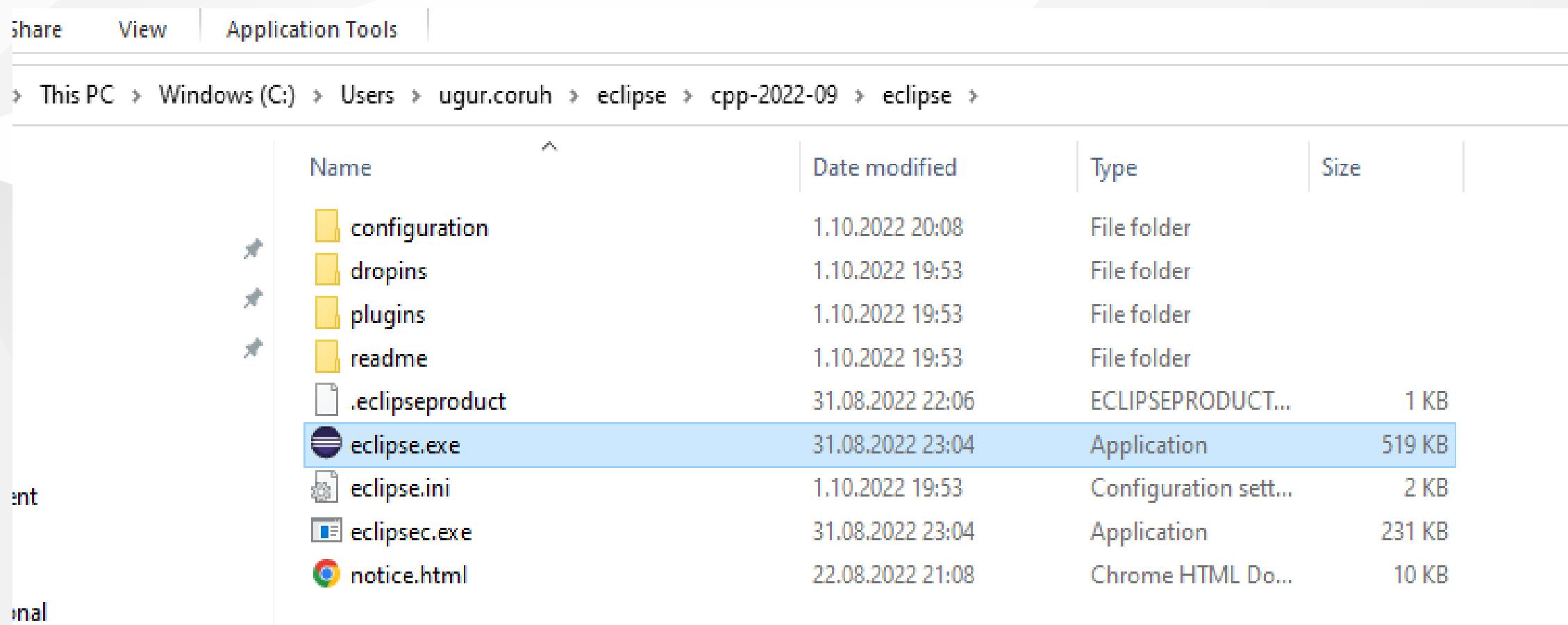
Eclipse (C/C++) - Compile Only / Debugging Has Problem (6)

- You can find your installation under your user folder



Eclipse (C/C++) - Compile Only / Debugging Has Problem (7)

- You can create shortcut to desktop for your working eclipse version.

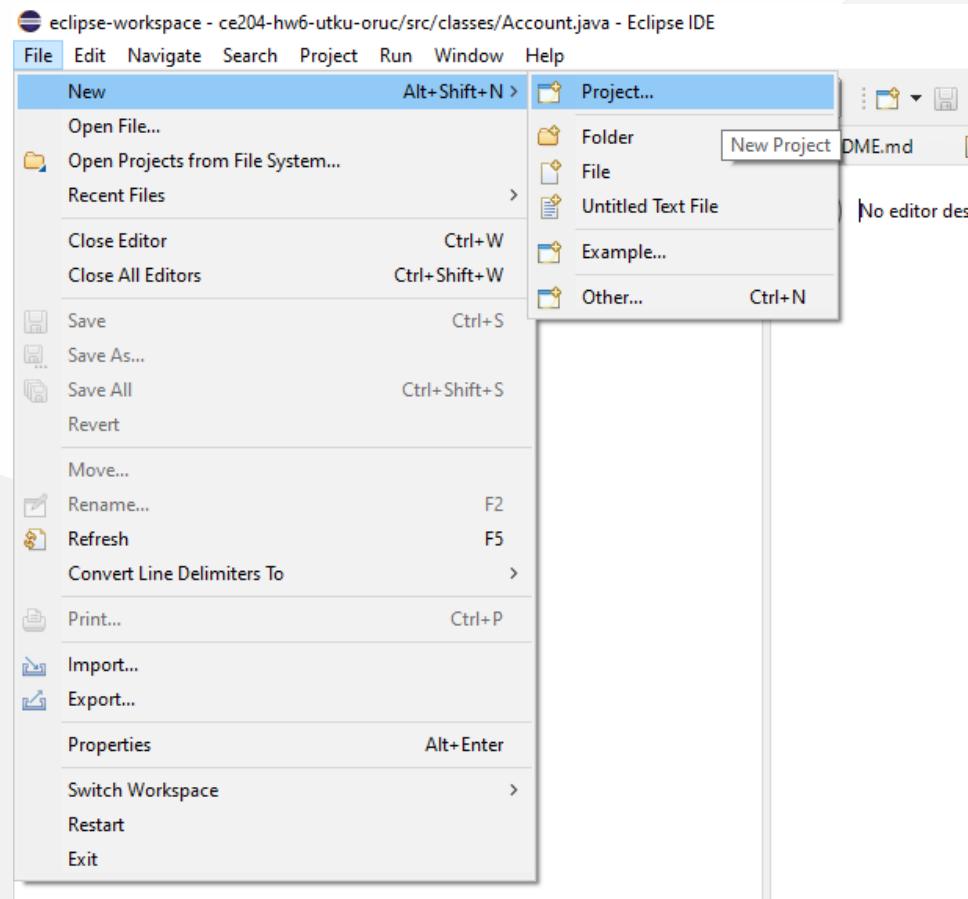


The screenshot shows a Windows File Explorer window. The address bar indicates the path: This PC > Windows (C:) > Users > ugur.coruh > eclipse > cpp-2022-09 > eclipse >. The main area displays a list of files and folders:

	Name	Date modified	Type	Size
...	configuration	1.10.2022 20:08	File folder	
...	dropins	1.10.2022 19:53	File folder	
...	plugins	1.10.2022 19:53	File folder	
...	readme	1.10.2022 19:53	File folder	
...	.eclipseproduct	31.08.2022 22:06	ECLIPSEPRODUCT...	1 KB
...	eclipse.exe	31.08.2022 23:04	Application	519 KB
...	eclipse.ini	1.10.2022 19:53	Configuration sett...	2 KB
...	eclipsec.exe	31.08.2022 23:04	Application	231 KB
...	notice.html	22.08.2022 21:08	Chrome HTML Do...	10 KB

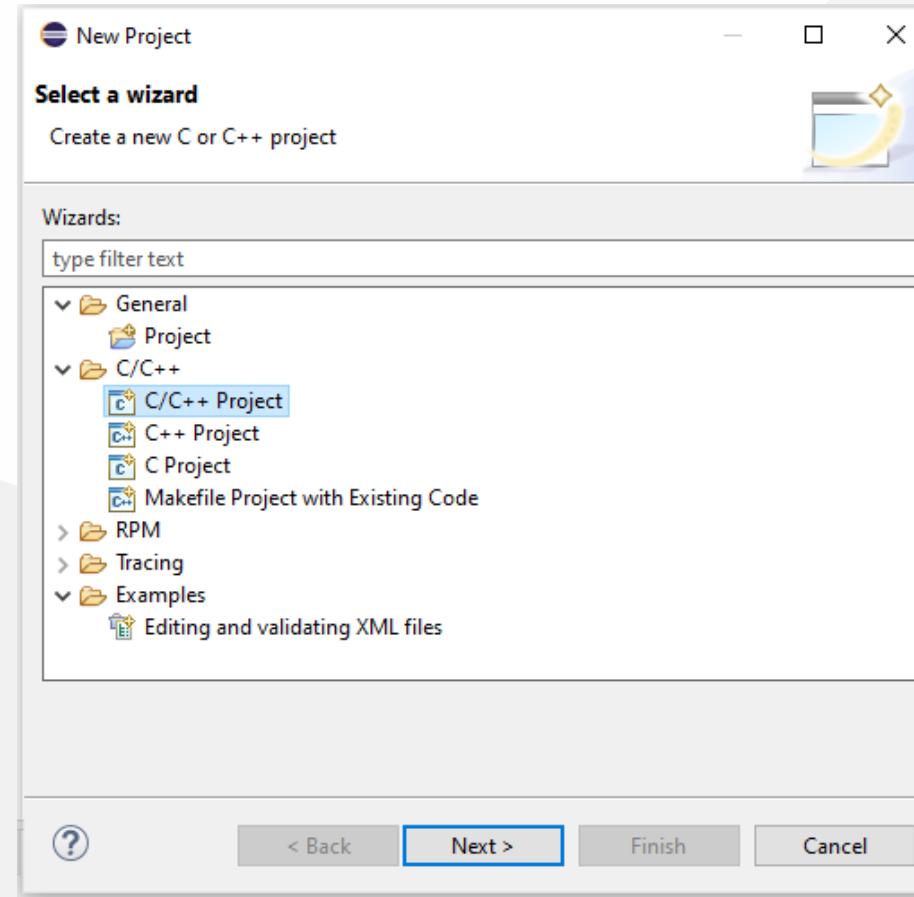
Eclipse (C/C++) - Compile Only / Debugging Has Problem (8)

- File -> New -> Project



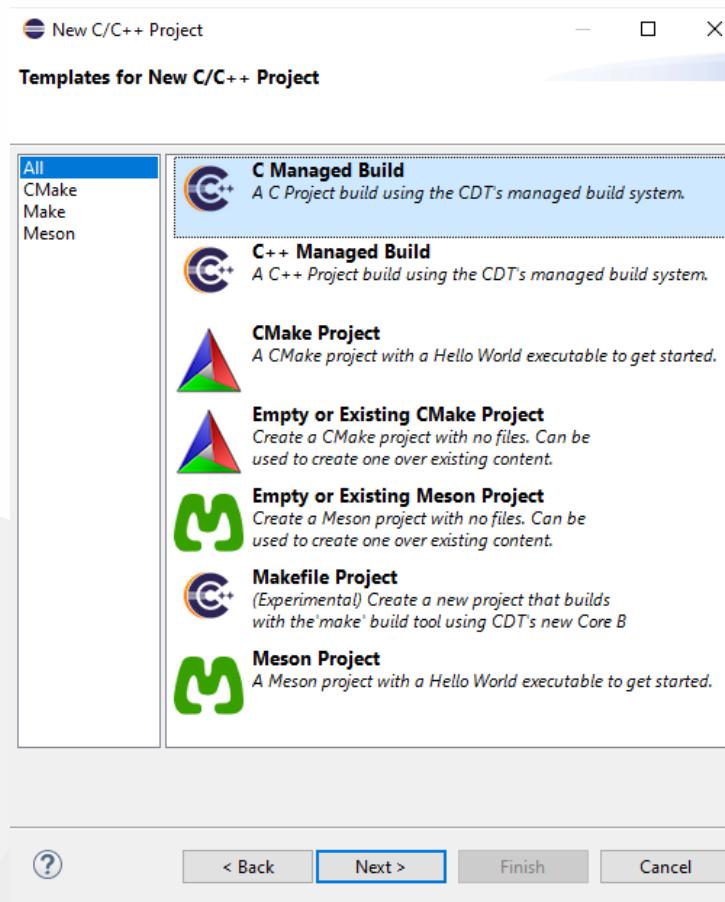
Eclipse (C/C++) - Compile Only / Debugging Has Problem (9)

- Select C/C++ Project



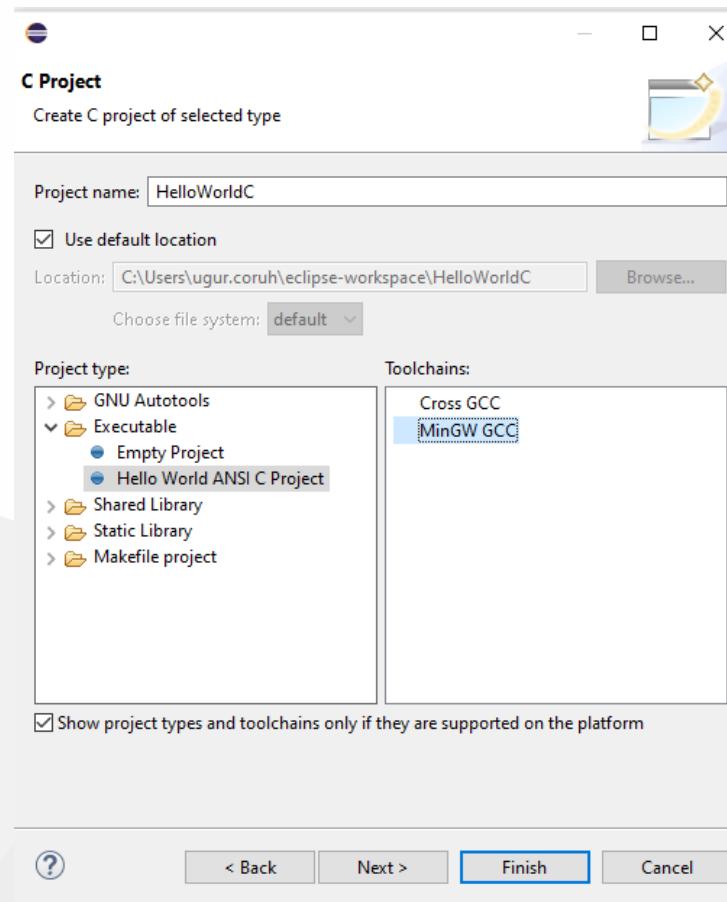
Eclipse (C/C++) - Compile Only / Debugging Has Problem (10)

- Select C Managed Build, Eclipse CDT will do job for us.



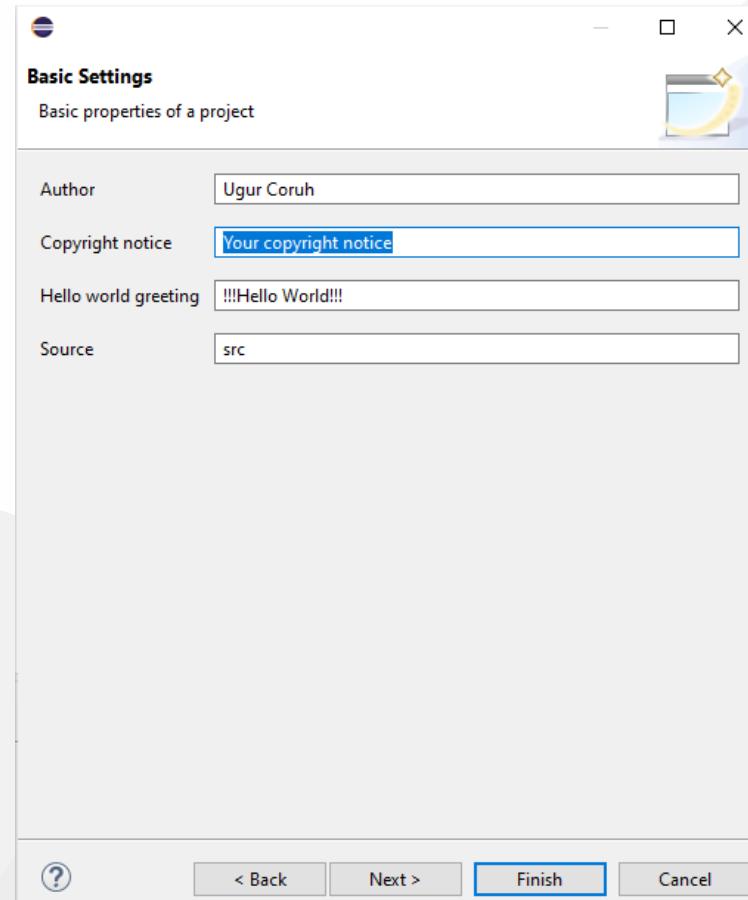
Eclipse (C/C++) - Compile Only / Debugging Has Problem (11)

- Give project name and select a basic template executable with MinGW GCC.



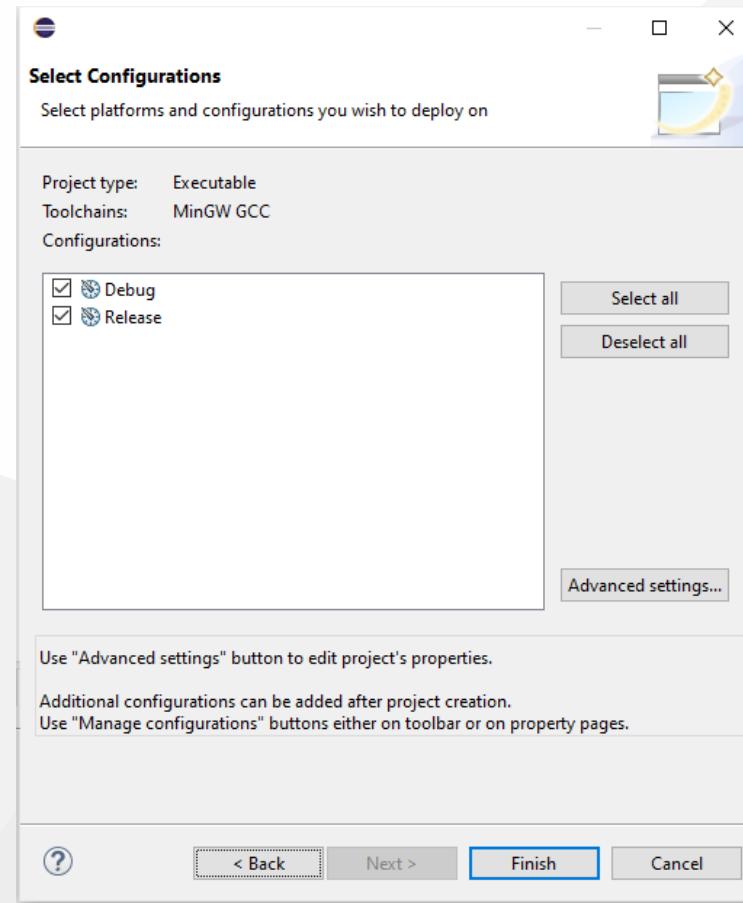
Eclipse (C/C++) - Compile Only / Debugging Has Problem (12)

- Configura Basic Settings



Eclipse (C/C++) - Compile Only / Debugging Has Problem (13)

- There are default Debug and Release configurations you can add your customized configurations from Advanced Settings.



- Project settings will be C Select Debug/Release configuration and then Build Application Project->Build All (Ctrl+B)
- HelloWorldC.exe will be generated

```
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\HelloWorldC.o" "..\\src\\HelloWorldC.c"
gcc -o HelloWorldC.exe "src\\HelloWorldC.o"
```

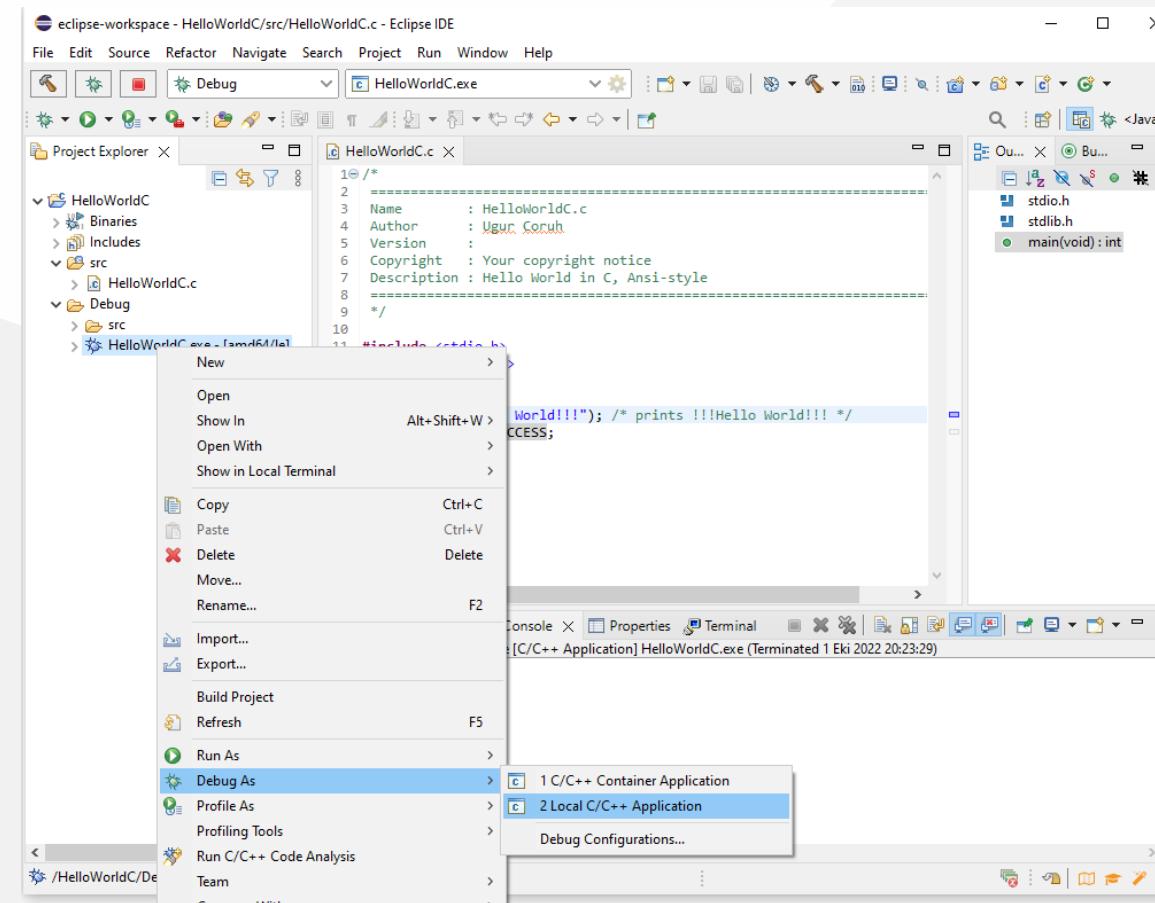
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HelloWorldC/src/HelloWorldC.c - Eclipse IDE
- Toolbar:** Includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help, and various tool icons.
- Project Explorer:** Shows the project structure with nodes for HelloWorldC, src (containing HelloWorldC.c), and Debug (containing HelloWorldC.exe).
- Code Editor:** Displays the content of HelloWorldC.c:1/*
2=====
3 Name : HelloWorldC.c
4 Author : Ugur Coruh
5 Version :
6 Copyright : Your copyright notice
7 Description: Hello World in C, Ansi-style
8=====
9 */
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 int main(void) {
14 puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
15 return EXIT_SUCCESS;
16 }
17
18
- Output View:** Shows the build log with the following entries:

```
20:19:15 **** Rebuild of configuration Debug for project HelloWorldC ****  
Info: Internal Builder is used for build  
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\HelloWorldC.o" "..\\src\\HelloWorldC.c"  
gcc -o HelloWorldC.exe "src\\HelloWorldC.o"  
20:19:16 Build Finished. 0 errors, 0 warnings. (took 610ms)
```

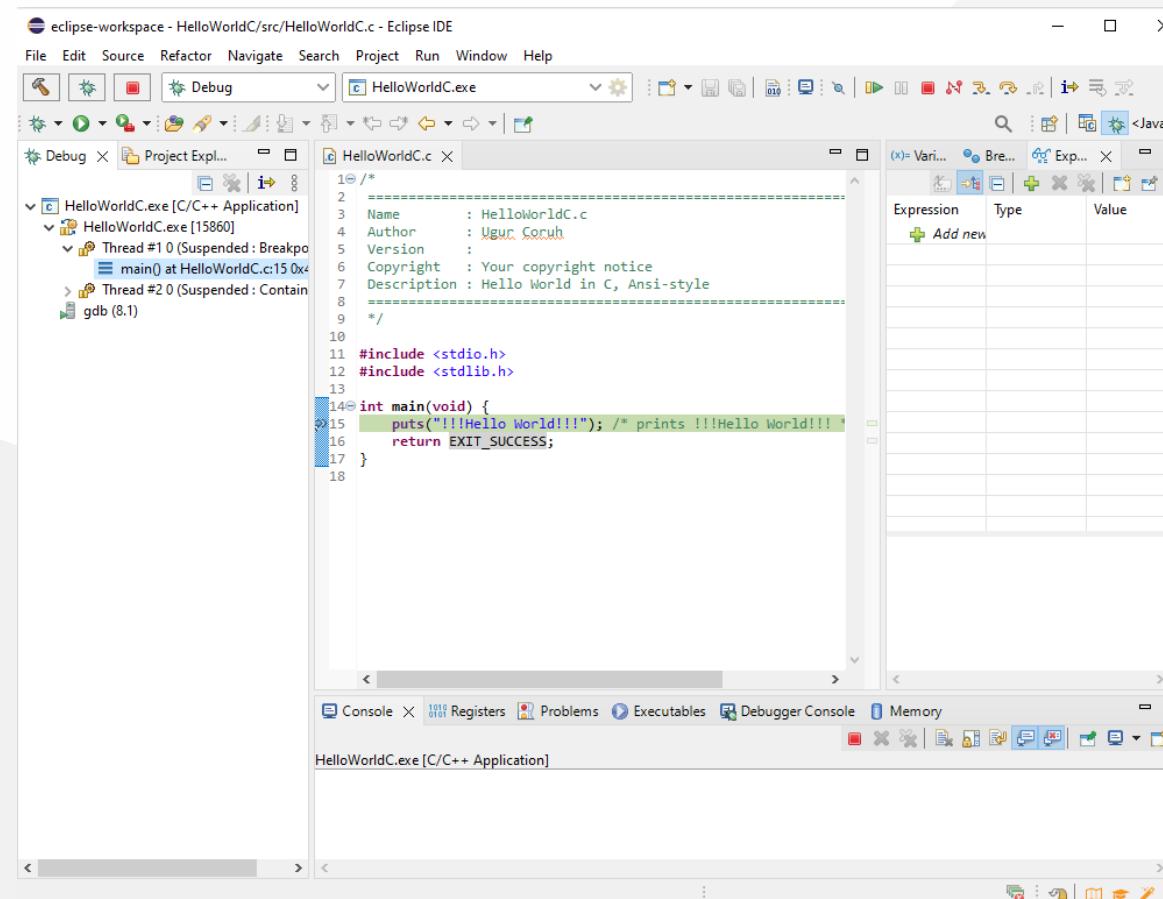
Eclipse (C/C++) - Compile Only / Debugging Has Problem (15)

- Before build if you want to debug application select debug configuration, put your breakpoints and then Build application again.
- Right click the generated executable Debug As -> Local C/C++ Application



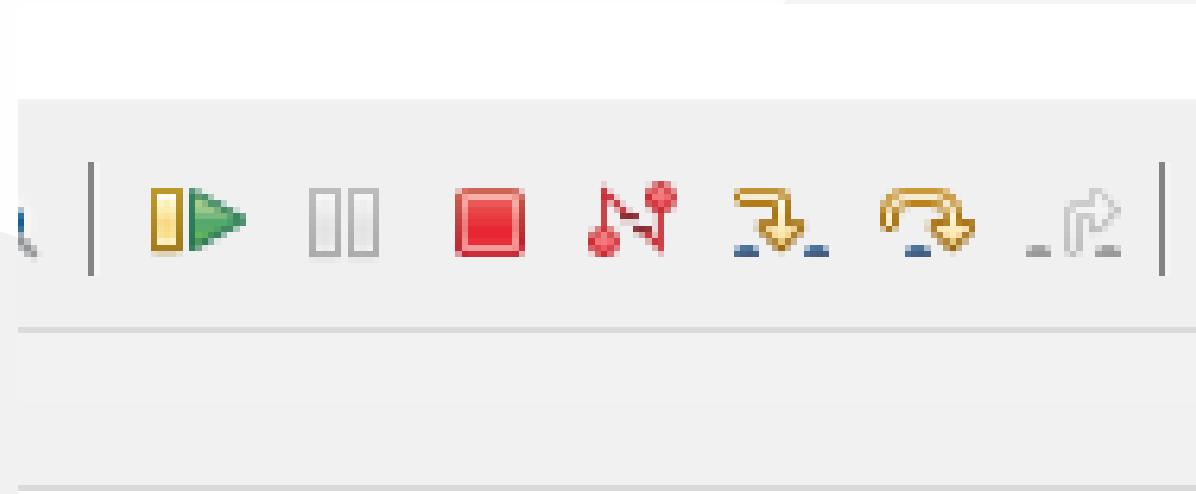
Eclipse (C/C++) - Compile Only / Debugging Has Problem (16)

- Debugger will start and stop at breakpoint as follow.



Eclipse (C/C++) - Compile Only / Debugging Has Problem (16)

- Check debug control shortcuts and use them



Eclipse (C/C++) - Compile Only / Debugging Has Problem (17)

- To watch variables use Expressions and Variables

Expression	Type	Value
(x)= name	char [20]	0x61fe00
(x)= name char	char	8 '\b'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	72 'H'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	0 '\0'
(x)= name char	char	112 'p'
(x)= name char	char	38 '&'
(x)= name char	char	108 'l'
(x)= name char	char	0 '\0'
Add new		

Eclipse (C/C++) - Compile Only / Debugging Has Problem (18)

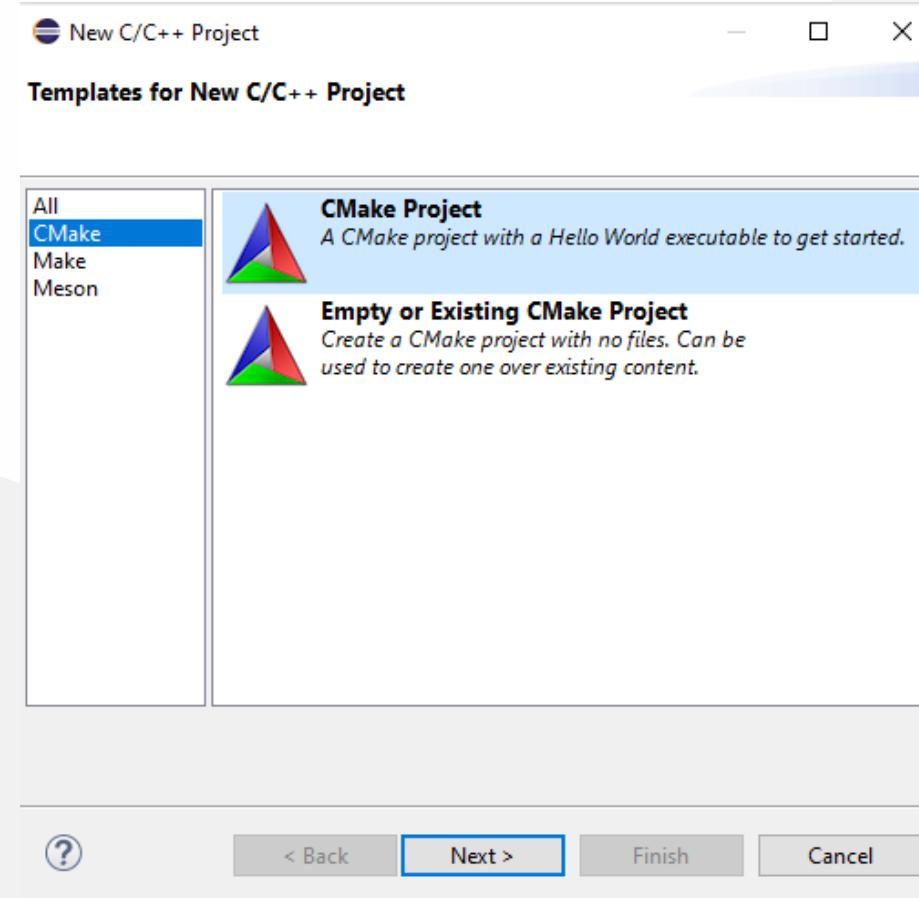
Name	Type	Value
(x)= name	char [20]	0x61fe00
(x)= name[0]	char	8 '\b'
(x)= name[1]	char	0 '\0'
(x)= name[2]	char	0 '\0'
(x)= name[3]	char	0 '\0'
(x)= name[4]	char	0 '\0'
(x)= name[5]	char	0 '\0'
(x)= name[6]	char	0 '\0'
(x)= name[7]	char	0 '\0'
(x)= name[8]	char	72 'H'
(x)= name[9]	char	0 '\0'
(x)= name[10]	char	0 '\0'
(x)= name[11]	char	0 '\0'
(x)= name[12]	char	0 '\0'
(x)= name[13]	char	0 '\0'
(x)= name[14]	char	0 '\0'
(x)= name[15]	char	0 '\0'
(x)= name[16]	char	112 'p'
(x)= name[17]	char	38 '&'
(x)= name[18]	char	108 'l'
(x)= name[19]	char	0 '\0'

Eclipse (C/C++) - Compile Only / Debugging Has Problem (19)

- for more visit eclipse webpage
 - [Effective Techniques for Debugging C & C++ | The Eclipse Foundation](#)
 - [Help - Eclipse IDE](#)

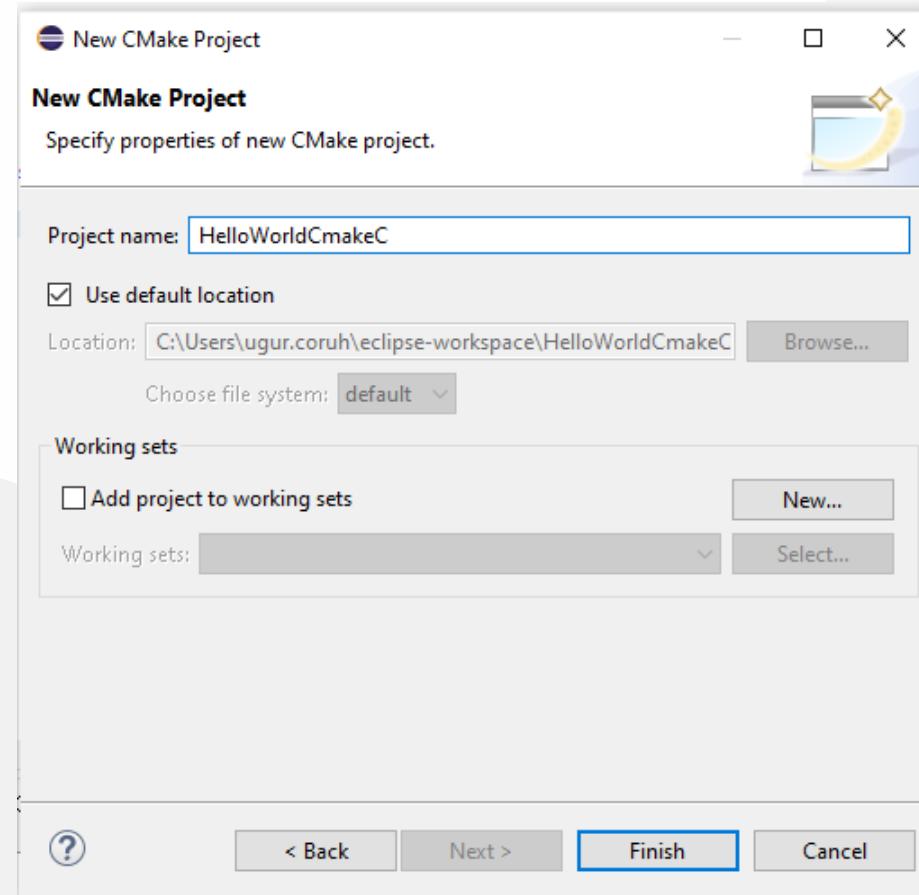
Eclipse (C/C++) - Compile Only / Debugging Has Problem (20)

- Generate CMAKE project from new Project and Select CMake Project Template



Eclipse (C/C++) - Compile Only / Debugging Has Problem (21)

- Give project name



Eclipse (C/C++) - Compile Only / Debugging Has Problem (22)

- This will generate default C++ Hello World project

The screenshot shows the Eclipse IDE interface with the following details:

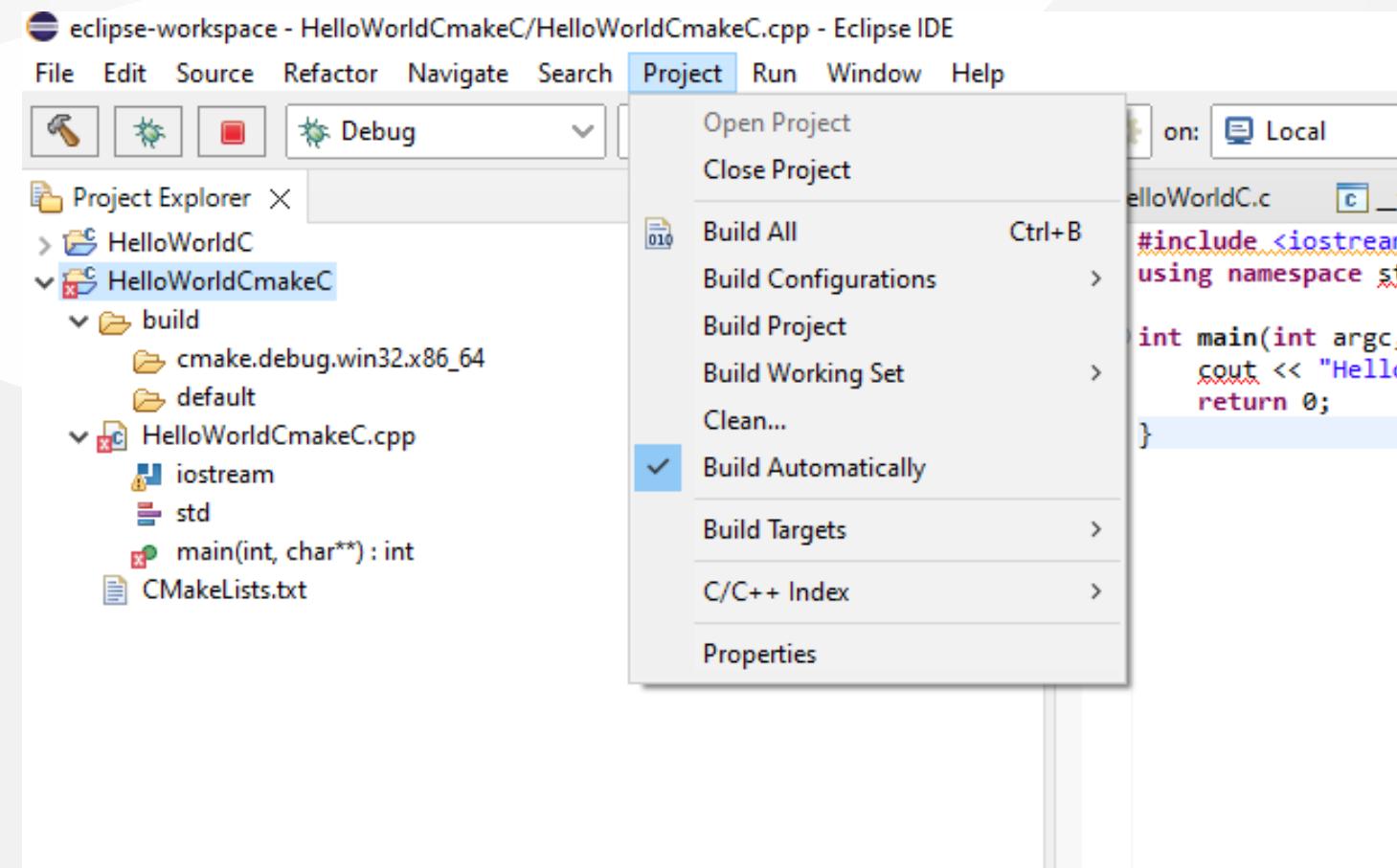
- Title Bar:** eclipse-workspace - HelloWorldCmakeC/HelloWorldCmakeC.cpp - Eclipse IDE
- Toolbar:** Includes icons for Run, Stop, Build, and Debug (which is selected).
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbars:** On the right side, there are toolbars for Local, Build, and Run.
- Project Explorer:** Shows the project structure:
 - HelloWorldC
 - >HelloWorldCmakeC
 - build
 - cmake.debug.win32.x86_64
 - default
 - HelloWorldCmakeC.cpp
 - iostream
 - std
 - main(int, char**) : int
 - CMakeLists.txt
- Editor Area:** Displays the code for `HelloWorldC.c`. There are several error markers (red exclamation points) on the first few lines of the code, indicating compilation errors.

```
#include <iostream>
using namespace std;

int main(int argc, char **argv) {
    cout << "Hello world";
    return 0;
}
```

Eclipse (C/C++) - Compile Only / Debugging Has Problem (23)

- Build Project



Eclipse (C/C++) - Compile Only / Debugging Has Problem (24)

- It will give following errors, for missing configurations. These errors are generated by CMAKE
- Then clean and rebuild again.

Errors occurred during the build.

Errors running builder 'CDT Core Builder' on project 'HelloWorldCmakeC'.

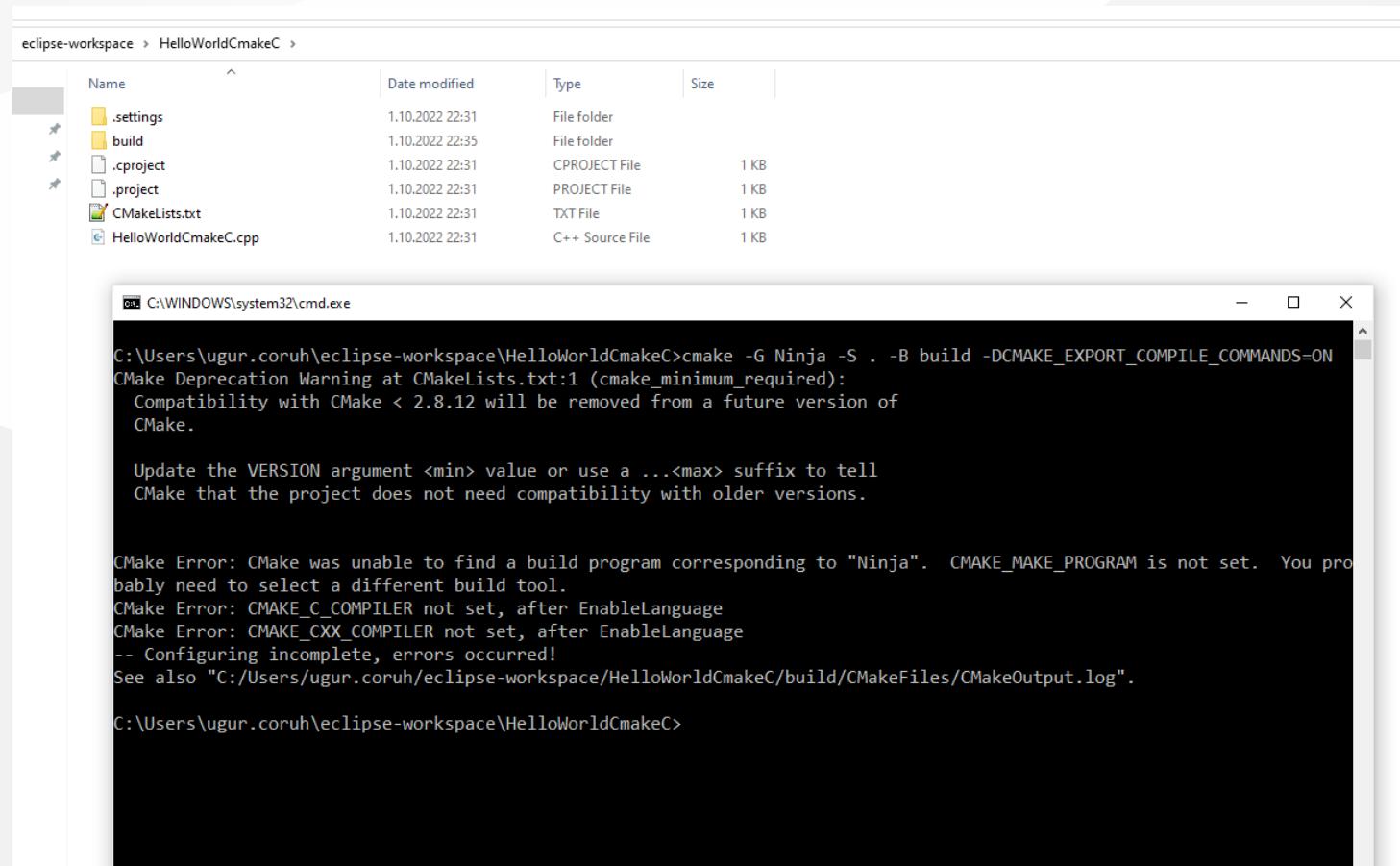
Resource '/HelloWorldCmakeC/build/cmake.debug.win32.x86_64/compile_commands.json' does not exist.

Resource '/HelloWorldCmakeC/build/cmake.debug.win32.x86_64/compile_commands.json' does not exist.

Resource '/HelloWorldCmakeC/build/cmake.debug.win32.x86_64/compile_commands.json' does not exist.

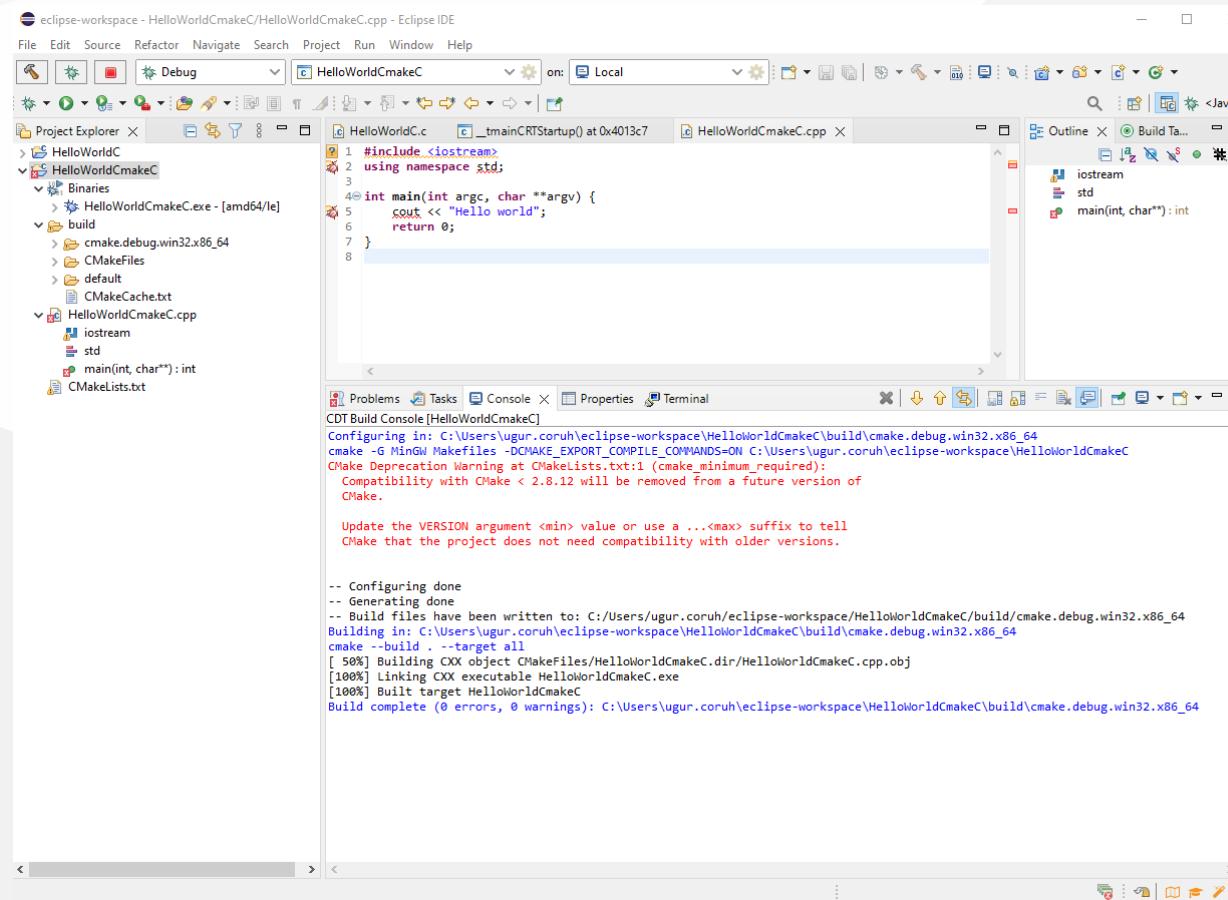
Resource '/HelloWorldCmakeC/build/cmake.debug.win32.x86_64/compile_commands.json' does not exist.

Eclipse (C/C++) - Compile Only / Debugging Has Problem (25)



Eclipse (C/C++) - Compile Only / Debugging Has Problem (26)

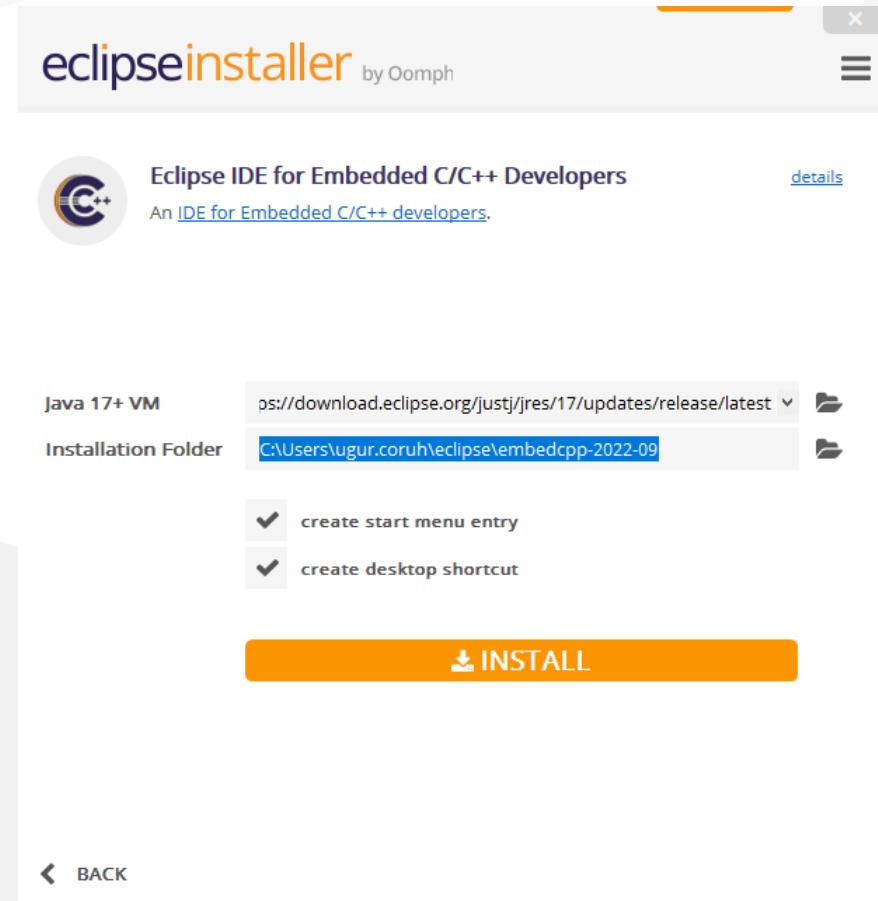
- After this operation first Clean project from Project menu and then Build All again



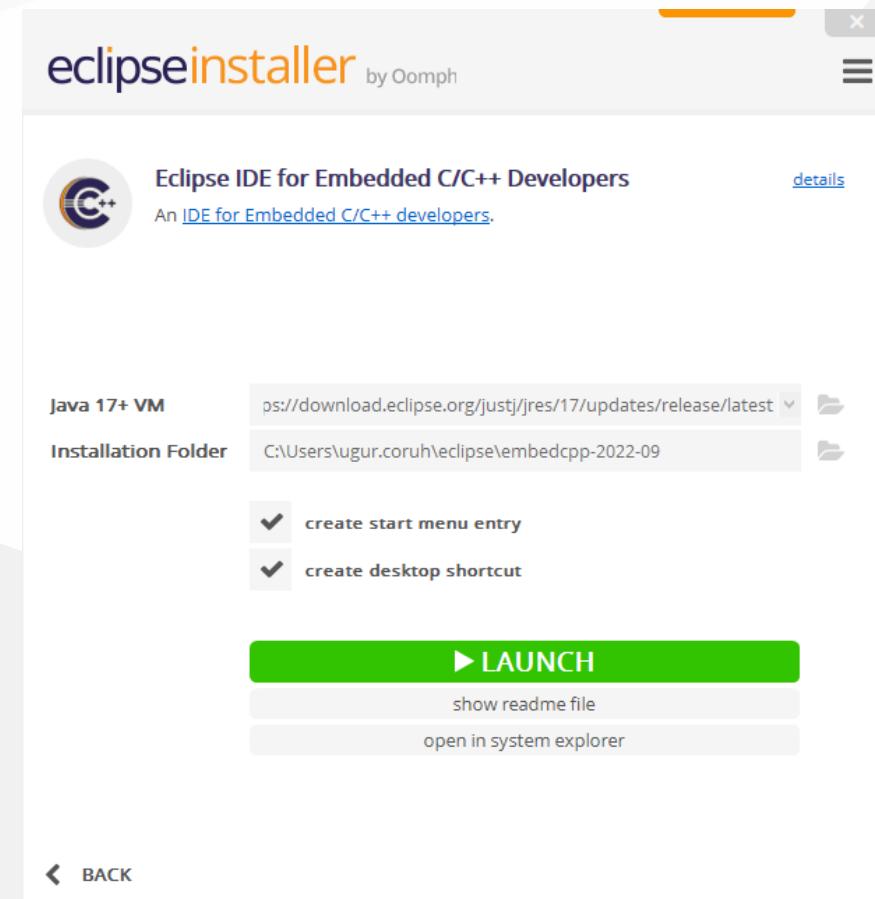
Eclipse (C/C++) - Compile Only / Debugging Has Problem (27)

- Eclipse with CMake project on windows
- JV - Science and stuff.

Eclipse (C/C++) - Compile Only / Debugging Has Problem (28)

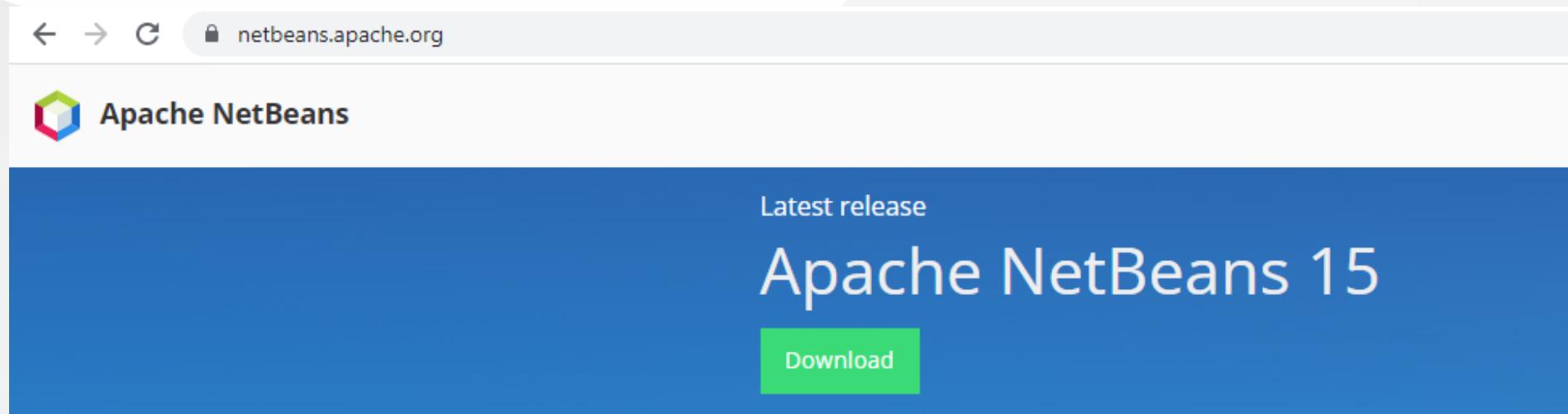


Eclipse (C/C++) - Compile Only / Debugging Has Problem (29)

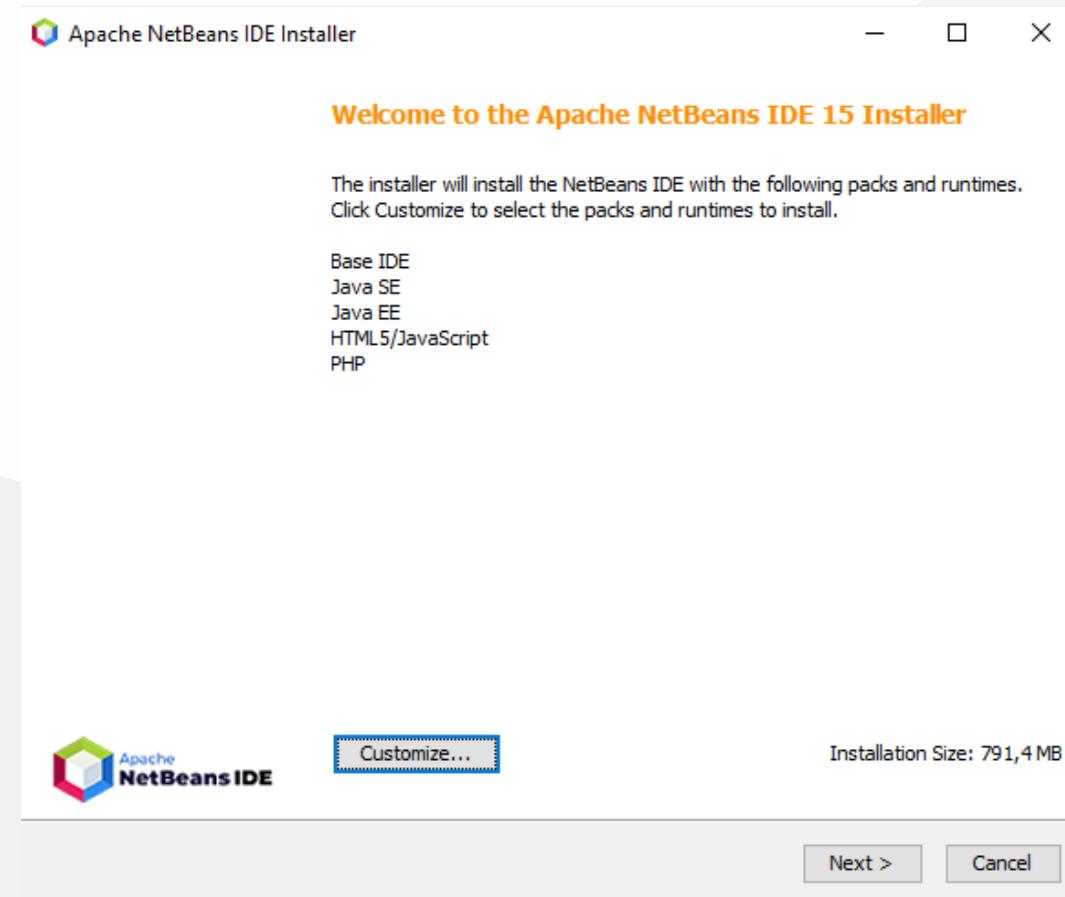


Netbeans (C/C++) - Manual Build/Clean/Run Command Setting Not Good Option for C/C++ Development (1)

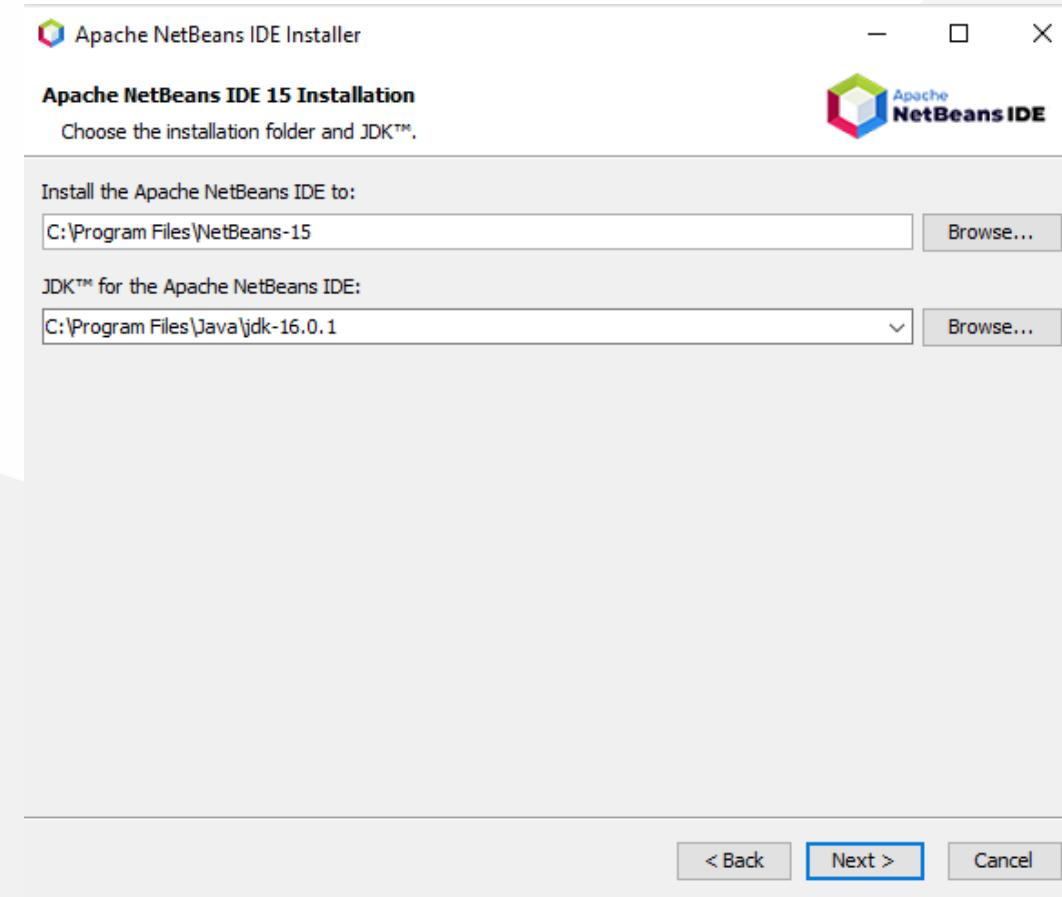
- <https://netbeans.apache.org/>
- C and C++ Tutorials



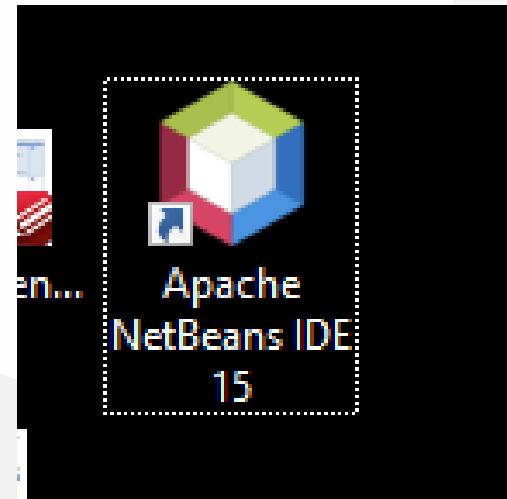
Netbeans (C/C++) - Manual Build/Clean/Run Command Setting Not Good Option for C/C++ Development (2)



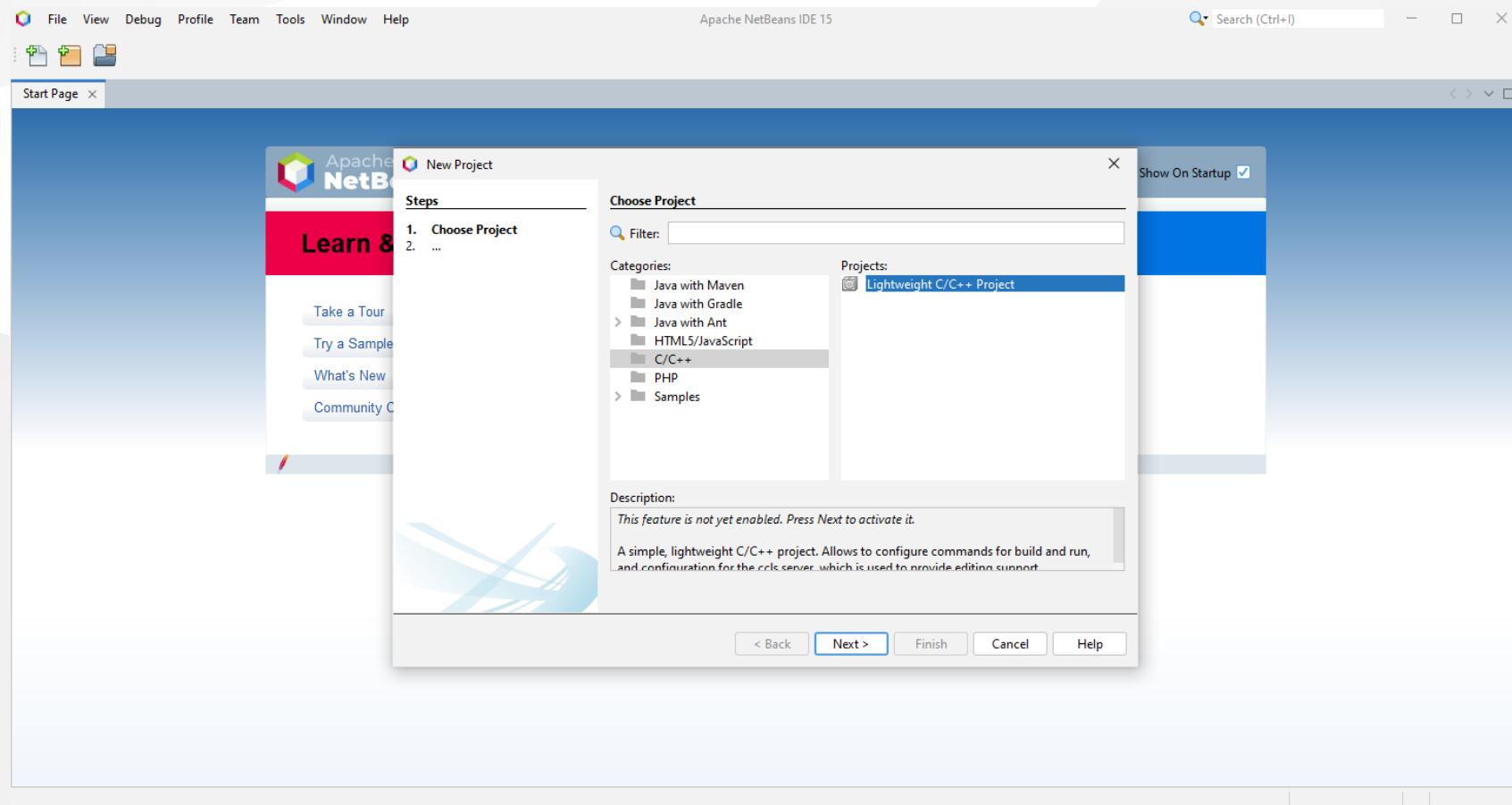
Netbeans (C/C++) - Manual Build/Clean/Run Command Setting Not Good Option for C/C++ Development (3)



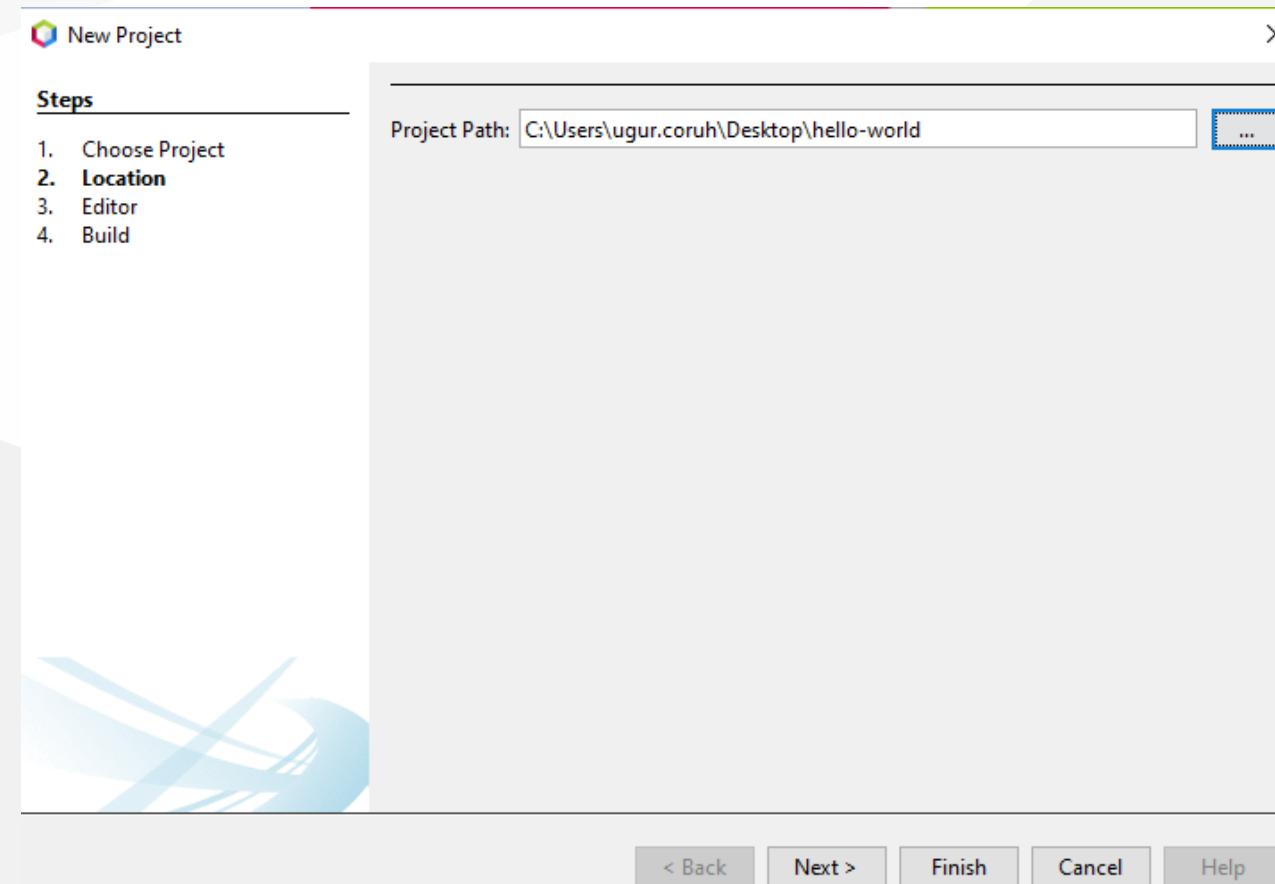
Netbeans (C/C++) - Manuel Build/Clean/Run Command Setting Not Good Option for C/C++ Development (4)



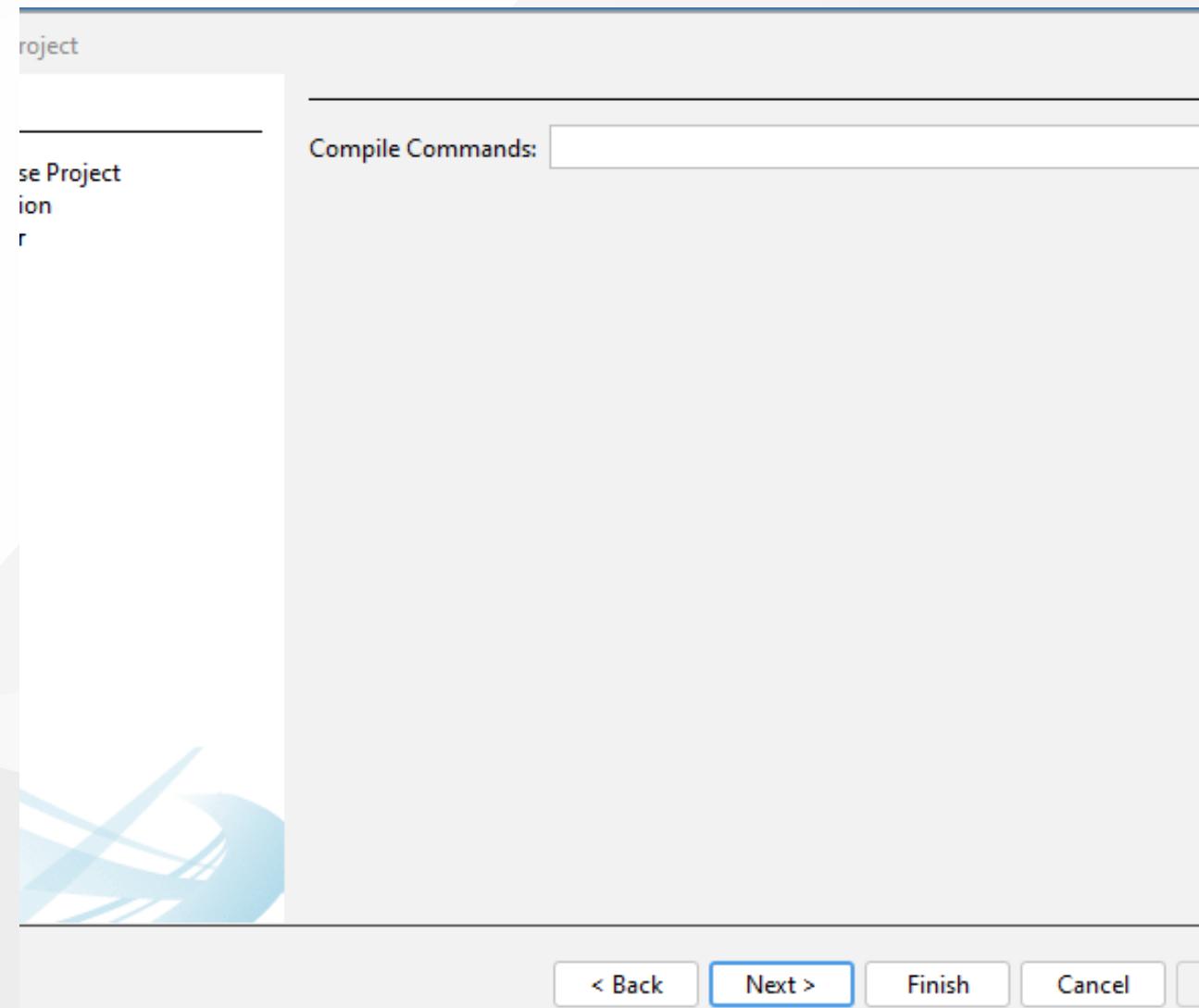
Netbeans (C/C++) - Manual Build/Clean/Run Command Setting Not Good Option for C/C++ Development (5)



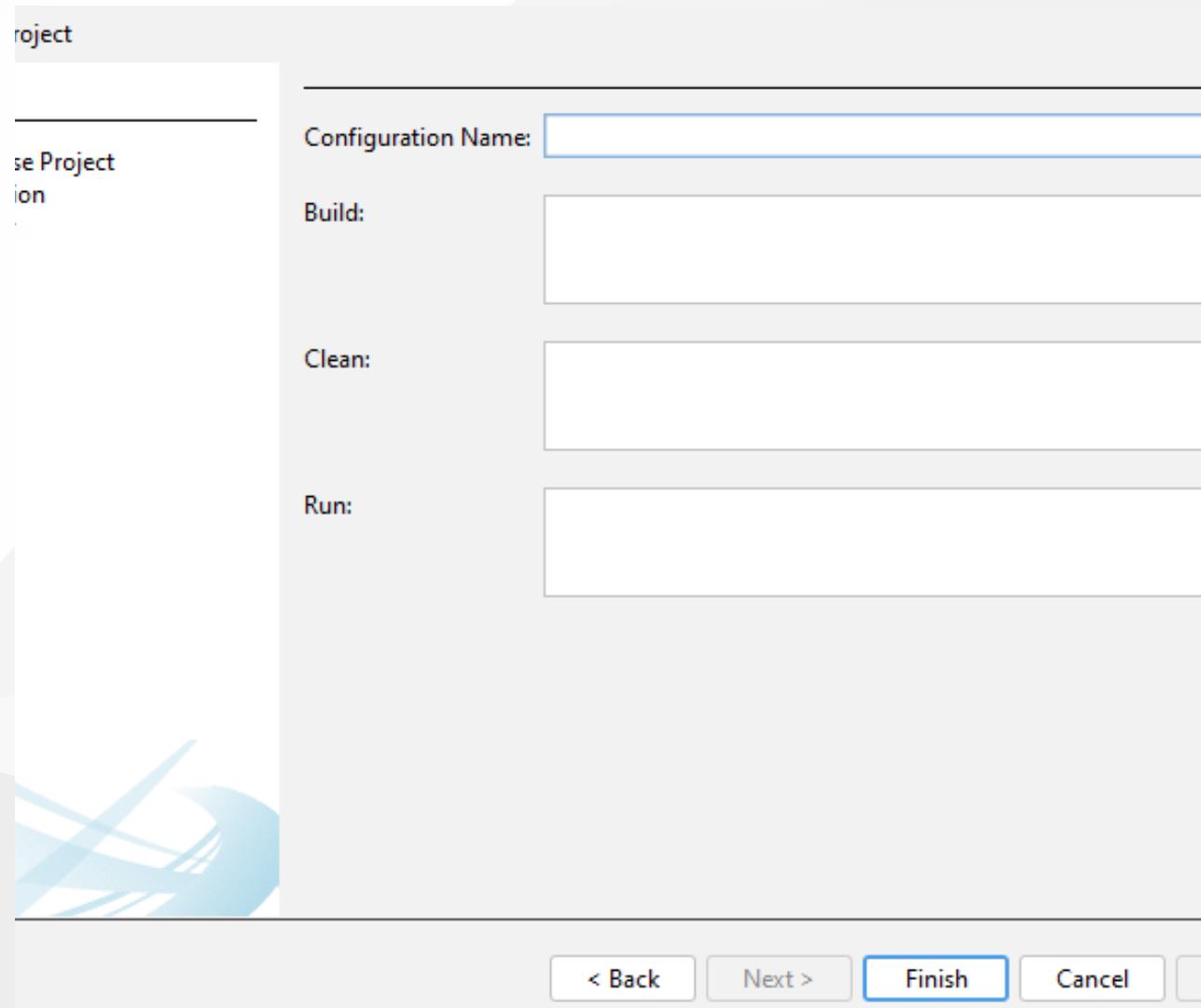
Netbeans (C/C++) - Manual Build/Clean/Run Command Setting Not Good Option for C/C++ Development (6)



Netbeans (C/C++) - Manuel Build/Clean/Run Command Setting Not Good Option for C/C++ Development (7)



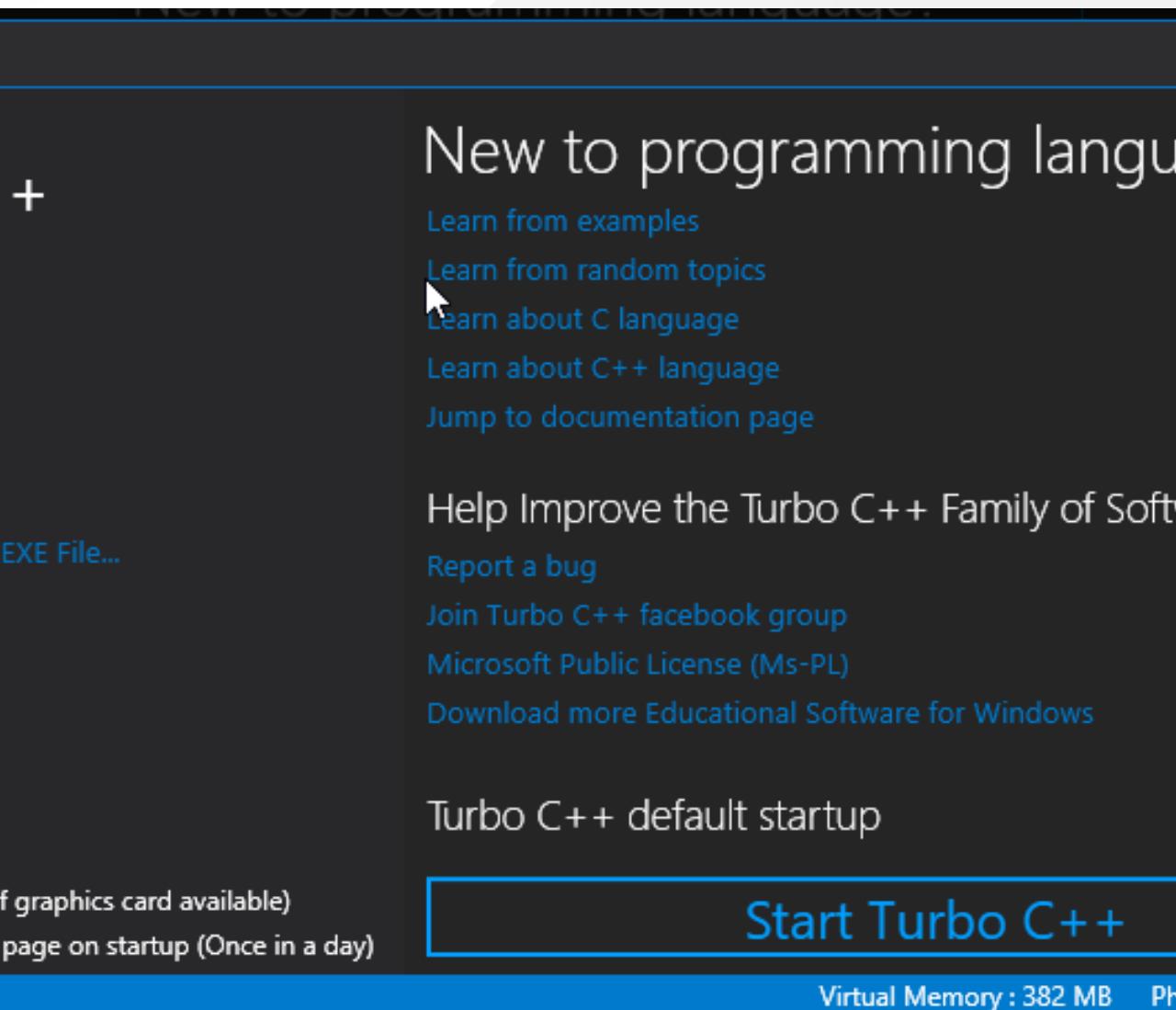
Netbeans (C/C++) - Manuel Build/Clean/Run Command Setting Not Good Option for C/C++ Development (8)



Turbo C/C++ (1)

Download [Turbo.C.3.2.zip](#)

- Download Turbo C++ for Windows 7, 8, 8.1, 10 and Windows 11 (32-64 bit) with full/window screen mode and many more extra features
- [Turbo C++ Shortcuts - C Programming Language Tutorials](#)



Turbo C/C++ (2)

A screenshot of the Turbo C/C++ integrated development environment. The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Help, and Windows. The title bar shows the path \PROJECTS\HELLOW\1.CPP. The code editor window contains the following C++ code:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"hello world c++";
    return 0;
}
```

The status bar at the bottom shows the line number :31 and several keyboard shortcuts: F2 Save, F3 Open, Alt-F9 Compile, F9 Make, and F10 Menu.

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;
}
```

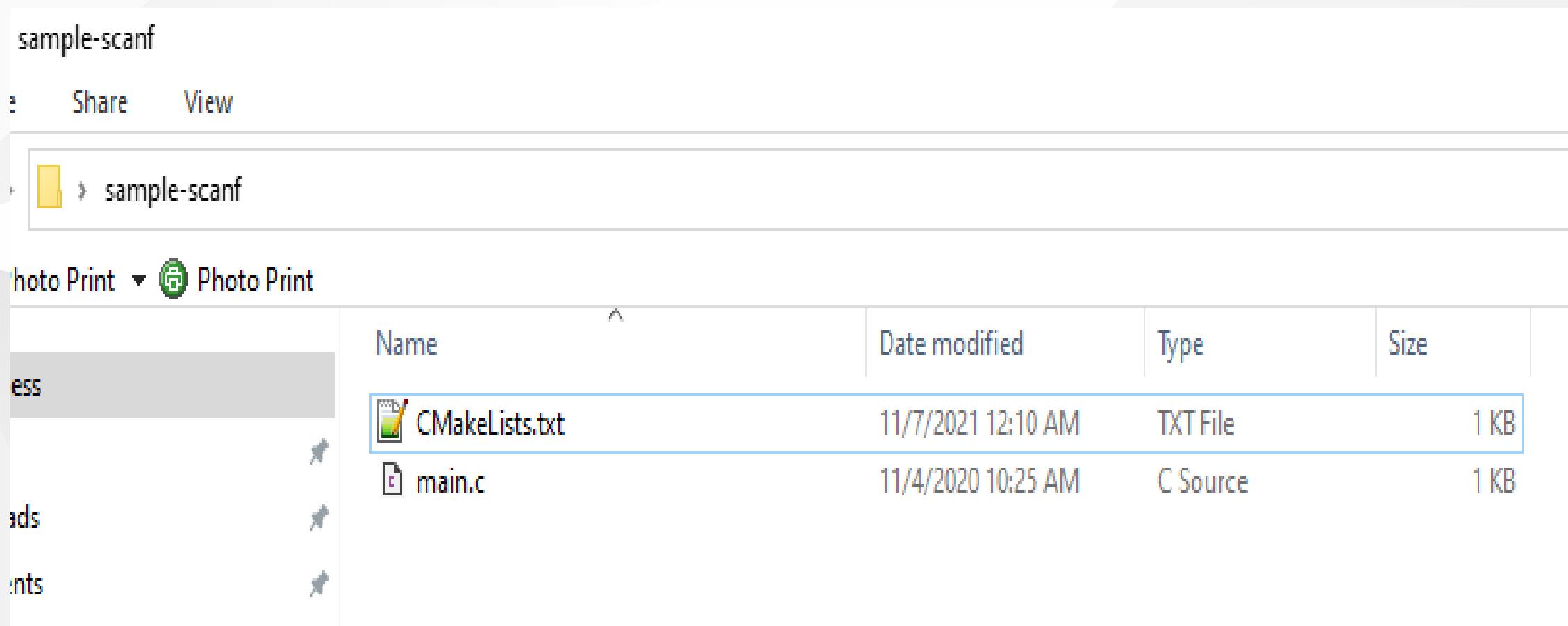
CMakeLists.txt

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



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

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

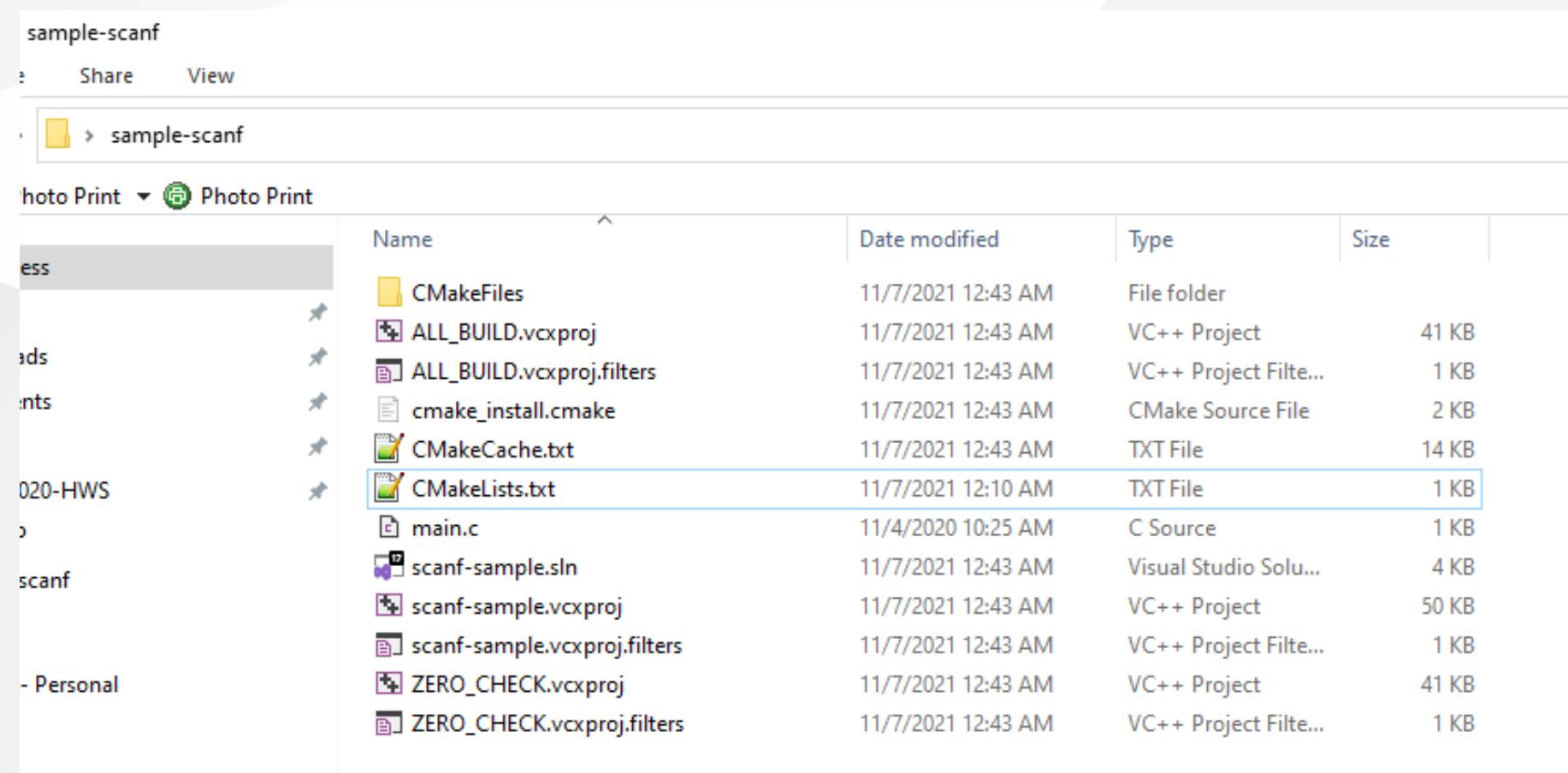
I have Visual Studio 2022 Community Edition Installed on My Computer, for these reason build tools are selected for visual studio environment and the following outputs are generated

```
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



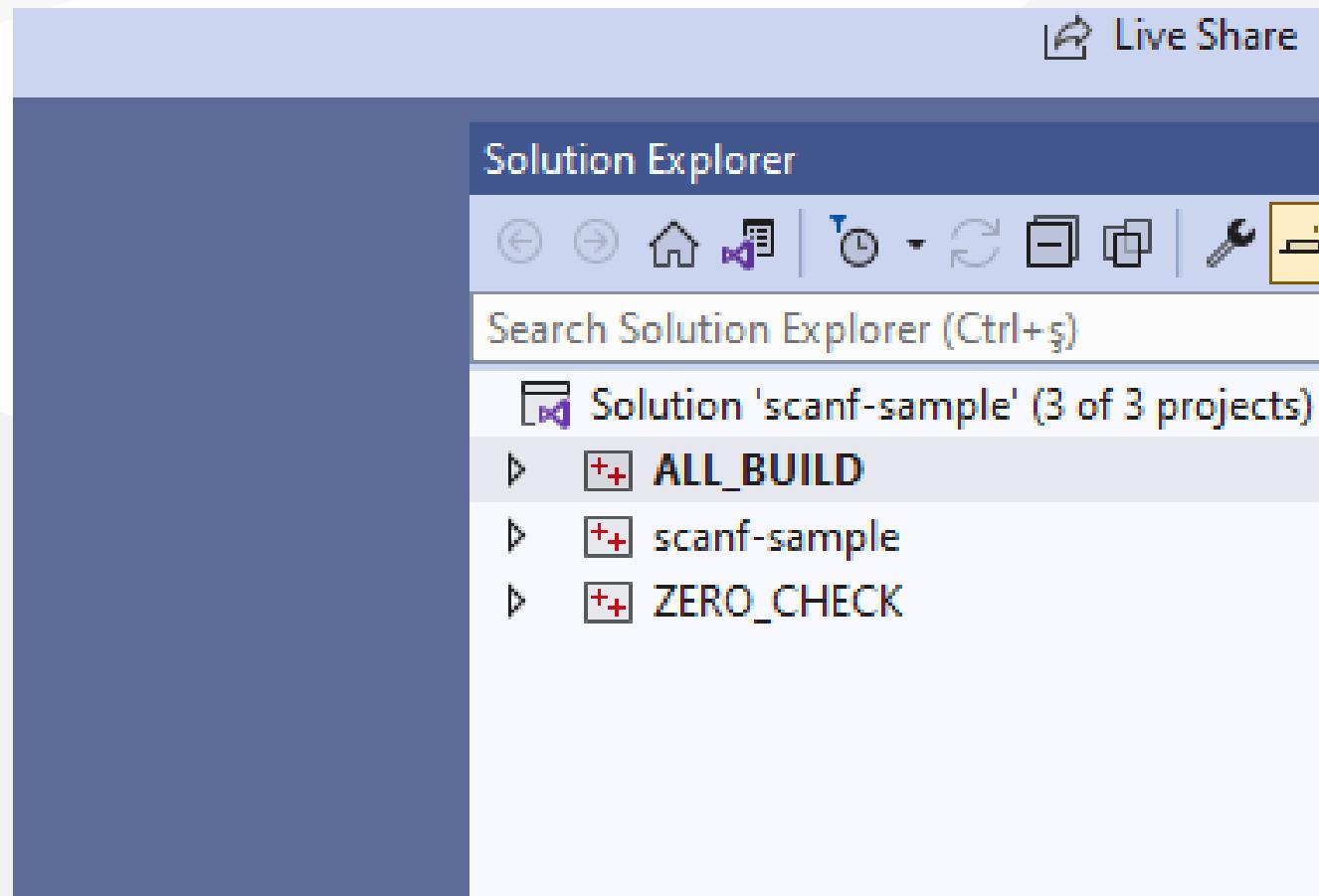
The screenshot shows a Windows File Explorer window with the following details:

- Path:** sample-scanf
- Toolbar:** Share, View, Photo Print, Photo Print
- File List:** A table showing the following files:

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
scansample.sln	11/7/2021 12:43 AM	Visual Studio Solu...	4 KB
scansample.vcxproj	11/7/2021 12:43 AM	VC++ Project	50 KB
scansample.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)

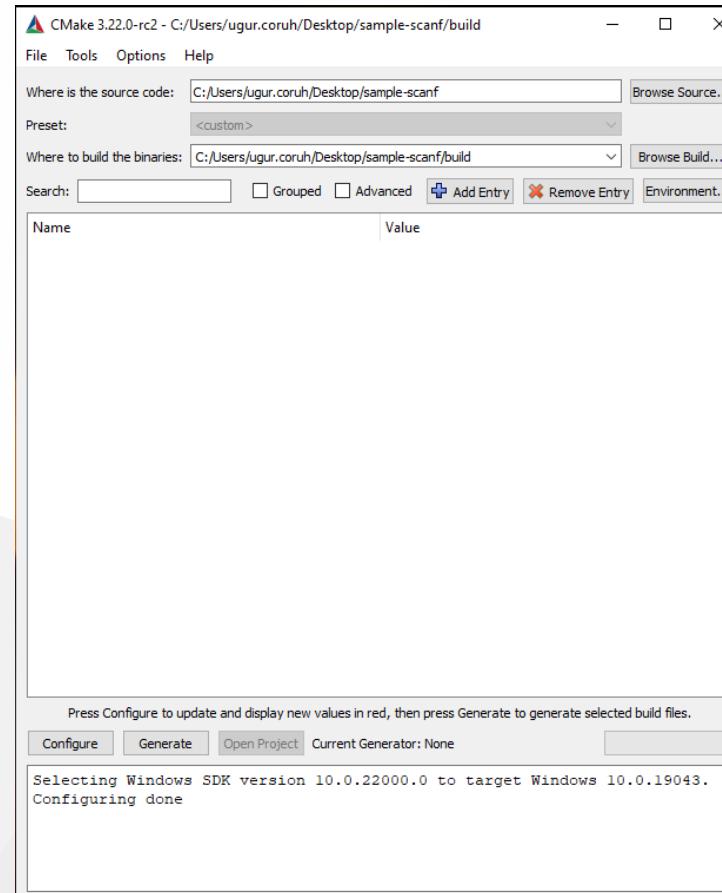
you can make scanf-sample with startup project with right click and then run on visual studio.

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



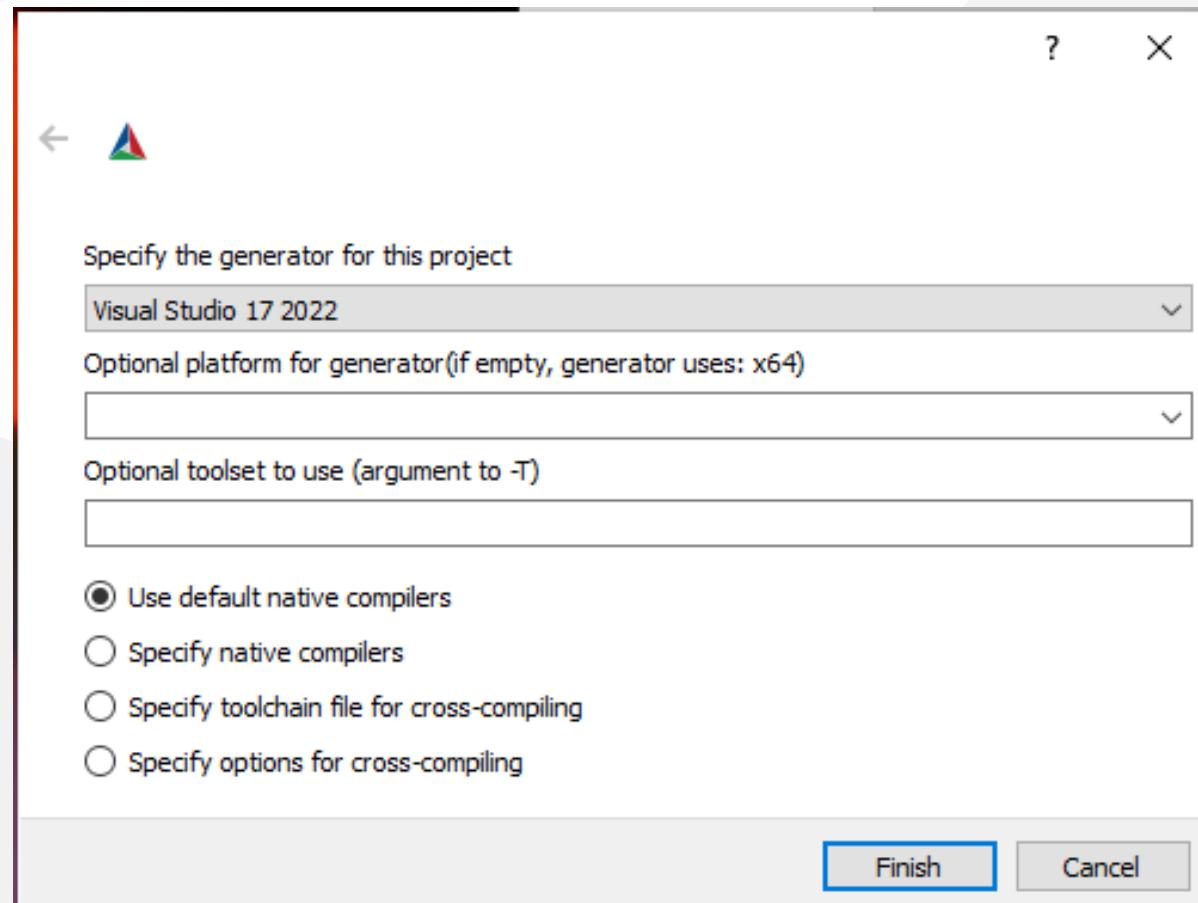
Cmake (C++/C) (12) (Windows Environment)

Open GUI and Select *File-> Delete Cache*

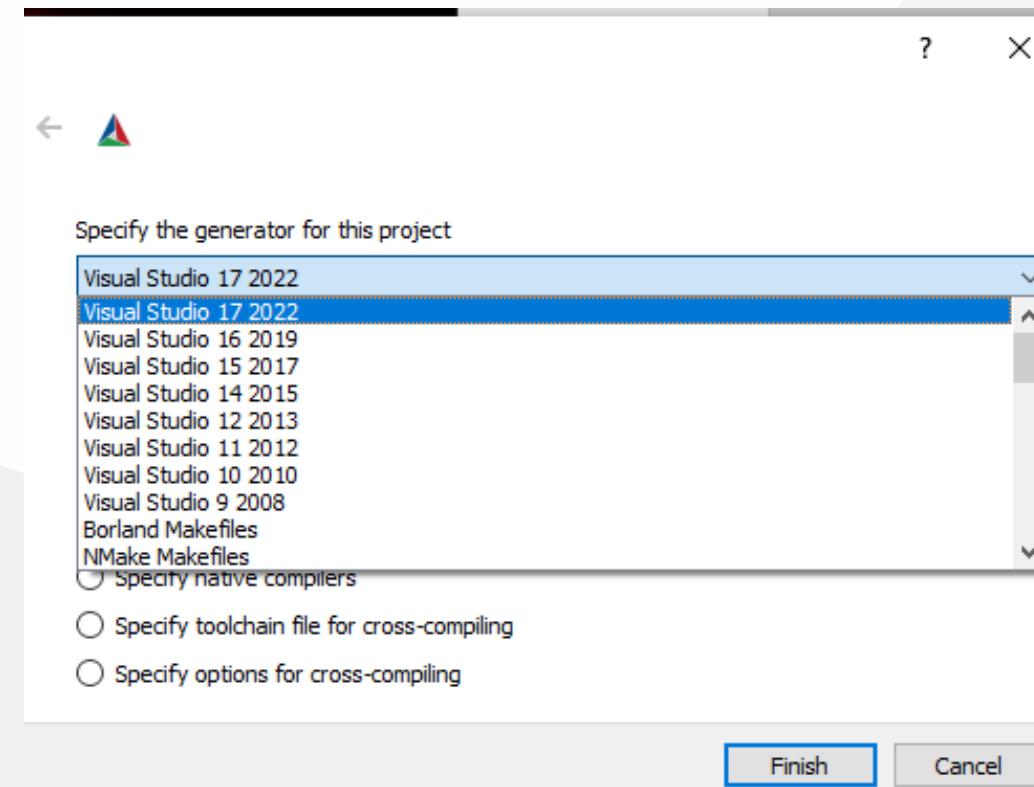


Cmake (C++/C) (13) (Windows Environment)

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](https://github.com/ttroy50/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

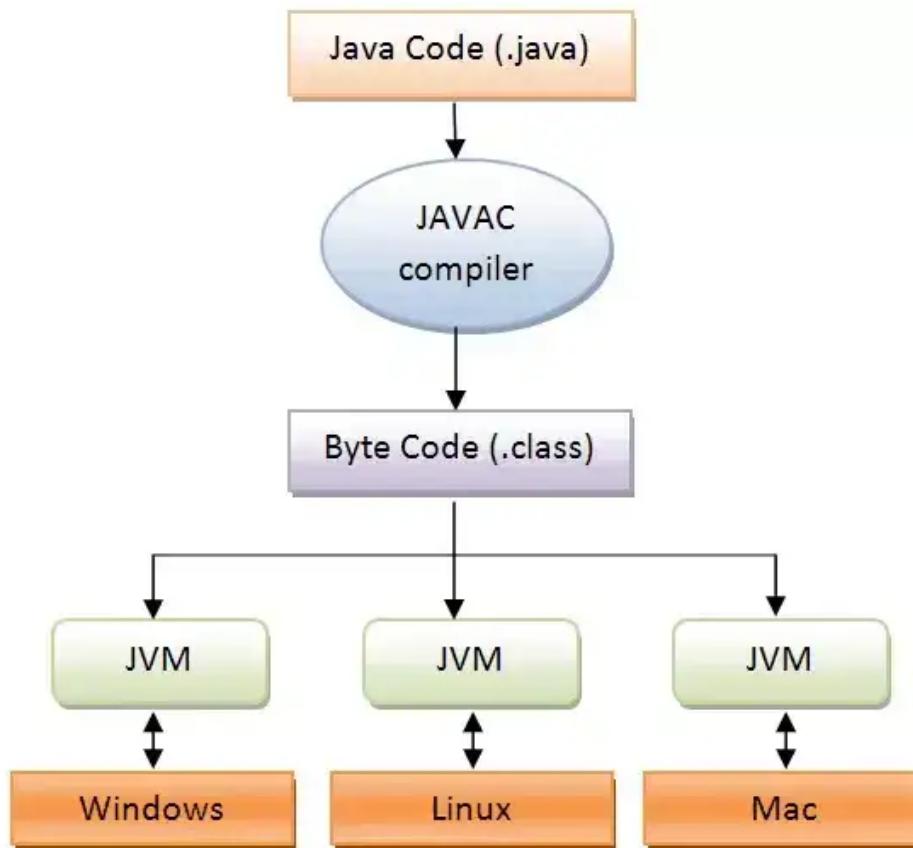
check hello-make sample



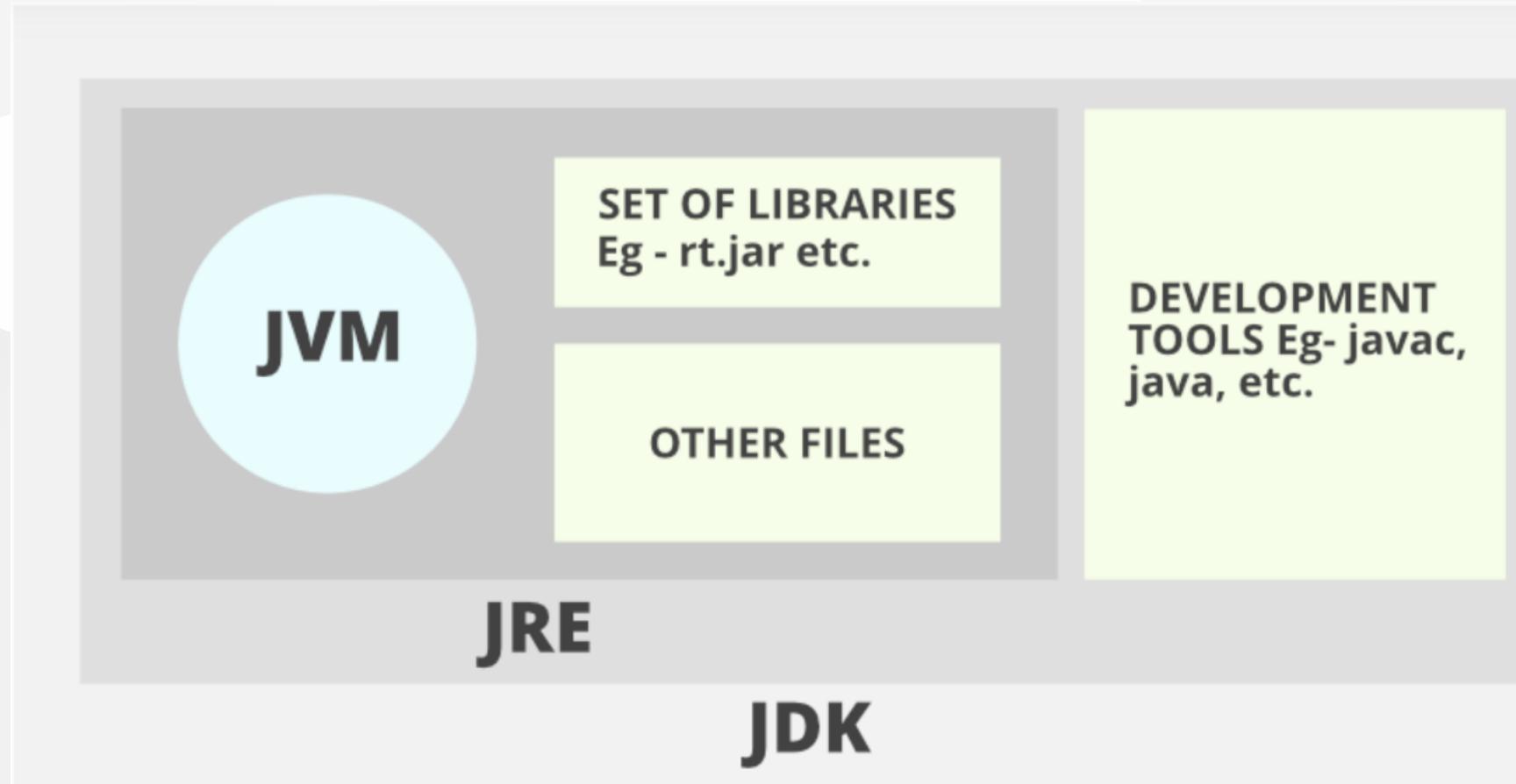
Make (4)

s-and-programming-l > Week-2 > hello-make		
Print		
	Name	Date mod
	compile.bat	11/7/2021
	hello.c	11/2/2021
	Makefile	11/2/2021

JAVA Environment and Development



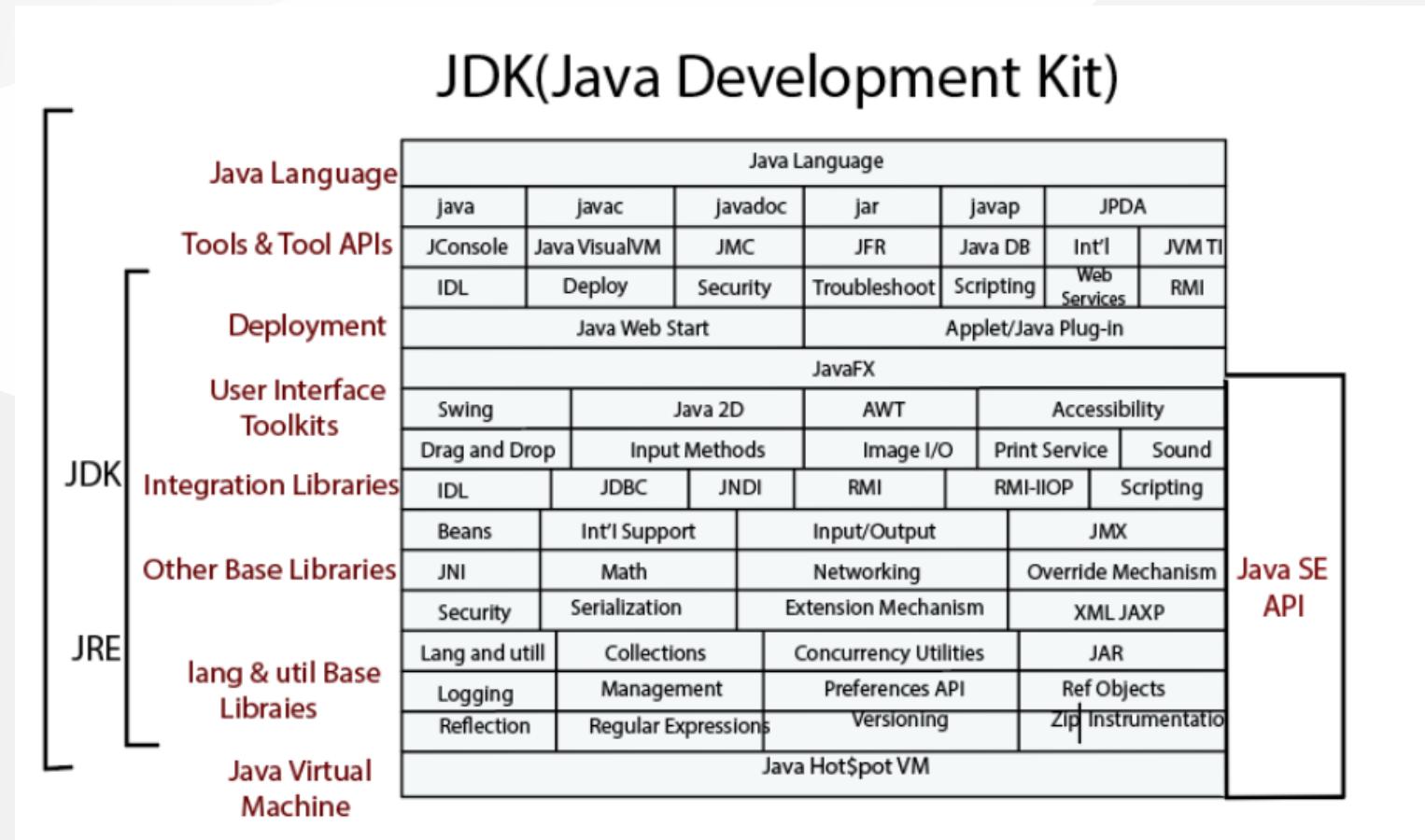
JDK and JRE Setup (1)



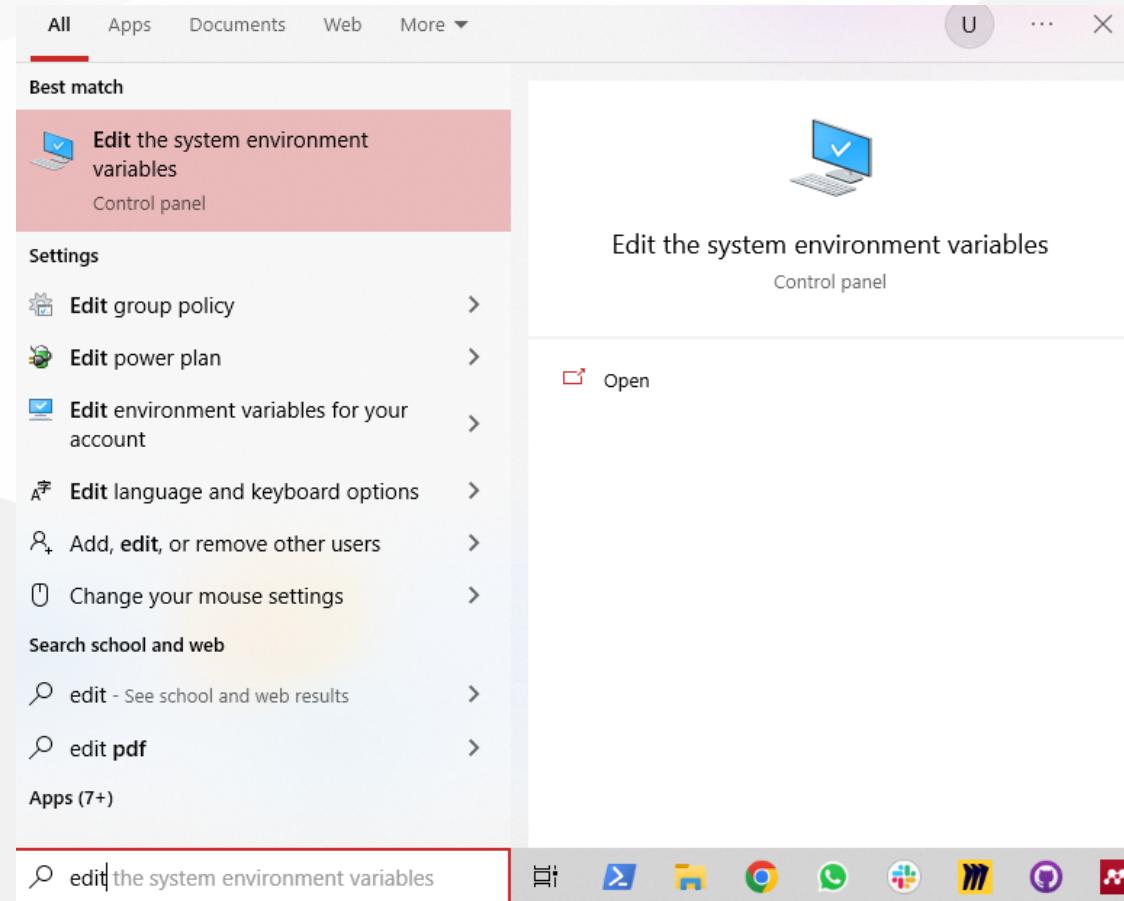
JDK and JRE Setup (2)

- JDK (Java Development Kit) is a Kit that provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) that includes two things
 - Development Tools(to provide an environment to develop your java programs)
 - JRE (to execute your java program).
- JRE (Java Runtime Environment) is an installation package that provides an environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.
- **JVM (Java Virtual Machine)** is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an ****interpreter****.

- Difference between JDK, JRE, JVM - TutorialAndExample

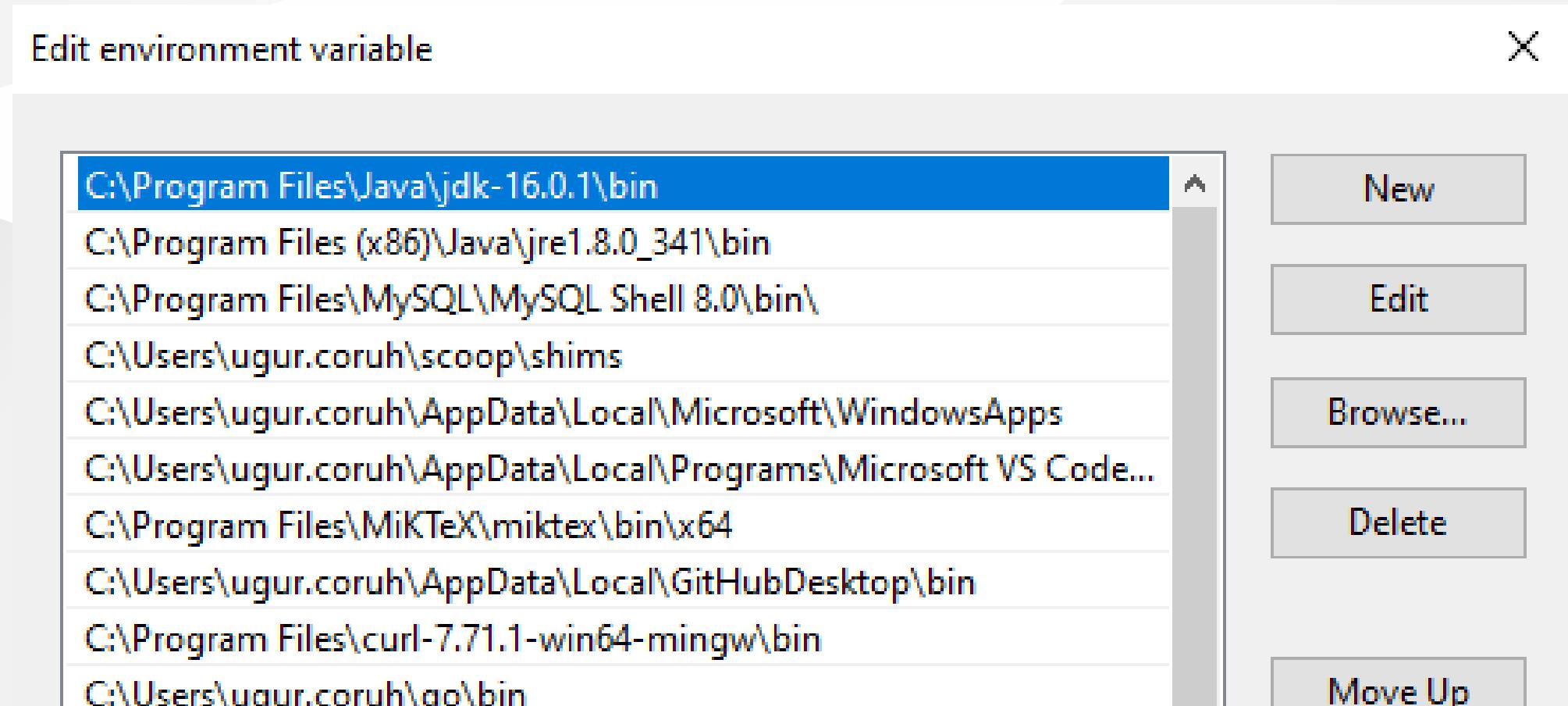


System Environments and Paths for Java (1)



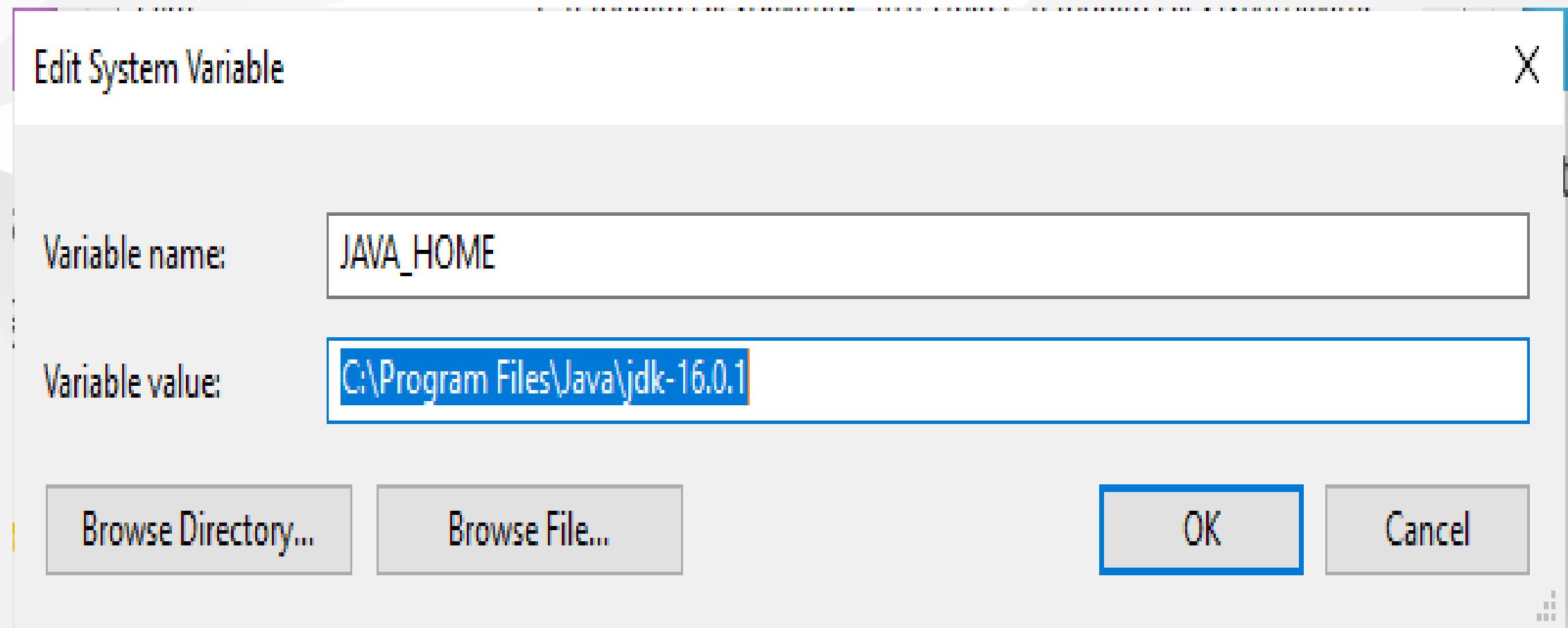
System Environments and Paths for Java (2)

- Select path variable (JDK should be set there)



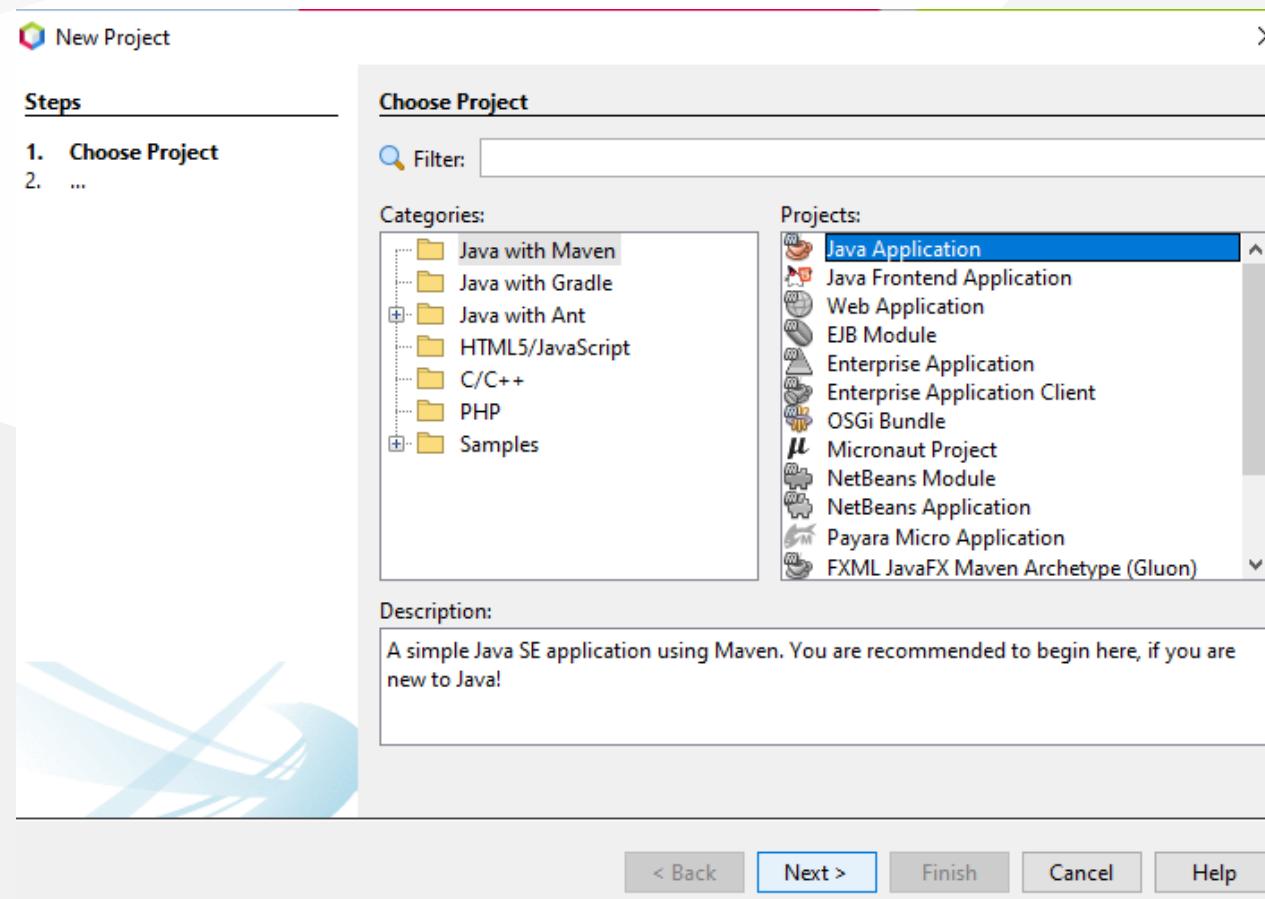
System Environments and Paths for Java (3)

- JAVA_HOME also should be set

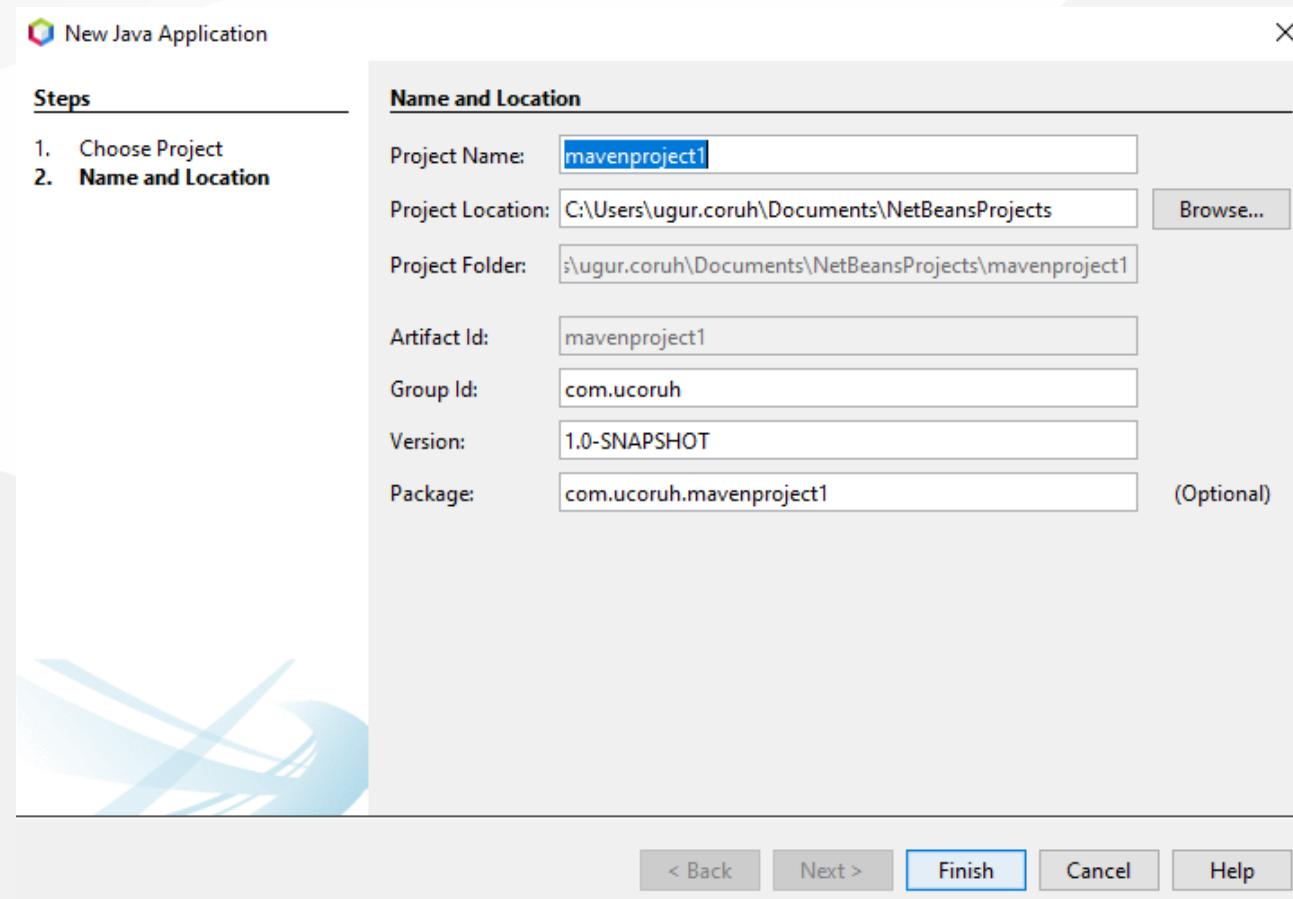


Netbeans (Java) (1)

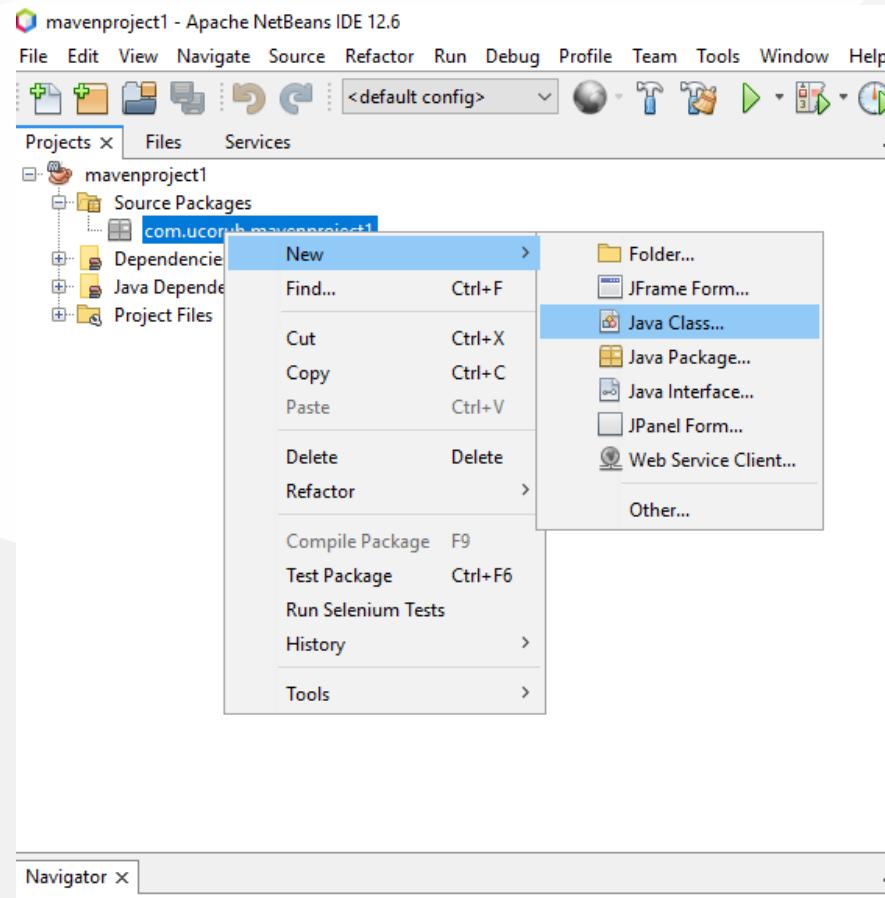
- Open New Project -> Java Project



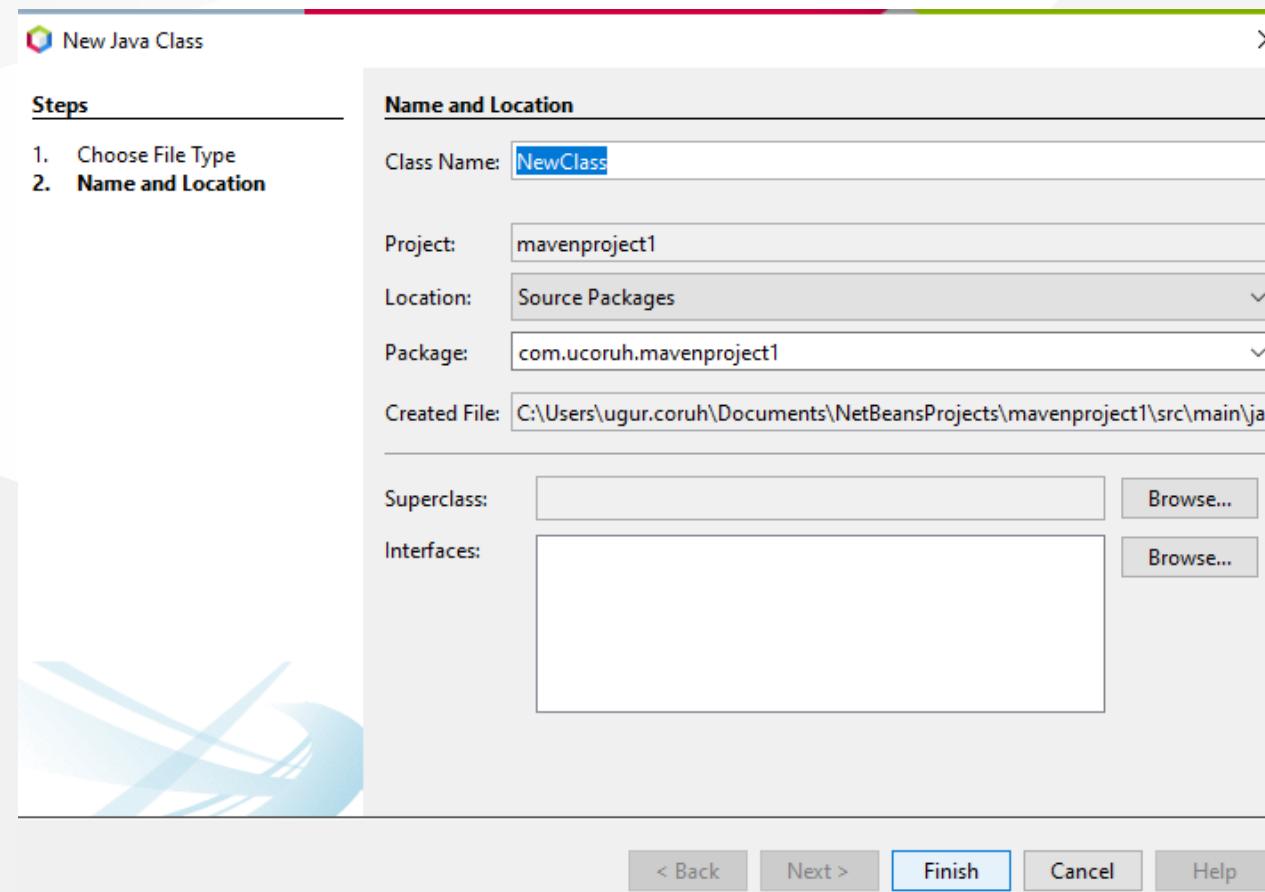
Netbeans (Java) (2)



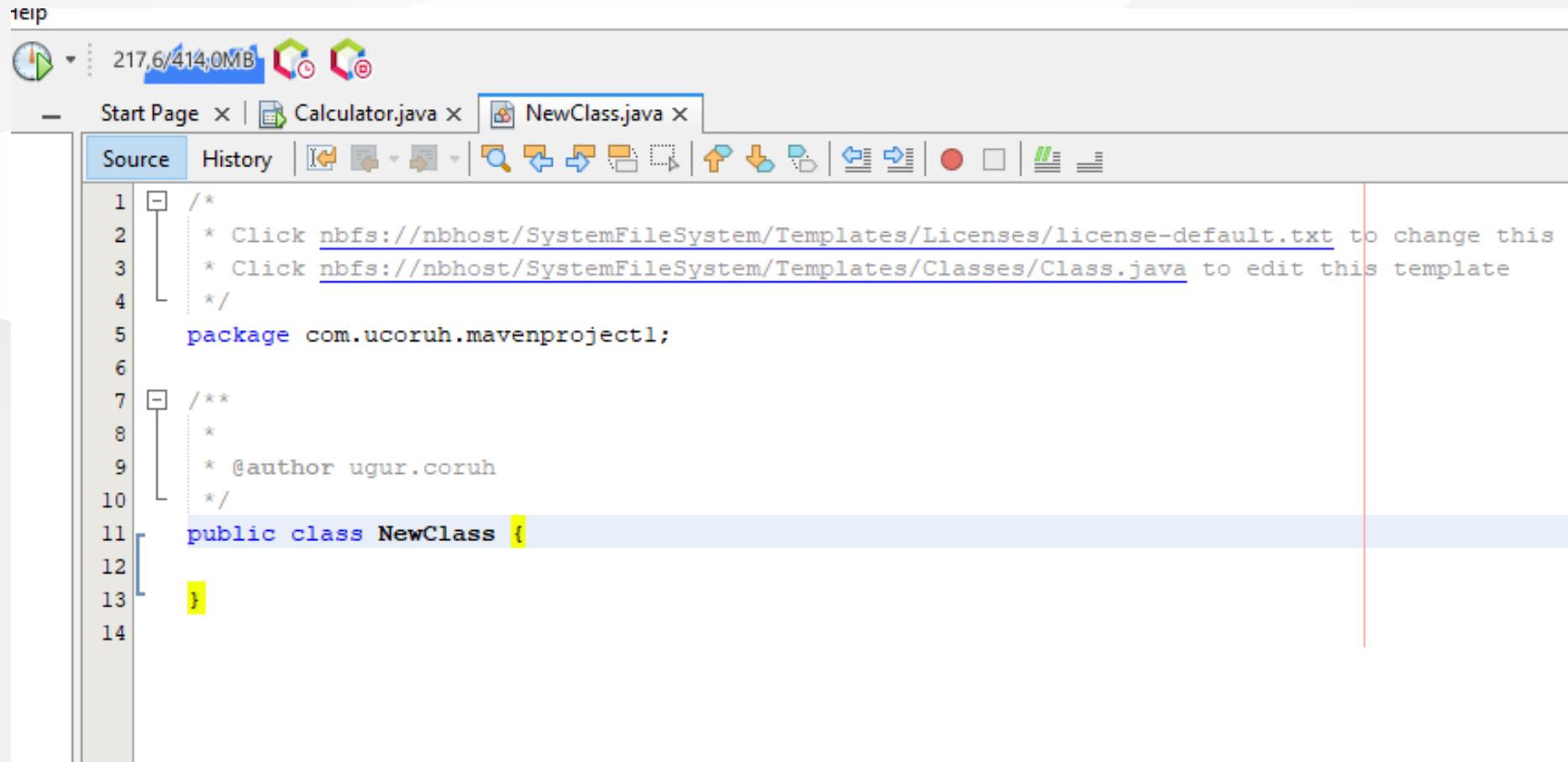
Netbeans (Java) (3)



Netbeans (Java) (4)



Netbeans (Java) (5)

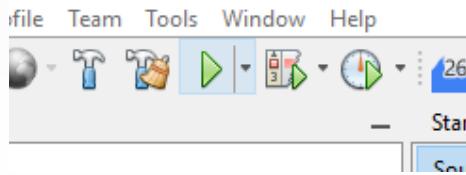


The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Help, 217,6/414,0MB, three circular icons.
- Toolbar:** Standard NetBeans toolbar with various icons for file operations.
- Project Bar:** Start Page, Calculator.java (closed), NewClass.java (selected).
- Source Tab:** Active tab, showing the Java code for NewClass.java.
- Code Editor:** Displays the following Java code:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package com.ucoruh.mavenproject1;
6
7  /**
8   *
9   * @author ugur.coruh
10  */
11 public class NewClass {
12
13 }
14
```

Update code and run



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.ucoruh.mavenproject1;

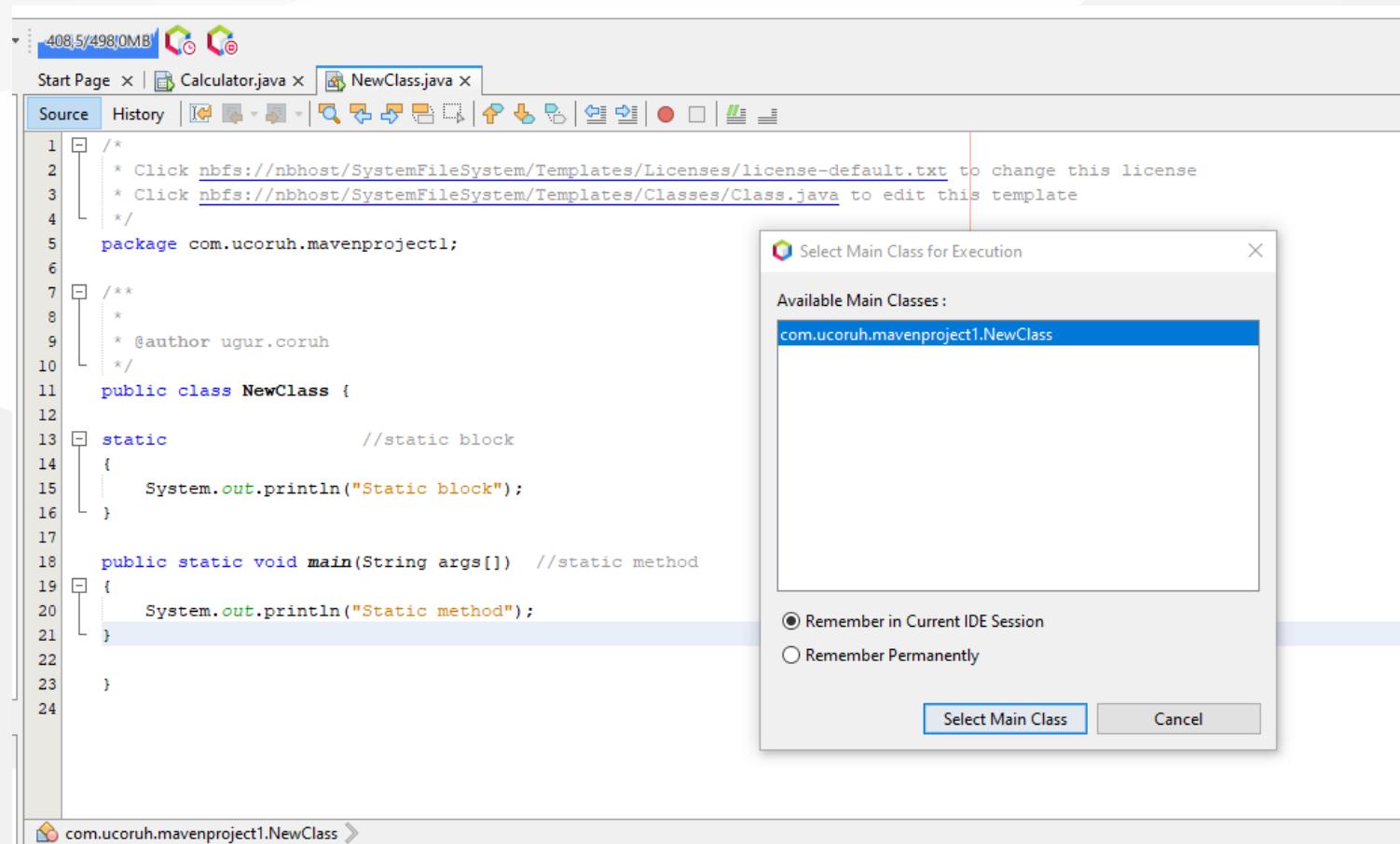
/**
 *
 * @author ugur.coruh
 */
public class NewClass {

    static //static block
    {
        System.out.println("Static block");
    }

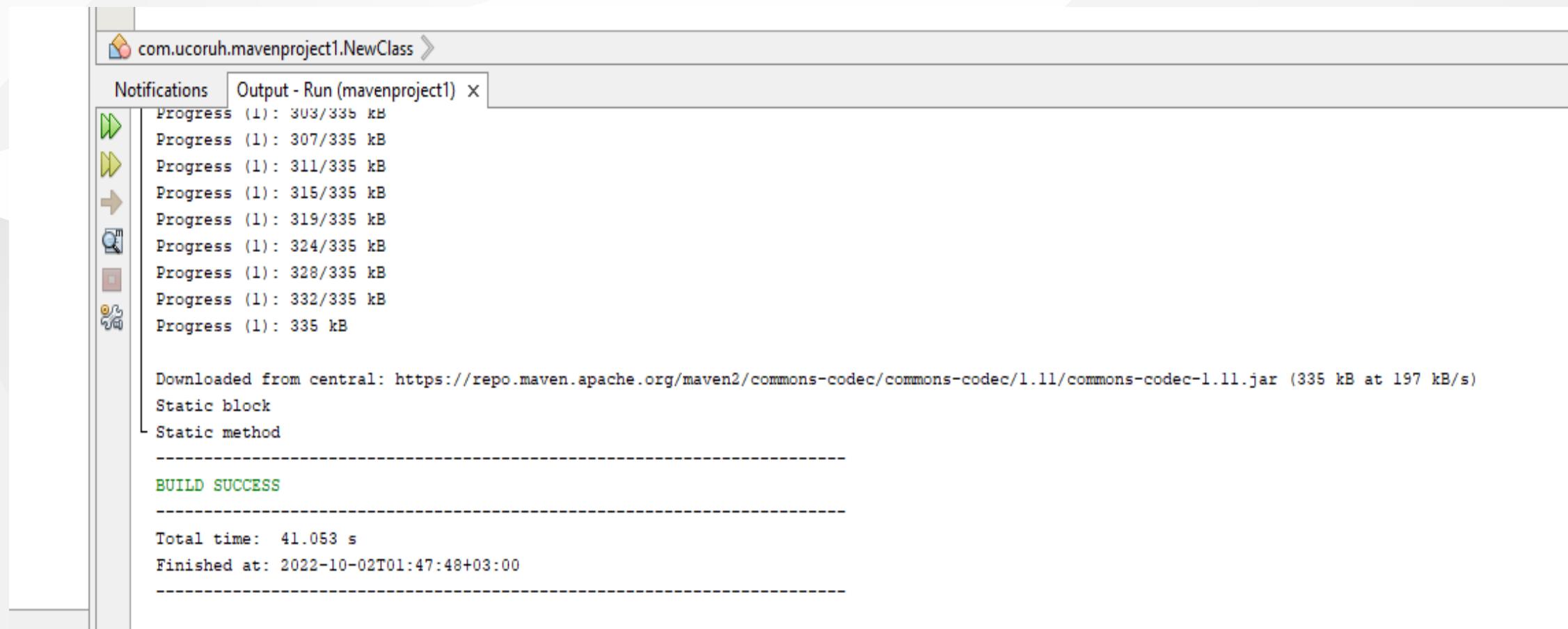
    public static void main(String args[]) //static method
    {
        System.out.println("Static method");
    }
}
```

RTEU CE103 Week-2

Netbeans (Java) (7)



Netbeans (Java) (8)



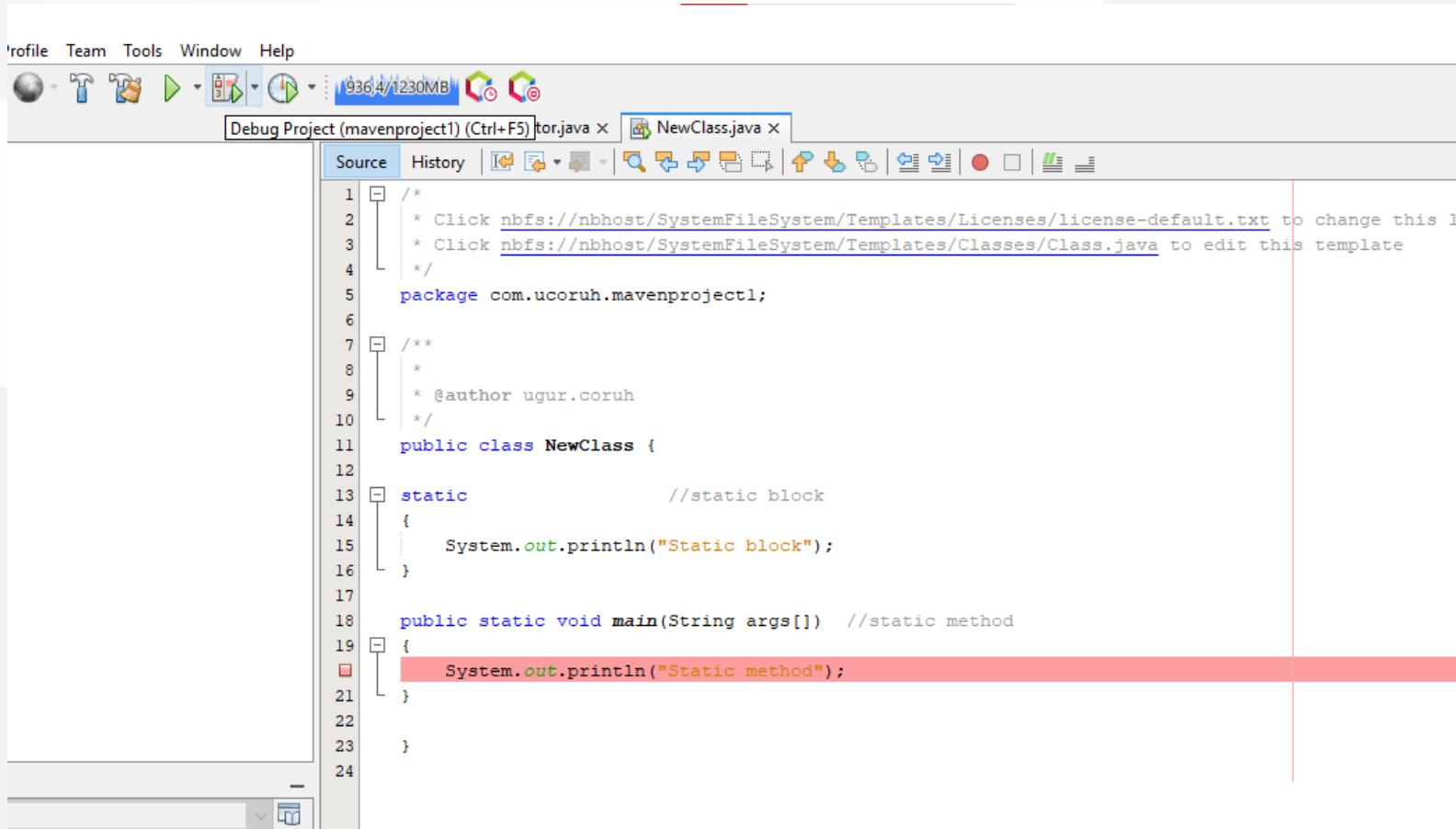
The screenshot shows the Netbeans IDE interface during a Maven build process for a project named "com.ucoruh.mavenproject1". The "Output - Run (mavenproject1)" tab is selected, displaying the following log output:

```
Notifications Output - Run (mavenproject1) x
Progress (1): 303/335 kB
Progress (1): 307/335 kB
Progress (1): 311/335 kB
Progress (1): 315/335 kB
Progress (1): 319/335 kB
Progress (1): 324/335 kB
Progress (1): 328/335 kB
Progress (1): 332/335 kB
Progress (1): 335 kB

Downloaded from central: https://repo.maven.apache.org/maven2/commons-codec/commons-codec/1.11/commons-codec-1.11.jar (335 kB at 197 kB/s)
Static block
Static method

-----
BUILD SUCCESS
-----
Total time: 41.053 s
Finished at: 2022-10-02T01:47:48+03:00
-----
```

Netbeans (Java) (9)



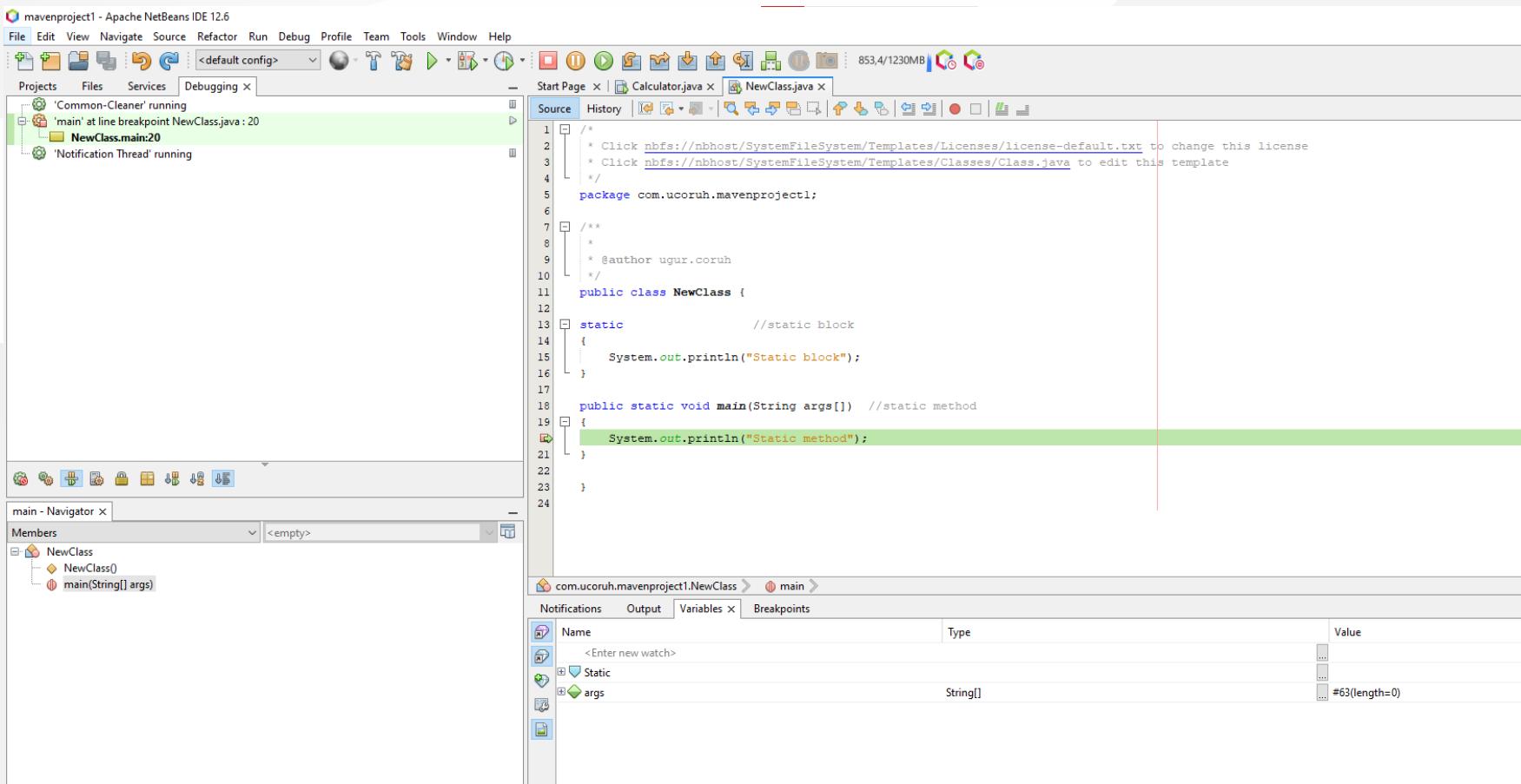
The screenshot shows the NetBeans IDE interface with the following details:

- Menu Bar:** Profile, Team, Tools, Window, Help.
- Toolbar:** Includes icons for file operations (New, Open, Save, Print), project management (Project Explorer, Navigator, Properties, Tools, Help), and search (Search, Find, Replace).
- Project Bar:** Debug Project (mavenproject1) (Ctrl+F5), tor.java x, NewClass.java x.
- Source Tab:** The current tab, showing the Java code for `NewClass.java`.
- Code Editor:** Displays the following Java code:

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license.
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package com.ucoruh.mavenproject1;
6
7  /**
8  * @author ugur.coruh
9  */
10 public class NewClass {
11
12     static //static block
13     {
14         System.out.println("Static block");
15     }
16
17     public static void main(String args[]) //static method
18     {
19         System.out.println("Static method");
20     }
21
22 }
23
24
```

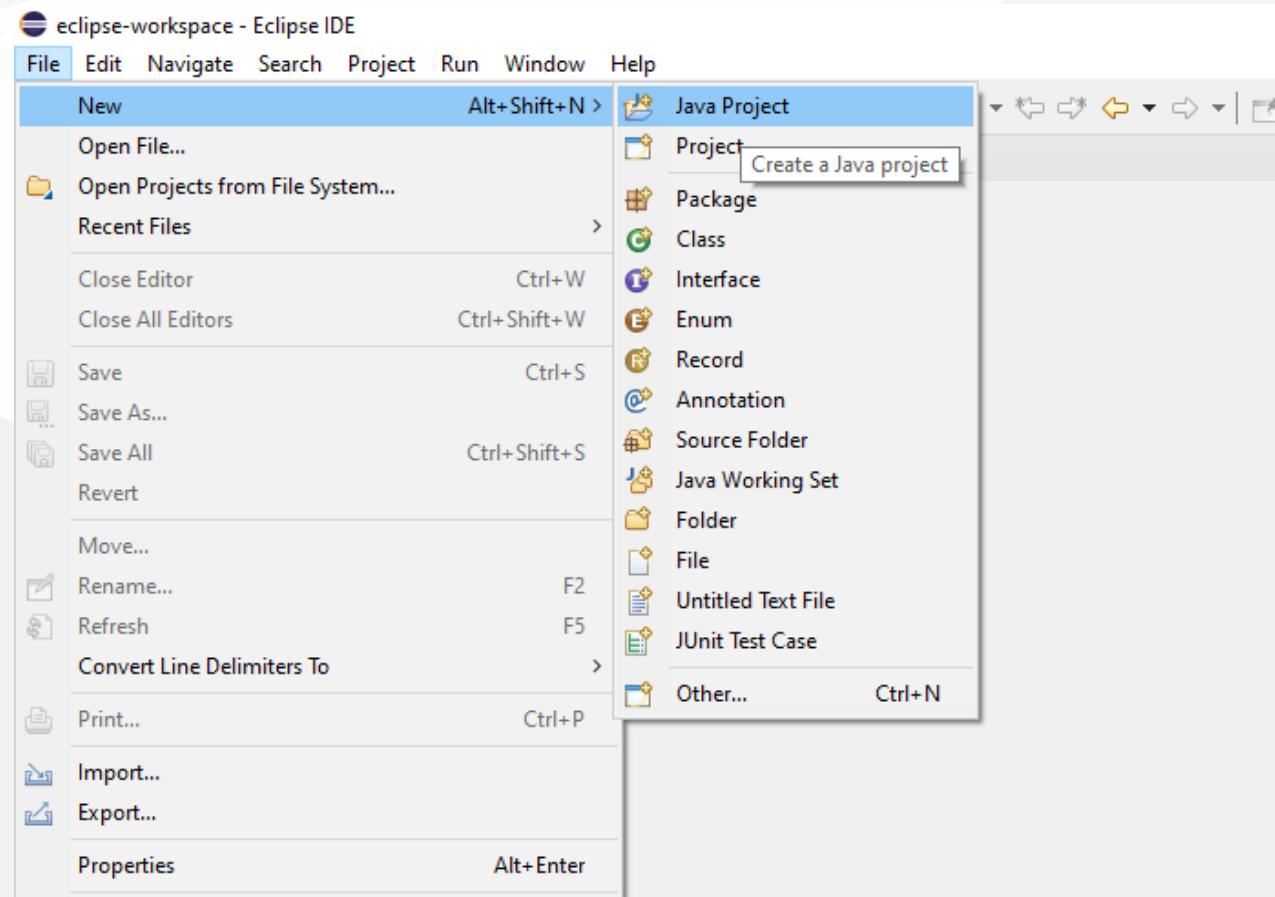
A red vertical bar highlights the static block and the main method, indicating they are currently selected or being edited.

Netbeans (Java) (10)

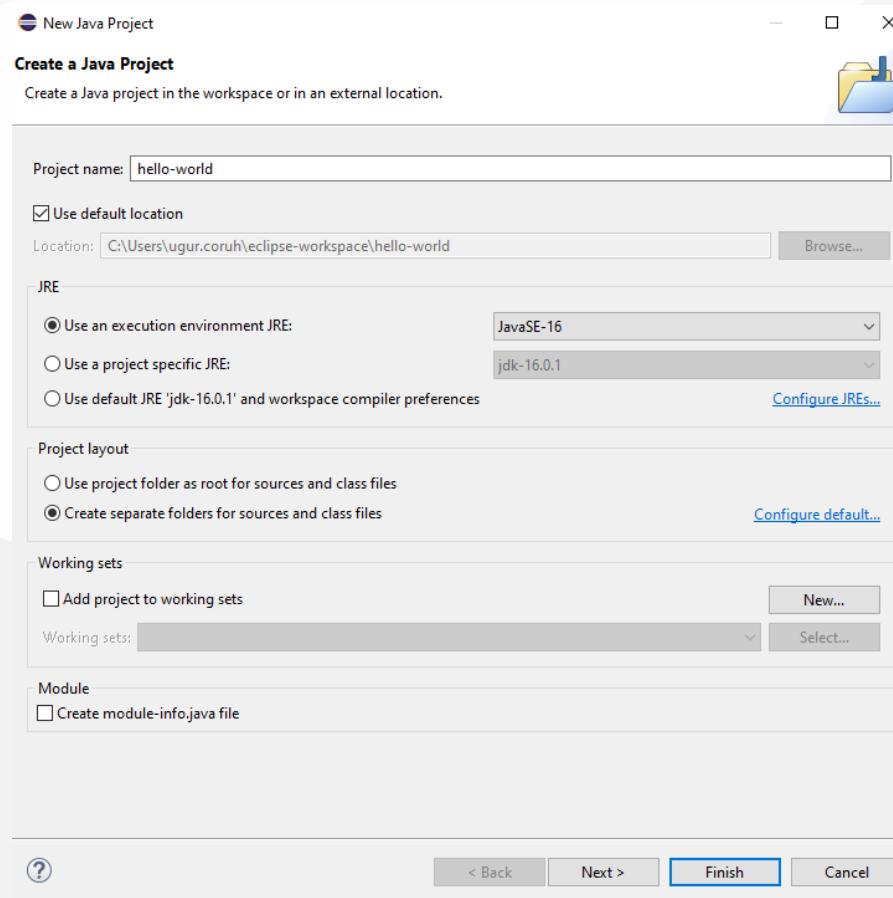


Eclipse (Java) (1)

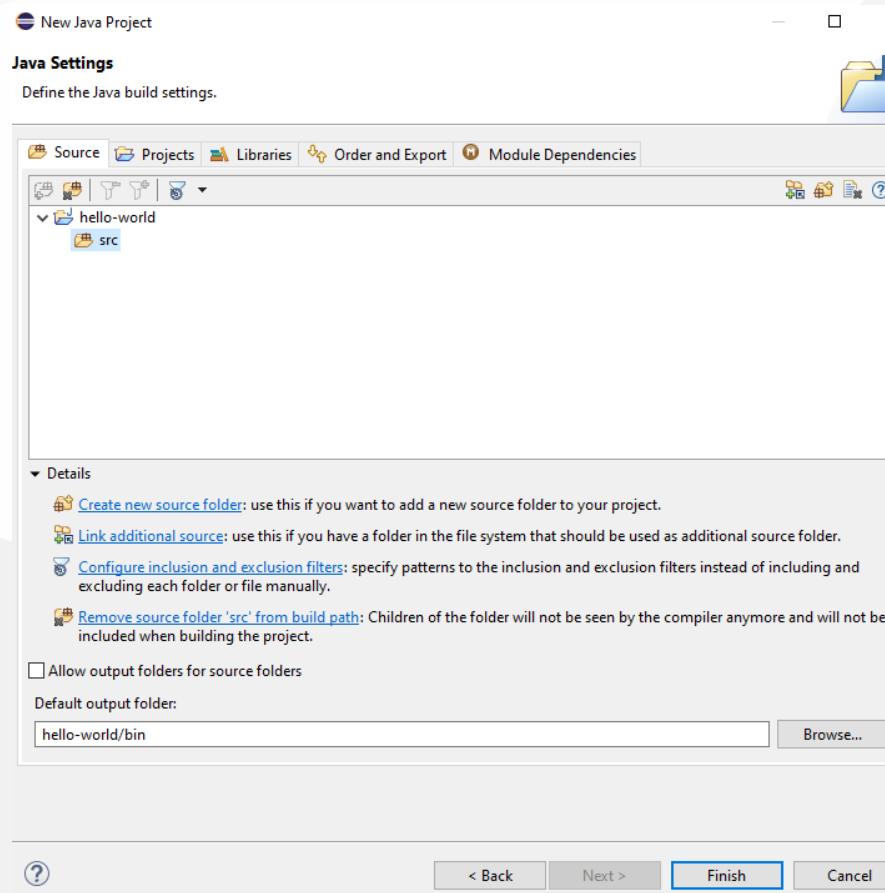
- Select File -> New Project



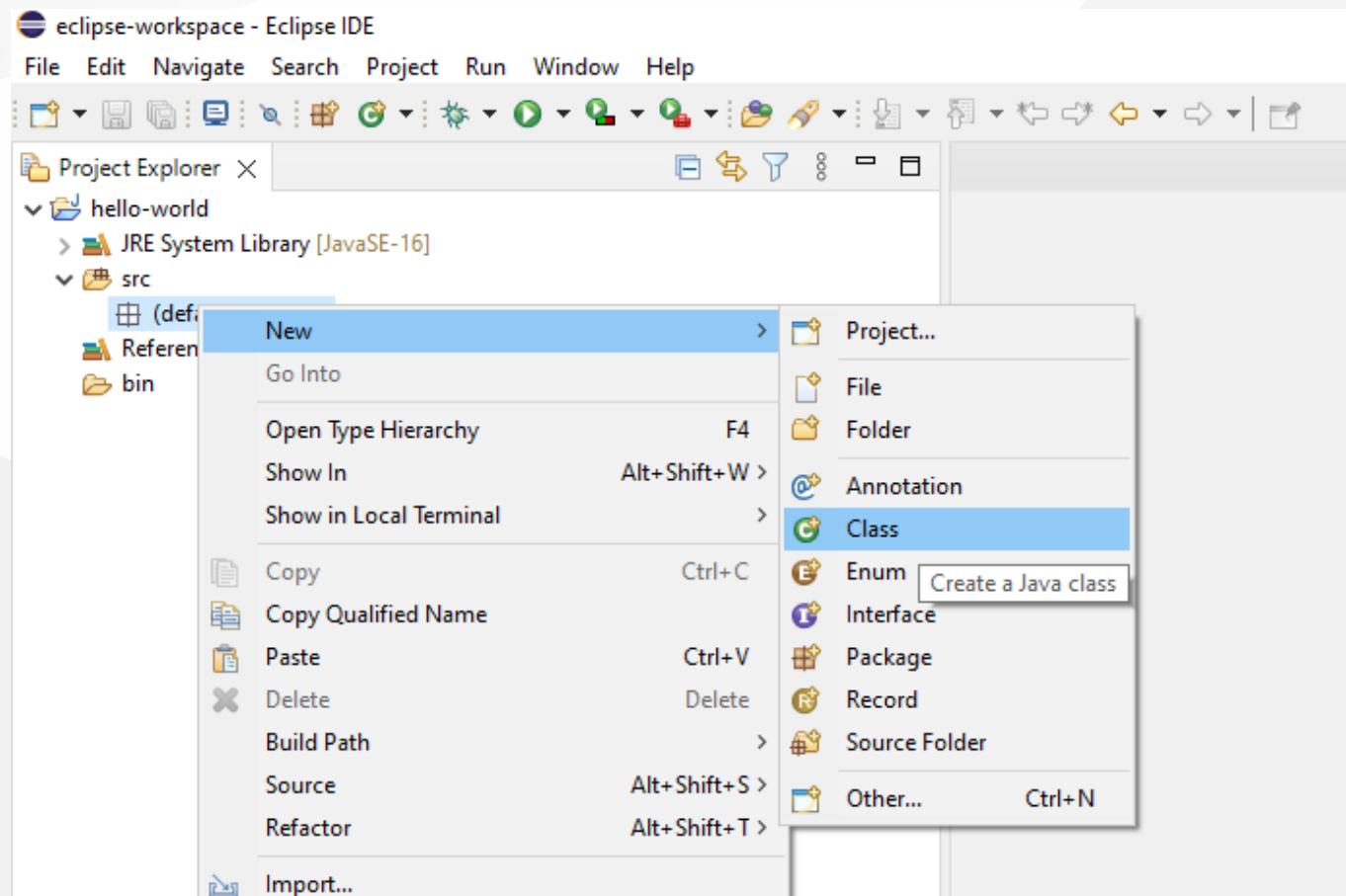
Eclipse (Java) (2)



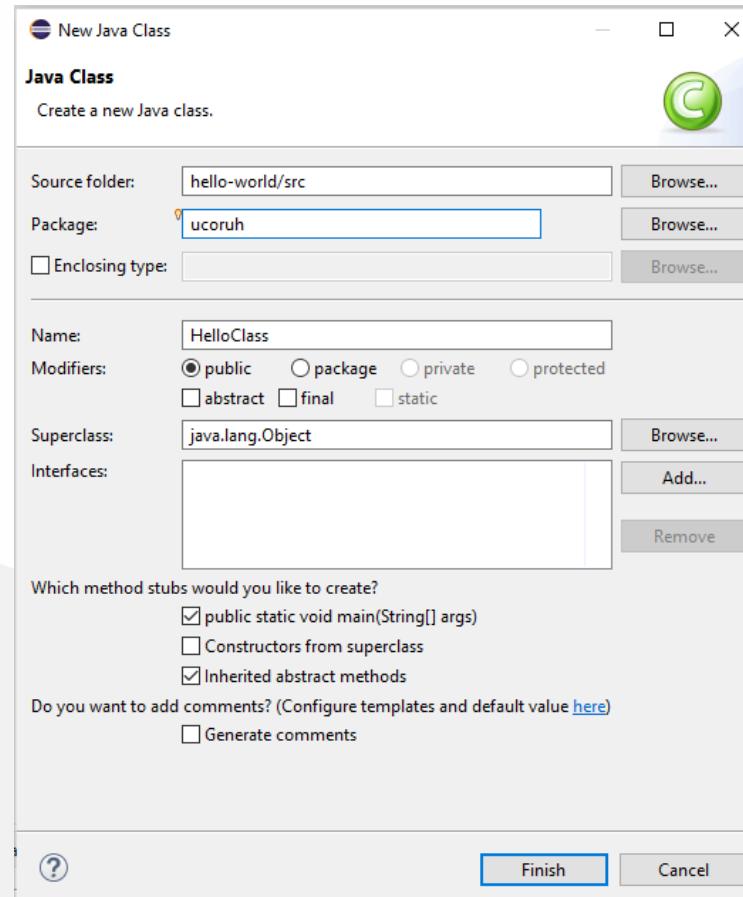
Eclipse (Java) (3)



Eclipse (Java) (4)



Eclipse (Java) (5)



Eclipse (Java) (6)

- Update source

```
package ucoruh;

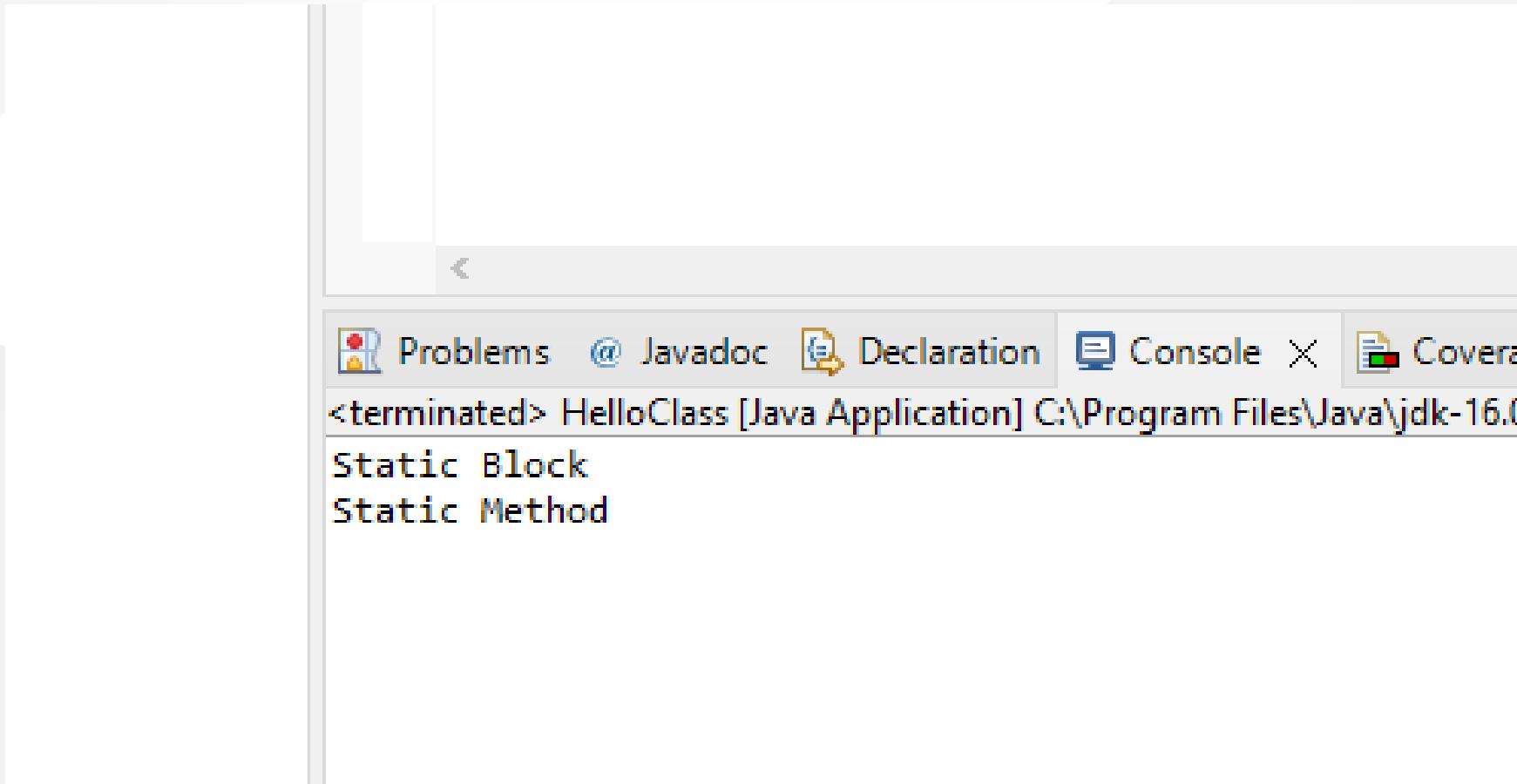
public class HelloClass {

    static {
        System.out.println("Static Block");
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Static Method");

    }
}
```

Eclipse (Java) (7)



Eclipse (Java) (8)

eclipse-workspace - hello-world/src/ucoruh/HelloClass.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug X Project Explorer

HelloClass [Java Application]

ucoruh.HelloClass at localhost:51681

Thread [main] (Suspended (breakpoint at line 6 in HelloClass))

HelloClass.<clinit>() line: 6

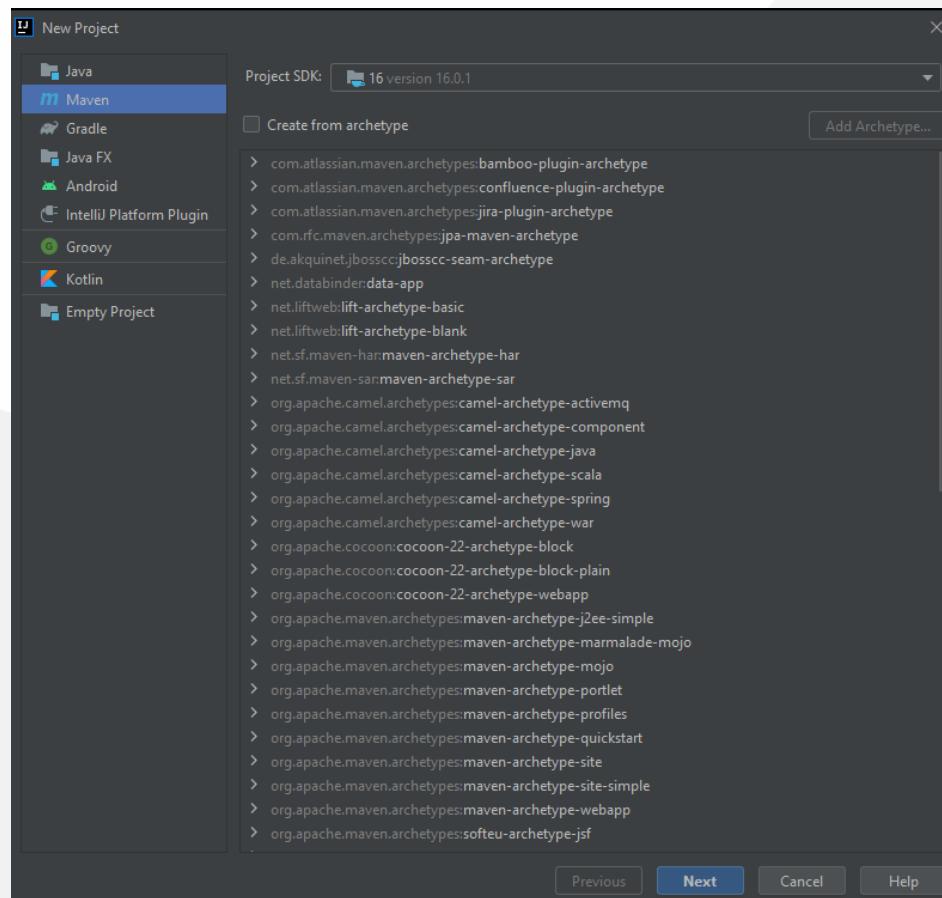
C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (2 Eki 2022 02:27:51) [pid: 15840]

HelloClass.java

```
1 package ucoruh;
2
3 public class HelloClass {
4
5     static {
6         System.out.println("Static Block");
7     }
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         System.out.println("Static Method");
12     }
13 }
14
15 }
16 }
```

IntelliJ Idea (Jet Brains) (Java)

- Download IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains
 - Select Community Version or Student Ultimate Version



VSCode (Java)

- Java Extension Run&Debug Java Files

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** HelloClass.java - HelloCode - Visual Studio Code.
- Sidebar:** RUN AND DEBUG (selected), RUN, PERIPHERALS (1 notification).
- Editor:** HelloClass.java (ucoruh package) with code:

```
1 package ucoruh;
2
3 public class HelloClass {
4
5     static {
6         System.out.println("Static Block");
7     }
8
9     public static void main(String[] args) {
10        // TODO Auto-generated method stub
11        System.out.println("Static Method");
12    }
13
14 }
```

A yellow dot at line 14 indicates a breakpoint.
- Bottom Bar:** PROBLEMS (2), OUTPUT, DEBUG CONSOLE (selected), TERMINAL, JUPYTER, ...; DEBUG: HelloClass.
- Terminal:** Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS C:\Users\ugur.coruh\Desktop\HelloCode> & 'C:\Program Files\Java\jdk-16.0.1\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:60025' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ugur.coruh\AppData\Roaming\Code\User\workspaceStorage\8bba6f3001e6b56b4be5234583be8ec0\redhat.java\jdt_ws\jdt.ls\java-project\bin' 'ucoruh.HelloClass'
Static Block
Static Method
PS C:\Users\ugur.coruh\Desktop\HelloCode>
- Status Bar:** Ln 15, Col 2 Tab Size: 4 CRLF ⚙ Java ⚙ Go Live ⚙ Prettier ⚙

Notepad++ (Java)

- How to Compile and Run Java Programs Using Notepad++

Cmake (Java)

- [UseJava — CMake 3.24.2 Documentation](#)
- [GitHub - ptitpoulpe/cmake-swig-java-example: An example of combining cmake, swig and java](#)

C# Environment and Development

Visual Studio Community Edition (C#)

//TODO//



Notepad++ (C#)

- This use command-line utilities for csharp, nppexec should be configured for this utility.
- [Compiling/Executing a C# Source File in Command Prompt - Stack Overflow](#)

```
c:\windows\Microsoft.NET\Framework\v3.5\
```

```
c:\windows\Microsoft.NET\Framework\v3.5\bin\csc.exe  
/t:exe /out:MyApplication.exe MyApplication.cs ...
```

Cmake (C#)

- GitHub - crud89/DotNetWithCMake: Your swiss army knife for creating .NET assemblies with CMake and integrating unmanaged code.

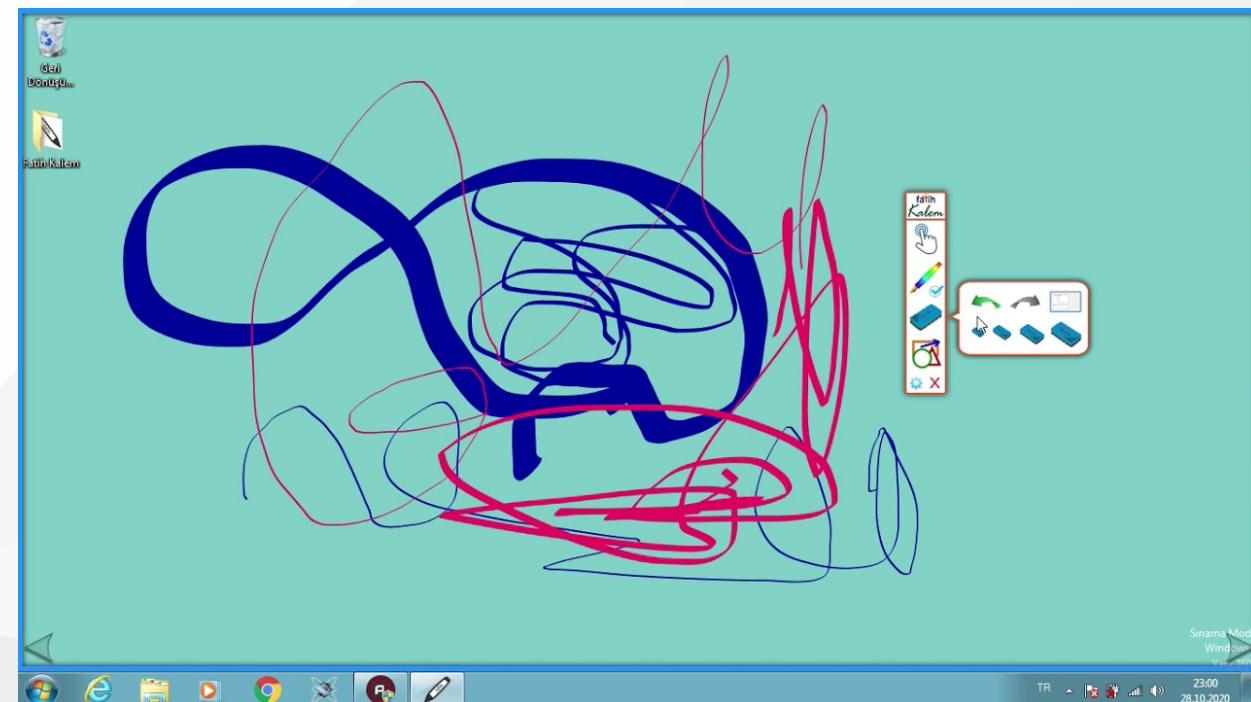
Common Tools and Platforms

Fatih Kalem



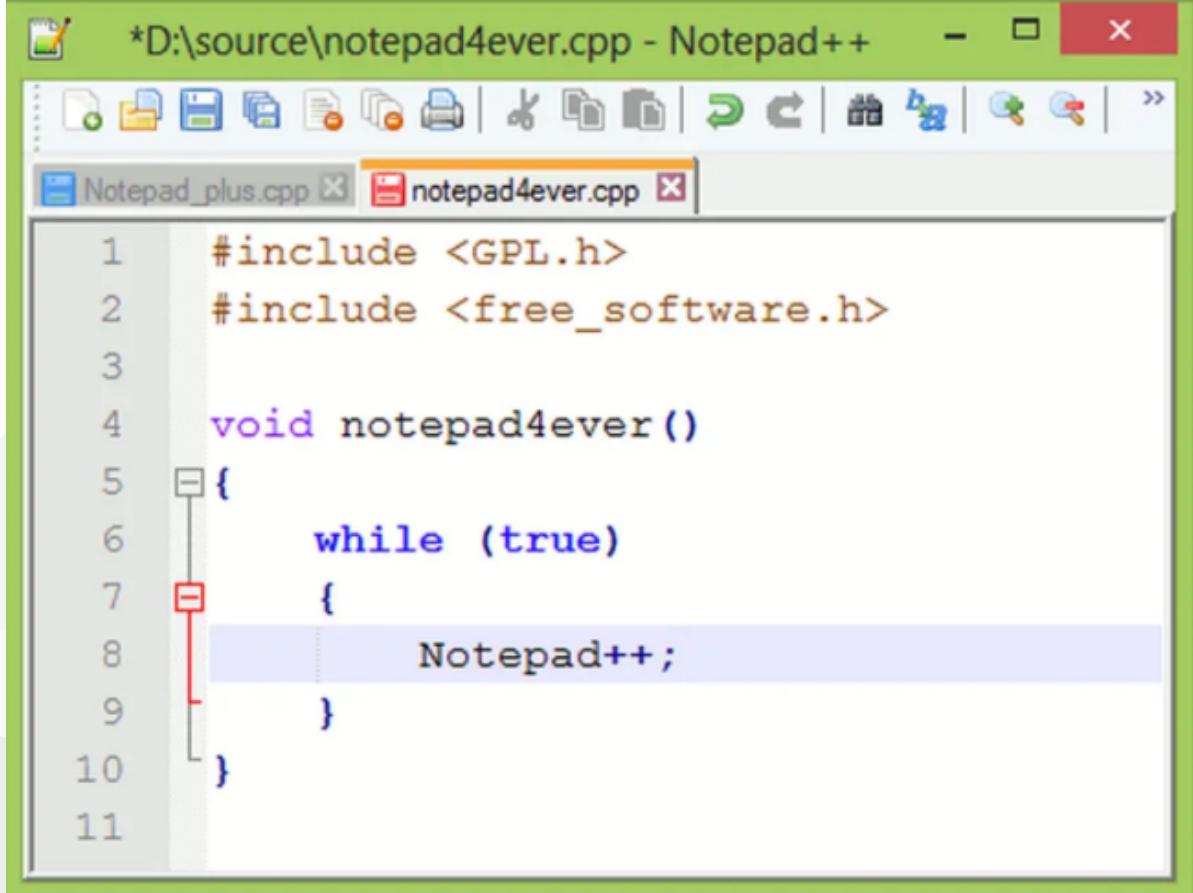
https://cdnvideo.eba.gov.tr/fatihkalem/fatihkalem_portable.zip

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



Notepad++ (Notepad for Source Code)

[Downloads | Notepad++](#)



The screenshot shows the Notepad++ application window. The title bar reads "*D:\source\notepad4ever.cpp - Notepad++". The toolbar contains various icons for file operations like Open, Save, Print, and Find. Below the toolbar is a tab bar with two tabs: "Notepad_plus.cpp" and "notepad4ever.cpp", where "notepad4ever.cpp" is currently selected. The main editor area displays the following C++ code:

```
1 #include <GPL.h>
2 #include <free_software.h>
3
4 void notepad4ever()
5 {
6     while (true)
7     {
8         Notepad++;
9     }
10 }
```

The code uses color-coded syntax highlighting: orange for preprocessor directives, purple for keywords like `#include` and `void`, blue for the function name `notepad4ever`, and black for the variable `Notepad++`. The line numbers are on the left side of the editor.

HxD (Hex Editor)



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

HxD - [Hard Disk 1]

File Edit Search View Analysis Extras Windows ?

ntdll.dll Explorer.EXE Hard Disk 1

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

000007DB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000007DC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000007DD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000007DE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000007DF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000007E00	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00	ëR.NTFS
000007E10	00	00	00	00	00	F8	00	00	3F	00	FF	00	3F	00	00	00ø.?.ÿ.?
000007E20	00	00	00	00	80	00	80	00	C5	EA	FF	0F	00	00	00	00é.e.ÿ....
000007E30	00	00	0C	00	00	00	00	AC	FE	FF	00	00	00	00	00	00¬þÿ....
000007E40	F6	00	00	00	01	00	00	A1	89	4C	88	9D	4C	88	8A	ö.....;‡L^..L^š	

Sector 6

Checksums

c:\WINDOWS\system32\ntdll.dll

Algorithm Checksum Usage

<input checked="" type="checkbox"/> CRC-32	AC43D78D	Ethernet and PKZIP
<input checked="" type="checkbox"/> SHA-1	379FD511C6C1EAAD77C7344AFA83A38999F13288	
<input checked="" type="checkbox"/> MD-5	95092EFBE367A108ECDD5D6E439754C3	

Expected result: AC43D78D

t: 7E03 Block: 7E03-7E06 Length: 4 Readonly Overwrite

Marktext (Markdown Syntax Editor)



- <https://marktext.app/>
- <https://github.com/marktext/marktext/releases>
- Download latest version
 - <https://github.com/marktext/marktext/releases/tag/v0.17.1>

The screenshot shows the Marktext application interface. On the left is a sidebar titled "Table Of Contents" with sections for "An h1 header", "An h2 header", and "An h3 header". The main content area displays the following text:

An h1 header

Paragraphs are separated by a blank line.
2nd paragraph. *italic*, **bold**, and `monospace`. Itemized lists look like:

- this one
- that one
- the other one

Note that --- not considering the asterisk --- the actual text content starts at 4-columns in.

Block quotes are written like so.

They can span multiple paragraphs,

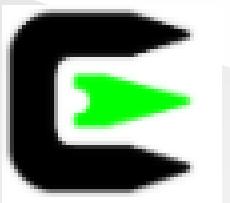
Use 3 dashes for an em-dash. Use 2 dashes for ranges (ex., "it's all in chapters 12–14"). Three dots ... will be converted to an ellipsis. Unicode is supported. ☺

An h2 header

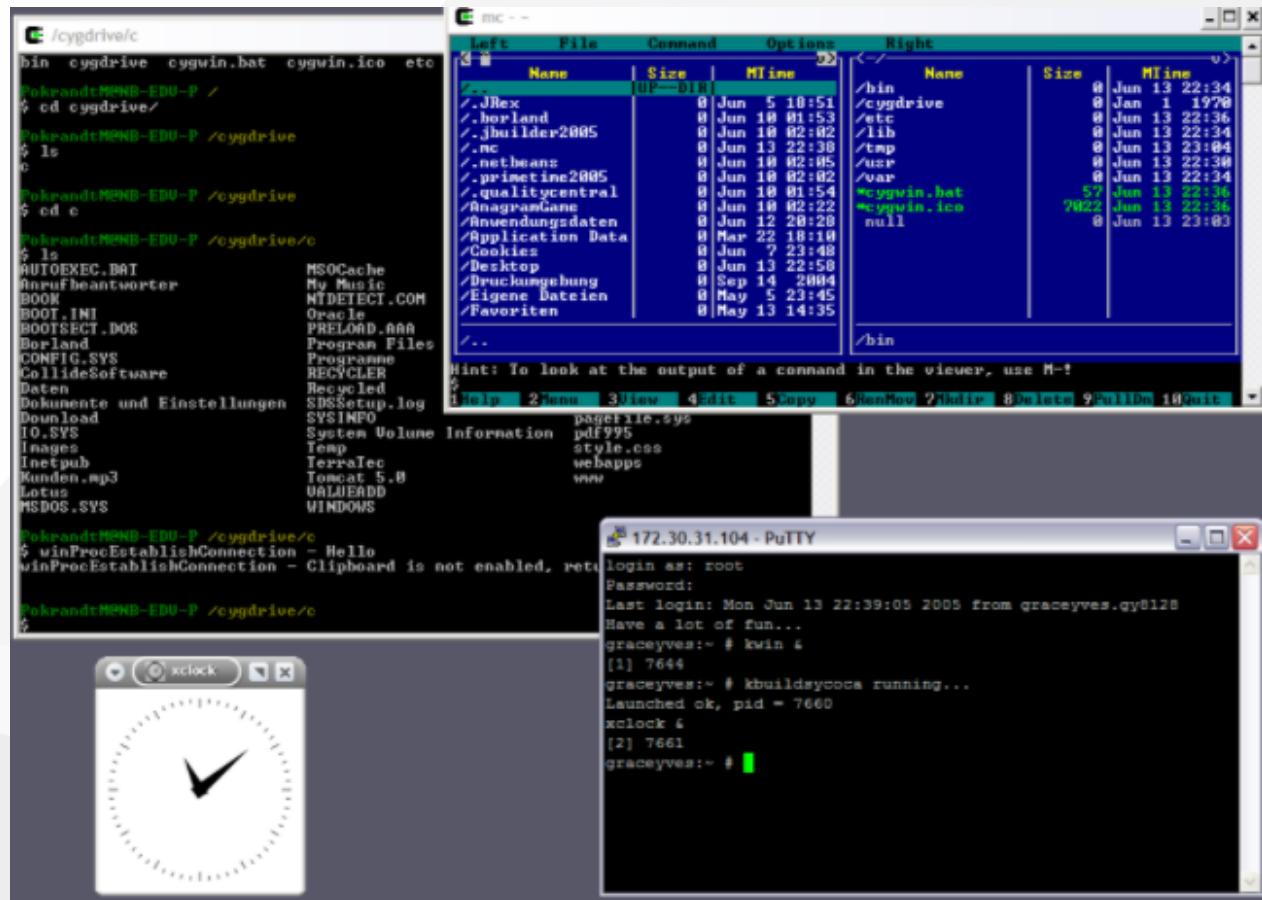
Here's a numbered list:

1. first item
2. second item
3. third item

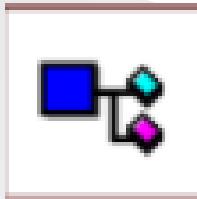
Cygwin (Linux environment for Windows)



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



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



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

The screenshot shows the Dependency Walker application interface. The left pane displays a tree view of the module dependencies for 'STOOGES.EXE'. The right pane contains two tables: one for exports and one for imports. Below these tables is a detailed table of modules with columns for Module, File Time Stamp, Link Time Stamp, File Size, Attr., Link Checksum, Real Checksum, CPU, and Subsystem. At the bottom, a text window shows the command-line output of the dependency walker process, including log messages about loading modules and performing DLL hooking.

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	0x0125FA5	0x0125FA5	x86	Console
SHEMP.DLL	11/14/2006 5:13p	11/14/2006 5:13p	2,560	A	0x00001CE7	0x00001CE7	x86	GUI

For Help, press F1

Doxxygen (Code Documentation)



Doxxygen: Doxygen

ME Project 1.0

The screenshot shows a web-based documentation interface for the 'ME Project 1.0' project. The top navigation bar includes tabs for 'Page' and 'Classes', with 'Classes' being the active tab. A search bar is located in the top right corner. The left sidebar contains links to 'Project', 'uses', 'Class List', 'acme' (which is expanded to show 'ACMESmartphone'), 'App', 'Dynamite', 'Class Index', and 'Class Members'. The main content area is titled 'acme.ACMESmartphone Class Reference'. It includes sections for 'Public Member Functions' (listing `ACMESmartphone`, `findRoadRunner`, and `zapRoadRunner`) and 'Public Attributes' (listing `LongLat`). A 'Detailed Description' section states: 'Works like a regular smartphone but also tracks roadrunners.' The bottom of the page indicates it was generated on 'Sun Sep 27 2015 09:05:27' for 'ACME Project' by 'doxygen'.

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, se

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

Sonarlint (Code Quality and Code Security Extension)



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



- <https://codepen.io/>
- CodePen is a social development environment. At its heart, it allows you to write code in the browser, and see the results of it as you build.
- A useful and liberating online code editor for developers of any skill, and particularly empowering for people learning to code. We focus primarily on front-end languages like HTML, CSS, JavaScript, and preprocessing syntaxes that turn into those things

A screenshot of a web browser displaying a CodePen project titled "SVG Stroke Animation". The interface shows three panels: HTML, CSS (SCSS), and JS. The HTML panel contains the code for an SVG element with a path named "hello". The CSS panel includes a background color rule and a keyframe animation for the ".fin" class. The JS panel contains a function that adds a class to the container element. Below the code editor is a preview area with a red background and a large, white, cursive "hello." text. The browser's address bar shows the URL "codepen.io/gau/pen/ZWZKxO".

```
div id="container">
  <svg id="hello-svg" data-name="hello" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 582 197">
    <title>Hello</title>
    <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-11.8-21,10.33-24.94,27.68-4.52,19.89-60%"/>
  

```

```
body {
  background: #ff837b;
}
#container {
  width: 600px;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
.fin {
  .path-1, .path-2,
  .path-3 {
    stroke-width: 2px;
    stroke: black;
    stroke-dasharray: 10 5;
    stroke-dashoffset: 10px;
    transition: stroke-dashoffset 0.5s ease-in-out;
  }
  &.path-1 {
    stroke-dasharray: 10 5;
    stroke-dashoffset: 10px;
  }
  &.path-2 {
    stroke-dasharray: 10 5;
    stroke-dashoffset: 10px;
  }
  &.path-3 {
    stroke-dasharray: 10 5;
    stroke-dashoffset: 10px;
  }
}
```

```
$function() {
  function animationStart() {
    $('#container').addClass('fin');
  }
  setTimeout(animationStart, 250);
}
```

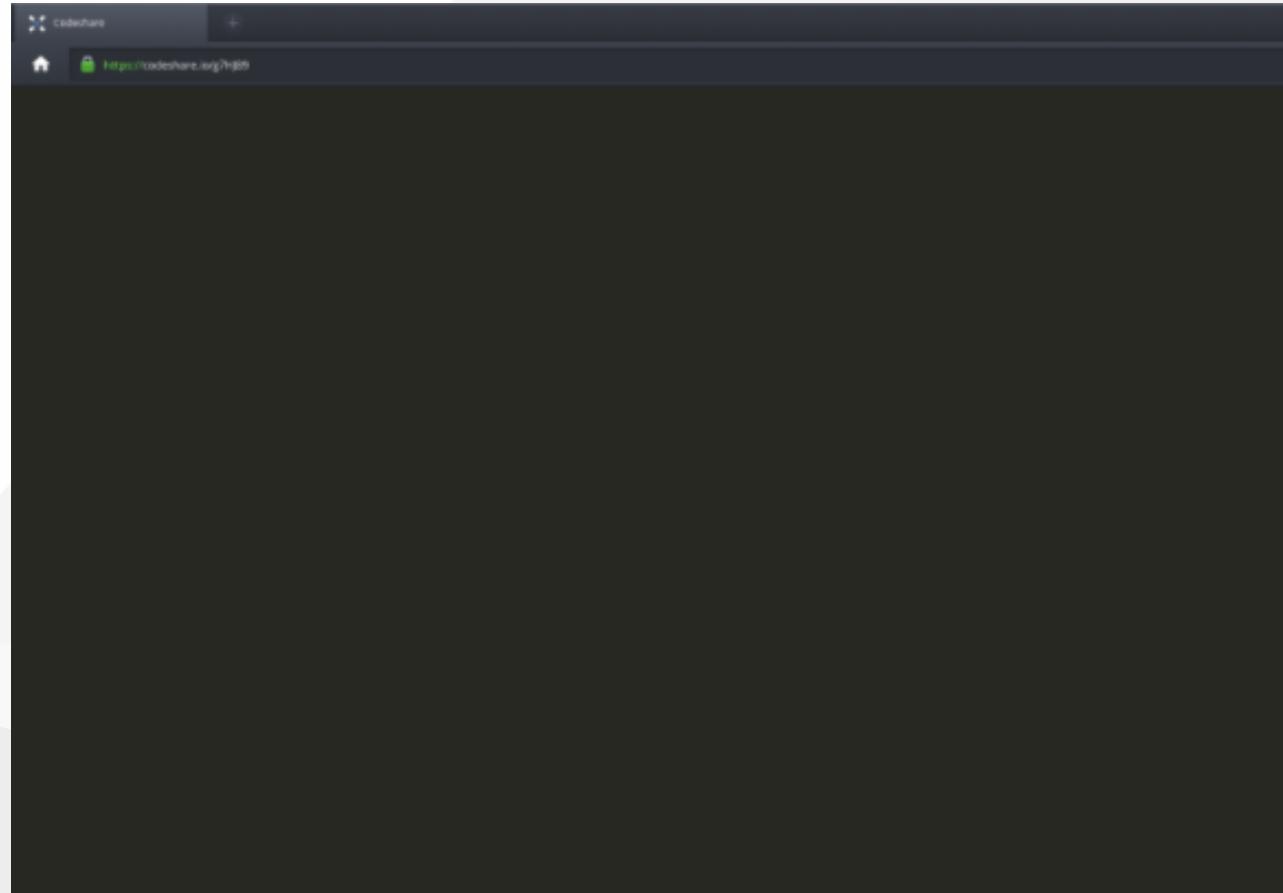
Codepen.io (online code sharing)

- Credit Card Sample on Codepen
 - <https://codepen.io/quinlo/pen/YONMEa>
- Checkout trends <https://codepen.io/trending>

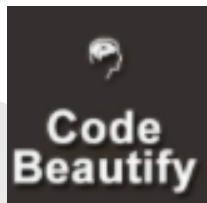
Codeshare.io (real-time code sharing)



- <https://codeshare.io/>
- Share Code in Real-time with Developers, An online code editor for interviews, troubleshooting, teaching & more...



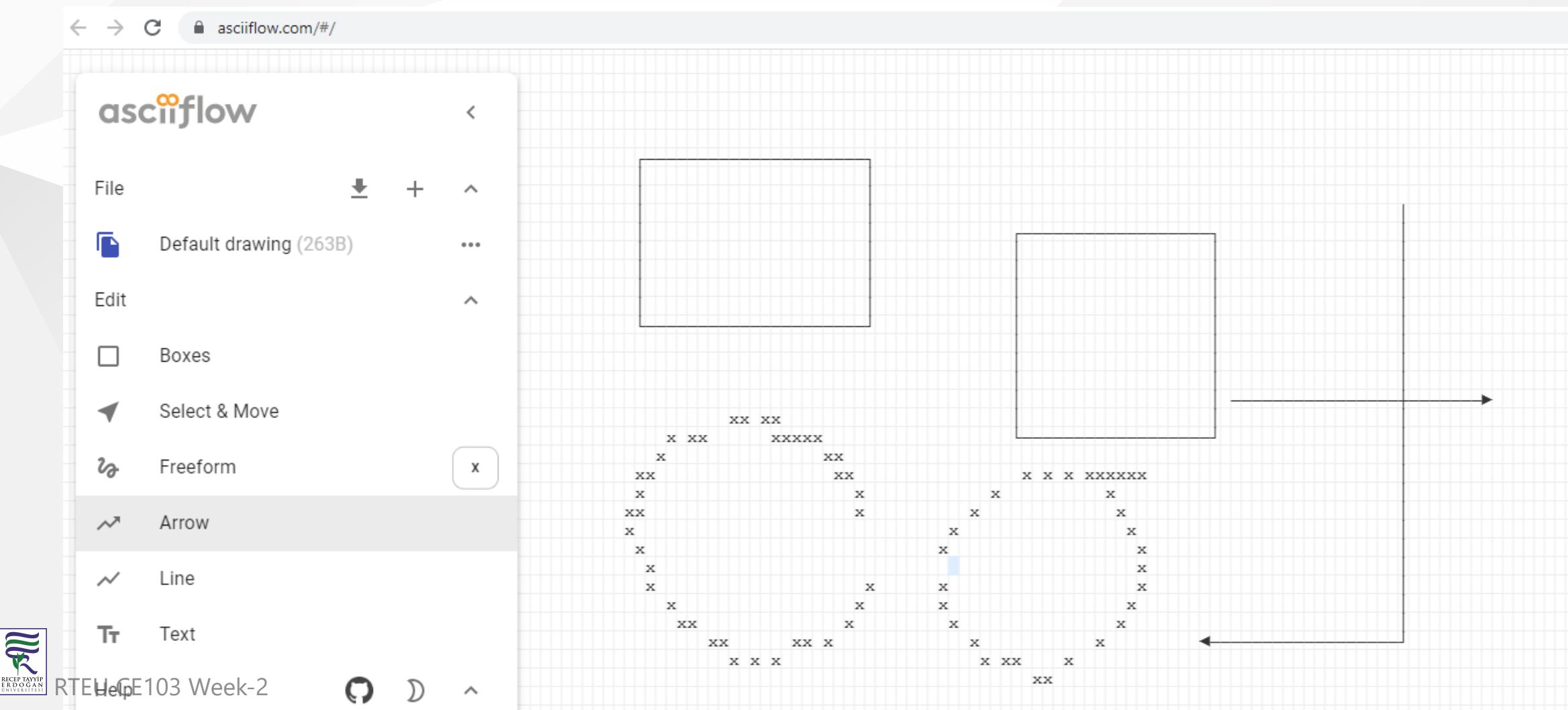
Codebeautify.org (online data conversion tools)



- Has several tools for developers (Code Formatter, JSON Beautifier, XML Viewer, Hex Converters and more...)
- <https://codebeautify.org/>

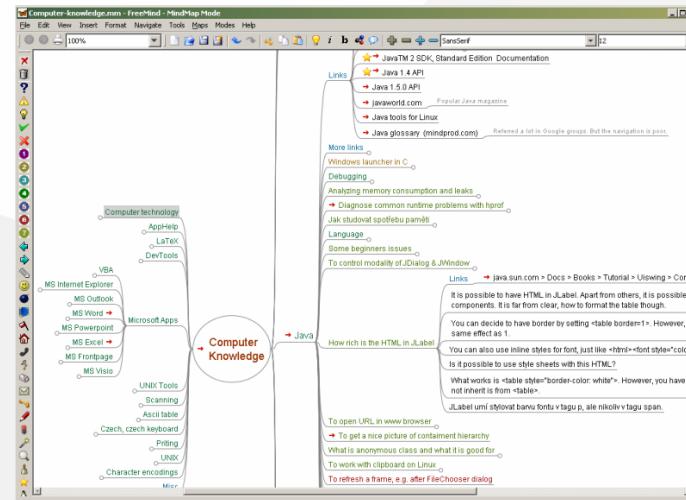
- Asciflow provides ascii based drawings that you can copy directly to textfiles and source codes.
Visit the following link

- <https://asciiflow.com/>



Freemind (opensource mindmap application)

- Freemind is open source java based desktop mindmap application. Can export files to several formats
 - Main Page - FreeMind



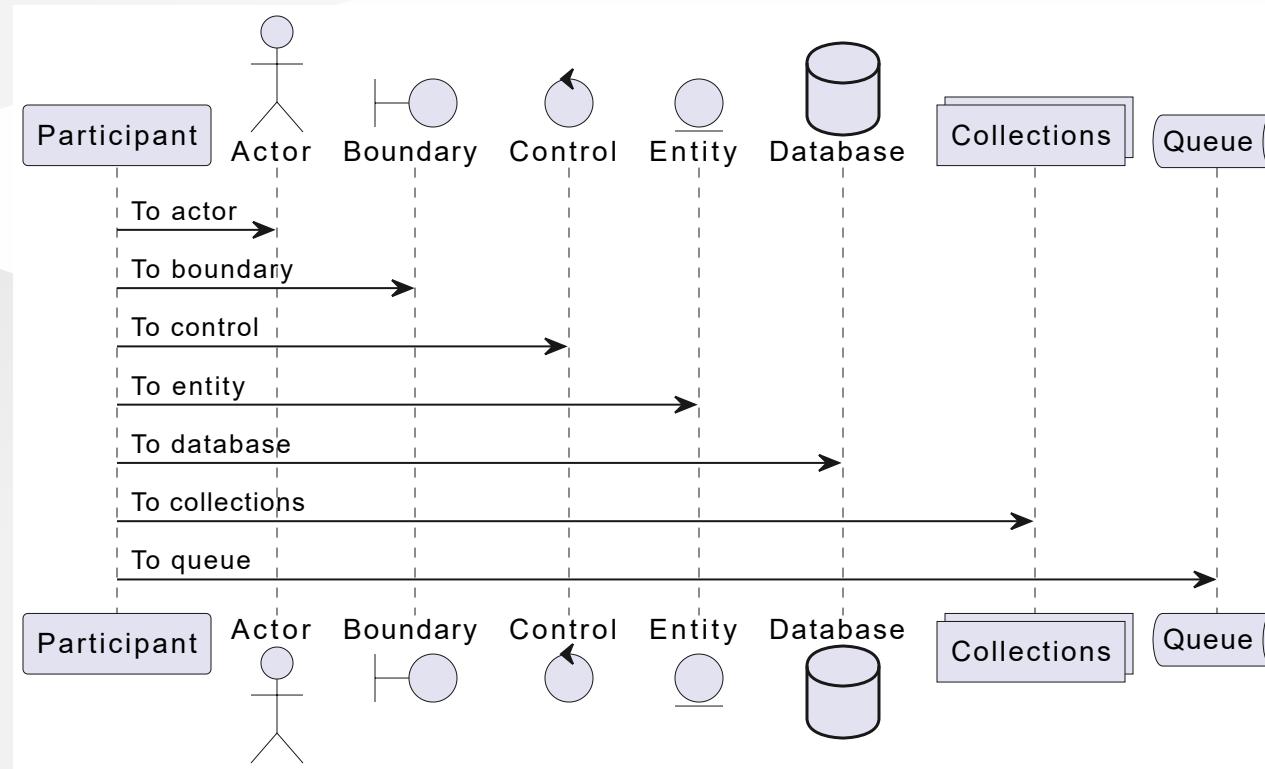
Mockup Designers

- Mockflow
 - [Signup - MockFlow](#)
- Wireflow
 - <https://wireflow.co/>

PlantUML (software designer)

- Text based designer for software engineers

- <https://plantuml.com/>

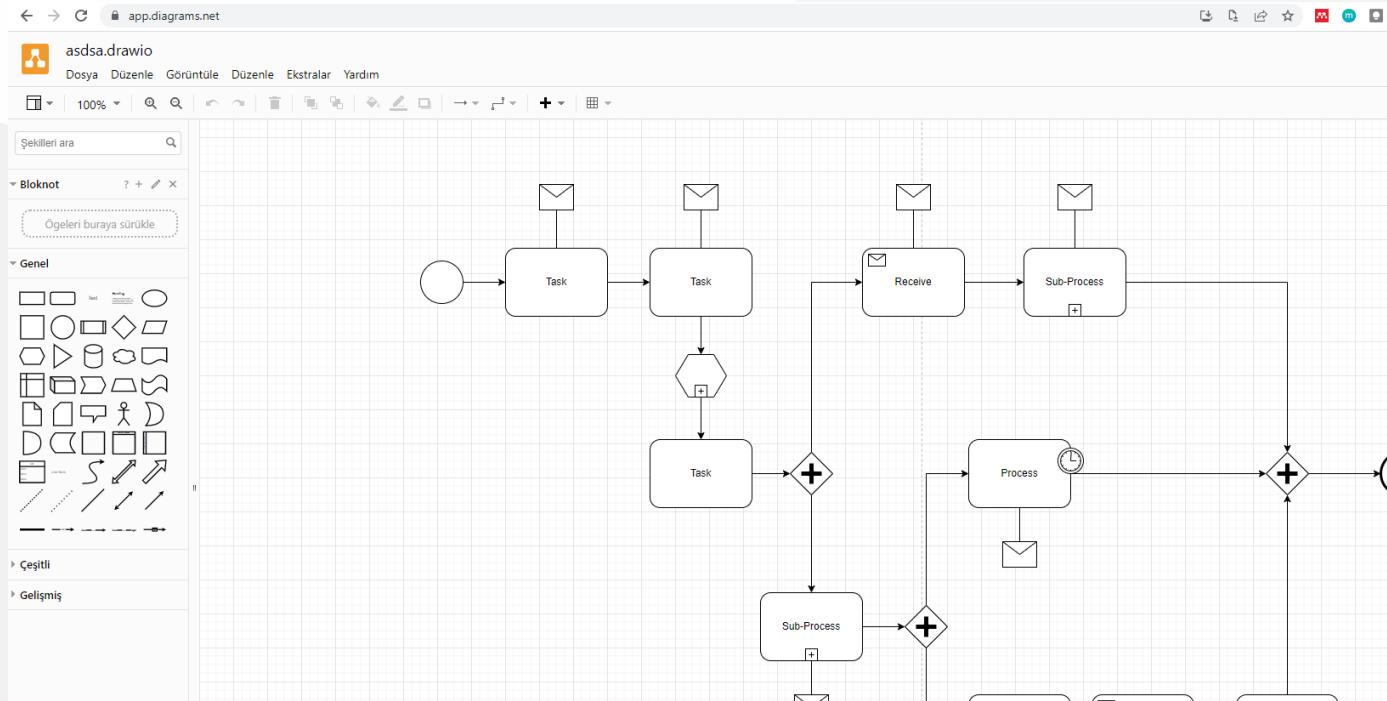


- Also visit course notes that related to plantuml [CE204 Object-Oriented Programming - RTEU](#)

[C204 Object Oriented Programming Course Notes](#)

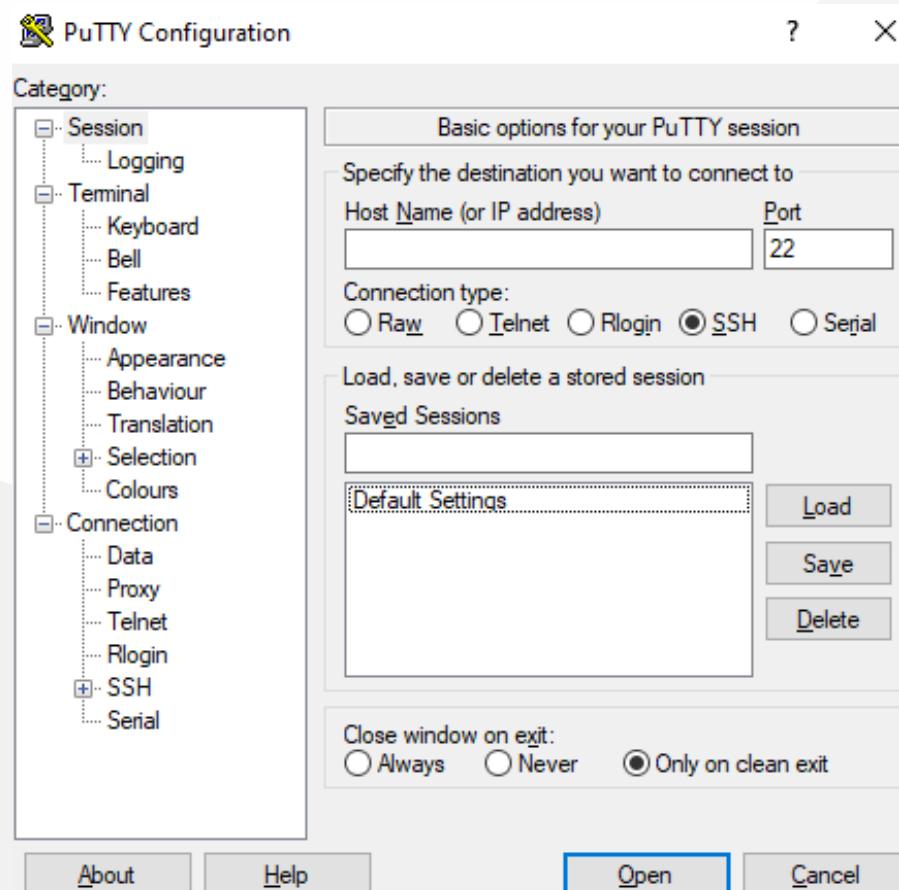
Drawio (drawing tool)

- Online and Offline Drawing Tool
 - <https://app.diagrams.net/>
- Offline Installer
 - [Releases · jgraph/drawio-desktop · GitHub](#)

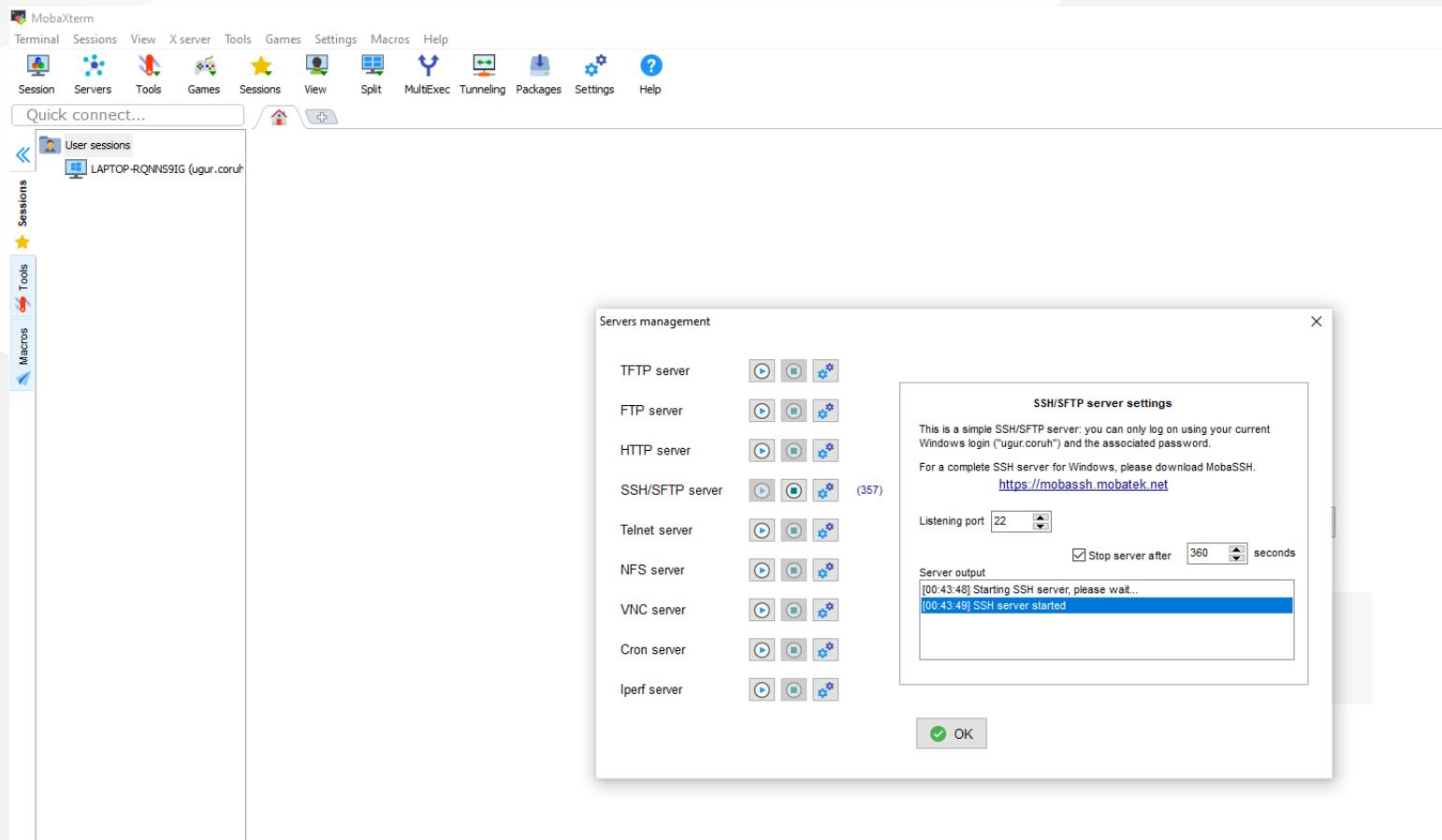


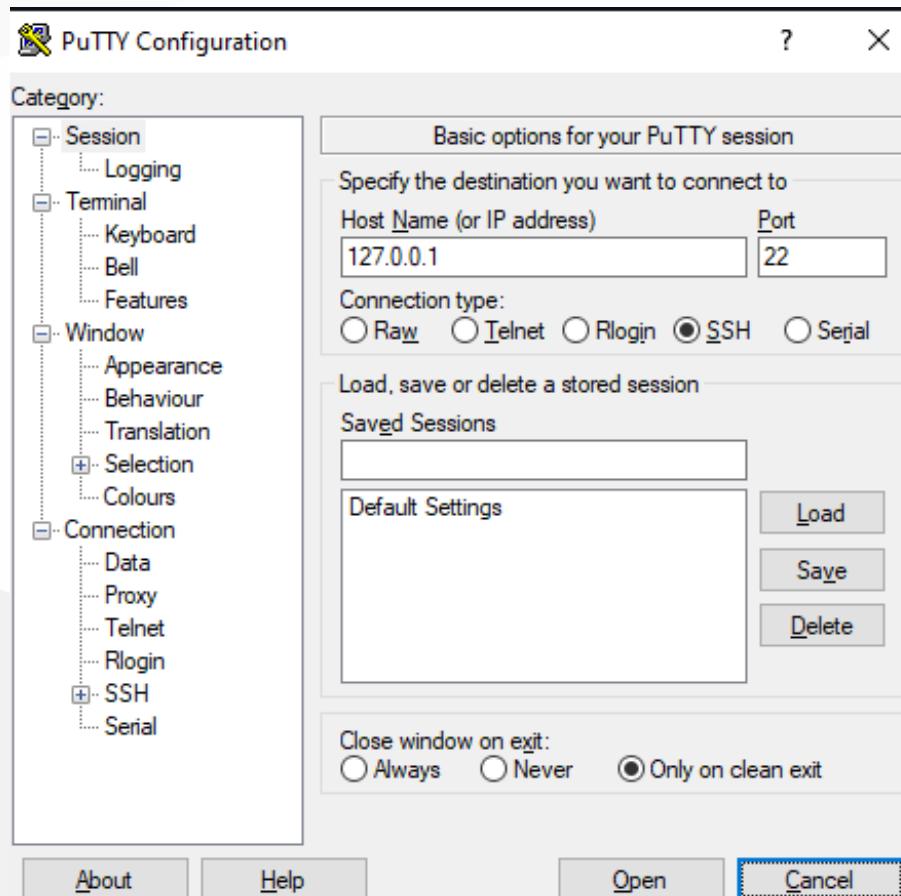
Putty (Remote Connection)

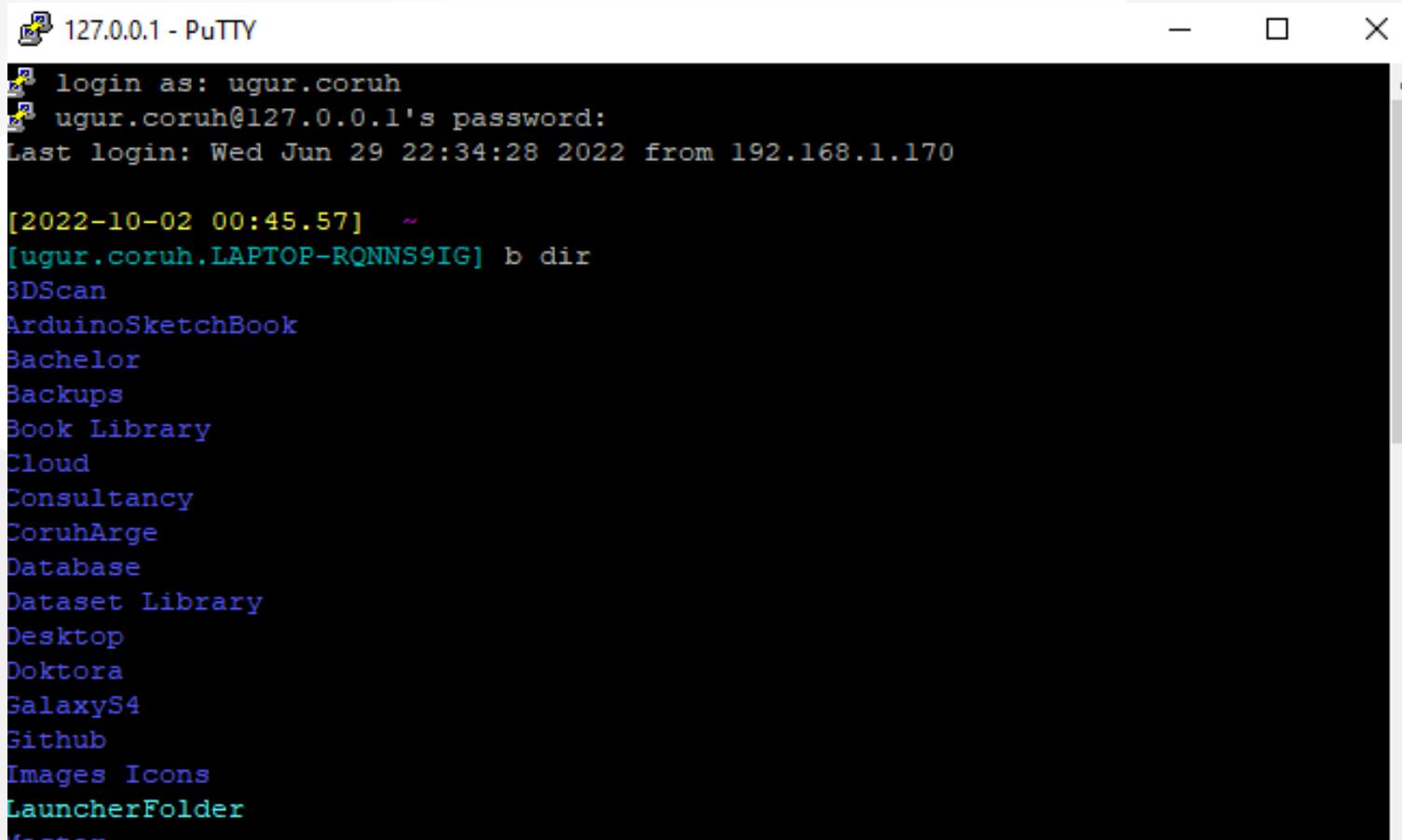
- Commonly use for SSH connection



- We can run a SSH server with MobaXterm and can connect to same computer with Putty.







A screenshot of a PuTTY terminal window titled "127.0.0.1 - PuTTY". The window shows a command-line interface where the user has logged in as "ugur.coruh" and is viewing their home directory. The directory listing includes several folders such as "BDScan", "ArduinoSketchBook", "Bachelor", "Backups", "Book Library", "Cloud", "Consultancy", "CoruhArge", "Database", "Dataset Library", "Desktop", "Doktora", "GalaxyS4", "Github", "Images Icons", "LauncherFolder", and "Muzik". The background of the terminal is black, and the text is white.

```
[2022-10-02 00:45.57] ~
[ugur.coruh.LAPTOP-RQNNS9IG] b dir
BDScan
ArduinoSketchBook
Bachelor
Backups
Book Library
Cloud
Consultancy
CoruhArge
Database
Dataset Library
Desktop
Doktora
GalaxyS4
Github
Images Icons
LauncherFolder
Muzik
```

Here are some useful examples for downloading files from the remote system over SSH protocol.

- This will connect to example.com server with user “username” and copy the **/backup/file.zip** file to local system directory **/local/dir**. To use theis command replace the values as per your environment.

```
scp username@example.com:/backup/file.zip /local/dir
```

- If the SSH is running on a non-standard port, You can specify the port using **-P** option with SCP command.

```
scp -P 2222 username@example.com:/backup/file.zip /local/dir
```

- If your remote server required a private key to connect server, You can use **-i** followed by a private key file path to connect your server using the SCP command. This can be helpful for AWS servers.

Upload file using SSH

You can also upload files to the remote server using SSH protocol using the SCP command. Use the following example command for uploading files to the SSH server.

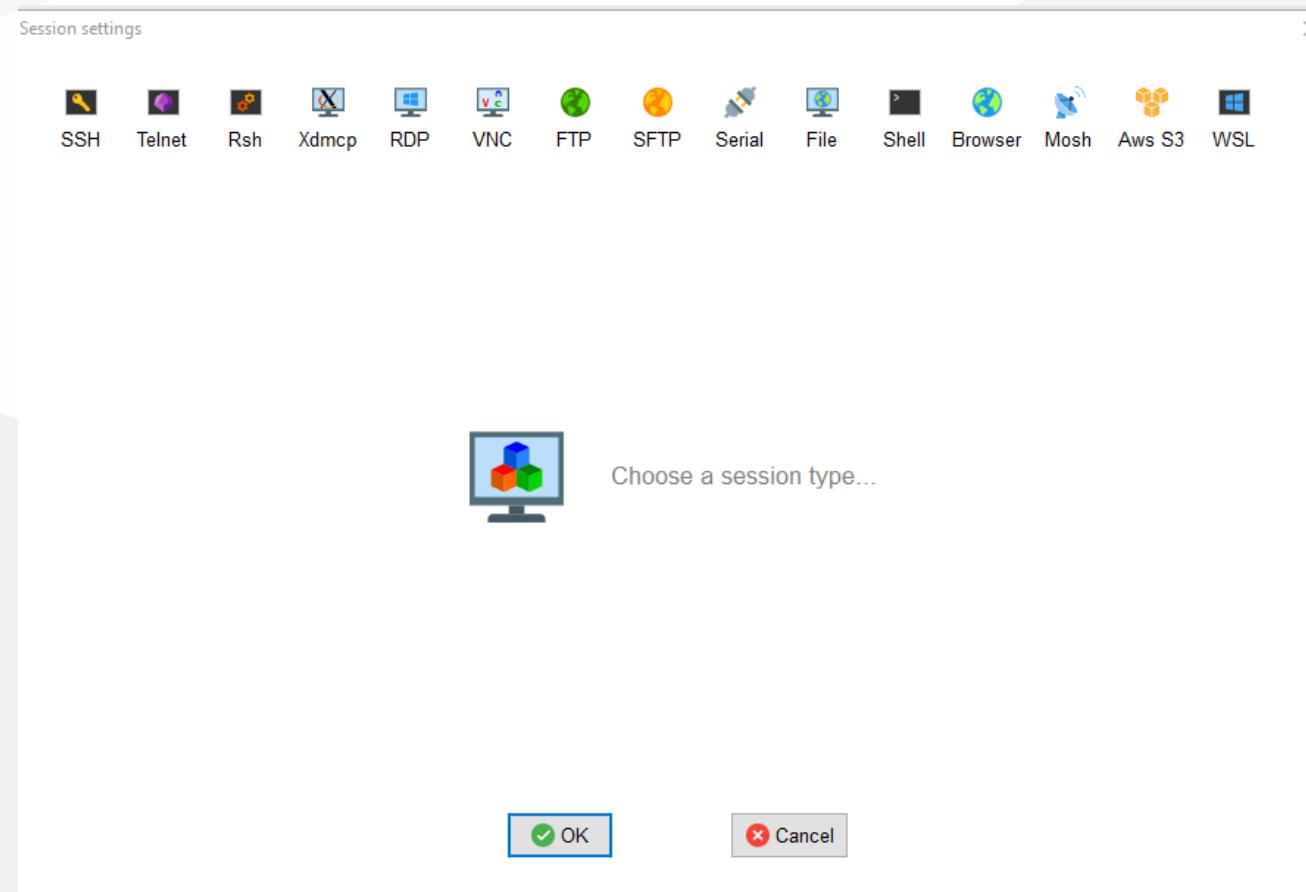
```
scp file.zip username@example.com:/remote/dir
```

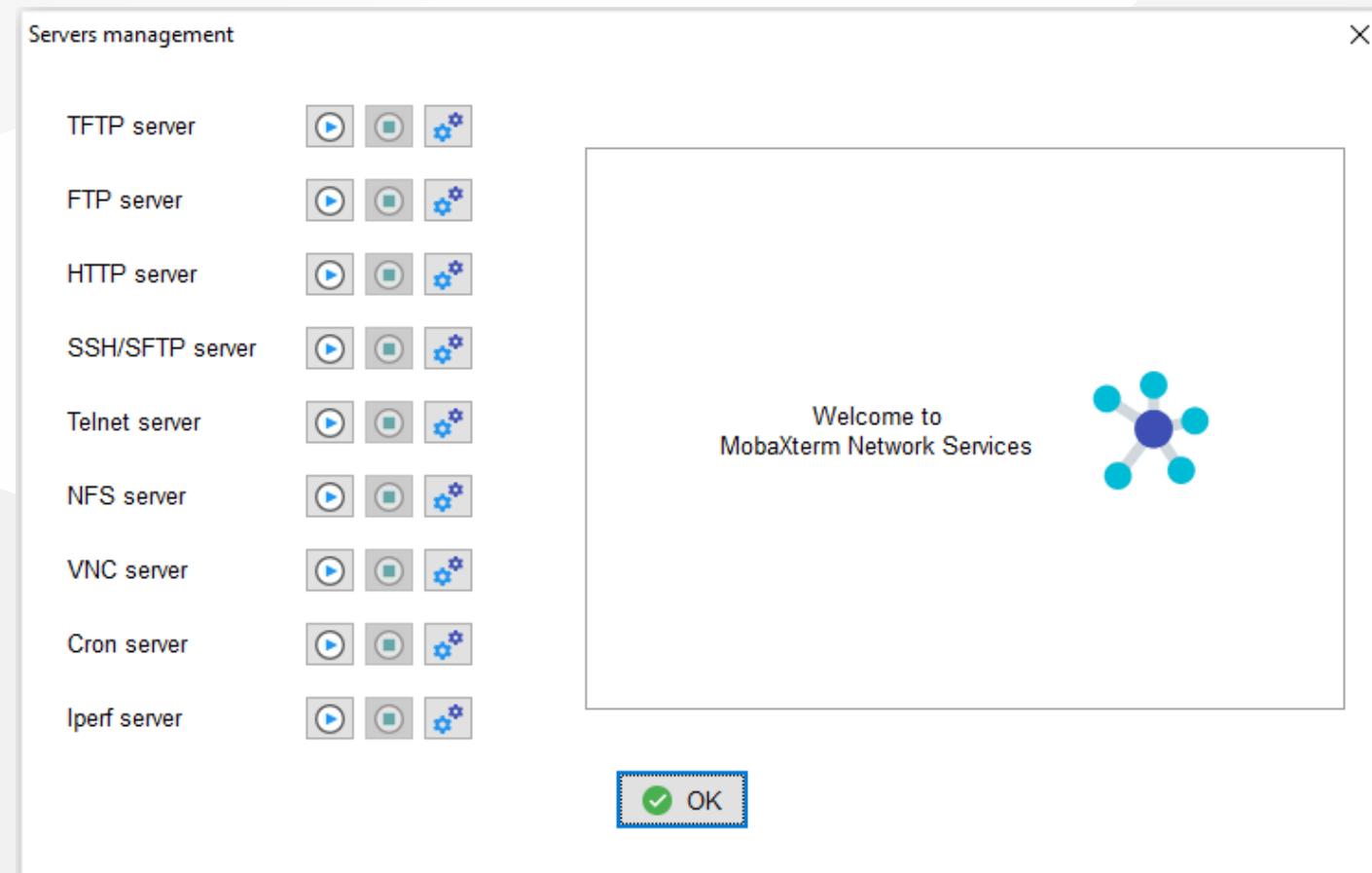
Similarly you can use -P switch to define port of the SSH server and -i to define private key for the user authentication.

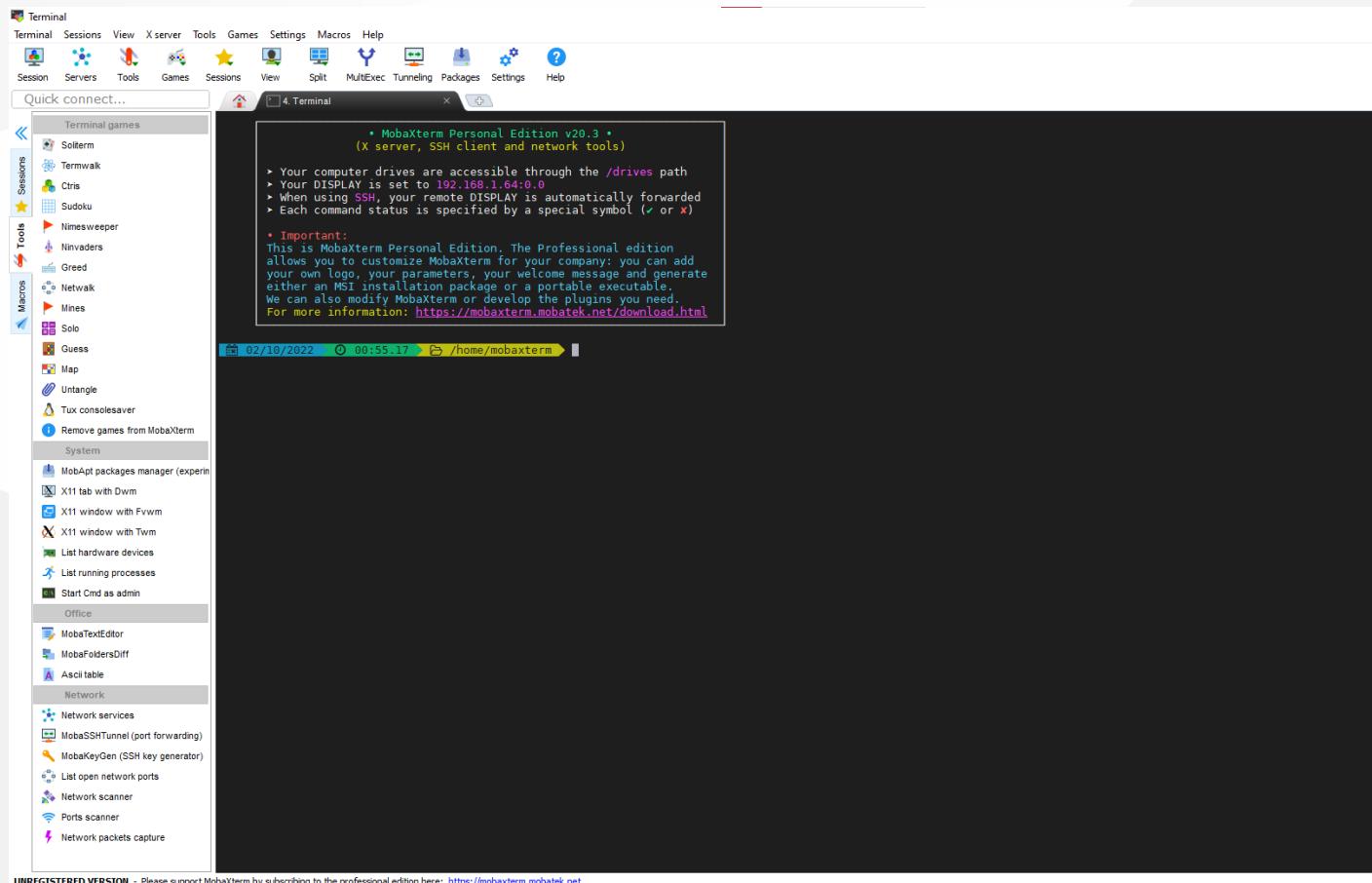
- Also you can use SSH tunnels for remote code development
 - [Developing on Remote Machines using SSH and Visual Studio Code](#)
 - [Visual Studio Code Server](#)

MobaXterm (Remote Connection)

- Multip Purpose Remote Connection Toolkit





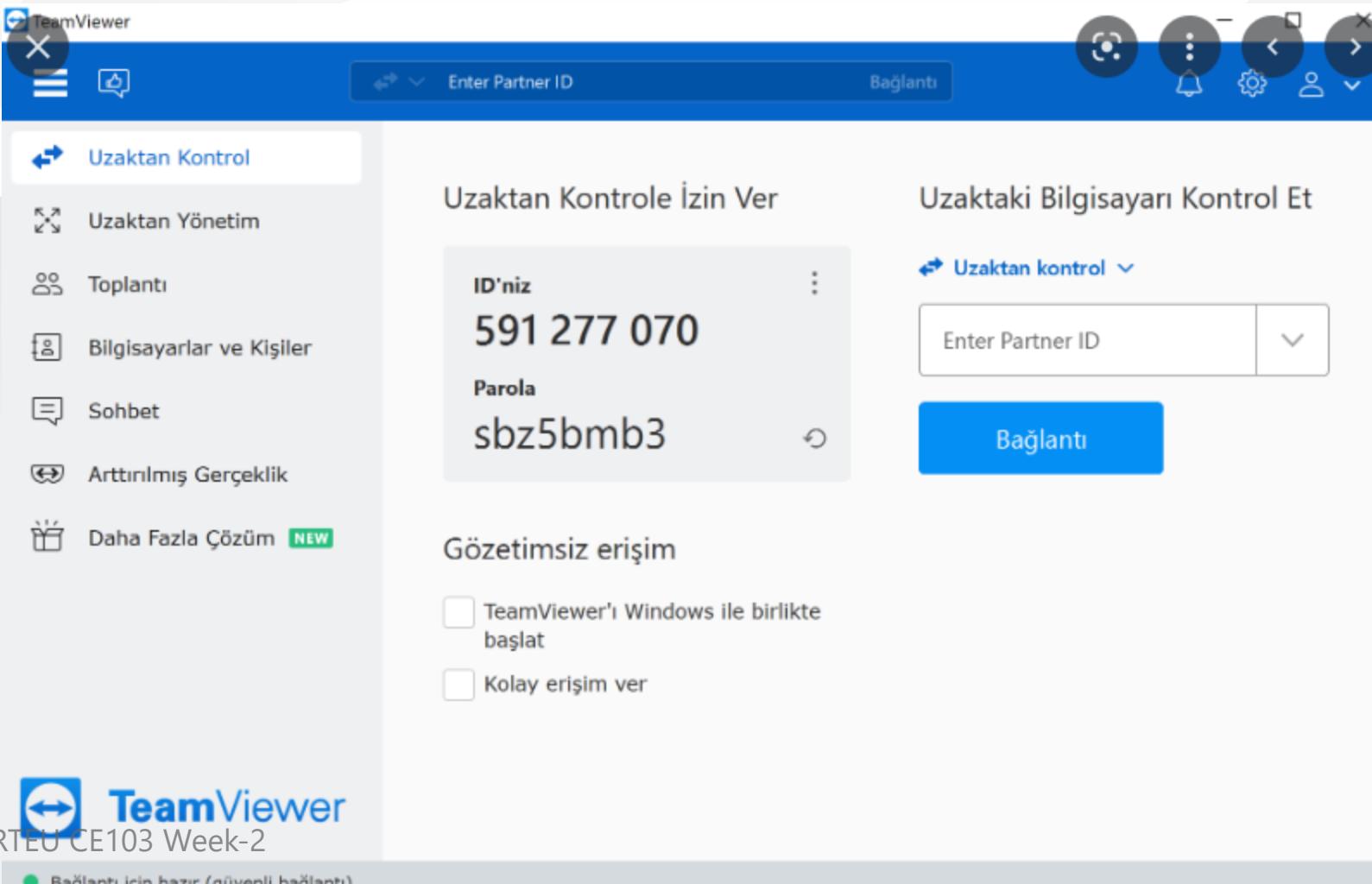


Teamviewer (Remote Connection)

CE103 Algorithms and Programming I

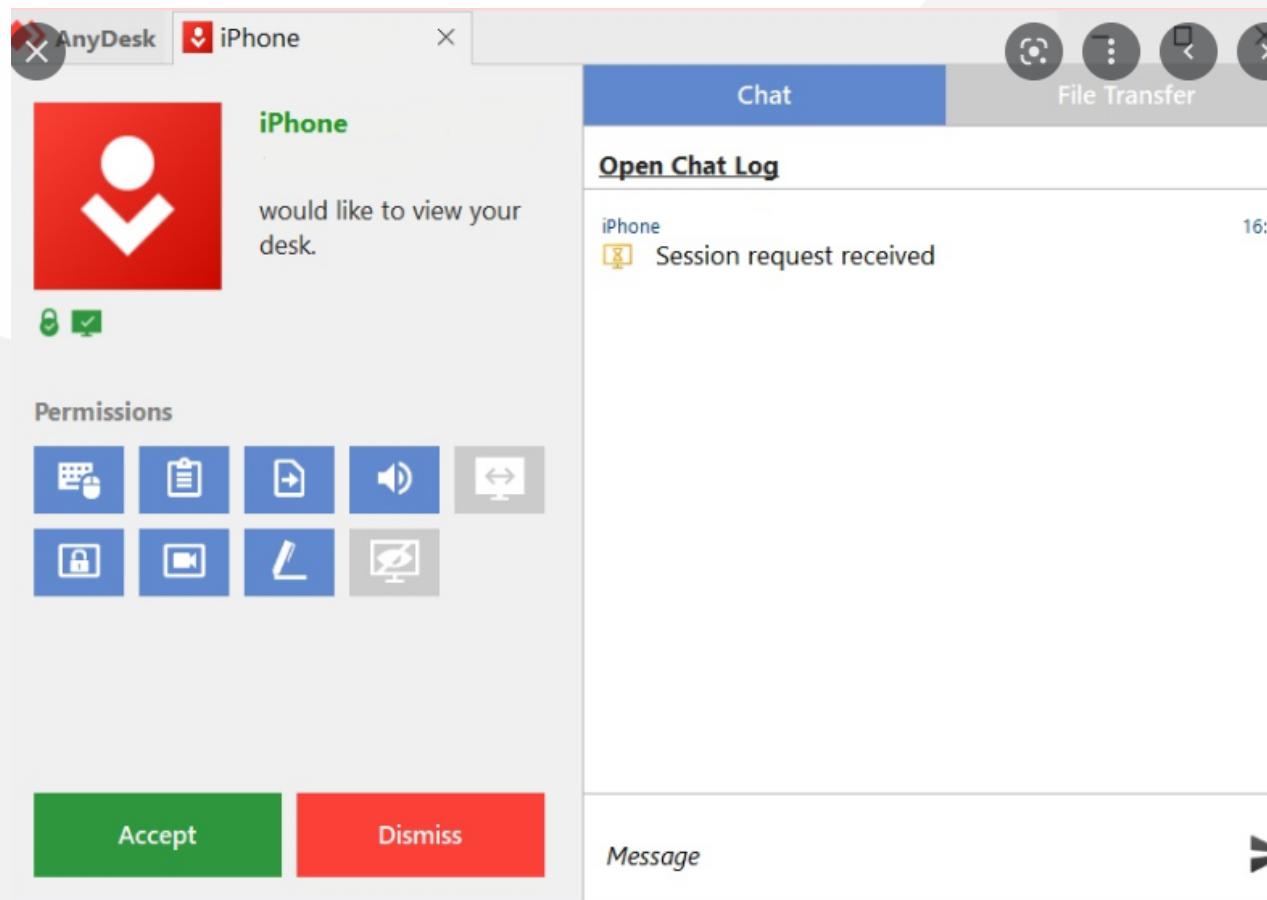
- Remote connection tool

- TeamViewer – Uzaktan Destek, Uzaktan Erişim, Hizmet Masası, Çevrimiçi İşbirliği ve Toplantılar

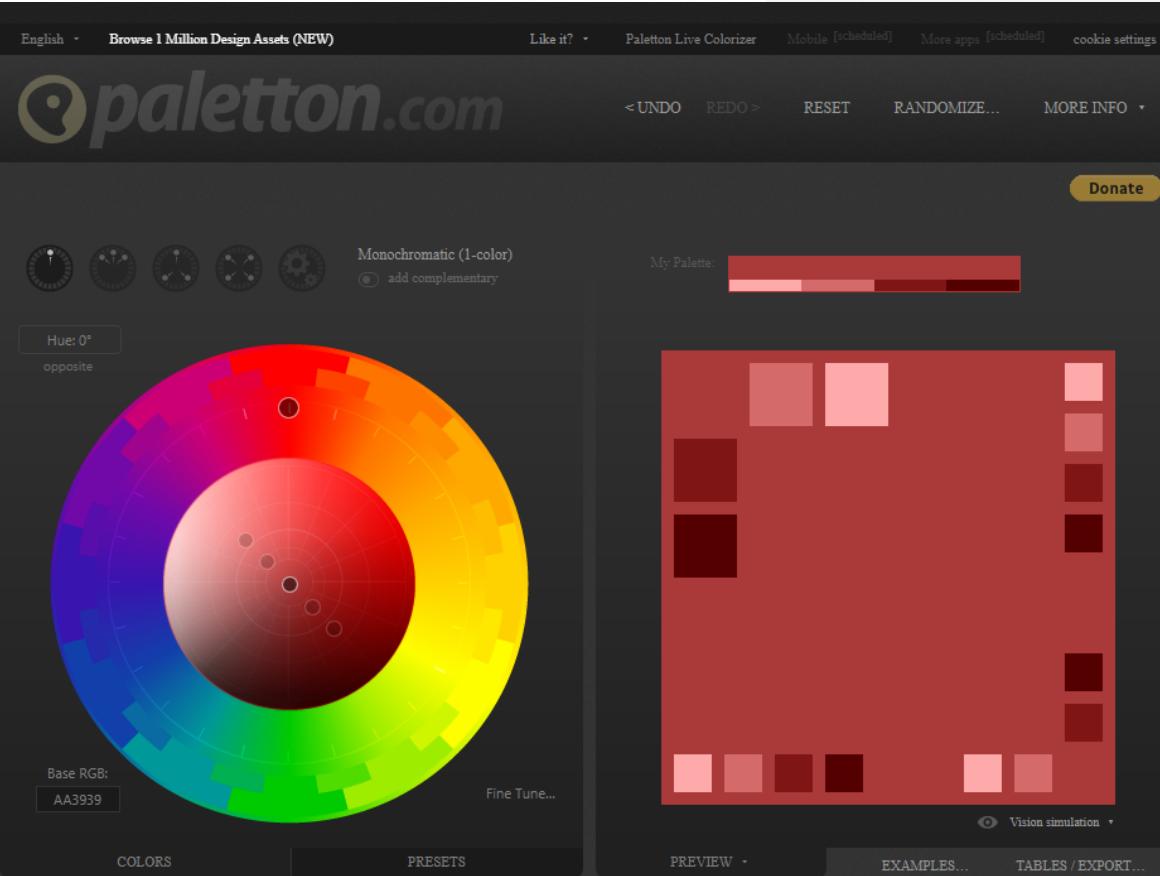


AnyDesk

- Remote connection tool
 - The Fast Remote Desktop Application – AnyDesk



- Generates color palettes and sample usages
 - [Paletton - The Color Scheme Designer](#)
 - <https://colorhunt.co/>
 - Also check [Colors Tutorial](#)

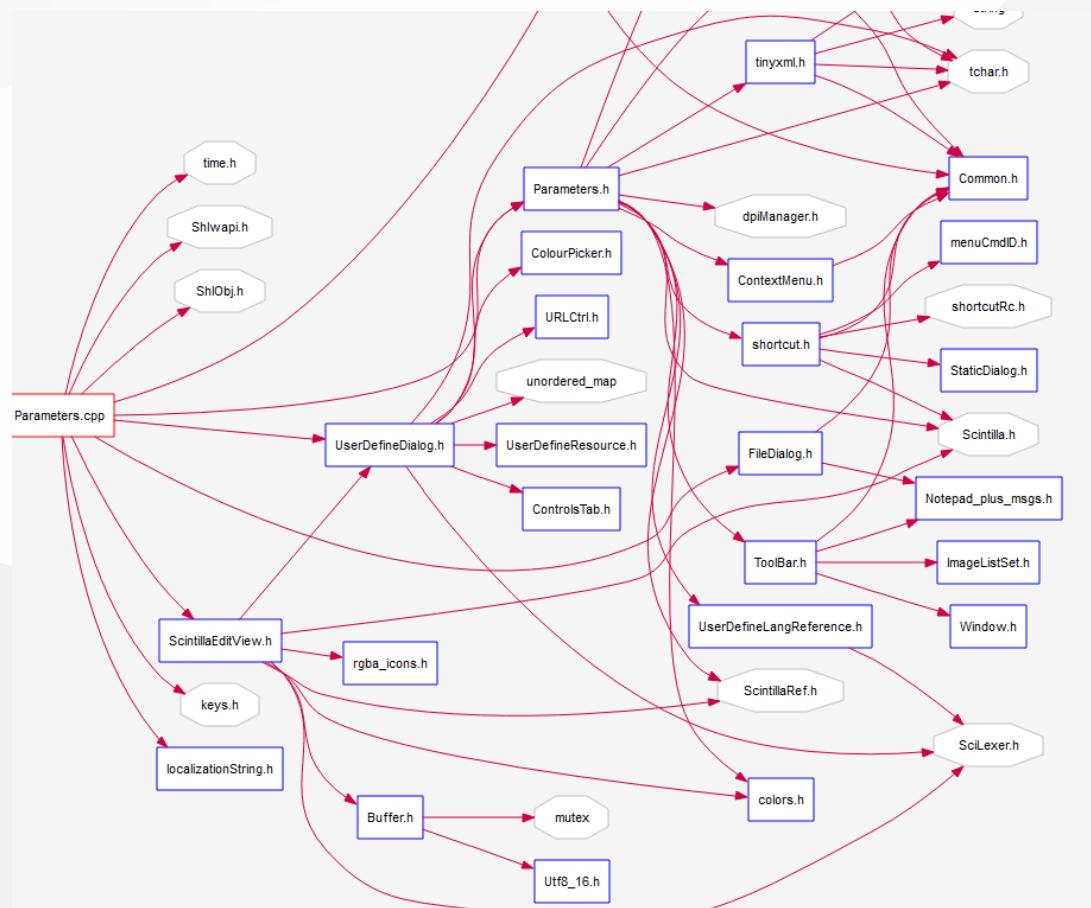




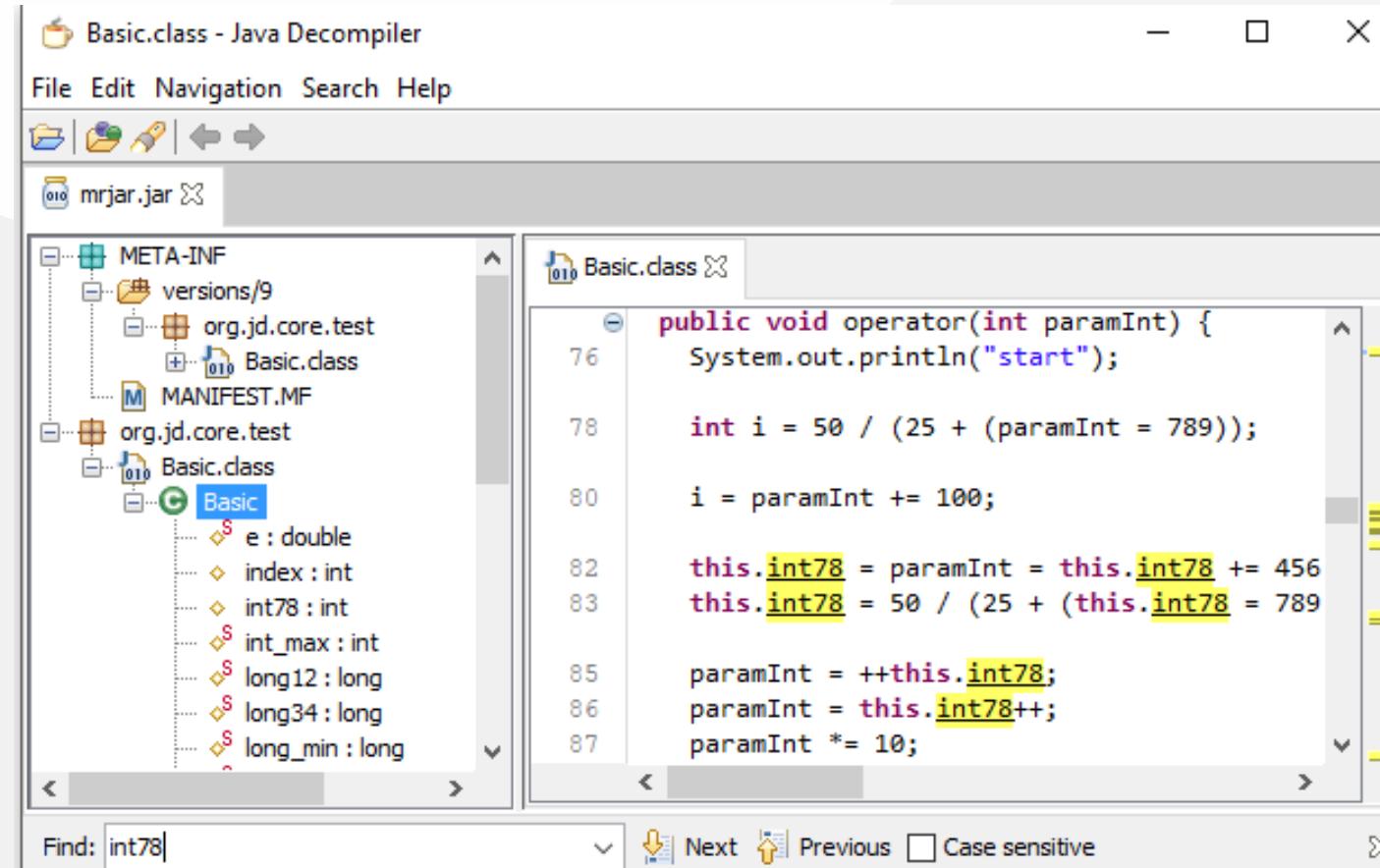
Understand (Static Code Analysis)

- <https://emenda.com/scitools-understand/>



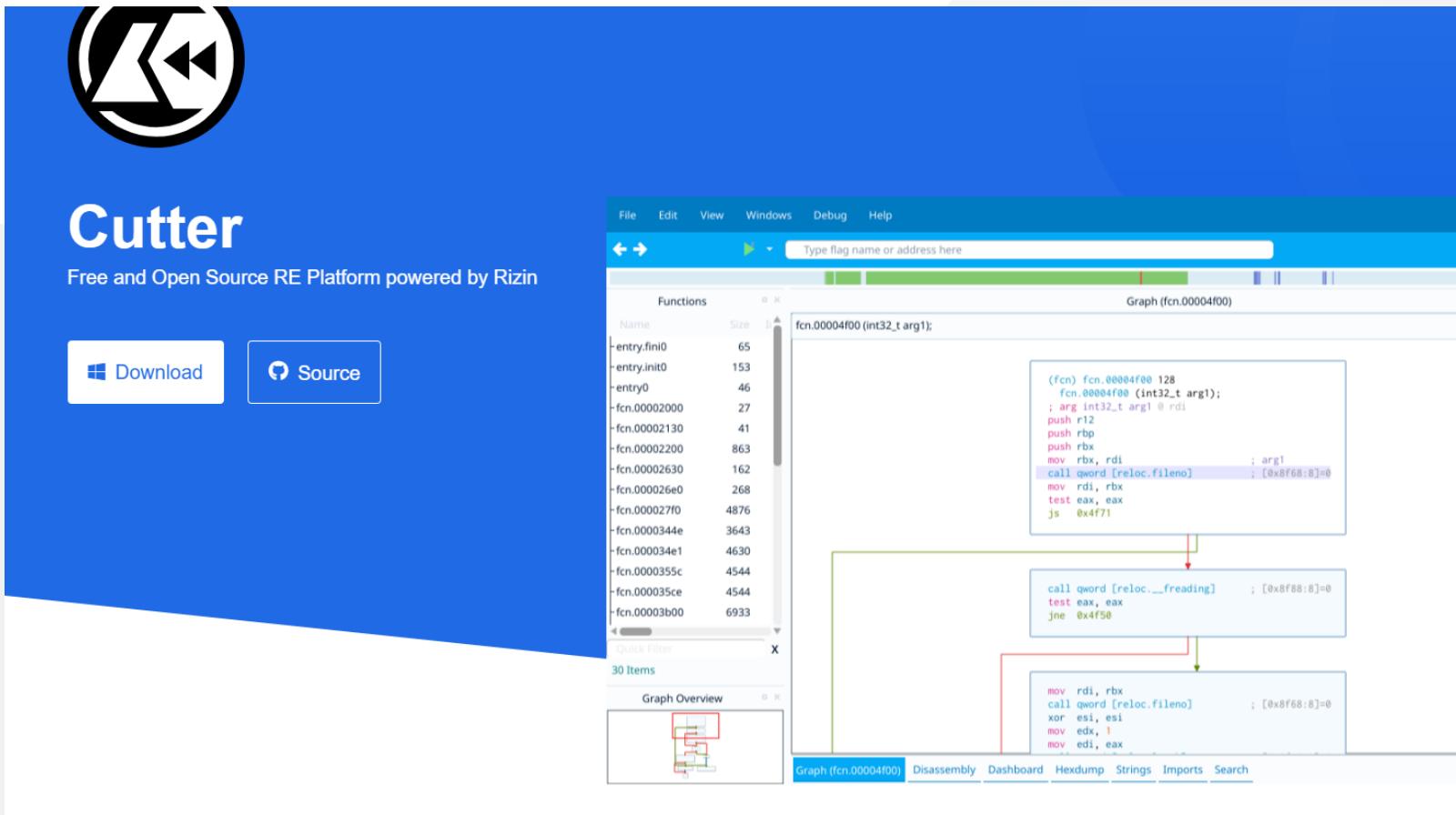


- Java Decompiler for Jar and Class Files, If code is not obfuscated it recover source code from compiled files. Just drag and drop files to GUI
 - <http://java-decompiler.github.io/>
 - You can use it standalone app or with eclipse



Cutter (Multi-Platform Reverse Engineering Tool)

- Cutter's goal is to be an advanced FREE and open-source reverse-engineering platform while keeping the user experience at mind. Cutter is created by reverse engineers for reverse engineers.
- <https://cutter.re/>



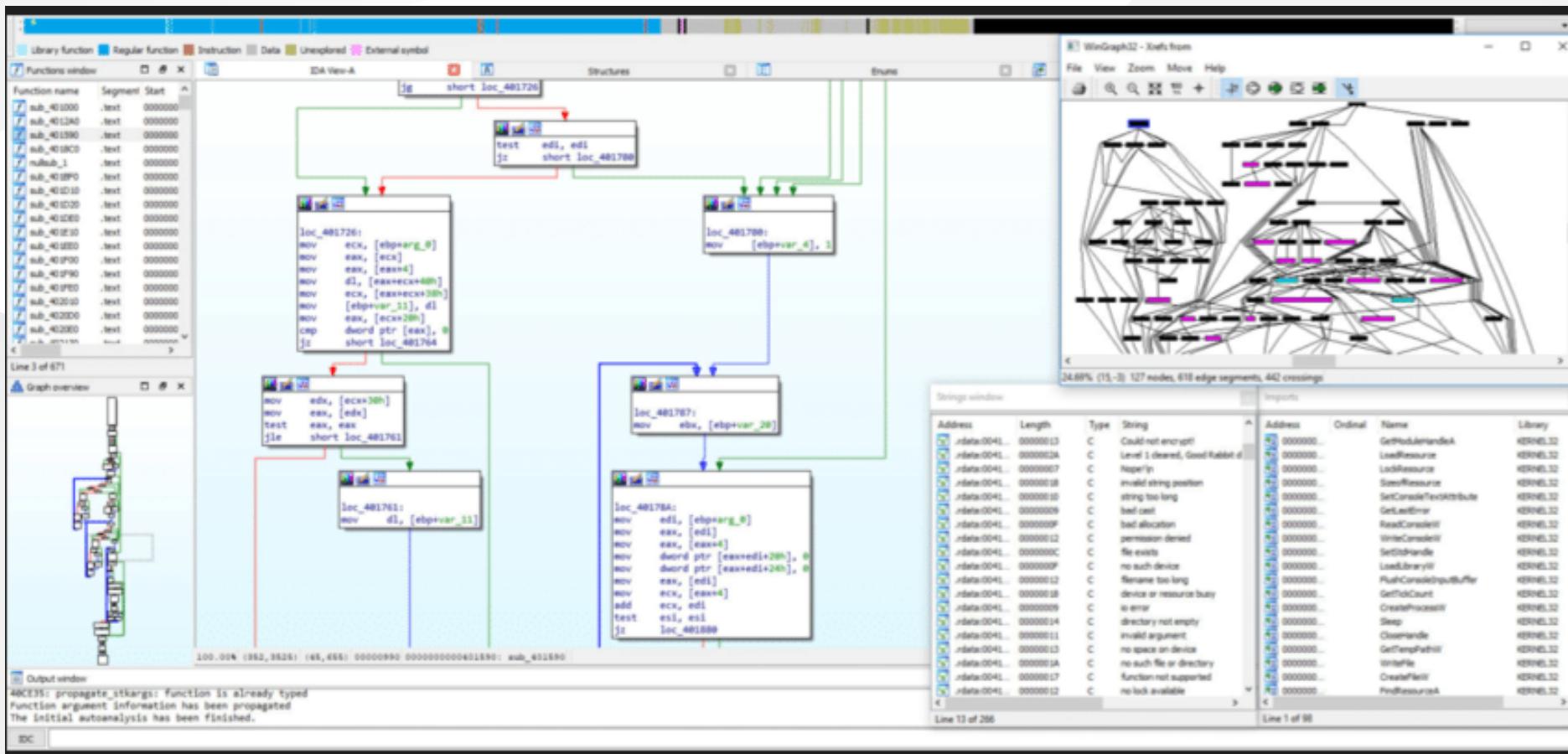
IDA Pro / Freeware (Native Reverse Engineering Tool)

- IDA Pro as a disassembler is capable of creating maps of their execution to show the binary instructions that are actually executed by the processor in a symbolic representation (assembly language). Advanced techniques have been implemented into IDA Pro so that it can generate assembly language source code from machine-executable code and make this complex code more human-readable.

IDA Pro / Freeware (Native Reverse Engineering Tool)

- Hex Rays - State-of-the-art binary code analysis solutions

IDA Pro / Freeware (Native Reverse Engineering Tool)



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

- This coding tutor tool helps you learn Python, JavaScript, C, C++, and Java by visualizing code execution.
 - <https://pythontutor.com/>

Python 3.6

```

1 def listSum(numbers):
2     if not numbers:
3         return 0
4     else:
5         (f, rest) = numbers
6         return f + listSum(rest)
7
8 myList = (1, (2, (3, None)))
9 total = listSum(myList)

```

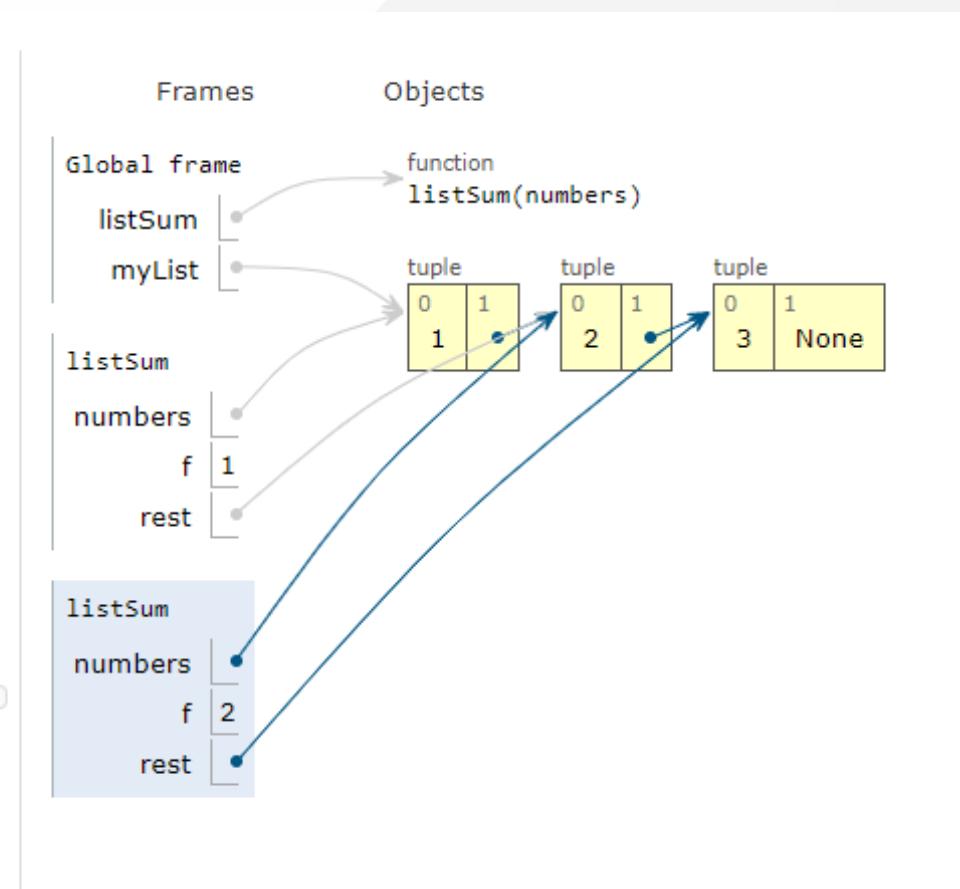
[Edit this code](#)

Green arrow: line that just executed
Red arrow: next line to execute

< Prev Next >

Step 11 of 22

Visualized using [Python Tutor](#)
[Customize visualization](#)



Assembly of C Code

- Multilanguage supported. Convert source code to assembly codes
 - <https://godbolt.org/>

The screenshot shows the Compiler Explorer interface with the following details:

Left Panel (C++ Source #1):

```

1 #include <stdio.h>
2 int addNumbers(int a, int b); // function prototype
3
4 int main()
5 {
6     int n1,n2,sum;
7
8     printf("Enters two numbers: ");
9     scanf("%d %d",&n1,&n2);
10
11    sum = addNumbers(n1, n2); // function call
12    printf("sum = %d",sum);
13
14    return 0;
15 }
16
17 int addNumbers(int a, int b) // function definition ...
18 {
19     int result;
20     result = a+b;
21     return result; // return statement
22 }
```

Right Panel (Assembly Output):

```

1 .LC0:
2     .string "Enters two numbers: "
3 .LC1:
4     .string "%d %d"
5 .LC2:
6     .string "sum = %d"
7 main:
8     push rbp
9     mov rbp, rsp
10    sub rsp, 16
11    mov edi, OFFSET FLAT:.LC0
12    mov eax, 0
13    call printf
14    lea rdx, [rbp-12]
15    lea rax, [rbp-8]
16    mov rsi, rax
17    mov edi, OFFSET FLAT:.LC1
18    mov eax, 0
19    call __isoc99_scanf
20    mov edx, DWORD PTR [rbp-12]
21    mov eax, DWORD PTR [rbp-8]
22    mov esi, edx
23    mov edi, eax
24    call addNumbers(int, int)
25    mov DWORD PTR [rbp-4], eax
26    mov eax, DWORD PTR [rbp-4]
27    mov esi, eax
28    mov edi, OFFSET FLAT:.LC2
29    mov eax, 0
30    call printf
31    mov eax, 0
32    leave
33    ret
34 addNumbers(int, int):
```

A message at the top right says: "Windows Compilers Temporarily Disabled. Hope to fix by next week".

Mobile Device Screen Sharing for Demo

- Show USB or Wifi connected Mobile Device Screen on PC and Provide Controls
 - [GitHub - Genymobile/scrcpy: Display and control your Android device](#)
 - Run `scrcpy` for single mobile phone.
 - [Open Source Project - Scrcpy now works wirelessly](#)

Travis-CI

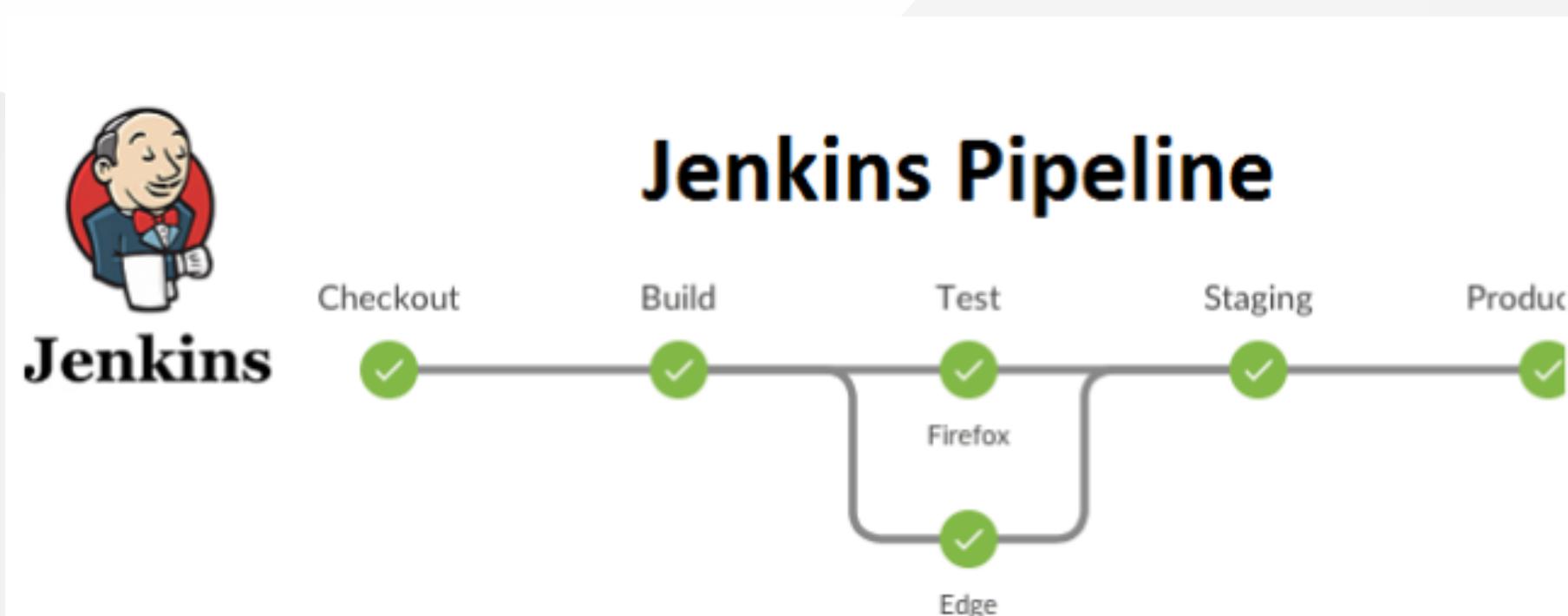
- Travis-CI is a continues integration platform
- Travis-CI free option removed for this reason, its not in our scope.
- It uses Travis.yml files for actions.

AppVeyor

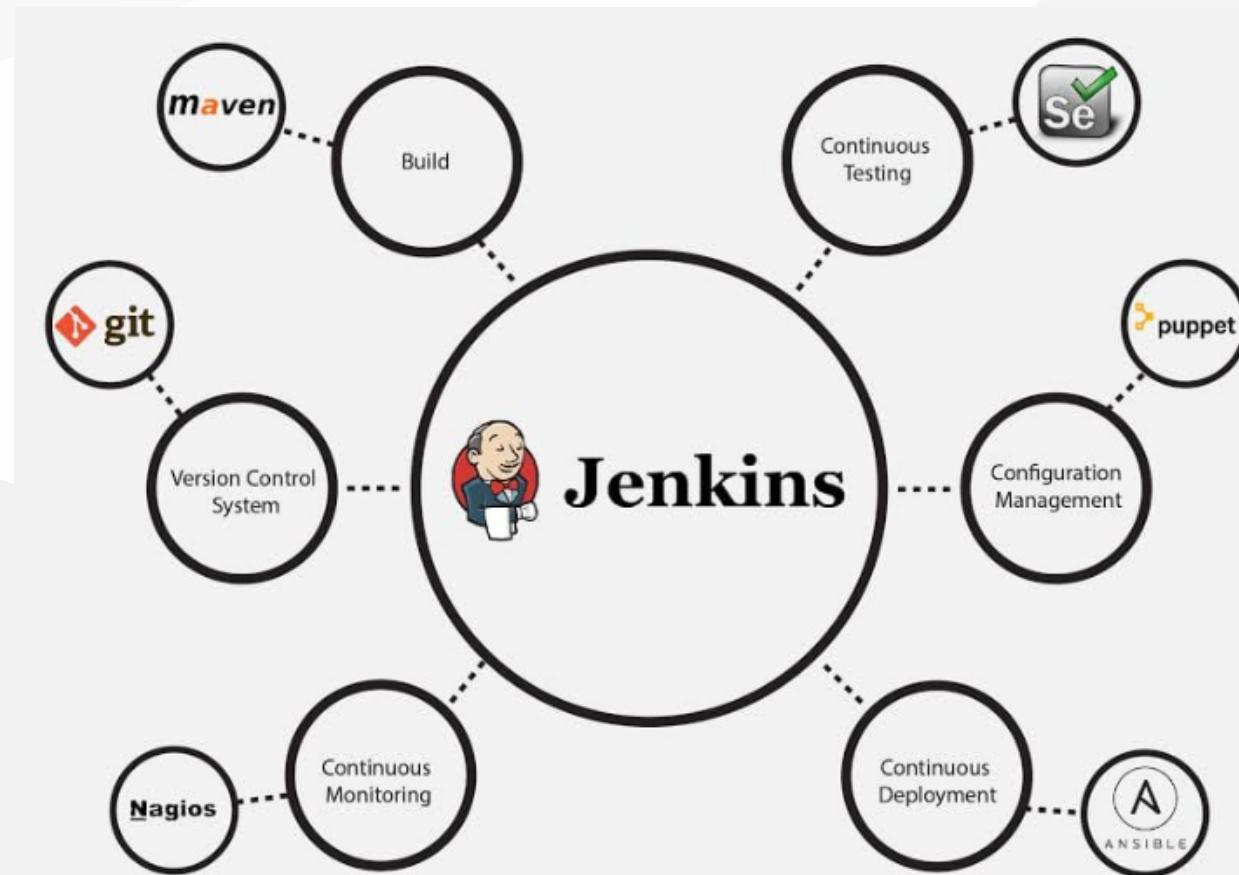
- Another CI platform it has free option for public builds.
 - <https://www.appveyor.com>
 - GitHub - Kimserey/hello-world-nuget
 - [hello-world-nuget/appveyor.yml at master · Kimserey/hello-world-nuget · GitHub](#)

Jenkins

- Self-hosted solution for CI operations, Has integration with Github and several platforms.
 - <https://www.jenkins.io/>
 - <https://www.jenkins.io/doc/pipeline/tour/hello-world/>



Jenkins





Configure Global Security

Enable security

TCP port for JNLP slave agents Fixed : Random Disable

Disable remember me

Access Control

Security Realm

Delegate to servlet container

Github Authentication Plugin

Global Github OAuth Settings

GitHub Web URI

GitHub API URI

Client ID

Client Secret

OAuth Scope(s)

Jenkins' own user database

LDAP

Unix user/group database

Authorization

Anyone can do anything

Github Committer Authorization Strategy

Github Authorization Settings

Admin User Names

Participant in Organization

Use Github repository permissions

Grant READ permissions to all Authenticated Users

Grant CREATE Job permissions to all Authenticated Users

Jenkins

- <https://www.jenkins.io/solutions/github/>

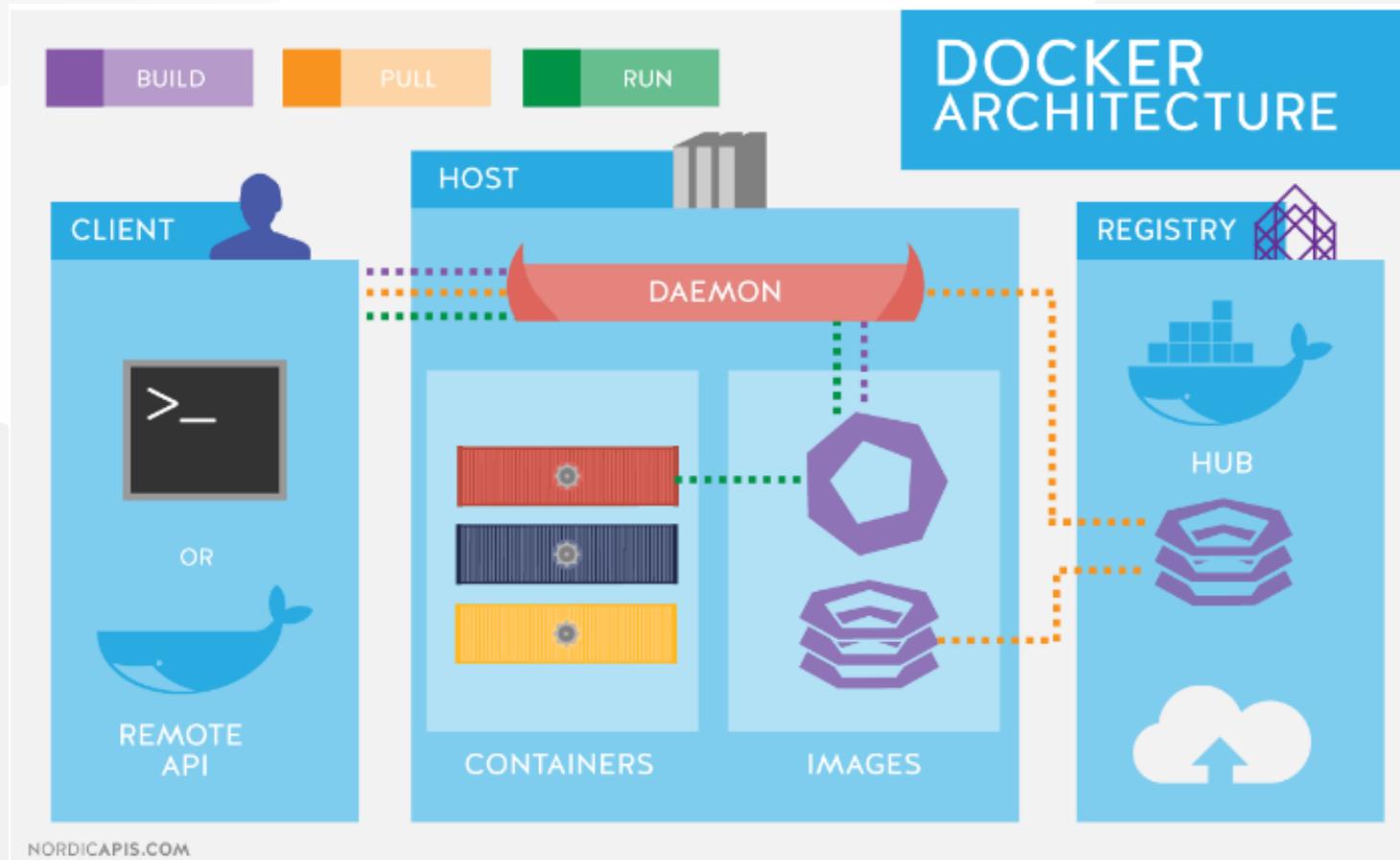
Vagrant

- Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.
 - <https://www.vagrantup.com/>
- Setup Development Environment with Vagrant
 - [Setting Up Development Environment Using Vagrant - Edureka](#)
 - [development-environment/Vagrantfile at master · gantsign/development-environment · GitHub](#)

Docker / Docker Compose / Kubernetes (1)

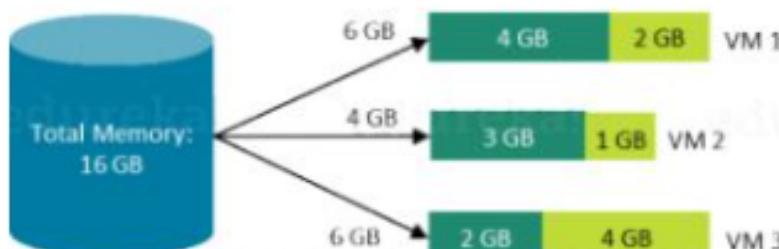
- Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development – desktop and cloud.
 - https://www.youtube.com/watch?v=nBwJm0onzeo&ab_channel=GaryExplains Dockerfile
 - <https://devopedia.org/docker>
- DockerHub
- Docker Compose Yaml
- Dockerrun.aws.json (AWS)

Docker / Docker Compose / Kubernetes (2)



Docker / Docker Compose / Kubernetes (3)

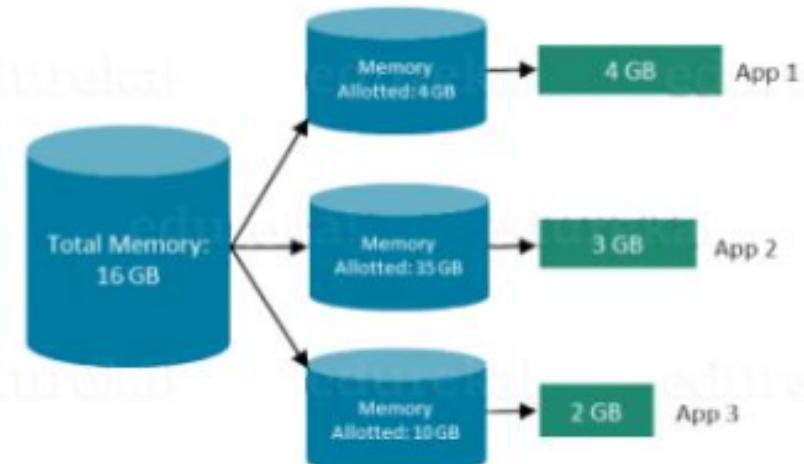
In case of Virtual Machines



- [Green Square] → Memory Used: 9 GB
- [Yellow-green Square] → Memory wasted: 7 GB

7 Gb of Memory is blocked and cannot be allotted to a new VM

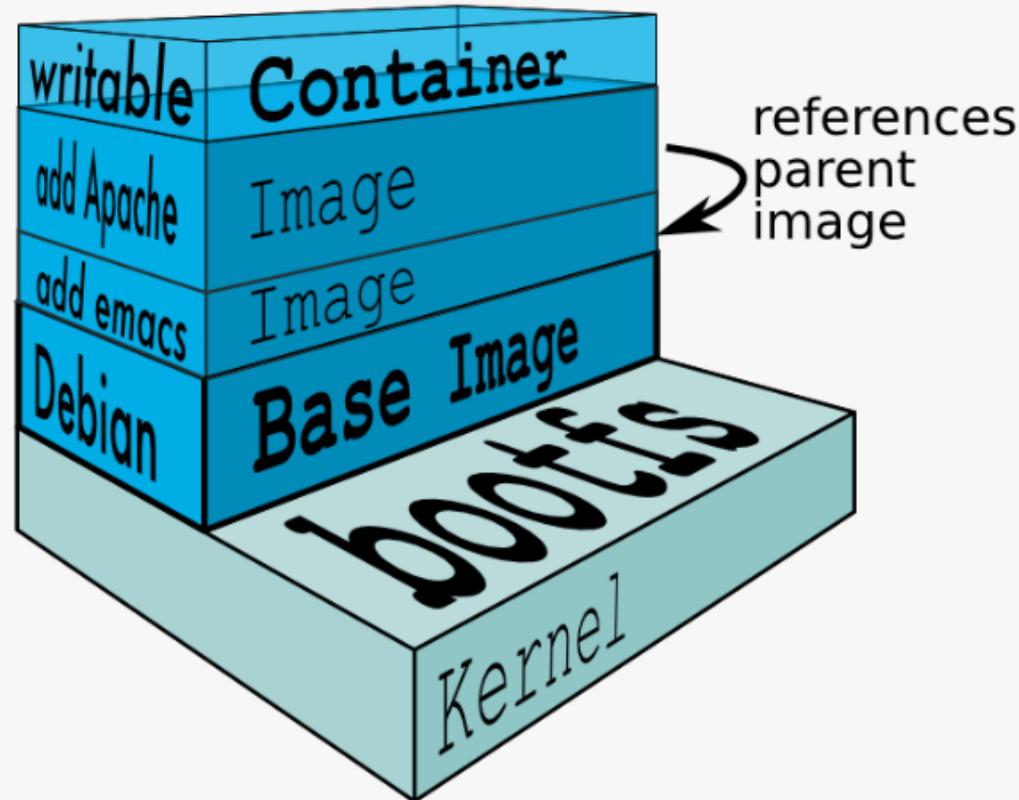
In case of Docker



- [Green Square] → Memory Used: 9 GB

Only 9 GB memory utilized;
7 GB can be allotted to a new Container

Docker / Docker Compose / Kubernetes (4)



Docker / Docker Compose / Kubernetes (5)

```
FROM node:9.3.0-alpine

RUN npm install -g @angular/cli@1.5.5 \
    && mkdir -p /usr/src/pintail-whoami

WORKDIR /usr/src/pintail-whoami

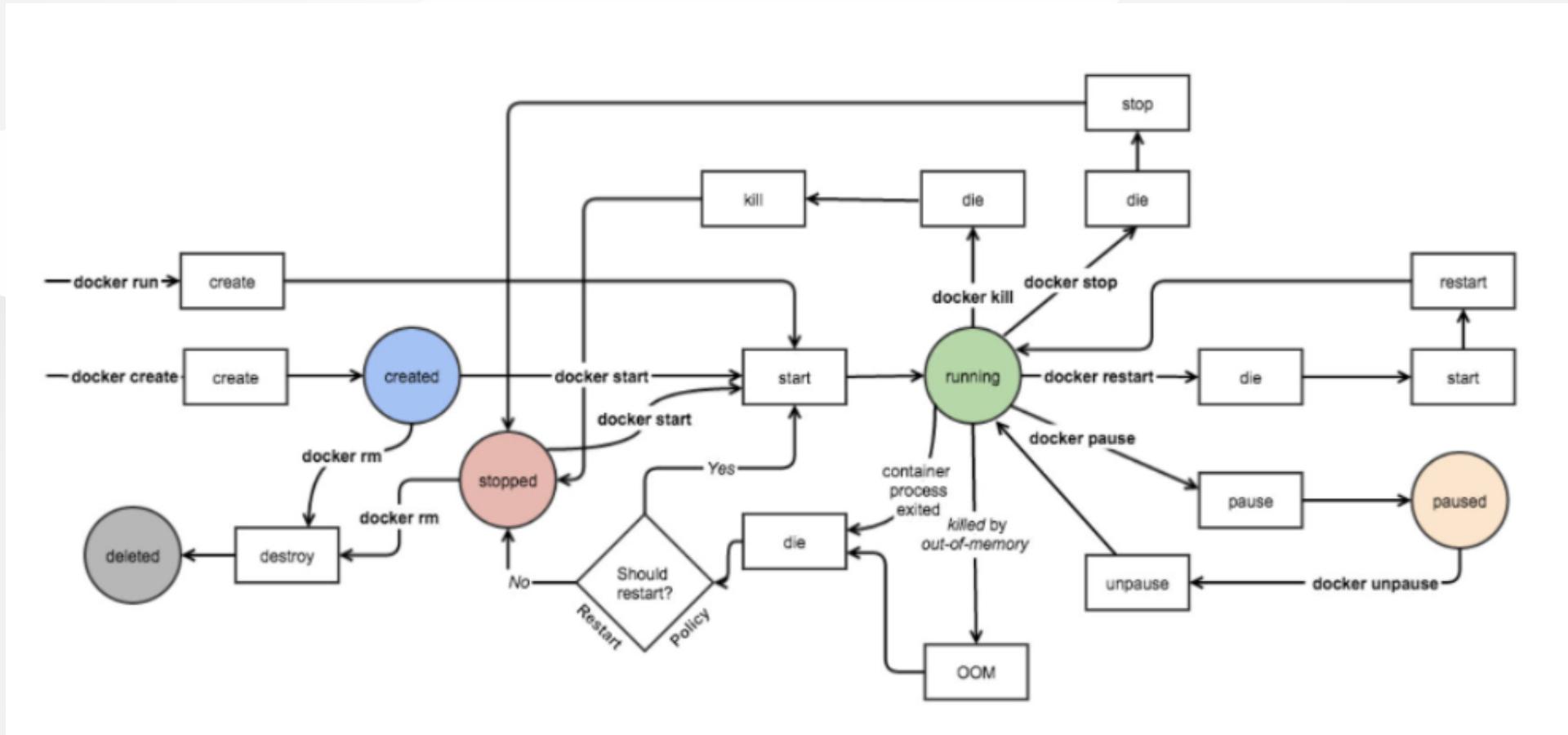
ADD . /usr/src/pintail-whoami

RUN npm install && ng build

EXPOSE 80

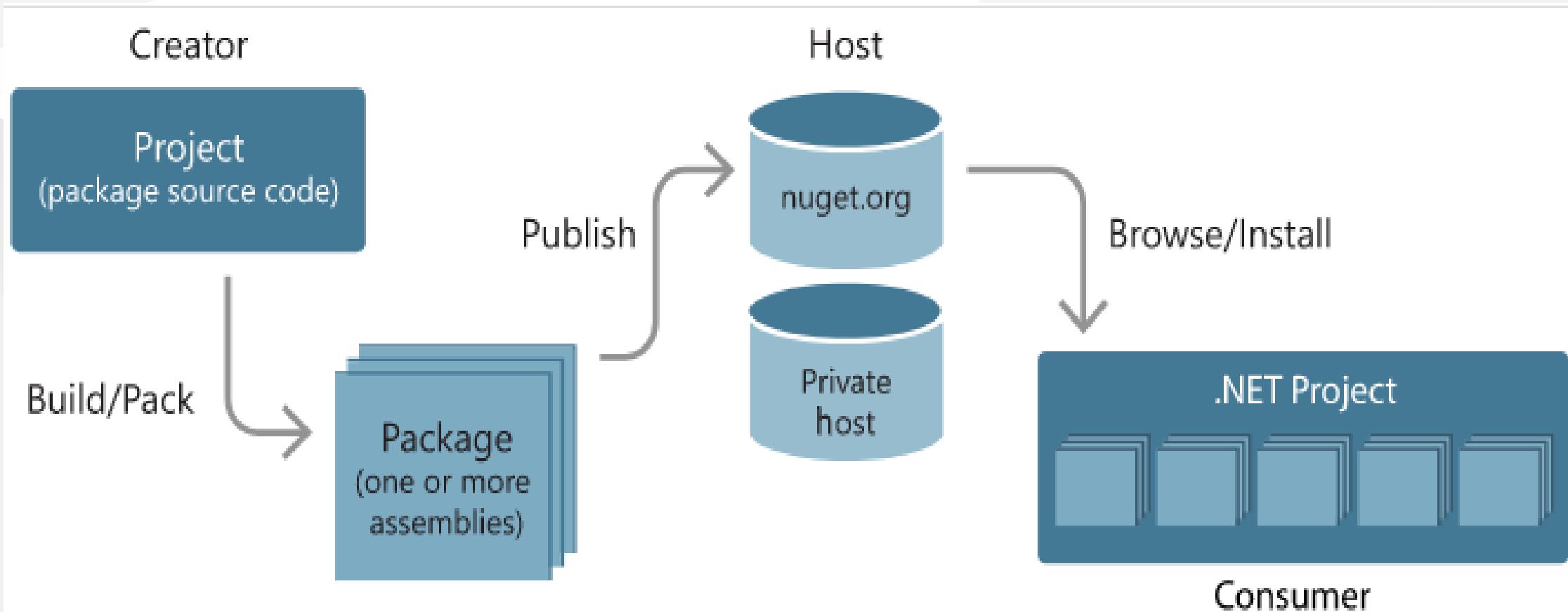
CMD node server.js $HOSTNAME
```

Docker / Docker Compose / Kubernetes (6)



NuGet Packages (1)

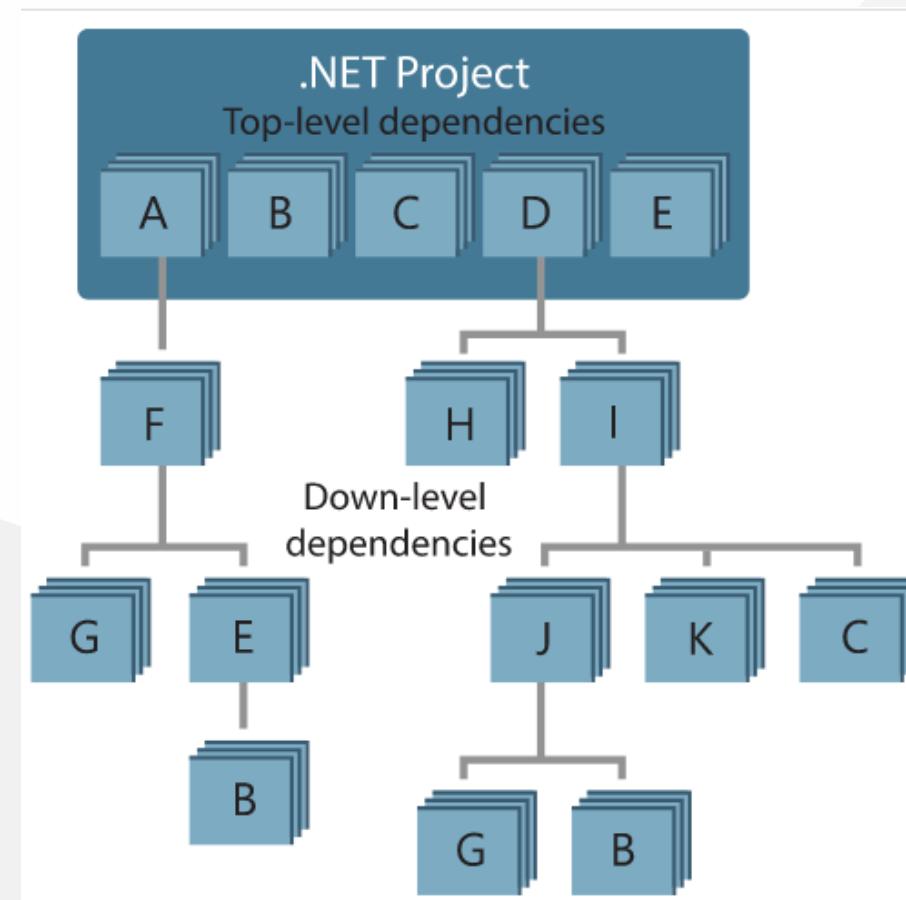
- <https://www.nuget.org/packages>
- What is NuGet and what does it do? | Microsoft Learn



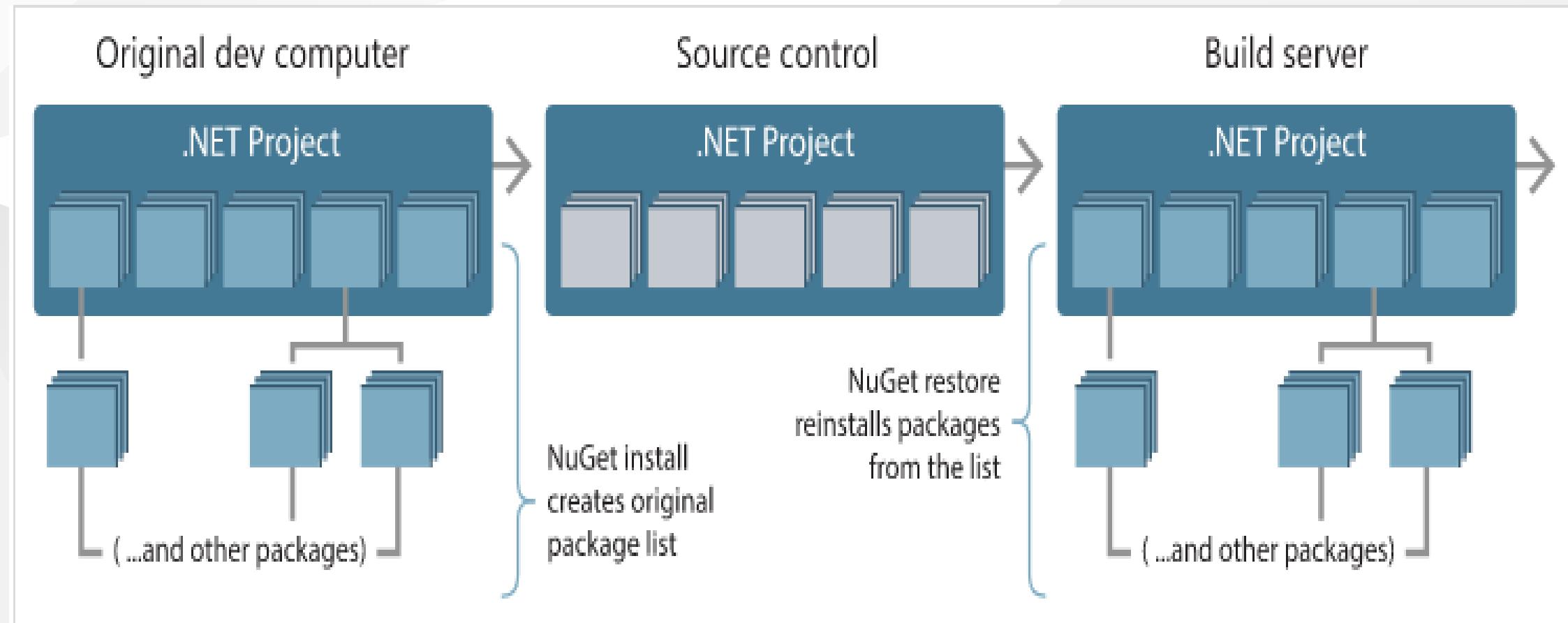
NuGet Tools (2)

Tool	Platforms	Applicable Scenarios	Description
dotnet CLI	All	Creation, Consumption	CLI tool for .NET Core and .NET Standard libraries, and for SDK-style projects that target .NET Framework (see SDK attribute). Provides certain NuGet CLI capabilities directly within the .NET Core tool chain. As with the <code>nuget.exe</code> CLI, the dotnet CLI does not interact with Visual Studio projects.
nuget.exe CLI	All	Creation, Consumption	CLI tool for .NET Framework libraries and non-SDK-style projects that target .NET Standard libraries. Provides all NuGet capabilities, with some commands applying specifically to package creators, some applying only to consumers, and others applying to both. For example, package creators use the <code>nuget pack</code> command to create a package from various assemblies and related files, package consumers use <code>nuget install</code> to include packages in a project folder, and everyone uses <code>nuget config</code> to set NuGet configuration variables. As a platform-agnostic tool, the NuGet CLI does not interact with Visual Studio projects.
Package Manager Console	Visual Studio on Windows	Consumption	Provides PowerShell commands for installing and managing packages in Visual Studio projects.
Package Manager UI	Visual Studio on Windows	Consumption	Provides an easy-to-use UI for installing and managing packages in Visual Studio projects.
Manage NuGet UI	Visual Studio for Mac	Consumption	Provide an easy-to-use UI for installing and managing packages in Visual Studio for Mac projects.
MSBuild	Windows	Creation, Consumption	Provides the ability to create packages and restore packages used in a project directly through the MSBuild tool chain.

Managing dependencies (3)

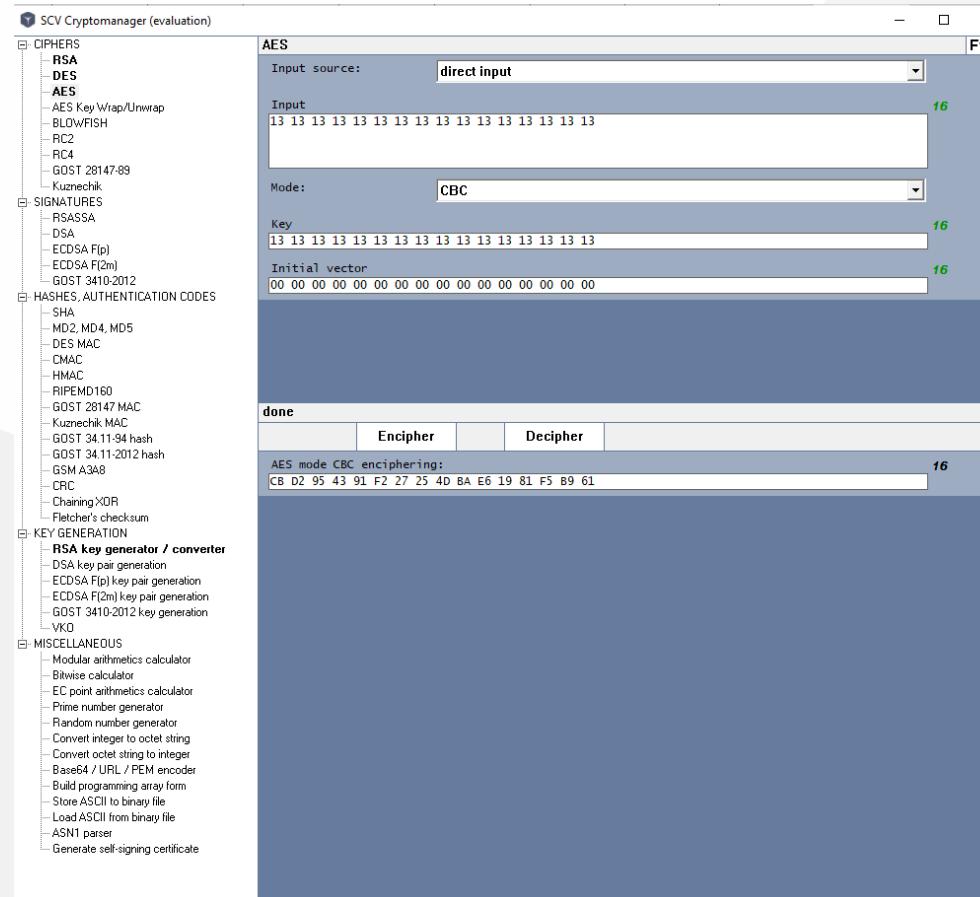


Tracking references and restoring packages (4)



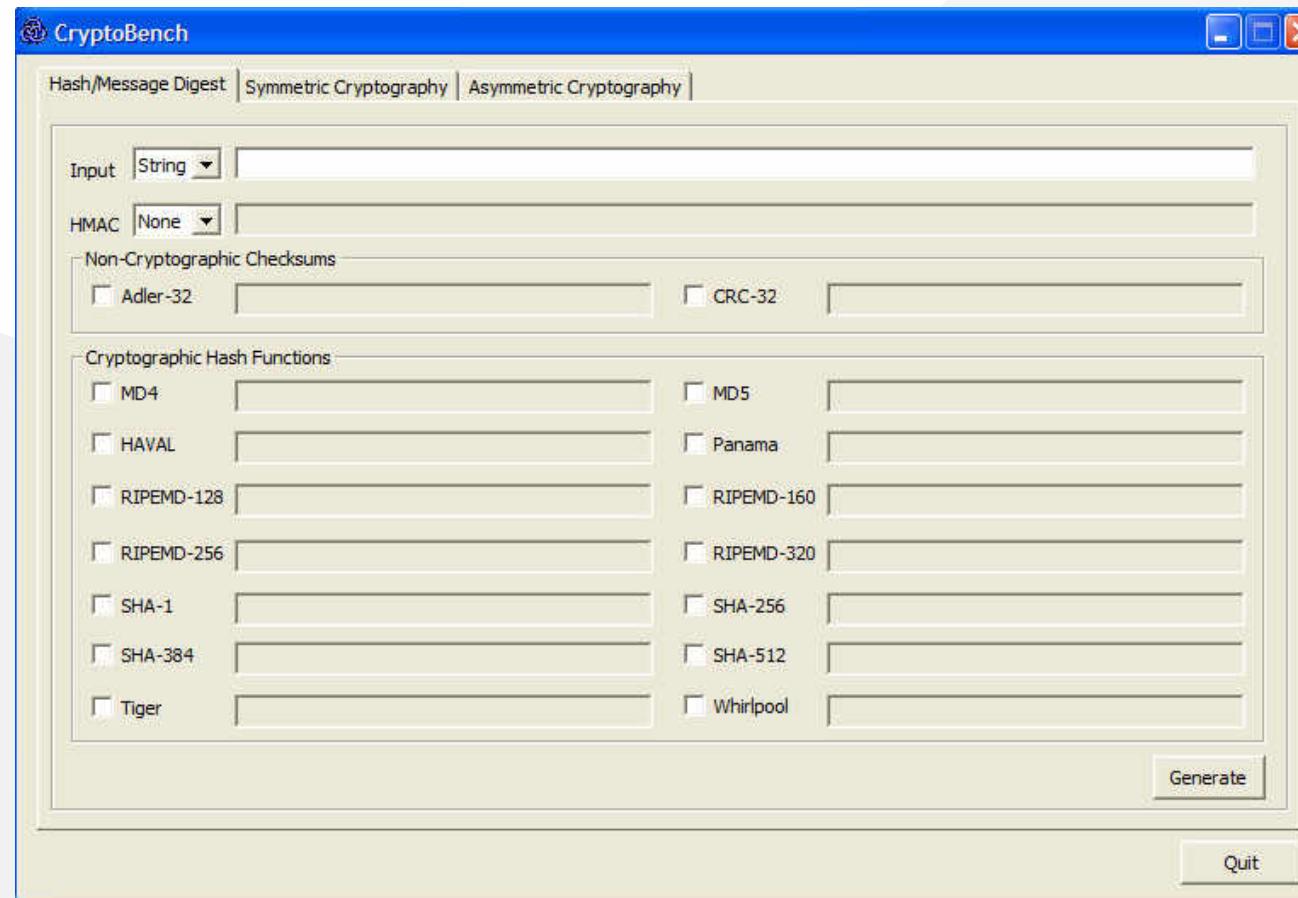
SCV Cryptomanager

- SCV Crypto Manager has several tools for cryptographic operations.
 - <https://cryptomanager.com/download.php>



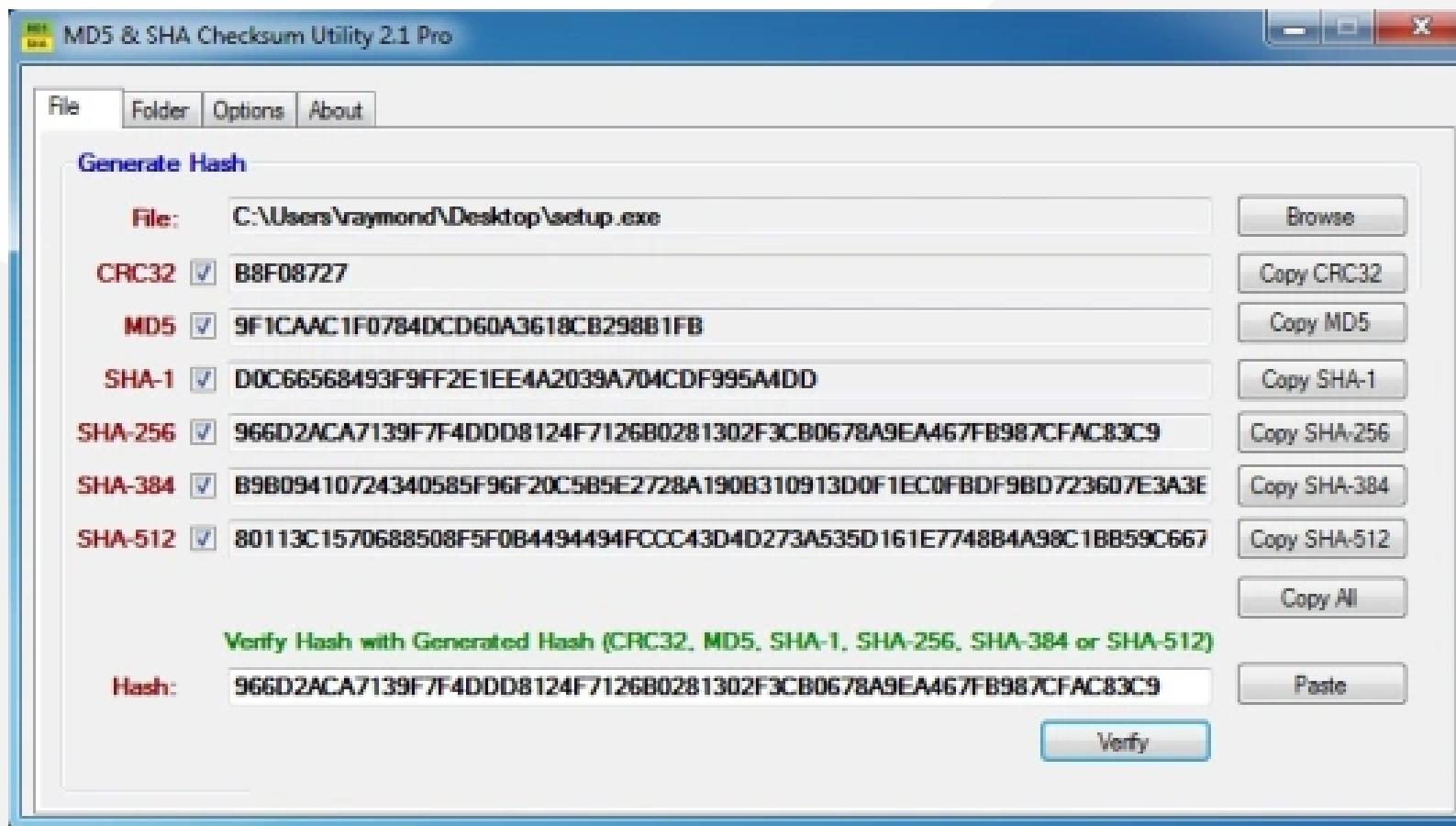
Addario CryptoBench

- CryptoBench can be used for hash and symmetric asymmetric encryption-decryption operations.
 - [CryptoBench Download Page](#)
 - <http://www.addario.org/files/CryptoBench v1.0.1.zip>



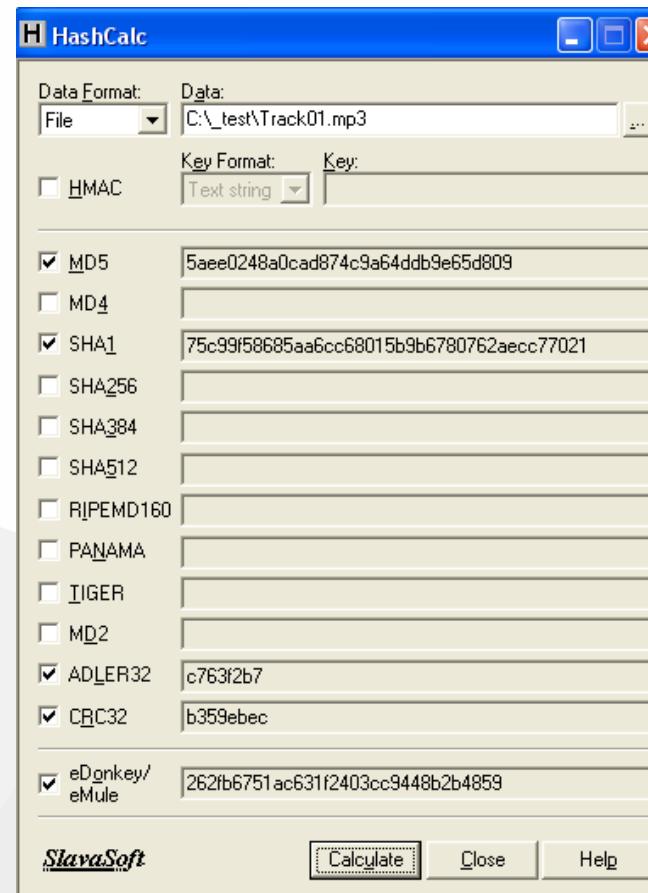
Raymond's MD5 & SHA Checksum Utility

- Hash Calculation Utility
- [MD5 & SHA Checksum Utility | Raymond's WordPress](#)

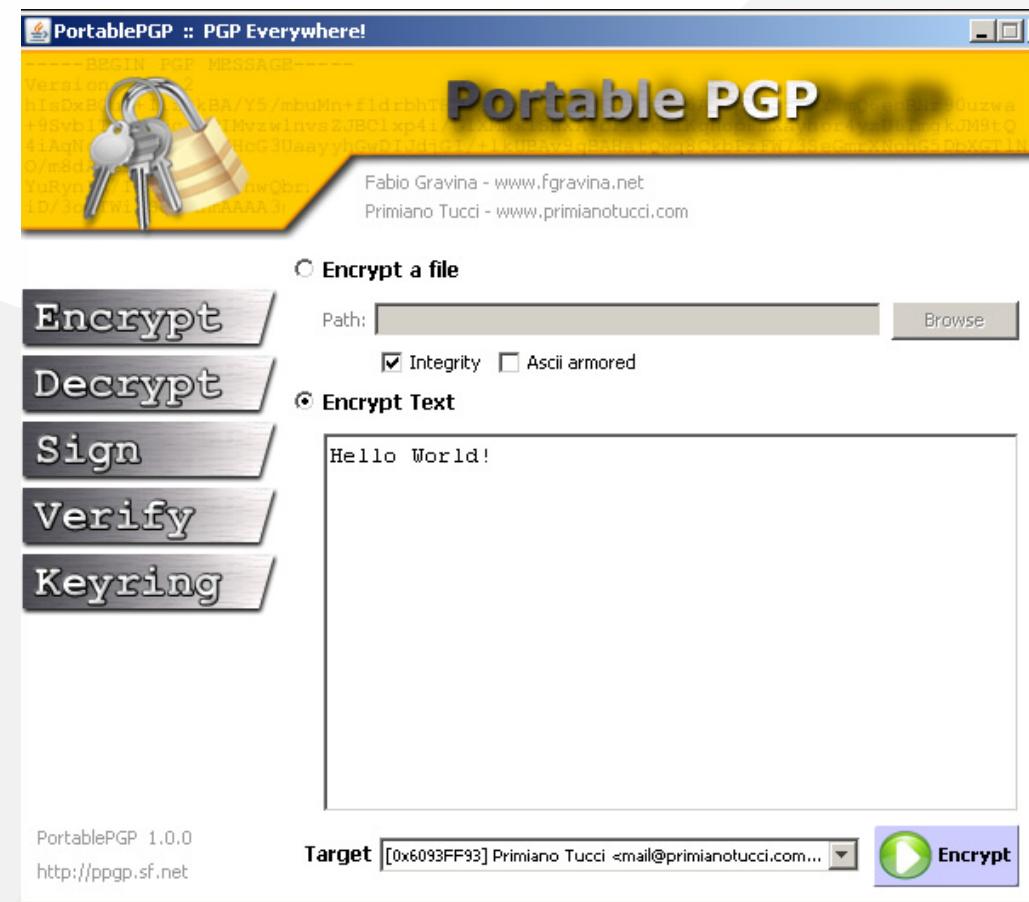


SlavaSoft HashCalc

- SlavaSoft HashCalc - Hash, CRC, and HMAC Calculator



- Portable PGP uses for the generation of PGP keys to transfer files securely via e-mail or other channels. You can encrypt or sign your documents with this tool then the receiver can open or verify your e-mail.
- <https://ppgp.sourceforge.net/>



Online Programming Environments

- Hackerrank
 - <https://www.hackerrank.com/>
- CS50 Sandbox
 - <https://sandbox.cs50.io/>
- Programiz C Online Complier
 - [Online C Compiler](#)

End – Of – Week – 2

