**Bazel**

Bazel is an open-source build and test tool.

It uses a human-readable, high-level build language.

Bazel supports projects in multiple languages and builds outputs for multiple platforms.

Bazel supports large codebases across multiple repositories, and large numbers of users.

Bazel is fast and reliable.

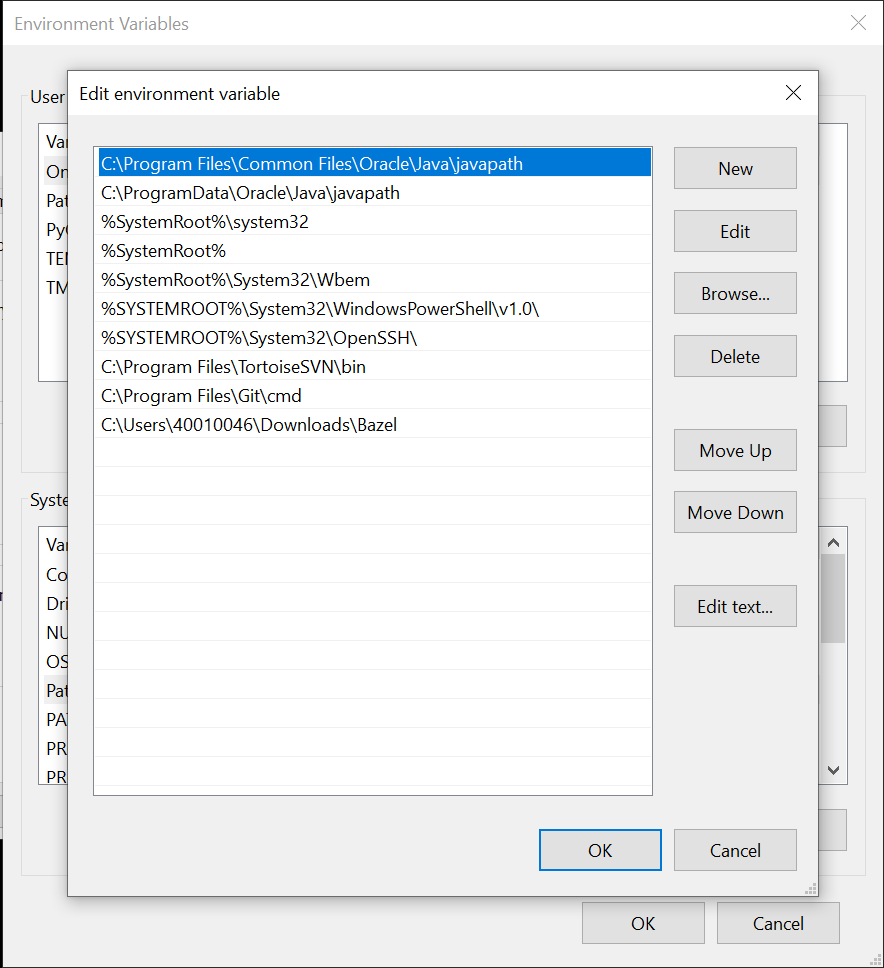
Bazel is multi-platform.

Bazel is extensible to multiple languages.

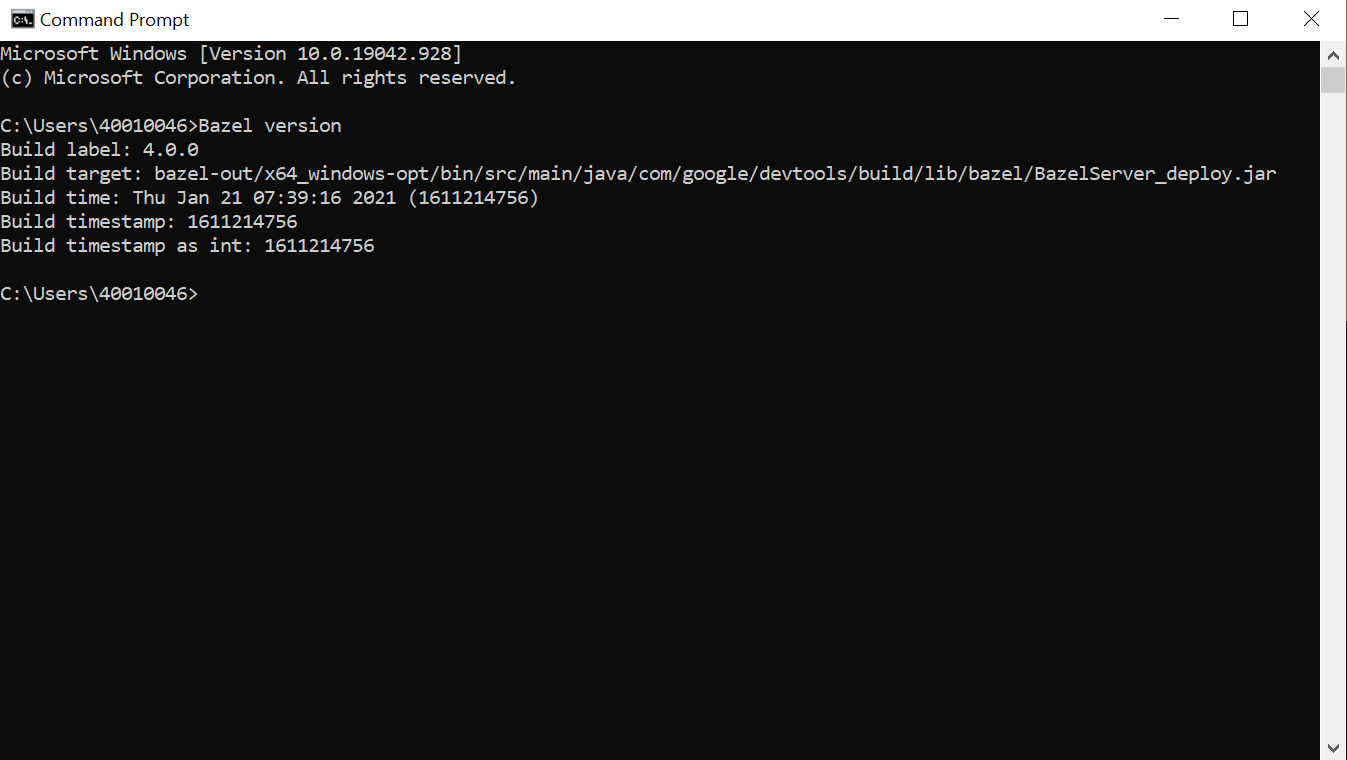
**Installation**

* Install the prerequisites : Visual C++ Redistributable for Visual Studio 2015
* Download the Bazel binary from GitHub.
* Set up your environment

set PATH=%PATH%;<path to the Bazel binary>



* To check the installation is correct, try to run: bazel version

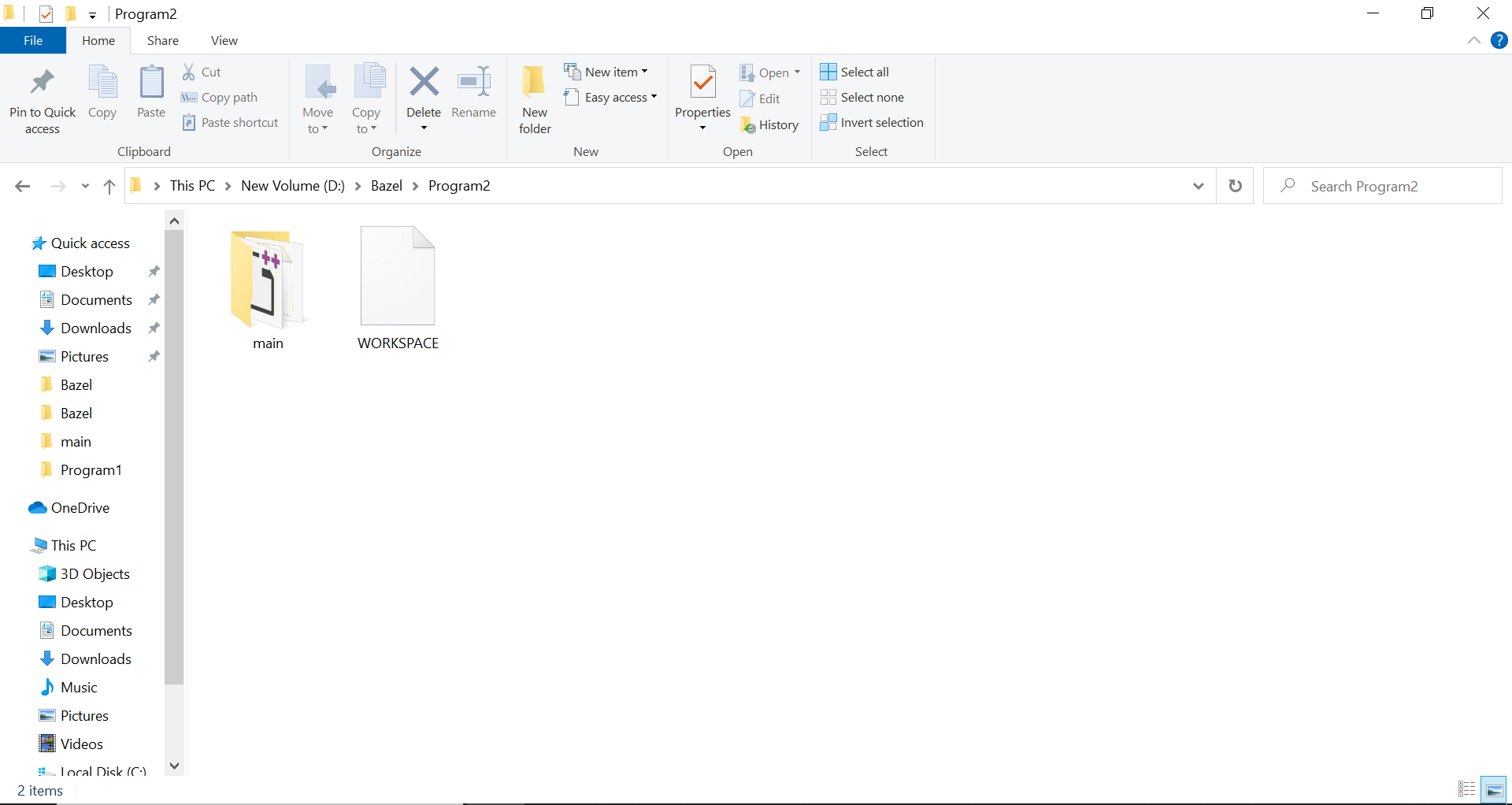


**Compilers & Language runtimes**

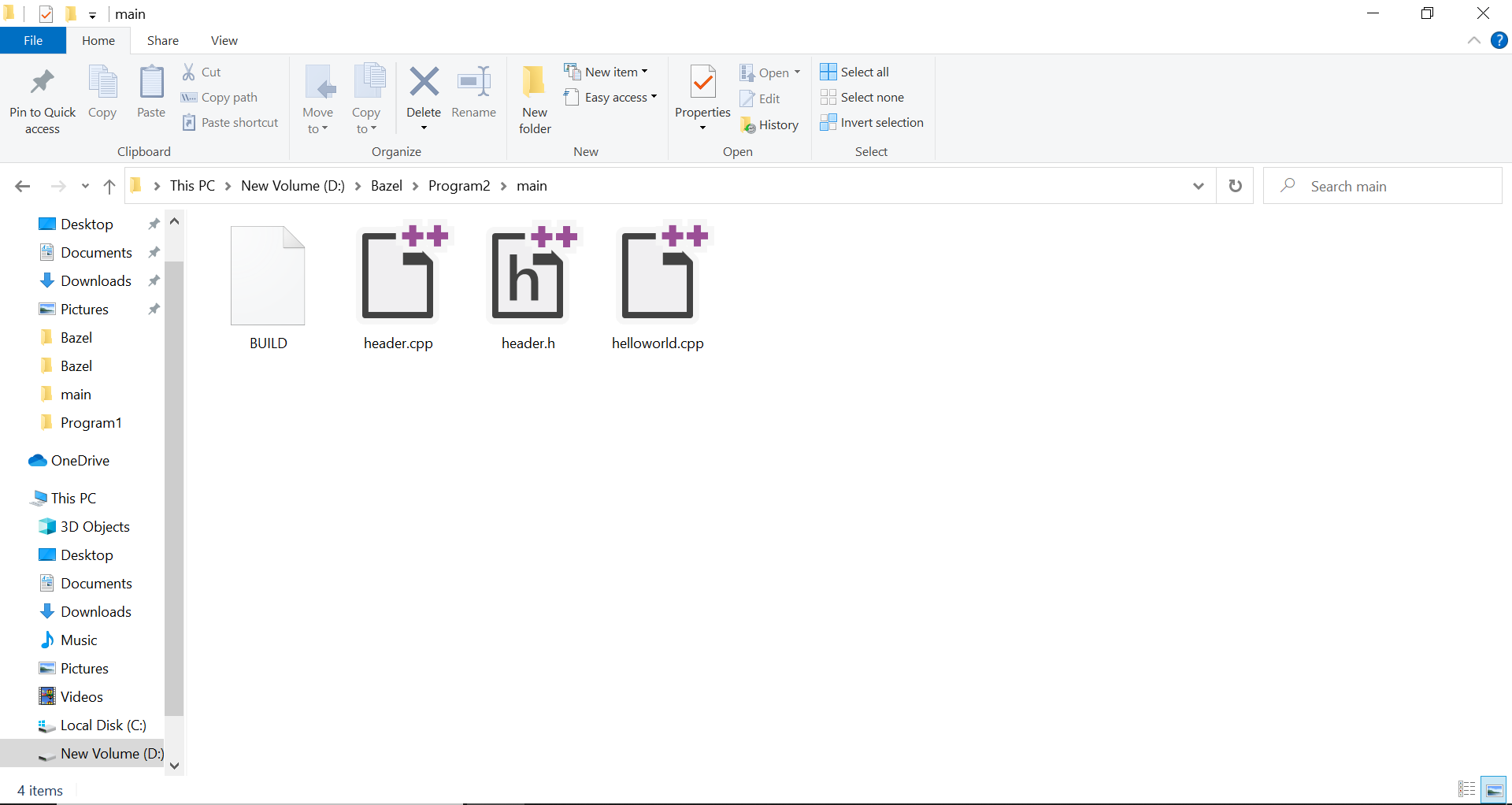
* MSYS2 x86\_64 contains Bash and common Unix tools.
* Build Tools for Visual Studio 2019. 2015 or newer can be used.
* Java SE Development Kit 11 (JDK) for Windows x64. 8, 9 or 10 can be used.
* Python 3.6 for Windows x86-64. 2.7 or newer can be used.

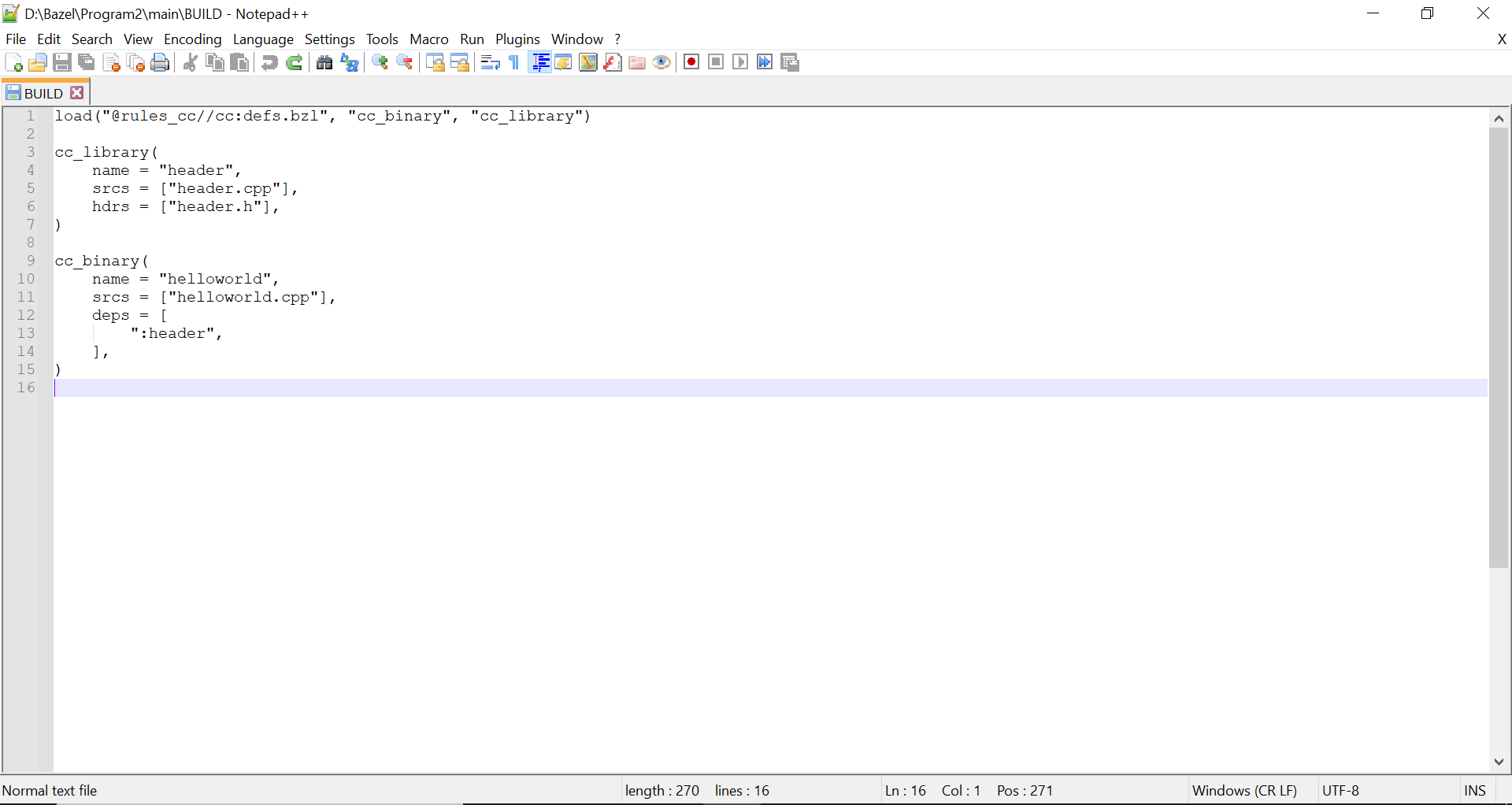
**CPP Program & Folder Structure**

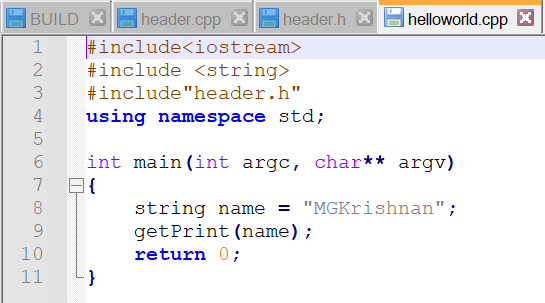
The code segments are placed inside the directory ‘main’. ‘WORKSPACE’ file allows for build process to take place. It may be empty, or may contain references to external dependencies required to build the outputs.

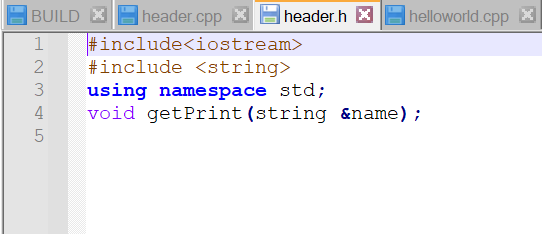


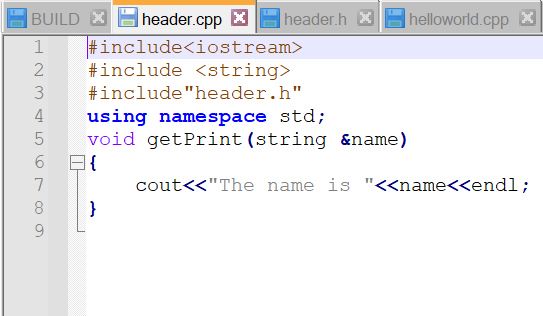
‘BUILD’ file contains several different types of instructions for Bazel. The most important type is the build rule, which tells Bazel how to build the desired outputs, such as executable binaries or libraries. ‘header’ files provide the required libraries to the CPP program. ‘helloworld.cpp’ has the code segment.





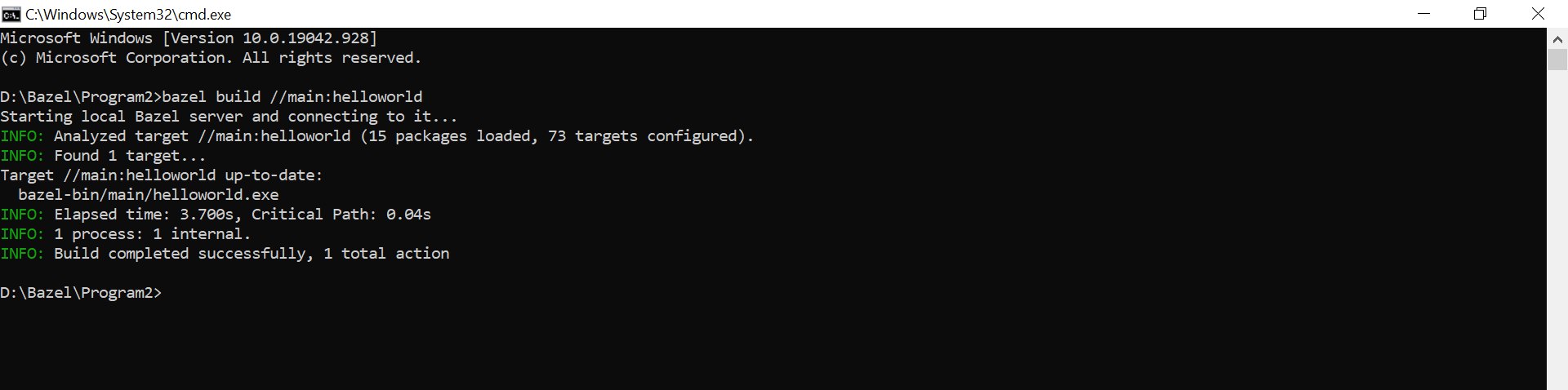




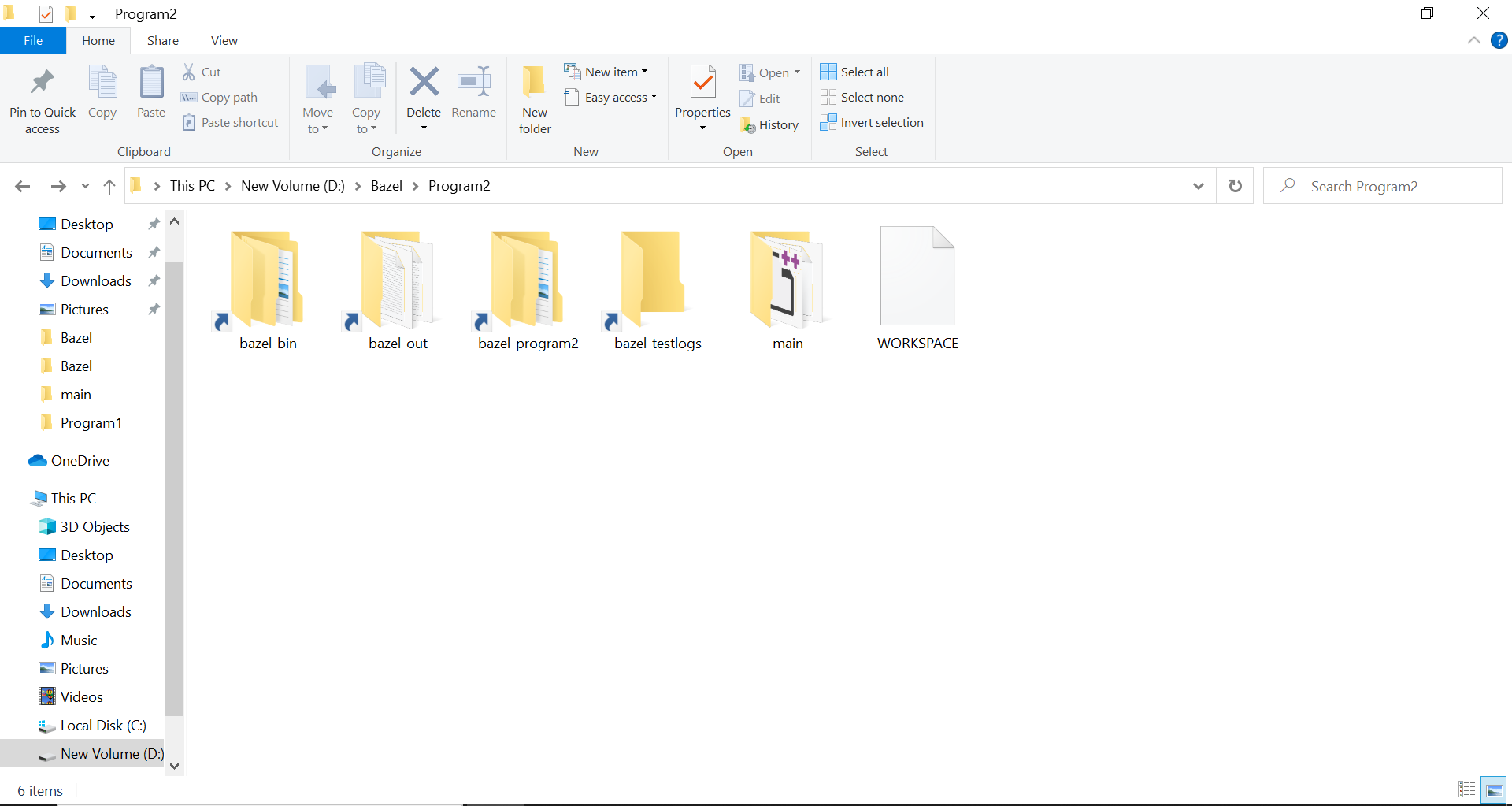


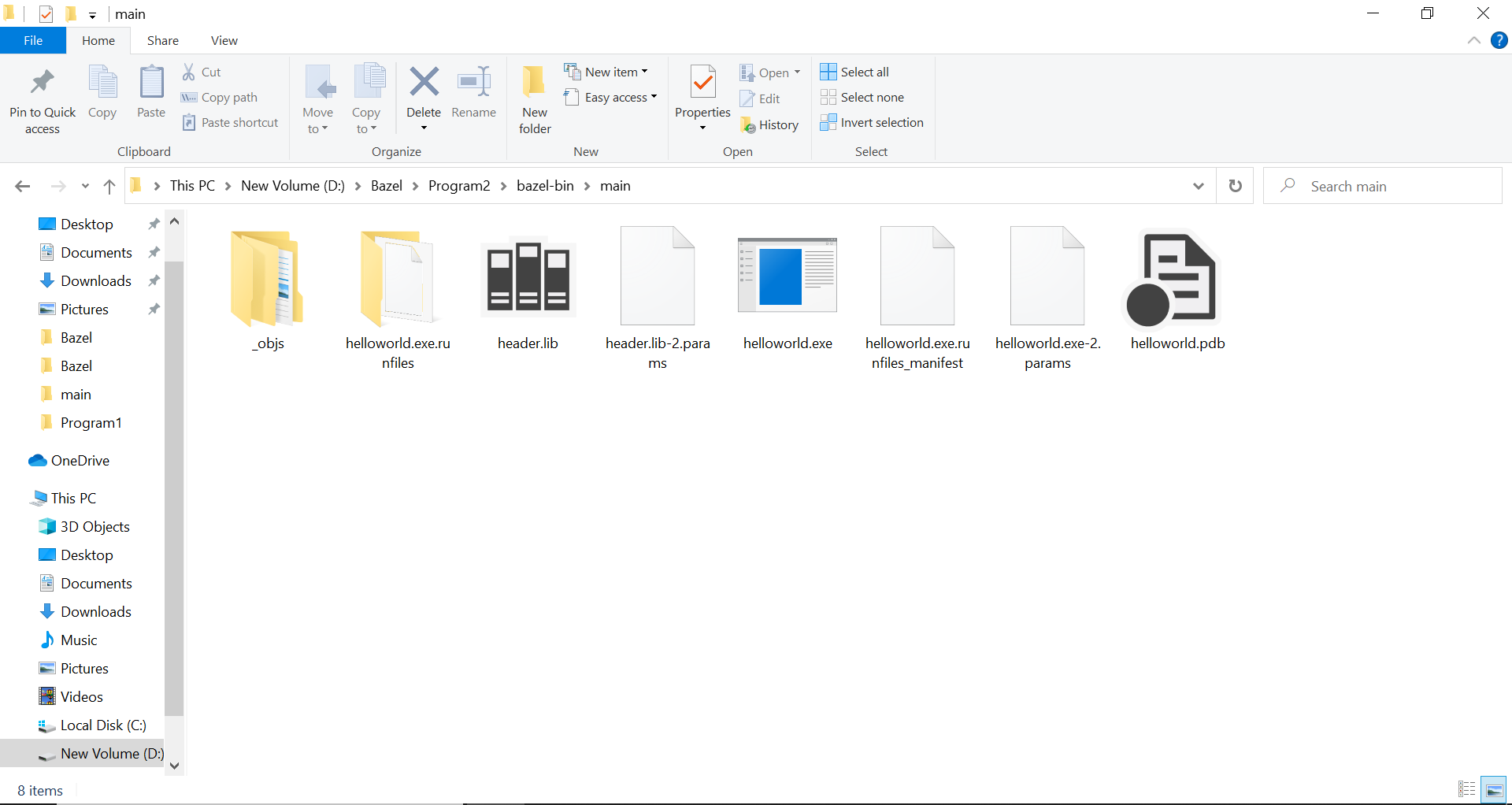
**CPP Build**

In Command Line, type <path>>bazel build //main:helloworld



Once the build process is successful, directories are created in the base location





In command line, execute the .exe file generated by Bazel.

