ENVIRONMENT SETUP

COS20007

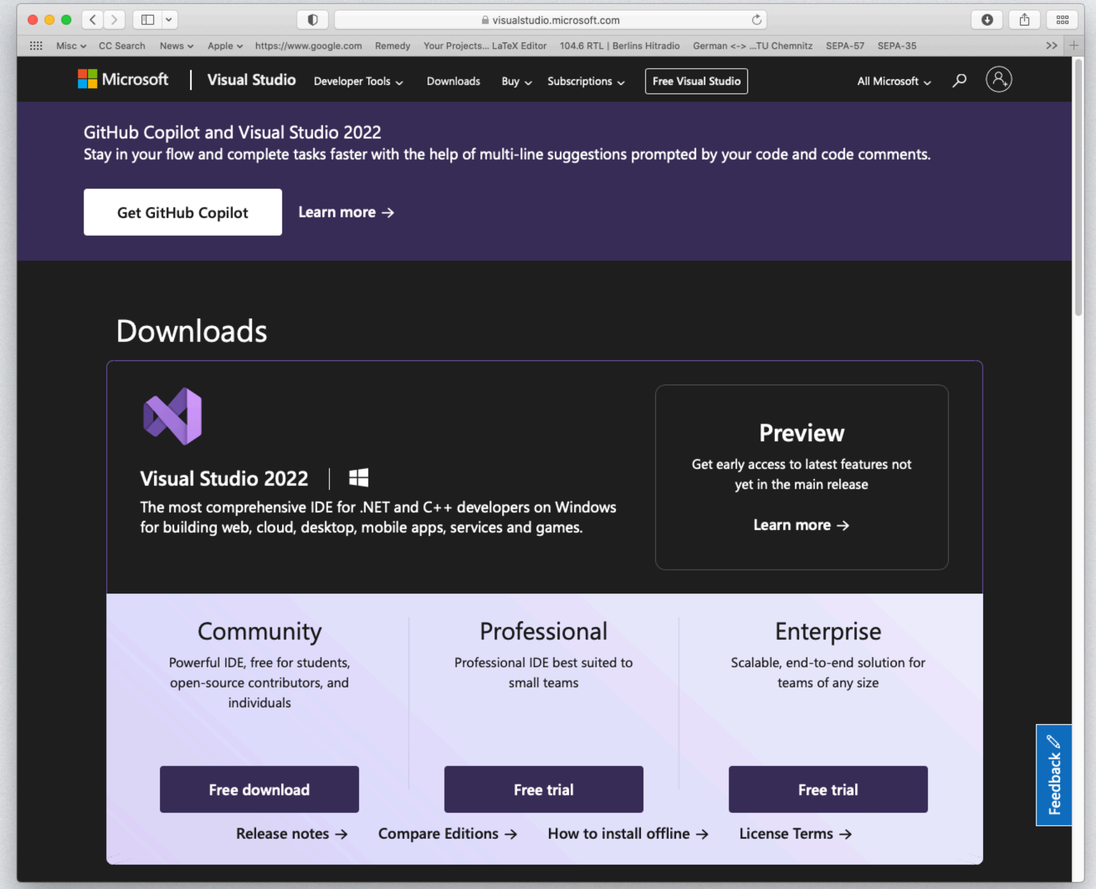
Windows

REQUIRED SOFTWARE PACKAGES

- In COS20007, you need to install and setup the following software packages:
  - Visual Studio 2022 (Community Edition) and .NET Desktop Development
  - MSYS2 (set of Unix-style command terminals and tools)
  - Git
  - SplashKit

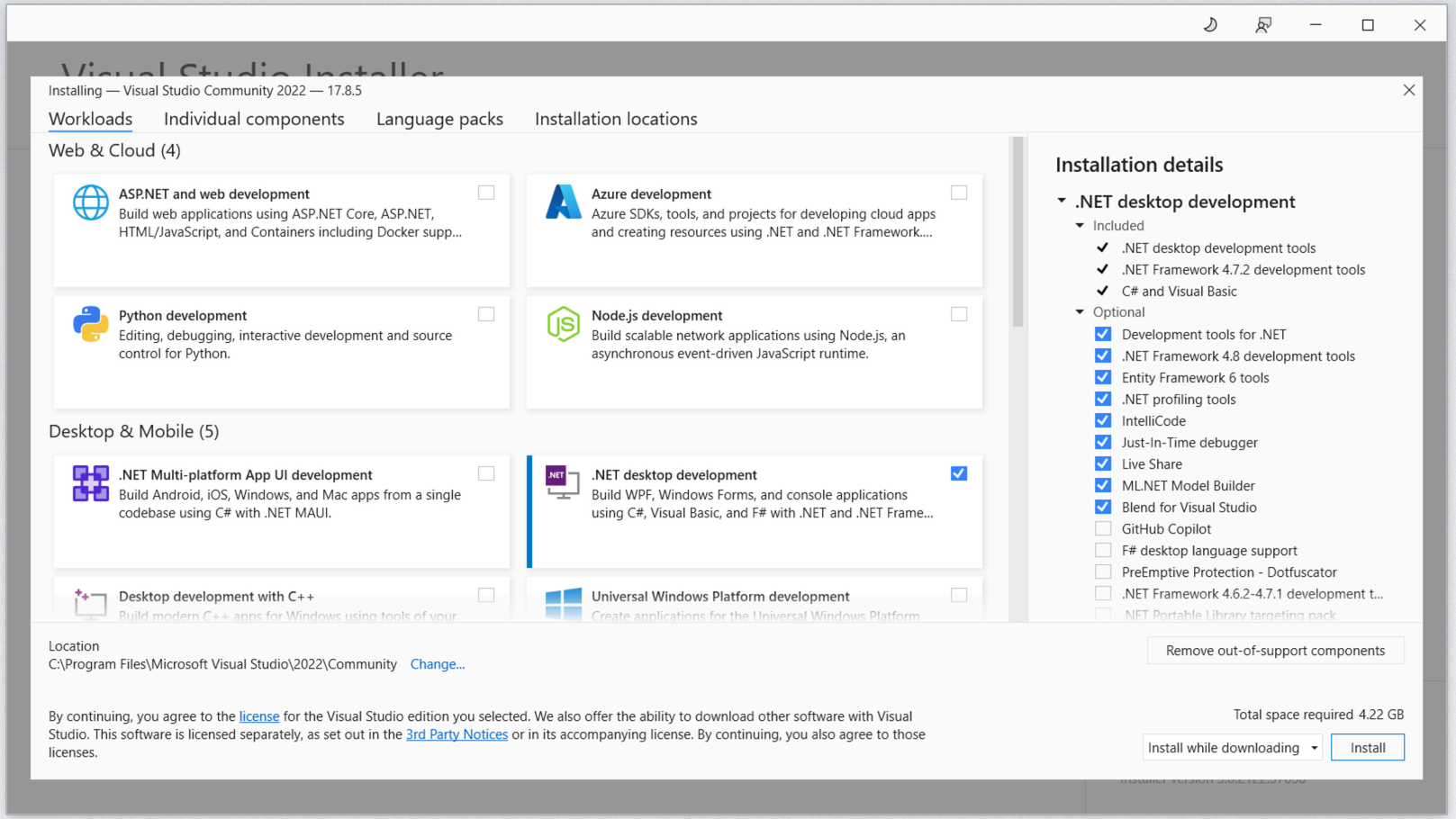
VISUAL STUDIO

VISUALSTUDIO.MICROSOFT.COM



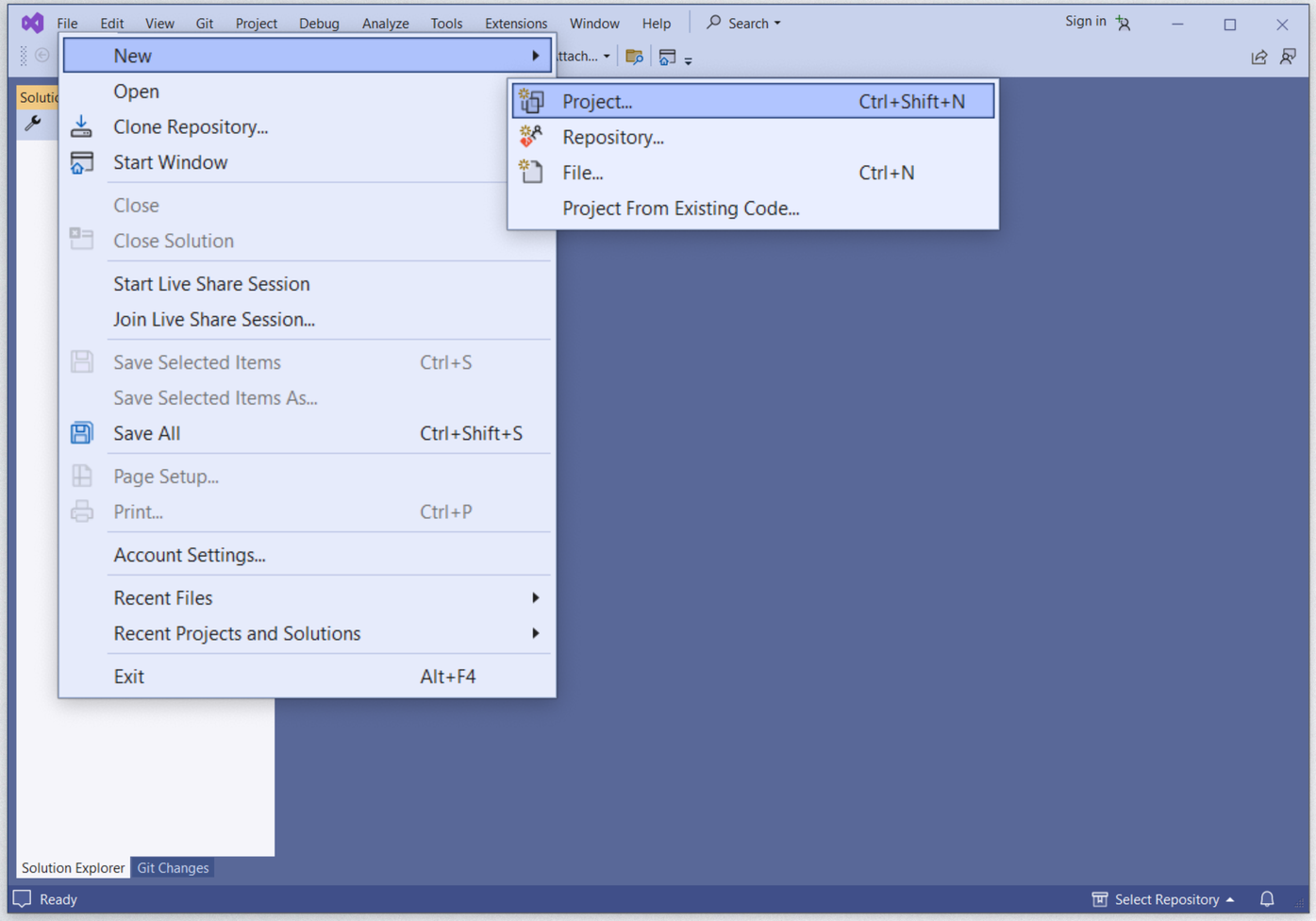
- The Community edition of Visual Studio 2022 is available at visualstudio.microsoft.com.
- Run VisualStudioSetup located in your Downloads folder.

VISUAL STUDIO INSTALLER



- VisualStudioSetup runs Visual Studio Installer and directs you to the Workloads.
- Select .NET desktop development, accept the default-selected features, and press Install to set up Visual Studio 2022 Community for .NET desktop development.

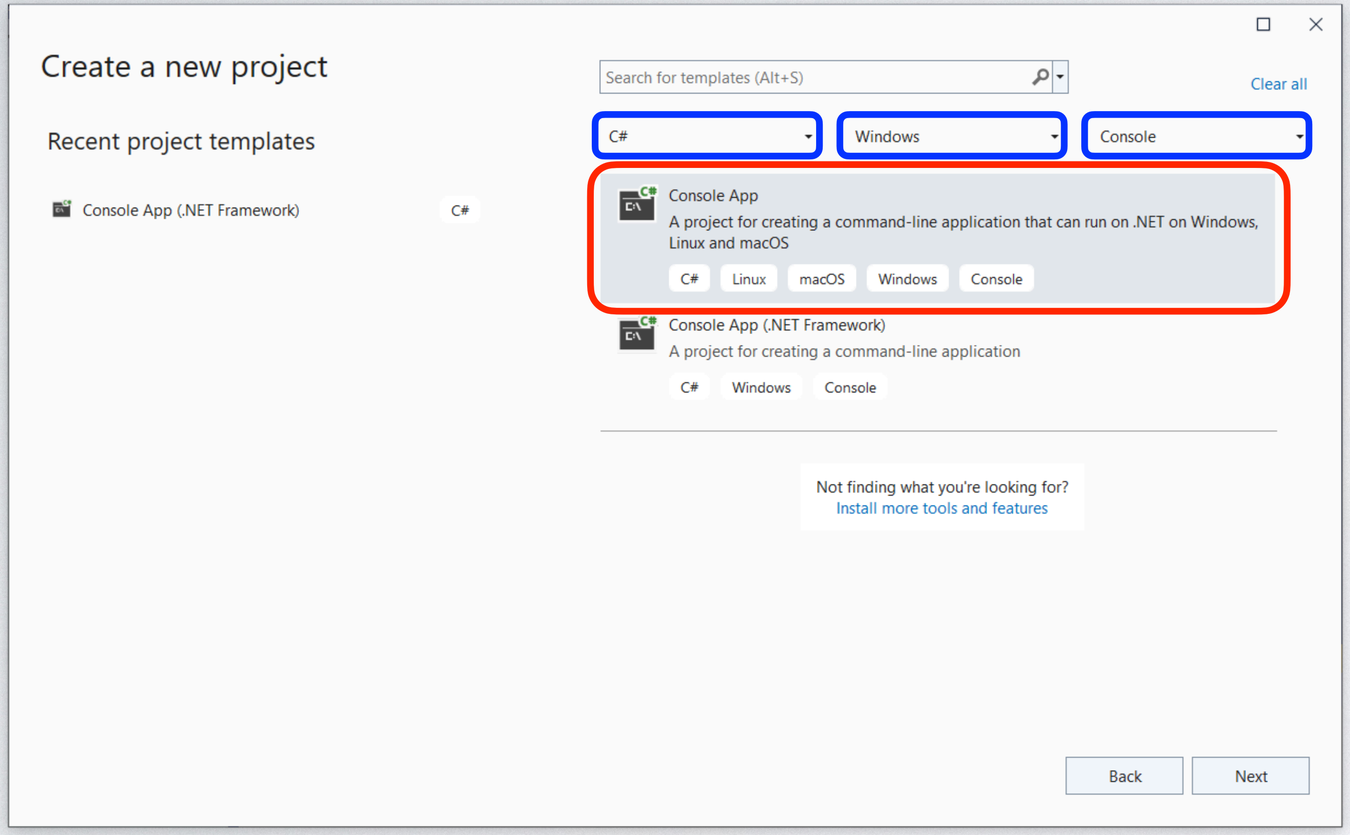
CREATE A NEW PROJECT



1. Open Visual Studio and continue without code.
2. Select File/New
3. Finally select Project

Please note that your mouse pointer have to hover over the menus to remain visible.

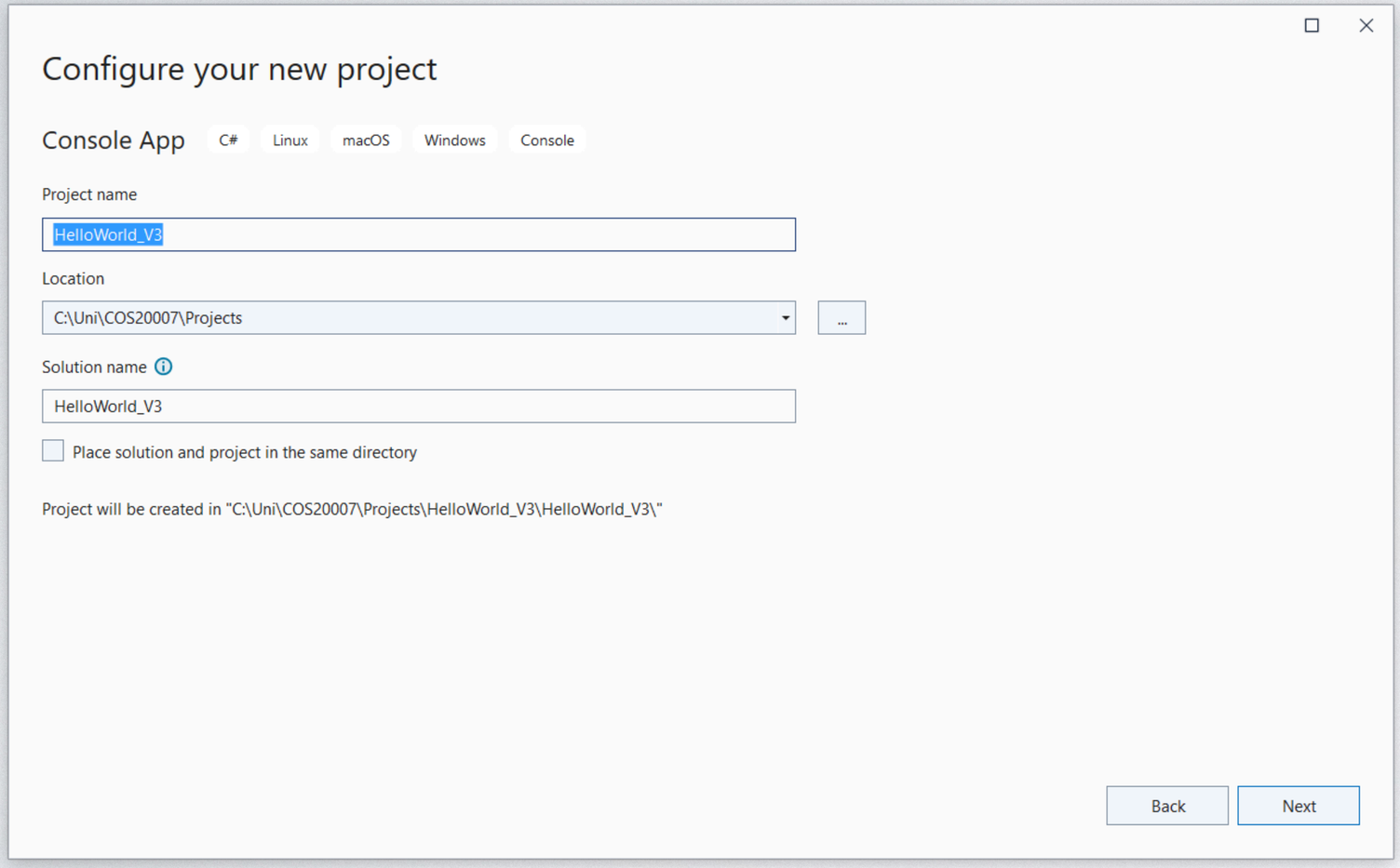
SET UP A CONSOLE APP



- Visual Studio offers a multitude of project types.
- In COS20007, we are working only with one project type: Console App.
- You can use the dropdown list to restrict the project type view to just the two shown on the left. Select
  - C# for Language
  - Windows for Platform
  - Console for Project Type

Please note that the project type Console App is a project template for .NET applications that can run on any CPU supporting Windows, Linux and MacOS.

SETTING UP PROJECT NAME AND LOCATION

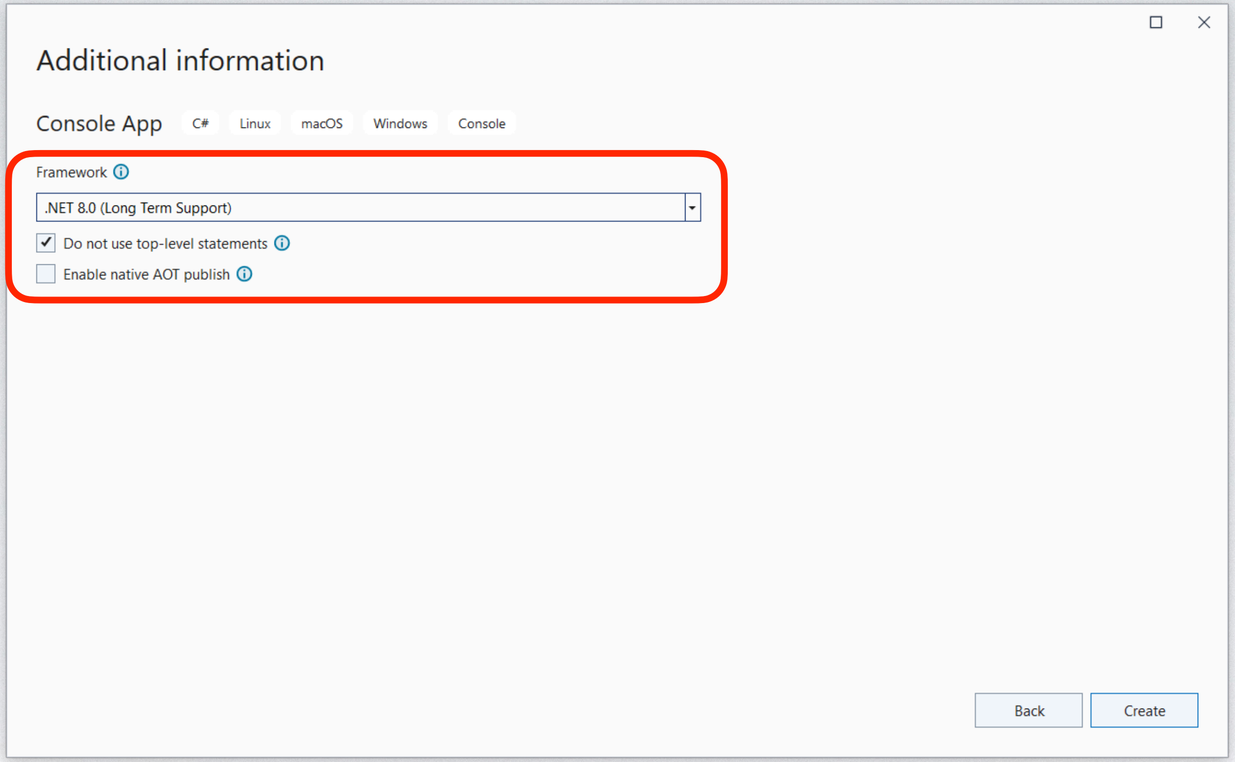


1. Select the project location. Click on … box. Never accept the Visual Studio default. Projects in this presentation a stored in the folder C:\Uni\COS20007\Projects. Choose a folder that you can easily find again. You may even consider a Cloud resource.

2. Give the project a name. Here the name is HelloWorld_V3. Any meaningful name will do.

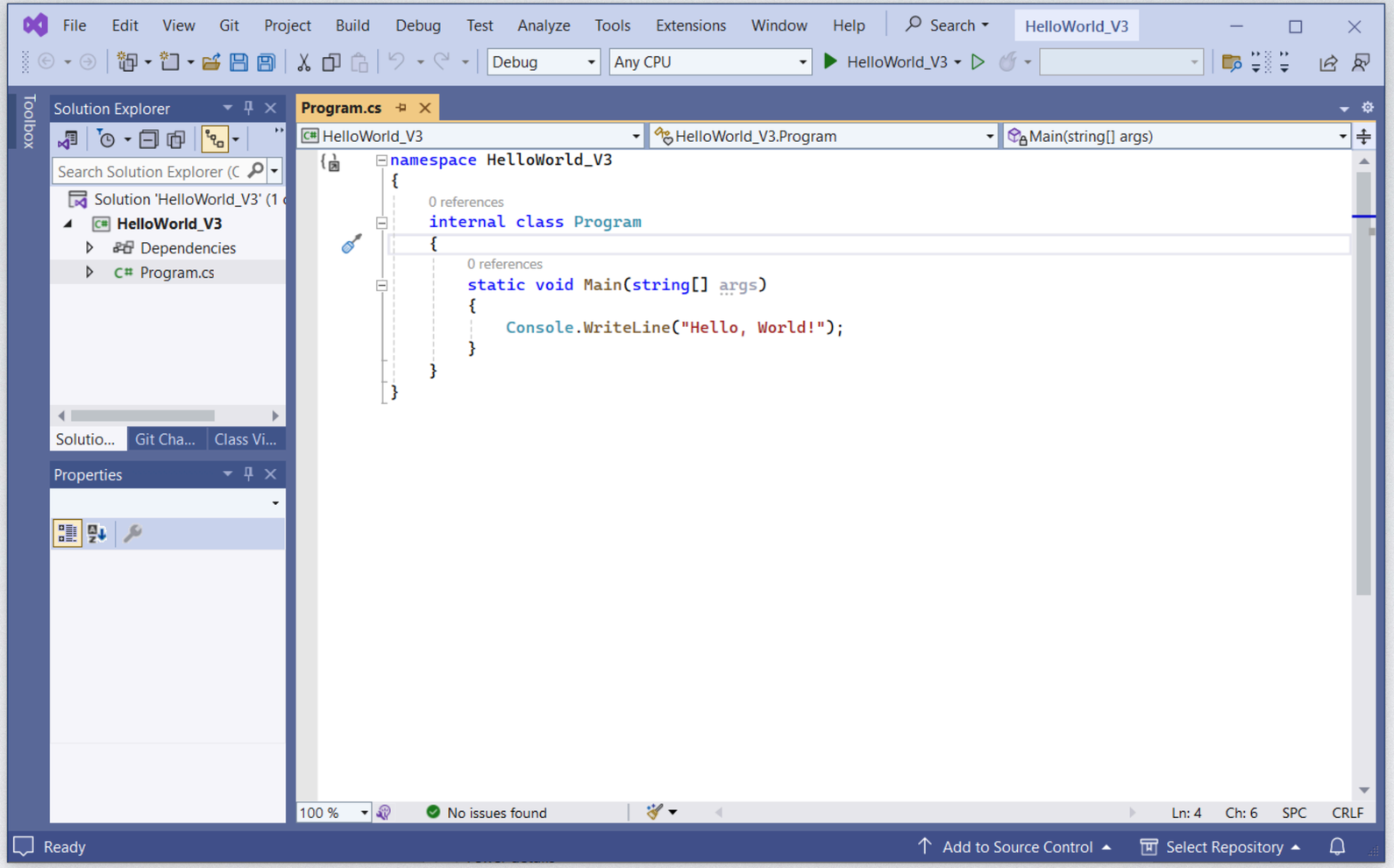
3. You can change the solution name also. This is useful when a solution contains several projects. Here, the solution name coincides with the project name as the solution only hosts one project.

SELECT FRAMEWORK



- When creating a Console App for .NET, we need to select the .NET framework that we want to work with.
- You can safely choose the latest framework (.NET 8.0 LTS). At a minimum, COS20007 requires .NET 6.0.
- Check "Do not use top-level statements." Visual Studio creates a "Hello, World!" program this way.

GENERATED PROJECT

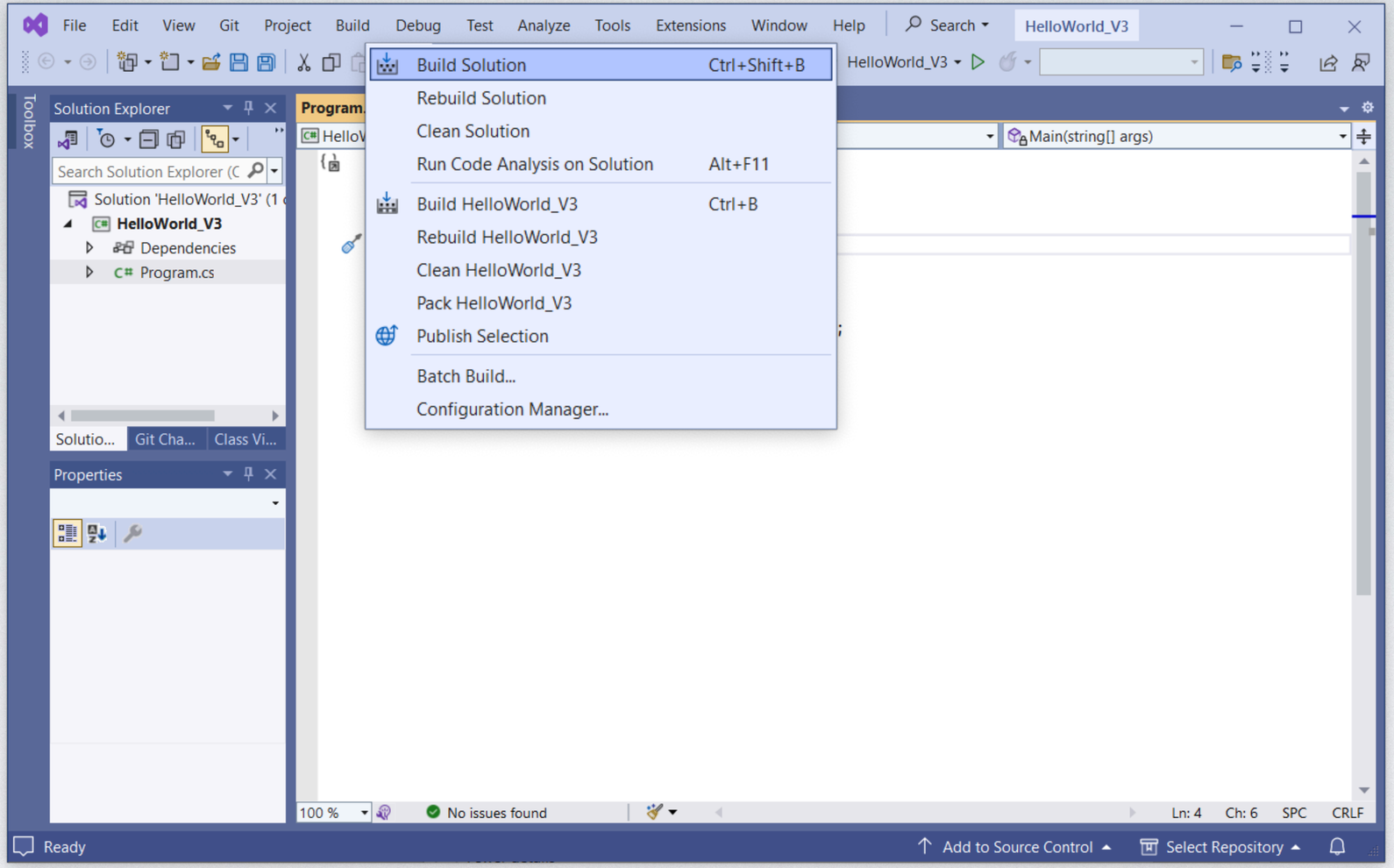


- The generated project template is a fully functional C# console application.

HELLO WORLD IN DETAIL

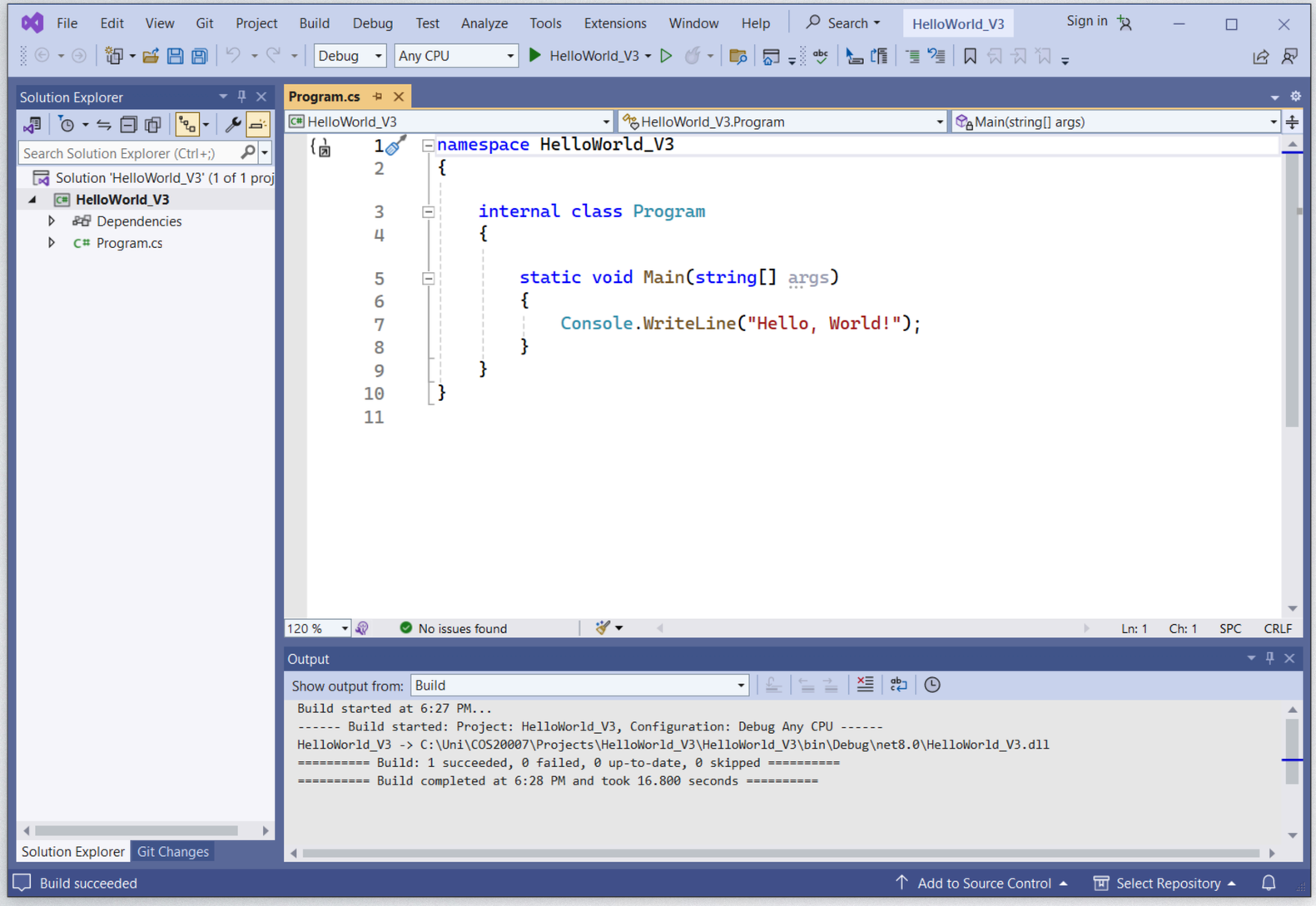
Language Construct	Interpretation
namespace HelloWorld_V3 { … }	A namespace declares a scope C# that can contain a set of related objects which should be logically grouped. Here, HelloWorld_V3 is the namespace for all code elements of our console application.
internal class Program { … }	C# is a class-based language. Every program must define at least one class that contains the main entry point. Classes in C# can be defined to restrict access to clients. Here, the class Program is marked internal, which means that members are only accessible with the same assembly (i.e., the same unit of deployment).
static void Main(**string**[] args) { … }	The function Main define the main entry point of the program. Every program must have at least a main entry point. The function Main is a *class member*, as it is marked **static**. Class members are shared by all objects of the class. Static functions can be called even if no objects of the class exist.
Console.WriteLine("Hello, World!");	Console is a class defined in namespace System (Here, the object Console is implicitly resolved to be a member of namespace System. Hence, there is no using System directive at the start.) The class Console represents the standard input, output, and error streams. The method WriteLine outputs the given string to the console and adds a system-specific newline character at the end.

BUILD PROJECT



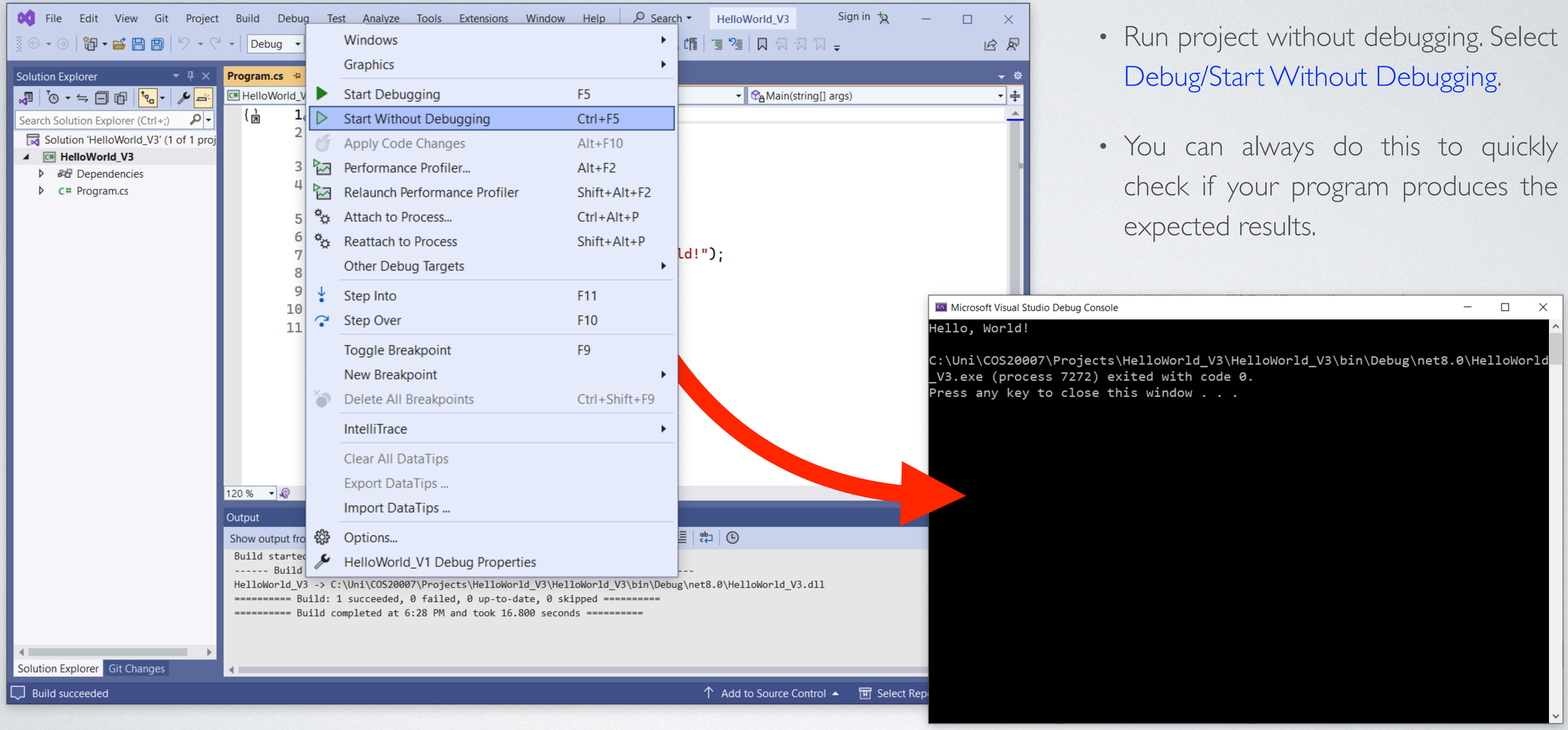
- Select Build/Build Solution to compile the code and build the executable.

BUILD RESULT



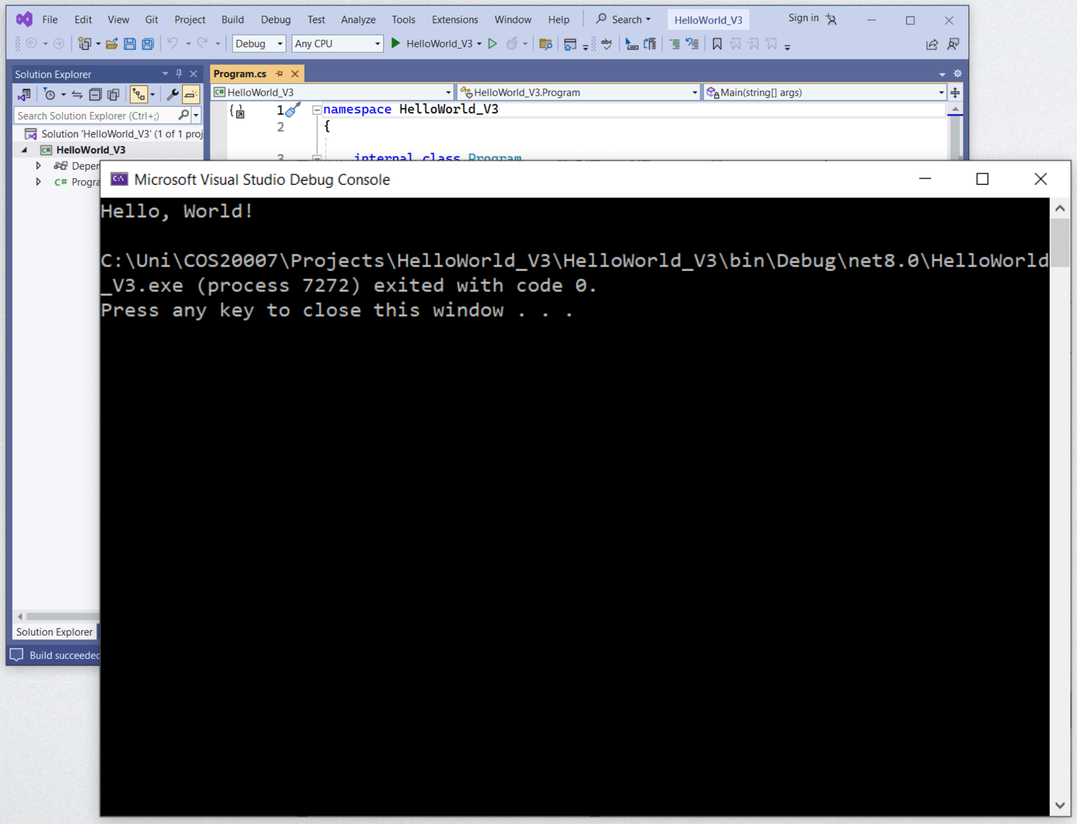
- The build project may take some time, but is should succeed.

TEST EXECUTABLE



- Run project without debugging. Select Debug/Start Without Debugging.
- You can always do this to quickly check if your program produces the expected results.

VISUAL STUDION INSTALLATION COMPLETE



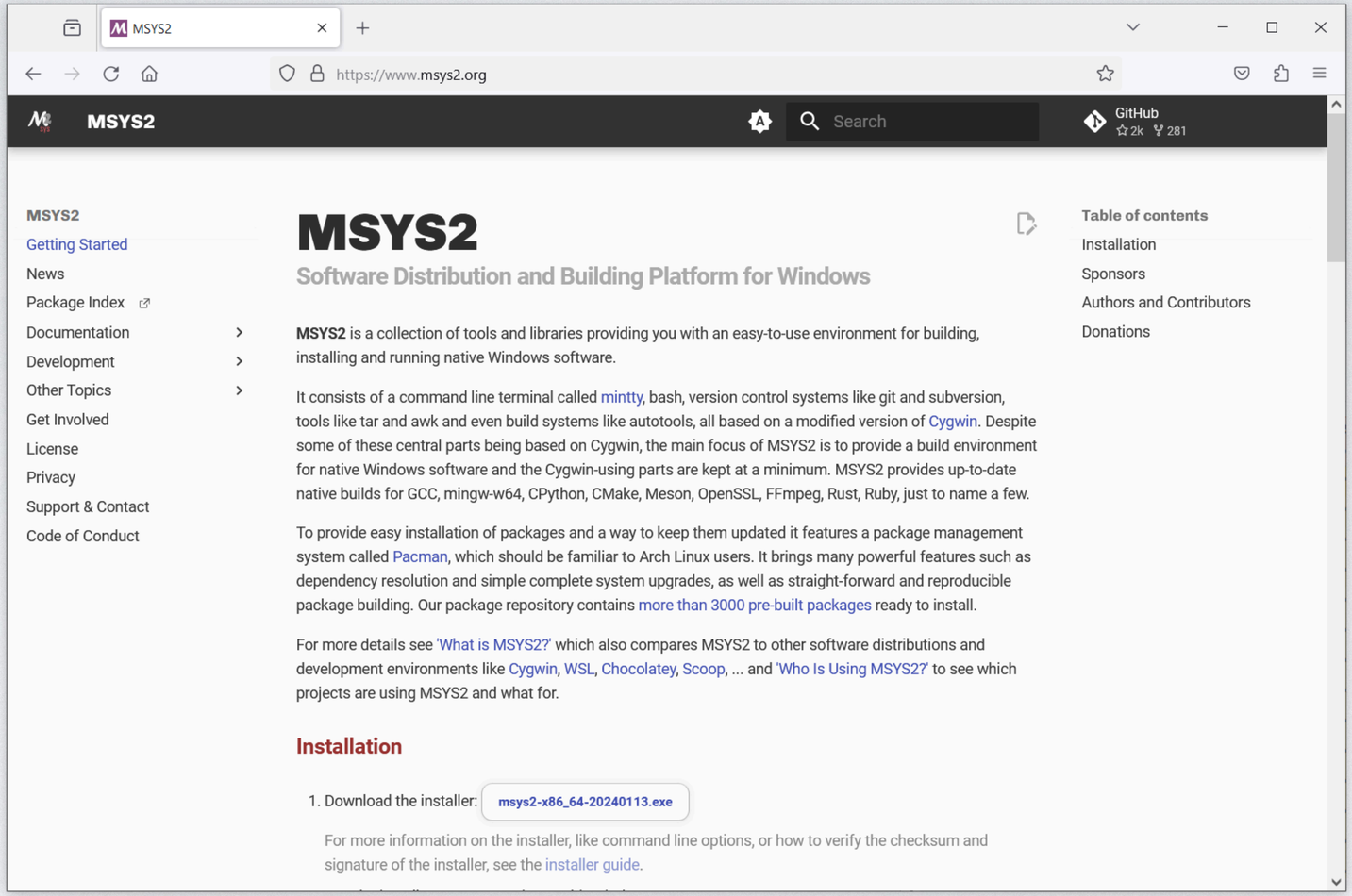
- If you get to this point, your Visual Studio installation is complete and you can start writing programs in C# using the .NET framework.

MSYS2

WHAT IS MSYS2

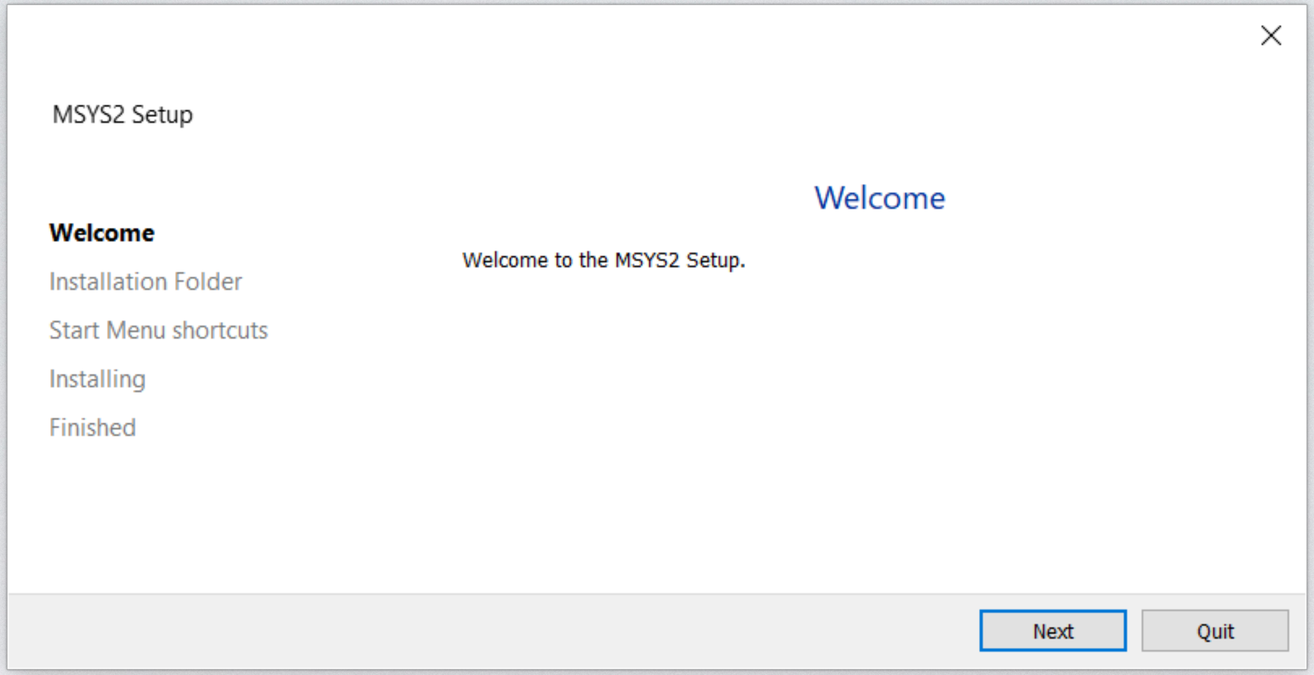
- MSYS2 is a collection of tools and libraries for building, installing, and running Windows applications.
- MSYS2 consists of a command line terminal (actually six different terminal types), bash command shell, git version control, and many more tools.
- MSYS2 terminals provide a Unix-like command line user interface. The four most important commands are *cd* - change directory, *ls* - list directory content, *pwd* - print current working directory, and *mkdir* - create a directory.
- MSYS2 represents Windows drives as paths. For example, C:\ maps to the path /c and C:\Users\Jane\COS20007 maps to /c/Users/Jane/COS20007.
- In the terminal ~ is the home directory for the current user (e.g., /home/Jane is Jane's home directory in the MSYS2 environment). The MSYS2 home directory must not be confused with the user's Windows home directory. They are different. Always store data and programs in a folder that starts with the path name /C/Users/Jane. Replace Jane with your user name.

WWW.MSYS2.ORG

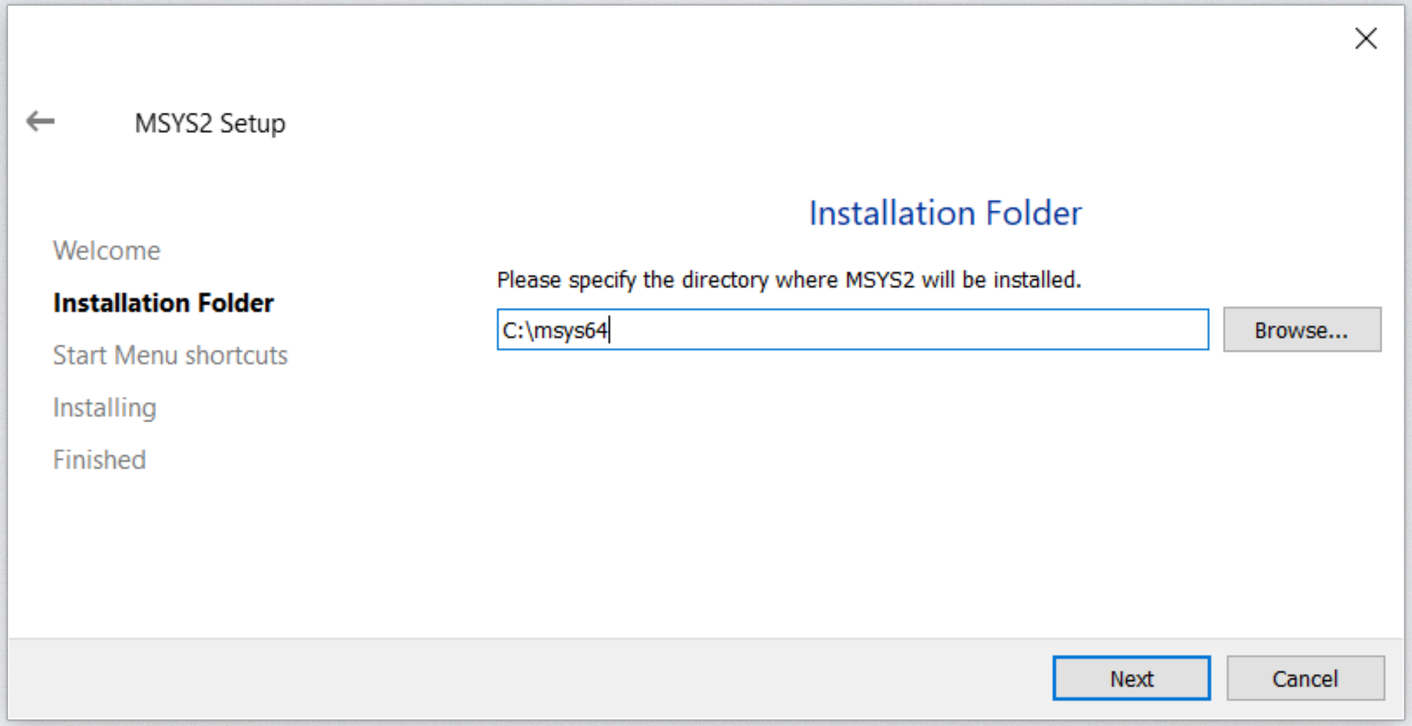


- Download the installer for 64 bit Windows.

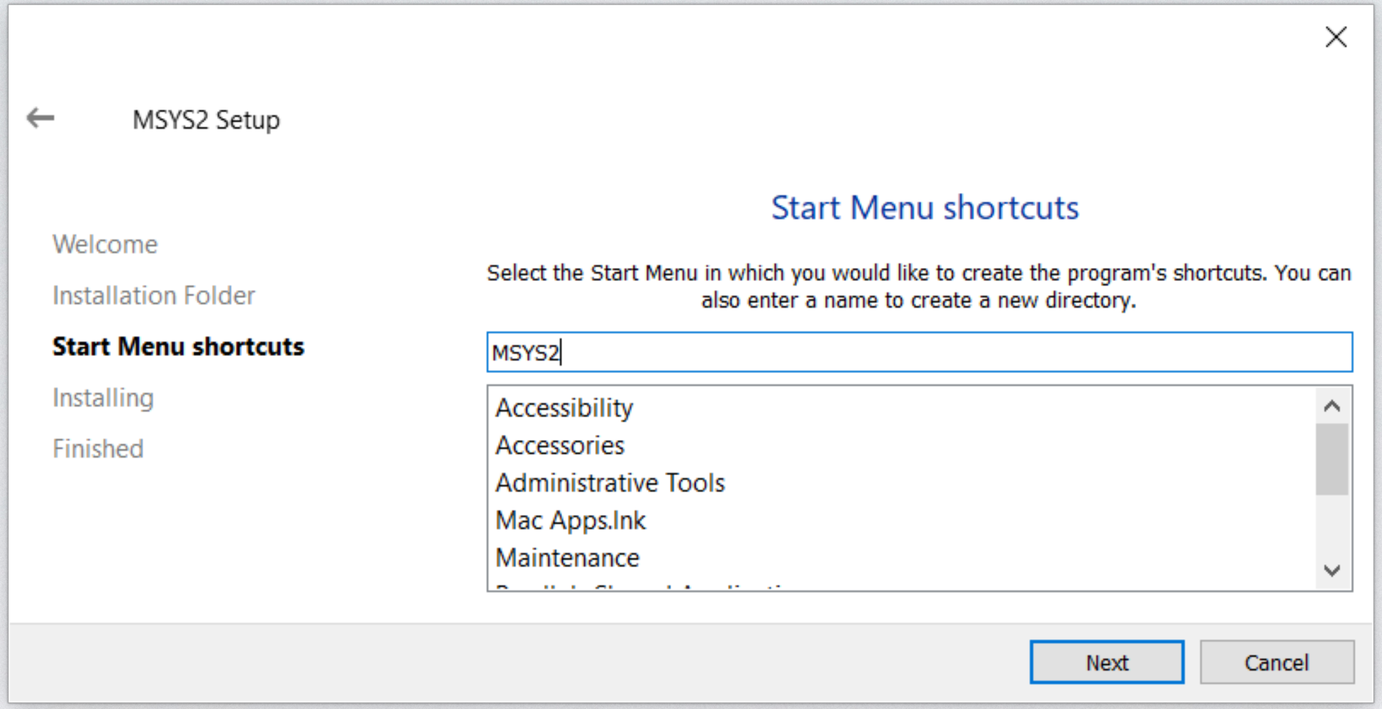
RUNNING THE MSYS2 INSTALLER



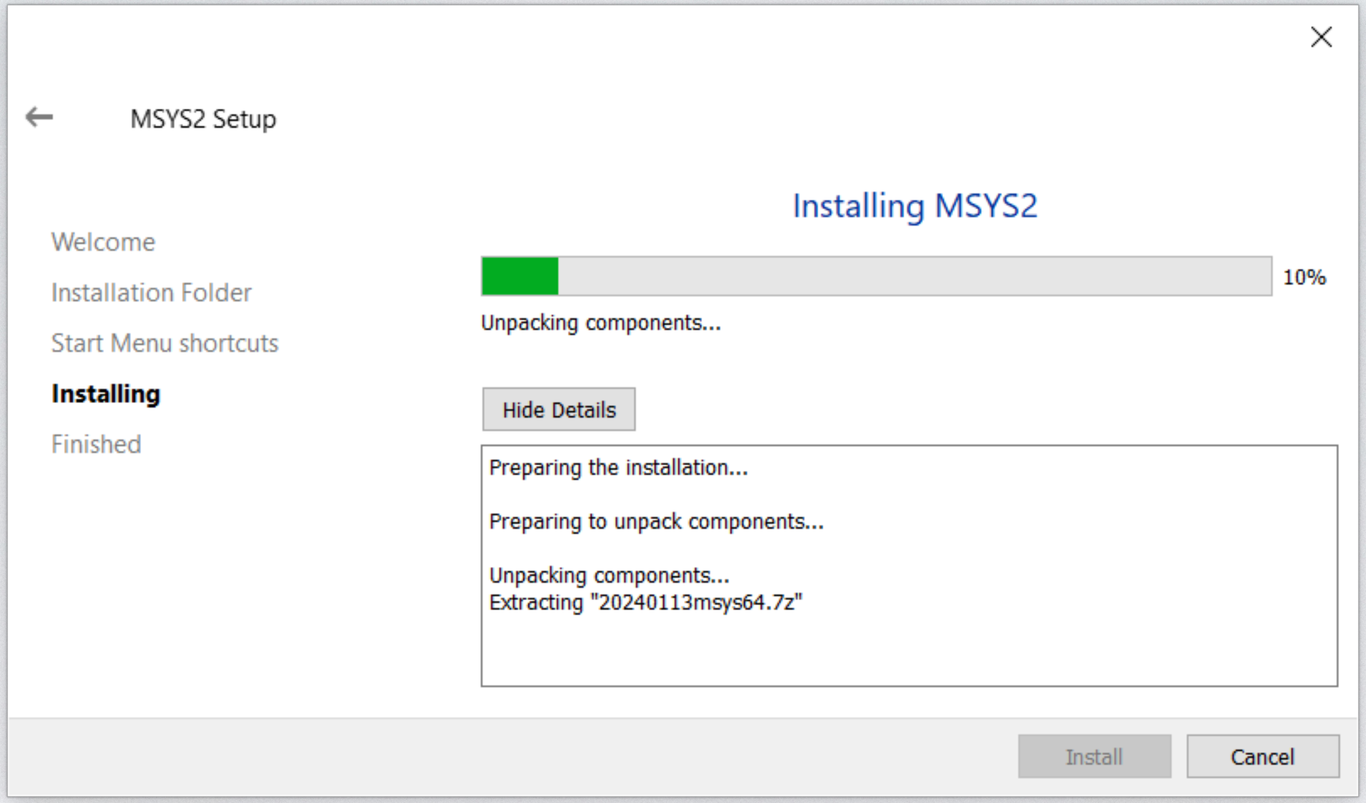
MSYS2 INSTALL FOLDER



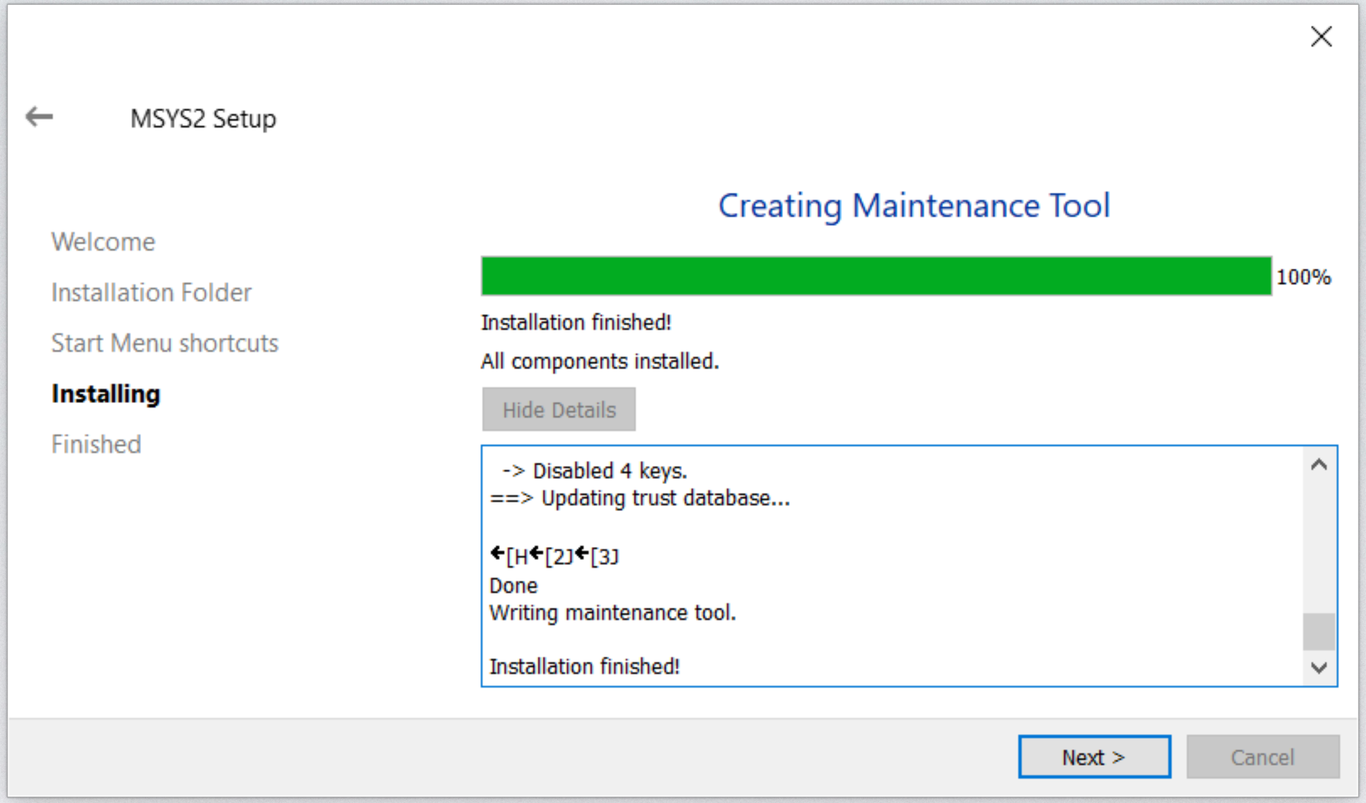
MSYS2 MENU SHORTCUTS



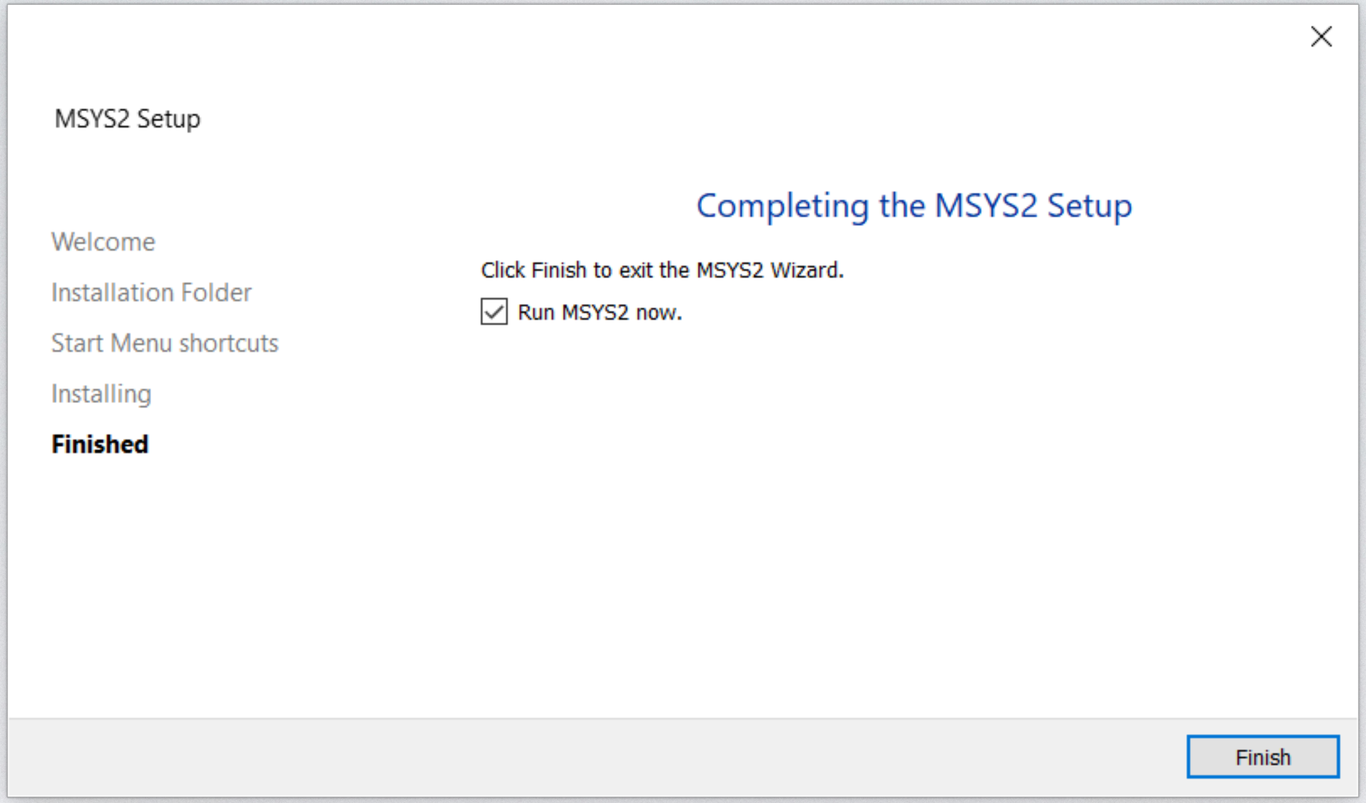
MSYS2 INSTALLATION STARTED



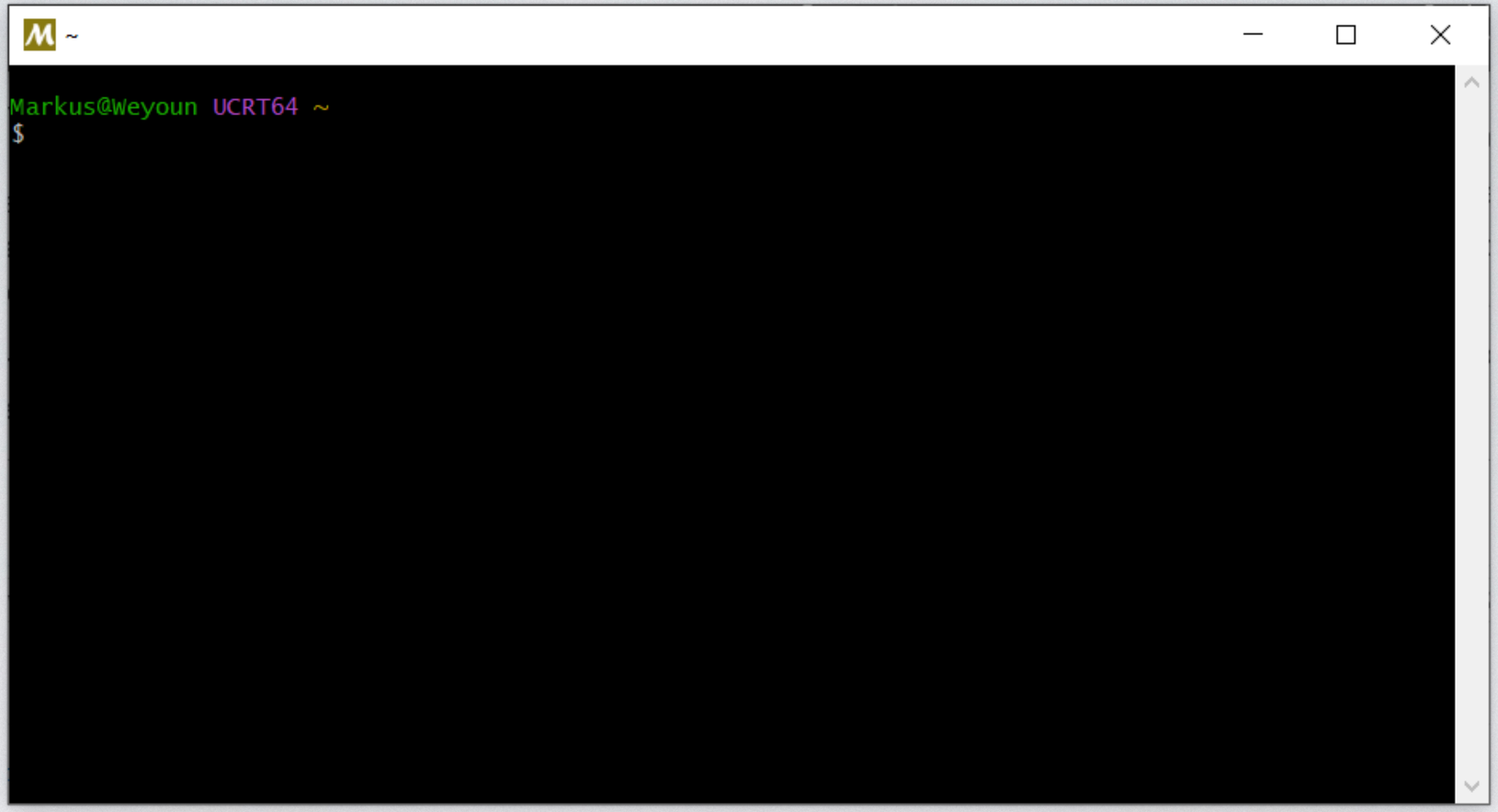
MSYS2 INSTALLATION COMPLETE



MSYS2 RUN AFTER INSTALLATION

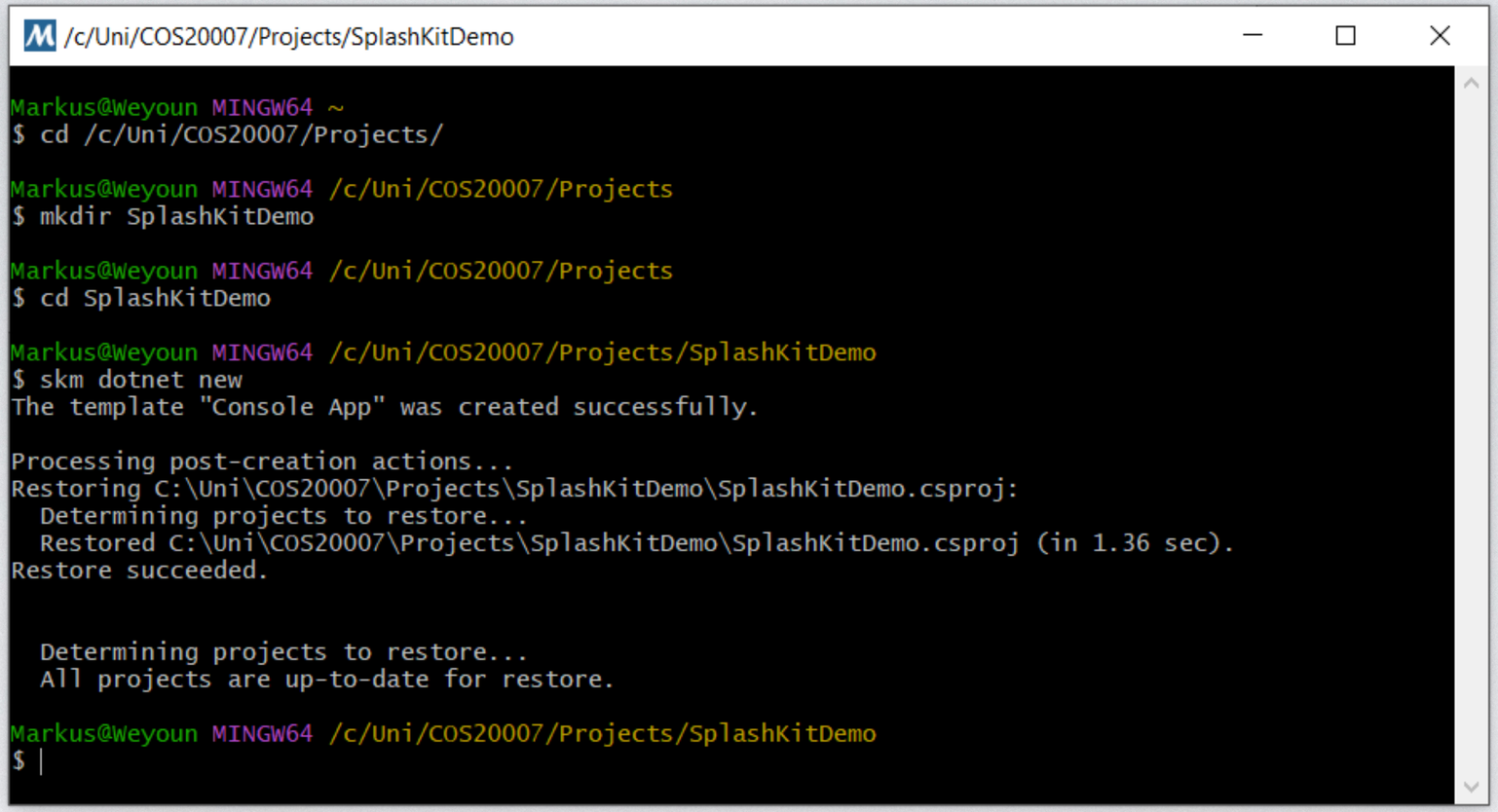


MSYS2 UCRT64 TERMINAL



- UCRT64 is the default terminal. But for development in COS20007, we need the MINGW64 terminal (SplashKit only works in MINGW64).

MSYS2 MINGW64 TERMINAL



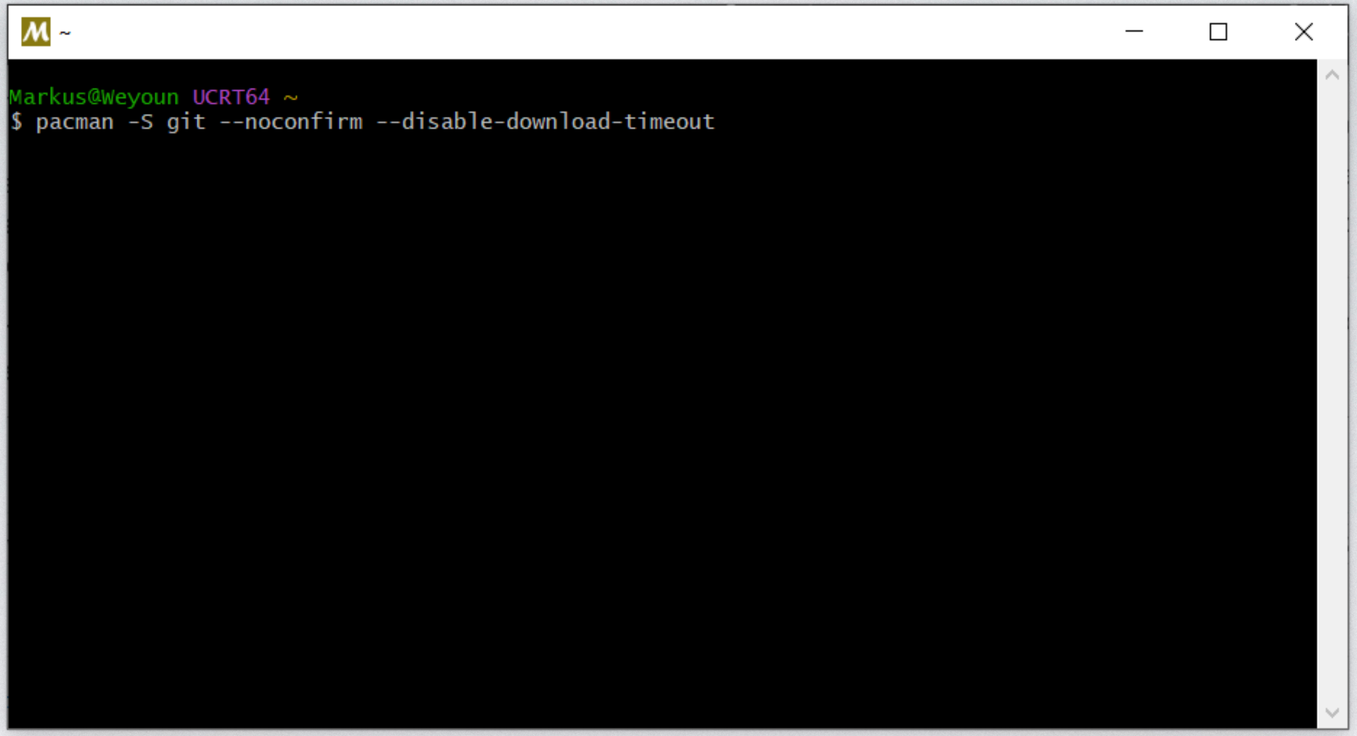
- SplashKit project creation only works in MINGW64.
- To create a SplashKit project, setup a project directory (e.g., SplashKitDemo), move into this directory, and run the command skm dotnet new.

GIT

WHAT IS GIT

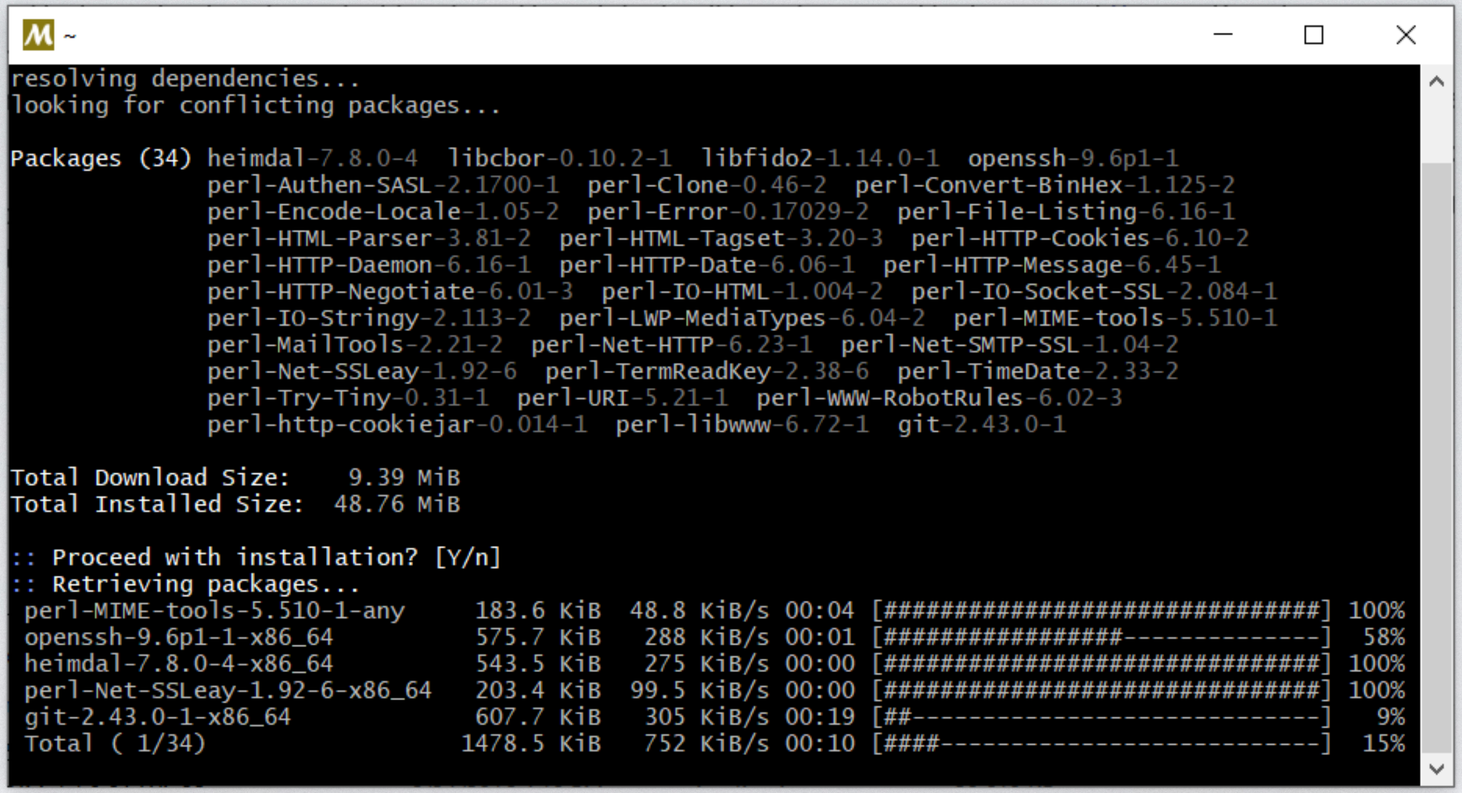
- Git is an open source distributed version control system.
- In COS20007, we use it initially to set up a local installation of SplashKit.
- Git is easy to learn and most modern development environments (e.g., Visual Studio) integrate version control via Git.

INSTALL GIT VIA MSYS2



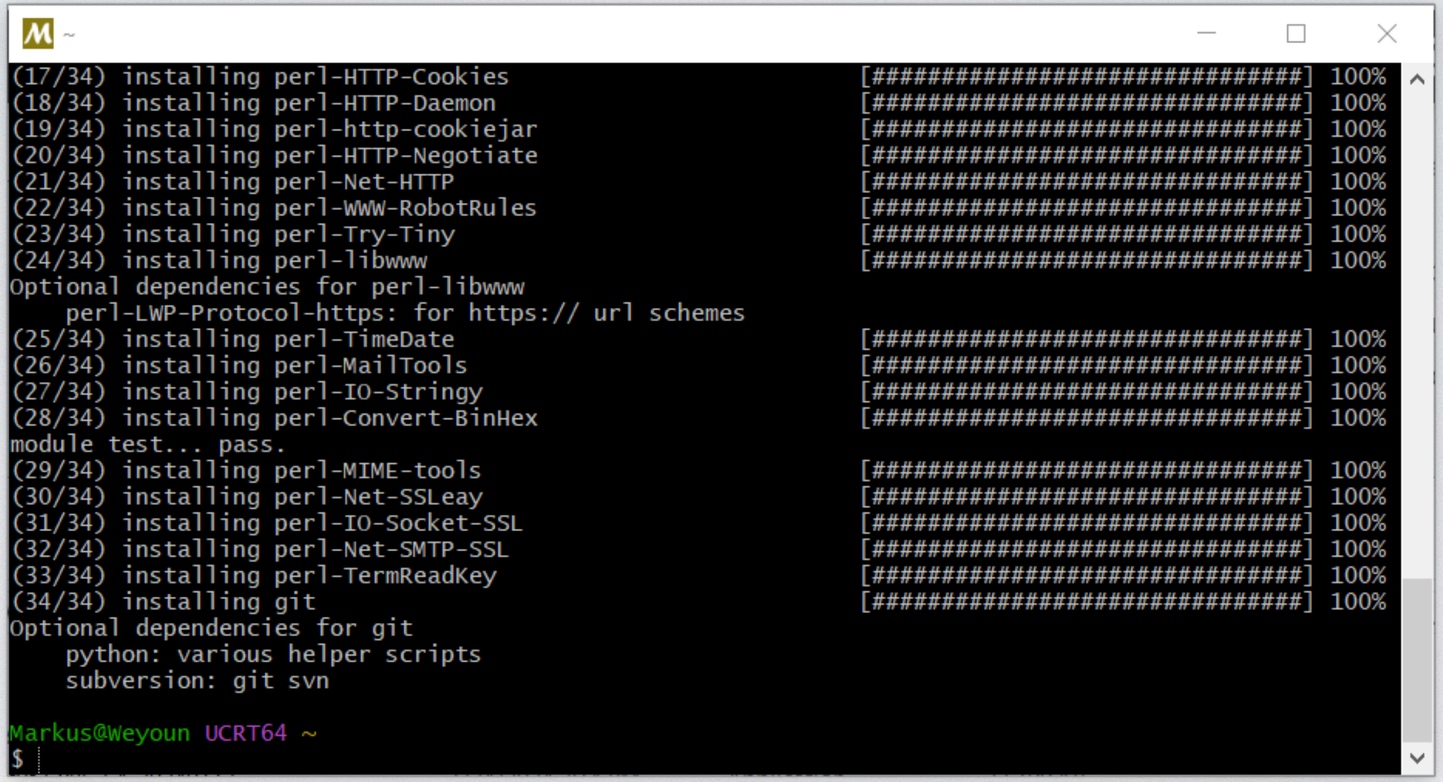
- Install git via MSYS2 and pacman, a library-based package manager.
- Open a MSYS2 terminal (UCRT64 or MINGW64) and type: **paceman -S git —noconfirm —disable-download-timeout**
- After the installation, git is available in MSYS2 terminals, but not in Windows cmd.

RUNNING PACMAN



- pacman automatically resolves all package dependencies.

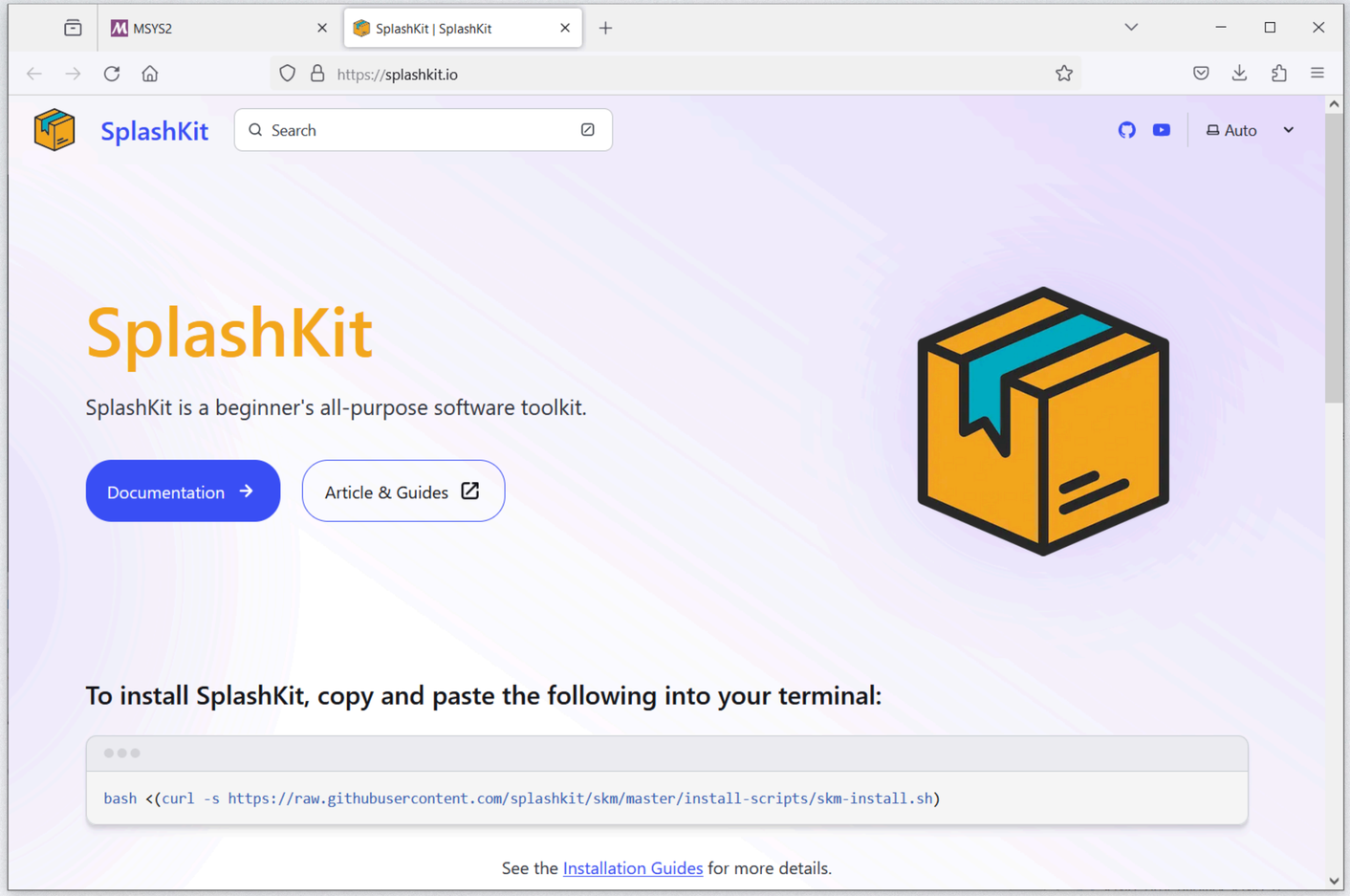
GIT INSTALLATION COMPLETE



- No errors have occurred while installing git.

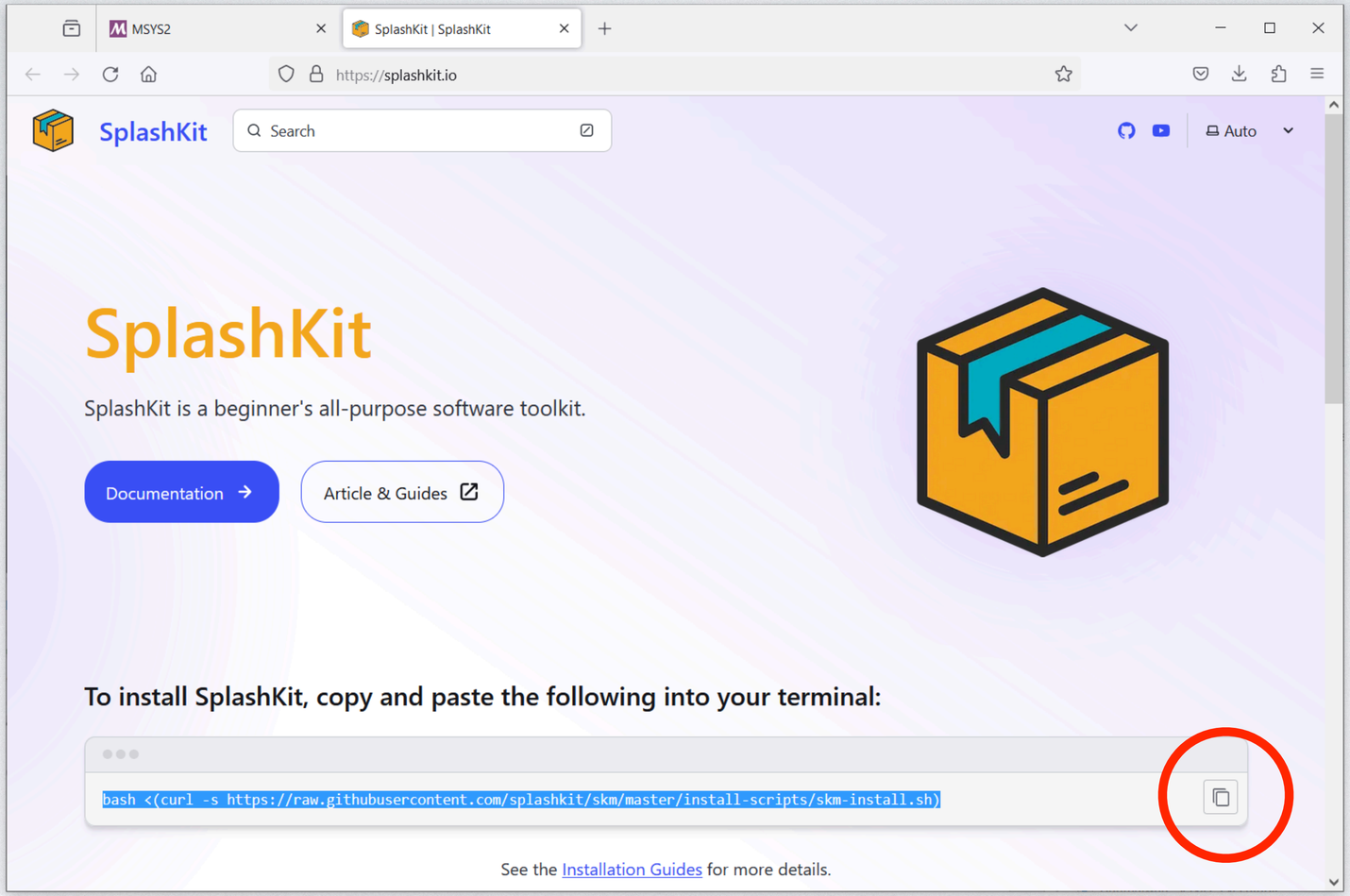
SPLASH KIT

SPLASH KIT HOME



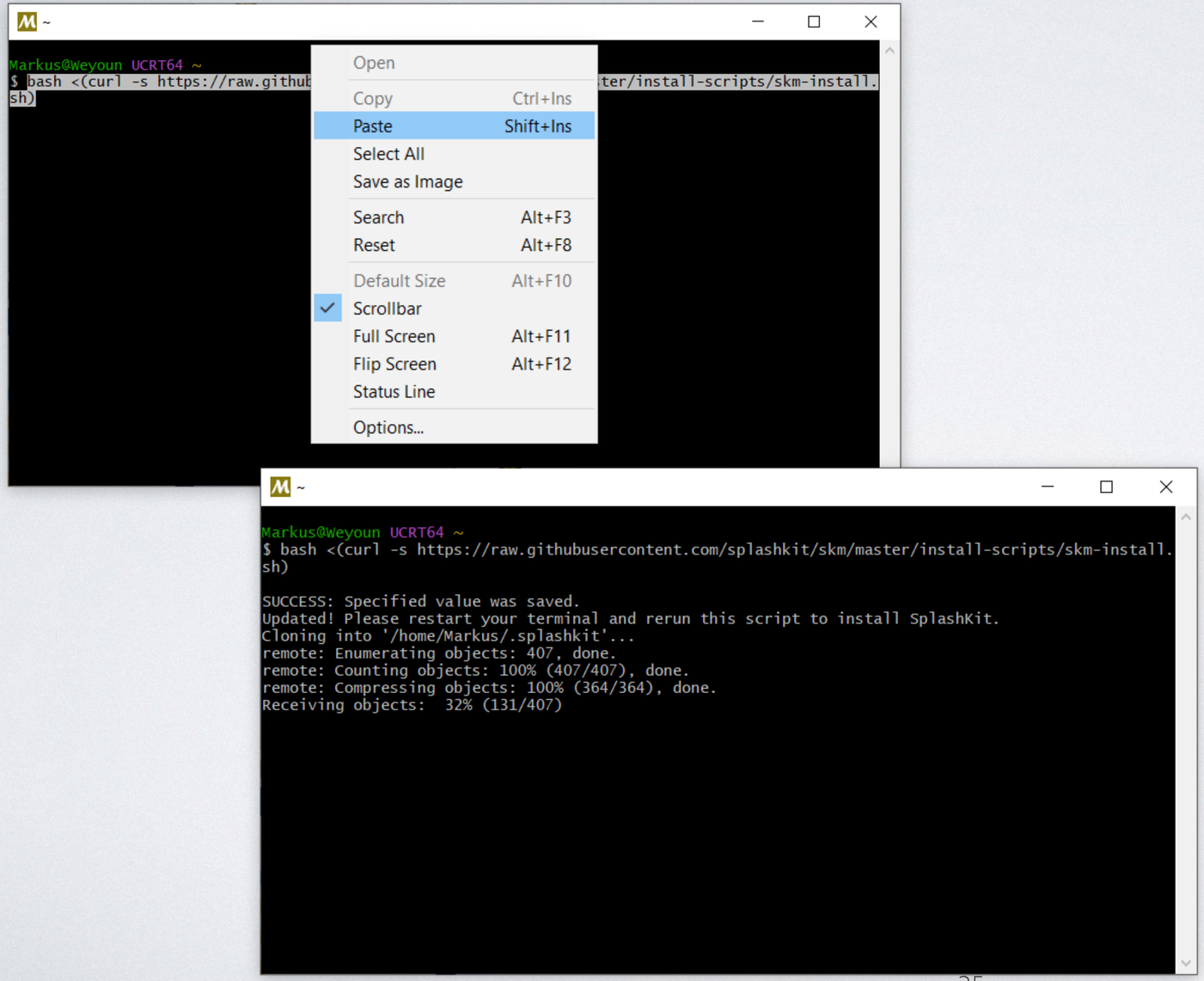
- SplashKit can be found at splashkit.io.

SPLASH KIT INSTALLATION COMMAND



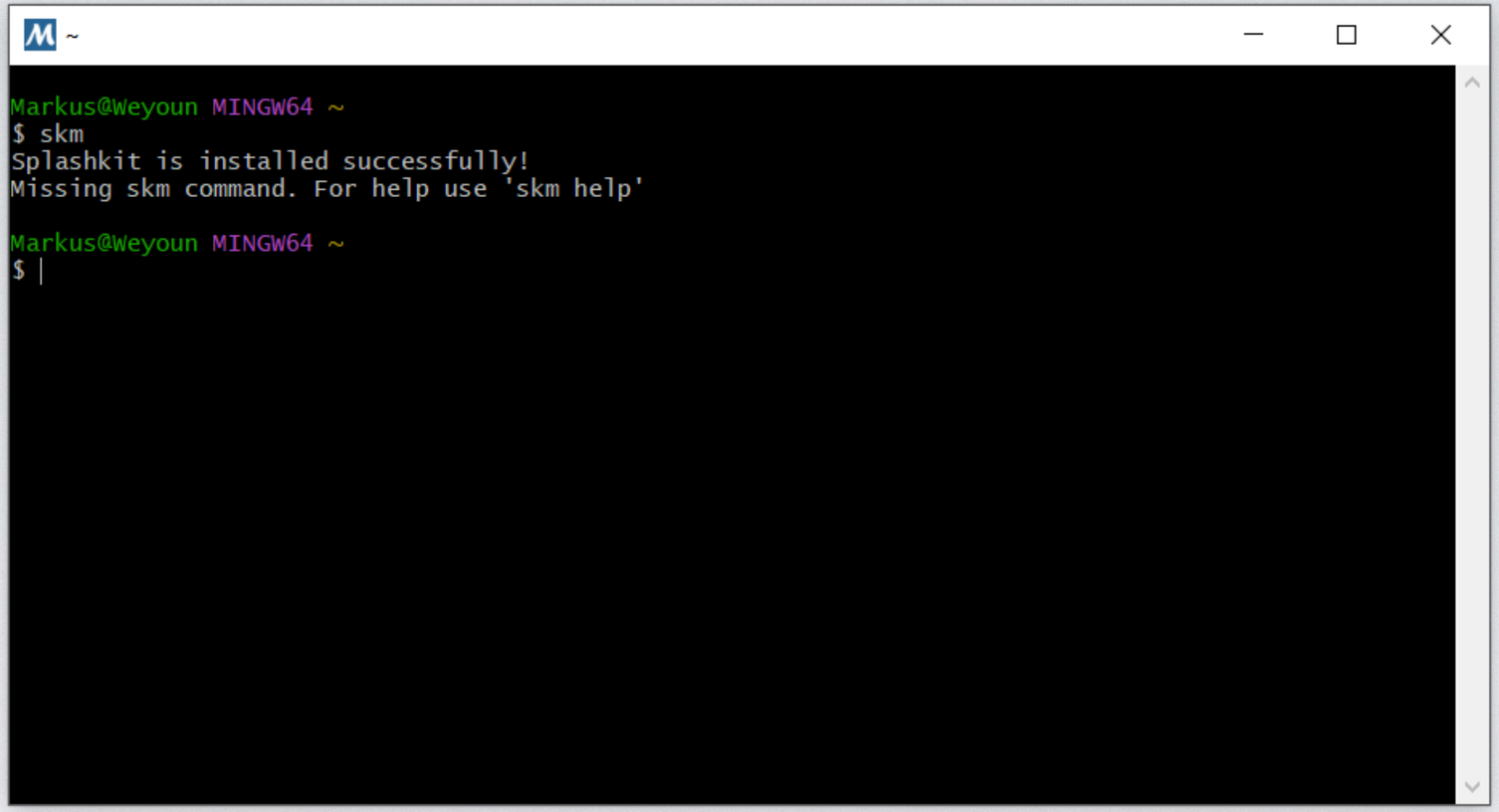
- Copy the SplashKit installation command line by clicking the copy button at the right of the text field.

SPLASH KIT INSTALLATION IN MSYS2



- Open a MSYS2 terminal (e.g., UCRT64 or MINGW64).
- Paste the copied installation command line into the terminal (Shift+Ins or right click mouse and select paste).
- Next, press Enter.
- Wait for the installation to finish.
- Close the terminal and start a new one. This time use MINGW64 as we need this terminal to create SplashKit projects.

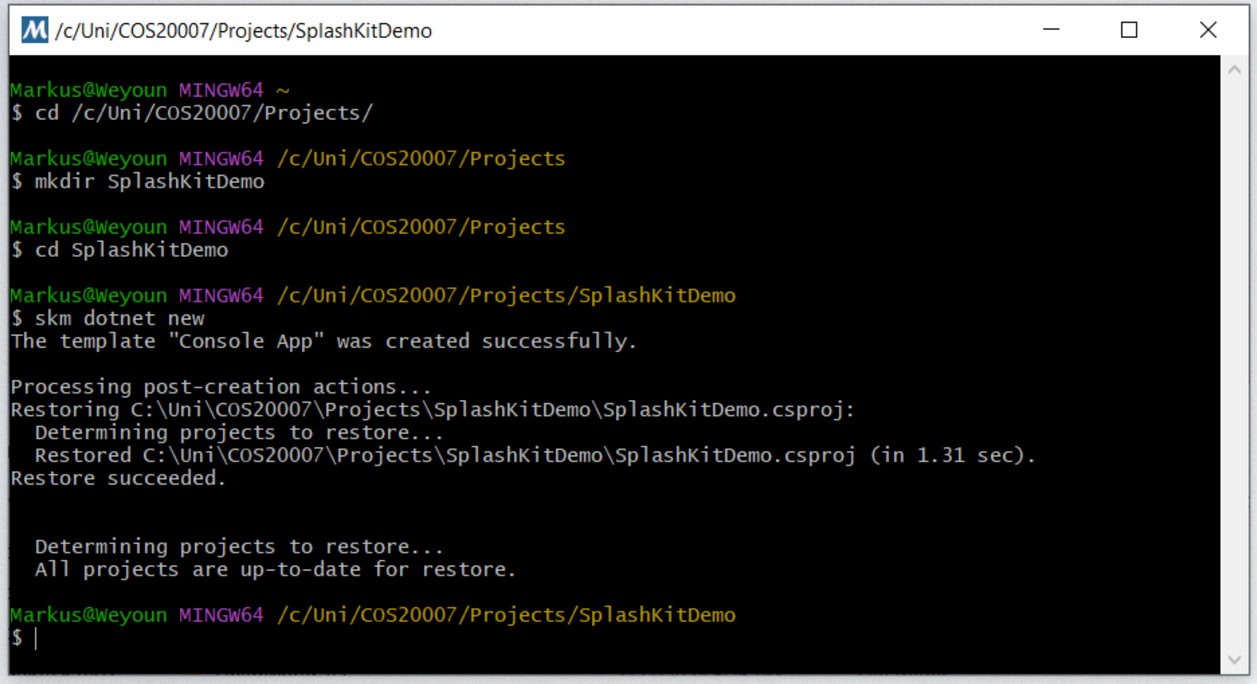
TEST SKM



- Type **skm** in the terminal to verify SplashKit is working.

SPLASH KIT DEMO

CREATE A SPLASH KIT PROJECT

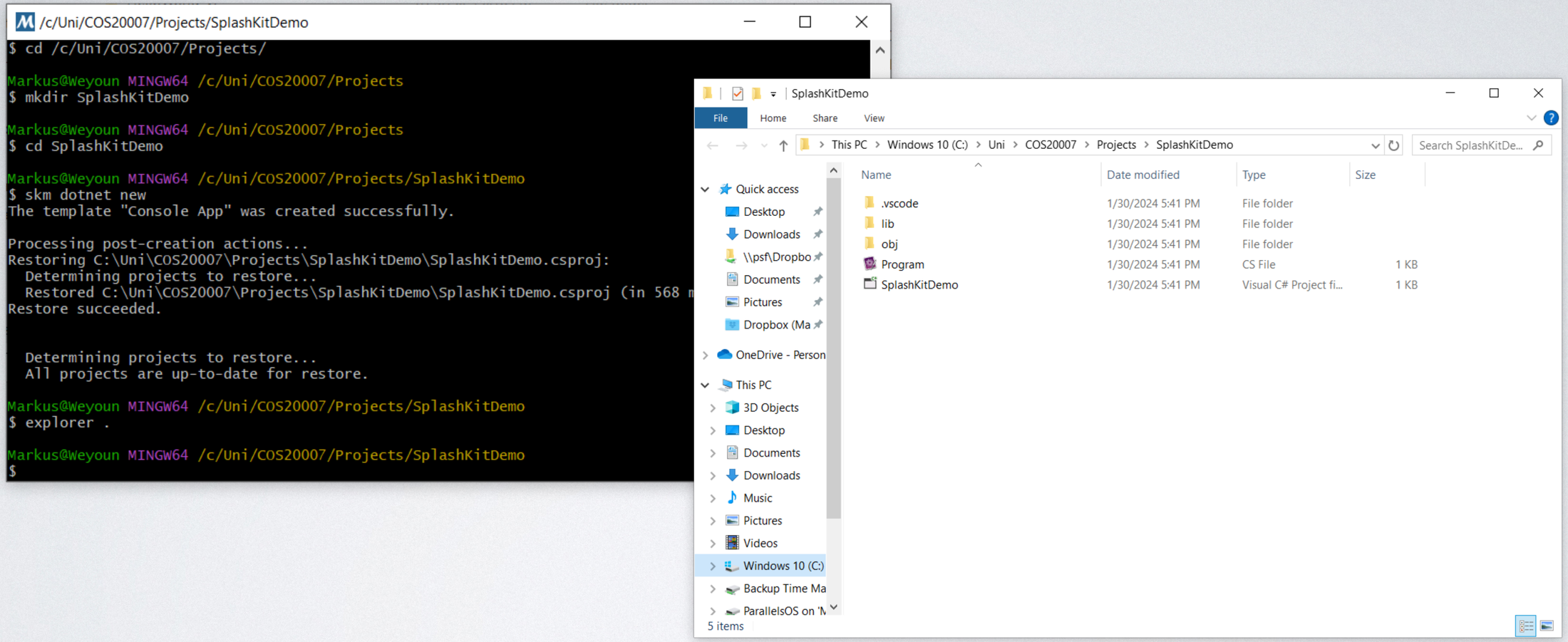


- To create a SplashKit project, take the following steps:

1. Open a MINGW64 terminal.
2. Change to the location of your COS20007 projects.
3. Create a new directory for your project (e.g., SplashKitDemo).
4. Change into the newly created project directory.
5. Run the command **skm dotnet new**.

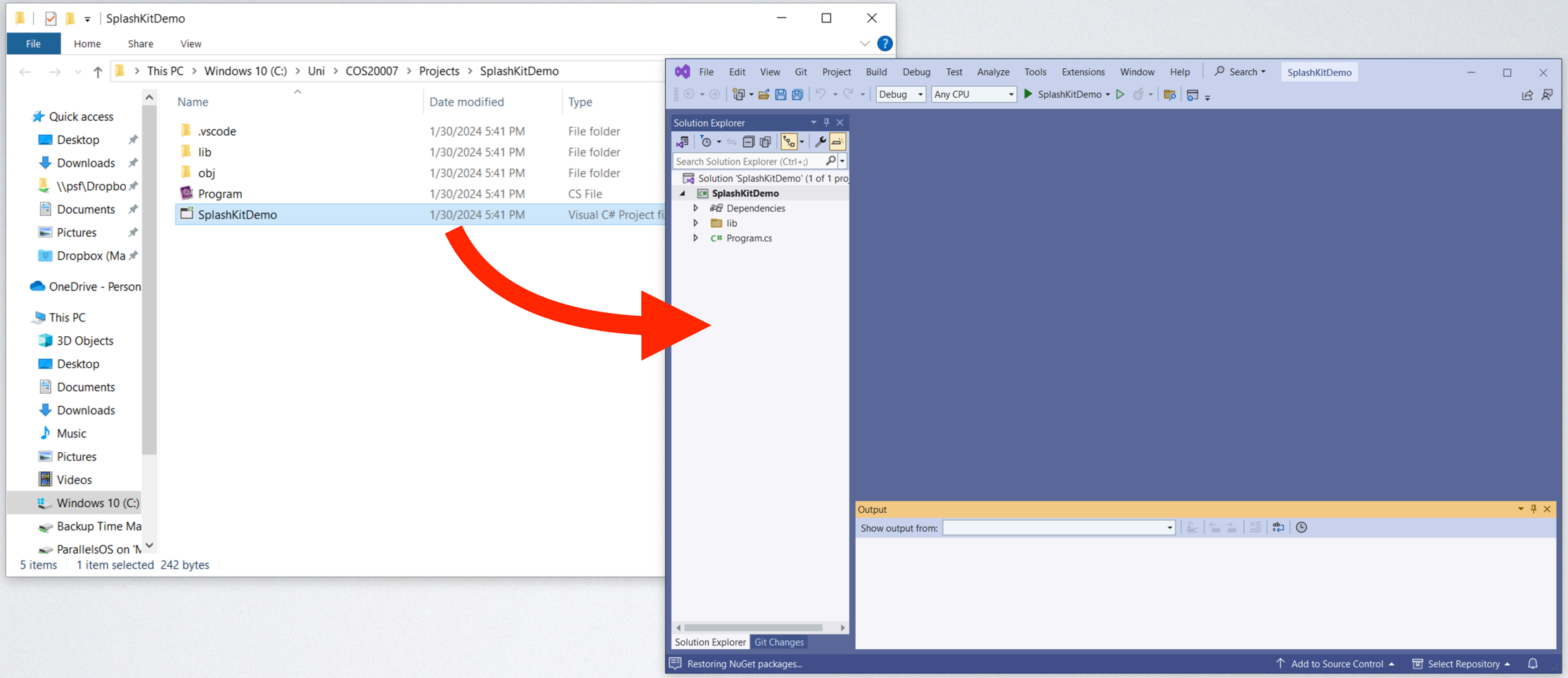
   The last step sets up a new SplashKit project.

OPEN THE PROJECT FOLDER



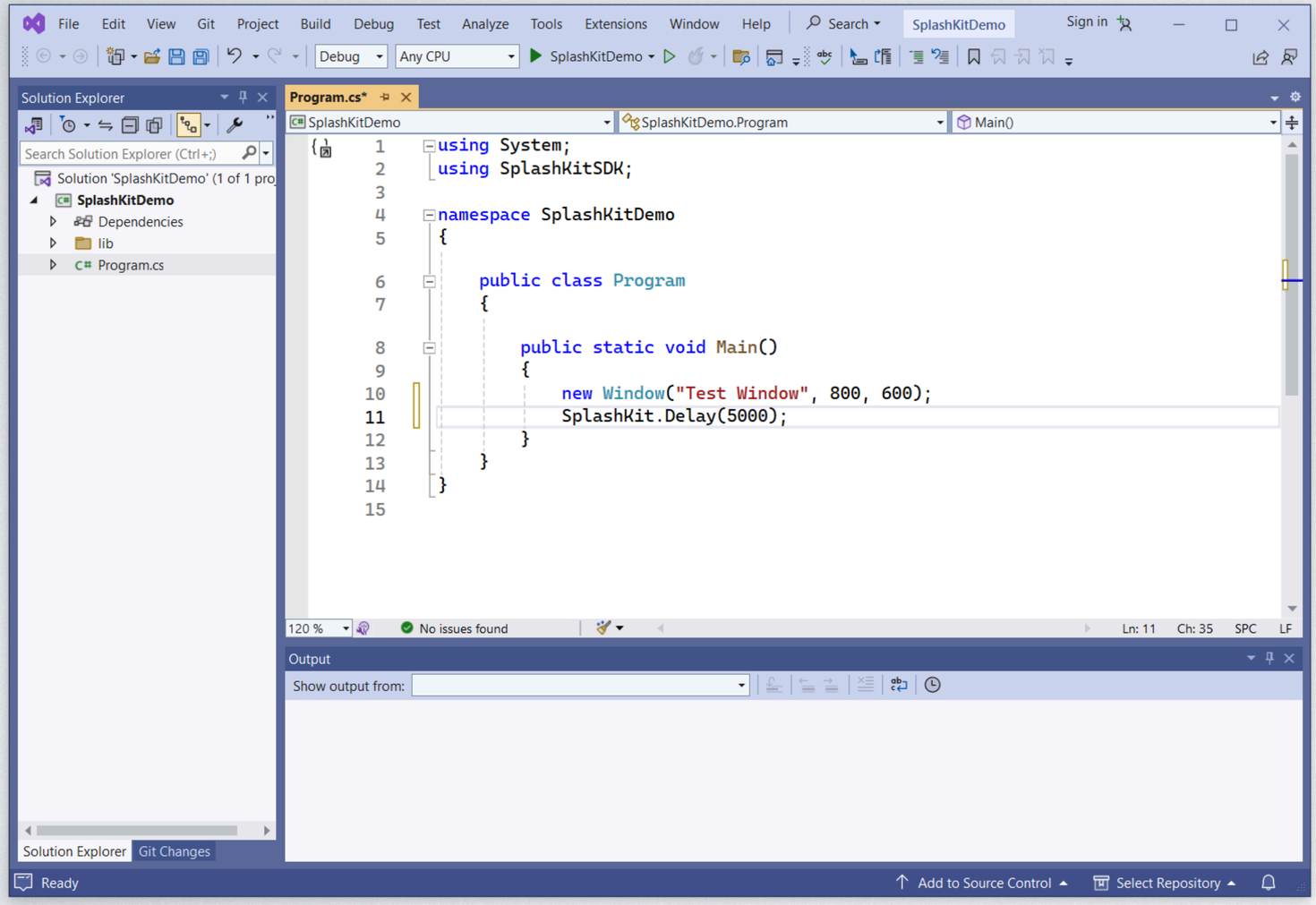
- Type **explorer .** to open the folder containing the newly-created project.

OPEN PROJECT IN VISUAL STUDIO



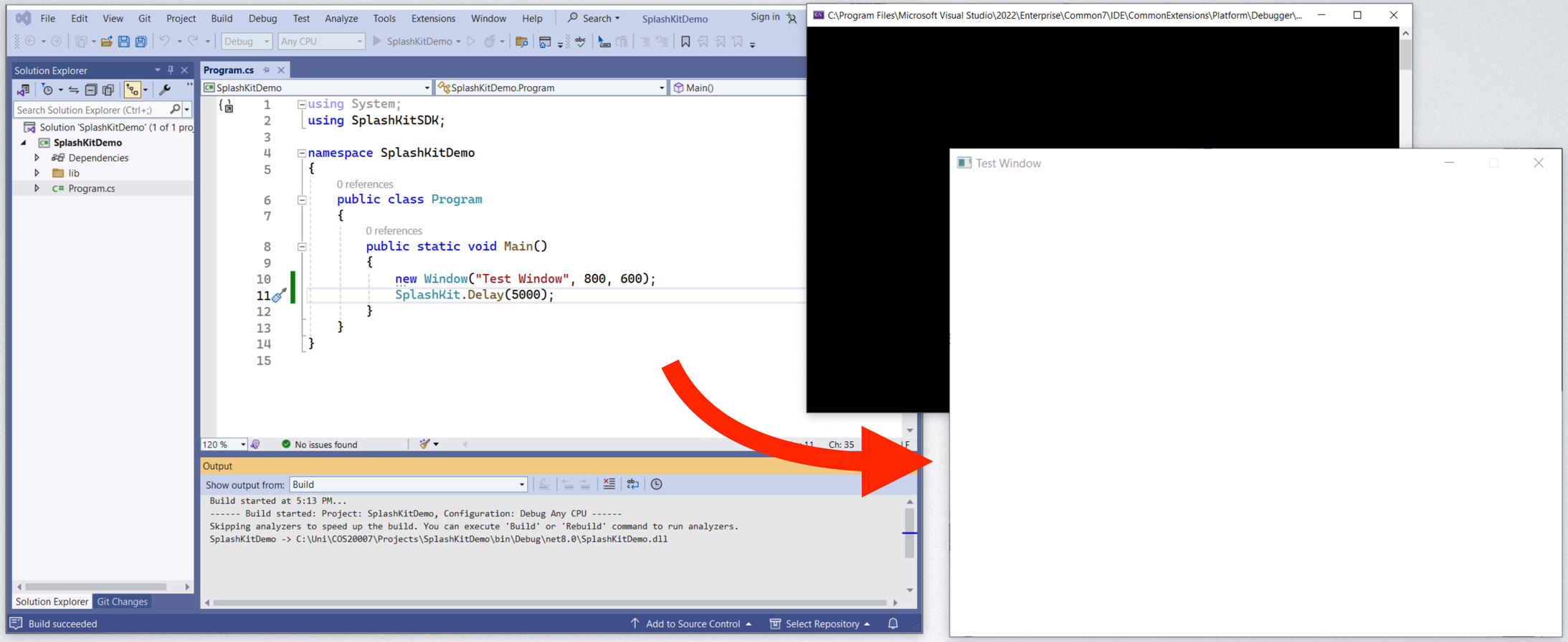
- Select the project file SplashKitDemo.csproj and double-click it to open the project in Visual Studio.

COMPLETE THE PROGRAM



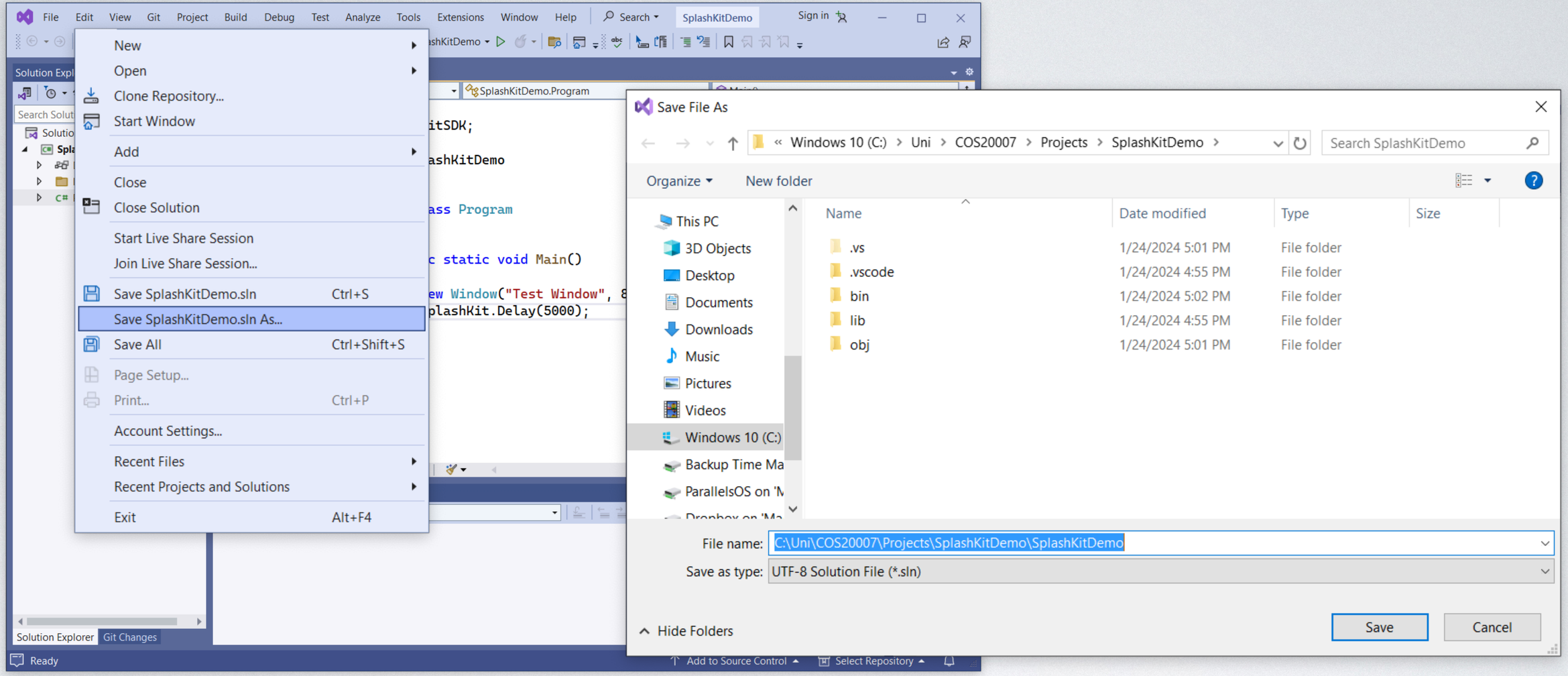
- Open Program.cs.
- In function Main, add the following two lines:
  - **new** Window("Test Window", 800, 600);
  - SplashKit.Delay(5000);
- The first line creates a window 800x600 with the title "Test Window", which is shown immediately.
- The second line instructs SplashKit to show the window for at least 5 seconds. Once 5 seconds have elapsed, the window is closed and the application terminates.

BUILD AND RUN



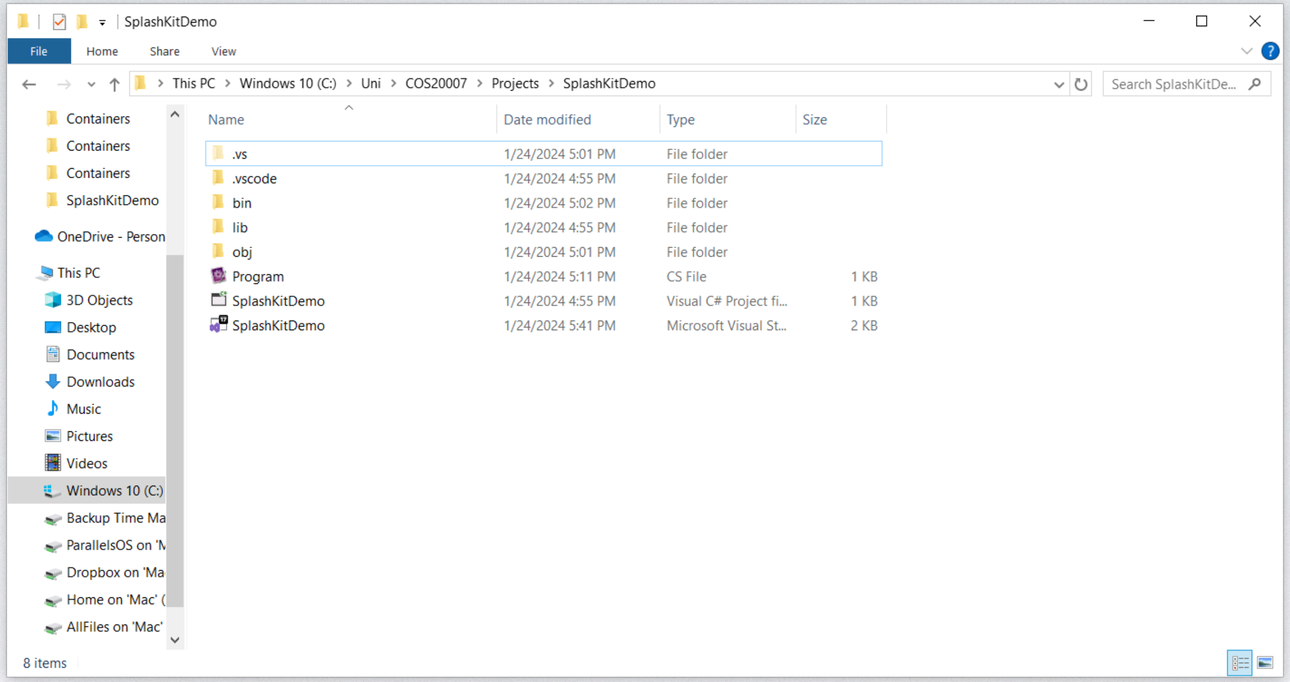
- Build and run the application by clicking on the run button ▶ .

SAVE SOLUTION



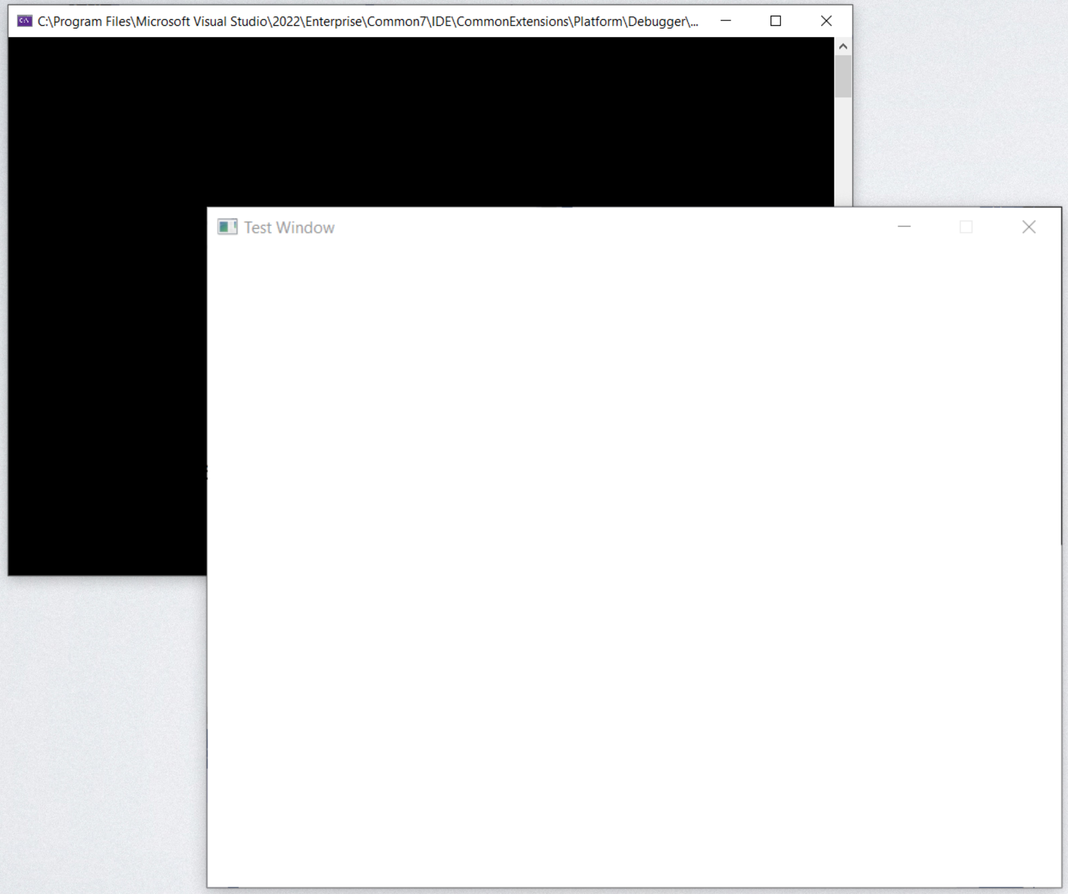
- Finally, select File/Save SplashKitDemo.sln As… to save the Visual Studio solution file.

NEW PROJECT FILE STRUCTURE



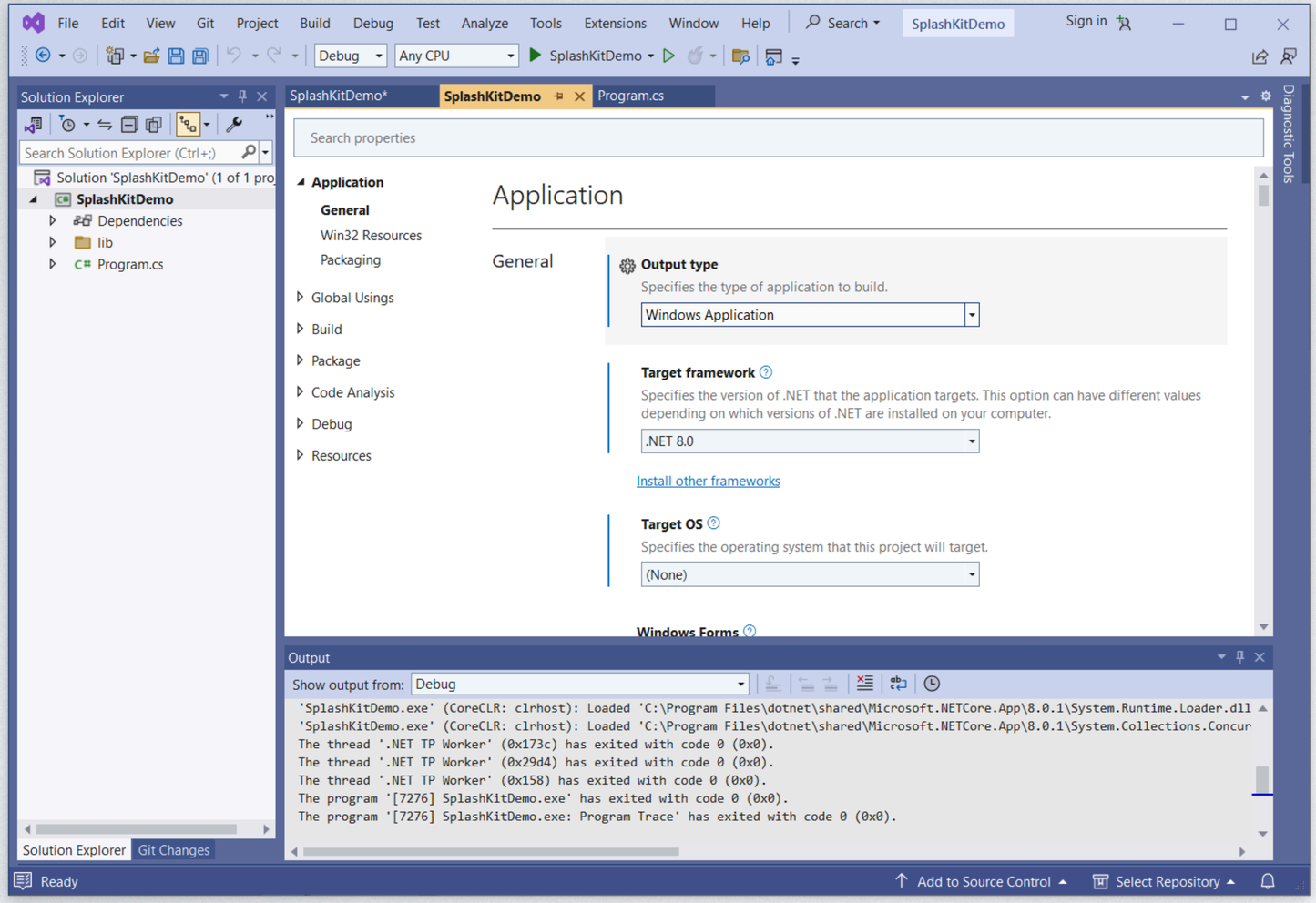
- The new project file structure contains a solution file which you use from now on to open your SplashKit project.
- Remember the distinction between project and solution name. The project name corresponds to the project file (i.e., SplashKitDemo.csproj), whereas the solution name matches the solution file (i.e., SplashKitDemo.sln).
- In Visual Studio, a solution can host multiple projects. If a solution only contains one project, generally, solution name and project name are the same.

SUPPRESSING THE TERMINAL WINDOW



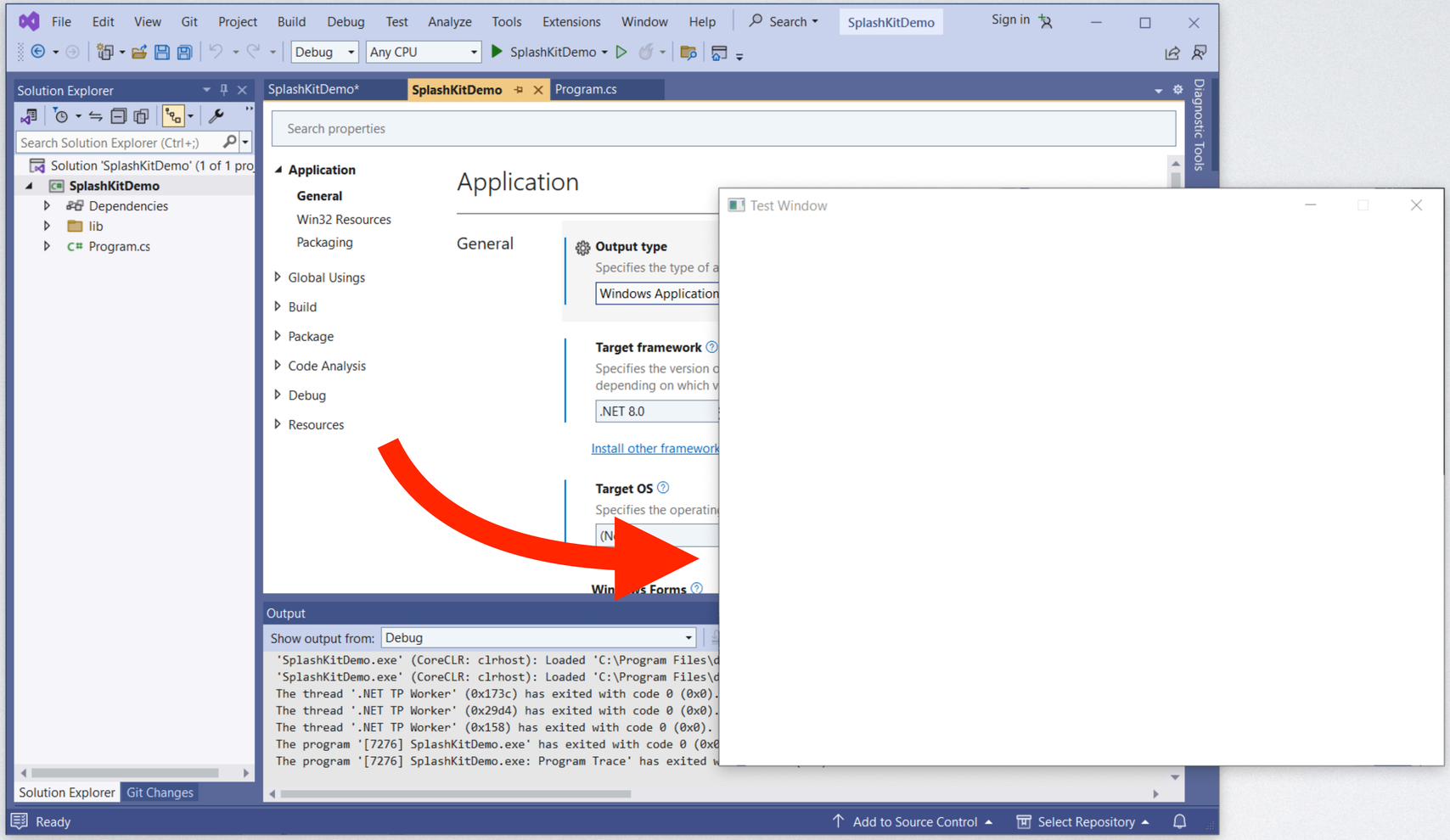
- Did you notice that when you run SplashKitDemo, two window opened: the main window and a terminal.
- This is the default behavior for SplashKit.
- The terminal allows for console output, if your application generates it.
- It is possible to suppress the terminal, if your application does not generate console output.

APPLICATION TYPE: WINDOWS APPLICATION



- Select the project in the solution explorer.
- Right-click and select properties.
- Visual Studio opens the project page for you project.
- Under Application/General change Output type from Console Application to Windows Application.
- Build the project again.

SPLASH KIT DEMO: NO TERMINAL



HAPPY CODING