# Anleitung zum Erstellen der Projekte

## Allgemeines

Wir empfehlen VS Code als Editor für Linux und das Windows Subsystem for Linux und Visual Studio Community für Windows. Grundsätzlich könnt ihr natürlich nutzen, was ihr schon kennt, allerdings können wir nur bei diesen Konfigurationen helfen.

## Linux

Unter Linux nutzen wir `CMake`. Dieses Programm generiert aus einer jeweils von uns bereitgestellten `CMakeLists.txt` Datei eine sog. Makefile, die vom Build-System `make` interpretiert wird, um das Projekt zu kompilieren.

Installiert dafür zunächst `CMake` und `build-essential` (enthält u. a. `make` und `gcc`). Auf Debian und Derivaten (z. B. Ubuntu) klappt das im Terminal mit

```
sudo apt-get update && sudo apt-get install cmake build-essential
```

Im Laufe dieses Prozess' fragt `apt-get`, ob die Pakete wirklich installiert werden sollen. Bestätigt hier mit Y (engl.) bzw. J (deutsch).

Für ein gegebenes Projekt navigiert ihr nun in den Ordner, in dem die `CMakeLists.txt` liegt. Dort erstellt ihr mittels

```
mkdir build
```

einen `build`-Ordner, wechselt mittels

```
cd build
```

in diesen Ordner und führt dort dann

```
cmake ..
```

aus. Das generiert die Makefile für das Projekt. Mittels

```
make
```

führt ihr diese aus und kompiliert so euer Projekt. Die generierten ausführbaren Dateien könnt ihr dann mit

```
./executable-name
```

ausführen.

Nehmt ihr Änderungen am Code vor, so müsst ihr nur `make` aufrufen, um das Projekt erneut zu kompilieren.

Bevor ihr abgebt, löscht bitte den `build` Ordner, in dem die von CMake generierten Dateien sowie die ausführbaren Dateien liegen.

## MacOS

### MacOS wird nicht von uns unterstützt

Unter MacOS sollte alles ähnlich funktionieren wie unter Linux. Verwendet am besten auch hier einfach `CMake`, um eure Projekte mittels `make` zu kompilieren oder ein `XCode`-Projekt zu generieren.

## Windows

Unter Windows gibt es zwei Möglichkeiten, die Projekte dieses Kurs' zu bearbeiten:

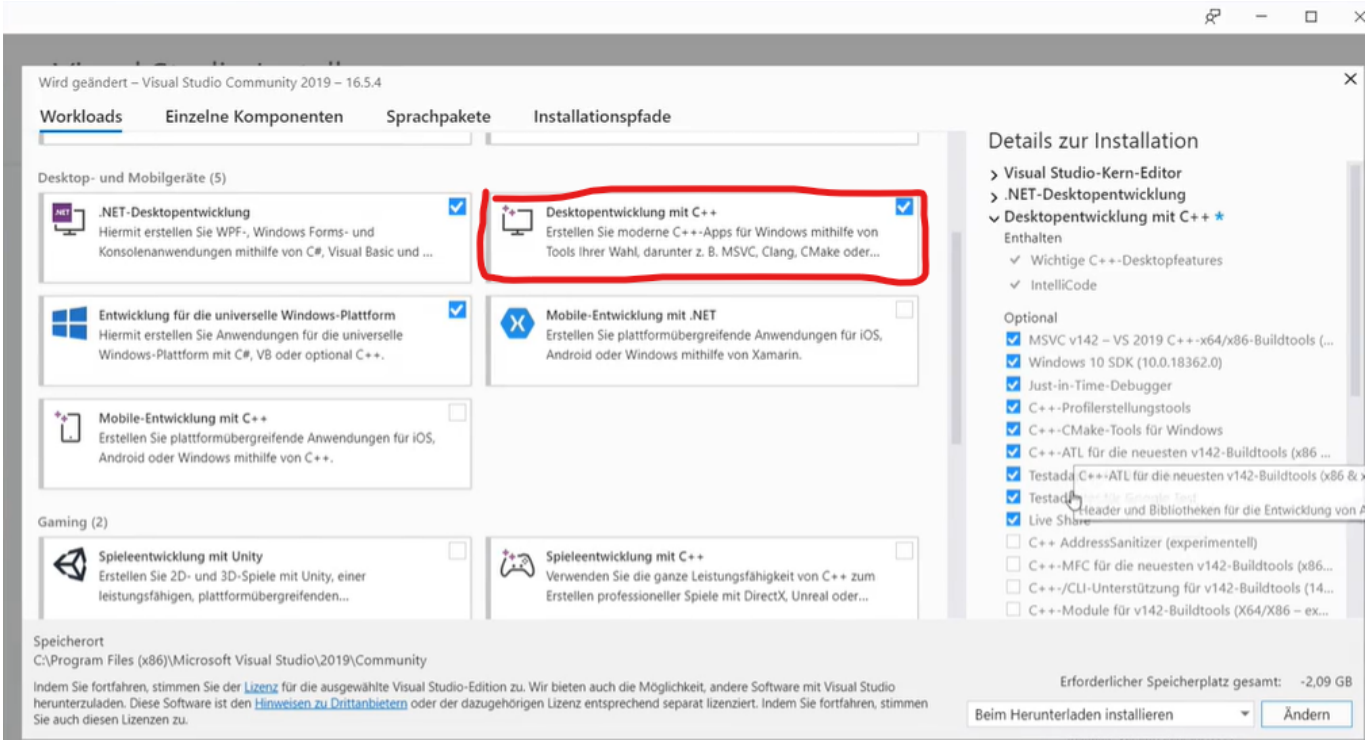
- Visual Studio (Community): Eine umfangreiche, sehr große (Download-Größe und Festplatte) IDE von Microsoft
- Windows Subsystem for Linux (WSL): Ermöglicht das Ausführen von Linux-Distros wie Ubuntu auf Windows.

### Visual Studio

#### Installation

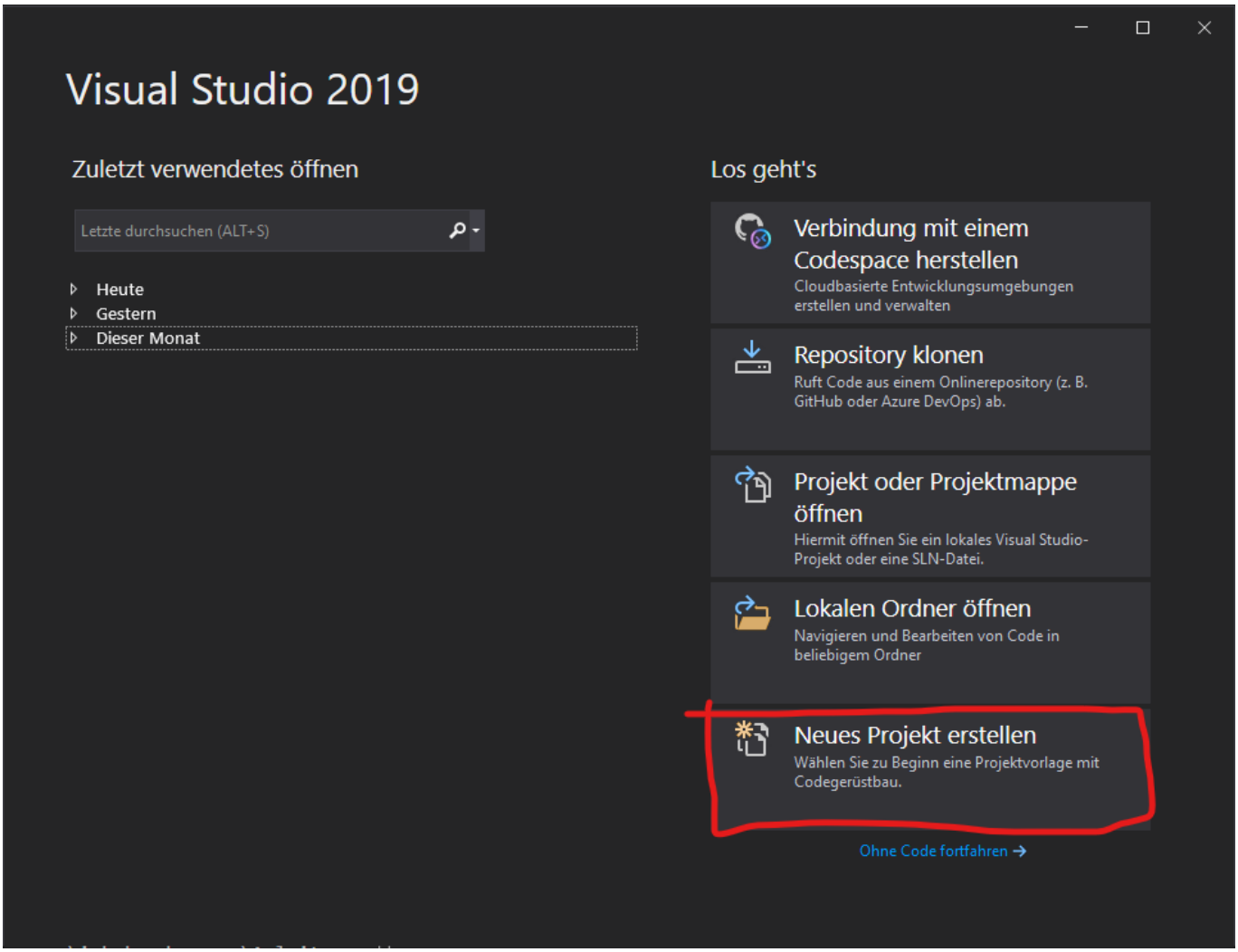
Ladet Visual Studio Community herunter.

Im Installer wählt ihr dann *Desktopentwicklung mit C++* aus. Alle anderen Optionen könnt ihr abwählen.

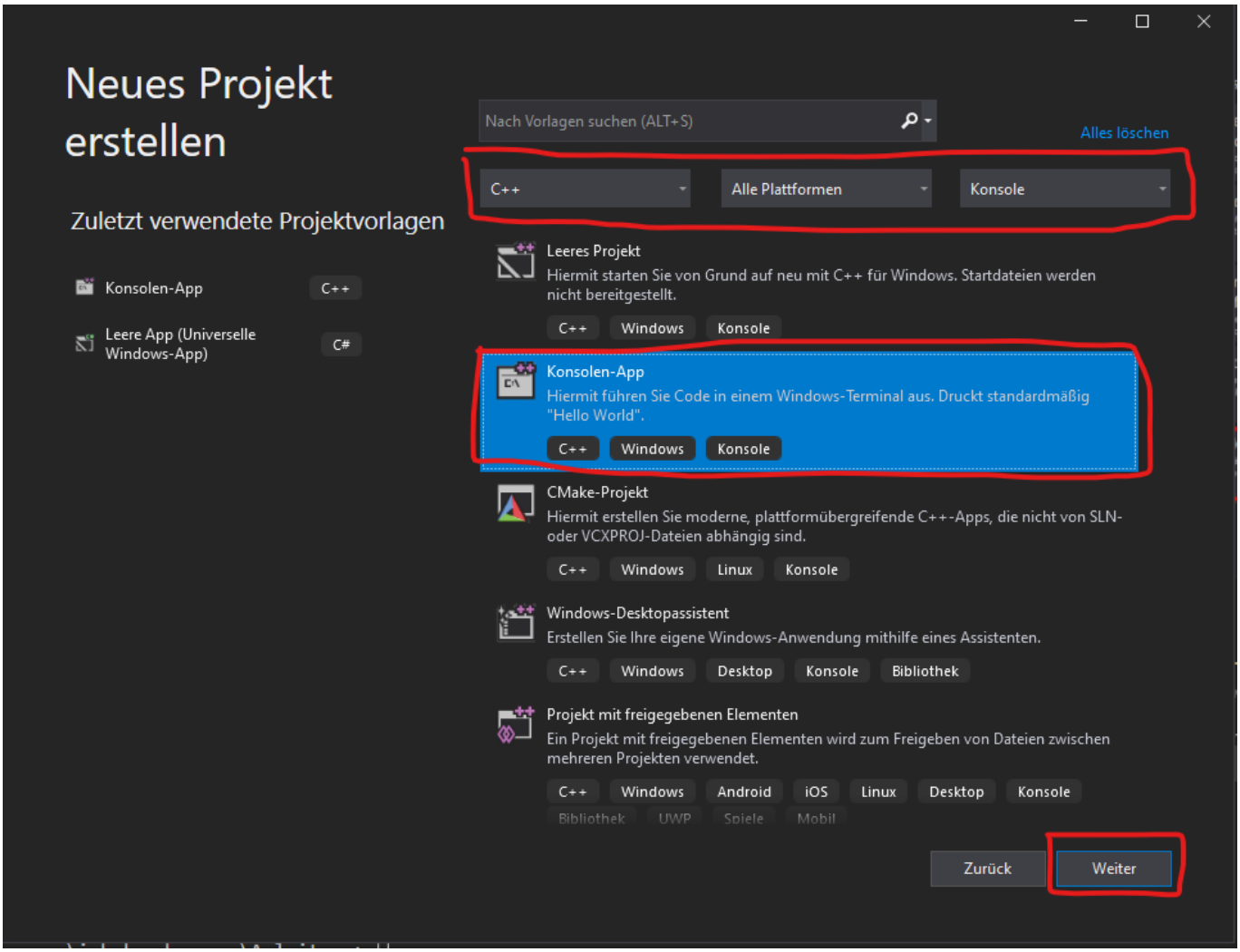



#### Projekt erstellen

Für Projekte, für die wir keine `.sln`-Datei bereitstellen, müssst ihr in Visual Studio ein sog. Konsolenprojekt erstellen. Wählt dazu auf der Visual Studio Startseite *Neues Projekt* aus:

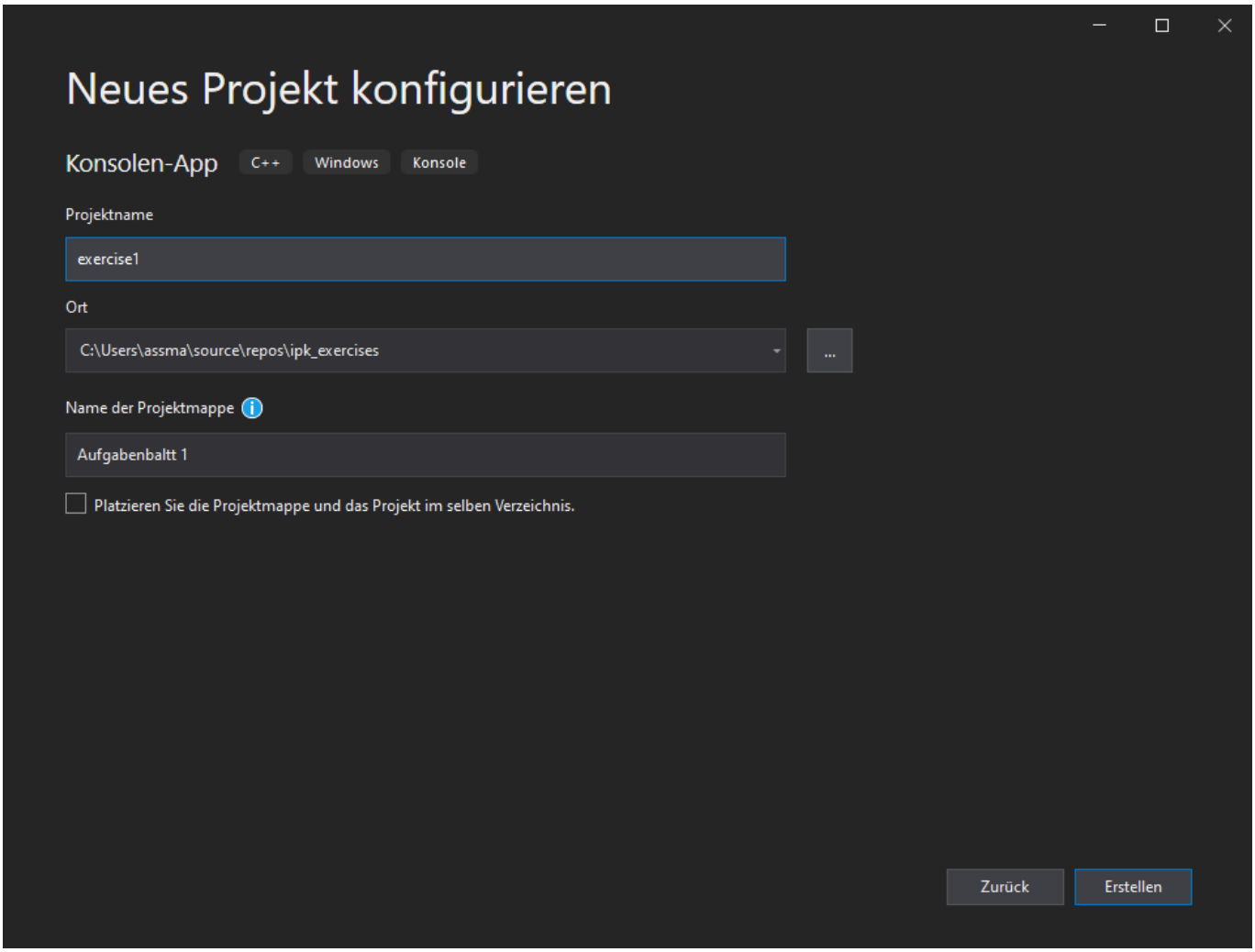



Wählt dort dann den Projekttyp Konsole für C++ aus:

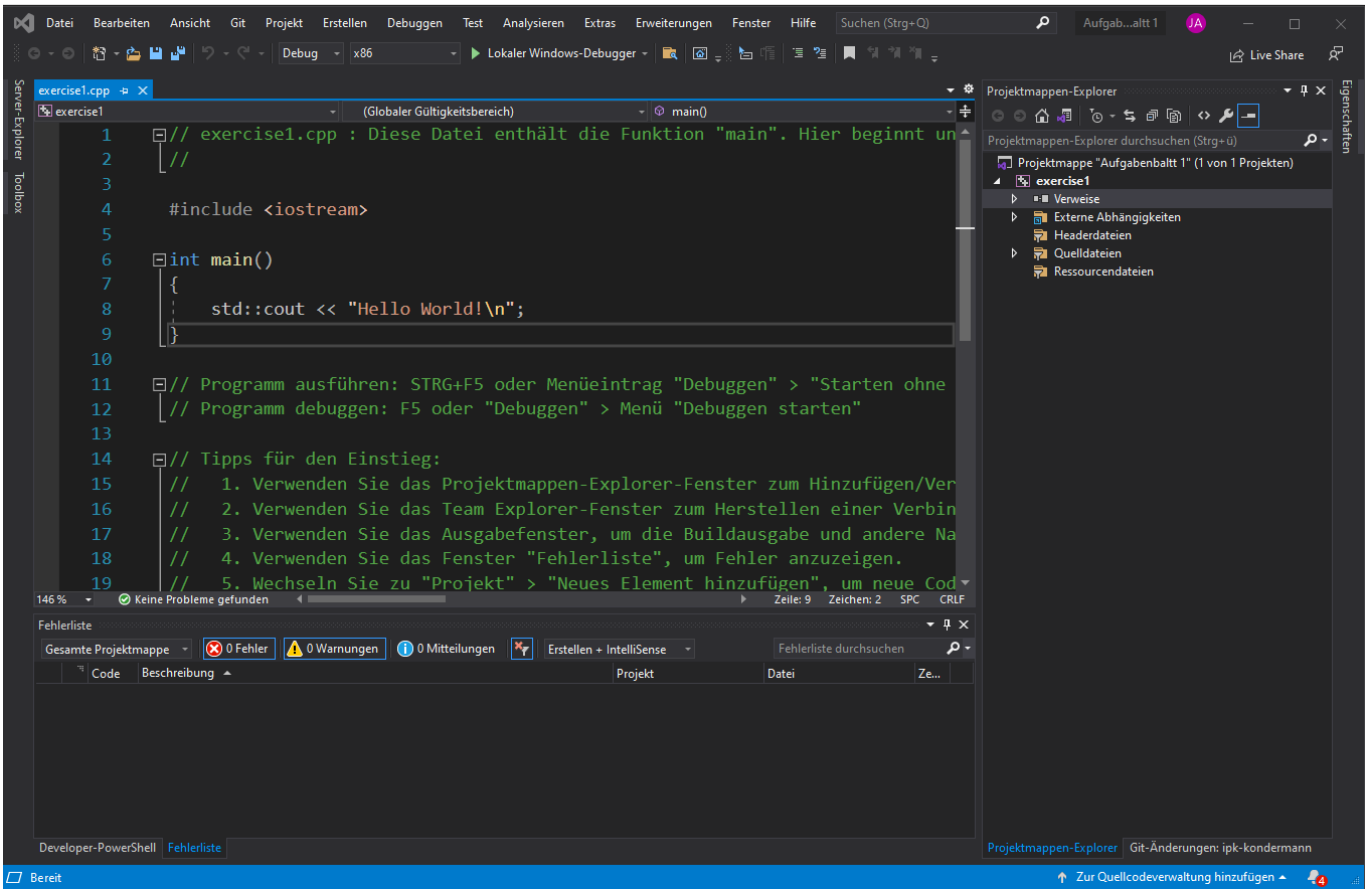



Im nächsten Schritt könnt ihr die Namen und Speicherort konfigurieren. Merkt euch auf jeden Fall den Speicherort, damit ihr später wisst, *wo* ihr die Dateien wiederfindet, sodass ihr sie abgeben könnt.

Visual Studio arbeitet mit sog. *Projektmappen*, in denen ggf. mehrere *Projekte* liegen können. Erstellt für jedes Blatt eine neue Projektmappe und setzt *Name der Projektmappe* im Dialogfenster auf `Aufgabenblatt X`. Für den Projektnamen könnt ihr immer `exercise1` angeben, weitere Projekte für `exerciseX` werden wir im nächsten Schritt erstellen.

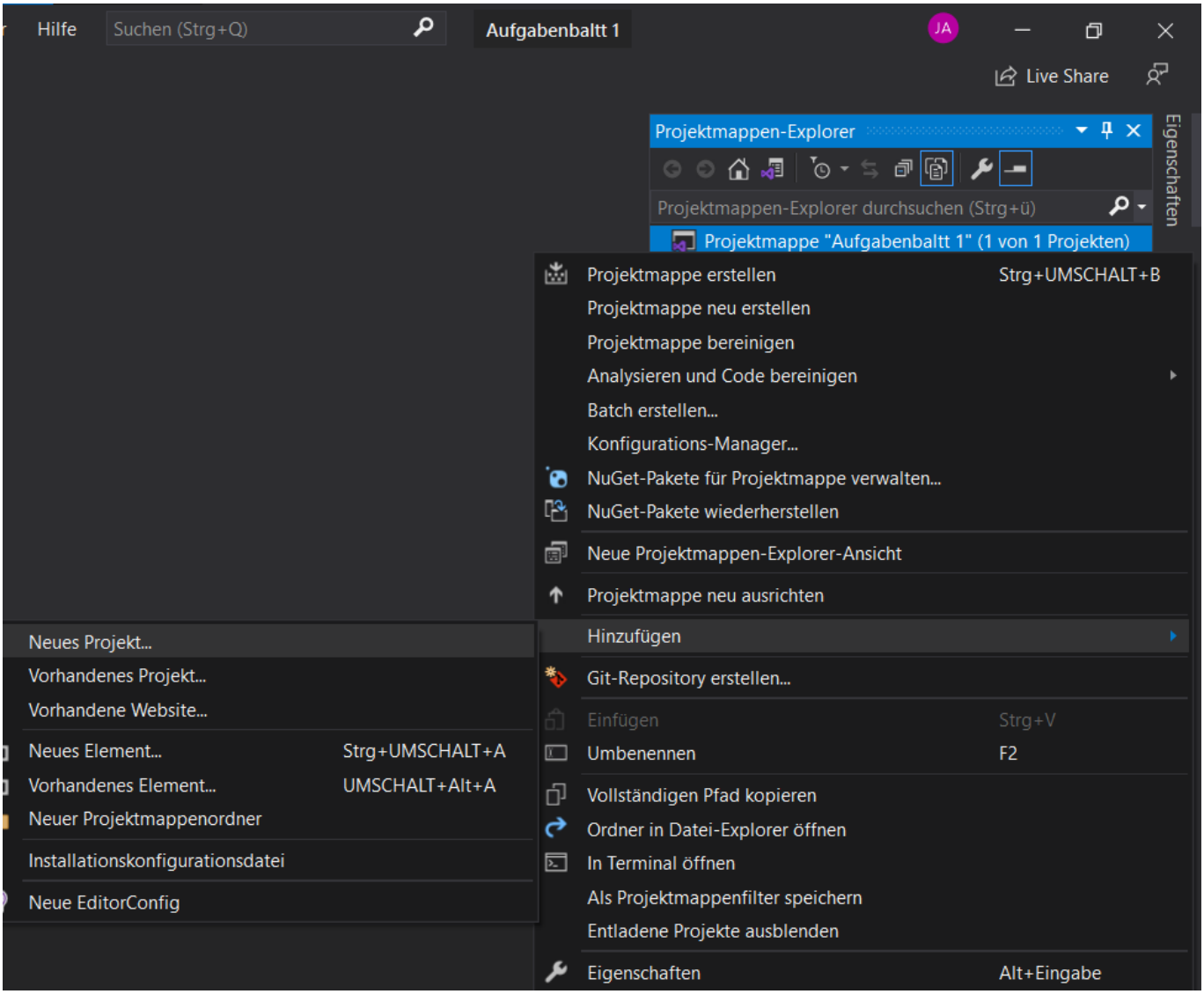



Nun erstellt Visual Studio das Projekt. Ihr solltet im Anschluss ungefähr das Folgende zu sehen sein:

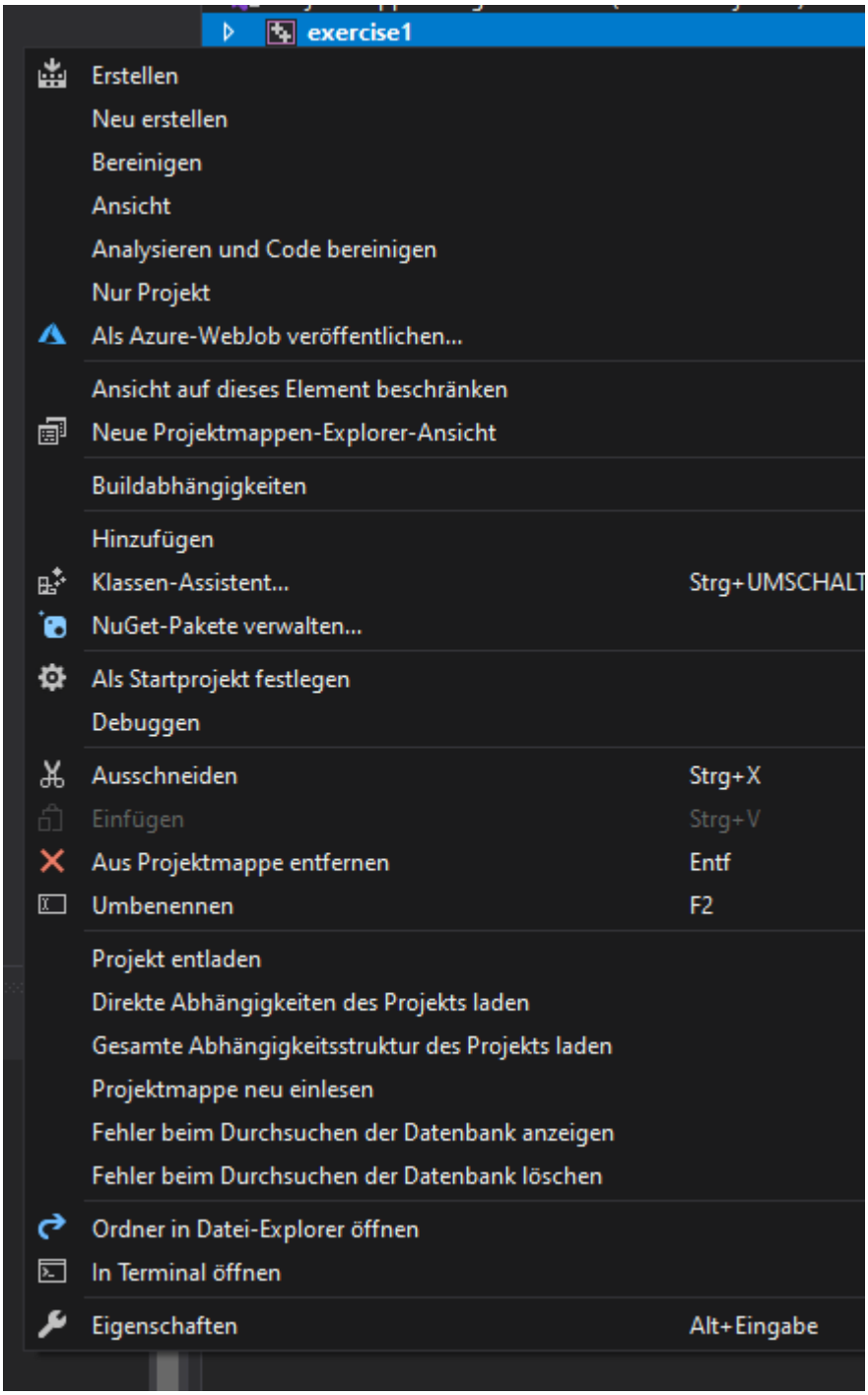


Ihr seht rechts den sog. *Projektmappen-Explorer*, der euch die Projektmappe(n), die Projekte und die Dateien anzeigt. Im erzeugten Projekt `exercise1` seht ihr unter

Auf den ersten Blättern sind viele verschiedene, voneinander unabhängige Aufgaben. Erstellt für jede Aufgabe ein neues Projekt in der Projektmappe. Dazu klickt ihr rechts auf die Projektmappe und wählt dann *Hinzufügen -> Neues Projekt*.




Wenn ihr nun ein Projekt mit F5 ausführt, so wird immer das aktuell ausgewählte "Startprojekt" ausgeführt, zu erkennen an der fetten Schrift des Projektnamens im Projektmappen-Explorer. Um ein anderes Projekt auszuführen müsst ihr es zum Startprojekt machen. Dazu klickt man im Projektmappen-Explorer rechts auf das Projekt und wählt *Als Startprojekt festlegen*.




**Vorhandenes Projekt öffnen**

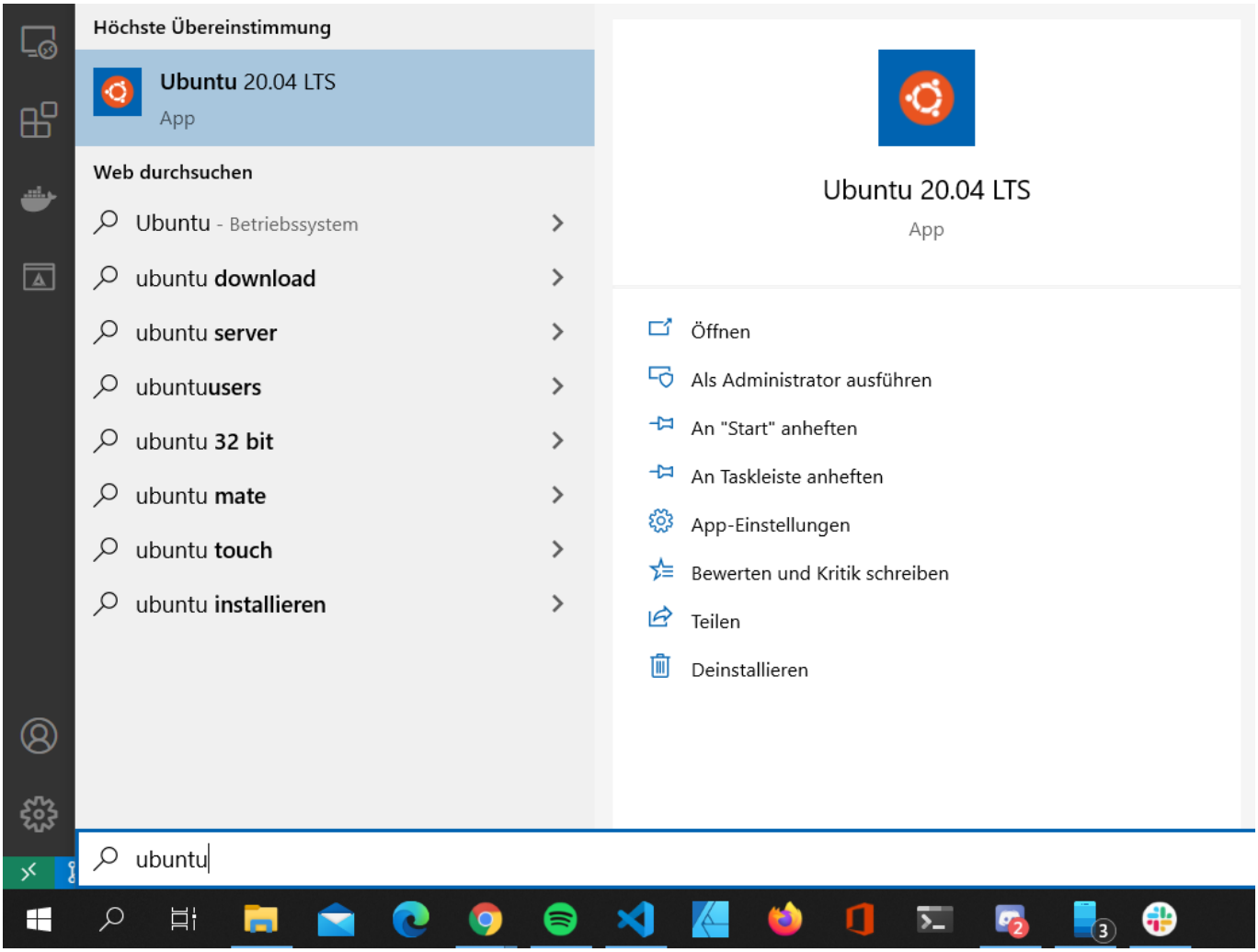
Nun könnt ihr die Projekte mittels der bereitgestelltn `.sln` (Solution) Dateien in Visual Studio öffnen und mit einem Klick auf den grünen Pfeil ausführen.

Für Abgaben löscht bitte die von Visual Studio erstellten `Debug` und `Release` Ordner, sodass nur noch die Quelldateien (enden mit `.cpp`) und die Visual Studio Dateien (`.vcxproj`, `.sln`) bleiben.

## Windows Subsystem for Linux

**Funktioniert nur mit Windows 10**

Ihr könnt das Windows Subsystem for Linux mittels dieser Anleitung installieren. Wählt dabei am besten Ubuntu als eure Distro. Nun könnt ihr ein Terminal im Subsystem öffnen. Dazu sucht ihr in Windows einfach nach *Ubuntu*:

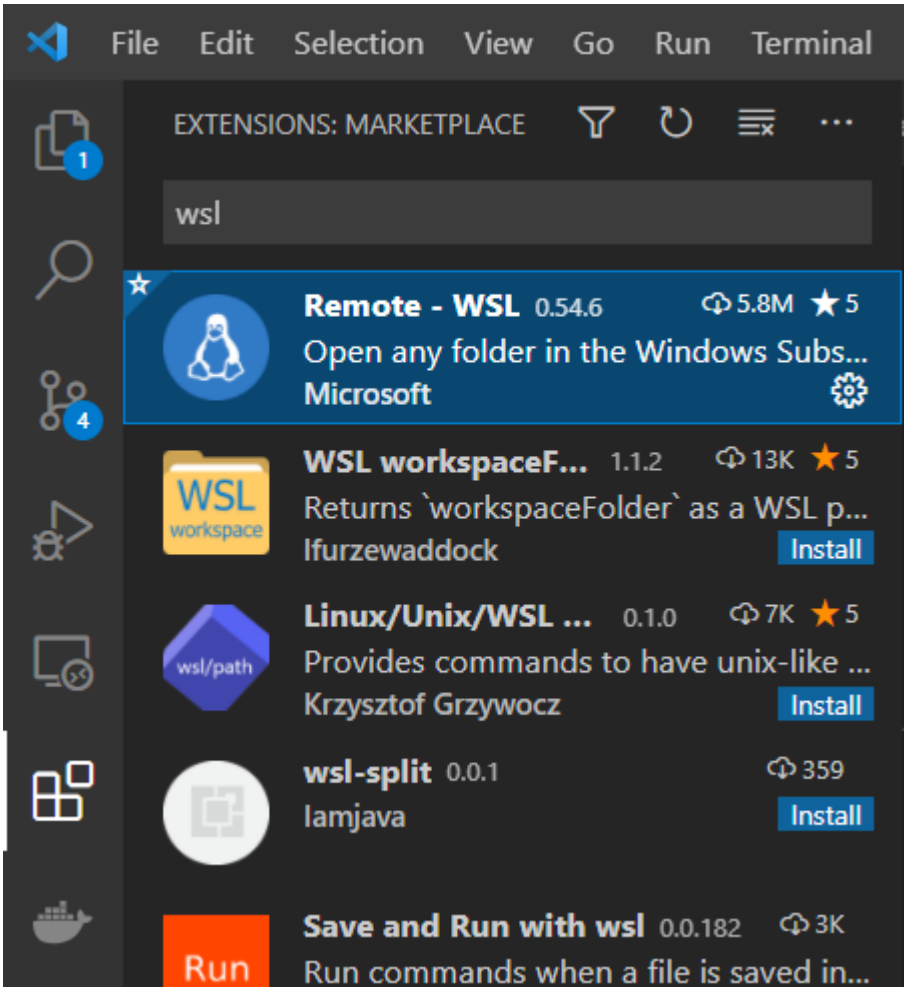



Anschließend müssen, wie bei Ubuntu auch, `CMake` und `build-essential` (enthält u. a. `make` und `gcc`) installiert werden. Dazu gebt ihr folgenden Befehl im WSL-Terminal ein:

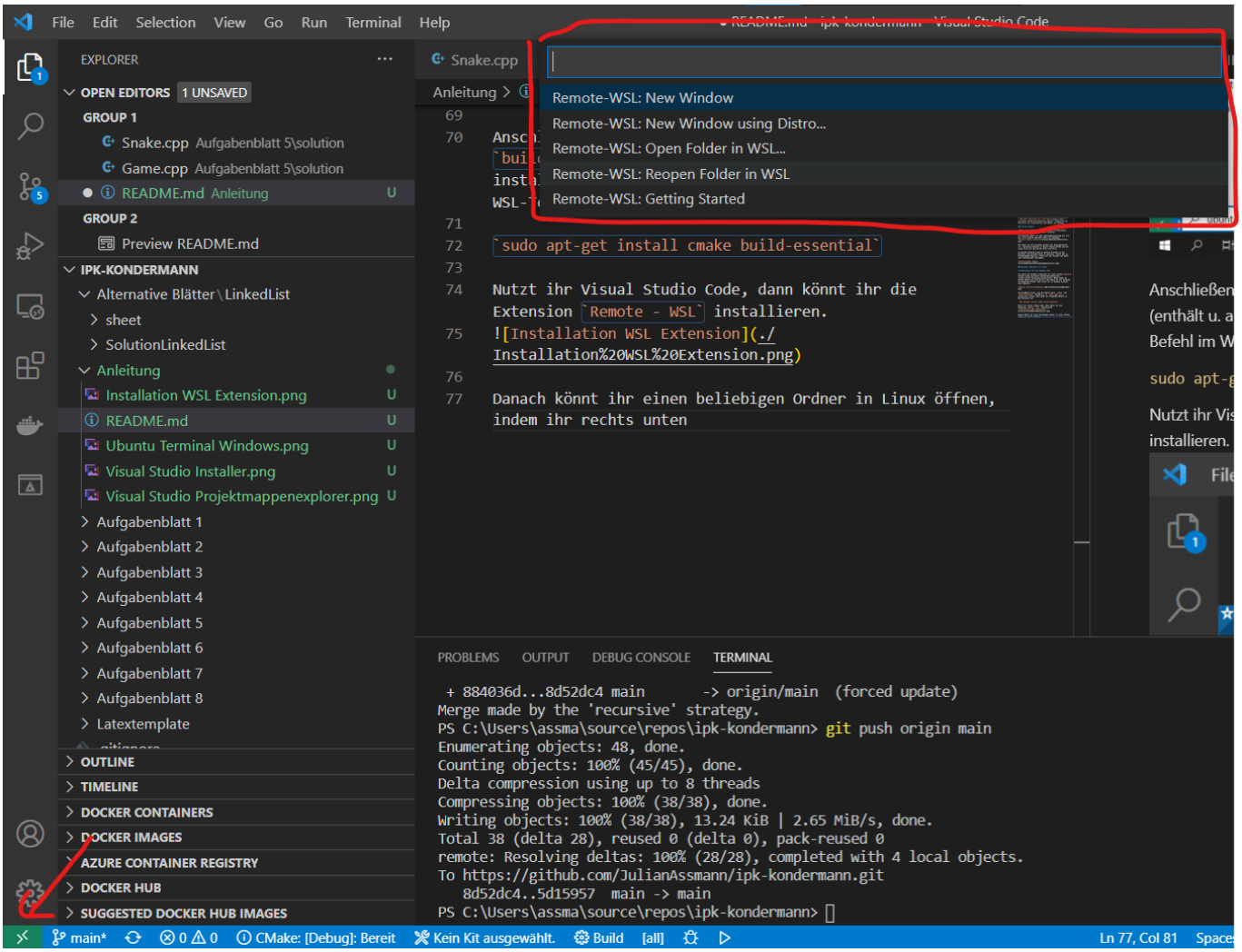
```
sudo apt-get update && sudo apt-get install cmake build-essential
```

Im Laufe dieses Prozess' fragt `apt-get`, ob die Pakete wirklich installiert werden sollen. Bestätigt hier mit Y (engl.) bzw. J (deutsch).

Nutzt ihr Visual Studio Code, dann könnt ihr die Extension `Remote - WSL` installieren.




Danach könnt ihr einen beliebigen Ordner in Linux öffnen, indem ihr links unten auf den grünen Doppelpfeil klicken und den entsprechenden Ordner in VS Code öffnen.




Den Rest der Anleitung findet ihr unter dem Abschnitt *Linux*.