

This course has already ended.

Development environment

The recommended and supported code editor for the course is [Visual Studio Code](#). It is free, versatile and open-source editor that works with many different languages through extensions. The exercises contain configurations for compiling and debugging with it, and it comes with git integration, which you can use for the exercises and the project. It is also possible to use some other editor or IDE through makefiles if you prefer.

The following are instructions for installing and configuring VS Code for C/C++ on different platforms. **You only need to follow one of the instructions.**

Remote connection to school computers and VS Code (available on all operating systems)


1. Install [Visual Studio Code](#)
2. Install [Remote Development extension pack](#) to VS Code.
3. Press the double arrow icon in the lower left corner and select "Remote-SSH: Connect to Host..."
4. Select "+ Add New SSH Host..." and write ssh, your Aalto username and `@lyta.aalto.fi` to the field that opens, i.e. if your username is `student1`, write `ssh student1@lyta.aalto.fi` If connection to lyta.aalto.fi does not seem to work, you can try *kosh.aalto.fi* instead.
5. Select an SSH configuration to update (usually the first one)
6. A popup appears in the lower right corner, from where you can use the Connect button to connect to the school server (can be later on done straight from the button in the lower left corner).
7. Choose "Linux", when you are asked about the platform of the remote host.
8. Choose "Continue" when you are asked "Are you sure you want to continue?"
9. Write your Aalto password to the field that opens.
10. Open the Extensions tab (lowest icon in the left side in VS Code) and search for "c++".
11. Install the C/C++ extension to VS Code in lyta.
12. Now you're ready to do the exercises: You can download the zip containing the exercises templates by downloading the .zip file from the link at the top of a chapter.

Windows



There are two options listed here for Windows. **You only need to follow one of them.** The first one runs the programs and gcc compiler through the Windows subsystem for Linux available on Windows 10. This means that the programs are run on an Ubuntu subsystem instead of Windows. The other choice is to use the Mingw-w64 package to compile and run the programs on Windows. We recommend using the first option as the server used for grading runs the assignments on Linux but either one is fine.

Recommended: GCC through WSL (Windows 10 only)

1. Install Windows Subsystem for Linux according to instructions [here](#), and install [Ubuntu 20.04 LTS](#) for it:
 - Open PowerShell as Administrator and run: `dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
 - Restart computer when prompted
 - Download Ubuntu 20.04 LTS from Microsoft Store and install it
 - After installing, launch it and configure:
 - Enter new UNIX username: (this will be your username inside Ubuntu)
 - New password & Retype new password: (set your admin password for Ubuntu)
 - Install the build-essential and gdb packages on WSL:

```
sudo apt update
sudo apt-get install build-essential gdb
```
 - Now you should have your Ubuntu 20.04 LTS configured!
 - To access Windows files from the Ubuntu terminal, go to folder `/mnt/`, which includes all your system drives mounted, for example `c` for your C: drive.
2. Install [Visual Studio Code](#)
 - **Note:** When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily open a folder in WSL using the code command.
3. Install the [Remote Development extension pack](#).
4. Install the [C/C++ extension for VS Code](#) on WSL side. Instructions copied from [Open VS Code in WSL](#).
 - Open a WSL terminal window (using the start menu item for Ubuntu 20.04 LTS).
 - Navigate to a folder you'd like to open in VS Code. This would be wherever you have stored your exercise files.
 - Type `code .` in the terminal. When doing this for the first time, you should see VS Code fetching components needed to run in WSL. This should only take a short while, and is only needed once.
 - Note: If this command does not work, you may need to restart your terminal or you may not have added VS Code to your path when it was installed.
 - After a moment, a new VS Code window will appear.
 - VS Code will now continue to configure itself in WSL and keep you up to date as it makes progress.
 - Once finished, you now see a WSL indicator in the bottom left corner, and you'll be able to use VS Code as you would normally!
 - Open the Extensions tab from the following icon (or press Ctrl+Shift+X) 
 - Search for C++ and install the C/C++ extension.
5. All done! Now you can move onto the next section about doing the exercises.

Other option: Mingw-w64

1. Install [Visual Studio Code](#)
2. Install [C/C++ extension for VS Code](#). You can either download it from the above link or do the following in VSCode:
 - Open the Extensions tab from the following icon (or press Ctrl+Shift+X) 
 - Type C++ in the search bar and install the C/C++-extension.
3. Download and install Mingw-w64 from [here](#).
 - Download from the link below 'MinGW-W64 Online Installer'
 - Make the following choices during the install: 
 - You may choose the install location yourself. These instructions assume the default location: `C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0`
4. Add the `bin` folder which is inside the `mingw64` folder inside the install location to the PATH environment variable:
 - (If these instructions don't work, ask help from the course assistants or try to look up instructions for your own operating system and language)
 - Search for `Edit environment variables` in Windows and select `Edit environment variables for your account`
 - Select the `Path` variable and press `Edit`
 - Select `New` and add the address of the `bin` folder. With the above location it would be `C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin`
 - Press `Ok`
 - If you have a console or VSCode open, you'll need to restart it for the changes to take effect.
5. All done! Now you can move onto the next section about doing the exercises.

Linux

1. Install the build-essential and gdb packages
 - **These are already installed on the Aalto computers**
 - Run the following commands:

```
sudo apt update
sudo apt-get install build-essential gdb
```
2. Install [Visual Studio Code](#)
 - **To install it on an Aalto computer, you have to download the .tar.gz instead of the .deb**
3. Install the [C/C++ extension for VS Code](#)

MacOS

1. Install [Visual Studio Code](#) on macOS.
 - More detailed information available [here](#)
2. Install the [C/C++ extension for VS Code](#).
 - You can install the C/C++ extension by searching for c++ in the Extensions view (⇧⌘X).
3. Ensure Clang is installed
 - Clang may already be installed on your Mac. To verify that it is, open a macOS Terminal window and enter the following command:

```
clang --version
```
 - If Clang isn't installed, enter the following command to install the command line developer tools:

```
xcode-select --install
```
4. Now you should be able to use the exercises, but note: If you can build the *main* options in the exercises but not the *test* ones, run `brew install gcc` and uncomment (remove the #) the second line in *scripts/test.make* file.