MTH 4300: Algorithms, Computers and Programming II

Fall 2025

Section: STRA

Lecture 1

Installing Visual Studio Code

For this class, I recommend using Visual Studio Code or Cursor in order to standardize our development environments.

- Visual Studio Code
- Cursor

By making sure that everyone is using the same development environment, we can ensure that we're all on the same page and it helps me debug any issues that may arise.

Of course, you're free to use whatever IDE you're comfortable with, but just know that I may not be able to fully support any issues you may have with your editor.

Installing a C++ compiler

For Windows, you should follow the instructions in the following link to install MinGW:

https://code.visualstudio.com/docs/languages/cpp#_example-install-mingwx64-on-windows

Once you've completed this, you should be able to run the following command in the terminal:

```
gcc --version
g++ --version
gdb --version
```

For MacOS, you should already have a C++ compiler installed if at any point you've installed Xcode.

To verify this, run the following command in the terminal:

```
clang --version
```

For other systems, you should follow the instructions in the following link to install a C++ compiler:

https://code.visualstudio.com/docs/languages/cpp

Installing CMake

In general, An Introduction to CMake is a good resource for learning more about CMake.

There is a section in the link above that explains how to install CMake on your system.

For those on MacOS, you should install CMake via Homebrew.

For those on Windows, you can download the installer from the following link: https://cmake.org/download/

Lab: Hello, World!

To conclude today's lecture, we'll be writing a simple program that prints `Hello, World!` to the terminal.

For this, create a new directory called `hello-world` in a directory of your choice.

Now within this directory, let's create the following:
- a file called `main.cpp` which will hold our code.
- a file called `CMakeLists.txt` which will hold our CMake configuration.
- a directory called `build` which will hold our build output.

Implementing `Hello, World!`

Within our `main.cpp` file, we'll need to implement the following:

```cpp
#include <iostream>

int main() {
  std::cout << "Hello, World!" << std::endl;
  return 0;
}
```

In C++, we can use the `std::cout` object to print to the terminal.

`std::endl` is a special object that tells the terminal to print a new line.

Running the program without CMake

In your terminal, run the following command to build an executable for `main.cpp`:

```
g++ -o hello-world main.cpp
```

What this is doing is taking advantage of the `g++` compiler to compile the `main.cpp` file and create an executable called `hello-world`. Note that `-o` is used to specify the name of the executable.

If successful, you should see that we'll have a our executable in the **current** directory. If you run the following command, you should see that we'll have a our executable in the **current** directory.

```
./hello-world
```

We should then see `Hello, World!` printed to the terminal.

Note that it's okay to compile and run the program without CMake for small standalone programs like this, but in general, when a project gets larger and more dependencies are added, we need a build system to help us manage the compilation process.

This is where CMake comes in.

Filling out the CMakeLists.txt file

CMakeLists.txt is a file that tells CMake how to build our project.

```
cmake_minimum_required(VERSION 3.20)
project(hello-world)

add_executable(hello-world main.cpp)
```

At a high-level, this file tells CMake to:
- use the minimum version of CMake that we support (3.20)
- name our project `hello-world`
- add a target called `hello-world` which will build our executable from the `main.cpp` file.

Configuring the CMake project

Before we can actually build our project, we first need to configure our project using CMake.

To do this, we'll need to run the following commands in the terminal:

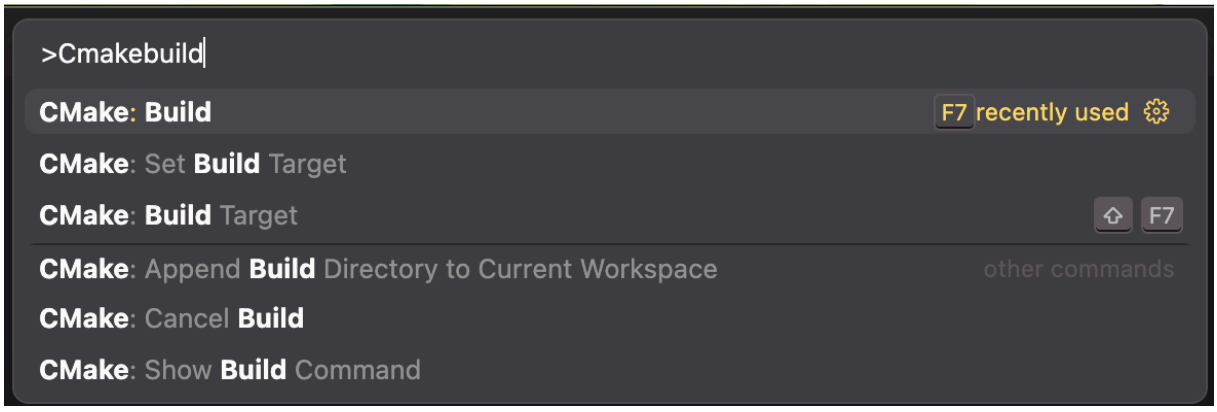
```
cd build
cmake ..
```

Building the project

Once this is setup, we can then build our project within Visual Studio Code.

If you open up the command palette which should be accessible via the following commands:
- `Ctrl+Shift+P` on Windows
- `Cmd+Shift+P` on Mac

You should be able to search for `CMake: Build` and select it.



If successful, you should see that we'll have a our executable in the `build` directory.

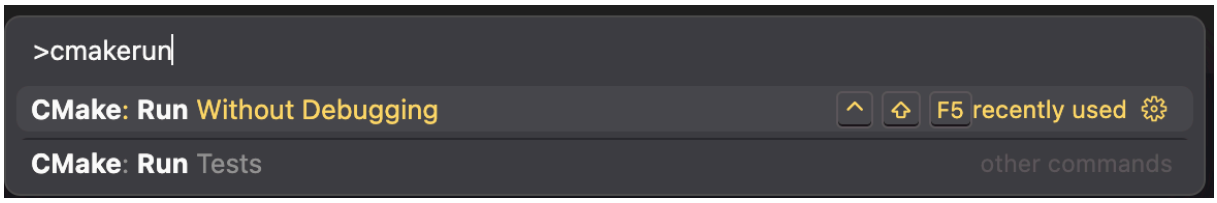
In this case, you're looking for the `hello-world` executable.

Running the program

Now that we have our executable, we can run it by running the following command in the terminal:

```
./build/hello-world
```

However, we also have the option of running the program within Visual Studio Code.



You should then see a new terminal open up in the bottom of the screen with our desired output!