COMP6771
Advanced C++ Programming

Week 1.2
C++ Environment

Why?

- Prepare yourself for the content in this course by:
  - Getting familiar with the basics of Gitlab
  - Getting familiar with the basics of the C++ environment
  - Building our first program
  - Testing our first program

Gitlab

- All of the coding in this course comes from Gitlab.
- If you aren't familiar with Gitlab, we have prepared "lab0" for you.
- It's important you're familiar with git adding, git committing, git pushing, and accepting merge requests.
- https://gitlab.cse.unsw.edu.au

First programs

```cpp
#include <iostream>

int main() {
  // put "Hello world\n" to the character output
  std::cout << "Hello, world!\n";
}
```

We can compile and execute this easily.

```
$ g++ -o hello hello.cpp
$ ./hello
```

First programs

```cpp
#include <iostream>

#include "age.h"

int main() {
  // put "Hello world\n" to the character output
  std::cout << getAge() << "\n";
}

int getAge() {
  return 5;
}
```

age.c

```cpp
int getAge();
```

age.h

We can compile and execute this easily.

```
$ g++ -o age age.cpp
$ ./age
```

First programs

```cpp
#include <iostream>

#include "age.h"

int main() {
  std::cout << getAge() << "\n";
}
```

age_main.c

```cpp
int getAge();
```

age.h

```cpp
#include <iostream>

#include "age.h"

int getAge() {
  return 5;
}
```

age_lib.c

We can compile and execute this too.

Declarations in .h files, definitions in .c files

```
$ g++ -o age age_main.cpp age_lib.cpp
$ ./age
```

The problem with classic compiling

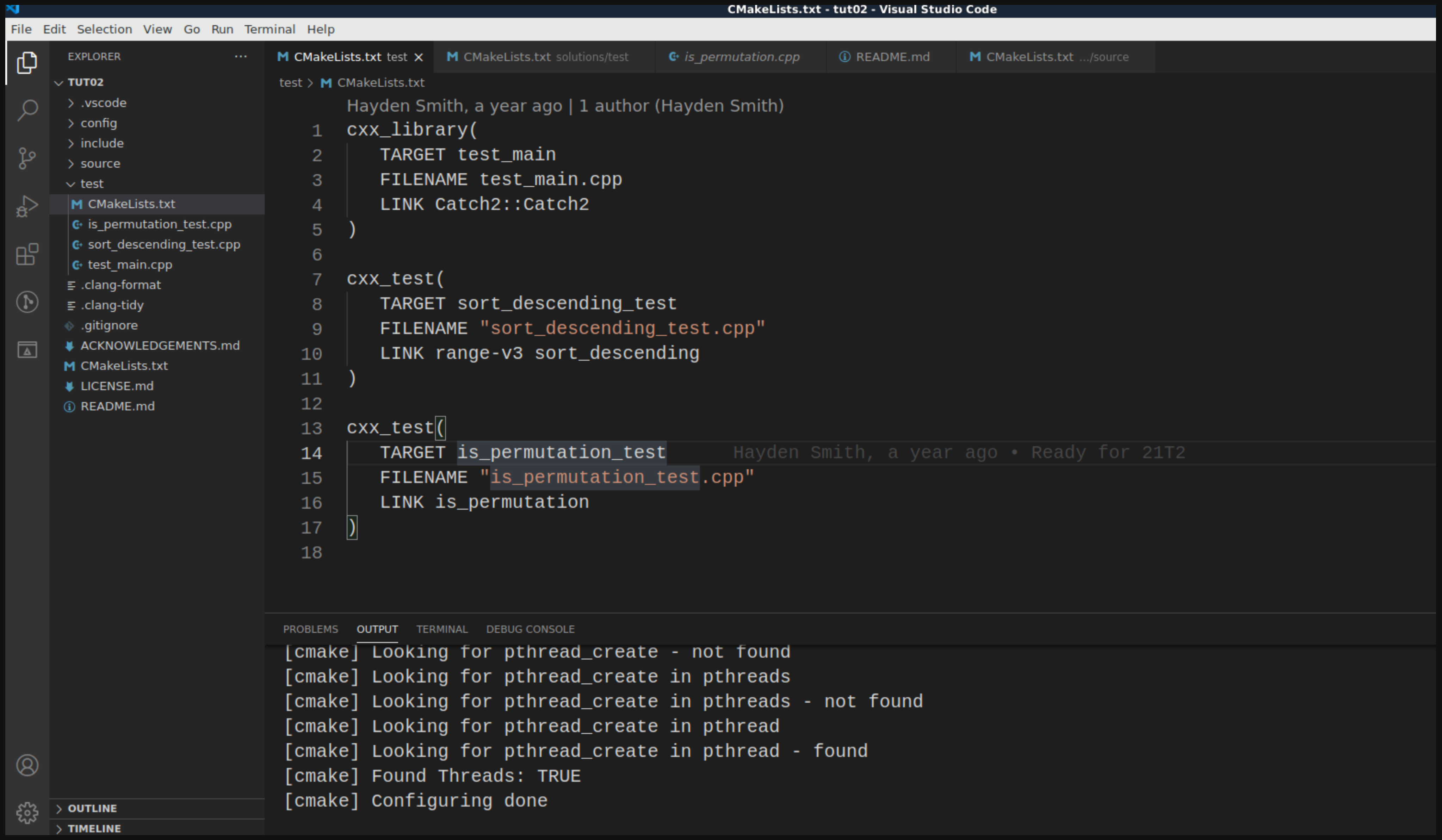
- Imagine having thousands of header and cpp files?
- You have a few options
  - Manually create each library and make sure you link all the dependencies
    - You would have to make sure you linked them all in the right order
  - Create one massive binary and give it all the headers and cpp files
    - Extremely slow
    - Hard to build just parts of the code (eg. To run tests on one file)
  - Makefiles
    - Unwieldy at large scale (hard to read and hard to write)
  - **Any better options?**

Managing larger projects

- The solution to this chaos is to use **build systems.**
- With these systems, you simply have to declare files and relationships between them, and the build system will figure out what to run for you.
- In COMP6771 we will be using CMake for compilation in conjunction with VScode for editing.

Managing larger projects

In COMP6771 we will be using CMake for compilation in conjunction with VScode for editing. We will be using C++20



```
cxx_library(
   TARGET test_main
   FILENAME test_main.cpp
   LINK Catch2::Catch2
)

cxx_test(
   TARGET sort_descending_test
   FILENAME "sort_descending_test.cpp"
   LINK range-v3 sort_descending
)

cxx_test(
   TARGET is_permutation_test
   FILENAME "is_permutation_test.cpp"
   LINK is_permutation
)
```

```
[cmake] Looking for pthread_create - not found
[cmake] Looking for pthread_create in pthreads
[cmake] Looking for pthread_create in pthreads - not found
[cmake] Looking for pthread_create in pthread
[cmake] Looking for pthread_create in pthread - found
[cmake] Found Threads: TRUE
[cmake] Configuring done
```

Managing larger projects

Let's follow instructions in SETUP.md of **tut01** to setup our environment. We can find **tut01** on Gitlab via Webcms3.

The rest of this lecture will be a demo of the basic setup.

Catch2

Catch2 is just one particular framework you can use to test with C++. More information on it can be found here.

Principles of testing

- Test API, not implentation
- Don't make tests brittle
    - If your code changes, your tests should change minimally
- Make tests simple
    - It should be obvious what went wrong
    - Don't put if statements or loops in your tests
    - Any complex code should be put in a well-named function

Feedback