Curses Wrapper

Time it Took Matthew

1.5 hours

What to Submit

A **FOLDER** named CursesWrapper that contains

- A CMakeLists.txt that can compile your program
    - You should start with the one that I've given you in the handout
    - Likely the only changes that you are going to need to make to it are in the add_library command if you add more files to your solution
- All of the .cpp and .h files in the CursesWrapper subfolder that I have given you
- Any additional .cpp and .h files that make up your solution

Problem Description

You may have noticed that when you were working with curses that there is a whole suite of functions that begin with w (wmove, waddch, winch, wgetch, etc). All of these functions accept as their first argument a WINDOW* and perform the operation on that window. I bet that you can see that what the creators of curses were trying to do is implement object-oriented programming in C but unfortunately C doesn't natively support that model. This makes curses much harder to use. So to make using curses easier you are going to create a class called WindowImplementation that wraps a subset of the curses function.

WindowImplementation should publicly inherit from Window, an interface that describes what operations I want you to implement. Most of your code is simply going to forward calls to the appropriate curses functions. You can, of course, add more functionality and members if you want/need to. **ALL** code you create should be inside of the **Curses** namespace.

Fake Curses

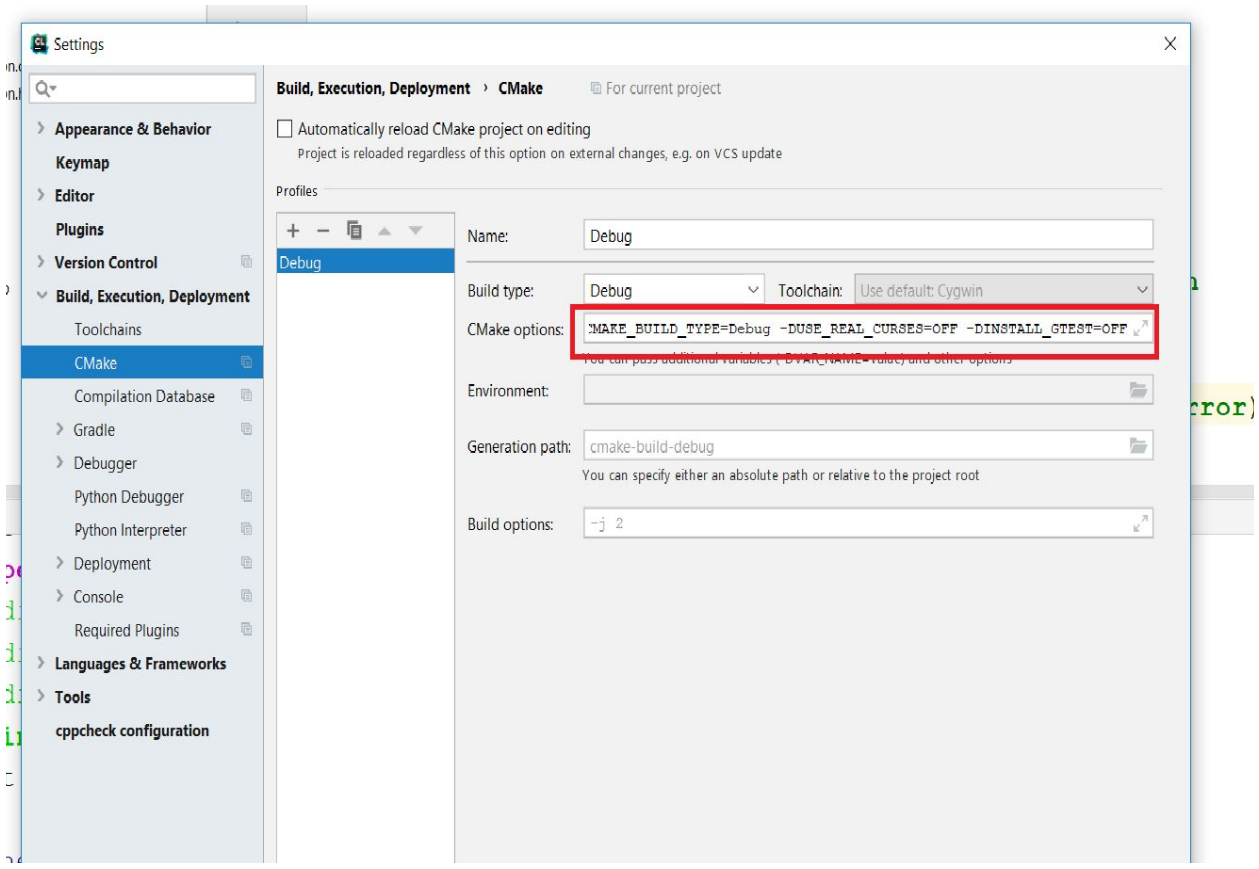
In order to be able to test both your code and mine, I had to create a fake implementation of some of the curses functions. You can find this code inside of the FakeCurses folder. The only functions you are allowed to call from curses are the ones inside FakeCurses.h. In your code, if you make sure to **`#include "ncurses.h"`** and not **`<ncurses.h>`**. The CMakeLists.txt is already set up to switch between real cuses and fake curses based on the CMake option USE_REAL_CURSES as long as **`#include "ncurses.h"`** is used (more on that below).

CMake Options

To run the test cases you must set the following variables in Cmake

- -DUSE_REAL_CURSES=OFF
    - This option controls whether we want to use real curses or use the fake curses code that I've provided you. The Google tests have to use the fake curses code to work so this option must be off for them to work. If you just want to experiment on your own and see if it really does work with curses you can set it to ON but make sure not to run the Google tests if you do

In CLion the area to enter these options is under Settings->Build, Execution,Deployment-> Cmake in the CMake options field.



Build, Execution, Deployment › CMake   For current project

☐ Automatically reload CMake project on editing

Project is reloaded regardless of this option on external changes, e.g. on VCS update

Profiles

Debug

Name: Debug

Build type: Debug   Toolchain: Use default: Cygwin

CMake options: `:MAKE_BUILD_TYPE=Debug -DUSE_REAL_CURSES=OFF -DINSTALL_GTEST=OFF`

You can pass additional variables (-DVAR_NAME=value) and other options

Environment:

Generation path: cmake-build-debug

You can specify either an absolute path or relative to the project root

Build options: -j 2