Concepts of C++ Programming (Exercises)

winter semester 2023

CIIP Team: David Frank, Jonas Jelten

Computational Imaging and Inverse Problems (CIIP)
Technical University of Munich

Tweedback

Tweedback today

The Tweedback session ID today is zj6f, the URL is:

`https://tweedback.de/zj6f`

Contents

Review

Review: Homework 1

In-Class Exercises

Debugging

CMake

Exercise

In-Class Exercise: Preprocessor

Working with the preprocessor

Create a program consisting of a `main.cpp` and `main.h`.

- `main.h` uses #define to define `CLOSE` as }
- Use `CLOSE` instead of } in `main.cpp`
- `main.cpp` uses #include to include `main.h`
- `main.cpp` uses #ifdef CLOSE for returning either 0 or 1.
- Inspect the resulting file after the preprocessor (`g++ -E -P`)

For more information about the preprocessor visit
`https://en.cppreference.com/w/cpp/preprocessor`

Contents

Review

Review: Homework 1

In-Class Exercises

Debugging

CMake

Exercise

Exercise 1

Exercise 1

Set up the assignments repository

- clone the repository to your machine

```
% git clone git@gitlab.lrz.de:cppcourse/ws2023/username_tasks.git
```

- configure pulling from upstream

```
% git remote add upstream git@gitlab.lrz.de:cppcourse/ws2023/tasks.git
```

- verify your setup

```
% git remote -v
origin  git@gitlab.lrz.de:cppcourse/ws2023/username_tasks.git (fetch)
origin  git@gitlab.lrz.de:cppcourse/ws2023/username_tasks.git (push)
upstream  git@gitlab.lrz.de:cppcourse/ws2023/tasks.git (fetch)
upstream  git@gitlab.lrz.de:cppcourse/ws2023/tasks.git (push)
```

Exercise 2

Exercise 2

Build the provided project manually - and fix the missing parts. Invoke the compiler directly in a shell, like you learned in the exercise session.

- in hw01/ you will see some source files

```
% ls
hw01.cpp  library.cpp  test.cpp
```

- build a shared library from library.cpp

```cpp
/**
 * Our precious library function, returning
 * the answer to the Ultimate Question of Life, the Universe, and Everything.
 */
int library_function() {
  return 1337;
}
```

Exercise 2

- build a shared library from library.cpp

  -
    ```
    g++ -std=c++20 -Wall -Wextra -shared -o libmylibrary.so library.cpp
    ```

Inspecting compiler flags

- Open the documentation with `man g++`
- Search the documentation using /
- Go to the next result using `n` and the previous using `N`

```
-std=
Determine the language standard.
```

```
-Wall
    This enables all the warnings about constructions
    that some users consider questionable.
```

```
-Wextra
    This enables some extra warning flags that are not enabled by -Wall.
```

```
-shared
    Produce a shared object which can then be linked
    with other objects to form an executable.
```

```
-o file
    Place the primary output in file file.
```

Exercise 2

- build a shared library from `library.cpp`
  - `g++ -std=c++20 -Wall -Wextra -shared -o libmylibrary.so library.cpp`
  - `ls`

```
hw01.cpp  libmylibrary.so  library.cpp  test.cpp
```

Exercise 2

- create a header file for this library and include it in hw01.cpp, so the library function can be accessed
  - create a header file `library.h`

```
#pragma once
int library_function();
```

  - include it in `hw01.cpp`

```
...
#include "library.h"
...
```

- build `hw01.cpp` and link it to the library:
  - build `hw01.cpp`

```
g++ -std=c++20 -Wall -Wextra -c -o hw01.o hw01.cpp
```

  - link it to the library

```
g++ -std=c++20 -Wall -Wextra -L . hw01.o -lmylibrary -Wl,-rpath '.' -o hw01
```

- run `./hw01`

```
1337
```

Exercise 2

- man g++

```
-Ldir
    Add directory dir to the list of directories to be searched for -l.
-llibrary
-l library
    Search the library named "library" when linking.
    ...
    Some targets also support shared libraries,
    which typically have names like liblibrary.so.
-Wl,option
    Pass option as an option to the linker.
```

- man ld

```
-rpath=dir
    Add a directory to the runtime library search path.
    This is used when linking an ELF executable with shared objects.
```

Exercise 3

Exercise 3

We use doctest to validate the solutions of upcoming homeworks.
Install doctest by manually cloning it.
Doctest can be cloned from GitHub: `https://github.com/doctest/doctest/`.
When one defines `DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN`, doctest will define its own main function. When a program contains a main function, it can be executed!

- Install doctest by manually cloning it.
  - `git clone git@github.com:doctest/doctest.git`
- Build `test.cpp` while including doctest and link it to library:
  - `g++ -std=c++20 -Wall -Wextra -Wl,-rpath '.' -I doctest -o hw01test test.cpp -L . -lmylibrary`

```
-I dir
    Add the directory dir to the list of directories
    to be searched for header files during preprocessing.
```

Exercise 3 (cont.)

- execute `./hw01test`

```
% ./hw01test
[doctest] doctest version is "2.4.8"
===============================================================================
test.cpp:6:
TEST CASE:  testing the library

test.cpp:7: ERROR: CHECK( library_function() == 42 ) is NOT correct!
values: CHECK( 1337 == 42 )

===============================================================================
[doctest] test cases: 1 | 0 passed | 1 failed | 0 skipped
[doctest] assertions: 1 | 0 passed | 1 failed |
[doctest] Status: FAILURE!
```

- fix the code until the test is happy!

Exercise 3 (cont.)

- fix `library.cpp`

```cpp
int library_function() {
  return 42;
}
```

- compile `library.cpp` and build `test.cpp`
- execute `./hw01test`

```
% ./hw01test
[doctest] doctest version is "2.4.8"
[doctest] run with "--help" for options
===============================================================================
[doctest] test cases: 1 | 1 passed | 0 failed | 0 skipped
[doctest] assertions: 1 | 1 passed | 0 failed |
[doctest] Status: SUCCESS!
```

Contents

Review

Review: Homework 1

In-Class Exercises

Debugging

CMake

Exercise

Writing basic programs

- `main` function, variables, types
- initialization, `const`
- blocks, storage durations
- basic loops & control structures (`if`, `for`, `while`, `do`, `switch`)

In-Class Exercise: Simple User Input

Square the input

Create a program that uses std::cin to get a number and print its square.

For more information about `std::cin` visit
`https://en.cppreference.com/w/cpp/io/cin`

In-Class Exercise: Namespaces, Functions and Loops

The loop namespace

Create a program that has the namespace `loop` that contains a function `void looping(int upper);` This function loops from 0 to `upper` printing the current value.

Here you find more information about:

- `https://en.cppreference.com/w/cpp/language/namespace`
- `https://en.cppreference.com/w/cpp/language/for`

In-Class Exercise: Constructing a Struct

A student struct

Create a `student` struct. A student consists of an id, a name, and a study program.

For more information about

- struct visit `https://en.cppreference.com/w/c/language/struct`
- string visit `https://en.cppreference.com/w/cpp/string/basic_string`

Contents

Review

Review: Homework 1

In-Class Exercises

Debugging

CMake

Exercise

Handling Error Messages

- You will encounter error messages when programming:

```
CONFLICT (content): Merge conflict in <filename>
```

```
test.cpp:7: ERROR: CHECK( library_function() == 42 ) is NOT correct!
```

```
g++: error: test.cpp: No such file or directory
```

```
test.cpp:(.text+0x1486b): undefined reference to `library_function()'
collect2: error: ld returned 1 exit status
```

- Strategies for handling error messages:
  1. Read the output – do you understand it?
  2. Use `man` or a search engine to better understand it.
  3. Ask on Zulip
     - State the problem
     - Include a screenshot or the command & output
     - Explain what you tried to solve the error

Handling Error Messages (Example)

```
% g++ -std=c++20 -lmylibrary test.cpp -Wall -Wextra -Wl,-rpath . -o hw01test -L . -I doctest
test.cpp:(.text+0x1486b): undefined reference to `library_function()'
collect2: error: ld returned 1 exit status
```

```
% man g++
-llibrary
...
It makes a difference where in the command you write this option;
the linker searches and processes libraries and object files
in the order they are specified.
Thus, foo.o -lz bar.o searches library z after file foo.o but before bar.o.
If bar.o refers to functions in z, those functions may not be loaded.
```

- `-lmylibrary` was called before `test.cpp`

Debugging

```
test.cpp:7: ERROR: CHECK( library_function() == 42 ) is NOT correct!
```

- Bugs can be localized by debugging:
  - Print debugging using `std::cout`

```cpp
// Inspect the value of library_number
int library_number = library_function();
// 1337
std::cout << library_number << std::endl;
```

  - Using a debugger (gdb, `lldb`)

The GNU Debugger (gdb)

- Run gdb

```
% gdb hw01
Reading symbols from hw01...
(No debugging symbols found in hw01)
(gdb)
```

- Recompile with debugging information -g

```
% man g++
-g  Produce debugging information in the operating system's native format.
     GDB can work with this debugging information.
```

```
% gdb hw01
Reading symbols from hw01...
(gdb)
```

The GNU Debugger (gdb) (cont.)

- Display the current position with `l`

```
(gdb) l
1    // main function to test your work locally
2
3    #include <iostream>
4    #include "library.h"
5
6
7    int main() {
8        int library_number = library_function();
9        std::cout << library_number << std::endl;
10    }
```

- Set a breakpoint using `break file:line`

```
(gdb) break hw01.cpp:8
Breakpoint 1 at 0x11d5: file hw01.cpp, line 8.
```

The GNU Debugger (gdb) (cont.)

- Run using `r`

```
(gdb) r
Starting program: /tmp/username_tasks/hw01/hw01

Breakpoint 1, main () at hw01.cpp:8
8         int library_number = library_function();
```

- Print something using `p`

```
(gdb) p library_function()
$1 = 42
```

The GNU Debugger (gdb) (cont.)

- Step into a function using s

```
(gdb) s
library_function () at library.cpp:5
5    int library_function() {
```

- Execute the next line using n

```
(gdb) n
6        return 42;
```

- Continue the program using c

```
(gdb) c
Continuing.
42
[Inferior 1 (process 1239804) exited normally]
```

Contents

Review

Review: Homework 1

In-Class Exercises

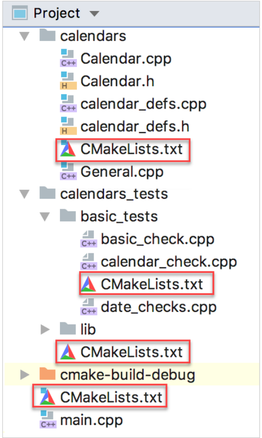
Debugging

CMake

Exercise

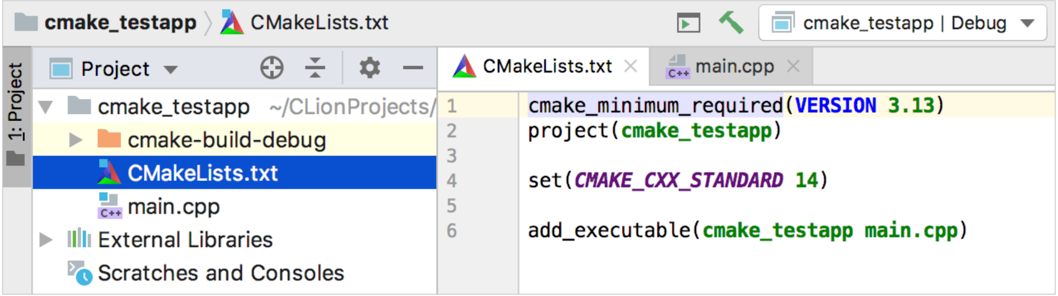
CMake

- meta build system capable of
  - cross-platform and cross-IDE compatibility
  - facilitating linking and other build-time tasks
  - generating out of source builds
- custom cmake programming language
- each project contains a `CMakeLists.txt` script containing instructions
- CMake takes the generic `CMakeLists.txt` and generates the corresponding project files
- output for `make`, `ninja`, Visual Studio, ...



```
Project ▾
▼ calendars
    Calendar.cpp
    Calendar.h
    calendar_defs.cpp
    calendar_defs.h
    CMakeLists.txt
    General.cpp
▼ calendars_tests
  ▼ basic_tests
      basic_check.cpp
      calendar_check.cpp
      CMakeLists.txt
      date_checks.cpp
  ▶ lib
    CMakeLists.txt
▶ cmake-build-debug
  CMakeLists.txt
  main.cpp
```

CMakeLists.txt



```
cmake_minimum_required(VERSION 3.13)
project(cmake_testapp)

set(CMAKE_CXX_STANDARD 14)

add_executable(cmake_testapp main.cpp)
```

- Specifies the required (minimum) version of CMake
- Provides a project name
- Defines the CXX standard to use
- Adds executable targets to the project

Example CMakeLists.txt

```
% cat CMakeLists.txt
cmake_minimum_required(VERSION 3.14)

project(hw01)

# require C++20
set(CMAKE_CXX_STANDARD 20)

set(EXECUTABLE_NAME hw01)
add_executable(${EXECUTABLE_NAME} hw01.cpp)
target_link_directories(${EXECUTABLE_NAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
target_link_libraries(${EXECUTABLE_NAME} library)
```

Example CMakeLists.txt (cont.)

- create a build directory `mkdir build; cd build`
- run CMake `cmake ..`

```
      ...
-- Configuring done
-- Generating done
-- Build files have been written to: /tmp/username_tasks/hw01/build
```

- run make `make`

```
Scanning dependencies of target hw01
[ 50%] Building CXX object CMakeFiles/hw01.dir/hw01.cpp.o
[100%] Linking CXX executable hw01
[100%] Built target hw01
```

- run `./hw01`

```
42
```

Contents

Review

Review: Homework 1

In-Class Exercises

Debugging

CMake

Exercise

Build system

Exercise 1:

Build and run `hw02` using CMake

Permutations and Combinations

Exercise 2:

Extend the functionality of the program to compute permutations and combinations

$$P\left(n, k\right) = \frac{n!}{\left(n - k\right)!} \qquad C\left(n, k\right) = \frac{n!}{k!\left(n - k\right)!}$$