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

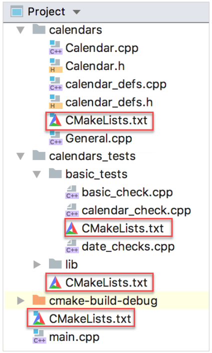
The Tweedback session ID today is zjqm, the URL is:

`https://tweedback.de/zjqm`

Contents

CMake structure

- CMake takes the generic `CMakeLists.txt` files and generates the corresponding project files
- The project contains a `CMakeLists.txt` script containing the general setup
- Each subdirectory contains another `CMakeLists.txt` script containing instructions
- The files define executable and library targets and the links between them



CMake Terminology

- target: a job, like an executable or library
- PUBLIC (default): inherited to linking targets or internal targets[1]
- PRIVATE: property only for internal usage[1]

[1] More details at `https://kubasejdak.com/modern-cmake-is-like-inheritance`

Useful CMake commands

- `add_library(<name> STATIC/SHARED <sources>)`
  create a library `name` consisting of `sources`
- `target_include_directories(<target> PUBLIC/PRIVATE <directories>)`
  include `directories` for compiling `target`
- `target_compile_features(<target> PUBLIC/PRIVATE <feature>)`
  required features for compilation, e.g., C++ 20
- `add_executable(<name> <sources>)`
  create an executable `name` consisting of `sources`
- `target_link_libraries(<target> PUBLIC/PRIVATE <library>)`
  link `target` with `library`

Contents

Build system

Exercise 1:

Build and run `hw02` using CMake

- Create your Homework's `CMakeLists.txt`
- Setup `doctest` for CMake

```cmake
# hw02/CMakeLists.txt
set(SOURCES combinatorics.cpp)

set(LIBRARY_NAME hw02)
set(EXECUTABLE_NAME runhw02)

add_library(${LIBRARY_NAME} SHARED ${SOURCES})
target_include_directories(${LIBRARY_NAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
target_compile_features(${LIBRARY_NAME} PUBLIC cxx_std_20)

add_executable(${EXECUTABLE_NAME} run.cpp)
target_link_libraries(${EXECUTABLE_NAME} ${LIBRARY_NAME})
```

Setup doctest for CMake

- Using the `doctest` GitHub repository (from homework 1)
  - `git clone https://github.com/doctest/doctest.git` - clone doctest
  - `cd doctest` - go to the doctest git repo
  - `mkdir build install` - create two directories
  - `cd build` - switch to `doctest/build`
  - `cmake ..` - run cmake and generate build files for `doctest`
  - `make` - compile the `doctest` code
  - `make install DESTDIR=../install` - create the doctest cmake package
- `cd ../../build` - switch to `username_tasks/build`
- `cmake ..  -Ddoctest_ROOT=../doctest/install/usr/local/` - run cmake
- `make testhw02` - build and link with doctest
- `./tests/hw02/testhw02` - run

Permutations and Combinations

Exercise 2:

Extend the functionality of the program to compute permutations and combinations

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

Factorial

Factorial

The factorial of n is the product of all positive integers less than or equal to n.

$$\begin{aligned} n! &= n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 3 \times 2 \times 1 \\ &= n \times (n-1)! \end{aligned}$$

```cpp
// combinatorics.cpp
int64_t factorial(int64_t val) {
  return val <= 1 ? 1 : val * factorial(val - 1);
}
```

Number of Permutations

Number of Permutations

Number of arrangements of k of n numbers, e.g. three numbers smaller than six.

$$P(n, k) = \frac{n!}{(n-k)!}$$

```cpp
// combinatorics.cpp
int64_t permutation(int64_t val, int64_t val2) {
  if (val < val2) { return 0; }
  return factorial(val) / factorial(val - val2);
}
```

Combination

Combination

Number of arrangements of k of n numbers where the order does not matter.

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

```cpp
// combinatorics.cpp
int64_t combination(int64_t val, int64_t val2) {
  if (val < val2) { return 0; }
  return factorial(val) / (factorial(val2) * factorial(val - val2));
}
```

Testing

- `make runhw02` - Build our executable
- `./hw02/runhw02` - Run our executable

```
perm(10, 3)=720
```

- `./tests/hw02/testhw02` - Run our test code

```
[doctest] doctest version is "2.4.8"
[doctest] run with "--help" for options
===============================================================================
[doctest] test cases:  3 |  3 passed | 0 failed | 0 skipped
[doctest] assertions: 19 | 19 passed | 0 failed |
[doctest] Status: SUCCESS!
```

Contents

Debugging

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

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

```cpp
// Inspect the returnvalue of the function
int result = some_function();
// 1337
std::cout << result << std::endl;
```

  - Using a debugger (gdb, `lldb`)

The GNU Debugger (gdb)

- Run gdb

```
% gdb --args yourprogram
Reading symbols from yourprogram...
(No debugging symbols found in yourprogram)
(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 --args yourprogram
Reading symbols from yourprogram...
(gdb)
```

The GNU Debugger (gdb) (cont.)

- Display the current position with `l` (`list`)

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

- Set a breakpoint using `break file:line`

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

- Or for a function `break somefunctionname`

The GNU Debugger (gdb) (cont.)

- run the program using `r`

```
(gdb) r
Starting program: /tmp/username_tasks/hw42/yourprogram

Breakpoint 1, main () at yourprogram.cpp:8
8     int result = some_function();
```

- print something using `p`

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

The GNU Debugger (gdb) (cont.)

- step into a function using `s`

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

- Execute the `next` line using `n`

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

- `continue` running the program using `c`

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

Contents

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: Constructing a Struct

A student struct

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

Here you find more information about:

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

Contents

String

String

- A sequence of `char`-like objects
- Elements stored contiguosly
- `std::string` declared in `string`
- More information at `https://en.cppreference.com/w/cpp/string`

String (cont.)

- Initialize

```cpp
// #include <string>
std::string a{"Hello"};  // Hello
```

```cpp
std::string b(5, '=');  // =====
```

- Print

```cpp
// #include <iostream>
std::cout << a << '\n';
std::cout << b << '\n';
```

```
Hello
=====
```

String (cont.)

- Loop

```cpp
std::string myString {"abcdefg"};
for (const auto& c : myString) std::cout << c << '\n';
```

```
a
b
c
d
e
f
g
```

In-Class Exercise: String Padding

String Padding

Create a program that pads a string given a `width`, for example:

```
==========
hi
==========
```

Here you find more information about:

- `https://en.cppreference.com/w/cpp/string`

Array

Array

- Encapsulates fixed size arrays
- Contiguosly stored elements
- Size determined at compile time
- `std::array<T,size>` declared in `array`
- More information at `https://en.cppreference.com/w/cpp/container/array`

Array (cont.)

- Initialize

```
// #include <array>
std::array<int, 5> a{1, 7, 2, 3, 9};
```

- Sort

```
// #include <algorithm>
std::sort(std::begin(a), std::end(a));
```

- Print

```
// #include <iostream>
for(const auto& s: a) {
  std::cout << s << ' ';
}
```

```
"1 2 3 7 9 "
```

Vector

Vector

- Automatic, dynamic storage handling
- Contiguosly stored elements
- `std::vector<T>` declared in `vector`
- More information at `https://en.cppreference.com/w/cpp/container/vector`

Vector (cont.)

- Initialize

```cpp
// #include <vector>
std::vector<int> v{7, 5, 16, 8};
std::vector<int> w(200, 5);  // {5, 5, ..., 5}
```

- Loop

```cpp
// #include <iostream>
for (const auto& i : v) {
  std::cout << i << " ";
}
```

```
"7 5 16 8 "  # space at the end
```

Vector Capacity

- Query the current vector capacity using the `capacity()` member function

```
std::vector<int> v{1,2};
std::cout << v.capacity() << '\n';
```

```
2
```

```
v.push_back(3);  // {1, 2, 3}
std::cout << v.capacity() << '\n';
```

```
4
```

```
v.insert(v.end(), {4, 5, 6, 7});  // {1, 2, 3, 4, 5, 6, 7}
std::cout << v.capacity() << '\n';
```

```
7
```

```
v.push_back(8);  // {1, 2, 3, 4, 5, 6, 7, 8}
```

```
14
```

Vector Capacity (cont.)

```cpp
std::cout << v.size() << '\n';
v.erase(v.end() - 7, v.end());  // remove last 7 elements
std::cout << v.size() << '\n';
std::cout << v.capacity() << '\n';
```

```
8
1
14
```

```cpp
v.shrink_to_fit();  // non-binding request to reduce capacity
std::cout << v.capacity() << '\n';
```

```
1
```

Vector Reserve Space

- If we know the size we can reserve it

```cpp
std::vector<int> v;
std::cout << v.capacity() << '\n';  // 0
v.reserve(100);
std::cout << v.size() << '\n';  // 0
std::cout << v.capacity() << '\n';  // 100
```

- Or fill it with a default value

```cpp
std::vector<int> w (100, 5);  // {5, 5, ..., 5}
std::cout << w.size() << '\n';  // 100
std::cout << w.capacity() << '\n';  // 100
std::cout << *(w.end() - 1) << '\n'; // 5
```

Contents

Vector Basics

Homework

Code a contacts list program, through the use of two vectors: one for names and one for numbers. The names and numbers should be kept at matching positions in the vectors:

```
first name  - first number
second name - second number
third name  - third number
...         - ...
```

Functionality

The following functionality is expected:

- Add a contact - disallow empty or duplicate names - return true for success.
- Get the number for a given name - return -1 if no such name is found.
- Add a function which returns the contacts list as string so one can print it.
- Remove a contact - does nothing if requested name was not part of the list - return true for success.
- Sort the contact list by name - watch out to keep the number list synchronized!
- Add a function to get the name that matches a number. return "" when not found.