Chair for System Simulation

Winter Term 2020/2021

Prof. Dr. Harald Köstler, Jan Hönig, Marco Heisig

Advanced Programming Techniques
Sheet 1 — First Steps

This sheet is not mandatory. It is only a preparation for the programming project later in the semester. However, we strongly recommend you to work on all these exercises.

Even though this sheet is not mandatory, we will grade your submissions. The purpose of this is to give you feedback and help you improve your skills.

Exercise 1 — Warm-Up

Please be aware that during the exercises of AdvPT, we will give only support for the Linux machines in the CIP-Pool. You can use them remotely with ssh. If you have never worked with the Linux terminal, we recommend to go threw some tutorials first.

Furthermore we expect you to know essentials like the difference between stack and heap or the use of pointers. We also recommend you to learn about Git, Make and CMake. These tools are invaluable for C++ development.

Here and on StudOn are some links regarding those topics, however feel free to explore this subjects on your own with your favourite search engine.

- Linux Tutorial
- Linux Terminal
- Stack vs Heap
- CMake
- Make by Example
- Make Tutorial
- Valgrind
- gdb Tutorial

a) Range Sum Write a program that queries the user for two numbers and sums the numbers in that range (including the first number, excluding the last number).

b) Factorial Write a program that prompts the user to enter a number and then calculates the approximate factorial of the given number as a double precision floating point number and writes it to the standard output. Verify your program at least against the following *test cases*:

$$0! = 1.0 \quad ; \quad 1! = 1.0 \quad ; \quad 6! = 720.0 \quad ; \quad 12! = 479001600.0$$
$$13! = 6227020800.0 \quad ; \quad 21! \approx 5.1091e19$$
$$35! \approx 1.0333e40 \quad ; \quad -1! = ?$$

c) Punctuation Write a program that reads a line from standard input and prints the line to standard output but with all punctuation removed. The resulting program should be usable as a filter like this:

```
./punctuation < with_punct.txt > no_punct.txt
```

Hint: Have a look at the following STL Header: `<cctype>`

Exercise 2 — Escaping the Grid

In this exercise, you will write a program that finds paths through a simple, two-dimensional grid. Before working on this exercise, you should download and extract the corresponding files from Studon.

Compile the exercise with Cmake and Make. We have shipped the exercies with tests and a main function to execute those.

a) My Very Own Grid Implement your very own two dimensional grid called `MyGrid`.

- Familiarize yourself with the interface that is defined in the abstract base class in `Grid.hpp`.
- Think about how you want to implement this interface, especially what member variables your class should have. Add these member variables in `MyGrid.hpp`.
- Implement a primary constructor that receives three arguments — the number of rows, the number of columns, and a default tile — and that returns a new `MyGrid` instance.
- Implement the copy constructor that takes a reference to an existing `Grid` (not to a `MyGrid`!), and that returns a `MyGrid` with the same size and contents.
- Implement the move constructor, too. The move constructor behaves similarly to the copy constructor, except that it may reuse the storage of the `Grid` whose rvalue reference we receive.
- Implement the destructor. Make sure to free any resources that you allocate in the constructor.
- Implement the assignment operator and a move assignment operator.
- Implement the `rows` and `cols` member functions.
- Implement the `validPosition` member function, and then use it to implement the two `()`-operators. Make sure that your `()`-operators throw an exception of the access is outside of the valid range of rows and columns.
- Implement the `print` member function. It should write the number of rows in the first line, the number of columns in the second line, and then each row of the grid in a separate line with one character per Tile. Use the auxiliary functions `tile_from_char` and `char_from_tile` to convert between tiles and characters.
- Implement the `read` static member function. It should be able to turn a text representation like the one produced by `print` back into a `MyGrid` instance.

If you want us to grade this exercise, you need to upload your `MyGrid.hpp` and `MyGrid.cpp` files (individually or as a zip file) on Studon.

b) Path Finding In this second task, you have to write a program that can "escape" the grid, by finding a path of floor tiles from the tile in the upper left corner (the tile whose row and column is one) to a boundary tile that is not a wall tile. No diagonal moves are permitted. Once you have found such a valid path, draw it to the grid by changing all tiles on the path to `Path`. To do so, implement the `escape` member function of the `Grid` class in the file `escape.cpp`. Don't forget that you can use the entire C++ standard library here.

For the test grid `C` that was included in the project skeleton you have received via Studon, your solution should look like this:

```
8
8
########
#*****.#
#.###*##
#.#.#*##
#.#..**#
#.#####*#
#....#*#
#######*#
```

If you want us to grade this exercise, you need to upload your `escape.cpp` file (individually or as a zip file) on Studon.
Extra challenge: Can you write your program so that it always finds one of the shortest paths out of the room?

Exercise 3 — C++ Variables and Basic Types

Complete the Studon Test "C++ Variables and Basic Types".

Exercise 4 — Function Resolution

Complete the following Studon tests:

a) Function overloading Demonstrate your basic knowledge in the resolution of functions by taking a quick StudOn Quiz ("Function resolution - basics").

b) Inherited functions Demonstrate your advanced knowledge in the resolution of functions by taking a quick StudOn Quiz ("Function resolution - classes and structs").

c) Template function specialization Demonstrate your even more advanced knowledge in the resolution of functions by taking a quick StudOn Quiz ("Function resolution - templates").

d) Mixed concepts Show off your extreme knowledge in the resolution of functions by taking a quick StudOn Quiz ("Function resolution - challenge").