Department of Informatics, Technical University of Munich
Scientific Computing in Computer Science
G. Chourdakis, J.-M. Gallard, F. Simonis

Winter term 2019/20

Advanced Programming Tutorial
Worksheet 0: Hello, tutorial!

Keywords: compiler, debugger, streams `std::cout` and `std::cin`, `std::string`

Note: The tasks marked with "**Idea**" are meant as optional ways that you can further explore a concept. It is fine if you don't get enough time to solve them or if you don't yet know the prerequisites, but we can also not present detailed solutions to these tasks.

Exercise 1: Hello World

Hello world is one of the most traditional introductions to almost every programming language. Therefore, we may not miss it in our tutorial!

```
1 #include <iostream>
2
3 int main(int argc, char *argv[])
4 {
5         std::cout << "Hello World!" << std::endl;
6         return 0;
7 }
```

1. Build and execute a "hello world" program using the command line of your operating system. For example: `g++ -o hello helloWorld.cpp` and `./hello`.
2. Build and execute a "hello world" program using an Integrated Development Environment.
3. Get familiar with your IDE: Where can you see the executed commands? How can you change the invoked compiler? What other features does it offer?
4. **Idea:** Read error messages patiently: Break your code to give you syntax errors. What does the compiler report?
5. **Idea:** What does `std::endl` do internally? Try also `std::flush` and `std::clog`.

Exercise 2: Hello Debugger

As soon as you get an error or wrong results in your program, you will need to look closer into your code and find the problematic line. Instead of printing values, you can use a Debugger to execute your code step-by-step.

1. Build and run the following code:

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
 std::string greeting = "Hello Debugger!";
 return 0;
}
```

2. To debug in your IDE, click next to line 5 to add a breakpoint. Then, instead of "Run" click "Debug". This will show you additional buttons, including a button to "step over" each line. On a separate pane, you can see the values of each variable, including `greeting`.
3. Keep stepping through the code until the program exits. What do you observe?
4. **Idea:** Repeat using GCC and GDB. Compile with `g++ -g -o hello helloDebugger.cpp`, debug with `gdb hello`, `break 5`, `run`, `step`, `quit`.

Exercise 3: Hello Person

We want to enhance the *Hello World* example to greet actual persons. We can get input from the user and store it in a string as follows:

```cpp
std::string s;
std::cin >> s;
```

1. Get the user's first and last name and display the message
   `Hello first_name last_name!`
2. **Idea:** Ask multiple users and store all their names in a file.
   Additional keywords: `std::ofstream`, loops.
3. **Idea:** Create a class `Person`. Use the constructor to prompt the user for a first name and a last name, and initialize the respective member variables of type `std::string`. Additional keywords: classes, constructors.

Tips on programming and compilers

- It is recommended to work with an IDE and a GCC compiler in Linux. Using the same compiler is mainly important for our code optimization exercises that will follow.
- Using an advanced editor is also enough. Share your favorite editor/IDE with the class!
- Use the `-Wall` flag during compilation and don't ignore the warnings. Warnings at their best could prevent better code optimization by the compiler; and at their worst could be errors in disguise.
- Comment your code extensively and use descriptive names. You will thank yourself later.
- Always read and try to fix the very first error in a list, before continuing.
- Try to always use a version control system (such as Git), even for your small private projects. We will talk about this later.
- More good practices for C++: https://isocpp.github.io/CppCoreGuidelines/

Compilers

We need a compiler with C++14 support:

- If you are using Linux, make sure you have at least GCC 6.1.
- If you are using Windows, you can install the Windows Subsystem of Linux or MinGW.
- If you are using macOS, Clang 3.4 should also suffice.
- Other options also exist (e.g. Microsoft Visual C++, Intel C++ Composer, etc).

IDEs / Editors

You are free to use your favorite editor/IDE. Some ideas:

- IDEs: Clion, Visual Studio Code (with the C++ extension), Eclipse (with CDT), QtCreator, Code::Blocks, Visual Studio (Windows), Xcode (macOS), ...
- Editors: Atom, Vim (command-line), Emacs (command-line), ...

Make sure that your IDE is correctly configured and can find the compiler.

Recap

Discuss the following questions:

- What is a compiler?
- What are differences between Matlab and C++?
- What are differences between C and C++?
- What is a preprocessor?
- How can we give output to the user?
- How can we get input from the user?
- How can we check data values without printing them?