Systems Engineering and Data Processing in C++
Summer Term 2021

Hasso Plattner Institute
Computer Graphics Systems Group

Prof. Dr. Jürgen Döllner
and Willy Scheibel

Assignments #1: Getting Started

Objectives

In your first assignments you will learn how to:

- use procedural concepts to implement basic algorithms in C++,
- get familiar with implementation specifics, their characteristics, and influence on the quality of an implementation,
- compare different implementations against run-time,
- use user-defined constructs to extend the C++ language, and
- compose a small C++ software system.

Rewards

For each successfully solved task you and your partner (optional) gain one bonus exam point. A serious shortcoming in fulfilling the requirements of a task will result in no bonus point regarding this task. You can gain at most 4 bonus points through exercise sheet.

Submission

All described artifacts has to be submitted to the course moodle system[1] as a zipped archive. The naming convention includes the assignment number and your *personal assessment numbers*[2] following naming convention: `assignment_1_paNr1.zip` or `assignment_1_paNr1_paNr2.zip` whether or not pair programming was applied. Compiled, intermediate, or temporary files should not be included. If pair programming is used, the results are turned in only once. The assignments #1 are due to the next tutorial on Mai 6th, 9:15 a.m. GMT+2.

Presentation

Feel free to present any of your solutions during the Zoom meeting in the upcoming tutorial. If you want to present your solution of a task, please send the source code beforehand via e-mail to willy.scheibel@hpi.de.

General Instructions

Pair Programming On these assignments, you are encouraged (not required) to work with a partner, provided you practice pair programming. Pair programming "is a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test." One partner is driving (designing and typing the code) while the other is navigating (reviewing the work, identifying bugs, and asking questions). The two partners switch roles every 30-40 minutes, and on demand, brainstorm.

Cross-Platform The assignments can be solved on all major platforms (i.e., Windows, Linux, macOS). The evaluation of submitted assignments can be carried out on any of these platforms. All results are required to be cross-platform, that is, they compile successfully and run indifferently with respect to their input and output.

Violation of Rules a violation of rules results in grading the affected assignments with 0 points.

- Writing code with a partner without following the pair programming instructions listed above represents a serious violation of the course collaboration policy.
- Plagiarism represents a serious violation of the course policy.

[1] https://moodle.hpi.de/course/view.php?id=174
[2] Information on how to obtain your personal assessment numbers are published on the Moodle, soon.

Task 1.1: Approximation of π

Relevant File: `montecarlopi/montecarlopi.cpp`

Description: Implement a command-line program that computes an approximation of the value of π. For this purpose, you should use a variant of the "quarter of circle" algorithm, based on a Monte-Carlo approach:

Generate 2D points (x, y) in the unit square $[0, 1] \times [0, 1]$ (e.g., uniform distribution). Count how many of them are inside the corresponding quarter of the unit circle (distance to $(0, 0)$ is less than 1). The ratio between points inside the circle and the total number of points approximates the value of $\frac{\pi}{4}$. Because *many* iterations (the recomputation of π with one more sample point as before) are required, take care of arithmetic issues.

- How many iterations does your implementation need to get a medium error of $< 0.01\%$ for the last 10000 iterations (means the result is within $[3.1413, 3.1419]$)?
- List the most exact π representation using the C++ data-types `float`, `double`, and `long double` along with their storage in bytes for your system. How many bits of mantissa in a floating-point number are necessary to encode the first 20 decimal digits of π?
- You may use random-number generators or apply other sampling-based strategies to compute the ratio.
- For your implementation, use only elements of standard C++20 and, if needed, compiler extensions.
- **Do not** use numerics libraries or any other third-party libraries or classes that provide specialized floats.

Submission Artifacts:

a) `montecarlopi.cpp` : Source code of the solution.

b) `montecarlopi.txt` : Results of the theoretical tasks (varying precisions for π).

Objectives: Procedural C++ elements, standard library functions, C++ floating-point arithmetics, random numbers.

Task 1.2: Palindrome Checker

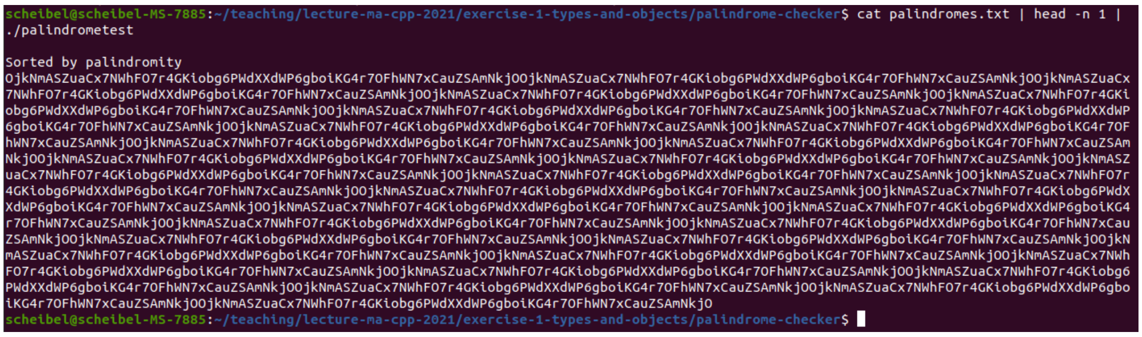
Relevant Files: `palindrome-checker/palindrome-checker.cpp`, `palindrome-checker/palindromes_medium.txt`, `palindrome-checker/palindromes_small.txt`

Description: Implement a command-line program that checks per-line input for their property of being a palindrome. The output lists all input lines, while all lines being palindromes are listed first. If two lines belong to the same category (palindrome, not palindrome), their order does not have to be the same as in the input; the sorting may be *unstable*.

Submission Artifacts:

a) `palindrome-checker.cpp` : Source code of the solution.

Example:



```
scheibel@scheibel-MS-7885:~/teaching/lecture-ma-cpp-2021/exercise-1-types-and-objects/palindrome-checker$ cat palindromes.txt | head -n 1 |
./palindrometest

Sorted by palindromity
OjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx
7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKi
obg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP
6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OF
hWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAm
NkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZ
uaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r
4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdX
XdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4
r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCau
ZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkN
mASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWh
FO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6
PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gbo
iKG4r7OFhWN7xCauZSAmNkjOOjkNmASZuaCx7NWhFO7r4GKiobg6PWdXXdWP6gboiKG4r7OFhWN7xCauZSAmNkjO
scheibel@scheibel-MS-7885:~/teaching/lecture-ma-cpp-2021/exercise-1-types-and-objects/palindrome-checker$ █
```

Competition: This is a competitive task with respect to program run time. If you want to compete in the open competition, submit your isolated source code of this task via e-mail to willy.scheibel@hpi.de until Mai 4th. Your source code is measured against all other submissions using the same palindrome dataset. The run time is measured using the `time` shell command. Willy will provide his solution as binary for a basic benchmark beforehand and will disclose his approach in the upcoming tutorial.

Objectives: Procedural C++ elements, standard library functions.

Task 1.3: Provider Information Library

Relevant Files: `library/provider.cpp`, `library/provider.h`, `library/providerlibtest.cpp`

Description: Implement a C++ library that allows access to the library's provider name and, conditionally, the creation date and time of the library. The library should

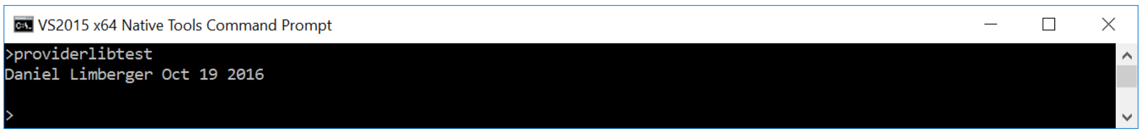
- declare a function `std::string providerInfo(bool date = false)` in a header file (provider.h);
- implement the corresponding function (provider.cpp); take your name(s) as provider name to be passed as compile-time arguments and use appropriate C++ built-in macros for build date and time;
- generate a dynamically linkable library (.dll, .so, or .dylib);
- handle symbol visibility explicitly in your library headers (i.e., handle `dllimport` and `dllexport` on Windows as well as `__attribute__ ((visibility ("default")))` on unixoid systems);
- compile `providerlibtest.cpp` without any modifications to the source code to test your library.

Submission Artifacts:

a) `provider.cpp`, `provider.h` : Source code of the solution.

b) `provider_make.txt` : The full compilation command used for generating the library and compilation of the test and linkage of the library for at least one platform.

Example:



```
>providerlibtest
Daniel Limberger Oct 19 2016

>
```

Objectives: C++ library building process and usage; predefined macros of C++.

Task 1.4: C++ Syntactic Sugar

Relevant File: `syntactic_sugar/syntax.cpp`.

Description: Consider the sample code which uses elements called "syntactic sugar" of C++. Provide class, struct, and function interfaces and implementations such that the example code compiles successfully and ensure that all assertions are valid (you should compile in debug mode as release mode typically disables assertions) when running the program. You *must not* change the given test code and the assertions. However, you may add additional header and source files if you want to modularize your solution.

Hint for the MSVC compiler: language extensions must be disabled by adding the `/Za` flag to the command line.

Submission Artifacts:

a) `syntactic_sugar/syntax.cpp`, `syntactic_sugar/...` : Source code of the solution.

Objectives: User-defined operators, user-defined literals, test-cases.