Draft der C++-Vorlesung vom 22.März 2024

1. Strukturierte Programmierung: Siehe [Haase23, §4].
   Kurze Demo der Loops-Examles 0, 1, 2, 3: Kommentare, Einrückungen etc.

2. Demonstration der Nutzung von `string`, `vector`, `array` [Haase23, §2.3.1-2.3.3].
   Code: demoString[1], demoVector[2]. strings_c_cpp[3]

3. Beispiel Geheimzahl (Errate Zahl im Intervall), Struktogramm[4] an Tafel.
   Code: v_2b[5] mit docu[6].
   - Funktionen `eingabe` und `geheimZahl`.
   - Umsetzung des Struktogramms in `geheimZahl` unter Verwendung der Funktion `eingabe`.
   - Unterscheide *Deklaration* (Prototype [Ankündigung]) von *Definition* (Implementierung) einer Funktion.
   - Zufallszahlengenerierung (`main.cpp:76`) im Intervall $[anf, ende]$ noch mit C-Zufallszahlen (`#include <cstdlib>`).

4. Dynamischer Vektor [Haase23, §5.1.1] und dessen Nutzung als Input- oder Output-parameter in Funktionen.
   Code: v_2c[7] mit docu[8].

5. Beispiel Geheimzahl mit Speicherung aller Versuche (dynamischer Vektor) und einem Vektor von Strings zur flexiblen Ausgabe.
   Code: v_3a[9] mit docu[10].
   - Auch mit `array` [Haase23, §5.1.2] von Strings möglich (`main.cpp:51`).
   - Zufallszahlengenerierung (`main.cpp:108-112`) im Intervall $[anf, ende]$ mit C++-Zufallszahlen `#include <random>`, zur Vertiefung[11].
   - Benötigt die Compileroption `-std=c++11` (oder höheres wie `-std=c++17`)

6. Trennung in Header- und Source-Files (Deklaration vs. Definition)
   Signum-Fkt. im Beispiel v_4c[12]: [clever[13]]
   Um mehrfaches Einbinden desselben Headerfiles in einem Sourcefile zu verhindern: Voranstellen von
   `#pragma once`[14]
   Das ältere Header Guarding via
   `#ifndef FKT_H_INCLUDED #define FKT_H_INCLUDED ... #endif` ist nicht mehr nötig.

7. **Signatur** einer Funktion:
   Dient der eindeutigen Zuordnung einer Funktion beim Aufruf.
   `namespace::classname::functionname (inputParamList, outputParamList)`

[1] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/demoString.cpp
[2] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/demoVector.cpp
[3] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/strings_c_cpp.zip
[4] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/5_PDFsam_vorlesung_1_2.pdf
[5] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_2b.zip
[6] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_2b/html
[7] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_2c.zip
[8] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_2c/html
[9] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_3a.zip
[10] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_3a/html
[11] https://en.cppreference.com/w/cpp/header/random
[12] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS24/v_4c.zip
[13] https://stackoverflow.com/questions/1903954/is-there-a-standard-sign-function-signum-sgn-in-c-c

- Return Type einer Funktion ist für Signatur dieser irrelevant.
- Übergabe eines Parameters via Copy oder via Referenz erzeugt die identische Signatur, da die zu übergebenden Parameter identisch sind.
- Eine Signatur ist der globale, eindeutige Bezeichner Ihrer Funktion.
- Sollen Funktionen auf die Übersetzungseinheit beschränkt bleiben (also noch global), dann ist der Funktionsdeklaration ein `static` voranzustellen.

Literatur

[Haase23] Gundolf Haase: Einführung in die Programmierung mit C++ (2023), *www*[15].

[Stroustrup10] Bjarne Stroustrup: Einführung in die Programmierung mit C++. Pearson Studium, München (2010).

[15] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf`