Programmieren in C++ Versuch 3

Prof. Dr. R. Manzke, Sommersemester 2022

Zielsetzungen

- OO Analyse und Design in C++
- Eigenständige Vertiefung des C++ Wissens anhand eines freieren Projekts

Zeitrahmen

- Es besteht Anwesenheitspflicht zu den Laboren.
- Versuch 3 soll zum 5. Labortermin abgenommen werden. Es stehen insofern 4 Wochen Bearbeitungszeit zur Verfügung.
- Jedes Team muss zum 4. Labortermin kurz seinen Status präsentieren (max. 3 Minuten pro Team).
- Die Abnahme erfolgt durch das Laborpersonal zum 5. Termin.
- Eine erfolgreiche Abnahme während der Laborzeit ist notwendig für die Erteilung des Labortestats.
- Der fertige Quellcode soll im Moodle abgegeben werden.
- Die Aufgabe kann im Team von max. 2 Personen oder alleine bearbeitet werden.

Aufgabenstellung

Sie können selbst einen Projektvorschlag unterbreiten oder bspw. eine grafische Anwendung bspw. mit der Bibliothek DearImGUI bauen.
Es müssen in jedem Fall folgende Anforderungen an ihr C++ Projekt erfüllt sein:

Anforderung	Beschreibung	Erfüllt
1	Erzeugung eines ausführbaren C++ Programms.	
2	Erzeugung einen C++ Bibliothek und Einbindung dieser in ihr Programm mit sinnvoller Modularisierung und eigenem Namespace.	
3	Nutzung einer externen C++ Bibliothek und Einbindung dieser in ihr Programm (Extern bedeutet hier eine Bibliothek, die nicht Teil der C++ Standardbibliothek ist).	
4	Nutzung von CMake.	

5	Objektorientierte Analyse einer realen Problemstellung und Umsetzung einer dazu passenden objektorientierten Architektur in C++.	
6	Nutzung von Funktionsüberladung.	
7	Nutzung von Operatorüberladung.	
8	Datei und Konsolen IO.	
9	Verwendung von C++ Smart Pointern (unique_ptr, shared_ptr) bei dynamisch erzeugten Objekten.	
10	Verwendung von weiteren Standard Containern zusätzlich zu std::vector und std::array.	
12	Nutzung von Laufzeitpolymorphismus.	
13	Nutzung von move-Semantik.	
14	Vermeidung von “rohen” Zeigern (Heapallokation mit new/delete).	
15	Erstellung von eigenen Template-Klassen und/oder -Funktionen.	
16	Debugging ihrer Anwendung mittels Debugger.	
17	Profiling ihrer Anwendung mittels Profiler.	
18	Versionskontrolle mit Git.	
19	Durchführung einer statischen Code Analyse (linting) des selbst geschriebenen C++ Codes mit Clang-Tidy. Hier können bspw. die Funktionalitäten aus der CLion IDE genutzt werden. Fehler bzw. Verbesserungsvorschläge aus der Code Analyse sollen entsprechend sinnvoll umgesetzt werden (nicht alle Vorschläge des Tools machen immer Sinn)!	
20	Mindestens 700 Codezeilen netto (d.h. ohne Kommentare und externe Bibliotheken = selbst geschriebener Code) pro Teammitglied. Ermittelt bspw. mit Metrics Plugin für Clion oder https://github.com/AlDanial/cloc	