Prüfungsleistung - Programmentwurf "Labor Softwareentwicklung I"

Inhalt der Prüfung

Entwerfen Sie ein Programm zur Verwaltung einer Bibliothek für Studierende. Dazu soll eine Hauptanwendung zur Verfügung gestellt werden, die verschiedene Interaktionen zur Verfügung stellt. Entsprechende Details finden sich im Kapitel Vorgaben zur Arbeit. Die Verwendung von *CMake*, *Git*, *gtest* und andere Softwarewerkzeuge fließen dabei genauso in die Bewertung wie der Quellcode selbst. Näheres dazu finden Sie im Kapitel Bewertungskriterien.

Bearbeitung und Abgabe

Bearbeitung

Die Bearbeitung soll in Gruppen von zwei Personen erfolgen. In Einzelfällen kann eine Bearbeitung auch einzeln durchgeführt werden. Dies sollte jedoch vorher abgeklärt werden.

Voraussetzungen

Auf der moodle-Seite des Kurs muss vorab:

1. Ein Gruppe für den Programmentwurf gewählt werden
2. Die Aufgabe *Bestätigung der Abgabeform* abgeschlossen sein

Bearbeitungszeitraum

- Der Bearbeitungszeitraum beginnt am Mittwoch, den **01.03.2023**.
- Der Bearbeitungszeitraum endet am Freitag, den **24.03.2023 um 20.00Uhr**.
  - Änderungen die nach 20.00Uhr eingereicht werden können nicht mehr gewertet werden

Abgabe

Die Abgabe der Arbeit erfolgt wie in der moodle-Aufgabe *Bestätigung der Abgabeform* angegeben. Wurde als Abgabeform ein Link zur Verfügung gestellt, so ist keine weitere Aktion erforderlich. Wurde die Abgabe in Dateiform gewählt, muss ein zip/tar Archiv mit allen Inhalten auf der moodle-Seite des Kurs hochgeladen werden.

Zur Verfügung gestellter Quellcode

Folgender Quellcode wird für den Programmentwurf vorab zur Verfügung gestellt:

Modul **book**

Für das Modul **book** werden vorab folgende Teile zur Verfügung gestellt

1. Struktur **book** so wie die für die Struktur **book** notwendigen Funktionsdeklarationen befinden sich in der Datei *book.h*. Diese Datei ist vollständig und muss im Zuge der Arbeit NICHT ergänzt werden.
2. Die reinen Rümpfe der Funktionsdefinition der notwendigen Funktionen für die Struktur **book** befinden sich in der Datei *book.c*. Diese Datei ist nicht vollständig und muss im Zuge der Arbeit ergänzt werden.
3. Die notwendigen Tests für die Funktionen der Struktur **book** befinden sich in der Datei *book_test.cpp*. Diese Datei ist vollständig und muss im Zuge der Arbeit NICHT ergänzt werden.

Modul **student**

Für das Modul **student** werden vorab folgende Teile zur Verfügung gestellt

1. Struktur **student** so wie die für die Struktur **student** notwendigen Funktionsdeklarationen befinden sich in der Datei *student.h*. Diese Datei ist vollständig und muss im Zuge der Arbeit NICHT ergänzt werden.
2. Die reinen Rümpfe der Funktionsdefinition der notwendigen Funktionen für die Struktur **student** befinden sich in der Datei *student.c*. Diese Datei ist nicht vollständig und muss im Zuge der Arbeit ergänzt werden.
3. Die notwendigen Tests für die Funktionen der Struktur **student** befinden sich in der Datei *student_test.cpp*. Diese Datei ist vollständig und muss im Zuge der Arbeit NICHT ergänzt werden.

Modul **library**

Für das Modul **library** werden vorab die leeren Dateien *library.h*, *library.c* und *library_test.cpp* zur Verfügung gestellt. Der Inhalt dieser Dateien muss im Zuger der Arbeit ergänzt werden.

Vorgaben zur Arbeit

Hauptanwendung

Die Hauptanwendung dieses Programms soll nach dem Start folgende Möglichkeiten zur Interaktion anbieten:

```
========================
Choose your library action:
Print library           [1]
Add student             [2]
Add book                [3]
Lend book               [4]
Return book             [5]
List books for student  [6]
Find a book             [7]
Exit                    [8]
========================
```

Die Bedeutung der einzelnen Aktionen soll sein:

Print library

Die aktuell im Speicher vorhandene Struktur der Bibliotheksdatenbank soll auf der Befehlszeile ausgegeben werden. Die Ausgabe soll alle vorhanden Bücher und Studenten umfassen.

Add student

Diese Aktion soll einen neuen Studierenden in der Datenbank hinzufügen. Dazu soll zur Eingabe eines Names auf der Befehlszeile aufgefordert werden. Die Eingabe von Leerzeichen muss dabei nicht unterstützt werden. Beim anlegen eines neuen Studierenden soll die notwendige Matrikelnummer eine (Pseudo-)Zufallszahl sein.

Add book

Diese Aktion soll ein neues Buch in der Datenbank hinzufügen. Dazu soll zur Eingabe eines Titels auf der Befehlszeile aufgefordert werden. Die Eingabe von Leerzeichen muss dabei nicht unterstützt werden. Beim anlegen eines neuen Buchs soll die notwendige ID eine (Pseudo-)Zufallszahl sein.

Lend book

Diese Aktion soll in der Datenbank hinterlegen, dass ein bestimmtes Buch von einem Studierenden ausgeliehen wurde. Dazu soll zur Eingabe einer Buch-ID und einer Matrikelnummer aufgefordert werden.

Return book

Diese Aktion soll in der Datenbank ein Buch nach einer Rückgabe wieder verfügbar machen. Dazu soll zur Eingabe einer Buch-ID aufgefordert werden.

List books for student

Diese Aktion soll für einen bestimmten Studierenden alle ausgeliehenen Bücher auf der Befehlszeile ausgeben. Die Ausgabe soll die Details des Buches und nicht nur die ID umfassen. Dazu soll zur Eingabe einer Matrikelnummer aufgefordert werden.

Find a book

Diese Aktion soll ein bestimmtes Buch in der Datenbank finden und auf der Befehlszeile ausgeben, ob und von wem das Buch derzeit ausgeliehen ist. Die Ausgabe soll die Details des entsprechenden Studierenden auf der Befehlszeile ausgeben. Dazu soll zur Eingabe einer Buch-ID aufgefordert werden.

Exit

Diese Aktion beendet die Hauptanwendung. Dabei wird der gesamte Speicher gelöscht. Es muss kein persistenter Speicher für die Datenbank implementiert werden.

Implementierungsdetails

Modul book

Ergänzen Sie für das Modul *book* die Funktionsrümpfe in der Datei **book.c** so, dass die vorab zu Verfügung gestellten Tests erfolgreich durchlaufen werden. Die Aufgabe der Funktionen sind:

- *book_init*: Initialisieren einer übergebenen Struktur *book*. Details zur Implementierung sind selbst zu erarbeiten.
- *book_print*: Ausgabe einer übergebenen Struktur *book* auf der Befehlszeile. Details zur Implementierung sind selbst zu erarbeiten.

Modul student

Ergänzen Sie für das Modul *student* die Funktionsrümpfe in der Datei **student.c** so, dass die vorab zu Verfügung gestellten Tests erfolgreich durchlaufen werden. Die Aufgabe der Funktionen sind:

- *student_init*: Initialisieren einer übergebenen Struktur *student*. Details zur Implementierung sind selbst zu erarbeiten.

- *student_print*: Ausgabe einer übergebenen Struktur *student* auf der Befehlszeile. Details zur Implementierung sind selbst zu erarbeiten.

Modul library

Für die Implementierung des Moduls *library* ist noch kein Quellcode vorgegeben. Entsprechend sind für das Modul zu entwerfen:

1. Die Struktur *library*
2. Die notwendigen Funktionsdeklarationen für die Struktur *library*
3. Die notwendigen Funktionsdefinitionen für die Struktur *library*
4. Die notwendigen Tests für die Funktionen der Struktur *library*. Der Umfang der Tests sollte ausreichend sein um die grundsätzliche Funktion nachzuweisen. Offensichtliche Sonderfälle (bspw. NULL-Pointer oder falsche Eingaben) sollen überprüft werden.

Hinweis: Eine möglicher (eventuell nicht vollständiger) Umfang an Funktionen könnte sein

```
void library_init(...);
student const *library_add_student(...);
book const *library_add_book(...);
void library_lend_book(...);
void library_return_book(...);
student const *library_find_book(...);
void library_list_books(...);
void library_list_students(...);
void library_list_books_4_student(...);
void library_print(...);
```

Allgemeine Vorgaben

Zu den allgemeinen Vorgaben zum Programmentwurf gehören:

- Das gesamte Projekt soll unter Verwendung von *CMake* realisiert werden.
- Die Struktur des Projektes soll der in der Vorlesungen erarbeiteten Struktur entsprechen (*include;src;test*)
- Die Tests sollen unter Verwendung der Testbibliothek *google-test* (gtest) geschrieben werden
- Das Projekt soll während der gesamten Bearbeitungszeit mittels des Versionsverwaltungssystems *Git* versioniert werden. Die Verwendung von GitHub ist empfohlen, jedoch nicht notwendig.
- IDE, Betriebssystem, Compiler etc. werden nicht näher vorgegeben, das Kapitel Bewertungskriterien sollte jedoch Beachtung finden.

Bewertungskriterien

- Die relevanten Alternativen für die Zielumgebung sind:
    - Suse Installation im Hörsaal der Vorlesung
    - Suse Installation in der zur Verfügung gestellten virtuellen Maschine (VirtualBox)
    - Virtuelle Umgebung
        - Der im learn-git Repository bereitgestellte *Codespace*
        - Der im learn-git Repository bereitgestellte *Gitpod Container*
        - relevante Compiler Alternative:

        - GCC 9.4.0
        - Clang 10.0.0

- Die Projektsprache ist **ENGLISCH**
- Das Projekt kann mittels CMake (Version >= 3.20) ohne Fehler und Warnungen konfiguriert und erzeugt werden.
- Das Projekt kann mittels eines der zulässigen Compiler ohne Fehler und Warnungen übersetzt werden.
- Die Fortschritte des Projektes sind anhand der Git-Historie klar zu erkennen und die Beiträge aller Beteiligen sind voneinander abzugrenzen.
- Git Commits sind in **ENGLISCH** und die Beschreibung ist aussagekräftig
- Der Quellcode ist ausreichend in **ENGLISCH** dokumentiert
- Der Quellcode des gesamten Projektes ist einheitlich formatiert
- Der Quellcode wurde vollständig in **ENGLISCH** geschrieben
- Variablen und Funktionsnamen sind deskriptiv und verständlich
- Variablentypen werden entsprechend dem Verwendungszweck gewählt
- Die Projektstruktur (Ordnerstruktur) entspricht der Konvention aus der Vorlesung
- Der Quellcode ist im notwendigen Umfang unter Verwendung von *gtest* getestet.
- Die *const correctness* wurde beachtet
- **Empfehlungen**
    - Die Aufteilung von Arbeitspaketen auf Entwickler:innen ist oftmals hilfreich und sinnvoll
    - Die Entwicklung einzelner Arbeitspakete in gesonderten *Git-Branches* vereinfacht oftmals Vieles
        - Ein regelmäßiger Merge/Rebase mit der Mainline verhindert den *großen Konflikt (Big Bang)* am Ende
    - Konsequente Anwendung der Formatierungsrichtlininen vereinfacht die Zusammenarbeit
- Offene Punkte:
    - doxygen
    - CI
        - build test
        - test execution
        - formatting tests
        - ...