AUFGABE 5.1: SOFTWARE-ENTWURF UND -TEST

In dieser Aufgabe werden Sie einen abstrakten Datentyp zur Verwaltung einer *Prioritätswarteschlange* implementieren und testen. Sie können hierbei davon ausgehen, dass Sie nicht den Einschränkungen eines eingebetteten Systems unterliegen – d. h. die Verwendung von `malloc` und `free` ist zulässig und Sie können die Details Ihrer Datentypen vollständig vor dem Anwender verbergen. Anschließend wird Ihre Warteschlangenimplementierung versendet und von anderen Gruppen getestet. Im Gegenzug testen Sie die Warteschlangenimplementierung weiterer Gruppen. Hierbei soll ein Informationsaustausch zu Ihrer Implementierung entstehen.

Hinweis: Beachten Sie, dass sich dieses Aufgabenblatt aus mehreren Teilblättern mit eigenen Abgabeterminen zusammensetzt.

1 Aufgabenstellung

Vermerken Sie Ihre Antworten zu den Fragen der einzelnen Aufgaben an den vorgesehenen Stellen in der vorgegebenen `answers.md`*. Bitte erstellen Sie, um die Abgabe durch Mergerequests zu vereinfachen,* ***pro Aufgabe einen eigenen Branch****. Um einen konsistenten Zustand zu gewährleisten, benutzen Sie dazu bitte folgenden Befehl:*

```
git fetch git@gitos.rrze.fau.de:i4/teaching/vezs/vezs24-vorgabe.git
aufgabe5 && git checkout -b aufgabe5 FETCH_HEAD
```

Aufgabe 1 *Kommunikation mit anderen Gruppen*

Für einige Aufgaben dieses Aufgabenblattes werden Sie mit anderen Teilnehmern zusammenarbeiten. Hierfür sollten Sie eine Mail erhalten haben, die Sie einem Projekt für die Entwicklung zuordnet sowie Ihnen mitteilt, wer für die Qualitätssicherung Ihres Projektes verantwortlich ist. Melden Sie sich, falls Sie **keine** Email erhalten haben sollten! Genauere Hinweise zur Verwendung der Kommunikationsinfrastruktur finden Sie in der besagten Email.

Implementierung

Aufgabe 2 *Arbeitsumgebung*

Initialisieren Sie die Vorgabe wie in den Übungsfolien besprochen. Machen Sie sich mit der Konfiguration und den vorgegebenen Dateien vertraut.

☞ `cd build`
☞ cmake ..

Aufgabe 3 *Programmschnittstelle*

Machen Sie sich mit der Schnittstelle und dem gewünschten Verhalten der Warteschlange vertraut. Generieren Sie hierfür die `doxygen`-Dokumentation der vorgegebenen C-Schnittstelle. Anschließend finden Sie unter `doc/html/index.html` die Dokumentation der einzelnen Funktionen sowie eine Spezifikation der Warteschlange in Fließtext.

☞ `make doxy`

☞ `priority_queue.h`

Aufgabe 4 *Programmskelett*

Erstellen Sie nun Funktionsstümpfe, also eine Implementierung in korrektem C ohne irgendeine Funktionalität, für die Funktionen der Schnittstelle, so dass das Projekt ohne Fehler übersetzt werden kann. Sichern Sie nun den aktuellen Stand Ihres Projekts, in dem Sie einen git-Commit erstellen.

Aufgabe 5 *Aufteilung*

Setzen Sie nun in den folgenden Teilaufgaben den Software- und Testfallentwurf um. Hierbei sollten die Testfälle für eine Funktion nicht von der gleichen Person geschrieben werden, die diese Funktion implementiert. Teilen Sie die Implementierungsarbeit innerhalb Ihrer Gruppe in möglichst unabhängige Arbeitspakete auf. Die Verwendung verschiedener Git-Branches kann diese Aufteilung erleichtern. *Wieso ist diese Herangehensweise sinnvoll? Dokumentieren Sie die Arbeitsaufteilung!*

Antwort:

Aufgabe 6 *Warteschlange*

Implementieren Sie die nötige Funktionalität für die vorgegebene Schnittstelle der Prioritätswarteschlange.
Hinweis: Es bietet sich an, die Warteschlange als einfach verkettete Liste zu implementieren.

Packen Sie Ihre fertige Implementierung in ein `tar`-Archiv und senden Sie dieses zum Testen an Ihr zugeteiltes Testteam (`qa`, siehe Aufgabe 1).
Bitte bauen Sie das tar-Archiv **unbedingt** mittels des **vorgegebenen Make-Targets**

☞ `make pack-review`

auf einem **Rechner des CIP-Pools** der Informatik, um Kompatiblität der ausgetauschten Artefakte zu gewährleisten.
Bitte markieren Sie unbedingt auch jeden versendeten Versionsstand in Ihrer Versionsverwaltung, bspw. als eigenen Commit!

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 13.12.2024
- Fragen bitte an `i4ezs@lists.cs.fau.de`