AUFGABE 5.2: SOFTWARE-ENTWURF UND -TEST – TEIL 2

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.

Testfallentwurf

In diesem Aufgabenteil widmen wir uns nun dem Entwurf von Testfällen für unsere Warteschlange.

Aufgabe 7 *Fehlerraumanalyse*

Skizzieren Sie die Fehlerhypothese für die Nutzung der Schnittstelle. *Welche Randfälle müssen Sie berücksichtigen und warum?*

Antwort:

Aufgabe 8 *Testfallentwurf*

Entwerfen Sie nun auf Basis Ihrer Analyse und der identifizierten Randbereiche der Spezifikation Testfälle, die geeignet sind, zu zeigen, dass die Implementierung Ihres Softwareentwurfs den Anforderungen entspricht und auch sonst kein nicht definiertes Verhalten besitzt. *Welche Kategorien von Testfällen gibt es?*

Antwort:

Aufgabe 9 *Testfälle*

Für die Implementierung der Testfälle sollen Sie die Testumgebung von `cmake` verwenden. Legen Sie hierbei pro Testfall eine eigene `.c`-Datei mit `main()`-Funktion im Unterverzeichnis `tests` an. Tragen Sie Ihre Tests in der Konfigurationsdatei `tests/CMakeLists.txt` in die passende der drei Testfalllisten ein. Die angedachten Nutzungweisen der einzelnen Listen (`EZS_PQ_GENERAL_TESTS`, `EZS_PQ_MALLOC_TESTS`, `EZS_PQ_OWN_ONLY_TESTS`) sind in der Konfigurationsdatei dokumentiert.

☞ make
☞ make test

Hinweis: Das von `cmake` erzeugte `test`-Target führt nur die Tests aus, ohne Ihr zu testendes Programm neu zu kompilieren. Es bietet sich daher an, Tests mittels `make && make test` aufzurufen.

Welche Anforderungen sind an gute Testfälle zu stellen? Wieso sind diese Eigenschaften jeweils sinnvoll?

Antwort:

Fremdtests

Aufgabe 10 *Testumgebung*

Um gleichzeitig die Fremdbibliothek testen und die weiteren Aufgaben bearbeiten zu können, bietet es sich an, ein separates Build-Verzeichnis zu erstellen. Die Option `BUILD_ALIEN_TEST` muss gesetzt sein, um unsere Testinfrastruktur nutzen zu können. Diese trifft unter anderem gewisse Sicherheitsvorkehrungen zum Ausführen fremden Codes, ist also dringend notwendig.

Setzen Sie eine neue Buildumgebung mittels folgender Befehle auf:

```
mkdir build-other && cd build-other
cmake .. -DBUILD_ALIEN_TEST=ON -DBUILD_WITH_COVERAGE=OFF
```

Innerhalb dieses Verzeichnisses können Sie die Fremdimplementierung nun wie gewohnt mittels `make test` testen.

Hinweis: Bitte achten Sie unbedingt nach dem Austauschen des statischen Archivs darauf, dass auch tatsächlich die neue Version der Bibliothek in den Tests verwendet wird (beispielsweise durch `make clean` oder Neuanlegen des Buildverzeichnisses).

Aufgabe 11 *Testen*

Sie haben die Implementierung der Warteschlange von bis zu zwei zufällig zugeteilten Gruppen erhalten. Speichern Sie diese in Ihrem Quellcode-Verzeichnis unter `review/lib_priority_queue_alien.a`. Testen Sie diese Implementierung mit Hilfe Ihrer Testfälle und teilen Sie der anderen Gruppe über die bekannte Mailinfrastruktur die Ergebnisse mit. Antworten Sie dazu einfach auf die ursprüngliche Nachricht.

☞ `make -B`
☞ `make test`

Sollten Sie von der Gruppe eine überarbeitete Implementierung erhalten, wiederholen Sie bitte das Vorgehen, bis alle Tests erfolgreich verlaufen.

Aufgabe 12 *Anpassungen und Fehlerbehebung*

Sie haben von den Ihnen zugeteilten Testgruppen Rückmeldung über die Testergebnisse Ihrer Warteschlage erhalten. Falls nötig, beheben Sie diese Fehler und senden Sie Ihre Implementierung erneut an die Testgruppen.

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 20.12.2024
- Fragen bitte an `i4ezs@lists.cs.fau.de`