Algorithmen und Datenstrukturen I

Blatt 4

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 21.10.2019 16:34

Aufgabe 1 — Linked List

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/RuZ8R8q_.

Im Repository finden Sie ein CMake-Projekt mit einer Listen-Library und Googletest. Implementieren Sie eine doppelt verkettete Liste in der Datei `list/list.cpp`, die zum vorhandenen Headerfile passt. Die bereits implementierten Tests sollen erfolgreich durchlaufen werden. Sie können diese natürlich noch zusätzlich erweitern (das ist immer eine gute Idee :-)).

Eine Liste besteht aus Elementen, die eine ganze Zahl als Wert enthalten. Die beiden Pointer `prev` und `next` zeigen auf den Vorgänger bzw. Nachfolger in der Liste. Das erste Element der Liste hat als Vorgänger den Null-Pointer `nullptr`. Das letzte Element der Liste hat als Nachfolger den `nullptr`.

Um überhaupt auf die verketteten Listenelemente zugreifen zu können, gibt es einen Zeiger auf das erste Element (`head`) und einen Zeiger auf das letzte Element (`last`).

Die Listenklasse enthält folgende Methoden:

- `search` sucht nach einem Wert `value` und gibt als Wahrheitswert zurück ob er gefunden wurde.
- `insert` fügt einen Wert `value` an der Position `position` ein. Dabei wird das Element das an der Position steht, um eine Position nach hinten verschoben. Es muss nur für positive Positionen von `1` bis `N+1` korrekt funktionieren, wobei `N` die Länge der Liste ist und `N+1` bedeutet: “hänge hinten an die Liste an”. Was

Sie mit zu kleinen oder zu großen Positionen machen, entscheiden Sie selbst und dokumentieren es entsprechend.

- `remove` entfernt den ersten gefundenen Wert `value`.
- `toVector` erzeugt einen Vektor mit Werten die den Elementen in der Reihenfolge in der Liste entsprechen.

Die Ausgabe einer Liste soll in eckigen Klammern mit Kommas getrennt erfolgen, also z.B.

```
[]
[1, 2, 3]
```

Die Ausgabe wird nicht von den Unittests überprüft.

Verwenden Sie normale Pointer (*raw pointer*) und keine Smart Pointer. Achten Sie darauf, dass der Destruktor den Speicher richtig frei gibt.