UNIVERSITÄT BONN

Algorithmen und Programmierung
Übungsblatt 9
WS 2022/23

Dr. Felix Jonathan Boes
Benedikt Bastin, Ellen Bundschuh, Anna Höpfner, Gina Muuss, Adrian Oeyen, Felix Roth, Thore Wolf

Ausgabe: 05.12.2022
Abgabe: keine Abgabe; Präsentation in der Übung

Aufgabe 1 (Doppelt verkettete Liste)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `Zettel_09` angelegt. Implementieren Sie die dort, unter Verwendung der Projektstruktur, die Klasse `DoublyLinkedList` und demonstrieren Sie Ihre Implementierung.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include -I./external -fsanitize=address \
        IHRE_QUELLDATEIEN \
        examples/aufg1.cpp -o aufg1
```

Aufgabe 2 (Binärbäume durchlaufen)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `Zettel_09` angelegt. Erweitern Sie die dort, unter Verwendung der Projektstruktur, Ihre Implementierung der Klasse `BinaryTree` wie folgt und demonstrieren Sie Ihre Implementierung.

(1) Jeder Knoten soll einen Weakpointer auf den jeweiligen Elternknoten besitzen (falls vorhanden). Implementieren Sie eine Memberfunktion (der Knoten), welche einen Sharedpointer auf den Elternknoten zurückgibt (falls vorhanden).

(2) Implementieren Sie eine Memberfunktion der Klasse `BinaryTree`, welche die Knoten von Binärbäumen vermöge der Preorderreihenfolge ausdruckt.

(3) Implementieren Sie eine Memberfunktion der Klasse `BinaryTree`, welche die Knoten von Binärbäumen vermöge der Inorderreihenfolge ausdruckt.

(4) Implementieren Sie eine Memberfunktion der Klasse `BinaryTree`, welche die Knoten von Binärbäumen vermöge der Postorderreihenfolge ausdruckt.

(5) Implementieren Sie eine Memberfunktion der Klasse `BinaryTree`, welche die Knoten von Binärbäumen vermöge der Levelorderreihenfolge ausdruckt.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include -I./external -fsanitize=address \
        IHRE_QUELLDATEIEN \
        examples/aufg2.cpp -o aufg2
```

Aufgabe 3 (Maxheaps am Beispiel)**.** Lösen Sie die folgenden Teilaufgaben mit Stift und Papier.

(1) Erzeugen Sie, wie in der Vorlesung demonstriert, aus dem folgenden Array einen linksvollständigen Binärbaum. `[68, 61, 30, 43, 20, 19, 23, 5, 21, 19, 13]`

(2) Zeigen Sie, dass dieser linksvollständige Binärbaum ein MaxHeap ist.

(3) Fügen Sie in diesen MaxHeap zuerst das Element `20` und anschließend das Element `40` ein. Demonstrieren Sie beim Einfügen alle, in der Vorlesung diskutierten, Teilschritte.

(4) Entfernen Sie im Anschluss das Element mit der größten Priorität. Demonstrieren Sie beim Entfernen alle, in der Vorlesung diskutierten, Teilschritte.

Aufgabe 4 (Maxheaps implementieren (Bonusaufgabe))**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `Zettel_09` angelegt. Implementieren Sie die dort, unter Verwendung der Projektstruktur, die Klasse `MaxHeap` mithilfe eines Arrays und demonstrieren Sie Ihre Implementierung. Schreiben und verwenden Sie dabei ein Hilfsfunktionen zur arithmetischen Bestimmung des Index des Elternknoten, des linken Kinds (falls existent) und des rechten Kinds (falls existent). Überlegen Sie sich dazu vorher, welchen Index der Elternknoten eines beliebigen Knoten mit Index k hat und welche Indizes die beiden Kindknoten eines beliebigen Knoten mit Index k haben müssen.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include -I./external -fsanitize=address \
        IHRE_QUELLDATEIEN \
        examples/aufg3.cpp -o aufg3
```