**Protokoll Aufgabe 1**

**Funktionen:**

**hashFunc():**

Man multipliziere die Summe mit dem Quotienten aus der Größe der Hashtabelle durch den maximalen ASCII Wert für ein Kürzel. Dies muss noch modulo der Hashtabellengröße gerechnet werden, um nicht den Speicherbereich zu überschreiten.

**Kollisionserkennung:**

Mit zwei quadratischen Sondierungsfunktionen quadraticProbingForName() und quadraticProbingForKuerzel() wird die Kollisionserkennung vorgenommen. Man könnte zwar diese überladen, jedoch haben wir uns in der Implementierung für zwei Implementierungen entschieden, da der übergebene struct\* jeweils ein anderer ist.

**Verwaltung der Kursdaten:**

Es können insgesamt 2003 Aktien gespeichert werden, welche drei Strings enthalten: Name, WKN und das Kürzel. Die Kursdaten selbst sind in einem weiteren struct-Array mit der Länge 30 gespeichert. Der jeweilige Zugriff über den Namen erfolgt mit dem struct mapNameToKuerzel.

**Löschalgorithmus:**

Die clearHeap()-Funktion löscht linear Elemente, wenn die Werte ungleich NULL sind.

**Aufwandsabschätzung:**

Der Zugriff auf einzelne Aktien ist sowohl im avarage Case als auch im best Case konstant O(1). Im worst Case sind es O(n). Der Füllgrad erfüllt die Mindestanforderungen, um bei maximal 1000 Aktien einen Füllgrad von 50% nicht zu überschreiten. Bei dieser Größe benötigt man bei unserer Datenstruktur mindestens eine Operation und maximal 1000.

Im Vergleich dazu sind die Zugriffe, das Auslesen und die Löschung einer Aktie in einem Array deutlich performanter mit jeweils O(1). Eine Verkettete Liste weist hingegen nur beim auslesen und löschen einen O(1) Aufwand auf, wenn der Optimalfall eintritt. Ansonsten ist das Einfügen grundsätzlich linear, genauso wie die durchschnittlichen und die am wenigsten performanten Fälle des Einlesens und Löschens.

**Aufwandsabschätzung Übersicht:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Einfügen** | **Hashmap** | **Array** | **Verkettete Liste** |
| Best Case | **O(1)** | **O(1)** | **O(n)** |
| Avg Case | **O(1)** | **O(1)** | **O(n)** |
| Worst Case | **O(n)** | **O(1)** | **O(n)** |
| **Auslesen/Löschen** |  |  |  |
| Best Case | **O(1)** | **O(1)** | **O(1)** |
| Avg Case | **O(1)** | **O(1)** | **O(n)** |
| Worst Case | **O(n)** | **O(1)** | **O(n)** |