Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Andreas Hildebrandt

8. Übung zur Vorlesung Programmierung II, SS 2003

Aufgabe 1: Laufzeitanalyse (10 Punkte)
InsertionSort kann für einen `vector` $v = [v_0, \ldots, v_{n-1}]$ folgendermassen rekursiv formuliert werden: v wird sortiert, indem zuerst $[v_1, \ldots, v_{n-1}]$ sortiert wird und dann $v[n]$ an die richtige Stelle eingefügt wird.

Stellen Sie eine Rekurrenzgleichung für diese Version von InsertionSort auf und lösen Sie diese.

Aufgabe 2: Hashtabellen (10 Punkte)
Gegeben Sei die Menge $I := \{4711, 2003, 1218, 7181, 6162, 4242, 1221\}$ und eine Hashfunktion $h(k) = k \mod 10$. Wie sieht eine zu Anfang leere Hashtabelle aus, wenn sie die Elemente der Menge I in der oben gebenen Reihenfolge mittels

- Chaining
- Linear probing
- Quadratic probing mit $c_1 = 1, c_2 = 2$
- Double hashing mit einer zweiten Hashfunktion $h_2(x) = 7 - (x \mod 7)$

einfügen?

Aufgabe 3: Divisionsmethode (10 Punkte)
Für eine Hashtabelle der Größe m, m prim, zerlegen wir die Schlüssel k in $r + 1$ *bytes* (also in Zeichen oder Teilstrings gleicher Länge), so dass $k =< k_0, \ldots, k_r >$, wobei alle k_i kleiner als m sein sollen. Es sei $a :=< a_0, \ldots, a_r >$ eine Sequenz von $r + 1$ zufällig aus der Menge $\{0, 1, \ldots, m - 1\}$ gewählten Elementen. Damit definieren wir für jedes a eine Hashfunktion $h_a(k)$ durch:

$$h_a(k) = \left(\sum_{i=0}^{r} a_i k_i \right) \mod m$$

Zeigen Sie:

- die Menge $\mathcal{H} := \bigcup_a h_a$ ist eine universelle Klasse von Hashfunktionen.

- Wenn man in der Definition von a fordert, dass alle a_i von Null verschieden sind, so ist $\mathcal{H}$ *nicht* universell.

Aufgabe 4: Hashtabellen, Implementierung (10 Punkte)
Implementieren Sie eine Template-Klasse `HashTabelleChaining<Typ, Hashfunktion>`. Dabei steht `Typ` für den Typ der in der Tabelle zu speichernden Daten. `Hashfunktion` kann entweder mit einem Funktionspointer oder einem Funktionsobjekt instanziiert werden, dessen `()` - Operator eine Variable vom Typ `Typ` auf `int` abbildet.
Die Größe m Ihrer Hashtabelle soll im Konstruktor mit angegeben werden. Implementieren Sie die Funktionen `einfuegen(Typ t)`, `entfernen(Typ t)` und `suche(Typ t)`, wobei mögliche Kollisionen durch chaining abgefangen werden. Sie können in Ihrer Implementierung die einfach verkettete Liste `list<Typ>` aus der *STL* verwenden.
Implementieren Sie eine Hashfunktion als Funktionsobjekt und testen Sie mit dieser Ihre Hashtabellenimplementierung.

Zusatzaufgabe 5: Hashtabellen, Implementierung II (10 Bonuspunkte)
Implementieren Sie die Klasse `HashTabelleOA<Typ, Hashfunktion, Addressfunktion>`, welche die gleiche Funktionalität zur Verfügung stellt wie `HashTabelleChaining`, allerdings Kollisionen durch Open Adressing auflöst. Die Art des Open Adressing kann dabei durch den dritten Templateparameter `Addressfunktion` gewählt werden, welcher durch ein Funktionsobjekt oder einen Funktionspointer realisiert werden kann.

Implementieren Sie je ein Funktionsobjekt zur Realisierung von linear probing, quadratic probing und double hashing und testen Sie Ihre Implementierung.

Abgabe: 20.06.2003