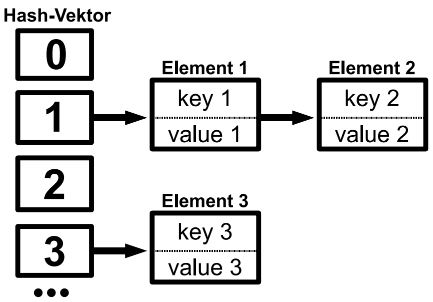
Prof. Dr. Franz Schweiggert   Institut für Angewandte Informationsverarbeitung   5. Dezember 2008
Matthias Röhm   Blatt 8

Allgemeine Informatik III / Systemnahe Software I (WS 2008)

Abgabetermin: 09.12.2008

1 Hashes

In dieser Aufgabe soll von Ihnen ein Hash implementiert werden. Ein Hash ist eine Datenstruktur die es erlaubt, über einen Schlüssel, direkt auf die diesem Schlüssel zugeordneten Werte zuzugreifen. Ein Hash-Element besteht somit aus einem Schlüssel/Wert Paar. Jeder Schlüssel darf nur **ein** Mal im Hash vorkommen. Der Zugriff auf einen Wert erfolgt somit **immer** über dessen eindeutigen Schlüssel. Intern werden die Daten eines Hashes in einem Vektor gespeichert.



Interne Repräsentation eines Hash.

Eine Hashfunktion *hash* bildet jeden möglichen Schlüssel auf einen Index im Vektor ab. Es kann vorkommen, dass die Hashfunktion *hash* für zwei **verschiedene** Schlüssel den **gleichen** Index ermittelt. Erzeugt die Hashfunktion für zwei Schlüssel $k1$, $k2$ *mit* $k1 \neq k2$ den gleichen Index $hash(k1) = hash(k2) = i$, so wird an der Position i des Vektors ein Zeiger auf eine Liste mit den Elementen $e1$ und $e2$ gespeichert. Diese Liste nimmt nun auch alle weiteren Elemente des Hashes auf, für die die Hashfunktion den Index i erzeugt.

2 Aufgabe (10 Punkte)

Schreiben Sie ein Hash-Modul „hash.c“, auf Basis der folgenden Header-Datei „hash.h“, in dem Strings als Schlüssel und Integer als Werte gespeichert werden sollen. Ein Element besteht also aus einem String, einem Integer und einem Zeiger auf das nächste Element der Liste. Ihr Modul soll die folgenden Funktionen implementieren, wobei die Hashfunktion *hash* gegeben ist:

```
#ifndef HASH_H
#define HASH_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

 typedef struct _hash *Hash;
 typedef struct _element *Element;

 struct _element {
        char *key;
        int value;
        Element next;
 };

 struct _hash {
        unsigned int size; //Size of vector
        Element *entries; //Pointer to vector
 };

 extern unsigned int hash(char *key, unsigned int size);
 extern void create_hash(Hash *h, unsigned int size );
 extern int put( Hash h, char *key, int value );
 extern int get( Hash h, char *key, int *value );
 extern int change_value( Hash h, char *key, int value );
 extern void print( Hash h );
#endif
```

- *unsigned int hash*(*char* $*$ *key*, *unsigned int size*);
  gibt einen Index für den Schlüssel *key* zwischen 0 und $size - 1$ zurück. Diese Funktion ist im Anhang beschrieben!
- *void create*(*Hash* $*h$, *unsigned int size*);
  erzeugt einen Hash mit einem Vektor der Große *size*. *size* sollte eine **Primzahl** sein.
- *int put*(*Hash h*, *char* $*$ *key*, *int value*);
  ein Schlüssel/Wert Paar wird in den Hash eingefügt. Falls der Schlüssel schon vorhanden ist, gibt die Funktion 0 zurück. Ansonsten 1.

- $int\ get(Hash\ h,\ char\ *key,\ int\ *value);$
  der zu dem Schlüssel *key* gehörende Wert wird aus dem Hash gelesen und in die als Zeiger übergebene Variable $*value$ gespeichert. Falls der Schlüssel nicht exisiert gibt die Funktion 0 zurück. Ansonsten 1.

- $int\ change\_value(Hash\ h,\ char\ *key,\ int\ value);$
  verändert den Wert des Elements mit dem Schlüssels *key*. Falls der Schlüssel nicht exisiert gibt die Funktion 0 zurück. Ansonsten 1.

- $void\ print(Hash\ h);$
  alle Schlüssel/Wert Paare des Hashes werden an die Standardausgabe ausgegeben.

Die Header-Datei „hash.h“ und die im Anhang aufgeführte Hash-Funktion können von der Vorlesungsseite heruntergeladen werden. Zum Testen Ihres Moduls können Sie von der Vorlesungsseite einen Quellcode zur Bestimmung der Worthäufigkeit eines Textes herunterladen. Das Programm benötigt natürlich ein korrekt funktionierendes Hash-Modul „hash.c“.

3 Anhang

```
//Die Hash-Funktion zur Berechnung des Hashwerts eines Strings
unsigned int hash(char *key, unsigned int size) {
        unsigned int h = 0;
        while(*key != '\0') {
                h = 31 * h + *key;
                key++;
        }
        return (h % size);
}
```

Viel Erfolg!