Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2015/16
Feldmann / Semmler / Lichtblau / Streibelt / Pujol / Rost

Aufgabenblatt 10

letzte Aktualisierung: 21. Januar, 01:01 Uhr

(d9a305622f2e47b6aa7c7d556e8212646e27bdeb)

Ausgabe: Donnerstag, 21.01.2016
Abgabe: spätestens Montag, 01.02.2016, 20:00

Thema: *Quicksort auf einfach verketteten Listen*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des tubIT/IRB mittels `gcc -std=c99 -Wall` kompilieren.
- Abgaben erfolgen prinzipiell immer in Gruppen à 2 Personen, welche in den Tutorien festgelegt wurden. Einzelabgaben sind explizit als solche gekennzeichnet.
- Die Abgabe erfolgt ausschließlich über SVN. Die finale Abgabe
  - für Gruppenabgaben erfolgt im Unterordner `Tutorien/t<xx>/Gruppen/g<xx>/Abgaben/Blatt<xx>/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Abgaben/Blatt<xx>/`

  wobei die Ordner von uns erstellt werden.
- Benutze für alle Abgaben von Programmcode das folgende Namensschema: `introprog_blatt0X_aufgabe0Y_Z.c`, wobei X durch die Blattnummer, Y durch die Aufgabe und Z durch die Unteraufgabe zu ersetzen ist.
  Beispiel: Aufgabe 1.2 wird zu: `introprog_blatt01_aufgabe01_2.c`
  Gib für jede Unteraufgabe maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Benenne alle anderen Abgaben (Pseudocode, Textaufgaben) wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Verwende auch hier eine Datei pro Aufgabe, nicht jedoch pro Unteraufgabe.

1. Aufgabe: Implementierung von Quicksort auf verketteten Listen (3 Punkte)

Gib den Quelltext in einer Datei mit folgendem Namen ab:
`introprog_blatt10_aufgabe01.c`

Implementiere den in der Vorlesung vorgestellten Sortieralgorithmus `quick_sort` in C. Deine Funktion soll als Eingabe eine verkettete Liste bekommen und die Elemente sortiert zurückgeben.

Dazu findest du hier den Pseudocode für Quicksort, der seine generelle Funktionalität darstellt. Erarbeite dir die Funktionalität der Funktion `partition()` aus den Vorlesungsfolien sowie dem zusätzlichen Foliensatz zu Quicksort auf ISIS.

Listing 1: Pseudocode Quicksort Algorithmus (rekursiv)

```
1  QuickSort (list tosort)
2
3          if (tosort.first != tosort.last)
4
5                  list left, right
6
7                  pivot ← Partition (tosort, left, right)
8                  QuickSort(left)
9                  QuickSort(right)
10
11                 if (left.first == NULL)
12                         left.first ← pivot
13                         left.last ← pivot
14                 else
15                         left.last.next ← pivot
16                 tosort.first ← left.first
17                 pivot.next ← right.first
18
19                 if (right.first == NULL)
20                         tosort.last ← pivot
21                 else
22                         tosort.last ← right.last
```

Hinweis: Beachte, dass beim Einfügen in die rechte bzw. linke Teiliste nicht neuer Speicher alloziert werden muss, sondern die jeweiligen `next`-Pointer entsprechend gesetzt werden.

Hinweis: Beachte für die Implementierung der Funktion `partition()`, dass sobald in der Liste weitere Elemente vorkommen, die denselben Wert wie das Pivotelement haben, diese in die rechte Teilliste eingefügt werden. Das Pivotelement selbst ist immer Rückgabewert der Funktion `partition()`. Gibt es also nur ein Element mit dem Wert des Pivotelements (also das Pivotelement selbst), ist dieses nicht Teil der rechten bzw. linken Teilliste (s. Zusatzmaterial: Quicksort), sondern tritt nur als Rückgabewert der Funktion `partition` auf.

Die Eingabedatei zu dieser Aufgabe enthält eine Liste der 100 beliebtesten Passwörter und deren Häufigkeit im Format:

```
Passwort Häufigkeit
```

Lies die Eingabedatei ein und speichere die Passwörter und ihre Häufigkeiten gemeinsam als Element einer einfach verketteten Liste ab. Dein Programm bekommt den Pfad zur Eingabedatei als erstes Argument.

Deine `quick_sort` Implementierung nimmt diese Liste als Eingabe und gibt am Ende die Passwort-Häufigkeits-Paare nach aufsteigender Häufigkeit sortiert aus. Das Ausgabeformat soll dabei dem Format der Eingabedatei entsprechen.

Du darfst dabei annehmen, dass die Eingabe diesem Format entspricht, sowie dass die Datei Leseberechtigung hat.

Listing 2: Vorgabe `main_blatt10.c`

```c
#include "blatt10.h"

int main(int argc, char** args)
{
   if (argc != 2)
   {
      printf("Nutzung: %s <Dateiname>\n",args[0]);
      return 1;
   }
   list mylist;
   init_list(&mylist);
   read_data(args[1],&mylist);
   qsort_list(&mylist);
   printf("Sortierte Liste:\n");
   print_list(&mylist);
   free_list(&mylist);
   return 0;
}
```

Codevorgabe:

Listing 3: Vorgabe `introprog_blatt10_aufgabe01_vorgabe.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "blatt10.h"

/*
Definiere hier die notwendigen structs, um eine verkettete Liste mit
Passwörtern (mit Passwort und Häufigkeit) zu erhalten.
 HIER typedef struct für list_element impplementieren
 HIER typedef struct für list impplementieren
*/

void init_list(list* mylist)
{
// HIER Liste initialisieren
}


// Diese Funktion fügt Listenelemente am Anfang der Liste an
void insert_front(list_element* le, list* mylist)
{
    // HIER Code einfügen
}

// Speicher für Listenelemente wieder freigeben
void free_list(list* mylist)
{
    // HIER Code einfügen
}


```



```c
// Namen, Zahlen Paare in Liste einlesen
void read_data(char* filename, list* mylist)
{
      // HIER Code einfügen:
        // * Speicher allozieren
        // * Daten in list_element einlesen
        // * insert_front benutzen um list_element in Liste einzufügen
}

list_element* partition( list* input, list* left, list* right )
{
    // HIER Code einfügen:
    // parition() Funktion implementieren
}

// Hauptfunktion des quicksort Algorithmus
void qsort_list(list* mylist)
{
    // HIER Code einfügen
}

// Liste ausgeben
void print_list(list* mylist)
{
    // HIER Code einfügen:
    // * Laufe über die list_element in mylist und gebe sie aus.
}
```