Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2014/15
Feldmann / Rost / Streibelt / Lichtblau / Deimel / Lucas

Aufgabenblatt 5

letzte Aktualisierung: 03. Dezember, 14:46 Uhr

(8458ba2ef81003cfd53411e392256d8875d0c86d)

Ausgabe: Mittwoch, 03.12.2014
Abgabe: spätestens Freitag, 12.12.2014, 18:00

Thema: *Quicksort auf 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>/Blatt<xx>/submission/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Blatt<xx>/submission/`
- Benutzen Sie 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 ersetzen ist.
  Beispiel: `introprog_blatt01_aufgabe01_1.2.c`
  Für jede Unteraufgabe geben Sie maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Alle anderen Abgaben (Pseudocode, Textaufgaben) benennen Sie wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Auch hier verwenden Sie eine Datei pro Aufgabe.

1. Aufgabe: Implementierung von Quicksort auf verketteten Listen (40 Punkte)

Geben Sie den Quelltext in einer Datei mit folgendem Namen ab:
`introprog_blatt05_aufgabe01.c`

Implementieren Sie den in der Vorlesung vorgestellten Sortieralgorithmus `quick_sort` in C. Ihre Funktion soll als Eingabe eine einfach verkettete Liste bekommen und die Elemente sortiert zurückgeben.

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

```
Passwort Häufigkeit
```

Definieren Sie sich einen passenden Datentyp, um Passwort und Häufigkeit gemeinsam abzulegen. Lesen Sie die Eingabedatei ein und speichern Sie Passwörter und Häufigkeiten gemeinsam als Element einer einfach verketteten Liste ab.

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

Hinweis: Beachten Sie, dass wir bei dieser Aufgabe weniger Vorgaben als bisher machen. Die Codevorgabe ist daher nur als grobe Struktur für die von Ihnen selbstständig zu enwickelnde Implementierung gedacht.

Halten Sie sich an folgende Codevorgabe:

Listing 1: Vorgabe `introprog_blatt05_aufgabe01_vorgabe.c`

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  /*
6  Definieren Sie hier die notwendigen structs, um eine verkettete Liste mit
7  Passwörtern (mit Passwort und Häufigkeit) zu erhalten.
8   HIER typedef struct für list_element impplementieren
9   HIER typedef struct für list impplementieren
10 */
11
12 void init_list(list* mylist)
13 {
14     mylist->first=NULL;
15     mylist->last=NULL;
16 }
17
18
19 // Diese Funktion fügt Listenelemente am Anfang der Liste an
20 void insert_front(list_element* le, list* mylist)
21 {
22     // HIER Code einfügen
23 }
24
25 // Speicher für Listenelemente wieder freigeben
26 void free_list(list* mylist)
27 {
28     // HIER Code einfügen
29 }
30
31
```

```
32 // Namen, Zahlen Paare in Liste einlesen
33 void read_data(char* filename, list* mylist)
34 {
35     assert(mylist != NULL);
36     FILE* f=fopen(filename,"rb");
37     assert(f != NULL);
38     while (1)
39     {
40         // HIER Code einfügen:
41         // * Speicher allozieren
42         // * Daten in list_element einlesen
43         // * insert_front benutzen um list_element in Liste einzufügen
44     }
45     fclose(f);
46 }
47
48 // Pivot finden, das die Liste aufteilt
49 list_element* partition( list* input, list* left, list* right )
50 {
51     list_element* pivot= // HIER pivot Element setzen
52     // HIER Code einfügen
53     return pivot;
54 }
55
56 // Hauptfunktion des quicksort Algorithmus
57 void qsort_list(list* mylist)
58 {
59     // HIER Code einfügen
60 }
61
62 // Liste ausgeben
63 void print_list(list* mylist)
64 {
65     // HIER Code einfügen:
66     // * Laufe über die list_element in mylist und gebe sie aus.
67 }
68
69 // Argumente einlesen, Liste kreieren, verarbeiten und ausgeben
70 int main(int argc, char** args)
71 {
72     if (argc != 2)
73     {
74         printf("Nutzung:␣%s␣<Dateiname>\n",args[0]);
75    return 1;
76     }
77     list mylist;
78     init_list(&mylist);
79     read_data(args[1],&mylist);
80     qsort_list(&mylist);
81     printf("Sortierte␣Liste:\n");
82     print_list(&mylist);
83     free_list(&mylist);
84     return 0;
85 }
```