Praktikum – Einführung in die Programmiersprache C
Sommersemester 2018

Aufgabenblatt 2

Aufgabe 2.1

Schreiben Sie ein Programm, das als Eingabe den Namen einer C-Quelldatei von der Konsole einliest, diese dann öffnet und syntaktisch analysiert.

Die Analyse soll folgende Überprüfungen enthalten:

- Korrekte Verwendung „runder Klammern“,
- Korrekte Verwendung „geschweifter Klammern“.

Im Falle eines Fehlers soll

- die Zeilennummer, in welcher der Fehler aufgetreten ist, und
- die Art der Klammer, bei der ein Fehler aufgetreten ist,

auf der Konsole ausgegeben werden.

Am Ende der erfolgreichen Überprüfung soll

- das Ergebnis der Analyse,
- die Anzahl der Zeilen im Quelltext und
- die jeweils tiefste Schachtelungsebene für beide Klammerarten

auf der Konsole ausgegeben werden.

Fehler sind:

- unbalancierte Klammerungen oder
- offene Klammerebenen am Dateiende.

Zum Testen sollten Sie selbst kleine Testdateien erzeugen, die garantiert korrekt sind oder alle denkbaren Fehlertypen möglichst vollständig abdecken.

Hinweise
Beachten Sie die Hinweise zu Funktionen von Aufgabe 1.4. Folgendes kleine Programm liest eine Datei zeichenweise ein und gibt den Inhalt auf `stdout` wieder aus.

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main(int argc, char **argv) {
	char c;
	FILE *file = fopen("example.c", "r");
	if (file == NULL) {
		perror(NULL);
		return -1;
	}
	while ((c = fgetc(file)) != EOF) {
		printf("%c", c);
	}
	fclose(file);
	return 0;
}
```

Aufgabe 2.2

Programmieren Sie ein Miniadressbuch, das bis zu 100 Adressen handhaben kann. Die Adressen werden in der Datei `adressen.dat` verwaltet. Eine Adresse wird in einer Zeile mit dem Format

`<vorname>\t<nachname>\t<telefonnummer>\n`

abgelegt. Alle Felder werden als Zeichenketten von maximal 25 Zeichen Länge betrachtet (Null-Terminierung beachten).
Innerhalb des Programms soll eine Adresse in einer Struktur verwaltet werden. Überprüfen Sie beim Einlesen die Korrektheit der Zeilen und geben Sie ggf. Warnhinweise auf der Konsole aus. Geben Sie die Anzahl der korrekt und fehlerhaft gelesenen Adresseinträge auf der Konsole aus. Nach dem Einlesen der Datei sollen folgende Eingaben von der Konsole korrekt verarbeitet werden können

- `n <nachname>`: Suche nach dem Nachnamen und Ausgabe aller zutreffenden Einträge in der Form `<vorname> <telefonnummer>\n` auf der Konsole,
- `v <vorname>`: Suche nach dem Vornamen und Ausgabe aller zutreffenden Einträge in der Form `<nachname> <telefonnummer>\n` auf der Konsole,
- `t <telefonnummer>`: Suche nach der Telefonnummer und Ausgabe aller zutreffenden Einträge in der Form `<vorname> <nachname>`.
- `q: Programmabbruch`

Bei allem anderen soll die Ausgabe eines Hinweises zur korrekten Verwendung des Programms erfolgen. Das Programm soll nur über die Eingabe von `q` verlassen werden, das heißt, es können beliebig viele Abfragen in einem Aufruf beantwortet werden.

Hinweise:
Verwenden Sie für die erforderlichen Zeichenkettenvergleiche die Funktion `strcmp`. Details zur Verwendung dieser Funktion erhalten Sie über `man strcmp`.
Die Anzahl der maximal möglichen Adressen soll als Konstante implementiert werden.
Die maximale Länge von Feldern soll über eine Konstante gesteuert werden.
Erzeugen Sie Testdateien, die bestimmte Formatierungsfehler enthalten und prüfen Sie, ob Ihr Programm korrekt funktioniert.
Sie dürfen folgende Codefragmente benutzen:

```
/* Definition des Typs Student */
typedef struct Student{
      char name[25 + 1];
      char matrikel[25 + 1];
} Student;

/* Instanz vom Typ Student */
Student s;
/* Zugriff auf die Bestandteile/Member */
s.name
s.matrikel

/* Verweis auf einen Studenten */
Student *sptr = &s;
/* Zugriff auf die Bestandteile/Member */
sptr->name
sptr->matrikel
```

Aufgabe 2.3

Sortieren Sie die Liste in Aufgabe 2.2 intern nach `<nachname>`. Verwenden Sie diese Eigenschaft beim Nachschlagen von Nachnamen. Erweitern Sie den Funktionsumfang des Programms um die Verarbeitung der Eingabe `s <dateinname>`: Es soll eine nach Nachnamen sortierte Liste der Form `<vorname>\t<nachname>\t<telefonnummer>\n` in `<dateiname>` ausgegeben werden.

Hinweise
Für die Sortierung und Suche gibt es zum Beispiel die Funktionen `qsort` und `bsearch`.