Aufgabe 4a: Modulare Programmierung

1 Lernziele

Anwenden der modularen Programmierung sowie Vertiefen der Kenntnisse über die Gültigkeit und Sichtbarkeit von Variablen. Sie können erste Erfahrungen sammeln im gemeinsamen Erstellen von Software: Man einigt sich auf eine Schnittstelle und jeder Beteiligte löst unabhängig vom anderen einen Teil der Aufgabe.

Die Aufteilung, wer aus der Zweiergruppe welches Modul implementiert, bleibt Ihnen überlassen. Am Ende des Praktikums müssen beide Module zusammen kompilierbar sein und das Programm die gewünschte Funktionalität aufweisen.

2 Aufgabe

Erstellen Sie ein Programm, das eine Textdatei einliest und die Wörter der Datei in umgekehrter Reihenfolge auf dem Bildschirm ausgibt. Verwenden Sie dazu eine Datenstruktur `stack_t`, die die folgenden Funktionen bereitstellt:

```
stack_t *createStack(void);
char isEmpty(stack_t *);
void push(stack_t *stack, char *value);
char *top(stack_t *stack);
void pop(stack_t *stack);
void destroyStack(stack_t *stack);
```

Beispiel:

- original: Dies ist nur ein kleiner, aber feiner Test!
- modifiziert: Test! feiner aber kleiner, ein nur ist Dies

Schreiben Sie ein Modul `stack.c`, das die obige Schnittstelle implementiert. Stellen Sie sicher, das auf die Komponenten der Struktur nur innerhalb des Moduls zugegriffen werden kann. Überlegen Sie sich, welche Komponenten die Struktur `stack_t` haben muss und definieren Sie mittels `typedef` einen Datentyp.

Schreiben Sie ein Modul `main.c`, das eine Textdatei einliest und die Wörter der Datei auf dem Bildschirm in umgekehrter Reihenfolge ausgibt. Der Name der Datei soll als Kommandozeilenparameter an das Programm übergeben werden.

Zu einem vollständigen Programm gehört eine Fehlerbehandlung!

3 Testat

Voraussetzung ist jeweils ein fehlerfreies, korrekt formatiertes Programm. Der korrekte Programmlauf muss anhand einer Beispieleingabe nachgewiesen werden. Sie müssen in der Lage sein, Ihr Programm zu erklären.