

Übungen zu Betriebssysteme

Ü1 – Aufgabe: filo

Wintersemester 2022/23

Henriette Hofmeier, Benedict Herzog, Timo Hönig

Bochum Operating Systems and System Software Group (BOSS)



RUHR
UNIVERSITÄT
BOCHUM

RUB

Aufgabe: filo

- Aufgabe: Einfach verkettete **LIFO-Liste**, die Ganzzahlen (`int`) verwaltet
- Dynamische wachsende Datenstrukturen (bspw. verkettete Listen) notwendig, wenn die Anzahl der Einträge a-priori unbekannt ist

Aufgabe: filo

- Aufgabe: Einfach verkettete LIFO-Liste, die Ganzzahlen (`int`) verwaltet
- Dynamische wachsende Datenstrukturen (bspw. verkettete Listen) notwendig, wenn die Anzahl der Einträge a-priori unbekannt ist
- Lernziele:
 - Dynamische Speicherverwaltung verstehen & gezielt einsetzen
 - Umgang mit "einfachen" Zeigern
 - Nutzung von zusammengesetzten Datentypen (Strukturen) und ggf. Typaliasen

```
struct listelement {  
    int value;  
    struct listelement *next;  
};  
typedef struct listelement listelement; // optional
```

- filo: **Kopf** der Liste als globale Variable

- Anlegen eines Listenelementes mittels `malloc(3)`

```
struct listelement *newElement;  
newElement = malloc(sizeof(struct listelement));  
if(newElement == NULL) { /* Fehlerbehandlung */ }  
  
// [...]
```

```
free(newElement);
```

- Zurückgegebener Speicher hat undefinierten/zufälligen Wert
 - Initialisierung muss per Hand erfolgen
- **Allokierter Speicher muss wieder freigegeben werden** → `free(3)`

- filo: Kopf der Liste als globale Variable
- Anlegen eines Listenelementes mittels malloc(3)

```
struct listelement *newElement;  
newElement = malloc(sizeof(struct listelement));  
if(newElement == NULL) { /* Fehlerbehandlung */ }  
  
// [...]
```

```
free(newElement);
```

- Zurückgegebener Speicher hat undefinierten/zufälligen Wert
 - Initialisierung muss per Hand erfolgen
- **Allokierter Speicher muss wieder freigegeben werden** → free(3)



Achtung: Speicherverwaltung in C ist umständlich und fehleranfällig!
Macht euch Gedanken, bevor ihr "mal schnell" was programmiert.

- Nur folgende Funktionen zu implementieren
 - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.

- Nur folgende Funktionen zu implementieren
 - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
 - `removeElement()`: Entfernt den **zuletzt eingefügten** Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.

- Nur folgende Funktionen zu implementieren
 - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
 - `removeElement()`: Entfernt den **zuletzt eingefügten** Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.
- Keinerlei Listen-Funktionalität in der `main()`-Funktion
 - **Alle** für die Verwaltung der Liste notwendigen Operationen werden in `insertElement()` und `removeElement()` ausgeführt
 - Allerdings: Erweitern der `main()` zum Testen erlaubt und **erwünscht**

- Nur folgende Funktionen zu implementieren
 - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
 - `removeElement()`: Entfernt den **zuletzt eingefügten** Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.
- **Keinerlei Listen-Funktionalität in der `main()`-Funktion**
 - **Alle** für die Verwaltung der Liste notwendigen Operationen werden in `insertElement()` und `removeElement()` ausgeführt
 - Allerdings: Erweitern der `main()` zum Testen erlaubt und **erwünscht**
- Sollte bei der Ausführung einer verwendeten Funktion (z. B. `malloc(3)`) ein Fehler auftreten, sind keine Fehlermeldungen auszugeben. Im Fehlerfall wird -1 zurück gegeben.