Praktikum – Einführung in die Programmiersprache C
Sommersemester 2018

Aufgabenblatt 3

Aufgabe 3.1

Schreiben Sie ein Programm, das einen Keller (englisch Stack) als Array der Größe 100 implementiert. Der Stack soll sowohl `Int`- als auch `Float`-Werte in beliebiger Kombination verwalten können (union).
Folgende Operationen sollen unterstützt werden:

- `push(element)`: ein Element (`int` oder `float`) wird auf die Spitze des Stacks abgelegt.
- `pop`: das Element von der Spitze des Stacks wird zurückgegeben und die Spitze ein Feld zurückgesetzt.

Der Stack arbeitet also nach dem Last-in-first-out-Prinzip.
Schreiben Sie eine Verwaltungsfunktion, die die aktuelle Spitzenposition auf dem Stack als statische Variable verwaltet. Die Funktion sollte in drei Modi aufgerufen werden können:

- `INQUIRE`: gibt den aktuellen Wert der Spitze zurück
- `PUSH`: prüft, ob noch Platz im Array verfügbar ist, und gibt ggf. die neue Spitzenposition (alt + 1) zurück (andernfalls –1)
- `POP`: prüft, ob der Stack leer ist, und gibt ggf. die neue Spitzenposition (alt – 1) zurück (andernfalls –1).

Die Aufrufmodi der Verwaltungsfunktion und die Datentypen der Stackelemente sollten über ein `enum` unterschieden werden (siehe Codefragment unten).
Die „eigentlichen" Zugriffsfunktionen `push` und `pop` (siehe oben) sollen diese Verwaltungsfunktion verwenden, um zu prüfen, ob ihre jeweilige Aktion zulässig ist, um die Stackspitze konsistent zu halten. Bei einem Fehler soll eine entsprechende Meldung ausgegeben werden. Der Stack selber kann global definiert werden, der Zugriff auf die Spitze soll aber nur über die statische Variable der Verwaltungsfunktion erfolgen.
Schreiben Sie eine kleine Bedienschnittstelle (analog zu Aufgabe 2.2), mit deren Hilfe Elemente zum Stack hinzugefügt und entfernt werden können, sowie die aktuelle Belegung abgefragt werden kann.

Hinweise
Sie dürfen folgende Codefragmente benutzen:

```
enum ModeEnum{
      INQUIRE,
      PUSH,
      POP
};

typedef enum ModeEnum Mode;

Mode m;
m = INQUIRE;
If (m == PUSH) { … }
```

```
typedef enum DataType {
      CHAR_TYPE,
      DOUBLE_TYPE
} DataType;
```

```
typedef struct Kellerelement {
      DataType type;
      union {
            char c;
            double d;
      } u;
} Kellerelement;

Kellerelement k;
k.type = DOUBLE_TYPE;
k.u.d = 4.5;
```

Aufgabe 3.2

Implementieren Sie die Verwaltungsfunktion des Stacks aus Aufgabe 3.1 mit Hilfe von Zeigern, das heißt die Variable, welche die aktuelle Stackposition verwaltet, soll als Zeiger auf Stackelemente und nicht als Index des Stackfeldes umgesetzt werden.
Die Zugriffsmethoden `push` und `pop` sollen dahingehend verändert werden, dass sie einen Zeiger auf das neue Element erhalten (`push`) bzw. einen Zeiger auf ein Element zurückgeben (`pop`).