Prof. Dr. Ralf Gerlich
Fakultät Informatik
ralf.gerlich@hfu.eu

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

Praktikum Programmieren
Übungsblatt 9
Termin: 14. Dezember 2023

Übung 1 *„Stack“*

Stellen Sie sich vor, Sie haben ein Magazin abonniert, von dem Sie einmal in der Woche ein neues Exemplar erhalten. Aufgrund Ihrer aktuellen Arbeitslast schaffen Sie es aber meist nicht, die aktuelle Ausgabe zu lesen, und legen Sie einfach auf einen Stapel zu den anderen. Auf dem Stapel liegt also immer zuoberst die aktuellste Aufgabe.

Irgendwann wächst der Stapel so weit, dass Sie sich nicht mehr trauen, eine Ausgabe aus der Mitte herauszuziehen. Ihnen bleibt nur die Option, den Stapel Schritt für Schritt von oben her abzubauen. Sehen können Sie auch nur das Titelbild der Zeitschrift, die oben auf dem Stapel liegt. Außerdem können Sie natürlich die Anzahl der Zeitschriften auf dem Stapel bestimmen.

In der Informatik bezeichnen eine solche Struktur als „Stack“ oder „Stapelspeicher“. Implementieren Sie für einen Stack mit Elementen vom Typ `char` die folgenden Funktionen:

- `void push(char c)`: Legt den `char`-Wert oben auf dem Stapel ab.
- `char pop()`: Entfernt den obersten `char`-Wert vom Stapel und gibt diesen Wert zurück.
- `char top()`: Liefert den Wert des obersten `char`-Werts vom Stapel, ohne diesen zu entfernen.
- `unsigned int size()`: Liefert die Anzahl der Einträge auf dem Stapel.
- `bool empty()`: Liefert `true` genau dann, wenn der Stapel leer ist.
- `void clear()`: Leert den Stack vollständig bzw. initialisiert ihn.

Verwenden Sie ein Array mit `char`-Elementen, um den Inhalt des Stapels zu verwalten. Der Stapel soll mindestens 100 Elemente enthalten können.

Überlegen Sie, ob Sie zur Verwaltung des Stapels noch weitere Variablen benötigen und welche Typen für diese Variablen sinnvoll wären. Fügen Sie Ihrem Programm entsprechende Variablendeklarationen hinzu.

Diese Funktionen könnten zum Beispiel wie folgt verwendet werden:

```
1  clear();
2  push('a');
3  push('b');
4  push('c');
5  printf("Oberstes Element: '%c'\n", top()); // => 'c'
6  pop();
7  printf("Anzahl Elemente: %u\n", size()); // => 2
8  pop();
9  printf("Oberstes Element: '%c'\n", top()); // => 'a'
10 pop();
11 printf("Ist Stack leer? %s\n",
12   (empty()?"ja":"nein")); // => "ja"
```

Prof. Dr. Ralf Gerlich
Fakultät Informatik
ralf.gerlich@hfu.eu

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

Testen Sie Ihre Implementierung.

Übung 2 *„Fehlerbehandlung“*

Wenn Sie es nicht schon in Aufgabe 1 getan haben: Überlegen Sie, ob es Situationen gibt, in denen die Ausführung der Operationen zu Problemen führen könnte. Ergänzen Sie Ihre Implementierung so, dass diese Fälle erkannt und die Ausführung der Operation – bei Ausgabe einer Fehlermeldung über `printf` – verweigert wird.

Funktionen, die einen Rückgabewert erfordern (z.B. `pop` oder `top`) sollen auch im Fehlerfall einen Wert zurückgeben. Wählen Sie hierfür einen beliebigen Wert aus.

Testen Sie Ihre Implementierung, indem Sie ein paar Fehlerfälle provozieren.

Übung 3 *„Ausgeglichene Klammerung“*

Lesen Sie eine Zeichenkette ein und überprüfen Sie mit Hilfe des Stacks aus Aufgabe 1, ob die Zeichenkette eine ausgeglichene Klammerung darstellt. Die Zeichenkette darf die Zeichen verb!(!, `)`, `[`, `]` sowie `{` und `}` enthalten.

Eine ausgeglichene Klammerung zeigt sich dadurch, dass alle geöffneten Klammern auch wieder durch die zugehörigen schließenden Klammern geschlossen werden, und zwar in der richtigen Reihenfolge. Beispiele hierfür sind

- `({})[([])]`
- `{{{}}}`
- `([([{}])]){}`

Die Regel verletzt wird bei den folgenden Beispielen:

- `([)]` (äußere Klammer wird vor innerer geschlossen)
- `(}` (schließende Klammer passt nicht zur öffnenden)
- `(((())))` (vier öffnende, aber nur drei schließende Klammern)

Um zu prüfen, ob die Klammerung korrekt ist, gehen Sie die Zeichenkette von vorne nach hinten durch:

- Treffen Sie auf eine öffnende Klammer, so legen Sie sie auf den Stapel.
- Treffen Sie auf eine schließende Klammer, so prüfen Sie, ob sie zur oberen Klammer auf dem Stapel passt. Wenn ja, entfernen Sie die Klammer vom Stapel und setzen Sie die Schleife fort. Wenn nein, geben Sie eine Fehlermeldung aus und beenden Sie das Programm.

Prüfen Sie am Ende, ob alle Klammern auch wieder geschlossen wurden.

Prof. Dr. Ralf Gerlich
Fakultät Informatik
ralf.gerlich@hfu.eu

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

Übung 4 *„Modularisierung“*

Sofern noch nicht in Aufgabe 3 geschehen, modularisieren Sie Ihre Implementierung:

- Erstellen Sie eine Header-Datei `stack.h`, die die Prototypen aller zum Stack gehörenden Funktionen enthält.
- Erstellen Sie eine Code-Datei `stack.cpp`, die die Implementierung der Funktionen aus `stack.h` enthält.
- Implementieren Sie das Hauptprogramm zur Prüfung der Klammerung in der Datei `klammerung.cpp`.

Nutzen Sie die `#include`-Anweisung, um `stack.h` in die `.cpp`-Dateien einzubinden.

Übung 5 *„Modularisierungsfehler“*

Provozieren Sie nun Fehler bei der Modularisierung und beobachten Sie das Verhalten:

- „Vergessen“ Sie, eine der Funktionen in `stack.cpp` zu implementieren.
- Duplizieren Sie eine der Funktionen aus `stack.cpp` in `klammerung.cpp`.
- Entfernen Sie die Datei `klammerung.cpp` aus dem Projekt.
- Entfernen Sie die `#include`-Anweisung für `stach.h` aus `stack.cpp` oder `klammerung.cpp`.
- Geben Sie bei einem Prototypen in `stack.hpp` einen anderen Rückgabetyp, mehr oder weniger Parameter oder andere Parametertypen als in der Implementierung in `stack.cpp` an.
- Entfernen Sie den Prototypen von `pop` aus `stack.hpp`.

Machen Sie sich mit den unterschiedlichen Fehlermeldungen vertraut.

Übung 6 *„Bubble-Sort“*

Um eine Folge von Zahlen aufsteigend zu sortieren, kann der sogenannte Bubble-Sort-Algorithmus verwendet werden. Dabei wird die Folge solange wiederholt von vorne nach hinten durchlaufen. In jedem Durchlauf wird jede Zahl in der Folge mit ihrem Nachfolger verglichen. Sind die beiden Zahlen nicht in der korrekten Reihenfolge, so werden sie miteinander vertauscht. Dies wird solange wiederholt, bis bei einem Durchlauf keine der Zahlen mehr vertauscht werden mussten – dann ist die Folge in der korrekten Reihenfolge.

Implementieren Sie die Funktion `void sort(int *a, unsigned int n)`. Die Funktion erwartet im Parameter `a` den Zeiger auf das erste Element eines Arrays von `n` Ganzzahlen. In der Funktion sollen die ersten `n` Ganzzahlen im angegebenen Array mit Hilfe des Bubble-Sort-Algorithmus' aufsteigend sortiert werden.

Die Funktion kann etwa wie folgt verwendet werden:

Prof. Dr. Ralf Gerlich
Fakultät Informatik
ralf.gerlich@hfu.eu

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

```
1 int array[]={15, 7, 13, 24, 9, 25};
2 sort(&array[0], 6);
3 for (int i=0; i<6; i++) {
4   printf("array[%d] = %d\n", i, array[i]);
5 }
```

Die Ausgabe dieses Programms sollte lauten:

```
array[0] = 7
array[1] = 9
array[2] = 13
array[3] = 15
array[4] = 24
array[5] = 25
```