

Übung 7

infT

Portierung einer Java-Klasse nach C (struct und Funktionen)

Sie erhalten die Dateien *StringHistory.java* und *TestStringHistory.java*, die gemäss folgender Aufgabenstellung implementiert wurden:

Die Klasse **StringHistory** - eine Modellklasse im MVC-Paradigma - ist zu entwerfen, zu implementieren und zu testen.

Die Schnittstellenliste der Klasse **StringHistory** verfüge u.a. über folgende Konstruktoren und Methoden:

```
public StringHistory(int maxStringCount)
public void add(String str)
public void clear()
public void setStringCount(int maxStringCount)
public String toString()
```

Mit Instantiieren einer **StringHistory** wird auch gleich die maximale Anzahl **maxStringCount** der Strings mitgeteilt, die diese History zu memorieren hat. **add(String str)** fügt jeweils der History einen String **str** hinzu. Ist die History einmal gefüllt, arbeitet sie nach dem Prinzip First-in, First-out (Fifo-Speicher). **clear()** leert die History. Mit **setStringCount(int maxStringCount)** kann die maximale Anzahl gespeicherter/zu speichernder Strings zu jedem Zeitpunkt verändert werden.

toString() liefert einen String wie folgt zurück:

```
1: <ältester Stringeintrag>
2: <zweitältester Stringeintrag>
...
n: <jüngster Stringeintrag>           mit n <= maxStringCount
```

Portieren Sie nun die Klasse **StringHistory** nach C, indem Sie eine geeignete Struktur

```
typedef struct _StringHistory {
    ...
} StringHistory;
```

implementieren und die Funktionen

| | |
|----------------------------|--|
| newStringHistory | erstellt eine neue StringHistory auf dem Heap |
| deleteStringHistory | entfernt die StringHistory vom Heap |
| add | fügt einen String in die History ein |
| setStringCount | setzt max. Anzahl speicherbarer Strings neu |
| clear | entfernt alle Strings aus der History |

realisieren.

Nehmen Sie eine Aufteilung in die Dateien *stringHistory.h*, *stringHistory.c* und *stringHistoryTest.c* vor. Geeignete Aufrufe in der Datei *stringHistoryTest.c* sollen die Funktionalität obiger Funktionen demonstrieren.

Hinweise:

- Die Datenstruktur mit ihren Membervariablen ist dynamisch aufzubauen und dementsprechend auch wieder abzubauen.
- Die Memoryverhältnisse für verschiedene Fälle ("leere" Struktur, im Wachsen begriffen, voll) sind nachfolgend veranschaulicht.

Memoryverhältnisse an Beispielen gezeigt:

