TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professur
Betriebssysteme

Algorithmen und Programmierung

Sichere Felder (Trainingsaufgabe)

Beim Zugriff auf nichtexistierende Feldpositionen kann es bei C-Arrays zum Programmabsturz kommen. Aus dem Grund sollten nun „sichere“ Felder mithilfe von Strukturen gebaut werden. Es sei die folgende Datenstruktur definiert:

```
1 typedef struct {
2     unsigned int size;
3     int *arr;
4 } array_t;
```

Implementieren Sie die folgenden Funktionen:

1. `void initEmpty(array_t *A)`, die ein leeres sicheres Feld erzeugt,
2. `void init(array_t *A, unsigned int size)`, die ein sicheres Feld der vorgegebenen Größe erzeugt und mit Nullen initialisiert,
3. `int get(array_t *a, unsigned int n, int *err)`, die das Element an der Position n ausgibt, falls n (strikt) kleiner als die Feldgröße ist. Sonst wird 0 zurückgegeben und der Wert des Fehlerparameter `err` auf -1 gesetzt,
4. `void put(array_t *a, int n, int elem)`, die das Element elem auf die `n`-te Position schreibt, wenn `n` kleiner als die Feldgröße ist. Ansonsten soll das Feld auf Größe `2 * n` vergrößert werden. Dabei müssen alle bestehen Elemente übernommen und alle neuen auf 0 gesetzt werden. Das Element an der Position `n` muss den übergebenen Wert `elem` enthalten.
5. `void clean(array_t *a)`, die den für das Feld reservierten Speicher wieder freigibt.

Alle o.g. Funktionen sollen in der Datei `safeArray.c` implementiert werden.

Die Datei `safeArray.h` sei gegeben und wie folgt definiert:

```
1 typedef struct {
2     unsigned int size;
3     int *arr;
4 } array_t;
5
6 void initEmpty(array_t *);
7 void init(array_t *, unsigned int);
```

```
8  int get (array_t *, unsigned int, int *);
9  void put(array_t *, unsigned int, int);
10 void clean(array_t *);
```

Reichen Sie nur die C-Datei ein, ein `Makefile` oder eine Implementierung von main() wird nicht benötigt.

Hinweise zur Aufgabenstellung

Für die Lösung dieser Aufgabe benötigen Sie folgende Grundkenntnisse:

- Kontrollfluss (`if`) in C
- Funktionen in C
- Schleifen in C
- Felder in C
- Datenstrukturen (`struct`) in C
- Dynamische Speicherverwaltung (`malloc`)
- Benutzung von `gcc`

Hinweise zur Abgabe

- Erstellen Sie eine ZIP- bzw. TGZ-Archivdatei, welche die geforderten Dateien enthält.
- Fügen Sie dem Archiv keine weiteren Dateien oder Ordner hinzu.
- Reichen Sie Ihre Lösung unter `https://osg.informatik.tu-chemnitz.de/submit` ein.
- Bis zum Abgabeende (Deadline), sofern gegeben, können beliebig neue Lösungen eingereicht werden, die die jeweils älteren Versionen ersetzen.
- Ihr Programm muss auf der Testmaschine übersetzbar sein. Deren Details sind auf dem OpenSubmit-Dashboard verfügbar.
- Ihre Lösung wird automatisch validiert. Sie werden über den Abschluss der Validierung per eMail informiert.