Übung 6

Abgabe: 05. Juni 2023, 18:00 Uhr

Aufgabe 6.1: Arraylist [10 Punkte, Link zur Abgabe] Bei dieser Aufgabe geht es darum, eine bekannte Datenstruktur inklusiver funktionierender Speicherverwaltung zu implementieren.

Konkret soll eine *Arrayliste* implementiert werden: Eine Arrayliste verwendet ein Array von Schlüsseln (hier: `int`s), um Listenoperationen wie das Einfügen, Löschen und Anhängen von Schlüsseln zu unterstützen. Details finden Sie im Aldat-Skript, Kapitel 3.1.1.

Unsere Arrayliste soll die Methoden unterstützen, die in der Datei `arraylist.c` im Repository spezifiert sind. Insbesondere soll die Liste sich dynamisch vergrößern und verkleinern, wenn ihre Kapazität nicht ausreicht bzw. wenn nur wenige Elemente in der Liste sind. Schreiben Sie Ihre Implementierung ebenfalls in diese Datei.

Um die Liste zu testen stellen wir Ihnen in `main.c` Code zur Verfügung, der eine Datei im folgenden Format liest:

Listing 1: Beispieleingabe

```
1 > cat example.in
2 5
3 a 3
4 i 0 1
5 d 1
```

Ergänzen Sie außerdem die Datei `main.c` entsprechend der Details in `main.c`. Verändern Sie die Datei `main.c` *ansonsten* nicht.

Die Liste soll nach jeder ausgeführten Operation auf die Standardausgabe ausgegeben werden. Ebenso soll die initiale Liste (die Liste vor dem Ausführen der ersten Operation) auf die Standardausgabe ausgegeben werden. Beides erledigt der Code in der Datei `main.c` bereits für Sie.

Hinweis: Sie bekommen Punkte für jede erfolgreich implementierte Operation. Die Punkte gibt es aber nur, wenn die Operation keine Speicherfehler verursacht! Verwenden Sie ggf. `valgrind`.

Aufgabe 6.2: Tokenizer [10 Punkte, Link zur Abgabe] In dieser Aufgabe soll Code geschrieben werden, der einen String am Trennzeichen `','` auftrennt und in Teilstrings zerlegt. Beispiel:

Listing 2: Beispieldurchlauf

```
1 > cat testcases/test2.in
2 19
3 eins,zwei,drei,vier
4 > ./tokenizer < testcases/test2.in
5 'eins'
6 'zwei'
7 'drei'
8 'vier'
```

Ihre Aufgabe: Implementieren Sie die Funktionen in der Datei `tokenizer.c`. Die Spezifikation finden Sie in `tokenizer.h`. Die Testfälle in `testcases/` demonstrieren den Umgang mit Randfällen.

Das Grundgerüst des Programms (Einlesen und Ausgeben der Testfälle, passende Funktionsaufrufe) wird im Repository gestellt. Verändern Sie nur die Datei `tokenizer.c`.

Aufgabe 6.3: Speicherfehler [10 Punkte, Link zur Abgabe] Im Repository befindet sich Quellcode `error.c`, der etliche Probleme mit der Speicherverwaltung hat.

Identifizieren Sie alle Probleme und markieren Sie die Probleme mit einem `//`-Kommentar.

Ändern Sie anschließend den Quellcode so, dass das Programm ohne Fehler und Warnungen kompiliert und keine Speicherfehler auftreten.

Dabei:

- Ändern Sie die Ausgabe des Programmes nicht.
- Sie dürfen Zeilen hinzufügen und die vorhanden Zeilen verändern, aber keine Zeilen löschen/auskommentieren.
- Löschen Sie keine `malloc`-Aufrufe und ändern Sie keine Funktionssignaturen

Hinweis: Diese Aufgabe wird manuell korrigiert und mit bis zu 10 Punkten bewertet. Das Bewertungskript vergibt immer 0 von 0 Punkten, Sie können es aber verwenden, um Hinweise auf mögliche Fehler zu bekommen.