Institut für Informatik
Daniel Neugebauer

Wintersemester 2019/20
Tag 9

Hardwarenahe Programmierung
Gruppe 6 (Stefan)

Es ist soweit: In dieser Übung nutzen Sie dynamische Speicherverwaltung in C! Außerderm verwenden Sie `valgrind`*, um Ihre Programme auf Speicherlecks zu untersuchen.*

Aufgabe 1 *Tier-Daten II*

Versuchen Sie bei der Lösung dieser Aufgabe möglichst viele Funktionen von Tag 8 wieder zu verwenden. Passen Sie dabei die Datenstruktur und die Funktionen, die Sie an Tag 8 geschrieben haben, geeignet an.

(a) Erweitern Sie Ihre Datenstruktur von Tag 8 so, dass Sie Tiere in einer verketteten Liste speichern können.

(b) Schreiben Sie folgende Funktionen für das Einfügen von Tieren in eine Liste:

- Eine Funktion `einfuegen_ende`, die den Start einer Tier-Liste und ein neues Tier erhält und die das Tier ans Ende der Liste anhängt.
- Eine Funktion `einfuegen_start`, die den Start einer Tier-Liste und ein neues Tier erhält und die das Tier an den Start der Liste einfügt.
- Eine Funktion `einfuegen_index`, die den Start einer Tier-Liste, ein neues Tier sowie eine Index-Position i erhält, und die das Tier an Position i der Liste einfügt. Dabei ist 0 der Index der ersten Listenposition. Falls der übergebene Index zu klein/groß für die Liste ist, soll an die erste/letzte Listenposition eingefügt werden.

(c) Ändern Sie die Funktionen `read_tiere` und `print_tiere` so ab, dass diese mit einer Liste von Tieren statt einem Array von Tieren arbeiten.

(d) Schreiben Sie ein Hauptprogramm, das mithilfe Ihrer Lösungen der vorherigen Teilaufgaben eine verkettete Liste mit 5 (oder mehr) Tieren aus einer Eingabe (z.B. Datei) erzeugt und die Liste anschließend auf der Konsole ausgibt.

(e) Schreiben Sie eine Funktion, die den Start einer Liste von Tieren und einen Namen übergeben bekommt, und die das erste Tier mit diesem Namen aus der Liste entfernt und löscht.

(f) Schreiben Sie eine Funktion, die eine Tier-Liste als Eingabe erhält und den Speicher der gesamten Liste wieder freigibt.

(g) Erweitern Sie Ihr Hauptprogramm: fügen Sie nach dem Einlesen der Tiere Befehle hinzu, um drei der Tiere wieder aus der Liste zu entfernen. Demonstrieren Sie dabei das Löschen des ersten Elements der Liste, des letzten Elements der Liste, und eines Elements im Innern der Liste.

(h) Überprüfen Sie Ihr Programm mir valgrind. valgrind darf keine Fehlerquellen melden. Das heißt mindestens: Der komplette allozierte Speicher muss bei Beendigung des Programms wieder freigegeben sein. Es darf keine als *possibly* oder *definitely lost* markierten Speicherbereiche und keine *invalid reads* oder *invalid writes* geben.

Aufgabe 2 *Dynamisches Array*

In dieser Aufgabe sollen Sie eine Datenstruktur erstellen, die ein String-Array mit dynamischer Kapazität implementiert. Der Speicher für die Strings muss dazu dynamisch alloziert werden. Achten Sie auf eine sorgfältige Speicherverwaltung und auf die Erfüllung aller Unittests!

(a) Definieren Sie in der Datei *dyn_array.h* die Datenstruktur `dyn_array` mit den Feldern `length` (Größe des Arrays), `fill_level` (Füllstand des Arrays) und `strings` (das String-Array selbst). Wählen Sie für die Felder geeignete Datentypen.

(b) Implementieren Sie die in *dyn_array.h* vorgegebenen Funktionsprototypen und prüfen Sie Ihre Implementierungen mit den vorgegebenen Unittests.

(c) Überprüfen Sie Ihr Programm mit valgrind (wie bei Aufgabe 1).