Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 – WS 2017/18

Prof. Dr. Michael Rohs, Oliver Beren Kaul, M.Sc., Tim Dünte, M.Sc.

Übungsblatt 12

Alle Übungen (bis auf die erste) müssen in Zweiergruppen bearbeitet werden. Beide Gruppenmitglieder müssen die Lösung der Zweiergruppe einzeln abgeben. Die Namen beider Gruppenmitglieder müssen sowohl in der PDF Abgabe als auch als Kommentar in jeglichen Quelltextabgaben genannt werden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 25.1. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/WiSe2017/Programmieren1. Die Abgabe muss aus einer einzelnen zip-Datei bestehen, die den Quellcode, ein pdf bei Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen bitte auf.

Aufgabe 1: Geometrische Objekte und Objektlisten

Die für diese Aufgabe relevanten Dateien sind object.h, shape.h, circle.{h|c}, rect.{h|c}, object_list.{h|c} und shapes_test.c. Änderungen sind nur in rect.c und shapes_test.c nötig. Die Struktur `Shape` soll ein beliebiges geometrisches Objekt repräsentieren, für das ein Flächeninhalt berechnet werden kann. Die Struktur `Circle` soll Kreise und die Struktur `Rect` soll Rechtecke repräsentieren. Die Implementierung von `Circle` ist vorgegeben.

a) Machen Sie sich mit dem Quelltext in object.h, shape.h, circle.h und circle.c vertraut. Die Definition von Class in object.h wurde in der Vorlesung besprochen. Die Struktur `ShapeClass` erweitert Class um einen Zeiger auf eine Funktion vom Typ `AreaFun`. Funktionen dieses Typs sollen der Berechnung von Flächeninhalten dienen. circle.h und circle.c zeigen eine Implementierung für Kreise (area_circle). Implementieren Sie nach diesem Muster die Funktionen in rect.c. Überprüfen Sie Ihre Implementierung mit den Beispielaufrufen in shapes_test.c.

b) Erstellen Sie in shapes_test.c eine Objektliste mit den Objekten `c1`, `c2`, `r1` und `r2`. Berechnen Sie mit einer `for`-Schleife die Summe der Flächen der Objekte in der Liste und geben Sie das Ergebnis aus.

c) Berechnen Sie nun in shapes_test.c die Summe der Flächen der Objekte in der Liste mit Hilfe der `reduce_list`-Funktion und Ihrer Implementation von `add_area` und geben Sie das Ergebnis aus.

d) Filtern Sie die Liste, so dass eine Ergebnisliste nur der `Circles` gebildet wird. Verwenden Sie dazu die Funktion `filter_list` und definieren Sie eine Testfunktion `is_circle`.

Aufgabe 2: Binärbäume

Die für diese Aufgabe relevanten Dateien sind integer_tree.{h|c}, integer_list.{h|c} und integer_tree_test.c. In dieser Aufgabe werden Operationen für einen Binärbaum implementiert. Der Binärbaum speichert ganze Zahlen.

a) In der `main`-Funktion von integer_tree_test.c findet sich das int-Array `values`. Fügen Sie die Elemente des Arrays geordnet in einen Binärbaum ein. Geben Sie die Elemente des Baums aus, so dass sie in aufsteigend sortierter Reihenfolge erscheinen.

b) Implementieren Sie die Funktion `free_tree` in integer_tree.c, die den Speicher des Baumes freigibt. Beachten Sie, dass der Baum nur ganze Zahlen speichert.

c) Die `print_tree`-Funktion in integer_tree.c gibt Bäume so aus, dass der Wert eines Knotens in der Mitte und die linken und rechten Unterbäume links und rechts davon erscheinen. Um die Struktur sichtbar zu machen wird jeder Knoten geklammert. Die Form ist also: (<left>, <value>, <right>). Der leere Baum wird als Unterstrich dargestellt. Beispiel:

```
(((_, 11, _), 1, (_, 12, _)), 0, (_, 2, (_, 7, _)))
```

Erweitern Sie die `print_tree`-Funktion so, dass Blätter in abgekürzter Form dargestellt werden. Für das Beispiel:

```
((11, 1, 12), 0, (_, 2, 7))
```

d) Implementieren Sie in integer_tree_test.c die Funktion `get_interval_rec`, die einen Baum, eine untere und eine obere Grenze übergeben bekommt. Die Funktion soll aus dem übergebenen Baum alle Elemente in der Ergebnisliste speichern und zurückgeben, die im Intervall [`lower`, `upper`] liegen.

e) Implementieren Sie nun das Filtern eines Intervalls auf andere Weise. Verwenden Sie die Funktion `filter_tree` und implementieren Sie die Funktion `in_interval`.

f) Implementieren Sie die Funktion `is_monotone`. Diese gibt genau dann wahr zurück, wenn für jeden Knoten im Baum gilt, dass die Werte der Kinder größer sind, als der Wert des Knotens. Für den leeren Baum soll die Funktion den Wert `true` zurückgeben.

Aufgabe 3: Table of Contents

In dieser Aufgabe erstellen Sie eine Datenstruktur, die ein Dokument wie eine Bachelorarbeit speichern kann. Zur Vereinfachung speichert die Struktur nur die Kapitelnamen, sowie die Anzahl der Seiten für jedes Kapitel. Schauen Sie sich TOC.c und TOC.h an. Nutzen Sie für das Allokieren von Speicher ausschließlich xmalloc bzw. xcalloc.

a) Implementieren Sie die Konstruktorfunktion `new_TNode`, die den Titel eines Kapitels, die Anzahl der Seiten dieses Kapitels sowie einen Zeiger auf eine Liste von Unterkapiteln übergeben bekommt. Als Rückgabe liefert Sie einen Pointer auf eine neue TNode. Implementieren Sie zusätzlich die Funktion `new_node`, die eine neue Node erstellt. `new_node` bekommt einen Verweis auf eine TNode übergeben, sowie einen Pointer auf eine nachfolgende Node.

b) Implementieren Sie die Funktionen `free_TNode` und `free_Nodes`. Die erste Funktion gibt den von einem Kapitel allokierten Speicher frei (einschließlich der Unterkapitel). Die zweite Funktion gibt den Speicher der übergebenen Node sowie aller nachfolgenden Nodes und deren Kapitel frei. Am Ende soll der komplette vom Baum allokierte Speicher freigegeben werden.

c) Berechnen Sie die Anzahl der Seiten, die ein Kapitel benötigt. Implementieren Sie dafür die Funktion `calculate_pages`, die eine TNode übergeben bekommt und die Summe der Seiten zurückgibt. Gehen Sie davon aus, dass jedes Kapitel (auch Unterkapitel) auf einer neuen Seite beginnt. Die Anzahl der Seiten für ein Kapitel ist die Summe aus den

Seiten des Kapitels plus die Anzahl der Seiten der Unterkapitel. Bsp. Das Kapitel Introduction hat eine eigene Seite plus die Seiten aus Motivation, Research Questions und Goals. Geben Sie die Anzahl der Seiten der Bachelorarbeit auf der Konsole aus.

d) Implementieren Sie die Funktion `print_TOC`, die das Inhaltsverzeichnis gut formatiert auf der Konsole ausgibt. Dazu gehören, die Hierarchie, Der Titel, sowie die Seitenzahl. Vgl. Sie folgendes Bild:

```
The thesis has: 127 pages.
My Bachelorthesis
1 Introduction ..........................   0
1.1 Motivation ..........................   1
1.2 Research Questions ..................   3
1.3 Goals ...............................   4
2 Basics ................................   7
2.1 Mathematical Basics ..................   8
2.2 Related Work .........................  11
2.3 Interaction Techniques ...............  21
3 Concept ...............................  26
3.1 Old System .........................  27
3.1.1 Structure .......................  31
3.1.2 Functions .......................  33
3.2 New System .........................  35
3.2.1 Functions .......................  41
3.2.2 Structure .......................  43
3.3 Changes ............................  45
3.3.1 Hardware Changes ................  48
3.3.2 Software Changes ................  50
4 Prototype .............................  52
4.1 Hardware ...........................  53
4.2 Software ...........................  65
5 Studies ...............................  73
5.1 Study One ..........................  74
5.1.1 Participants ....................  75
5.1.2 Setup ..........................  76
5.1.3 Procedure ......................  79
5.2 Study Two ..........................  83
5.2.1 Setup ..........................  84
5.2.2 Procedure ......................  86
5.2.3 Participants ....................  88
6 Evaluation ............................  90
6.1 Methods ............................  91
6.2 Results ............................  95
7 Conclusion ............................  97
8 Literature ............................  99
9 Apendix .............................. 102
```

Hinweise zum Editieren, Compilieren und Ausführen:

- mit Texteditor `file.c` editieren und speichern
- `make file` ← ausführbares Programm erstellen
- `./file` ← Programm starten (evtl. ohne `./`)
- Die letzten beiden Schritte lassen sich auf der Kommandozeile kombinieren zu:
  `make file && ./file`