Prof. Dr. A. Weber
Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
29.11.2018

Wintersemester 2018/2019

Übungen zur Vorlesung
Algorithmisches Denken und imperative Programmierung (BA-INF-014)
Aufgabenblatt 6

Zu bearbeiten bis: 08.12.2018

Aufgabe 1 *(Matrizenmultiplikation - 4 Punkte)*

Gegeben seien zwei Matrizen $A \in \mathbb{R}^{I \times J}$ und $B \in \mathbb{R}^{K \times L}$. Das Produkt $C = A \times B \in \mathbb{R}^{I \times L}$ kann berechnet werden, falls die Spaltenanzahl J der Matrix A gleich der Zeilenanzahl K der Matrix B ist. Die Einträge der Matrix C ergeben sich durch:

$$c_{i,l} = \sum_{k=1}^{K} a_{i,k} b_{k,l} \qquad i = 1 \dots I, \quad l = 1, \dots, L.$$

Schreiben Sie ein Programm, das zwei Matrizen miteinander multipliziert. Falls die Anzahl der Spalten der ersten Matrix ungleich der Anzahl der Zeilen der zweiten Matrix ist, soll eine entsprechende Fehlermeldung ausgegeben werden.

Aufgabe 2 *(Struct - 7 Punkte)*

Gegeben sei folgende Struktur:

```
typedef struct {
   char nachname[30];
   float note;
} student;
```

Schreiben Sie ein Programm unter Benutzung dieser Struktur, welches folgende Funktion erfüllen soll:

- Das Programm erfragt vom Benutzer die Anzahl N Studenten, zu welchen im nächsten Schritt Daten eingegeben werden sollen.
- Als nächstes werden N Datensätze bestehend aus Name und Note eingelesen. Überprüfen Sie dabei auch, ob die eingegebene Note gültig ist (Noten zwischen 1.0 und 5.0). Bei ungültiger Eingabe soll erneut ein Wert angegeben werden können.
- Das Programm gibt die Namen aller Studenten aus, welche die beste Note erhalten haben. (es können mehrere Studenten zugleich die besten sein!). Die beste Note ist nicht notwendigerweise 1.0 sondern die beste der erreichten Note.
- Das Programm berechnet schliesslich die Durchschnittsnote aller Studenten und gibt die Durchschnittsnote aus.

Stellen Sie selbst Speicher auf dem Heap bereit, indem Sie die Funktion *malloc* nutzen. Vergessen Sie nicht, den Speicher schließlich wieder freizugeben.

Aufgabe 3 *(Studentendatenbank - 0,5+0,5+5+1=7 Punkte)*

a) Schreiben Sie eine Struktur *student*, die eine Matrikelnummer, einen Vornamen und einen Nachnamen mit jeweils passenden Datentypen enthält.
b) Deklarieren Sie einen Pointer auf Ihre Struktur, allokieren Sie den notwendigen Speicher und initialisieren Sie Matrikelnummer, Vornamen und Nachnamen mit sinnvollen Werten. Schreiben Sie anschließend eine Funktion *print_student*, die einen Pointer auf einen Studenten übergeben bekommt und seine Details in einer einzelnen Zeile ausgibt. Testen Sie die Funktion mit ihrem eben angelegten Beispiel.
c) Legen Sie jetzt ein Array von Pointern auf Studenten der Größe 20 an. Initialisieren Sie zunächst alle Elemente mit NULL. Schreiben Sie folgende Operationen, die auf dem Array arbeiten sollen:

- *print_students*, die das Array übergeben bekommt und die Funktion print_student verwendet, um Details aller im Array vorhandenen Studenten auszugeben.
- *clear_students*, die das Array übergeben bekommt und es zurücksetzt. Vergessen Sie nicht, den Speicher der enthaltenen Elemente freizugeben.
- *get_name*, die das Array und eine Matrikelnummer übergeben bekommt, diese im Array sucht und anschließend bei einem Treffer den dazugehörigen Namen ausgibt.

- *add_student*, die das Array und Daten zum einzufügenden Studenten (Matrikelnummer, Vorname, Nachname) übergeben bekommt und diesen in das Array einfügt, sofern noch Platz vorhanden ist.
- *remove_student*, die das Array und eine Matrikelnummer übergeben bekommt, diese im Array sucht und das zugehörige Element entfernt.

d) Testen Sie Ihre Funktionen mit der folgenden Abfolge, wobei die nicht genannten Parameter von Ihnen frei wählbar sind:
add_student mit Matrikelnummer 42; get_name 42; add_student mit Matrikelnummer 30; remove_student mit Matrikelnummer 42; print_students; clear_students; print_students

Aufgabe 4 *(Integrale - 2 Punkte)*

In C ist es auch möglich, Pointer auf Funktionen zu definieren. Sie müssen sich in dieser Aufgabe nicht mit ihrer Verwendung auseinandersetzen. Implementieren Sie die unten dargestellte Funktion *integrate*, die $\int_l^r fnct(x)dx$ approximiert und testen Sie anschließend Ihre Funktion durch Ausführung des vorgegebenen Testlaufs. Die Werte sollten nahe an denen im Kommentar beschriebenen Referenzen liegen.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x) {
   return 0.4*pow(x,5)+pow(x,3);
}

double integrate(double(*fnct)(double), double left, double right, double stepsize) {
   // Sie koennen die Funktion fnct wie gewohnt aufrufen
   // Ihre Implementierung ...
}

int main(int argc, char**argv) {
   printf("Integral von Sinus [0, pi]: %f\n", integrate(&sin, 0, M_PI, 0.0001)); //2
   printf("Integral von Sinus [0, 2pi]: %f\n", integrate(&sin, 0, 2*M_PI, 0.0001)); //0
   printf("Integral von f(x)=0.4x^5+x^3 [0, 3]: %f\n", integrate(&f, 0, 3, 0.0001)); //ca. 68,84
   return 0;
}
```