Prof. Dr. A. Weber
Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
21.11.2016

Wintersemester 2016/2017

Übungen zur Vorlesung
Algorithmisches Denken und imperative Programmierung (BA-INF-014)
Aufgabenblatt 5

Zu bearbeiten bis: 03.12.2016

Hinweis: In der Woche vom 28.11. - 01.12. findet die erste Online-Aufgabe zu Ihren gewohnten Tutorienzeiten statt, bei der Sie eine vor Ort gestellte Aufgabe lösen müssen. Daher haben Sie für die Bearbeitung dieses Übungszettels zwei Wochen Zeit, die Abgabefrist ist der 03.12.16.

Aufgabe 1 *(Parameterübergabe - 1+1=2 Punkte)*

Gegeben sei das folgende C-Programm.

```
#include <stdio.h>

void cbR(int *x) {
   printf("Ergebnis 1: %d.\n", *x);
   (*x) += 12;
   printf("Ergebnis 2: %d.\n", *x);
}

void cbV(int x) {
   printf("Ergebnis 4: %d. \n", x);
   x += 12;
   printf("Ergebnis 5: %d.\n", x);
}

int main(int argc, char**argv) {
   int a=10;

   cbR(&a);
   printf("Ergebnis 3: %d.\n", a);

   cbV(a);
   printf("Ergebnis 6: %d.\n", a);

   return 0;
}
```

- Welche Ergebnisse liefert das Progamm? Begründen Sie Ihre Antwort.
- Erklären Sie allgemein den Unterschied zwischen „call by reference“ und „call by value“.

Aufgabe 2 *(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:

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

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 3 *(Umgang mit Strings - 2+3=5 Punkte)*

a) Palindrome sind Wörter, die vorwärts und rückwärts gelesen gleich sind, z.B. Ebbe.

- Schreiben Sie eine Funktion, die als Argument einen String erhält und diesen umgekehrt ausgibt.
- Schreiben Sie eine Funktion, die zurückgibt, ob ein als Argument übergebener String ein Palindrom ist.

b) Die Caesar-Verschlüsselung ist ein einfaches symmetrisches Verschlüsselungsverfahren.

- Implementieren Sie die Funktionen *char*encrypt(char*s)* bzw. *char*decrypt(char*s)*, die einen übergebenen String mittels ROT13 verschlüsseln bzw. entschlüsseln. Dabei wird jeweils jeder Buchstabe durch seinen dreizehnten Nachfolger im Alphabet ersetzt. Sie müssen dabei nur das lateinische Alphabet mit seinen 26 Buchstaben betrachten, alle anderen Zeichen lassen Sie unverändert. **Tipp:** Betrachten Sie für die Lösung der Aufgabe unbedingt eine ASCII-Tabelle. Testen Sie Ihre Funktion, indem Sie einen vom Benutzer Ihres Programms eingegebenen String einlesen, diesen verschlüsseln, ausgeben und direkt wieder entschlüsseln.
- Schreiben Sie nun eine Funktion *char*encrypt(char*s, int k)* sowie *char*decrypt(char*s, int k)*, die um eine beliebige Stelle k, $k \in \mathbb{N}_0, k < 27$ verschiebt.

Aufgabe 4 *(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 5 *(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;
}
```