Prof. G. Zachmann
Dipl.-Inf. C. Giesemann

TU Clausthal
Institut für Informatik
15. Dezember 2005

Wintersemester 2005/2006

Übungen zu Grundlagen der Programmierung in C - Blatt VII

Abgabe vom 21.12.2005 bis 10.1.2006 in der angemeldeten Übung

Aufgabe 1 (Matrixmultiplikation, *8 Punkte*)

Hinweise:
Das Produkt von zwei N x N Matrizen $A = (a_{ij})_{1 \le i \le N, 1 \le j \le N}$ und $B = (b_{ij})_{1 \le i \le N, 1 \le j \le N}$ zu Matrix $C = (c_{ij})_{1 \le i \le N, 1 \le j \le N}$:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \dots & b_{NN} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1N} \\ c_{21} & c_{22} & \dots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \dots & c_{NN} \end{pmatrix}$$

läßt sich wie folgt berechnen:

$$c_{ij} = \sum_{k=1}^{N} a_{ik} \cdot b_{kj}, \forall i, j : 1 \le i \le N, 1 \le j \le N$$

N soll als Konstante mit `const int N = <Wert>;` im Programm deklariert werden. Überall, wo die Größe **N** benötigt wird, soll mit der Konstante gearbeitet werden.
Speichern Sie die Matrizen als 2-dimensionale Arrays. Verwenden Sie die Konstante **N** als Arraygröße. Beachten Sie den Unterschied zwischen Index (von 0 bis N-1) und der zugehörigen mathematischen Darstellung (von 1 bis N).
Mittels 3 ineinander geschachtelter `for` Schleifen kann die Multiplikation durchgeführt werden.

1. Schreiben Sie ein Programm, das zwei 3 x 3 Matrizen in Arrays einliest, das Produkt berechnet und, wie oben angedeutet, Matrizen und Produkt (in Spalten angeordnet, ohne Klammern) ausgibt.
2. Ändern Sie das Programm, um zwei 5 x 5 Matrizen einzulesen.

Aufgabe 2 (Enums und Structs, *6 Punkte*)

Gegeben seien folgende Typ-Definitionen:

```
enum State
{
    DOWN,
    UP
};
```

```
struct MachineRec
{
    int id;
    float failRate;
    State state;
    float downTimeInSeconds;
    float cost;
};
```

1. Schreiben Sie ein Programm, in dem folgende Werte in 2 Variablen vom Typ `MachineRec` gespeichert werden. Die Werte sollen im Programm fest zugewiesen werden. Geben Sie im Programm den Inhalt dieser Records auf dem Bildschirm aus. Dabei soll der Wert von *state* im Klartext ausgegeben werden ("DOWN" oder "UP"). Die *down time* soll nur dann ausgegeben werden, wenn *state* den Wert DOWN hat.

   Maschine 1:
   - id: 17
   - failRate: 0.01
   - state: DOWN
   - downTimeInSeconds: 70
   - cost: 599,90

   Maschine 2:
   - id: 26
   - failRate: 0.06
   - state: UP
   - downTimeInSeconds: 0
   - cost: 399,00

Aufgabe 3 (Filmdatenbank, *8 Punkte*)

Hinweise:
Man kann mit `scanf` Minuten:Sekunden einlesen, in dem man den `:` als Trennzeichen im Formatstring verwendet: `scanf("%d:%d", &arg1, &arg2);`

1. Schreiben Sie ein Programm, das eine kleine Filmdatenbank mit maximal 100 (als Konstante siehe Aufgabe 1 deklariert) Filmen realisiert. Zu jedem Film müssen folgende Daten gespeichert werden:
   - Titel
   - Genre
   - Regisseur
   - Produktionsjahr (als `int`)
   - Produktionsland
   - Laufzeit in Minuten:Sekunden (als weitere eingebundene `struct` mit zwei `int`)

   Die Möglichkeiten bei Genre sollen zuvor als `enum` deklariert werden. Zulässig sind nur Action, Thriller, Science-Fiction, Drama, Horror und Musical. Der Benutzer soll angeben, wie viele Filme eingegeben werden sollen. Das Programm soll überprüfen, ob der Wert zulässig ist und dann den Benutzer nacheinander auffordern, was er als nächstes eingeben soll. Nach dem letzten Eintrag soll der zuvor eingelesene Inhalt der Datenbank ausgegeben werden. Überlegen Sie sich, wie Sie das Genre einlesen wollen. Bei der Ausgabe soll das Genre im Klartext ausgegeben werden. Weisen Sie den Benutzer darauf hin, wie er die Laufzeit einzugeben hat und geben Sie diese mit `:` getrennt auch wieder aus.