Dipl.Phys. Gerald Kempfer

Beuth Hochschule für Technik Berlin
- University of Applied Sciences -
Fachbereich VI – Informatik und Medien
Studiengang Technische Informatik Bachelor

Lehrveranstaltung "Informatik I für TI-Bachelor"
Übungsblatt 9

Hinweise:

Dieses Übungsblatt ist zur Zulassung zu der Klausur erfolgreich zu bearbeiten ("*Erfolgreich*" bedeutet: Keine Programmabstürze bzw. Endlosschleifen, Aufgabenstellung einschl. der Nebenbedingungen müssen eingehalten sowie Kommentierung und Einrückung korrekt sein!).

Die Aufgaben werden überwiegend in den Übungszeiten bearbeitet. Allerdings genügt die Zeit hierfür unter Umständen nicht, so dass Sie auch außerhalb dieser Zeiten die Aufgaben bearbeiten müssen. Der Abgabetermin für diese Aufgabe ist **spätestens** der **10. Juli 2015**.

Nutzen Sie die Übungen auch, um ggf. Fragen, die sich in den Vorlesungen ergeben haben, anzusprechen.

Aufgabe:

Ziel dieser Übung ist das Verwenden von Arrays (speziell zweidimensionale Arrays) und Funktionen.

Erstellen Sie die Funktionen zum vorgegebenen Hauptprogramm. Die Funktionsschnittstellen werden in der Headerdatei `matrix.h` vorgegeben. Dort finden Sie auch die Beschreibungen zu den einzelnen Funktionen. Das Hauptprogramm sowie die Headerdatei dürfen nicht verändert werden! Sie dürfen aber in dem zu erstellenden Modul weitere lokale (`static`-) Funktionen hinzufügen. So ist es z.B. sinnvoll, dass die Funktion `getMatrix` zum Einlesen der einzelnen Zahlen eine Funktion `getNumber` aufruft.

Ferner müssen Sie die beiden unveränderlichen(!) Variablen `MaxRows` und `MaxColumns` in Ihrem Modul `matrix.c` als globale Variablen anlegen. Initialisieren Sie beide Variablen mit einem beliebigen Wert (z.B. `5`). Ihre Funktionen müssen so flexibel sein, dass sie nach einer Änderung dieser Variablen und einem Neucompilieren des Programmes noch korrekt funktionieren.

In einem weiteren Modul `tools.c` und der dazugehörigen Headerdatei `tools.h` werden die folgenden zwei Funktionen untergebracht:

```
void clearBuffer();
int askAgain(int Row, int Col);
```

Schreiben Sie für diese Funktionen auch die Beschreibungen in der Headerdatei (analog zur vorgegebenen Headerdatei `matrix.h`).

Das Programm soll benutzerfreundlich sein, d.h. dem Benutzer soll mitgeteilt werden, was er tun soll und was er falsch gemacht hat.

Das Compilieren, Linken und Starten des Programms soll wieder mittels einer Make-Datei durchgeführt werden.

Zur Erinnerung:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{pmatrix}$$

wobei $c_{ij} = \sum_{k=1}^{2} a_{ik} * b_{kj}$ mit $i = 1 \ldots 3$ und $j = 1 \ldots 4$

Beispiel:

```
Geben Sie bitte die Zeilenanzahl der 1. Matrix ein (1..8): 8___
Geben Sie bitte die Spaltenanzahl der 1. Matrix ein (1..8): 2___

|   1   2 |
|   2   1 |
|   1   2 |
|   2   1 |
|   1   2 |
|   2   1 |
|   1   2 |
|   2   1 |

Geben Sie bitte die Spaltenanzahl der 2. Matrix ein (1..8): 8___
|   1   2   3   4   5   6   7   8 |
|   8   7   6   5   4   3   2   1 |


|   1   2 |    |   1   2   3   4   5   6   7   8 |    |  17  16  15  14  13  12  11  10 |
|   2   1 |    |   8   7   6   5   4   3   2   1 |    |  10  11  12  13  14  15  16  17 |
|   1   2 |  * |                                     =  |  17  16  15  14  13  12  11  10 |
|   2   1 |                                          |  10  11  12  13  14  15  16  17 |
|   1   2 |                                          |  17  16  15  14  13  12  11  10 |
|   2   1 |                                          |  10  11  12  13  14  15  16  17 |
|   1   2 |                                          |  17  16  15  14  13  12  11  10 |
|   2   1 |                                          |  10  11  12  13  14  15  16  17 |

Moechten Sie noch einmal (j/n) ? j
```

ueb09.c:

```
#include <stdio.h>
#include "escapesequenzen.h"
#include "matrix.h"
#include "tools.h"

int main()
{
   int Matrix1[MaxRows][MaxColumns];
   int Matrix2[MaxRows][MaxColumns];
   int MatrixResult[MaxRows][MaxColumns];
   int RowCount1, ColCount1;
   int RowCount2, ColCount2;
   int RowCountResult, ColCountResult;

   do
   {
      CLEAR;
      HOME;

      RowCount1 = getNumberOfRows(1, 1, 1);
      ColCount1 = getNumberOfColumns(2, 1, 1);
      getMatrix(4, 1, RowCount1, ColCount1, Matrix1);

      RowCount2 = ColCount1;
      ColCount2 = getNumberOfColumns(5 + RowCount1, 1, 2);
      getMatrix(6 + RowCount1, 1, RowCount2, ColCount2, Matrix2);

      RowCountResult = RowCount1;
      ColCountResult = ColCount2;
      calcMatrix(RowCount1, ColCount1, ColCount2, Matrix1, Matrix2, MatrixResult);

      CLEAR;
      HOME;
      printMatrix(1,  1, RowCount1, ColCount1, Matrix1);
      POSITION((RowCount1 + RowCount2) / 4 + 1, ColCount1 * 5 + 6);
      printf("*");
      printMatrix(1, ColCount1 * 5 + 9, RowCount2, ColCount2, Matrix2);
      POSITION((RowCount1 + RowCount2) / 4 + 1, (ColCount1 + ColCount2) * 5 + 14);
      printf("=");
      printMatrix(1, (ColCount1 + ColCount2) * 5 + 17, RowCountResult, ColCountResult, MatrixResult);
   } while (askAgain(2 + ((RowCount1 > RowCount2) ? RowCount1 : RowCount2), 1));
   CLEAR;
   HOME;
   printf("Programm wurde beendet!\n\n");
   return 0;
}
```

matrix.h:

```
#ifndef MATRIX_H
  #define MATRIX_H MATRIX_H

  /*************************************************************************
   * maximale Groesse der Matrizen
   *************************************************************************/
  extern int const MaxRows, MaxColumns;

  /*************************************************************************
   *  getNumberOfRows
   *  Fragt den Benutzer nach der Anzahl der Matrixzeilen
   *  Parameter: Row, Col: Zeile und Spalte, in der die Benutzerfrage beginnt
   *             MatrixNo: Nr. der Matrix, fuer die nach der Zeilenanzahl gefragt wird
   *  Ergebnis : Gibt die Anzahl der Matrixzeilen zurueck
   *             (ist garantiert eine Zahl zwischen 1 und MaxRows!)
   *************************************************************************/
  int getNumberOfRows(int Row, int Col, int MatrixNo);

  /*************************************************************************
   *  getNumberOfColumns
   *  Fragt den Benutzer nach der Anzahl der Matrixspalten
   *  Parameter: Row, Col: Zeile und Spalte, in der die Benutzerfrage beginnt
   *             MatrixNo: Nr. der Matrix, fuer die nach der Spaltenanzahl gefragt wird
   *  Ergebnis : Gibt die Anzahl der Matrixspalten zurueck
   *             (ist garantiert eine Zahl zwischen 1 und MaxColumns!)
   *************************************************************************/
  int getNumberOfColumns(int Row, int Col, int MatrixNo);

  /*************************************************************************
   *  getMatrix
   *  Schreibt eine leere Matrix auf den Bildschirm (fuer die Zahlen werden
   *  erst einmal Unterstriche geschrieben). Dann werden die Werte der Matrix
   *  zeilenweise an den entsprechenden Positionen in der Matrix vom Benutzer
   *  erfragt. Jeweils nach der Eingabe einer Zahl wird diese gleich noch
   *  einmal formatiert an die entsprechende Position geschrieben.
   *  Die Zahlen in der Matrix duerfen maximal vier Zeichen lang sein
   *  (d.h. die Zahlen duerfen nur im Bereich von -999 bis 9999 liegen!)
   *  Parameter: Row, Col: Zeile und Spalte, in der die Matrix auf dem Bildschirm beginnt
   *             RowCount, ColCount: Zeilen- und Spaltenanzahl der genutzten Matrix
   *             Matrix: Matrix, fuer die die Zahlen eingelesen werden
   *  Ergebnis : nichts
   *************************************************************************/
  void getMatrix(int Row, int Col, int RowCount, int ColCount, int Matrix[MaxRows][MaxColumns]);

  /*************************************************************************
   *  calcMatrix
   *  Multipliziert zwei Matrizen.
   *  Parameter: RowCount1, ColCount1: Zeilen- und Spaltenanzahl der ersten Matrix
   *             ColCount2: Spaltenanzahl der zweiten Matrix
   *             M1, M2: die Matrizen, die multipliziert werden
   *             MResult: Ergebnis-Matrix
   *  Ergebnis : nichts
   *************************************************************************/
  void calcMatrix(int RowCount1, int ColCount1, int ColCount2, int M1[MaxRows][MaxColumns],
                  int M2[MaxRows][MaxColumns], int MResult[MaxRows][MaxColumns]);

  /*************************************************************************
   *  printMatrix
   *  Gibt eine Matrix auf dem Bildschirm aus.
   *  Parameter: Row, Col: Zeile und Spalte, in der die Matrixausgabe beginnt
   *             RowCount, ColCount: Zeilen- und Spaltenanzahl der Matrix
   *             Matrix: Matrix, die auf dem Bildschirm ausgegeben werden soll
   *  Ergebnis : nichts
   *************************************************************************/
  void printMatrix(int Row, int Col, int RowCount, int ColCount, int Matrix[MaxRows][MaxColumns]);

#endif
```