Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
27.11.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1. Schreiben Sie ein Programm, welches ein Wort einliest und überprüft ob es sich bei dem eingegeben Wort um ein Palindrom handelt. Ein Palindrom ist ein Wort, welches von vorne und hinten gelesen gleich lautet, z.B.: Anna, Otto, Reliefpfeiler. Speichern Sie den Source-Code unter `palindrom.c` in das Verzeichnis `serie09`.

Aufgabe 9.2. Schreiben Sie eine Funktion `unique`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert, doppelte Einträge streicht und den Vektor in gekürzter Form zurückgibt. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Die Länge n der Vektoren ist dynamisch zu realisieren. Schreiben Sie ein aufrufendes Hauptprogramm, in dem n und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird. Speichern Sie den Source-Code unter `unique.c` in das Verzeichnis `serie09`.

Aufgabe 9.3. Schreiben Sie eine Funktion `transpose`, die zu einer dynamisch gespeicherten Matrix $A \in \mathbb{R}^{m\times n}$ (vom Typ `double**`) die transponierte Matrix $A^T \in \mathbb{R}^{n\times m}$ berechnet. Dabei sind die Einträge von A^T gerade durch $(A^T)_{jk} = A_{kj}$ definiert. Im Hauptprogramm sollen die Dimensionen $m, n \in \mathbb{N}$ sowie die Einträge der Matrix A eingelesen und A^T ausgegeben werden. Speichern Sie den Source-Code unter `transpose.c` in das Verzeichnis `serie09`.

Aufgabe 9.4. Schreiben Sie Funktionen `int countValueInRow(int** matrix, int m, int n, int val, int row)` und `int countValueInColumn(int** matrix, int m, int n, int val, int col)`, die zu einer gegebenen $m \times n$-Matrix die Anzahl der Einträge in der gegebenen Zeile/Spalte zurückliefern, die mit `val` übereinstimmen. Speichern Sie den Source-Code unter `countValueInRowCol.c` in das Verzeichnis `serie09`.

Aufgabe 9.5. Schreiben Sie ein Programm, welches das bekannte Spiel *Tic Tac Toe* realisiert. Ihr Hauptprogramm könnte z.B. so aussehen:

```
int player=1;
int winner;
int** playboard = newPlayboard();
resetPlayboard(playboard);
while(!isGameFinished(playboard)){ // Solange noch kein Sieger klar & Felder frei.
    makeMove(playboard,player);  // Lese Spielzug von der Tastatur ein
                                 // und speichere ihn ins Spielfeld.
    printPlayboard(playboard);   // Gib die neue Spielsituation am Bildschirm aus.
    player=changePlayer(player); // Wechsle den Spieler.
}
winner=getWinner(playboard);
printWinnerMessage(winner);
delPlayboard(playboard);
```

Die Funktion `makeMove` sollte dabei den Spieler solange nach einem Spielzug fragen, bis dieser gültig ist. (Felder dürfen nicht überschrieben werden!) Für die Funktion `getWinner` können Sie unter anderem Aufgabe 9.4 verwenden. Die Funktion `isGameFinished` kann wiederum die Funktion `getWinner` verwenden. Speichern Sie den Source-Code unter `tictactoe.c` in das Verzeichnis `serie09`.

Aufgabe 9.6. Schreiben Sie eine Funktion `getRowSum` welche die für jede Zeile der übergebenen Matrix $A \in \mathbb{R}^{m\times n}$ die Zeilensumme bestimmt und in einem Array a zurückgibt, d.h. $a_j = \sum_{k=0}^{n-1} A_{jk}$ für alle $j = 0, \dots, m-1$. Der Speicher für das Array a soll dynamisch allokiert werden und die Matrix sei vom Typ `double **`. Der Funktionskopf kann daher folgendermaßen aussehen

```
double * getRowSum(double ** matrix, int m, int n)
```

Speichern Sie den Source-Code unter `getRowSum.c` in das Verzeichnis `serie09`.

Aufgabe 9.7. Schreiben Sie eine Bibliothek zur Verwaltung von *spaltenweise* gespeicherten $m \times n$-Matrizen. Implementieren Sie die folgenden Funktionen

- `double* mallocmatrix(int m, int n)`
  Allokieren von Speicher für eine spaltenweise gespeicherte $m \times n$- Matrix.
- `double* freematrix(double* matrix)`
  Freigeben des allokierten Speichers einer Matrix.
- `double* reallocmatrix(double* matrix, int m, int n, int mNew, int nNew)`
  Reallokieren und initialisieren von neuen Einträgen.

Speichern Sie die Funktionssignaturen in das Header-File `dynamicmatrix.h`. Schreiben Sie auch entsprechende Kommentare zu den Funktionen in das Header-File. In die Datei `dynamicmatrix.c` kommt dann die Implementierung der Funktionen.

Aufgabe 9.8. Erweitern Sie die Bibliothek aus Aufgabe 9.7 um folgende Funktionalitäten

- `void printmatrix(double* matrix, int m, int n)`
  Gibt eine spaltenweise gespeicherte $m \times n$-Matrix als Matrix am Bildschirm aus. Die 2×3-Matrix `double matrix[6]={1,2,3,4,5,6}` soll wie folgt ausgegeben werden:

  ```
  1 3 5
  2 4 6
  ```

- `double* scanmatrix(int m, int n)`
  Allokiert Speicher für eine Matrix und liest die Koeffizienten der Matrix von der Tastatur ein.
- `double* cutOffRowJ(double* matrix, int m, int n, int j)`
  Schneidet die j-te Zeile aus einer $m \times n$-Matrix heraus.
- `double* cutOffColK(double* matrix, int m, int n, int k)`
  Schneidet die k-te Spalte aus einer $m \times n$-Matrix heraus.