Vor allem die mathematischen Wissenschaften zeichnen sich aus durch Ordnung, Symmetrie und Beschränkung; und dies sind die größten Formen des Schönen.
Aristoteles
(384-322 v.Chr., Philosoph)

UNIVERSITÄT DES SAARLANDES

FR Mathematik
Dr. S. Weißer
D. Seibel, B. Sc.

0. Übung zur Vorlesung

Numerik II im SoSe 2018

Keine Abgabe

Bitte melden Sie sich bis spätestens **20. April 2018** zur Vorlesung an. Dies geschieht auf der Internetseite der Vorlesung:
`www.num.uni-sb.de/numerik2`

Dieses Blatt dient als Vorbereitung für spätere Programmieraufgaben und soll die Kenntnisse in C auffrischen. Es bleibt unbenotet und wird in der dritten Vorlesungswoche besprochen.

Aufgabe 0.1. (0 Punkte)

Schreiben Sie ein C-Programm, das vom Benutzer eine Jahreszahl einliest und ausgibt, ob es sich um ein Schaltjahr handelt. Achten Sie darauf, dass keine negativen Zahlen eingegeben werden können. Ein Jahr ist ein Schaltjahr, falls es durch 4 teilbar ist. Kann man die Jahreszahl außerdem durch 100 teilen, so handelt es sich nur dann um ein Schaltjahr, falls sie auch noch durch 400 teilbar ist.

Aufgabe 0.2. (0 Punkte)

Erstellen Sie ein Programm in C, das die Kurve

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \sqrt{t}\cos(\pi t) \\ \sqrt{t}\sin(\pi t) \end{pmatrix}$$

auf dem Intervall $[0, 4]$ in 101 äquidistanten Punkten auswertet und anschließend die Kurvenpunkte in eine Datei ausgibt. Die Datei soll so aufgebaut sein, dass jeder Kurvenpunkt in einer Zeile steht und die Koordinaten durch ein Leerzeichen oder einen Tabulator getrennt sind. Haben Sie die Datei erzeugt, so können Sie die Kurve mit `gnuplot` visualisieren. Fügen Sie dem Graphen auch eine Überschrift hinzu und beschriften Sie die Koordinatenachsen.
Hinweise: Um die Sinus- und Cosinusfunktion, sowie die Konstante π (`M_PI`) verwenden zu können, müssen Sie die Headerdatei „math.h“ einbinden und beim Kompilieren die Mathematikbibliothek linken. Letzteres erreichen Sie durch

```
gcc ihr_programm.c -lm
```

Die Datei muss zuerst mit `fopen()` geöffnet werden, um anschließend mit `fprintf()` in sie zu schreiben. Vergessen Sie nicht, die Datei mit `fclose()` wieder zu schließen. Hier ein kleines Beispiel, das eine Datei öffnet und Zahlen und Worte, getrennt durch Tabulatoren auf zwei Zeilen verteilt, schreibt.

```c
#include <stdio.h>
#include <stdlib.h>
...
FILE *file = fopen("test.dat", "w");
if(file == NULL) {
  fprintf(stderr, "Fehler beim Oeffnen der Datei\n");
  exit(1);
}
fprintf(file, "%d\t %f\n", 1, 2.3);
fclose(file);
...
```

Ein FAQ und die Dokumentation zu `gnuplot` finden Sie unter

`http://www.gnuplot.info/`

Aufgabe 0.3. (0 Punkte)

Schreiben Sie ein Programm in C, das zwei Matrizen A und B im ASCII-Format aus Textdateien ausliest, das Matrixprodukt $C = AB$ berechnet und das Ergebnis C im ASCII-Format in einer Textdatei abspeichert. Testen Sie Ihre Implementierung für die Matrizen

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{3\times 4} \quad \text{und} \quad B = \begin{pmatrix} 0 & 2 \\ 1 & 4 \\ 3 & 6 \\ 9 & 8 \end{pmatrix} \in \mathbb{R}^{4\times 2}.$$

Das Format für die Textdateien sieht dabei wie folgt aus. In der ersten Zeile stehen die Anzahl der Zeilen und Spalten der Matrix. Darunter sind die Einträge der Matrix zeilenweise aufgelistet. Beispielsweise

$$\begin{pmatrix} 4 & 2 \\ 5 & 7 \end{pmatrix} \quad \rightarrow \quad \begin{matrix} 2 & 2 \\ 4 & \\ 2 & \\ 5 & \\ 7 & \end{matrix}$$

Die Funktion zum Einlesen einer Matrix benötigt den Pfad zur Datei, aus der gelesen werden soll, und übermittelt die Dimension der Matrix mittels `Call by Reference`. Bei erfolgreichem Einlesen gibt die Funktion die Matrix in Form eines Zeigers zurück.

```c
double *matrix_read(char *path,   // Pfad der Datei
                    int *m,       // Anzahl der Zeilen
                    int *n,       // Anzahl der Spalten
);
```

Für das Speichern einer Matrix geht man ähnlich vor, wobei nun die Matrix gegeben ist.

```c
void matrix_write(char *path,   // Pfad der Datei
                  double *A,    // Matrix
                  int m,        // Anzahl der Zeilen
                  int n,        // Anzahl der Spalten
);
```

Für die Matrixmultiplikation werden zwei Matrizen mitsamt Dimensionen als Argumente übergeben. Achten Sie darauf, dass die Anzahl der Zeilen von A und die Anzahl der Spalten von B übereinstimmen.

```c
double *matrix_mult(double *A,   // Matrix A
                    int m,       // Anzahl der Zeilen von A
                    int n,       // Anzahl der Spalten von A
                    double *B,   // Matrix B
                    int k,       // Anzahl der Zeilen von B
                    int l        // Anzahl der Spalten von B
);
```

Hinweis: In den Funktionen `matrix_read()` und `matrix_mult()` muss jeweils zuerst Speicher für die Matrizen alloziert werden. Dies geschieht mit dem Befehl `calloc(#Einträge, sizeof(double))`. Am Ende Ihres Hauptprogramms müssen die Matrizen mittels `free()` freigegeben werden.

Aufgabe 0.4. (0 Punkte)
Die GNU Scientific Library (GSL) ist eine quelloffene C-Programmbibliothek, die eine Vielzahl von Funktionen für numerische Berechnungen bereitstellt. Informationen zur Installation und die Dokumentation finden Sie unter

https://www.gnu.org/software/gsl/

Wiederholen Sie nun Aufgabe 0.3. unter Zuhilfenahme der GSL.
Hinweis: Um die GSL einsetzen zu können, müssen Sie die benötigten Header inkludieren. Für diese Aufgabe genügt es, gsl/gsl_matrix.h und gsl/gsl_blas.h einzubinden. Darüber hinaus müssen Sie das Linken anpassen:

```
gcc ihr_programm.c -lgsl -lgslcblas -lm
```

Matrizen werden in der GSL durch die gsl_matrix Struktur dargestellt.

```c
typedef struct
{
  size_t size1;      // Anzahl der Zeilen
  size_t size2;      // Anzahl der Spalten
  size_t tda;
  double *data;      // Feld mit Eintraegen der Matrix
  gsl_block *block;
  int owner;
} gsl_matrix;
```

Um eine Matrix zu allozieren, verwenden Sie die Funktion

```c
gsl_matrix *gsl_matrix_calloc (size_t n1,     // Anzahl der Zeilen
                               size_t n2);    // Anzahl der Spalten
```

und entsprechend

```c
void gsl_matrix_free (gsl_matrix *m);
```

um den Speicher wieder freizugeben.
Benutzen Sie außerdem die Routinen

```c
int gsl_matrix_fprintf (FILE * stream, const gsl_matrix *m,
                        const char *format);
```

zum Ausgeben der Matrix und

```c
int gsl_matrix_fscanf (FILE * stream, const gsl_matrix *m);
```

zum Einlesen der Matrix. Wählen Sie hierbei %f als Formatstring. Für die Matrixmultiplikation bietet es sich an, die BLAS-Funktion gsl_blas_dgemm() zu verwenden. Um das Matrixprodukt AB in C abzuspeichern, rufen Sie

```c
gsl_blas_dgemm (CblasNoTrans, CblasNoTrans, 1.0, A, B, 0.0, C);
```

auf, wobei hier A,B und C jeweils Zeiger auf gsl_matrix sind.