**Familienname:**

**Vorname:**

**Matrikelnummer:**

| | |
|---|---|
| Aufgabe 1 (2 Punkte): | |
| Aufgabe 2 (3 Punkte): | |
| Aufgabe 3 (2 Punkte): | |
| Aufgabe 4 (3 Punkte): | |
| Aufgabe 5 (2 Punkte): | |
| Aufgabe 6 (1 Punkte): | |
| Aufgabe 7 (2 Punkte): | |
| Aufgabe 8 (7 Punkte): | |
| Aufgabe 9 (3 Punkte): | |
| Aufgabe 10 (5 Punkte): | |
| Gesamtpunktzahl: | |

# Schriftlicher Test zu C (90 Minuten)
# VU Einführung ins Programmieren für TM

## 1. Oktober 2012

**Aufgabe 1 (2 Punkte).** Was ist der Unterschied und der Zusammenhang zwischen einer Variable und einem Pointer?

**Lösung zu Aufgabe 1.**

**Aufgabe 2 (3 Punkte).** Schreiben Sie eine Funktion `swap`, die die Werte zweier Variablen $a$ und $b$ vertauscht. Warum funktioniert das folgende Vorgehen nicht? Wie würde man es richtig implementieren?

```
void swap(double a, double b) {
  double tmp = a;
  a = b;
  b = tmp;
}
```

**Lösung zu Aufgabe 2.**

**Aufgabe 3 (2 Punkte).** Schreiben Sie einen Struktur-Datentyp `Matrix` zur Speicherung quadratischer Matrizen $A \in \mathbb{R}^{n\times n}$ beliebiger Dimension $n \in \mathbb{N}$. In der Struktur sollen neben der Dimension $n$ die Koeffizienten $A_{jk}$ vom Typ `double` gespeichert werden, wobei die Matrix $A$ spaltenweise im Speicher abgelegt werde. Eine $(3 \times 3)$-Matrix

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$$

werde also beispielsweise in einem Vektor `entries` mit

$$\texttt{entries} = (a_{00}, a_{10}, a_{20}, a_{01}, a_{11}, a_{21}, a_{02}, a_{12}, a_{22})$$

gespeichert. **Diese Struktur soll auch in allen nachfolgenden Aufgaben verwendet werden.**

**Lösung zu Aufgabe 3.**

**Aufgabe 4 (3 Punkte).** Schreiben Sie eine Funktion `newMatrix`, die für gegebenes $n \in \mathbb{N}$ eine Matrix $A \in \mathbb{R}^{n\times n}$ allokiert und initialisiert.

**Lösung zu Aufgabe 4.**

**Aufgabe 5 (2 Punkte).** Schreiben Sie eine Funktion `delMatrix`, die den Speicher einer Matrix $A \in \mathbb{R}^{n \times n}$ freigibt und den `NULL`-Pointer zurückgibt.

**Lösung zu Aufgabe 5.**

**Aufgabe 6 (1 Punkt).** Schreiben Sie eine Funktion `getMatrixDim`, die die Dimension $n \in \mathbb{N}$ einer Matrix $A \in \mathbb{R}^{n \times n}$ zurückgibt.

**Lösung zu Aufgabe 6.**

**Aufgabe 7 (2 Punkte).** Schreiben Sie eine Funktion `getMatrixEntry`, die für gegebene Indizes $j, k$ den Koeffizienten $A_{jk}$ von $A \in \mathbb{R}^{n \times n}$ zurückgibt. Begründen Sie Ihre Index-Formel (spaltenweise Speicherung!).

**Lösung zu Aufgabe 7.**

Gegeben sei eine Matrix $A \in \mathbb{R}^{n\times n}$ mit ihrer normalisierten LU-Zerlegung $A = LU$, wobei $L \in \mathbb{R}^{n\times n}$ eine normalisierte untere Dreiecksmatrix und $U \in \mathbb{R}^{n\times n}$ eine obere Dreiecksmatrix mit $\prod_{j=1}^{n} u_{jj} \neq 0$ seien, d.h. es gilt

$$\begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,n-1} \\ a_{10} & a_{11} & \dots & a_{1,n-1} \\ \vdots & \vdots & & \vdots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,n-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \ell_{10} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n-1,0} & \dots & \ell_{n-1,n-2} & 1 \end{pmatrix} \begin{pmatrix} u_{00} & u_{01} & \dots & u_{0,n-1} \\ 0 & u_{11} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-2,n-1} \\ 0 & \dots & 0 & u_{n-1,n-1} \end{pmatrix}.$$

Eine solche Faktorisierung ist brauchbar, wenn man für viele $b \in \mathbb{R}^n$ das Gleichungssystem $Ax = b$ lösen muss.

**Aufgabe 8 (7 Punkte).** Nicht jede reguläre Matrix $A \in \mathbb{R}^{n\times n}$ besitzt eine normalisierte LU-Zerlegung. Wenn aber $A = LU$ eine normalisierte LU-Zerlegung ist, so gilt

$$\begin{aligned} u_{ik} &= a_{ik} - \sum_{j=0}^{i-1} \ell_{ij} u_{jk} \quad \text{für } i = 0, \dots, n-1, \quad k = i, \dots, n-1, \\ \ell_{ki} &= \frac{1}{u_{ii}} \Big( a_{ki} - \sum_{j=0}^{i-1} \ell_{kj} u_{ji} \Big) \quad \text{für } i = 0, \dots, n-1, \quad k = i+1, \dots, n-1, \\ \ell_{ii} &= 1 \quad \text{für } i = 0, \dots, n-1, \end{aligned}$$

wie man leicht über die Formel für die Matrix-Matrix-Multiplikation zeigen kann. Alle übrigen Einträge von $L, U \in \mathbb{R}^{n\times n}$ sind Null. Man schreibe eine Funktion `computeLU`, die $(n \times n)$-Matrizen $A$, $L$ und $U$ übernimmt und die Matrizen $L$ und $U$ mit den Werten der LU-Zerlegung überschreibt. Sie dürfen davon ausgehen, dass $L$ und $U$ als $(n\times n)$-Matrizen mit Nulleinträgen initialisiert sind.

**ACHTUNG: Überlegen Sie sich zunächst, wie viele geschachtelte Schleifen Sie brauchen und in welcher Reihenfolge diese durchlaufen werden müssen, damit alle auftretenden Produkte $\ell_{ij} u_{jk}$ bzw. $\ell_{kj} u_{ji}$ bereits berechnet worden sind!**

**Lösung zu Aufgabe 8.**

**Aufgabe 9 (3 Punkte).** Was versteht man unter *Aufwand*? Was besagt die Landau-Notation $\mathcal{O}(N^\alpha)$? Welchen Aufwand hat die Berechnung der LU-Zerlegung in Landau-Notation? Begründen Sie Ihre Antwort.

**Lösung zu Aufgabe 9.**

**Aufgabe 10 (5 Punkte).** Was tut die folgende Funktion `func` bei Übergabe der Matrix?

$$A = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 3 \\ 1 & 2 & 0 & 2 \\ 17 & 4 & 4 & 1 \end{pmatrix} \tag{1}$$

Geben Sie tabellarisch wider, welchen Wert die Variablen zum angegebenen Zeitpunkt haben. Erweitern Sie hierfür die untenstehende Tabelle geeignet. Welche Funktionalität wird durch `func` bereitgestellt? Was ist an dieser Lösung ineffizient realisiert, und wie könnte man das effizienter gestalten?

```
int func(matrix* mat) {
  double foo = 0;
  int mp, dp, tf;
  mp = 1;
  for (dp = 0; dp < getMatrixDim(mat); ++dp) {
    for (tf = dp+1; tf < getMatrixDim(mat); ++tf) {
      foo = getMatrixEntry(mat,dp,tf);
      if ( foo != 0 ) {
        mp = 0;
      }
      /* WERT DER VARIABLEN ZU DIESEM ZEITPUNKT */
    }
  }
  return mp;
}
```

| mp | foo | dp | tf |
|---|---|---|---|
| | | | |