Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (2 Punkte):
Aufgabe 2 (2 Punkte):
Aufgabe 3 (1 Punkt):
Aufgabe 4 (4 Punkte):
Aufgabe 5 (1 Punkt):
Aufgabe 6 (1 Punkt):
Aufgabe 7 (2 Punkte):
Aufgabe 8 (4 Punkte):
Aufgabe 9 (4 Punkte):
Aufgabe 10 (2 Punkte):
Aufgabe 11 (5 Punkte):
Aufgabe 12 (2 Punkte):
Aufgabe 13 (5 Punkte):
Aufgabe 14 (2 Punkte):
Aufgabe 15 (3 Punkte):

Gesamtpunkte (40 Punkte):

Schriftlicher Test (120 Minuten)
VU Einführung ins Programmieren für TM

30. Juni 2017

Aufgabe 1 (2 Punkte). Was ist eine rekursive Funktion und was darf dabei nicht fehlen? Erläutern Sie das Konzept anhand eines selbstgewählten Beispiels und geben Sie einen entsprechenden C/C++ Code an!

Lösung zu Aufgabe 1.

Aufgabe 2 (2 Punkte). Eine untere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ ist eine Matrix mit der Eigenschaft $A_{jk} = 0$ für $j < k$, d.h.

$$A = \begin{pmatrix} A_{00} & 0 & 0 & \dots & 0 \\ A_{10} & A_{11} & 0 & \dots & 0 \\ A_{20} & A_{12} & A_{22} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ A_{n-1,0} & \cdots & \cdots & A_{n-1,n-2} & A_{n-1,n-1} \end{pmatrix}$$

Zur effizienten Speicherung wird $A \in \mathbb{R}^{n\times n}$ in Form eines Vektors $a \in \mathbb{R}^N$ mit $N = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}$ abgelegt, d.h. $A_{jk} = a_\ell$ für einen geeigneten Index ℓ, der eindeutig von j und k abhängen muss. Leiten Sie eine Formel für ℓ her (in Abhängigkeit von $j, k \in \{0, \dots, n-1\}$). Begründen Sie Ihre Formel.

Hinweis. Am einfachsten ist zeilenweise Speicherung der Einträge A_{jk} für $j \geq k$, d.h.

$$a = (A_{00}, A_{10}, A_{11}, A_{20}, A_{21}, A_{22}, \dots, A_{n-1,n-1}) \in \mathbb{R}^N.$$

Lösung zu Aufgabe 2.

Hinweis. In den folgenden Aufgaben seien die unteren Dreiecksmatrizen $A \in \mathbb{R}^{n \times n}$ in Objekten der C++ Klasse `TriMatrix` gespeichert, die unten definiert ist. Neben Konstruktor (mit optionalem Initialisierungswert), Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es Methoden, um die Dimension n auszulesen (`size`) und die Zeilensummennorm zu berechnen (`norm`) . Der Koeffizientenvektor `coeff` speichert nur die $\frac{n(n+1)}{2}$ nicht-trivialen Einträge A_{jk} mit $j \geq k$, auf die mittels `A(j,k)` lesend und schreibend zugegriffen wird:

```
1 class TriMatrix {
2 private:
3   int n;
4   double* coeff;
5 public:
6   TriMatrix(int n=0, double init=0);
7   TriMatrix(const TriMatrix&);
8   ~TriMatrix();
9   TriMatrix& operator=(const TriMatrix&);
10  int size() const;
11  const double& operator()(int j, int k) const;
12  double& operator()(int j, int k);
13  double norm() const;
14 };
```

Aufgabe 3 (1 Punkt). Erläutern Sie die Bedeutung der beiden `const` in Zeile 11 der Klassendefinition.

Lösung zu Aufgabe 3.

Aufgabe 4 (4 Punkte). Schreiben Sie den Konstruktor der Klasse `TriMatrix`. Stellen Sie mittels `assert` sicher, dass $n \geq 0$ ist, wobei für $n = 0$ eine leere Matrix angelegt werde.

Hinweis. Beachten Sie, dass `coeff` ein Vektor der Länge $\frac{n(n+1)}{2}$ ist.

Lösung zu Aufgabe 4.

Aufgabe 5 (1 Punkt). Schreiben Sie den Destruktor der Klasse `TriMatrix`.

Lösung zu Aufgabe 5.

Aufgabe 6 (1 Punkt). Schreiben Sie die Methode `size` der Klasse `TriMatrix`.

Lösung zu Aufgabe 6.

Aufgabe 7 (2 Punkte). Schreiben Sie den Koeffizientenzugriff der Klasse `TriMatrix` für `const`-Objekte. Stellen Sie mittels `assert` sicher, dass für $A \in \mathbb{R}^{n \times n}$ die Indizes $0 \leq k \leq j \leq n-1$ erfüllen.

Hinweis. Verwenden Sie Ihre Formel aus Aufgabe 2.

Lösung zu Aufgabe 7.

Aufgabe 8 (4 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `TriMatrix`.

Lösung zu Aufgabe 8.

Aufgabe 9 (4 Punkte). Implementieren Sie die Methode `norm` der Klasse `TriMatrix`, die von einer unteren Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ die Zeilensummenorm

$$\|A\| := \max_{j=0,\ldots,n-1} \sum_{k=0}^{n-1} |A_{jk}|$$

berechnet und zurückgibt.

Hinweis. Den Absolutbetrag eines `double`-Wertes liefert die Funktion `fabs`. Beachten Sie, dass die Klasse `TriMatrix` nur den Zugriff auf Einträge A_{jk} mit $j \geq k$ erlaubt!

Lösung zu Aufgabe 9.

Aufgabe 10 (2 Punkte). Beweisen Sie mathematisch, dass das Produkt $C = AB \in \mathbb{R}^{n\times n}$ zweier unterer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ wieder eine untere Dreiecksmatrix ist, indem Sie die Laufindizes der Summe des allgemeinen Matrizenprodukts

$$C_{j\ell} = \sum_{k=0}^{n-1} A_{jk} B_{k\ell} \quad \text{für } j, \ell = 0, \ldots, n-1$$

mithilfe der Dreiecksstruktur von A und B vereinfachen.

Hinweis. Eine untere Dreiecksmatrix $C \in \mathbb{R}^{n\times n}$ ist durch $C_{j\ell} = 0$ für $j < \ell$ charakterisiert.

Lösung zu Aufgabe 10.

Aufgabe 11 (5 Punkte). Überladen Sie den `*` Operator so, dass er das Produkt $C = AB \in \mathbb{R}^{n\times n}$ zweier unterer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ berechnet. Stellen Sie mittels `assert` sicher, dass A und B dieselbe Dimension haben.

Hinweis. Beachten Sie, dass die Funktion nur Koeffizienten C_{jk} für $0 \leq k \leq j \leq n-1$ berechnen soll und auch nur auf entsprechende Koeffizienten von A und B zugreifen darf. Verwenden Sie dazu Ihre Erkenntnisse aus Aufgabe 10.

Lösung zu Aufgabe 11.

Hinweis. In den folgenden Aufgaben seien Vektoren $x \in \mathbb{R}^n$ in Objekten der C++ Klasse `Vector` gespeichert, die unten definiert ist. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es eine Methode, um die Dimension n auszulesen (`size`). Auf die Koeffizienten x_j des Vektors kann mittels `x(j)` für $0 \leq j \leq n-1$ zugegriffen werden. Sie müssen keine der genannten Methoden implementieren!

```
class Vector {
private:
  int n;
  double* coeff;
public:
  Vector(int n=0, double init=0);
  Vector(const Vector&);
  ~Vector();
  Vector& operator=(const Vector&);
  int size() const;
  const double& operator()(int j) const;
  double& operator()(int j);
};
```

Aufgabe 12 (2 Punkte). Leiten Sie für gegebenes $b \in \mathbb{R}^n$ eine Formel her, um für eine untere Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ mit $A_{jj} \neq 0$ für alle $j = 0, \ldots, n-1$ die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ zu berechnen, indem Sie die Formel des Matrix-Vektor-Produkts

$$b_j = (Ax)_j = \sum_{k=0}^{n-1} A_{jk} x_k$$

mithilfe der Dreiecksstruktur von A vereinfachen.

Hinweis. Eine untere Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ ist durch $A_{jk} = 0$ für $j < k$ charakterisiert.

Lösung zu Aufgabe 12.

Aufgabe 13 (5 Punkte). Überladen Sie den `|` Operator so, dass `x = A|b` für eine untere Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ (vom Typ `TriMatrix`) und einen Vektor $b \in \mathbb{R}^n$ (vom Typ `Vector`) die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ (als Objekt vom Typ `Vector`) berechnet. Stellen Sie mittels `assert` sicher, dass A und b passende Dimension haben und dass $A_{jj} \neq 0$ für alle $j = 0, \ldots, n-1$.

Hinweis. Verwenden Sie Ihre Formel aus Aufgabe 12. Beachten Sie, dass die Klasse `TriMatrix` nur den Zugriff auf Einträge A_{jk} mit $j \geq k$ erlaubt!

Lösung zu Aufgabe 13.

Aufgabe 14 (2 Punkte). Bestimmen Sie den Aufwand Ihrer Funktion aus Aufgabe 13. Falls die Funktion für $n = 10^3$ eine Laufzeit von 2 Sekunden hat, welche Laufzeit erwarten Sie aufgrund des Aufwands für $n = 5 \cdot 10^3$? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 14.

Aufgabe 15 (3 Punkte). Was ist der Shell-Output des folgenden Programms?

```
#include <iostream>
using std::cout;
using std::endl;
using std::string;

class Father {
private:
  string name;
public:
  Father() { name = "nobody"; cout << "1" << endl; }
  Father(string name) { this->name = name; cout << "2: " << name << endl; }
  ~Father() { cout << "3: " << name << endl; }
  Father(const Father& foo) { name = foo.name; cout << "4: " << name << endl; }
};

class Son : public Father {
public:
  Son() : Father("Klaus") { cout << "5" << endl; }
  Son(string name) : Father(name) { cout << "6" << endl; }
  ~Son() { cout << "7" << endl; }
  Son(const Son& foo) { cout << "8" << endl; }
  Son(const Father& bar) : Father(bar) { cout << "9" << endl; }
};

int main() {
  Father foo("Hans");
  Son bar = foo;
  Son* son = new Son(foo);
  Father* father = new Father("Fritz");
  delete son;
  delete father;
  return 0;
}
```

Lösung zu Aufgabe 15.