Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunkte (40 Punkte):

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

29. Juni 2018

Aufgabe 1 (3 Punkte). Was versteht man unter *Call-by-Value*? Was versteht man unter *Call-by-Reference*? Was ist der Unterschied in der Praxis? Gibt es beides in C++?

Lösung zu Aufgabe 1.

Aufgabe 2 (3 Punkte). Was sind die Bestandteile M, $e_{\min}$, $e_{\max}$ des Gleitkommazahlsystems $\mathbb{F}(2, M, e_{\min}, e_{\max})$? Wie lässt sich jede Gleitkommazahl $x \in \mathbb{F}(2, M, e_{\min}, e_{\max})$ darstellen? Welchen Wert haben die größte und die kleinste positive normalisierte Gleitkommazahl im `double`-Gleitkommazahlsystem $\mathbb{F}(2, 53, -1021, 1024)$?

Lösung zu Aufgabe 2.

Hinweis. In den folgenden Aufgaben betrachten wir Matrizen $A \in \mathbb{R}^{n \times n}$ als Objekte der C++ Klasse `Matrix`, die unten definiert ist. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator, gibt es eine Methode, um die Dimension n auszulesen (`dim`) und um zu bestimmen, ob A eine untere Dreiecksmatrix ist (`isLowerTriangular`). Die Koeffizienten A_{jk} erhält man mittels `A(j,k)`. Intern wird eine Matrix $A \in \mathbb{R}^{n \times n}$ spaltenweise in einem dynamischen Vektor der Länge n^2 gespeichert.

```
1 class Matrix {
2 private:
3   int n;
4   double* entry;
5
6 public:
7   Matrix(int n = 0);
8   ~Matrix();
9   Matrix(const Matrix&);
10  const Matrix& operator=(const Matrix&);
11
12  int dim() const;
13  double& operator()(int j, int k);
14  const double& operator()(int j, int k) const;
15
16  bool isLowerTriangular() const;
17 };
```

Aufgabe 3 (2 Punkte). Matrizen sollen intern spaltenweise gespeichert werden, d.h. die Matrix $A \in \mathbb{R}^{n\times n}$ liegt als Vektor

$$a = (A_{00}, A_{10}, A_{20}, \ldots, A_{n-1,0}, A_{01}, A_{11}, \ldots, A_{n-1,1}, \ldots, A_{0,n-1}, \ldots, A_{n-1,n-1}) \in \mathbb{R}^{n^2} \tag{1}$$

im Speicher. Leiten Sie eine Formel her, die jedem Indexpaar (j, k) mit $j, k \in \{0, \ldots, n-1\}$ den eindeutigen Index $\ell \in \{0, \ldots, n^2 - 1\}$ zuordnet, sodass $a_\ell = A_{jk}$ gilt. Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 3.

Aufgabe 4 (3 Punkte). Schreiben Sie den Konstruktor der Klasse `Matrix`, der für $n \in \mathbb{N}_0$ eine Matrix $A \in \mathbb{R}^{n \times n}$ anlegt und die Koeffizienten mit Null initialisiert. Für $n = 0$ soll kein zusätzlicher Speicher angelegt werden. Stellen Sie mittels `assert` sicher, dass $n \geq 0$ gilt.

Lösung zu Aufgabe 4.

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

Lösung zu Aufgabe 5.

Aufgabe 6 (4 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `Matrix`.

Lösung zu Aufgabe 6.

Aufgabe 7 (1 Punkt). Implementieren Sie den Operator `()` der Klasse `Matrix` für `const`-Objekte um auf Koeffizienten A_{jk} der Matrix mittels `A(j,k)` zuzugreifen. Stellen Sie mittels `assert` sicher, dass für $A \in \mathbb{R}^{n \times n}$ die Indizes $j, k \in \{0, \ldots, n-1\}$ erfüllen.

Hinweis. Verwenden Sie Ihre Formel aus Aufgabe 3.

Lösung zu Aufgabe 7.

Aufgabe 8 (3 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.

$$A = \begin{pmatrix} A_{00} & 0 & 0 & \dots & 0 \\ A_{10} & A_{11} & 0 & \dots & 0 \\ A_{20} & A_{21} & A_{22} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ A_{n-1,0} & \cdots & \cdots & A_{n-1,n-2} & A_{n-1,n-1} \end{pmatrix}$$

Schreiben Sie die Methode `isLowerTriangular` der Klasse `Matrix`. Falls $A \in \mathbb{R}^{n \times n}$ eine untere Dreiecksmatrix ist, soll die Methode `true` zurückgeben, anderenfalls `false`.

Lösung zu Aufgabe 8.

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 (der den Vektor als Nullvektor anlegt), Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es eine Methode, um die Dimension n auszulesen (`dim`). 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* entry;

public:
  Vector(int n = 0);
  ~Vector();
  Vector(const Vector&);
  const Vector& operator=(const Vector&);

  int dim() const;
  double& operator()(int j);
  const double& operator()(int j) const;
};
```

Aufgabe 9 (3 Punkte). Überladen Sie den Operator `*` so, dass er für eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Vektor $x \in \mathbb{R}^n$ das Matrix-Vektor-Produkt $b = A * x \in \mathbb{R}^n$ berechnet, d.h.

$$b_j = \sum_{k=0}^{n-1} A_{jk} x_k \quad \text{für alle } j = 0, \ldots, n-1.$$

Stellen Sie mittels `assert` sicher, dass A und x dieselbe Dimension haben. Verwenden Sie die Signatur

```
const Vector operator*(const Matrix& A, const Vector& x);
```

Lösung zu Aufgabe 9.

Aufgabe 10 (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

$$b_j = (Ax)_j = \sum_{k=0}^{n-1} A_{jk} x_k \quad \text{für alle } j = 0, \ldots, n-1$$

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 10.

Aufgabe 11 (5 Punkte). Überladen Sie den `|` Operator so, dass `x = A|b` für eine untere Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ (vom Typ `Matrix`) 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 eine untere Dreiecksmatrix ist, dass A und b passende Dimension haben und dass $A_{jj} \neq 0$ für alle $j = 0, \ldots, n-1$. Verwenden Sie die Signatur

```
const Vector operator|(const Matrix& A, const Vector& b);
```

Hinweis. Verwenden Sie Ihre Formel aus Aufgabe 10.

Lösung zu Aufgabe 11.

Aufgabe 12 (2 Punkte). Berechnen Sie den Aufwand Ihrer Funktion aus Aufgabe 11. Falls die Funktion für $n = 10^3$ eine Laufzeit von 0.1 Sekunden hat, welche Laufzeit erwarten Sie aufgrund des Aufwands für $n = 5 \cdot 10^3$? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 12.

Aufgabe 13 (4 Punkte). Schreiben Sie eine Funktion `pascal`, die für gegebenes $n \in \mathbb{N}$ eine untere Dreiecksmatrix $A \in \mathbb{R}^{n \times n}$ erzeugt, deren Zeilen die Stufen des Pascal'schen Dreiecks sind: Jede Zeile dieses Schemas beginnt und endet mit 1. Die restlichen Zahlen werden jeweils als Summe der beiden nächstgelegenen Zahlen der Zeile darüber gebildet. Für $n = 5$ gilt beispielsweise

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array} \qquad \text{bzw. als Dreiecksmatrix} \qquad A = \begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} \in \mathbb{R}^{5 \times 5}$$

Stellen Sie mittels `assert` sicher, dass $n \geq 1$ gilt.

Lösung zu Aufgabe 13.

Aufgabe 14 (4 Punkte). Was ist der Shell-Output des folgenden Programms?

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

class Base{
protected:
  double x;
public:
  Base(double x = 0) {
    this->x = x;
    cout << "Base, Constructor, x = " << x <<  endl;
  }
  Base(const Base& input) {
    this->x = input.x;
    cout << "Base, Copy Constructor, x = " << x << endl;
  }
  void printData() const {
    cout << "Base, printData, x = " << x << endl;
  }
  virtual void printClass() const {
    cout << "I am of class Base!" << endl;
  }
};
class Derived:public Base {
protected:
  double y;
public:
  Derived():Base(0) {
    this-> y = 0;
    cout << "Derived, Standard Constructor" << endl;
  }
  Derived(double y) {
    this-> x = 1;
    this-> y = y;
    cout << "Derived, Constructor, x = " << x << ", y = " << y << endl;
  }
  Derived(const Derived& input) {
    this-> x = input.x;
    this-> y = input.y;
    cout << "Derived, Copy Constructor, x = " << x << ", y = " << y << endl;
  }
  void printData() const {
    cout << "Derived, printData, x = " << x << ", y = " << y << endl;
  }
  virtual void printClass() const {
    cout << "I am of class Derived!" << endl;
  }
};
```

```
int main(){
  Derived fs(2);
  Derived cmp(fs);
  Base* dp = &fs;
  dp->printClass();
  dp->printData();
  fs.printClass();
  fs.printData();
  return 0;
}
```

Lösung zu Aufgabe 14.