
Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunktzahl:

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

26. Juni 2015

Aufgabe 1 (2 Punkte). Schreiben Sie einen C-Struktur-Datentyp `CVector` zur Speicherung von Vektoren $x \in \mathbb{C}^n$. In der Struktur sollen neben der Länge $n \in \mathbb{N}$ die komplexen Zahlen $x_j \in \mathbb{C}$ in Form einer dynamischen `double**` Matrix der Dimension $n \times 2$ gespeichert werden, d.h. $\text{Re}(x_j) = x[j][0]$ und $\text{Im}(x_j) = x[j][1]$.

Hinweis. Verwenden Sie die Struktur `CVector` auch in den Aufgaben 2–6.

Lösung zu Aufgabe 1.

Aufgabe 2 (3 Punkte). Schreiben Sie eine C-Funktion `newCVector`, die einen Vektor $x \in \mathbb{C}^n$ allokiert und mit Null initialisiert.

Lösung zu Aufgabe 2.

Aufgabe 3 (1 Punkt). Schreiben Sie eine C-Funktion `getCVectorLength`, die die Länge n eines Vektors $x \in \mathbb{C}^n$ zurückgibt

Lösung zu Aufgabe 3.

Aufgabe 4 (1 Punkt). Schreiben Sie eine C-Funktion `getCVectorCoeff`, die den Pointer `z` auf einen Koeffizienten $x_j \in \mathbb{C}$ zurückgibt, d.h. `z[0]` ist der Realteil von x_j und `z[1]` ist der Imaginärteil von x_j . Stellen Sie mittels `assert` sicher, dass der Index j zulässig ist.

Lösung zu Aufgabe 4.

Aufgabe 5 (6 Punkte). Schreiben Sie eine C-Funktion `sort`, die einen Vektor $x \in \mathbb{C}^n$ nach den folgenden Kriterien aufsteigend sortiert und durch den sortierten Vektor überschreibt: Es gilt $x_j < x_{j+1}$, falls

- $|x_j| < |x_{j+1}|$,
- $|x_j| = |x_{j+1}|$ und $\operatorname{Re}(x_j) < \operatorname{Re}(x_{j+1})$,
- $|x_j| = |x_{j+1}|$ und $\operatorname{Re}(x_j) = \operatorname{Re}(x_{j+1})$ und $\operatorname{Im}(x_j) < \operatorname{Im}(x_{j+1})$.

Sie können jeden Sortieralgorithmus nehmen, den Sie kennen, z.B. Bubble-Sort, Merge-Sort oder Selection-Sort (= MinSort).

Hinweis. Sie dürfen voraussetzen, dass es eine Funktion `cabs` gibt, die den Absolutbetrag einer komplexen Zahl berechnet, d.h. `cabs(getCVectorCoeff(x, j))` liefert $|x_j|$.

Lösung zu Aufgabe 5.

Aufgabe 6 (2 Punkte). Welchen Aufwand hat Ihre Funktion `sort` aus Aufgabe 5? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 6.

Aufgabe 7 (5 Punkte). Schreiben Sie eine C++ Klasse `Bruch` zur Darstellung eines Bruchs $x = p/q$, wobei $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ als `int` gespeichert werden. Daneben soll die Klasse die folgenden Methoden bereitstellen:

- Standardkonstruktor (ohne Parameter), der $p = 0$ und $q_0 = 1$ setzt.
- Konstruktor, der $p, q \in \mathbb{Z}$ mit $q \neq 0$ als Input übernimmt und den Bruch speichert.
- Vorzeichenoperator, der zu x den Bruch $-x$ liefert.
- Zugriffsmethoden `setZaehler`, `getZaehler` für den Zähler.
- Zugriffsmethoden `setNenner`, `getNenner` für den Nenner.
- Methode `kuerzen`, die p und q durch die gekürzte Darstellung $p/q = p_0/q_0$ ersetzt.
- Type Casting von `Bruch` auf `double`.

Schreiben Sie an dieser Stelle nur die Klassendefinition. Es ist hier keine Funktionalität zu implementieren.

Hinweis. Verwenden Sie die Klasse `Bruch` in den Aufgaben 8–12.

Lösung zu Aufgabe 7.

Aufgabe 8 (3 Punkte). Schreiben Sie die Konstruktoren der Klasse `Bruch`. Stellen Sie mittels `assert` sicher, dass die Übergabeparameter zulässig sind, d.h. $q \neq 0$. Beachten Sie den Fall $q < 0$, bei dem intern $(-p)/|q|$ gespeichert wird.

Lösung zu Aufgabe 8.

Aufgabe 9 (1 Punkt). Schreiben Sie das Type Casting von Bruch auf double.

Hinweis: Vorsicht mit der Integerdivision!

Lösung zu Aufgabe 9.

Aufgabe 10 (5 Punkte). Schreiben Sie die Methode `kuerzen` der Klasse `Bruch`. Dabei sollen p und q durch $p_0 \in \mathbb{Z}$ und $q_0 \in \mathbb{N}$ überschrieben werden, wobei $p = gp_0$ und $q = gq_0$, mit $g \in \mathbb{N}$ maximal. Gehen Sie dazu wie folgt vor:

- Für $p = 0$ gilt $p_0 = 0$ und $q = 1$.
- Für $p \neq 0$ ist g der größte gemeinsame Teiler von $|p|$ und q . Diesen können Sie mit dem Euklid-Algorithmus bestimmen. Für $a, b \in \mathbb{N}$ funktioniert dieser Algorithmus wie folgt:
 - (i) Im Fall $a = b$, ist der größte gemeinsame Teiler klar.
 - (ii) Anderenfalls garantiere $a < b$ durch Vertauschen und ersetze b durch $b - a$.
 - (iii) Wiederhole die beiden Schritte (i)–(ii), bis $a = b$ gilt.

Lösung zu Aufgabe 10.

Aufgabe 11 (3 Punkte). Überladen Sie den Operator $*$, um das Produkt $x \cdot y$ zweier Brüche zu berechnen. Das Ergebnis soll die gekürzte Form des Produkt-Bruches sein.

Hinweis. Sie dürfen die `set-` und `get-`Methoden verwenden, ohne diese zu implementieren.

Lösung zu Aufgabe 11.

Aufgabe 12 (3 Punkte). Überladen Sie den Operator $+$, um die Summe $x + y$ zweier Brüche zu berechnen. Das Ergebnis soll die gekürzte Form des Summen-Bruches sein.

Hinweis. Sie dürfen die `set-` und `get-`Methoden verwenden, ohne diese zu implementieren.

Lösung zu Aufgabe 12.

Aufgabe 13 (5 Punkte). Was ist der Output des folgenden Programms?

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

class Basisklasse {
protected:
    int N;
public:
    Basisklasse(int n = 0) {
        N = n;
        cout << "Konstr. Basisklasse, N = "<< N << endl;
    }
    virtual ~Basisklasse() {
        cout << "Destr. Basisklasse, N = "<< N << endl;
    }
    virtual void print() {
        klasse();
        cout << " N = "<< N << endl;
    }
    void Add() {
    }
    virtual void klasse() const {
        cout << "In Basisklasse aber virtual ,";
    }
    void klasse() {
        cout << "In Basisklasse,";
    }
};

class Abgeleitet : public Basisklasse {
public:
    Abgeleitet(int n = 0) {
        N = n;
        cout << "Konstr. Abgeleitet, N = "<< N << endl;
    }
    ~Abgeleitet() {
        cout << "Destr. Abgeleitet, N = "<< N << endl;
    }
    void print() const {
        klasse();
        cout << "const N ="<< N << endl;
    }
    void print() {
        klasse();
        cout << " N = "<< N << endl;
    }
    void Add(){
        N = N + 100;
    }
    void klasse() const {
        cout << "In Abgeleitet fuer const, ";
    }
    void klasse() {
        cout << "In Abgeleitet,";
    }
};

int main() {
    Basisklasse dp(1);
    Abgeleitet mr(10);
    Basisklasse* bs = &mr;
    {
        const Abgeleitet ah(200);
        dp.Add();
        mr.Add();
        bs->Add();
        ah.print();
    }
    dp.print();
    mr.print();
    bs->print();

    return 0;
}
```

Lösung zu Aufgabe 13.

