
Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunkte (40 Punkte):

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

30. September 2016

Aufgabe 1 (2 Punkte). Nennen Sie mindestens 2 Bedeutungen für das Schlüsselwort `const` in C++. Geben Sie jeweils ein kurzes Beispiel für die korrekte Verwendung!

Lösung zu Aufgabe 1.

```

1  class Fraction {
2  private:
3      int numerator;
4      int denominator;
5  public:
6      Fraction(int=0, int=1);
7      Fraction(const Fraction&);
8      Fraction& operator=(const Fraction&);
9      ~Fraction();
10     const Fraction operator-() const;
11     operator double() const;
12
13     void reduce();
14     int getNumerator() const;
15     int getDenominator() const;
16     void setNumerator(int);
17     void setDenominator(int);
18 };
19
20 const Fraction operator+(const Fraction&, const Fraction&);
21 const Fraction operator-(const Fraction&, const Fraction&);
22 const Fraction operator*(const Fraction&, const Fraction&);
23 const Fraction operator/(const Fraction&, const Fraction&);
24
25 std::ostream& operator<<(std::ostream&, const Fraction&);

```

Hinweis. In den folgenden Aufgaben sollen Sie Teile der Funktionalität der Klasse `Fraction` implementieren, die den elementaren Umgang mit Brüchen $x = p/q$ realisiert. Dabei soll der Zähler (engl. *numerator*) als ganze Zahl $p \in \mathbb{Z}$ mit Vorzeichen gespeichert werden, der Nenner (engl. *denominator*) als natürliche Zahl $q \in \mathbb{N}$ ohne Vorzeichen, d.h. $q > 0$. Die Klasse enthält folgende Methoden:

- Konstruktor (Zeile 6),
- Kopierkonstruktor (Zeile 7),
- Destruktor (Zeile 8),
- Zuweisungsoperator (Zeile 9),
- Vorzeichen (Zeile 10),
- Type Cast auf `double` (Zeile 11)
- Kürzen eines Bruches (Zeile 13, engl. *reduce*)
- Methoden für Zugriff auf Zähler und Nenner (Zeile 14–17).

Ferner stehen die arithmetischen Operationen (Zeile 20–23) sowie der Stream-Operator (Zeile 25) zur Verfügung.

Aufgabe 2 (3 Punkte). Schreiben Sie den Konstruktor der Klasse `Fraction`. Achten Sie darauf, dass $x = p/q$ mit $q > 0$ gelten soll. Der Fall $q = 0$ soll mittels `assert` abgefangen werden, grundsätzlich sei aber $q \neq 0$ (also mit Vorzeichen) als Input erlaubt. Der Bruch soll in der gekürzten Form abgespeichert werden, d.h. $x = p'/q'$ mit $q' > 0$ sowie $p' \in \mathbb{Z}$ und $q' \in \mathbb{N}$ teilerfremd. Verwenden Sie dazu ggf. die Methode `reduce`.

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `Fraction`.

Lösung zu Aufgabe 3.

Aufgabe 4 (2 Punkte). Schreiben Sie den Type Cast von `Fraction` auf `double`.

Lösung zu Aufgabe 4.

Aufgabe 5 (5 Punkte). Schreiben Sie die Methode `reduce` der Klasse `Fraction`, um einen Bruch $x = p/q$ mit $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ auf die gekürzte Form $x = p'/q'$ zu bringen (mit $p' \in \mathbb{Z}$ und $q' \in \mathbb{N}$ teilerfremd). Bestimmen Sie dazu zunächst mittels Euklid-Algorithmus den größten gemeinsamen Teiler $g \in \mathbb{N}$ von $|p| \in \mathbb{N}_0$ und $q \in \mathbb{N}$. Dann gilt $x = p'/q'$ mit $p' := p/g$ und $q' := q/g$, und p' und q' sind teilerfremd.

Hinweis. Den Absolutbetrag von Integers erhält man in C++ mittels der Funktion `std::abs`.

Euklid-Algorithmus. Für zwei natürliche Zahlen $a, b \in \mathbb{N}$ bestimmt der Algorithmus den größten gemeinsamen Teiler $g \in \mathbb{N}$. Dazu werden die folgenden Schritte iteriert, bis $a = 0$ gilt:

- Stelle durch Vertauschen sicher, dass $a \geq b$ gilt.
- Überschreibe a durch $a - b$.

Wenn der Algorithmus terminiert, ist $g = b$ der größte gemeinsame Teiler der anfangs gegebenen Zahlen.

Lösung zu Aufgabe 5.

Aufgabe 6 (1 Punkt). Schreiben Sie die Methode `getNumerator` der Klasse `Fraction`.

Lösung zu Aufgabe 6.

Aufgabe 7 (3 Punkte). Schreiben Sie die Addition für zwei Brüche. Überladen Sie dazu den `+` Operator. Das Ergebnis werde als Bruch in gekürzter Form zurückgegeben.

Hinweis. Sie dürfen alle Methoden der Klasse `Fraction` verwenden, auch wenn Sie diese nicht implementiert haben.

Lösung zu Aufgabe 7.

```
1 class Vector {
2 private:
3     int length;
4     Fraction* coeff;
5 public:
6     Vector(int n = 1);
7     ~Vector();
8     Vector(const Vector&);
9     Vector& operator=(const Vector&);
10
11     int getLength() const;
12     const Fraction& operator[](int j) const;
13     Fraction& operator[](int j);
14     void sort();
15 };
16
17 std::ostream& operator<<(std::ostream&, const Vector&);
```

Hinweis. In den folgenden Aufgaben sollen Sie Teile der Funktionalität der Klasse **Vector** implementieren, mit der man Vektoren von Brüchen betrachten kann. Die Klasse enthält folgende Methoden:

- Konstruktor (Zeile 6),
- Destruktor (Zeile 7)
- Kopierkonstruktor (Zeile 8),
- Zuweisungsoperator (Zeile 9),
- Rückgabe der Vektorlänge (Zeile 11)
- Zugriff auf Koeffizienten mittels [] (Zeile 12–13)
- Sortieren (Zeile 14).

Ferner steht der Stream-Operator (Zeile 17) zur Verfügung.

Aufgabe 8 (2 Punkte). Schreiben Sie den Konstruktor der Klasse `Vector`. Stellen Sie mittels `assert` sicher, dass die Vektorlänge ≥ 1 ist.

Lösung zu Aufgabe 8.

Aufgabe 9 (1 Punkt). Schreiben Sie den Destruktor der Klasse `Vector`.

Lösung zu Aufgabe 9.

Aufgabe 10 (4 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `Vector`.

Lösung zu Aufgabe 10.

Aufgabe 11 (2 Punkte). Schreiben Sie den Vektorzugriff mittels Operator `[]` der Klasse `Vector` für konstante Objekte. Stellen Sie mittels `assert` sicher, dass der gegebene Index j zulässig ist.

Lösung zu Aufgabe 11.

Aufgabe 12 (6 Punkte). Schreiben Sie die Methode `sort` der Klasse `Vector`, die den Vektor aufsteigend sortiert und mit dem sortierten Koeffizientenvektor überschreibt.

Beispiel. Der Vektor

$$x = \left(\frac{0}{1}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{2}{9} \right)$$

soll durch Aufruf von `sort` durch

$$x = \left(\frac{0}{1}, \frac{2}{9}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2} \right)$$

überschrieben werden.

Hinweis. Aufgrund des Type Cast von `Fraction` auf `double` steht Ihnen der „ganz normale“ Vergleichsoperator `<` bereits zur Verfügung. Sie dürfen einen beliebigen Sortieralgorithmus verwenden.

Lösung zu Aufgabe 12.

Aufgabe 13 (2 Punkte). Bestimmen Sie den Aufwand Ihrer Funktion aus Aufgabe 12 für einen Vektor x der Länge n . Falls die Funktion für $n = 10^3$ eine Laufzeit von 1,5 Sekunden hat, welche Laufzeit erwarten Sie für $n = 10^4$? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 13.

Aufgabe 14 (5 Punkte). Was ist der Output des folgenden Codes? Was ist die mathematische Bedeutung des Vektors x , der in der Klasse angelegt wird?

```
1  #include <iostream>
2  #include <cassert>
3  using std::cout;
4
5  class pn {
6  private:
7      int n;
8      int* x;
9  public:
10     pn(int n);
11     ~pn();
12     int check(int k);
13 };
14
15 pn::pn(int n) {
16     cout << "++ init\n";
17     this->n = n;
18     x = new int[n+1];
19     for (int j=0; j<=n; ++j) {
20         x[j] = j;
21     }
22     for (int j=2; j<=0.5*n; ++j) {
23         if ((x[j] != 0) && (j*j <= n)) {
24             for (int k=j*j; k<=n; k = k + j) {
25                 x[k] = 0;
26             }
27             cout << j << ": ";
28             for (int k=2; k<=n; ++k) {
29                 cout << x[k] << ", ";
30             }
31             cout << "\n";
32         }
33     }
34 }
35
36 pn::~~pn() {
37     cout << "++ free\n";
38     delete[] x;
39 }
40
41 int pn::check(int k) {
42     assert(2 <= k && k <= n);
43     if (x[k] > 0) {
44         return 1;
45     }
46     else {
47         return 0;
48     }
49 }
50
51 int main() {
52     pn dp(15);
53     cout << "14 -> " << dp.check(14) << "\n";
54 }
```

Lösung zu Aufgabe 14.

