Dirk Praetorius
Michele Ruggeri
Bernhard Stiftner

Sommersemester 2016
24.05.2016

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Was ist der Output des folgenden Programms? Erklären Sie warum! Was ist der Unterschied zwischen den verschiedenen verwendeten Varablentypen?

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

const int proc(int & input){input = input*2; return input;}
int proc(const int & input){ int output = input; return output;}

void swap(int& x, int& y){
int tmp;
tmp = x;
x = y;
y = tmp;
}

void swap(const int& x,const int& y){;}

int main() {

int var1 = 1;
int var2 = 2;
int var3 = proc(var1);
int var4 = proc(var2);
const int var5 = proc(var1);
const int var6 = proc(var2);
int var7 = proc(proc(var1));
int var8 = proc(proc(var2));
int& var9 = var1;
int& var10 = var2;
const int& var11 = proc(var1);
const int& var12 = proc(var2);

swap(var3,var4);
swap(var5,var6);
swap(var7,var8);
swap(var9,var10);
swap(var11,var12);

cout << "var1 = " << var1 << " var2 = " << var2 << endl;
cout << "var3 = " << var3 << " var4 = " << var4 << endl;
cout << "var5 = " << var5 << " var6 = " << var6 << endl;
cout << "var7 = " << var7 << " var8 = " << var8 << endl;
cout << "var9 = " << var9 << " var10 = " << var10 << endl;
cout << "var10 = " << var11 << " var11 = " << var12 << endl;
```

```
return 0;
}
```

Aufgabe 10.2. Für die Personalabteilung der Universität ist es sehr mühsam, Studenten immer nur einzeln hinzuzufügen oder aus dem System zu löschen. Überladen Sie die Methoden `graduate` und `newStudent` der Klasse `University` aus Aufgabe 8.7 so, dass die Anzahl der abschließenden bzw. neu hinzukommenden Studenten mit übergeben werden kann. Schreiben Sie außerdem Konstruktoren die Ihre Universität mit sinnvollen Daten befüllen. Wird das Objekt nicht direkt initialisiert, so soll `numStudents = 0`, `city = noWhere` und `name = noName` eingetragen werden. Überladen Sie den `<<`-operator um sämtliche Daten von `University` ausgeben zu können. Testen Sie Ihre Implementierungen geeignet! Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.3. Schreiben Sie eine Klasse `Alkohol` zur Speicherung von verschiedenen alkoholischen Getränken. Diese soll folgende Member-Variablen enthalten: Name, Alkoholgehalt in Prozent, Preis in €. Weiters soll für diese Klasse neben einem passenden Konstruktor auch der `operator<` definiert werden. Dieser soll nach besserem Verhältnis $\frac{\text{Vol.\%}}{€}$ bewerten. Definieren Sie auch die Methoden `getName()`, `getPreis()` und `getVolProz()`.
Hinweis: Für die Überladung des Operators `<` beachte man die allgemeine Syntax

```
bool operator<(const type& lhs, const type& rhs);
```

Hier bezeichnet `type` einen beliebigen Datentyp. In unserem Fall ist das also `Alkohol`. Überladen Sie darüber hinaus den `<<`-operator um sämtliche Daten von `Alkohol` ausgeben zu können. Testen Sie Ihre Implementierungen geeignet! Speichern Sie den Source-Code unter `Alkohol.{hpp/cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.4. Schreiben Sie eine Funktion `sortAlkohol` die ein Array der Länge $n > 0$ mit Einträgen vom Typ `Alkohol` aus Aufgabe 10.3 aufsteigend nach dem Verhältnis $\frac{\text{Vol.\%}}{€}$ sortiert. Sie dürfen jeden Sortieralgorithmus verwenden, den Sie kennen. Schreiben Sie darüber hinaus ein `main`-Programm, indem Sie die Länge n und die Einträge vom Typ `Alkohol` einlesen und `sortAlkohol` aufrufen. Testen Sie Ihre Implementierung geeignet! Speichern Sie den Source-Code unter `sortAlkohol.cpp` in das Verzeichnis `serie10`.

Aufgabe 10.5. Schreiben Sie eine Klasse `Bruch` zur Darstellung eines Bruchs $x = p/q$, wobei $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ als `int` gespeichert werden. Implementieren Sie

- den Standardkonstruktor (ohne Parameter), der $p = 0$ und $q = 1$ setzt,
- einen Konstruktor, der $p, q \in \mathbb{Z}$ mit $q \neq 0$ als Input übernimmt und den Bruch speichert,
- den Kopierkonstruktor,
- den Zuweisungsoperator und
- den Destruktor.

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. Implementieren Sie darüber hinaus die Zugriffsmethoden

- `setZaehler`, `getZaehler` für den Zähler und
- `setNenner`, `getNenner` für den Nenner.

Testen Sie ihre Implementierung geeignet! Speichern Sie den Source-Code unter `Bruch.{hpp/cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.6. Implementieren Sie eine Methode `kuerzen` der Klasse `Bruch` aus Aufgabe 10.5. Dabei sollen der Zähler p und Nenner 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 $q_0 = 1$.
- 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.

Testen Sie ihre Implementierung geeignet!

Aufgabe 10.7. Implementieren Sie die Type Castings von `Bruch` aus Aufgabe 10.5 auf `double` und auf `int`. Das Type Casting auf `int` soll die Nachkommastellen abschneiden. Beispielsweise soll casten von 3/2 als Ergebnis 1 liefern. Implementieren Sie auch den cast von double auf `Bruch`. Berücksichtigen Sie nur die ersten 9 Nachkommastellen. Übersetzen Sie diesen Anteil in einen Bruch mit dem Nenner 10^9 und kürzen Sie dann. Überladen Sie auch den Vorzeichenoperator $-$, der zum `Bruch` x den `Bruch` $-x$ liefert. Überladen Sie schließlich den `<<`-Operator um den `Bruch` $x := p/q$ in der Form `p/q` ausgeben zu können. Testen Sie Ihre Implementierung geeignet!

Aufgabe 10.8. Überladen Sie die Operatoren $+$, $-$, $*$ und $/$ um die Summe, die Differenz, das Produkt und den Quotienten zweier Brüche im Format `Bruch` aus Aufgabe 10.5 zu berechnen. Stellen Sie bei $/$ mittels assert sicher, dass Sie nicht durch 0 dividieren. Das Ergebnis soll in allen Fällen gekürzte Form haben. Testen Sie ihre Implementierung geeignet!