Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
05.06.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Erklären Sie den Unterschied zwischen Referenzen und Pointern mit eigenen Worten. Schreiben Sie exemplarisch einen Code, welcher die Inhalte zweier Variablen vertauscht, einmal mit Pointern und einmal mit Referenzen. Welche Vorteile bringt die Verwendung von Referenzen gegenüber Pointern mit sich? Welche Nachteile?

Aufgabe 10.2. 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<` überladen 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 10.3. Schreiben Sie eine Funktion `sortAlkohol` die ein Array der Länge $n > 0$ mit Einträgen vom Typ `Alkohol` aus Aufgabe 10.2 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 10.4. Erweitern Sie die Klasse `Fraction` von Folie 229 um

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

Aufgabe 10.5. Implementieren Sie eine Methode `reduce` für die Klasse `Fraction` aus Aufgabe 10.4, um einen Bruch auf die gekürzte Form $x = p/q$ zu bringen (mit $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ teilerfremd). Verwenden Sie einen beliebigen Algorithmus, um den größten gemeinsamen Teiler von Nenner und Teiler zu bestimmen, z.B. den Euklid-Algorithmus (VO Folien 96–98 und Folie 106) oder einen Algorithmus für die Primfaktorzerlegung von Integern. Testen Sie Ihre Implementierung geeignet!

Aufgabe 10.6. Überladen Sie die Operatoren $+$, $-$, $*$ und $/$ um die Summe, die Differenz, das Produkt und den Quotienten zweier Brüche im Format `Fraction` aus Aufgabe 10.4 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!

Aufgabe 10.7. Überladen Sie die Vergleichsoperatoren `==`, `!=`, `<`, `<=`, `>`, `>=` für die Klasse `Fraction` aus Aufgabe 10.4. Vergleichen dabei Sie Zähler und Nenner der Brüche direkt (*Hinweis:* Sie müssen einen gemeinsamen Nenner bestimmen). Testen Sie Ihre Implementierung geeignet!

Aufgabe 10.8. Was ist der Output des folgenden Programms? Erklären Sie warum! Was ist der Unterschied zwischen den verschiedenen verwendeten Variablentypen?

```
#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 << "var11 = " << var11 << " var12 = " << var12 << endl;

return 0;
}
```