Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
09.06.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1. Adaptieren Sie den Code der Klasse `Matrix` aus Aufgabe 10.5 und Aufgabe 10.6 so, dass Sie `new` bzw. `delete` anstatt `malloc` bzw. `free` verwenden. Was sind die Unterschiede zwischen `new` bzw. `delete` und `malloc` bzw. `free`? Unbewusst haben Sie letzte Woche beim Schreiben der Klasse `Matrix` bereits die "Dreierregel" aus der Vorlesung angewendet. Was besagt diese Regel? Warum ist sie in diesem Zusammenhang wichtig?

Aufgabe 11.2. Überladen Sie den Operator `*` und den Operator `/` für die Klasse `Matrix` aus Aufgabe 10.5 und Aufgabe 10.6, sodass die punktweise Multiplikation bzw. Division durchgeführt wird. Überladen Sie den Operator `*` erneut, um auch `Matrix * double` und `double * Matrix` durchführen zu können. Fügen Sie, wenn notwendig, Sicherheitsabfragen ein. Schreiben Sie auch ein Programm, in welchem Sie die Implementierung testen.

Aufgabe 11.3. Überladen Sie den Operator `<<` für die Klasse `Matrix` aus Aufgabe 10.5 und Aufgabe 10.6 um `cout << A` für eine Matrix A ausführen zu können. Testen Sie ihre Implementierung.

Aufgabe 11.4. 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`.

Aufgabe 11.5. Implementieren Sie ein einfaches *Tic Tac Toe* Spiel. Falls Sie Ihnen nicht bekannt sind, können Sie die Regeln unter `http://de.wikipedia.org/wiki/Tic_Tac_Toe` nachlesen. Das Spiel soll mindestens die folgenden Kriterien erfüllen:

1. Es sollen zwei Spieler gegeneinander antreten können die jeweils abwechselnd am Zug sind.
2. Es soll automatisch erkannt werden, wann ein Spieler gewonnen hat.
3. Nach jedem Zug soll das aktuelle Spielfeld grafisch in der Konsole ausgegeben werden.
4. Kann sich am Ende keiner der Spieler durchsetzen, so soll `'unentschieden'` ausgegeben werden.

Verwenden Sie die Klasse `Matrix` für das Spielfeld. Jeder Spieler soll ein eigenes Zeichen verwenden, z.B.: `'0'` für Spieler 1 und `'1'` für Spieler 2. Das Spielfeld soll dabei ein $n \times n$ Feld mit $n \geq 3$ sein.

Aufgabe 11.6. Implementieren Sie alle nötigen Konstruktoren und Destruktoren für die Klasse `Bruch` aus Aufgabe 10.2 um die "Dreierregel" zu erfüllen. Implementieren Sie außerdem eine Methode `kuerzen()`. (Hinweis: "Recyclen" Sie den Code aus Aufgabe 9.4.) Überladen Sie dann die Operatoren +,-,* und /. Verwenden Sie die Methode `kuerzen()` passend. Fügen Sie, wenn notwendig, Sicherheitsabfragen ein. Das Ergebnis soll wiederum ein Bruch sein. Testen Sie ihre Implementierung an geeigneten Beispielen.

Aufgabe 11.7. Überladen Sie die Vergleichsoperatoren `<,<=,>,>=` für die Klasse `Bruch` aus Aufgabe 10.2. Testen Sie ihre Implementierung an geeigneten Beispielen.

Aufgabe 11.8. Implementieren Sie eine Klasse `Person`, welche die Datenfelder `name` und `adresse` enthält. Leiten Sie von dieser Klasse eine Klasse `Student` ab, welche die zusätzlichen Datenfelder `matrikelnummer` und `studium` enthält. Leiten Sie von der Klasse `Person` auch eine Klasse `Arbeiter` ab. Erweitern Sie diese Klasse um die Datenfelder `gehalt` und `arbeit`. Schreiben Sie für alle Klassen die zugehörigen Zugriffsfunktionen, Konstruktoren und Destruktoren. Schreiben Sie ein main-Programm und testen Sie ihre Implemtierung!