Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
02.06.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Als Kunde einer Bank kann man auch mehrere Sparkonten besitzen. Erstellen Sie eine Klasse `Kunde` welche eine Liste von Sparkonten beinhaltet. Weiters soll die Klasse auch ein Objekt der Klasse `Name` aus Aufgabe 9.3 enthalten.Implementieren Sie Methoden zum Hinzufügen und Löschen von Konten. Schreiben Sie dann eine Methode die das Guthaben auf allen vorhandenen Sparkonten berechnet. Überlegen Sie welche Funktionen noch sinnvoll wären. Speichern Sie den Source-Code unter `kunde.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.2. Implementieren Sie die Zugriffsfunktionen der Klasse Bruch, welche durch

```
class Bruch {
   long zaehler;
   unsigned long nenner;
public:
   Bruch();
   Bruch(long zaehler, unsigned long nenner);
   setZahler(long z);
   setNenner(unsigned long n);
   double getWert();
};
```

gegeben ist. Die Methode `getWert` soll dabei den Dezimalwert des Bruchs zurückgeben. Achten Sie bei den `set`-Funktionen auf eventuelle Sicherheitsabfragen. Implementieren Sie auch den Konstruktor `Bruch()`, welcher den Zähler auf 0 und den Nenner auf 1 setzen soll. Speichern Sie den Source-Code unter `bruch.cpp` in das Verzeichnis `serie10`.

Aufgabe 10.3. Was gibt folgender Code am Bildschirm aus und warum?

```
#include <iostream>
#include <string>
using namespace std;

class T1 {
   string t1;
public:
   T1(string val) { cout << "Ich bin Konstruktor von " << val << endl; t1=val; }
   T1() { cout << "Ich bin Konstruktor von default" << endl; t1="default"; }
   ~T1() { cout << "Ich bin Destruktor von " << t1 << endl; }
};

int main() {
   T1 bert("bert");
   T1 bob;
   T1 def("bob");
   return 0;
}
```

Aufgabe 10.4. Erweitern Sie die Klasse Vector aus der Vorlesung mit einer Methode `unique`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert und doppelte Einträge streicht. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Speichern Sie den Source-Code unter `unique.cpp` in das Verzeichnis `serie10`.

Aufgabe 10.5. Schreiben Sie eine Klasse `Matrix` zur Speicherung von Matrizen der Größe $m \times n$. Die Einträge sollen hierbei als langer `double*`-Vektor der Länge mn gespeichert werden. Schreiben Sie Methoden, welche die Möglichkeit bieten Einträge zu erstellen, bzw. auszulesen. Achten Sie hierbei auf eventuelle Sicherheitsabfragen. Schreiben Sie einen Konstruktor, der bei übergebenen Werten $m, n \in \mathbb{N}$ eine Nullmatrix der Größe $m \times n$ erzeugt, sowie einen Standardkonstruktor, der eine leere Matrix der Größe 0×0 generiert. Natürlich darf auch ein entsprechender Destruktor nicht fehlen. Speichern Sie den Source-Code unter `matrix.{hpp/cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.6. Erweitern Sie die Klasse `Matrix` aus Aufgabe 10.5 mit

- einen Zuweisungsoperator,
- einen Kopierkonstruktor,
- einen Operator um sinnvoll $-A$ ausführen zu können,
- einer Methode `scanMatrix(int n,int m)`, die eine $n \times m$-Matrix von der Tastatur einliest,
- einer Methode `printMatrix()`, die die Matrix am Bildschirm ausgibt,
- und einer Methode `transpose` zum Überschreiben einer gespeicherten Matrix mit der transponieren Matrix. (Hinweis: Die Transponierte $A^T \in A \in \mathbb{R}^{n \times m}$ einer Matrix $A \in \mathbb{R}^{m \times n}$ ist durch die Beziehung $\left(A^T\right)_{jk} = A_{kj}$ definiert. Es gilt dann $\left(A^T\right)^T = A$.)

Schreiben Sie weiters ein aufrufendes main Programm um ihre implementierten Methoden zu testen. Speichern Sie den Source-Code unter `matrix.{hpp/cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.7. 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 10.8. Schreiben Sie ein `Makefile` für die aktuelle Übungsserie. Dieses soll zumindest folgende Dinge umfassen:

- Erzeugen von Programmen aller von Ihnen gelösten Aufgaben.
- Das Generieren einer Bibliothek und deren Verwendung in einem Programm.