Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
07.01.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 13

Aufgabe 13.1. Schreiben Sie eine Funktion `unique`, welche die doppelten Einträge aus einem Vektor entfernt und den gekürzten Vektor zurückliefert. Verwenden Sie hierzu den C++-Standardcontainer `vector`. Erklären Sie die Unterschiede zu Ihrer ersten Implementierung. Speichern Sie den Source-Code unter `unique.cpp` in das Verzeichnis `serie13`.

Aufgabe 13.2. Schreiben Sie eine Klasse `University`. Diese soll neben den Feldern `anzStudents`, `city` und `name` die Methoden `graduate` und `newStudent` haben. Wird `graduate` aufgerufen, so verringert sich die Anzahl der Studenten um 1, wohingegen `newStudent` die Anzahl um 1 erhöht. Alle Datenfelder sollen als `private` deklariert sein. Sie müssen sich also zusätzlich `get-` und `set`-Methoden schreiben. Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie13`.

Aufgabe 13.3. Für die Personalabteilung der Universität ist es sehr mühsam, Studenten immer nur einzeln hinzuzufügen oder aus dem System zu löschen. Schreiben Sie also Funktionen denen die Anzahl der abschließenden bzw. neu hinzukommenden Studenten mit übergeben werden kann. (Hinweis: Wie bei Konstruktoren können Sie hierfür die gleichen Funktionsnamen `graduate` und `newStudent` verwenden, solange die Funktionen andere Signatur erhalten. Dieses Vorgehen nennt man *Funktionen überladen.*) Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie13`.

Aufgabe 13.4. 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 `serie13`. *Hinweis:* Verwenden Sie `new` und `delete[]` um Speicher zu allokieren bzw. freizugeben. Achten Sie auch auf die "Dreierregel".

Aufgabe 13.5. Überladen Sie den Operator `+` für die Klasse `MyVector` aus der VO. Schreiben Sie auch ein Programm in welchem Sie die Implementierung testen.

Aufgabe 13.6. Überladen Sie den Operator `*` für die Klasse `MyVector` aus der VO. Dabei sei die Multiplikation zweier Vektoren $x, y \in \mathbb{R}^n$ komponentenweise definiert, d.h die j-te Komponente des Produktvektors $z = x * y$ lautet $z_j = x_j y_j$. Schreiben Sie auch ein Programm in welchem Sie die Implementierung testen.

Aufgabe 13.7. 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 13.8. Was tut der folgende C++ Code? Welche Funktionalität haben die Funktionen `func1` und `func2`? Wie sieht die Bildschirmausgabe aus?

```
#include <iostream>
#include <vector>

using namespace std;

void func2(vector<double> &dp, int mp);
```

```
void func1(vector<double> &dp)
{
   int mp = dp.size();
   func2(dp,mp);

   for (int i = mp-1; i>=1; i--){
      for(int j = 0; j<i; j++){
         if (dp[j] > dp[j+1])
            swap(dp[j], dp[j+1]); //Vertausche dp[j] und dp[j+1]
      }
      func2(dp,mp);
   }
}

void func2(vector<double> &dp, int mp){
   for (int k = 0; k<= mp-1; k++){
      cout << dp[k] << endl;
   }
}

int main(){
   vector<double> a(4,0);
   a[0] = 14;
   a[1] = 12;
   a[2] = 7;
   a[3] = 4;
   func1(a);
}
```