Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
20.12.2013

**Übungen zur Vorlesung**
**Einführung in das Programmieren für TM**

**Serie 12**

**Aufgabe 12.1.** Schreiben Sie eine Klasse `Stoppuhr` welche zur Simulation einer Stoppuhr dienen soll. Die Stoppuhr bestehe dabei aus zwei Knöpfen. Wird der erste Knopf gedrückt, so soll die Zeitmessung gestartet werden. Wird dieser Knopf nochmals gedrückt, wird die Zeitmessung gestoppt. Der zweite Knopf dient dazu die Zeit wieder zurückzusetzen. Schreiben Sie dazu die Methoden `pushButtonStartStop` und `pushButtonReset`. Implementieren Sie weiters eine Methode, welche die verstrichene Zeit im Format `hh:mm:ss.xx` ausgibt. (Beträgt die gemessene Zeit also zwei Minuten so soll `00:02:00.00` ausgegeben werden) Sie können diese Stoppuhr nun dazu verwenden Zeitmessungen durchzuführen. Speichern Sie den Source-Code unter `stoppuhr.{hpp,cpp}` in das Verzeichnis `serie12`.
*Hinweis:* Verwenden Sie den Datentyp `clock_t` und die Funktion `clock()` aus der Bibliothek `time.h`. Vermutlich ist es auch sinnvoll eine Variable `isRunning` vom Typ `bool` einzuführen. Bei Betätigen des ersten Knopfes wird diese Variable entweder von `false` auf `true` gesetzt oder umgekehrt.

**Aufgabe 12.2.** Schreiben Sie eine Klasse `Hangman` mit den Methoden `guessChar`, `solve` und `newString`. Die Klasse soll nun einen string der Länge $n$ speichern, den es zu erraten gilt. Nach und nach dürfen mittels `guessChar` Buchstaben geraten werden, wobei die Methode immer den Index, bzw. die Indizes der eingegebenen Buchstaben auf dem Bildschirm ausgibt. Falls der eingegebene Buchstabe im gesuchten Wort nicht vorkommt, soll eine entsprechende Meldung ausgegeben werden. Der Spieler verliert, wenn er 8 mal falsch geraten hat. Schreiben Sie alle nötigen Zugriffsfunktionen und Konstruktoren und außerdem die Methoden `solve` und `newString` um das Rätsel zu lösen bzw. das Spiel mit einem neuen Wort neu zu beginnen. Testen Sie Ihr Spiel indem Sie mittels einer geeigneten Schleife solange neue Buchstaben raten, bis Sie entweder gewonnen oder verloren haben. Speichern Sie den Source-Code unter `hangman.{hpp,cpp}` in das Verzeichnis `serie12`.

**Aufgabe 12.3.** Erweitern Sie die Klasse `Bruch` aus der Vorlesung um die `public` Methode `void kuerzen()`, die die gekürzte Darstellung des Bruchs `zaehler/nenner` bestimmt. Dazu benütze man den euklidischen Algorithmus aus der Vorlesung. Weiters implementiere man eine Methode `setWert(string wert)`, welche eine beliebige Gleitpunktzahl in einen Bruch umwandelt. Die Zahl ist dabei als String gegeben. Um diese Methode zu erstellen können Sie wie folgt vorgehen. Zunächst sucht man in dem String nach dem Dezimalpunkt und zählt die Nachkommastellen. Dann löscht man den Dezimalpunkt aus dem String heraus. Den String, welcher nun eine natürliche Zahl repräsentiert, kann nun mittels der Funktion `atoi` in eine `int` Variable umgewandelt werden. Diese Zahl benützt man als Zähler. Als Nenner verwende man $10^p$ wobei $p \in \mathbb{N}$ die Anzahl der Nachkommastellen sei. Speichern Sie den Source-Code unter `bruch2.cpp` in das Verzeichnis `serie12`.
*Hinweis:* Mit der Methode `find` der Klasse `string` können Sie nach einem bestimmten Zeichen im String suchen, z.B.: `int pos = wert.find('.')` gibt die Stelle des Dezimalpunkts im String `wert` an. Mit `wert.erase(pos,k)` werden ab der Stelle `pos` die `k` darauf folgenden Zeichen im String gelöscht. Die Funktion `atoi` aus der Standardbibliothek `cstdlib` konvertiert einen gegebenen String (im C-Stil) in eine `int` Variable. Um die Zeichenkette eines Strings zu erhalten kann man die Methode `c_str()` der Klasse `string` benutzen.

**Aufgabe 12.4.** Erstellen Sie eine Klasse `Name` welche zwei String-Variablen `vorname` und `nachname` enthält. Implementieren Sie auch die Zugriffsfunktion `setName`, welche einen String übernimmt, diesen in Vor- und Nachname aufteilt und in die entsprechenden Variablen abspeichert. Beachten Sie auch, dass mehrere Vornamen vorhanden sein können! Schreiben Sie weiters eine Methode `printName` die Vor- und Nachname am Bildschirm ausgibt. Bei mehreren Vornamen soll, beginnend ab dem zweiten Vornamen, jeder weitere Vorname mit dem Anfangsbuchstaben und einem Punkt abgekürzt werden! Beim Namen `Max Maxi Mustermann` soll also `Max M. Mustermann` am Bildschirm erscheinen. Speichern Sie den Source-Code unter `name.{hpp,cpp}` in das Verzeichnis `serie12`.

**Aufgabe 12.5.** Erstellen Sie eine Klasse `SparKonto` mit den Variablen `kontonummer`, `guthaben` und `zinssatz`. Ferner sollen noch die `get` und `set` Funktionen für die Variablen `zinssatz` und `kontonummer` implementiert werden. Um das Guthaben zu ändern schreiben Sie die Methoden `abheben` und `einzahlen`. Beachten Sie, dass Sie bei einem Sparkonto nicht ins Minus gehen können. Der Zinssatz und die Kontonummer dürfen natürlich auch nicht negativ werden. Schließlich implementiere man noch die Methode `berechneGuthaben`. Speichern Sie den Source-Code unter `sparkonto.{hpp,cpp}` in das Verzeichnis `serie12`.

**Aufgabe 12.6.** Als Kunde einer Bank kann man auch mehrere Sparkonten besitzen. Erstellen Sie eine Klasse `Kunde` welche eine Liste von Sparkonten beinhaltet. Benutzen Sie dazu den Container `vector`. Weiters soll die Klasse auch ein Objekt der Klasse `Name` aus Aufgabe 12.4 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 `serie12`.

**Aufgabe 12.7.** 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 dynamischen Bibliothek und deren Verwendung in einem Programm. Zum Beispiel können Sie eine dynamische Bibliothek für die Stoppuhr erzeugen. Diese Bibliothek können Sie dann in jedem Programm, welches eine Methode zur Zeitmessung benötigt, einsetzen.

**Aufgabe 12.8.** 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;
}
```