Dirk Praetorius,
Michele Ruggeri

Wintersemester 2015/16
16.-17.12.2016

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Die Standardeingabe `cin` in C++ liest eine Eingabe, nur bis zum ersten Trennzeichen. Schreiben Sie eine Funktion `myFullName`, die Ihren Vor- und Nachnamen über die Tastatur einliest und jeweils in einem string abspeichert. Diese beiden Strings sollen nun in einem einzelnen String gespeichert werden, der schließlich ausgegeben wird. Speichern Sie den Source-Code unter `MyFullName.cpp` in das Verzeichnis `serie10`. Testen Sie Ihren Code an einem geeigneten Beispiel. Kennen Sie eine Möglichkeit um längere Tastatureingaben einzulesen?

Aufgabe 10.2. 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 `serie10`.

Aufgabe 10.3. 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. Rufen Sie dann `kuerzen()` auf. Überladen Sie schließlich die Methode `setWert` geeignet, um auch `setWert(n)` für n vom Typ `int` in sinnvoller Weise ausführen zu können. Speichern Sie den Source-Code unter `bruch.{hpp,cpp}` in das Verzeichnis `serie10`. Testen Sie Ihren Code an einem geeigneten Beispiel.
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 10.4. 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 `serie10`.
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 10.5. 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.

Aufgabe 10.6. Für die Personalabteilung der Universität ist es sehr mühsam, Studenten immer nur einzeln hinzuzufügen oder aus dem System zu löschen. Überladen Sie die Methoden `graduate` und `newStudent` der Klasse `University` aus Aufgabe 9.7 so, dass die Anzahl der abschließenden bzw. neu hinzukommenden Studenten mit übergeben werden kann. Schreiben Sie außerdem Konstruktoren die Ihre Universität mit sinnvollen Daten befüllen. Wird das Objekt nicht direkt initialisiert, so soll `numStudents = 0`, `city = noWhere` und `name = noName` eingetragen werden. Erweitern Sie die Klasse zusätzlich um eine `plot` -Routine, welche sämtliche Daten von `University` ausgibt. Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.7. 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. Schreiben Sie ein main Programm und testen Sie ob ihre Implementierung korrekt ist. Speichern Sie den Source-Code unter `hangman.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.8. Lt. der Vorlesung ist der Zugriff auf Members einer Klasse vom Typ `private` nur über `set`- und `get`-Methoden der Klasse möglich. Wie lautet die Ausgabe des folgenden C++ Programms? Warum ist das möglich? Erkären Sie warum das schlechter Programmierstil ist.

```
#include <iostream>
using std::cout;
using std::endl;

class Test{

private:
  int N;

public:
  void setN(int N_in) { N = N_in; };
  int getN(){ return N; };
  int* getptrN(){ return &N; };

};


int main(){

  Test A;
  A.setN(5);
  int* ptr = A.getptrN();
  cout << A.getN() << endl;
  *ptr = 10;
  cout << ptr << endl;
  cout << A.getN() << endl;

  return 0;
}
```