Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
19.05.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1. Alternativ kann man die Ableitung $f'(x)$ auch

- durch den zentralen Differenzenquotienten

$$\Phi(h) := \frac{f(x+h) - f(x-h)}{2h} \quad \text{für } h > 0$$

- und durch den einseitigen Differenzenquotienten

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{für } h > 0$$

approximieren. Schreiben eine Funktion `diff` die den zentralen Differenzenquotienten verwendet und eine Funktion `diff2` die den einseitigen Differenzenquotienten verwendet. Übergeben Sie die Funktion f als Parameter an `diff` und `diff2`! Überlegen Sie, welche weiteren Parameter diese Funktionen benötigen. Testen Sie Ihre Funktionen mit $f(x) = \exp(x)$ an $x = 1$. Vergleichen Sie die Laufzeit. Speichern Sie den Source-Code unter `diffcomp.c` in das Verzeichnis `serie09`.

Aufgabe 9.2. 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 `serie09`. Testen Sie Ihren Code an einem geeigneten Beispiel. Kennen Sie eine Möglichkeit um längere Tastatureingaben einzulesen?

Aufgabe 9.3. 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.cpp` in das Verzeichnis `serie09`.

Aufgabe 9.4. 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.cpp` in das Verzeichnis `serie09`. 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 9.5. Schreiben Sie eine Klasse `University`. Diese soll neben den Feldern `numStudents`, `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.cpp` in das Verzeichnis `serie09`.

Aufgabe 9.6. 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.cpp` in das Verzeichnis `serie09`.

Aufgabe 9.7. 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.cpp` in das Verzeichnis `serie09`.
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 9.8. 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;
}
```