Dirk Praetorius,
Gregor Gantner

Wintersemester 2018/19
20./21.12.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1. Schreiben Sie einen Strukturdatentyp `polynomial` zur Speicherung von Polynomen, die bezüglich der Monombasis dargestellt sind, d.h. $p(x) = \sum_{j=0}^{n} a_j x^j$. Es ist also der Grad $n \in \mathbb{N}_0$ sowie der Koeffizientenvektor $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ zu speichern. Schreiben Sie alle nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`). Speichern Sie den Source-Code unter `polynomial.c` in das Verzeichnis `serie08`.

Aufgabe 8.2. Die Summe $r = p + q$ zweier Polynome p, q ist wieder ein Polynom. Schreiben Sie eine Funktion `addPolynomials`, die die Summe r berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 8.1. Zum Test schreibe man eine Funktion, die zwei Polynome einliest und deren Summe ausgibt. Speichern Sie den Source-Code unter `addPolynomials.c` in das Verzeichnis `serie08`.

Aufgabe 8.3. Das Produkt $r = pq$ zweier Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{j=0}^{n} b_j x^j$ ist wieder ein Polynom. Schreiben Sie eine Funktion `prodPoly`, die das Produktpolynom r berechnet und in der Struktur aus Aufgabe 8.1 speichert. Überlegen Sie sich zunächst, welchen Grad das Polynom r hat und wie sich die Koeffizienten berechnen lassen. Schreiben Sie ein aufrufendes Hauptprogramm, in dem p und q eingelesen und $r = pq$ ausgegeben wird. Testen Sie Ihren Code an einem geeigneten Beispiel. Speichern Sie den Source-Code unter `prodPoly.c` in das Verzeichnis `serie08`.

Aufgabe 8.4. Schreiben Sie eine Funktion `evalPoly`, die für ein gegebenes Polynom p und einen Punkt $x \in \mathbb{R}$ den Funktionswert $p(x)$ berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 8.1. Testen Sie Ihren Code an einem geeigneten Beispiel. Speichern Sie den Source-Code unter `evalPoly.c` in das Verzeichnis `serie08`.

Aufgabe 8.5. Schreiben Sie eine Funktion `evalDiffPoly`, die für ein gegebenes Polynom p, eine Ableitungsordnung $k \in \mathbb{N}_0$ und einen Punkt $x \in \mathbb{R}$ den Funktionswert $p^{(k)}(x)$ zurückgibt. Dabei soll das Polynom $p^{(k)}$ nicht explizit gebildet und gespeichert werden. Verwenden Sie für das Polynom p die Struktur aus Aufgabe 8.1. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem der Grad n, das Polynom p, die Ableitungsordnung k und der Punkt x eingelesen werden und $p^{(k)}(x)$ ausgegeben wird. Speichern Sie den Source-Code unter `evalDiffPoly.c` in das Verzeichnis `serie08`.

Aufgabe 8.6. Mit dem Satz von Taylor (de.wikipedia.org/wiki/Taylor-Formel) kann eine glatte Funktion $f\colon \mathbb{R} \to \mathbb{R}$ durch ein Polynom vom Grad $n \in \mathbb{N}$ lokal um eine Stelle x_0 approximiert werden:

$$f(x) \approx \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0)(x - x_0)^k.$$

Hier bezeichnet $f^{(n)}$ die n-te Ableitung von f. Schreiben Sie eine Funktion, die für ein gegebenes Polynom p die Approximation vom Grad n um x_0 berechnet und diese wiederum als Polynom zurückgibt. Dabei sollen die Polynome in der Struktur aus Aufgabe 8.1 gespeichert werden. Achten Sie außerdem auf eine effiziente Berechnung von $1/k!$ und $(x-x_0)^k$. Speichern Sie den Source-Code unter `approxPolynomials.c` in das Verzeichnis `serie08`. *Hinweis:* Um die Korrektheit ihrer Funktion zu überprüfen, kann es hilfreich sein, das ursprüngliche Polynom, sowie die Approximation zeichnen zu lassen (WolframAlpha, ...).

Aufgabe 8.7. Erstellen Sie eine Klasse `SparKonto` mit den Variablen `kontonummer`, `guthaben` und `zinssatz`. Ferner sollen noch die `get` und `set`-Methoden 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`. Testen Sie Ihren Code entsprechend. Speichern Sie den Source-Code unter `sparkonto.cpp` in das Verzeichnis `serie08`.

Aufgabe 8.8. Laut der Vorlesung ist der Zugriff auf `private` Members einer Klasse 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;
}
```