Dirk Praetorius,
Michele Ruggeri

Sommersemester 2017
13.06.2011

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 12

Aufgabe 12.1. Schreiben Sie die Klassendefinition zu einer Klasse `Polynomial` zur Speicherung von Polynomen vom Grad $n \in \mathbb{N}$, die bezüglich der Monombasis dargestellt sind, d.h.

$$p(x) = \sum_{j=0}^{n} a_j x^j.$$

In der Klasse soll neben dem dynamischen Vektor $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ der Koeffizienten `(double*)` auch der Grad $n \in \mathbb{N}$ gespeichert werden. Implementieren Sie die folgenden Funktionalitäten:

- Destruktor, Konstruktor zum Allokieren des Null-Polynoms mit Grad n, Copy-Konstruktor,
- Zuweisungsoperator,
- Zugriff auf die Koeffizienten des Polynoms mittels `[ ]`, d.h. für $0 \le j \le n$ liefert `p[j]` dann a_j und
- die Möglichkeit ein Polynom `p` mit `cout << p` in der Monombasis ausgeben zu können.

Schreiben Sie auch ein main-Programm, in welchem Sie die Implementierung testen.

Aufgabe 12.2. Die Summe zweier Polynome ist wieder ein Polynom. Implementieren Sie für die Klasse `Polynomial` aus Aufgabe 12.1 die nötige Funktionalität, um zwei Polynome p und q mittels `r=p+q` zu addieren. Implementieren Sie hinaus darüber die Möglichkeit, für eine skalare Größe $a \in \mathbb{R}$ im Format `double` bzw. `int` und für ein Polynom p die Operationen `r=a+p` oder `r=p+a` in sinnvoller Weise ausführen zu können. Schreiben Sie auch ein main-Programm, in welchem Sie die Implementierung testen.

Aufgabe 12.3. Das Produkt zweier Polynome ist wieder ein Polynom. Implementieren Sie für die Klasse `Polynomial` aus Aufgabe 12.1 die nötige Funktionalität, um zwei Polynome p und q mittels `r=p*q` zu multiplizieren. Implementieren Sie hinaus darüber die Möglichkeit, für eine skalare Größe $a \in \mathbb{R}$ im Format `double` bzw. `int` und für ein Polynom p die Operationen `r=a*p` und `r=p*a` in sinnvoller Weise ausführen zu können. Schreiben Sie auch ein main-Programm, in welchem Sie die Implementierungen testen.

Aufgabe 12.4. Für $k \ge 0$ ist die k-te Ableitung $p^{(k)}$ eines Polynoms p wieder ein Polynom. Implementieren Sie für die Klasse `Polynomial` aus Aufgabe 12.1 folgende Funktionalitäten:

- die Möglichkeit die k-te Ableitung von einem Polynom p mittels `p(k,x)` auszuwerten, wobei $x \in \mathbb{R}$ (`double`) und $k \ge 0$ (`int`);
- für $k = 0$ sei der Aufruf `p(x)` erlaubt, d.h. `p(0,x)` und `p(x)` liefern gleiche Ergebnisse;
- die Möglichkeit die k-te Ableitung von einem Polynom p mittels `p(k)` zu erstellen, wobei $k \ge 0$ (`int`).

Schreiben Sie auch ein main-Programm, in welchem Sie die Implementierung testen.

Aufgabe 12.5. Implementieren Sie eine Methode `computeZero` für die Klasse `Polynomial` aus Aufgabe 12.1 zur Bestimmung einer Nullstelle des Polynoms mit Hilfe des Newton-Verfahrens. Dieser wird eine Zahl $x_0 \in \mathbb{R}$ übergeben, mit Hilfe derer dann durch das Newton-Verfahren eine Nullstelle des Polynoms berechnet und zurückgegeben werden soll. Hierbei gehen Sie wie folgt vor: Ausgehend vom Startwert x_0 definieren Sie induktiv die Folge $\{x_k\}$ durch

$$x_k = x_{k-1} - p(x_{k-1})/p'(x_{k-1}) \quad \text{für } k \ge 1,$$

wobei p' die Ableitung von p bezeichnet. Die Folge $\{x_k\}$ konvergiert dann gegen eine Nullstelle von p. Für eine gegebene Toleranz τ soll die Iteration abgebrochen werden, falls entweder

$$|p'(x_k)| \leq \tau$$

oder

$$|p(x_k)| \leq \tau$$

gilt. Im ersten Fall soll ein Hinweis ausgegeben werden, dass das numerische Ergebnis wahrscheinlich falsch ist. Außerdem soll das Programm in diesem Fall kontrolliert beendet werden (`assert`). Im zweiten Fall werde x_k als Approximation der gesuchten Nullstelle zurückgegeben. Testen Sie Ihren Code entsprechend!

Aufgabe 12.6. Implementieren Sie eine Methode `computeIntegral` für die Klasse `Polynomial` aus Aufgabe 12.1 zur Bestimmung von Integralen eines Polynoms p. Für zwei Zahlen $\alpha, \beta \in \mathbb{R}$ mit $\alpha < \beta$ soll die Methode das Integral

$$\int_\alpha^\beta p(x)\, dx$$

berechnen und zurückgegeben. Für $p(x) = \sum_{j=0}^n a_j x^j$ gilt

$$\int_\alpha^\beta p(x)\, dx = \sum_{j=0}^n \frac{a_j(\beta^{j+1} - \alpha^{j+1})}{j+1}.$$

Wie kommt man auf diese Formel? Stellen Sie mittels `assert` sicher, dass $\alpha < \beta$ gilt. Testen Sie Ihren Code entsprechend!

Aufgabe 12.7. Zwei Polynome sind identisch genau dann wenn sie denselben Grad haben und alle Koeffizienten gleich sind. Überladen Sie den `==` entsprechend! Überprüfen Sie dabei für jeden Koeffizienten nicht auf Gleichheit sondern lediglich ob die Abweichung kleiner als eine vorgegebene Toleranz ist. Warum ist das sinnvoll? Schreiben Sie auch ein main-Programm, in welchem Sie die Implementierung testen.

Aufgabe 12.8. Das Taylor-Polynom vom Grad n einer Funktion $f \in C^n(\mathbb{R})$ an einer Stelle $x_0 \in \mathbb{R}$ ist gegeben durch

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

Schreiben Sie einen Konstruktor für die Klasse `Polynomial` aus Aufgabe 12.1 der für ein $n \geq 0$ wahlweise das Taylor-Polynom vom Grad n der Funktionen `sin`, `cos` oder `exp` in $x_0 = 0$ erzeugt. Testen Sie Ihren Code entsprechend!