Dirk Praetorius,
Marcus Page

Sommersemester 2012
24.05.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1*. *[Matrixklasse]* Schreiben Sie eine Klasse zur Speicherung von Matrizen $A \in \mathbb{R}^{m\times n}$. Die Koeffizienten sollen hierbei in in der Form `<vector<vector<double> >` gespeichert werden. Die Klasse soll außerdem über eine Methode zur Ausgabe der Frobeniusnorm verfügen, sowie einen Konstruktor bereitstellen, der die Matrix mit einem gegebenen `double` Vektor der Länge $m * n$ initialisiert. Erweitern Sie hierzu die Klasse aus der Vorlesung geeignet.

Aufgabe 10.2*. *[Zugriffsspezifizierer]* Was bedeuten die Zugriffsspezifizierer `private` und `public`? Was sind die Vorteile ihrer Verwendung? Wieso ist Zugriffskontrolle in C++ stärker als in C?

Aufgabe 10.3*. *[Matrixoperationen]* Schreiben Sie Methoden `add`, `multiply` und `subtract` mit denen Sie Matrizen addieren, multiplizieren bzw. subtrahieren können. Der Rückgabewert soll wieder in der Matrixklasse aus Aufgabe 10.1 gespeichert werden. Stellen Sie sicher, dass die zugehörigen Dimensionen für die entsprechende Operation auch passen und reagieren Sie entsprechend, wenn dies nicht der Fall ist. Sie kennen mehrere Methoden, Matrizen an Funktionen zu übergeben. Welche sind das? Worin unterscheiden sich diese Möglichkeiten?

Aufgabe 10.4*. *[Vektorklasse]* Schreiben Sie eine Klasse `vektor` zur Speicherung von Vektoren $v \in \mathbb{R}^n$. Die Klasse soll über die nötigen `get-` und `set`-Methoden verfügen (Sicherheitsabfragen!) und über Konstruktoren analog zur Matrixklasse aus Aufgabe 10.1. Außerdem soll die Methode `getnorm` die ℓ^2-Norm des Vektors v, gegeben durch

$$\|v\|_{\ell^2} := \sqrt{\sum_{i=0}^{n-1} v_i^2}$$

zurückgeben.

Aufgabe 10.5. *[Lösen]* Erweitern Sie Ihre Matrixklasse um eine Methode `solve` der Sie den Vektor $b \in \mathbb{R}^n$ übergeben und die das Gleichungssystem $Ax = b$ für $A \in \mathbb{R}^{n\times n}$ löst. Die Lösung soll wieder als `vektor` gespeichert werden. Falls $m \neq n$, werde der NULL-Pointer zurückgegeben.
Hinweis: Sofern die Dimensionen stimmen, dürfen Sie davon ausgehen, dass das System eindeutig lösbar ist.

Aufgabe 10.6. *[Klassendefinition]* Konstruieren Sie eine Klasse `Polynomial` die ein Polynom p der Länge $n \in \mathbb{N}$ mittels seiner Koeffizienten $a_i \in \mathbb{R}$ abspeichert. Der Datentyp soll also

$$p = \sum_{j=0}^{n-1} a_j x^j \tag{1}$$

speichern können. Die Klasse soll außerdem die folgenden Funktionen beinhalten:

- `getZeros`: bestimmt die Nullstelle(n) des Polynoms (bis zum Grad 2) und gibt sie am Bildschirm aus. Hierbei soll zusätzlich überprüft werden ob der Grad des Polynoms kleiner als 3 ist und ansonsten entsprechend reagiert werden.
- `derivative`: bestimmt die erste Ableitung des Polynoms und gibt sie wieder als Pointer auf ein Polynom zurück.
- `antiderivative`: bestimmt eine Stammfunktion des Polynoms und gibt sie wieder als Pointer auf ein Polynom zurück.

Die Datenfelder sollen selbstverständlich als `private` deklariert werden, Sie müssen sich also Zugriffsfunktionen schreiben (Sicherheitsabfragen!). Schreiben Sie außerdem Konstruktoren um das Nullpolynom, bzw. ein Polynom der Länge n zu initialisieren. Die Koeffizienten sollen hierbei im Konstruktor von der Tastatur eingelesen werden. Außerdem soll es die Möglichkeit geben zwei Polynome zu addieren.
Hinweis: In dieser Aufgabe sollen die Funktionalität der Methoden `getZeros`, `derivative` und `antiderivative` noch nicht implementiert werden.

Aufgabe 10.7. *[Nullstellen & Operatoren]* Implementieren Sie die Funktion `getZeros`, sowie die Möglichkeit zwei Polynome zu addieren.
Tip: Es könnte sinnvoll sein einen neuen Konstruktor zu schreiben um das Zielpolynom leichter zu erzeugen.

Aufgabe 10.8. *[Polynome & Ableitungen]* Implementieren Sie die Funktionen `derivative` und `antidervative` wie oben beschrieben.

Aufgabe 10.9. *[Interpolation]* Schreiben Sie eine Funktion `interp` der vier reelle Zahlen x_1, x_2, y_1, y_2 übergeben werden und die die Punkte $(x_1, y_1), (x_2, y_2)$ linear interpoliert. Das Ergebnis ist ein Polynom und soll als Pointer auf ein solches zurückgegeben werden.

Aufgabe 10.10. *[Pointer auf Objekte]* Wo ist der Unterschied zwischen `new/delete` und `malloc/free`? Wieso ist es wichtig, bei Objekten die erste Variante zu verwenden?