Dirk Praetorius,
Marcus Page

Sommersemester 2012
29.05.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1*. *(Pointer & Referenzen)*
Schreiben Sie eine Funktion `minMax` der Sie ein `int`-Array übergeben und die den größten und den kleinsten Eintrag zurückliefert. Welche Möglichkeiten kennen Sie um das zu realisieren? Implementieren Sie beide und vergleichen Sie den Code.

Aufgabe 11.2*. *(Komplexe Zahlen, typecasts überladen)*
Schreiben Sie einen Datentyp `Complex` zur Speicherung von komplexen Zahlen. Dieser soll Real- und Imaginärteil jeweils als `double` speichern. Außerdem soll der Datentyp über entsprechende Zugriffsfunktionen, sowie eine `print`-Methode zur Ausgabe verfügen. Orientieren Sie sich hierbei an der Vorlesung. Eine reelle Zahl $x \in \mathbb{R}$ ist auch eine komplexe Zahl. Implementieren Sie einen typecast, bzw. einen entsprechenden Konstruktor der es Ihnen erlaubt einer Variable vom Typ `Complex` einen `double`-Wert zuzuweisen und umgekehrt (nur Realteil werde verwendet). Erstellen Sie außerdem ein UML-Diagramm Ihrer neuen Klasse `Complex`.

Aufgabe 11.3*. *(Arithmetik komplexer Zahlen, Operatoren überladen, gemischte Arithmetik)*
Implementieren Sie die Arithmetik komplexer Zahlen, d.h. implementieren Sie die nötige Funktionalität um zwei komplexe Zahlen $a, b \in \mathbb{C}$ mittels `+, -, *` addieren, subtrahieren und multiplizieren zu können. Implementieren Sie außerdem *gemischte Arithmetik*, d.h. die Möglichkeit eine komplexe Zahl $a \in \mathbb{C}$ mit einer `double` Zahl $b \in \mathbb{R}$ addieren, subtrahieren und multiplizieren zu können. Was müssen Sie hierbei beachten?

Aufgabe 11.4*. *(Arithmetik von Matrizen)*
Implementieren Sie die Arithmetik von Matrizen, d.h. implementieren Sie die nötige Funktionalität um zwei Matrizen $A \in \mathbb{R}^{m \times n}$ und $B \in \mathbb{R}^{p \times q}$ mittels `+, -, *` addieren, subtrahieren und multiplizieren zu können. Achten Sie hierbei darauf die Dimensionen entsprechend zu überprüfen. Achten Sie ferner darauf die entsprechenden Matrizen geschickt zu übergeben, d.h. nicht einfach zu kopieren. Erweitern Sie hierzu die Matrixklasse aus der Vorlesung und dem letzten Übungszettel geeignet.

Aufgabe 11.5. *(gemischte Arithmetik von Matrizen)*
Überladen Sie die Multiplikationsroutine aus der letzten Aufgabe so, dass – je nach Signatur – auch eine Matrix-Vektor-Multiplikation möglich ist. Vergessen Sie nicht, vorab die Dimensionen zu überprüfen. Erweitern Sie Ihre Matrixklasse außerdem um eine `print`-Methode welche die Matrix am Bildschirm ausgibt.

Aufgabe 11.6. *(Vererbung)*
Schreiben Sie eine Klasse `SquareMatrix` zur Speicherung (vollbesetzter) quadratischer Matrizen. Diese soll mit möglichst wenig Implementierungsaufwand erstellt werden. Leiten Sie daher die Klasse `SquareMatrix`

von `Matrix` ab. Was sind die Vorteile bei diesem Vorgehen? Was müssen Sie bei der Basisklasse beachten? Testen Sie Ihre neue Klasse, indem Sie zwei quadratische Matrizen anlegen, diese addieren und das Ergebnis schließlich ausgeben lassen.

Aufgabe 11.7. *(LU-Zerlegung)*
Erweitern Sie die Klasse `SquareMatrix` um die Methode `getLU`, die die normalisierte LU-Zerlegung der Matrix zurückgibt (falls diese existiert). Der Rückgabewert $R \in \mathbb{R}^{n \times n}$ sei wieder vom Typ `SquareMatrix`, wobei die beiden Dreiecksmatrizen L und U beide in R gespeichert werden sollen. Die Diagonale von L muss hierbei nicht explizit gespeichert werden. Besitzt die Matrix keine LU-Zerlegung, so werde NULL zurückgegeben. Orientieren Sie sich hierbei an den Aufgaben 7.3 und 7.5.

Aufgabe 11.8. *(Determinante)*
Die Determinante einer Matrix $A \in \mathbb{R}^{n \times n}$ kann über die normalisierte LU-Zerlegung aus der letzten Aufgabe berechnet werden. Es gilt namlich $det(A) = det(L)det(U) = det(U) = \prod_{j=0}^{n-1} u_{jj}$. Erweitern Sie die Klasse `SquareMatrix` um eine Methode `detLU`, die die Determinante über die LU-Zerlegung berechnet und zurückgibt. Die Matrix selbst soll hierbei nicht überschrieben werden. Stellen Sie außerdem sicher, dass Sie den Speicher für die LU-Zerlegung auch wieder entsprechend freigeben.

Aufgabe 11.9. *(Pointer & Referenzen)*
Worin besteht der Unterschied zwischen Pointern und Referenzen? Was sind die Vor- und Nachteile? Sind Pointer in C++ überhaupt noch relevant?

Aufgabe 11.10. Implementieren Sie ein einfaches *Tic Tac Toe* Spiel. Falls Sie Ihnen nicht bekannt sind, können Sie die Regeln unter `http://de.wikipedia.org/wiki/Tic_Tac_Toe` nachlesen. Das Spiel soll mindestens die folgenden Kriterien erfüllen:

1. Es sollen zwei Spieler gegeneinander antreten können die jeweils abwechselnd am Zug sind.
2. Es soll automatisch erkannt werden, wenn ein Spieler gewonnen hat.
3. Nach jedem Zug soll das aktuelle Spielfeld grafisch in der Konsole ausgegeben werden.
4. Kann sich am Ende keiner der Spieler durchsetzen, so soll `'unentschieden'` ausgegeben werden.

Tip: Planen Sie Ihre Datenstrukturen und den Programmverlauf bevor sie mit der tatsächlichen Implementierung beginnen.