Dirk Praetorius,
Marcus Page
Thomas Führer

Wintersemester 2012/2013
14.01.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 14

Aufgabe 14.1. (Klassendefinition)
Schreiben Sie eine Klasse `ComplexVector` zur Speicherung von komplexwertigen Vektoren. Die Real- sowie Imaginäreinträge des Vektors sollen jeweils als `vector<double>` gespeichert werden. Die Klasse soll außerdem über `get-` und `set`-Methoden verfügen um sowohl Einträge, als auch die Länge des Vektors auszulesen. Die Länge soll hierbei nicht explizit gespeichert, sondern über den Eintragsvektor ermittelt werden. Denken Sie daran, entsprechende Sicherheitsabfragen zu schreiben.

Aufgabe 14.2. (Multiple Rückgabewerte)
Erweitern Sie die Klasse `ComplexVector` um eine Methode `getMaxAbs`, die den größten Absolutbetrag $|z| = \sqrt{a^2 + b^2}$ mit $z = a + ib$ berechnet und zurückliefert. Außerdem soll der entsprechende Index und die Zahl selbst zurückgegeben werden. Sie können hierfür die Klasse `Complex` aus Serie 11 verwenden. Welche Möglichkeiten kennen Sie, um mehrere Rückgabewerte zu realisieren? Implementieren Sie beide.

Aufgabe 14.3. (Vererbung)
Ein rellwertiger Vektor ist auch ein komplexwertiger Vektor dessen Imaginäranteile gleich Null sind. Schreiben Sie eine Klasse `RealVector`, die Sie entsprechend von `ComplexVector` ableiten. Lassen Sie sich auch hier den maximalen Absolutbetrag ausgeben. Verwenden Sie hierfür die von `ComplexVector` geerbte Methode `getMaxAbs`. Was müssen Sie dabei beachten?

Aufgabe 14.4. (Untere Dreiecksmatrix)
Schreiben Sie eine Klasse `LowerTriangular`, die Sie von `SquareMatrix` ableiten. Redefinieren sie die `get-` und `set`-Methoden so, dass unnötige Nulleinträge nicht gespeichert werden müssen. Erstellen Sie ein UML Diagramm der Klassen `SquareMatrix`, `UpperTriangular` und `LowerTriangular`. Aus dem UML-Diagramm sollen neben den vollen Signaturen der einzelnen Methoden auch die Zusammenhänge zwischen den einzelnen Matrixtypen deutlich werden.

Aufgabe 14.5. (Polymorphie bei Matrizen)
Für eine obere Dreiecksmatrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

ist die LU-Zerlegung ganz einfach durch $A = I_n * A$ definiert, wobei I_n die $n \times n$ Einheitsmatrix bezeichnet. Redefinieren Sie die Methode `getLU` in `UpperTriangular` so, dass diese vereinfachte Methode verwendet wird. Schreiben Sie eine Funktion `LUfun`, der eine quadratische Matrix $A \in \mathbb{R}^{n \times n}$ übergeben wird und die dann nichts anderes tut, als die LU-Zerlegung von A am Bildschirm auszugeben. Achten Sie darauf, dass

apriori unbekannt ist, ob es sich um eine vollbesetzte oder um eine obere Dreiecksmatrix handelt. Dennoch soll jeweils der entsprechende Algorithmus verwendet werden. Wie können Sie das bewerkstelligen?

Aufgabe 14.6. (Konstante Methoden)
Was bedeutet das Schlüsselwort `const`? Erklären Sie die Unterschiede in der Zugriffskontrolle zwischen C und C++. Wieso wird hierauf soviel Wert gelegt? Definieren Sie alle Methoden aus `SquareMatrix`, `UpperTriangular` und `LowerTriangular` als `const`, für die sich das anbietet.

Aufgabe 14.7. (Inline Funktionen)
Was ist der Vorteil von `inline`-Funktionen. Wieso haben diese gerade in der objektorientierten Programmierung einen hohen Stellenwert?

Aufgabe 14.8. (Templates)
Schreiben Sie ein Funktionstemplate `myPower` zur Potenzierung für allgemeine Datentypen. Probieren Sie dieses gleich einmal aus, indem Sie es für eine quadratische Matrix vom Typ `SquareMatrix` anwenden. So soll bei Eingabe von `myPower(A, 3)` etwa A^3 für ein $A \in \mathbb{R}^{n \times n}$ berechnet werden. Für einen `double`-Wert $x \in \mathbb{R}$ soll die Eingabe `myPower(x, 3)` ebenfalls x^3 berechnen.

Aufgabe 14.9. (Exception handling)
Was sind die Vorteile der Fehlerbehandlung mittels *exceptions* im Vergleich zu einfachen Sicherheitsabfragen?

Aufgabe 14.10. (Qualitätssicherung)
Wie funktioniert Qualitätssicherung in der modernen Softwareentwicklung. Welche Arten von Testverfahren gibt es? Wo liegen die Unterschiede? Welche sind realistisch? Wie lange sollte man einen Code vor der Auslieferung testen? Was bedeutet großer Zeitdruck für Qualität, Funktionalität bzw. Kosten einer Software?