Dirk Praetorius,
Marcus Page,
Thomas Führer

Wintersemester 2012/13
31.12.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 12

Aufgabe 12.1*. *(Vererbung)* Implementieren Sie eine Klasse `Person`, welche die Datenfelder `name` und `adresse` enthält. Schreiben Sie auch die zugehörigen Zugriffsfunktionen. Leiten Sie von dieser Klasse eine Klasse `Student` ab, welche die zusätzlichen Datenfelder `matrikelnummer` und `studium` enthält. Leiten Sie von der Klasse `Person` auch eine Klasse `Arbeiter` ab. Erweitern Sie diese Klasse um die Datenfelder `gehalt` und `arbeit`. Erstellen Sie ein UML-Diagramm dieser Klassen. Überlegen Sie sich auch, welche Zugriffsspezifizierer für welche Datenelemente verwendet werden und begründen Sie dies.

Aufgabe 12.2*. *(Funktionen redefinieren)* Erstellen Sie für die Basisklasse `Person` aus Aufgabe 12.1 eine Methode `void print()`, welche den Namen und die Adresse einer Person am Bildschirm ausgibt. Redefinieren Sie diese Funktion dann jeweils für die Klassen `Student` und `Arbeiter` (Es sollen die zusätzlich definierten Datenelemente auch ausgegeben werden). Schreiben Sie dann noch ein Hauptprogramm, in welchem die `print`-Funktionen der verschiedenen Klassen getestet werden sollen.

Aufgabe 12.3*. *(Zeitmessung)* Schreiben Sie eine Klasse `Stoppuhr` welche zur Simulation einer Stoppuhr dienen soll. Die Stoppuhr bestehe dabei aus zwei Knöpfen. Wird der erste Knopf gedrückt, so soll die Zeitmessung gestartet werden. Wird dieser Knopf nochmals gedrückt wird die Zeitmessung gestoppt. Der zweite Knopf dient dazu die Zeit wieder zurückzusetzen. Schreiben Sie dazu die Methoden `pushButtonStartStop` und `pushButtonReset`. Implementieren Sie weiters eine Methode, welche die verstrichene Zeit im Format `hh:mm:ss.xx` ausgibt. (Beträgt die gemessene Zeit also zwei Minuten so soll `00:02:00.00` ausgegeben werden) Sie können diese Stoppuhr nun dazu verwenden Zeitmessungen durchzuführen. Messen Sie die Laufzeit der Funktion `minsort` aus der VO (Folie 88) für Vektoren verschiedenster Länge. Speichern Sie den Source-Code unter `stoppuhr.{h,cpp}` in das Verzeichnis `serie12`. *Hinweis:* Verwenden Sie den Datentyp `clock_t` und die Funktion `clock()` aus der Bibliothek `time.h`. Vermutlich ist es auch sinnvoll eine Variable `isRunning` vom Typ `bool` einzuführen. Bei Betätigen des ersten Knopfes wird diese Variable entweder von `false` auf `true` gesetzt oder umgekehrt.

Aufgabe 12.4*. *(LU-Zerlegung)* Wir betrachten die Klasse `Matrix` und die davon abgeleitete Klasse `SquareMatrix` aus der VO (siehe Folien 106–107). Implementieren Sie für die Klasse `SquareMatrix` die Methode `computeLU`, welche die LU-Zerlegung berechnet. Der Rückgabewert (Matrix $R \in \mathbb{R}^{n\times n}$) sei dabei wieder vom Type `SquareMatrix`, wobei die beiden Dreiecksmatrizen L und U in R gespeichert werden sollen. Die Diagonale von L muss hierbei nicht explizit gespeichert werden. (Warum?) Orientieren Sie sich für die LU-Zerlegung an Aufgabe 9.2.

Aufgabe 12.5*. *(Aufwand LU-Zerlegung)* Welchen Aufwand besitzt die LU-Zerlegung (siehe Aufgabe 9.2 bzw. 12.4)? Schreiben Sie das Ergebnis in der $\mathcal{O}$-Notation auf und erklären Sie wie sie auf das Ergebnis gekommen sind.

Aufgabe 12.6. *(Aufwand, Schwach-besetzte Matrizen)* In der VO (Folie 87) haben Sie gelernt wie man den Aufwand einer Matrix-Vektor Multiplikation bestimmt. Im Folgenden betrachten wir schwach-besetzte quadratische ($n \times n$-) Matrizen. Üblicherweise beträgt der Aufwand zur Speicherung von diesen Matrizen $\mathcal{O}(n)$. In Aufgabe 7.4 haben Sie bereits schwach-besetzte Matrizen und die Matrix-Vektor Multiplikation für schwach-besetzte Matrizen kennengelernt. Wie groß ist der Aufwand der Matrix-Vektor Multiplikation für schwach-besetzte Matrizen (in Abhängigkeit von n)? Begründen Sie ihre Antwort!

Aufgabe 12.7. *(Determinante)* Die Determinante einer Matrix $A \in R^{n \times n}$ kann über die normalisierte LU-Zerlegung aus Aufgabe 12.4 berechnet werden. Es gilt namlich $\det(A) = \det(L)\det(U) = \det(U) = \prod_{j=1}^{n} u_{jj}$. 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 12.8. *(Strukturen, Klassen)* Erklären Sie die Unterschiede zwichen Strukturen und Klassen. Worin unterscheiden sich die beiden Konzepte? Welche Gemeinsamkeiten gibt es?

Aufgabe 12.9. *(Pointer,Referenzen)* Erklären Sie das Prinzip eines Pointers und einer Referenz? Was bedeutet einen Pointer zu dereferenzieren? Welche Unterschiede gibt es zwichen Pointer und Referenzen? In welchen Situationen würden Sie eher Referenzen statt Pointer benützen und umgekehrt? Was bedeutet *Call by Reference* bzw. *Call by Value*. Wie kann man *Call by Reference* realisieren? Schreiben Sie eine Funktion, welche zwei Integervariablen vertauscht. Benutzen Sie hierzu *Call by Reference*. Kann man diese Funktion mit *Call by Value* realisieren? Warum oder warum nicht?

Aufgabe 12.10. *(Ganzzahlen)* Welche Typen kennen Sie um Ganzzahlen darzustellen? Was bedeutet das Schlüsselwort `unsigned`? Wieviel Ganzzahlen kann man mit $n \in \mathbb{N}$ Bits darstellen?

Aufgabe 12.11. *(Zahlendarstellung)* In der Vorlesung haben Sie gelernt, dass es zu jeder Zahl $x \in \mathbb{R}$

- ein Vorzeichen $\sigma \in \{\pm 1\}$,
- Ziffern $a_j \in \{0, 1\}$ und
- einen Exponenten $e \in \mathbb{Z}$

gibt, so dass

$$x = \sigma\Big(\sum_{k=1}^{\infty} a_k 2^{-k}\Big)2^e$$

gilt. Skizzieren Sie den Beweis dieser Aussage. Diese Darstellung ist im Allgemeinen nicht eindeutig. Sie kann aber in einem Gleitkommazahlensystem $\mathbb{F}(2, M, e_{\min}, e_{\max})$ eindeutig gemacht werden. Was wird zusätzlich gefordert damit diese Darstellung dann eindeutig ist? Warum? Was ist das erste implizite Bit? Welche Typen kennen Sie zur Darstellung von Gleitpunktzahlen im Rechner? Wodurch unterscheiden sich diese?

Aufgabe 12.12. *(Rekursive Funktion)* Was ist eine rekursive Funktion? Was muss man sicherstellen wenn man eine rekursive Funktion programmiert? Schreiben Sie eine Funktion, welche die Faktorielle einer Zahl $n \in \mathbb{N}$ berechnet. Implementieren Sie diese Funktion einmal mittels Rekursion und einmal mittels Schleifen. Welche Version würden Sie bevorzugen und warum?