Programmierübungen

Wintersemester 2006/2007

7. Übungsblatt
8. Dezember 2006

Abgabe bis Samstag, 16. Dezember 23:59 Uhr.

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Erarbeiten Sie Lösungsideen zu den Aufgaben möglichst in Kleingruppen. Es wird jedoch von Ihnen erwartet, dass jeder Teilnehmer eine eigene Lösung abgibt. Sollten kopierte Quelltexte abgegeben werden, so werden grundsätzlich alle Kopien mit 0 Punkten bewertet. In den Vortragsfolien der Programmierübungen oder im Skript zur Einführung in die Informatik abgedruckte Quelltexte können verwendet werden, müssen aber der Programmierrichtlinie entsprechend formatiert und kommentiert werden.

Beachten Sie die Programmierrichtlinie und kommentieren Sie Ihren Quelltext. Dokumentieren Sie unbedingt Ihre Lösungsidee in den Quelltext-Kommentaren.

`http://www.iste.uni-stuttgart.de/ps/Lehre/WS0607/inf-prokurs`

Am Montag 11.12.2006, 8:00 Uhr in V38.01 wird das Aufgabenblatt kurz vorgestellt und Sie haben Gelegenheit Fragen zu den Aufgaben zu stellen.

Aufgabe 7.1: Mitarbeiterdatenbank (15 Punkte)

Entwerfen Sie ein Programm, mit dessen Hilfe Anfragen an eine einfache Personaldatenbank gestellt werden können. Die Personaldatenbank soll von Ihrem Programm als Textdatei gelesen werden können. Sie besteht aus bis zu 100 Einträgen, jeder Eintrag besteht aus genau einer Zeile in folgendem Format: `<key>#<name>#<boss>`. Der Text `<key>` steht stellvertretend für eine Personalnummer (eine natürliche Zahl im Bereich 1 – 100), `<name>` steht stellvertretend für den Namen des Mitarbeiters und `<boss>` steht für die Personalnummer des direkten Vorgesetzten des Mitarbeiters, bzw. 0 falls der Mitarbeiter der Geschäftsleitung angehört.

Beispiel für eine Mitarbeiterdatenbank:

```
1#Bill#0
2#Alice#1
20#Bob#1
5#Karl#0
17#Thomas#2
```

Ihr Programm soll „PManager“ heißen. Es wird durch Kommandozeilen-Argumente gesteuert. Das Programm wird gestartet, führt seine Arbeit durch und beendet sich wieder. Als Personaldatenbank wird immer die Datei `personnel.txt` verwendet. Jeder Aufruf hat dabei das Format:

pmanager `command [arg [arg2]]`

Anstelle von `command` können dabei die folgenden Befehle verwendet werden, `arg` und `arg2` haben vom Befehl abhängige Bedeutung und werden nicht immer angegeben:

list Das Programm gibt eine Liste aller Mitarbeiter mit Ihren Personalnummern aus. `arg` und `arg2` werden nicht benötigt.

supervisor Gibt Name und Personalnummer des direkten Vorgesetzten zu einem bestimmten Mitarbeiter aus. `arg` muss angegeben werden und muss ein Name oder eine Personalnummer sein. `arg2` wird nicht benötigt.

peers Gibt eine Liste der Namen und Personalnummern der Kollegen eines Mitarbeiters `arg` aus. Mit Kollegen sind hier nur diejenigen Mitarbeiter gemeint, die den selben direkten Vorgesetzten haben. Beispiel mit obiger Datenbank:

```
$ pmanager peers Alice
20: Bob
```

team Gibt eine Liste aus Namen und Personalnummern der Mitarbeiter mit (möglicherweise indirektem) Vorgesetztem `arg` aus. Beispiel mit obiger Datenbank:

```
$ pmanager team Bill
 2: Alice
20: Bob
17: Thomas
```

hire Das Programm fügt der Mitarbeiter-Datenbank einen neuen Mitarbeiter hinzu. `arg` identifiziert den direkten Vorgesetzten des neuen Mitarbeiters und `arg2` ist der Name des neuen Mitarbeiters. Eine freie Personalnummer wird vom Programm automatisch gewählt. Das Programm speichert die veränderte Mitarbeiterdatenbank in die Textdatei. Es gibt eine Meldung aus, um dem Benutzer die erfolgreiche Datenbankänderung zu quittieren. Beispiel mit obiger Datenbank:

```
$ pmanager hire Bill Oskar
Hired Oskar, reporting to Bill.
```

Sollten die Eingaben des Benutzers zu Fehlern führen (z. B. es werden mehr als 100 Mitarbeiter eingestellt, in einem Befehl entspricht `arg` keinem oder mehr als einem existierenden Mitarbeiter, die Datenbank ist nicht les-/schreibbar, . . .), so sollen für den Benutzer verständliche Fehlermeldungen ausgegeben werden.

Das Programm darf davon ausgehen, dass für jeden Mitarbeiter durch wiederholtes Verfolgen des `boss`-Eintrags nach endlich vielen Schritten der Eintrag 0 gefunden wird (es existieren keine Zyklen). Diese Eigenschaft kann – muss aber nicht – durch das Programm überprüft werden. Auch falls der Benutzer mit dem Kommando `hire` einen solchen Zyklus einführt, muss das Programm diesen Benutzungsfehler nicht diagnostizieren.

Entwerfen Sie Datenstrukturen selbst und strukturieren Sie Ihr Programm in überschaubare Einheiten, erstellen Sie mindestens ein Paket.

Aufgabe 7.2: Weihnachtsmann im Wald (5 Punkte)

Auf seinen Reisen verirrt sich der Weihnachtsmann im Wald. Er schaut sich um und stellt überrascht fest, dass alle Bäume völlig regelmäßig auf einem quadratischen Raster stehen und zudem einen punktförmigen Querschnitt haben. Von seinem Standpunkt aus kann er allerdings nicht alle Bäume sehen, da weiter entfernte manchmal durch nähere verdeckt werden. Für den Bereich $x, y \in \{-4 \dots 4\}$ zeichnet er in folgende Skizze alle für ihn sichtbaren Bäume ein:

```
    -4  -3  -2  -1   0   1   2   3   4
-4      *       *       *       *
-3  *       *   *       *   *       *
-2      *       *       *       *
-1  *   *   *   *   *   *   *   *   *
 0              *   W   *
 1  *   *   *   *   *   *   *   *   *
 2      *       *       *       *
 3  *       *   *       *   *       *
 4      *       *       *       *
```

Schreiben Sie ein Programm, das den Benutzer zur Eingabe der vier ganzen Zahlen $x_{min}, x_{max}, y_{min}, y_{max}$ auffordert und daraufhin analog zu obiger Skizze für den durch diese Koordinaten bestimmten Bereich ausgibt. Der Weihnachtsmann steht immer im Ursprung des Koordinatensystems.