

Programmierübungen

Wintersemester 2007/2008

7. Übungsblatt

14. Dezember 2007

Abgabe bis Freitag, 4. Januar 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/WS0708/inf-prokurs>

Aufgabe 7.1: Doppelt verkettete Listen

(10 Punkte)

Listen-Datentypen werden üblicherweise als abstrakte Datentypen gekapselt. Das hat den offensichtlichen Vorteil, dass man den Listentyp in vielen Programmeinheiten verwenden kann und trotzdem die Listen-Operationen nur einmal implementiert werden müssen. Sollte eine Änderung an der Datenstruktur notwendig werden, so muss diese nur an einer Stelle im Programm umgesetzt werden.

Auf der Webseite zu den Programmierübungen erhalten Sie die Spezifikation des Pakets `Person_Lists`. Als Element-Typ wurde der Typ `Person` verwendet. Der Typ `Cell_Ref` stellt einen Verweis auf eine Zelle der Liste dar und dient als Listenanker. Der Typ `List_Cursor` modelliert eine Referenz auf ein bestimmtes Element der Liste.

Implementieren Sie das Paket `Person_Lists`. Beachten Sie, dass die Liste zu jedem Zeitpunkt nach Name alphabetisch sortiert sein soll. Die Implementierung einiger Unterprogramme ist bereits vorgegeben. Diese können Sie verwenden oder wahlweise durch Ihre eigene ersetzen.

Es wird erwartet, dass nicht mehr benötigter Speicher (d. h. nicht erreichbare Listenzellen) freigegeben wird. Eine Instanz von `Ada.Unchecked_Deallocation` ist im Body des Pakets bereits vorgesehen.

Hinweise:

- In einer üblichen Listenimplementierung ist der Typ `List_Cursor` ein Zeiger auf eine Zelle. Um eine logische Unterscheidung zwischen Liste und Cursor zu erreichen, wurde ein separater Typ deklariert. In einem Hauptprogramm, das das Listenpaket verwendet, ist also keine Zuweisung zwischen `Cell_Ref` und `List_Cursor` erlaubt, obwohl diese technisch möglich wäre.
- Achten Sie auf die „-- TODO:“ Kommentare. Diese zeigen an, welche Unterprogramme noch von Ihnen geschrieben werden müssen.

Aufgabe 7.2: Telefonliste

(8 Punkte)

Schreiben Sie ein Programm „Phonebook“:

1. Das Programm liest die Text-Datei `phonelist.txt`, die nach unten angegebenen Format aufgebaut ist.
2. Das Programm fragt den Benutzer nach einem Namensteil (eine Zeichenkette) und sucht alle Personen aus der Telefonliste, deren Name diese Zeichenkette enthält. Das Programm gibt eine Liste der in Frage kommenden Einträge aus.

Verwenden Sie zur Darstellung der Telefonliste das Paket `Person_Lists`.

Format der `phonelist.txt`: Die Datei enthält genau 2 Zeilen Text pro Eintrag. Die erste Zeile jedes Eintrags ist der Name einer Person, die zweite Zeile die Telefonnummer dieser Person. Die Datei enthält keinerlei weiteren Text und auch keine Leerzeilen. Kann die Datei nicht gelesen werden, so soll eine für den Benutzer verständliche Fehlermeldung ausgegeben werden, und das Programm soll abbrechen.

Beispiel:

```
Hans Maier
+497111234567
Jaques C.
+4971529876543
Max Muster
+497111232334
```

Aufgabe 7.3: Fehlersuche

(2 Punkte)

Das Programm `Crash` (das Sie auf der Webseite erhalten können) verwendet das Listenpaket aus Aufgabe 7.2. Das Programm enthält jedoch einen gravierenden Programmierfehler, der speziell im Zusammenhang mit Zeigern steht. Finden Sie den Fehler, dokumentieren Sie die Fehlerursache und korrigieren Sie das Programm. Beachten Sie, dass die Quelltext-Kommentare die Absicht des Programmierers und nicht die tatsächliche Semantik des Programms wiedergeben.

Zur Korrektur können Sie dem Paket `Person_Lists` (auch in der `person_lists.ads`) weitere Operationen hinzufügen.

Hinweise:

- Zeichnungen der tatsächlichen Zeigerziele können Ihnen helfen den tatsächlichen Programmablauf zu verstehen.
- Das konkrete Fehlerverhalten des Programms ist abhängig davon, wie Sie das Paket `Person_Lists` implementiert haben sowie von Version des Compilers und Betriebssystem. Beobachten Sie nicht nur das Verhalten des ausgeführten Programms, sondern untersuchen Sie den Quelltext.

Aufgabe 7.4: Fehlervermeidung

(bis zu 1 Zusatzpunkt)

Für diese Aufgabe kann ein Zusatzpunkt vergeben werden, jedoch kann die Punktzahl für die Bearbeitung des gesamten Übungsblatts 20 nicht übersteigen.

Der in Aufgabe 7.3 erkannte Fehler kann durch den Compiler bereits erkannt werden. Hierzu muss die Deklaration des Typs `Person_List` verändert werden. Führen Sie diese Änderung durch. Dadurch wird die Übersetzung des Originalprogramms `Crash` mit einer Fehlermeldung abbrechen. Das Programm `Phonebook` aus Aufgabe 7.2 muss aber selbstverständlich weiterhin funktionieren.

Hinweis: die Implementierung des Pakets `Person_Lists` braucht nicht verändert zu werden.

Aufgabe 7.5: Weihnachtsmann im Wald

(—)

Hinweis: Die Bearbeitung dieser Aufgabe ist freiwillig. Es können keine Punkte erworben werden.

Auf seiner jährlichen Schlittenfahrt 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. Der Weihnachtsmann steht im Ursprung des Koordinatensystems, ein `*` repräsentiert einen sichtbaren Baum:

| | | | | | | | | | |
|----|----|----|----|----|---|---|---|---|---|
| | -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 die sichtbaren Bäume für den durch diese Koordinaten bestimmten Bereich analog zu obiger Skizze ausgibt.

Frohe Weihnachten!