Programmierübungen

Wintersemester 2006/2007

6. Übungsblatt
1. Dezember 2006

Abgabe bis Samstag, 9. 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 4.12.2006, 8:00 Uhr in V38.01 wird das Aufgabenblatt kurz vorgestellt und Sie haben Gelegenheit Fragen zu den Aufgaben zu stellen.

Aufgabe 6.1: Ausnahmen (4 Punkte)

Ein Paar natürlicher Zahlen a, b kann in einer Zahl kodiert werden durch die Funktion f mit: $f(a, b) = 2^a \cdot 3^b$. Wie Sie wissen, wird in einem Ada-Programm bei Zuweisung eines Werts, der den zulässigen Wertebereich verlässt, die Ausnahme Constraint_Error erhoben. Schreiben Sie ein Programm „Encode“ mit folgendem Ablauf:

- Der Benutzer wird aufgefordert, zwei natürliche Zahlen a und b einzugeben und tut dies.
- Das Programm gibt die Kodierung $f(a, b)$ aus, falls diese Zahl ohne Überlauf berechnet werden konnte.
- Falls die Berechnung der Kodierung einen Constraint_Error zur Folge hatte, so gibt das Programm den Text „Kodierung kann nicht berechnet werden“ aus. Verwenden Sie zum Erzeugen dieser Meldung einen Exception-Handler.

Schreiben Sie ein zweites Programm „Decode“, das aus einer kodierten Zahl wieder das ursprüngliche Zahlenpaar berechnet. Geben Sie eine sinnvolle Meldung aus, falls der Benutzer eine Zahl eingibt, die nicht nach obigem Verfahren kodiert wurde (z. B. 17).

Aufgabe 6.2: Telefonliste (8 Punkte)

Schreiben Sie ein Programm „Phonebook“:

1. Das Programm liest die Text-Datei `phonelist.txt`, die nach unten angegebenem 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 Einträge in der Telefonliste einen Record mit folgender Deklaration:

```
type Person is
   record
      Name  : Ada.Strings.Unbounded.Unbounded_String;
      Phone : Ada.Strings.Unbounded.Unbounded_String;
   end record;
```

Sie dürfen die Anzahl der Telefonbucheinträge auf 100 begrenzen. Enthält die Datei `phonelist.txt` mehr als 100 Einträge, so sollte das Programm eine Fehlermeldung ausgeben, aber trotzdem starten und die ersten 100 Einträge zur Verfügung stellen. Kann die Datei nicht gelesen werden, so soll eine für den Benutzer verständliche Fehlermeldung ausgegeben werden, und das Programm soll abbrechen.

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.

Beispiel:

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

Aufgabe 6.3: Mehrdimensionale Felder (8 Punkte)

In dieser Aufgabe soll die Frage untersucht werden, wie viele Spielzüge ein Springer mindestens braucht, um von einem bestimmten Startfeld ausgehend ein beliebiges Feld eines Schachbretts zu erreichen. Der Springer hat in jedem Zug die Möglichkeit zwei Felder weit in eine beliebige Richtung (westlich, östlich, nördlich oder südlich) und ein Feld in eine Richtung die im rechten Winkel abzweigt, zu gehen (z. B. 2 Felder westlich und 1 Feld südlich).

Laden Sie von der Übungswebseite die Spezifikation des Pakets Chess herunter und implementieren Sie die Prozedur Knight_Distance.

In Werten des zwei-dimensionalen Arraytyps Chess_Board wird zu jedem Feld des Schachbretts (angegeben durch einen Buchstaben und eine Zahl, z. B. A5) gespeichert, in wieviel Zügen dieses Feld erreicht werden kann.

Schreiben Sie ein Programm, das den Benutzer zur Eingabe eines Buchstabens und einer Zahl auffordert und diese Eingabe als Startfeld des Springers interpretiert. Als Ausgabe erzeugt daraufhin das Programm eine Matrix, die für jedes Feld des Schachbretts die Anzahl der benötigten Züge angibt.