Programmierkurs I (Inf., W-Info.), Übungsblatt 11

Claus/Weicker, Wintersemester 03/04 Abgabetermin: 22.01.2004, 23:59 Uhr

Bitte beachten Sie die Hinweise von Aufgabenblatt 8 sowie die Angaben in den jeweiligen Spezifikationsdateien $*$.ads.

Bitte denken Sie daran, dass für die Vergabe eines Scheins mindestens 40 Punkte aus den letzten 4 Aufgabenblättern benötigt werden! Das letzte Blatt wird Nummer 13 sein.

Aufgabe 1: Klammerproblem (leicht) 5 Punkte

Wir betrachten ein Klammerproblem mit verschiedenen Arten von Klammern. Die Menge der korrekten Klammerfolgen sei die Sprache, die von der folgenden Grammatik erzeugt wird:

$$S \rightarrow S\,S \mid a \mid \{\,S\,\} \mid (\,S\,) \mid [\,S\,]$$

Schreiben Sie eine Funktion

```
function Teste_Klammerung (Ausdruck: String) return Boolean;
```

die genau dann `true` liefert, wenn der in der Zeichenkette enthaltene Ausdruck korrekt geklammert ist. Benutzen Sie einen Keller (d.h. eine LIFO-Liste), um sich die offenen Klammern zu merken. Geben Sie Ihre Funktion in der Datei `klammerung.adb` ab. Die dazugehörige Datei `klammerung.ads` ist auf der Programmierkursseite im Internet erhältlich.

Aufgabe 2: Median (mittel) 6 Punkte

Der *Median* einer Folge von n Zahlen ist der Wert, den man erhält, wenn man die Folge sortiert und dann das mittlere Element nimmt; besteht die Folge aus einer geraden Anzahl von Zahlen, sind zwei Werte in der Mitte. Dann sei für diese Aufgabe die erste der beiden Zahlen der gesuchte Wert.
Beispiel: Der Median der Folge 8, 4, 1, 2, 16 ist 4. Der Median der Folge 1, 2, 4, 8 ist 2.
Implementieren Sie ein Paket `median.adb`, welches die Prozedur

```
procedure Neue_Zahl (X: in Natural);
```

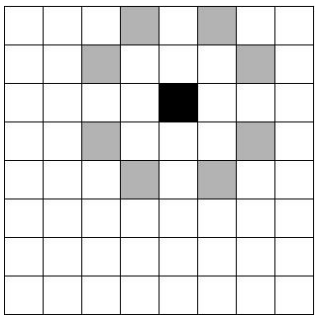
enthält, um durch iteratives Aufrufen eine beliebig lange Zahlenfolge einzulesen. Zu beliebigen Zeitpunkten kann der Median der aktuellen Zahlen durch die Prozedur

```
procedure Berechne_Median (Wert: out Natural);
```

abgefragt werden.
Speichern Sie die Zahlen in einer global im Paket deklarierten verketteten Liste. Um die Medianabfrage zu vereinfachen, ist es sinnvoll, neue Zahlen so einzufügen, dass die Liste immer sortiert ist.
Die Spezifikationsdatei `median.ads` ist auf der Internetseite derVeranstaltung erhältlich.

Aufgabe 3: Springer (schwer) 9 Punkte

Ein Schachbrett besteht aus 64 Feldern, die in acht Reihen und acht Spalten angeordnet sind. Der "Springer" ist eine Spielfigur, die sich in einem Spielzug um zwei Felder horizontal oder vertikal und um ein Feld im rechten Winkel dazu bewegen kann, sofern sie dadurch nicht aus dem Schachbrett herausspringt. In der Abbildung unten kann ein Springer, der auf dem schwarzen Feld steht, zu einem der acht grauen Felder springen.



Schreiben Sie eine Prozedur

```
procedure Zugzahl (X, Y: in Indexbereich; Zuege: out Spielfeld);
```

die für einen auf dem Feld (x, y) platzierten Springer berechnet, wieviele Spielzüge minimal benötigt werden, um jedes Feld des Spielbretts zu erreichen. Diese Werte werden in einer Matrix (Datentyp Spielfeld) zurückgegeben. In obigem Bild sind die in einem Spielzug erreichbaren Feld für die Position $(x, y) = (5, 3)$ grau markiert. Das Feld $(5, 3)$ würde den Wert 0 zu gewiesen bekommen, da kein Spielzug erforderlich ist. Die hierfür benötigten Datentypen sind wie folgt definiert:

```
subtype Indexbereich is Integer range 1..8;
type Spielfeld is array (Indexbereich, Indexbereich) of Natural;
```

Achten Sie insbesondere auch auf die Effizienz Ihrer Prozedur. Geben Sie die Implementation in der Datei `springer.adb` ab. Die dazugehörige Datei `springer.ads` ist auf der Programmierkursseite im Internet erhältlich.

Hinweise

- Zur Abgabe wird das eClaus-System verwendet:
  `http://eclaus.informatik.uni-stuttgart.de`
- Die Abgabe zu jeder Teilaufgabe besteht aus einem kompilierbaren Ada-Quelldatei. In jeder Aufgabe wird ein Dateiname vorgegeben. Verwenden Sie diesen bitte für das Hauptprogramm. Auch sind in der Aufgabe Angaben zu Ein- und Ausgabetexten sowie zur Formatierung der Ausgabe enthalten. Bitte folgen Sie diesen Angaben so genau wie möglich.
- Beachten Sie beim Programmieren bitte die folgenden Hinweise („kleine Programmierrichtlinie“).
  - Halten Sie einzelne Bestandteile überschaubar, z.B. indem Sie nur eine Anweisung pro Zeile schreiben, sowie pro Zeile höchstens 80 Zeichen, höchstens 40 Zeilen pro Prozedur/Funktion, nicht mehr als 800 Zeilen pro Datei und höchstens 5 Parameter bei Prozeduren/Funktionen benutzen.
  - Bezeichner sollen selbsterklärend und maximal 15 Zeichen lang sein. Bezeichner enthalten nur Buchstaben, den Bindestrich oder den Unterstrich.
  - Durch Einrückung um 2 Zeichen soll die logische Gliederung eines Programms verdeutlicht werden. Beispielsweise wird der Anweisungsteil einer IF-Verzweigung eingerückt, während „`end if;`“ nicht mehr eingerückt wird. Auch auf der nächsten Zeile fortgesetzte Zeilen werden um 2 Zeichen eingerückt.
  - Zu Beginn des Programms muss in Kommentaren das Konzept und die Lösungsidee des Programms ausführlich erläutert werden.
  - Auch im Programmtext sind Kommentare einzufügen, um den Code zu erläutern.
- Beachten Sie, dass jede Abgabe individuell vom jeweiligen Studierenden erstellt werden muss. Werden von den Tutoren Plagiate erkannt, d.h. exakte oder leicht modifizierte Kopien, werden für die Aufgabe keine Punkte vergeben. Falls Sie die Lösung der Aufgaben vor der Abgabe in Gruppen besprechen, achten Sie darauf, dort nur das generelle Konzept abzuklären und die Programmierung jedem selbst zu überlassen.

- Bei Fragen wenden Sie sich bitte an Ihren Tutor oder an die Übungsleitung: `Karsten.Weicker@fmi.uni-stuttgart.de` oder Tel. 7816-337
- Weitere Veranstaltungshinweise einschließlich der Übungsblätter finden Sie unter: `http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ws03-04/ada95/`