Programmierkurs I (Inf., W-Info.), Übungsblatt 4

Claus/Weicker, Wintersemester 03/04 — Abgabetermin: 20.11.2003, 23:59 Uhr

Aufgabe 1: Palindrom (leicht) 3 Punkte

Ein Palindrom ist ein Wort oder ein Satz, welches bzw. welcher vorwärts und rückwärts gelesen identisch ist, z.B. `neben` oder `stets`. Schreiben Sie ein Programm `palindrom.adb`, welches eine vom Benutzer eingegebene Zeichenkette (beliebiger Länge) darauf überprüft, ob sie diese Eigenschaft hat.

```
Geben Sie einen Text ein: neben
Es ist ein Palindrom.

Geben Sie einen Text ein: Claus
Es ist kein Palindrom.
```

Hinweis: Sie können wie bei den Aufgaben der vorherigen Übungsblätter den Datentyp `Unbounded_String` benutzen.

Aufgabe 2: Strukturiertes Programmieren (leicht) 5 Punkte

Schreiben Sie ein Programm `freitag.adb`, das alle Freitage eines Jahres ausgibt, die auf den 13. eines Monats fallen. Das Programm erfragt als Eingabe den Wochentag des letzten Tags des Vorjahres sowie die Information, ob das aktuelle Jahr ein Schaltjahr ist. Gestalten Sie Ihr Programm so, dass es bei Einhaltung aller Programmierrichtlinien möglichst kurz ist.

```
Letzter Wochentag des Vorjahres [0=Mo,..,6=So]: 2
Ist das Jahr ein Schaltjahr [j/n]: j
Freitag der 13. in dem/n Monat(en): 2, 8
```

Aufgabe 3: Einlesen einer Datei (mittel) 6 Punkte

Schreiben Sie ein Programm `dateiinfo.adb`, das einen Dateinamen erfragt, die entsprechende Datei aus dem aktuellen Verzeichnis einliest und verschiedene Informationen über diese Datei berechnet. Geben Sie die Anzahl der Zeilen, die Länge der längsten Zeile sowie die durchschnittliche Zeilenlänge aus.

```
Dateiname: abstract.txt
Anzahl der Zeilen: 11
Laenge der laengsten Zeile: 79
Durchschnittliche Zeilenlaenge: 63
```

Die Beispieldatei ist auf der Webseite der Vorlesung erhältlich.

Hinweis: Wie in der ersten Übungsstunde durch die Tutoren erläutert wurde, öffnet die Funktion `Open(fileid,In_File,filename)` eine Datei *filename* für einen lesenden Zugriff. `fileid` muss als Variable des Datentyps `File_Type` deklariert sein. Mit der Funktion `Close(fileid)` kann die Datei wieder geschlossen werden. Die Zeilen der Datei können mit der Funktion `Get_Line(fileid)` eingelesen und mit der Funktion `To_String` in einen gewöhnlichen String transformiert werden (aus dem Paket `Ada.Strings.Unbounded.Text_IO`). Mit der Funktion `End_Of_File` kann abgefragt werden, ob das Ende der Datei erreicht ist. Für die Dateiverarbeitung müssen die Pakete `Text_IO` und `Integer_Text_IO` eingebunden werden. Schlagen Sie die genaueren Spezifikationen der Funktionen gegebenenfalls im Internet nach.

Aufgabe 4: Textsuche (schwer) 6 Punkte

Schreiben Sie ein Programm `textsuche.adb`, welches zunächst zwei Zeichenketten s und t einliest. Anschließend soll überprüft werden, ob die Zeichenkette t als Teil der Zeichenkette s vorkommt.

```
Geben Sie einen Text ein: ervidevivivede
Geben Sie einen Suchtext ein: vive
Suchtext ist enthalten

Geben Sie einen Text ein: ervidevivivede
Geben Sie einen Suchtext ein: vivice
Suchtext ist nicht enthalten
```

Benutzen Sie den Datentyp `Unbounded_String` zur Speicherung der Texte. Für die Suche sollten Sie ausschließlich mit den Funktionen `Get_Line`, `Length` und `Element` auskommen.

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/`