Programmierkurs I (Inf., W-Info.), Übungsblatt 2

Claus/Weicker, Wintersemester 03/04 — Abgabetermin: 06.11.2003, 24 Uhr

Aufgabe 1: Caesarcode (leicht) — 7 Punkte

Sie haben in den Übungen zur Einführungsvorlesung den Caesar-Code kennengelernt. Dieser dient zur Verschlüsselung von Zeichenketten. Er lässt sich wie folgt beschreiben:

> Als Parameter sei eine natürliche Zahl k vorgegeben ($0 < k < 26$). Eine Zeichenkette wird kodiert, indem jeder Buchstabe durch seinen k-ten Nachfolger im Alphabet ersetzt wird. Als Nachfolger von `Z` sei `A` anzusehen.

Schreiben Sie ein Programm `caesar.adb`, welches den Parameter k und eine Zeichenkette als Eingabe nimmt und letztere entsprechend der oberen Vorschrift kodiert ausgibt. Zeichen, die keine Buchstaben sind, sollen nicht verändert werden. Bauen Sie als erstes eine Verschlüsselungstabelle

```
type Tabelle is array (Character) of Character;
```

auf, die Sie im Weiteren zur Verschlüsselung benutzen.

```
Geben Sie k ein: 16
Geben Sie den Text ein: Zbyqhkwwsobox scd dyvv.
Ergebnis: Programmieren ist toll.
```

Hinweis 1: Benutzen Sie zur Speicherung der Zeichenketten den Datentyp `Unbounded_String` für Zeichenketten unbeschränkter Länge. Hierfür werden die beiden Pakete `Ada.Strings.Unbounded` und `Ada.Strings.Unbounded.Text_IO` benötigt. Für eine Variable `u:Unbounded_String` liest die Anweisung `u:=Get_Line` eine solche Zeichenkette ein, wobei das Ergebnis die Eingabe des Benutzers bis zum ersten Zeilenumbruch enthält. Mit `Length(u)` können Sie die Länge, mit `Element(u,i)` das `i`-te Zeichen in `u` abfragen. `Replace_Element(u,i,c)` ersetzt das `i`-te Zeichen in `u` durch das Zeichen `c`.
Hinweis 2: Ggf. benötigen Sie den ASCII-Code, der sich auf den Folien 96 und 97 der Einführungsvorlesung befindet. Zu einem gegebenen Zeichen `c` finden Sie mit `character'pos(c)` seinen ASCII-Code, und mit `character'val(x)` finden Sie zu einer natürlichen Zahl `x` dasjenige Zeichen, welches den ASCII-Code `x` hat.

Aufgabe 2: Klammerfolge (leicht) — 5 Punkte

In mathematischen Ausdrücken werden zur Unterteilung gewöhnlich Klammern verwendet. Als korrekt geklammert gilt eine Zeichenkette genau dann, wenn

- zu jeder öffnenden Klammer `'('` eine schließende Klammer `')'` vorhanden ist und umgekehrt;
- in jedem solchen Klammerpaar die öffnende Klammer irgendwo links von der schließenden Klammer steht.

Schreiben Sie ein Programm `klammerung.adb`, welches überprüft, ob eine gegebene Zeichenkette korrekt geklammert ist. Alle Zeichen außer `'('` und `')'` spielen für diese Korrektheit keine Rolle.

```
Geben Sie einen Ausdruck ein: ((a-b)())
Korrekte Klammerung

Geben Sie einen Ausdruck ein: ((()))
Falsche Klammerung
```

Beachten Sie bitte Hinweis 1 der ersten Aufgabe dieses Übungsblatts für die Eingabe und Speicherung von Zeichenketten.

Aufgabe 3: Zahlendarstellung (mittel) 8 Punkte

In der Vorlesung haben Sie gelernt, wie man Zahlen zu einer beliebigen positiven oder negativen Basis darstellen kann. Schreiben Sie ein Programm `zahlen.adb`, welches zwei beliebige natürliche Zahlen z und b (wobei $b > 1$) einliest und anschließend die Darstellung von z zur Basis b ausgibt. Sie sollen die Darstellung *selbst* errechnen und *nicht* wie in Aufgabe 1 auf Blatt 1 die vorgefertigte Ausgabefunktion von Ada dazu benutzen; letztere beherrscht im übrigen nur Basen zwischen 2 und 16. In der Ausgabe geben Sie bitte jede Ziffer in der Dezimaldarstellung wie im folgenden Beispiel an.

```
Geben Sie die darzustellende Zahl ein: 840
Geben Sie die Basis ein: 13
Ergebnis: (4)(12)(8)

Geben Sie die darzustellende Zahl ein: 840
Geben Sie die Basis ein: 0
Fehlerhafte Eingabe
```

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/`