Programmierkurs I (Inf., W-Info.), Übungsblatt 3

Claus/Weicker, Wintersemester 03/04 Abgabetermin: 13.11.2003, 23:59 Uhr

Aufgabe 1: Zahlendarstellung II (mittel) 12 Punkte

In Aufgabe 3 auf Blatt 2 haben Sie ein Programm geschrieben, welches die Darstellung natürlicher Zahlen zu einer beliebigen positiven Basis b errechnet. Erweitern Sie dieses Programm zu `zahlen2.adb` auf folgende Weise: Es soll die Darstellung einer nicht-negativen reellen Zahl r zu einer Basis b mit t Nachkommastellen berechnet werden, wobei r sowie die natürlichen Zahlen $b > 1$ und t vom Benutzer vorgegeben werden. Die Ausgabe ist eine Zahl `vorkomma.nachkomma`, wobei `nachkomma` die Länge t hat. Die i-te Stelle von `nachkomma` (von links) habe die Wertigkeit b^{-i}. Da Sie nur eine begrenzte Anzahl von Nachkommastellen ausgeben, ist die ausgegebene Zahl u.U. nur eine Annährung von r. Runden Sie die letzte Ziffer daher so auf bzw. ab, dass r möglichst genau dargestellt wird. Im Gegensatz zur Vorlesung werden die Ziffern > 9 nicht durch Buchstaben sondern immer durch Zahlen des Dezimalsystems dargestellt.

```
Geben Sie die darzustellende Zahl ein: 6.28
Geben Sie die Basis ein: 5
Geben Sie die Anzahl der Nachkommastellen ein: 4
Ergebnis: (1)(1).(1)(2)(0)(0)

Geben Sie die darzustellende Zahl ein: 0.924
Geben Sie die Basis ein: 13
Geben Sie die Anzahl der Nachkommastellen ein: 4
Ergebnis: (0).(12)(0)(2)(0)
```

Hinweis: Um die geforderte Formatierung der Ausgabe einzuhalten, können Sie zur Ausgabe von ganzen Zahlen den Befehl `Put` wie folgt verwenden: In einem zweiten Argument wird die Anzahl der benötigten Zeichen spezifiziert. D.h. `put(13,4)` schreibt zwei Leerzeichen und dann die Zahl 13. Wird als zweites Argument eine 0 angegeben, wird die Zahl ohne führende Leerzeichen ausgegeben.

Aufgabe 2: Münzproblem (mittel) 8 (+ 8) Punkte

Entwickeln Sie ein Programm, welches eine möglichst minimale Anzahl Münzen für die Darstellung eines beliebigen Geldbetrags berechnet. Hierfür können bis zu 5 Münzwerte in aufsteigender Reihenfolge sowie der darzustellende Münzbetrag eingegeben werden. Mit der Eingabe '0' wird die Eingabe der Münzwerte abgebrochen. Die Ausgabe besteht aus der Anzahl der verschiedenen Münzen. Ihr Programm kann davon ausgehen, dass jeder Geldbetrag darstellbar ist.

a) Implementieren Sie zunächst einen naiven Algorithmus in der Datei `muenzen.adb`, der iterativ immer den größtmöglichen Münzwert benutzt.

```
Naechster Muenzwert: 1
Naechster Muenzwert: 7
Naechster Muenzwert: 11
Naechster Muenzwert: 0
Geldbetrag: 25
Ergebnis: 3x1 0x7 2x11

Naechster Muenzwert: 1
Naechster Muenzwert: 3
Naechster Muenzwert: 10
Naechster Muenzwert: 17
Naechster Muenzwert: 25
Geldbetrag: 49
Ergebnis: 1x1 2x3 0x10 1x17 1x25
```

Hinweis: Benutzen Sie je ein `array` um die Münzwerte und die Anzahl der Münzen zu speichern. Dadurch lässt sich das Programm sehr kompakt formulieren.

b) **Zusatzaufgabe (schwer)** Schreiben Sie ein Programm `muenzen2.adb`, das tatsächlich die minimale Anzahl der Münzen berechnet. Durchsuchen Sie hierfür alle Möglichkeiten, die Münzen zu kombinieren. Aus Laufzeitgründen sollten Sie möglichst frühzeitig Kombinationen ausschließen, die den Geldbetrag übersteigen.

```
Naechster Muenzwert: 1
Naechster Muenzwert: 7
Naechster Muenzwert: 11
Naechster Muenzwert: 0
Geldbetrag: 25
Ergebnis: 0x1 2x7 1x11
```

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/`