Dr. S. Lewandowski | **Einführung in die Informatik I (autip)** | WS 2008/09

Übungsblatt 04

Ausgabe: 12.11. | Abgabeschluss: Mittwoch, 19.11., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

Von jedem Aufgabenblatt werden maximal 20 Punkte auf den Schein angerechnet.

Es wird gebeten, bei den Freitext-Aufgaben, die Abgabe bitte NICHT als Word-Datei einzusenden. Bitte entweder per Copy-&-Paste übertragen oder als Plain-Text-Datei oder in den Formaten PDF bzw. PostScript.

Bei den Programm-Dateien fehlte überraschend häufig die Dateiendung. Bitte darauf achten, dass die Ada-95-Dateien die Dateiendung „.adb“ haben.

Kopierte Abgaben werden ab diesem Aufgabenblatt strenger bewertet. Bei offensichtlich kopierten Abgaben werden die Punkte jeweils halbiert.

Ein paar Worte noch zu den Zusatzaufgaben: Diese richten sich insbesondere an Studierende, die schon gute Programmier(vor)kenntnisse haben. Sie dienen hier als Anreiz, eigene Grenzen auszuloten. Der Inhalt der Zusatzaufgaben ist für die Klausuren nicht relevant! Konzentrieren Sie sich also zunächst auf die Lösung der eigentlichen Aufgaben.

---

1. (3 Punkte, mittel) **Aufwandsabschätzungen:** Geben Sie für folgendes Programmfragment die Anzahl der Schritte abhängig von der Eingabe `N` an (jeder Vergleich, jede Zuweisung, jeder Operator, jeder Prozedur- und Funktionsaufruf ... soll jeweils als ein Schritt gezählt werden).

```
...
  N : natural;

  function SoWhat (Maxi : natural) return natural is
    Sum : natural := 0;
  begin
    for I in 1..Maxi loop
      Sum := Sum + I;
    end loop;
    return Sum;
  end SoWhat;

begin
  Get(N);
  while N>0 loop
    Put(SoWhat(N));
    N := N-1;
  end loop;
end;
```

2. (2+2 Punkte) **Rolle Rückwärts**

   (a) (2 Punkte, leicht-mittel) Schreiben Sie eine rekursive Prozedur, die solange Zahlen einliest, bis die Zahl 0 eingeben wurde, und die eingegebenen Zahlen danach in umgekehrter Reihenfolge wieder ausgibt. Bei Eingabe der Zahlen 5 <Return>, 12 <Return>, 7 <Return>, 0 <Return> soll die Ausgabe 7 12 5 lauten. Zur Lösung dieser Aufgabe sind die bisher gelernten Sprachelemente von Ada 95 ausreichend!

   Fügen Sie Ihre Idee in den Programmkopf als Kommentarzeilen mit ein und kommentieren Sie Ihr Programm. Achten Sie auch auf sinnvolle Einrückungen.

   (b) (2 Punkte) Bei der einfachsten Lösung der Teilaufgabe (a) sind die eingebenen Zahlen nach der Ausgabe verloren. Wie müssten Datenstrukturen aussehen, um die Zahlen danach noch weiterverarbeiten zu können? Muss die Anzahl der einzugeben Zahlen dazu vorher bekannt sein? Skizzieren Sie eine mögliche Lösung. (Eine Variante werden wir bald in der Vorlesung kennenlernen, eine weitere später im Semester)

3. (4 Punkte) **Pythagoräische Zahlen-Tripel:** Der Satz von Pythagoras besagt, dass in einem rechtwinkligen Dreieck die Summe der Quadrate der Katheten gleich dem Quadrat der Hypothenuse ist. Im Allgemeinen sind in einem rechtwinkligen Dreieck nicht alle drei Kantenlängen ganzzahlig. Es gibt jedoch Beispiele, bei denen dies der Fall ist, z.B. $3^2 + 4^2 = 5^2$.

   (a) (4 Punkte, leicht) Schreiben Sie ein Programm, das eine Zahl n einliest und alle Zahlen-Tripel (a, b, c) mit $1 \leq a \leq b \leq c \leq n$ ausgibt. Für $n = 20$ soll die Ausgabe z.B. so aussehen:

```
Berechnung aller Pythagoraeischen Zahlen-Tripel (a,b,c) mit 1<=a<=b<=c<=20:
   1. Zahlen-Tripel: (3,4,5)
   2. Zahlen-Tripel: (5,12,13)
   3. Zahlen-Tripel: (6,8,10)
   4. Zahlen-Tripel: (8,15,17)
   5. Zahlen-Tripel: (9,12,15)
   6. Zahlen-Tripel: (12,16,20)
```

(b) **Zusatzaufgabe (6 Punkte, sehr schwer, nur für Tüftler)**: Der Aufwand bei Teil (a) ist bei der einfachen Lösung proportional zu n^3. Wir wollen dies hier nun wesentlich beschleunigen.
Für alle ungeraden n ist $(n, (n^2-1)/2, (n^2+1)/2)$ ein Pythagoräisches Zahlen-Tripel (d.h. auch, es gibt unendlich viele solcher Zahlen-Tripel, die nicht nur Vielfache voneinander sind), dabei beträgt die Differenz zwischen der zweiten und dritten Zahl stets genau 1.
Entwickeln Sie einen Ausdruck, der die Pythagoräischen Zahlen-Tripel mit Differenz 2 zwischen der zweiten und dritten Zahl beschreibt, verallgemeinern Sie dies (Hinweis: Die so gefundenen Zahlen sind nicht immer ganzzahlig, bei der Ausgabe müssen solche Tripel dann ausgelassen werden).
Schreiben Sie mit dem so entwickelten Ausdruck ein Programm zur Berechnung der gesuchten Zahlen-Tripel. Vergleichen Sie experimentell die Laufzeiten von der einfachen und Ihrer neuen Lösung, fügen Sie den Vergleich als Kommentarzeilen mit ein. Können Sie auch mathematisch abschätzen, wie groß der Aufwand zur Berechnung nun ist?

4. (4+2+3 Punkte) **Die Zahl** π gibt das Verhältnis zwischen Umfang und Durchmesser eines Kreises an oder aus zweidimensionaler Sicht das Verhältnis der Kreisfläche zum Quadrat des Radius. Diese letztere Eigenschaft wollen wir nun nutzen, um experimentell die Zahl $\pi/4$ anzunähern.

(a) (4 Punkte, mittel) Wählt man zufällig einen Punkt im Quadrat mit den Eckkoordinaten (0,0) und (1,1), so liegt der Punkt mit Wahrscheinlichkeit $\pi/4$ im Kreis mit Mittelpunkt im Ursprung und Radius 1.
Ermitteln Sie experimentell den Wert π (Anteil der Zufallspunkte im Viertelkreis mal 4). Untersuchen Sie die Schwankungen und Abweichungen vom exakten Wert. (Hinweis: im Paket `Ada.Numerics` ist die Konstante `pi` bereits definiert.)
Lassen Sie sich von folgendem Programm inspirieren, wie man mit Ada 95 Zufallszahlen erzeugen kann.

```
with Ada.Text_IO, Ada.Numerics.Float_Random;
use Ada.Text_IO, Ada.Numerics.Float_Random;

procedure RandDemo is
  G : generator;
  Zufall : float;
begin
  Reset (G);
  for i in 1..20 loop
    Zufall := Random (G);
    if Zufall < 0.5 then Put_Line("Kopf"); else Put_Line("Zahl"); end if;
  end loop;
end RandDemo;
```

Details finden sich im Abschnitt A.5.2 des Reference Manuals, grob gesagt ist folgendes zu tun: Ein Objekt vom Typ `Generator` deklarieren, dieses *einmal* als Parameter von `Reset` verwenden, anschließend beliebig oft als Parameter von `Random`, das Ergebnis letzterer Funktion ist dann ein „zufälliger“ Wert im Bereich zwischen 0 und 1. (Randbemerkung (zur Lösung der Aufgabe nicht nötig): Da die direkte Möglichkeit ganzzahlige Zufallszahlen in Ada 95 zu erzeugen etwas komplizierter ist, begnügen wir uns mit dieser etwas einfacheren Variante – schreiben Sie sich ggf. eine `function Random_Int(n:natural) return natural` selbst, die eine Zufallszahl zwischen 0 und `n-1` erzeugt.)
(2 Punkte, leicht) Bei manchen schlechten Zufallszahlengeneratoren erhält man bei zeitlich sehr nah hintereinander erzeugten Zufallszahlen sehr ähnliche – oder im schlimmsten Fall gar gleiche – Ergebnisse. Simulieren Sie diesen Fall, indem Sie nur für die x-Koordinate eine Zufallszahl erzeugen lassen und für die y-Koordinate denselben Wert verwenden. Welche Zahl wird mit Ihrem Programm nun angenähert? Begründen Sie Ihre Annahme und fügen dies als Kommentarzeilen mit ein.

(b) (3 Punkte, mittel) Mit Brüchen kann man die Zahl π für viele Anwendungen ausreichend gut annähern. Die am häufigsten verwendete Näherung ist 22/7.
Schreiben ein Programm, dass systematisch Brüche testet und – wenn eine bessere Näherung gefunden wurde – diese und den Abstand zur Zahl π ausgibt. Wählen Sie die Grenze für die getesteten Zähler und Nenner etwa so, dass die Laufzeit unter 15 Sekunden bleibt. Welches ist Ihre beste gefundene Näherung?
