Programmierübungen

Wintersemester 2006/2007

5. Übungsblatt
24. November 2006

Abgabe bis Samstag, 2. Dezember 23:59 Uhr.

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Erarbeiten Sie Lösungsideen zu den Aufgaben möglichst in Kleingruppen. Es wird jedoch von Ihnen erwartet, dass jeder Teilnehmer eine eigene Lösung abgibt. Sollten kopierte Quelltexte abgegeben werden, so werden grundsätzlich alle Kopien mit 0 Punkten bewertet. In den Vortragsfolien der Programmierübungen oder im Skript zur Einführung in die Informatik abgedruckte Quelltexte können verwendet werden, müssen aber der Programmierrichtlinie entsprechend formatiert und kommentiert werden.

Beachten Sie die Programmierrichtlinie und kommentieren Sie Ihren Quelltext. Dokumentieren Sie unbedingt Ihre Lösungsidee in den Quelltext-Kommentaren.

`http://www.iste.uni-stuttgart.de/ps/Lehre/WS0607/inf-prokurs`

Am Montag 27.11.2006, 8:00 Uhr in V38.01 wird das Aufgabenblatt kurz vorgestellt und Sie haben Gelegenheit Fragen zu den Aufgaben zu stellen.

Aufgabe 5.1: Berechnung von π (5 Punkte)

Die Zahl π bezeichnet die Fläche des Einheitskreises (der Kreis mit Mittelpunkt $(0, 0)$ und Radius 1). Mit Hilfe eines Zufallsexperiments kann die Zahl π geschätzt werden. Hierzu wählt ein Programm n-mal einen zufälligen Punkt innerhalb des Quadrats $Q = \{(x, y) \mid x \in [0..1),\ y \in [0..1)\}$ und zählt, wie oft dieser Punkt innerhalb des Einheitskreises liegt; das sei s mal der Fall. Aufgrund der Tatsache, dass der Flächeninhalt des Quadrats 1 ist und der Flächeninhalt des Schnitts mit dem Einheitskreis $\frac{\pi}{4}$ lässt sich π schätzen. Man erwartet bei großem n:

$$\frac{\frac{\pi}{4}}{1} \approx \frac{s}{n}$$

Schreiben Sie ein Programm, das dieses Experiment durchführt und lassen Sie es nach je 1.000 Punkten die aktuelle Näherung an π ausgeben.

Verwenden Sie den vordefinierten Subtyp Float. Funktionen zur Berechnung von Quadratwurzeln finden Sie im Paket Ada.Numerics.Elementary_Functions. Ein Pseudozufallszahlen-Gererator gibt es in Ada.Numerics.Float_Random. Schlagen Sie im Ada Reference Manual nach. Ein Link findet sich am Ende der Webseite zu den Programmierübungen.

Hinweis: Die Prozedur Ada.Numerics.Float_Random.Reset wird in jedem Programmablauf nur einmal aufgerufen, um den Generator zu initialisieren.

Aufgabe 5.2: Abstrakter Datentyp (10 Punkte)

Ein abstrakter Datentyp (ADT) ist ein Typ dessen sichtbare Eigenschaften ausschließlich durch eine Menge von Unterprogrammen definiert sind. Diese Unterprogramme werden als die Schnittstelle des ADT bezeichnet. Sie besitzen formale Parameter des

ADT, oder der Typ ihres Rückgabewerts ist der ADT. Programme, die den ADT verwenden, brauchen nur dessen Schnittstelle zu kennen, nicht jedoch seine konkrete Darstellung oder die Implementierung der Unterprogramme.

Ein ADT wird in Ada innerhalb einer Paket-Spezifikation als privater Typ deklariert. Programmeinheiten, die das Paket mit einer **with** clause einbinden, können Variablen des Typs deklarieren, Zuweisungen an diese Variablen ausführen und den Gleichheitsoperator = auf zwei Werte des Typs anwenden. Weitere Funktionalität muss durch die Unterprogramme des Pakets angeboten werden (Schnittstelle des ADT).

Auf der Webseite zu den Programmierübungen erhalten Sie die Spezifikation eines Pakets Fractions. In diesem Paket wird der ADT Fraction als privater Typ deklariert.

```
type Fraction is private;
```

Im privaten Teil der Paketspezifikation (eingeleitet durch die Zeile, in der nur das Schlüsselwort **private** steht) ist die tatsächliche Darstellung des Typs als Record angegeben. Diese Deklaration ist in der Implementierung des Pakets sichtbar, nicht jedoch in einem Hauptprogramm, welches das Paket Fractions verwendet.

```
type Fraction is
   record
      Numerator   : Integer := 0;
      Denominator : Positive := 1;
   end record;
```

Der Typ Fraction soll einen mathematischen Bruch modellieren. Die Komponente Numerator speichert den Zähler des Bruchs, Denominator den Nenner.

1. Erstellen Sie eine Implementierung des Pakets Fractions, passend zu der Paket-Spezifikation. Achten Sie in Ihrem Quelltext besonders auf Vorzeichen und vermeiden Sie soweit möglich Überläufe in Ihren Berechnungen.

2. Schreiben Sie ein Hauptprogramm „Calculator“, das Fractions verwendet, um einen einfachen Taschenrechner zu simulieren. Der Taschenrechner arbeitet folgendermaßen:

   a) Der Taschenrechner fordert die Benutzerin zur Eingabe eines ersten Bruchs auf und liest diesen Bruch ein. Dieser Bruch ist der erste Operand a.

   b) Der Taschenrechner fordert die Benutzerin zur Eingabe einer der vier Grundrechenarten oder des Gleichheitszeichens auf und liest die Eingabe $\circ$ (eines der Zeichen `+`, `-`, `*`, `/` oder `=`).

   c) Wurde das Zeichen `=` eingegeben, beendet sich das Programm.

   d) Der Taschenrechner fordert die Benutzerin zur Eingabe eines Bruchs auf und liest diesen ein. Dieser Bruch ist der zweite Operand b.

   e) Der Taschenrechner berechnet das Resultat $r := a \circ b$ und gibt es aus. Dann setzt der Taschenrechner das Resultat als ersten Operanden ein $a := r$ und fährt mit Schritt 2b fort.

Hinweise:

- Beachten Sie, dass Ada implizit den Operator = für jeden privaten Typ deklariert. Damit dieser Operator korrekt funktioniert, muss jeder Bruch normalisiert werden. D. h. jeder Bruch muss vollständig gekürzt sein, nur der Zähler eines Bruchs

darf negativ werden, der Nenner jedoch nicht, und die 0 wird immer mit Nenner 1 dargestellt.

- In dem Paket Fraction_Helpers, das ebenfalls auf der Webseite erhältlich ist, werden die aus den früheren Übungen bereits bekannten Unterprogramme LCM und GCD zur Verfügung gestellt. Sie können sie benutzen oder wahlweise durch Ihre eigene Implementierung ersetzen.
- Um im Hauptprogramm die Operatoren aus dem Paket Fractions zu verwenden, können Sie (neben den anderen Möglichkeiten) eine use type clause einsetzen:

  use type Fractions.Fraction;

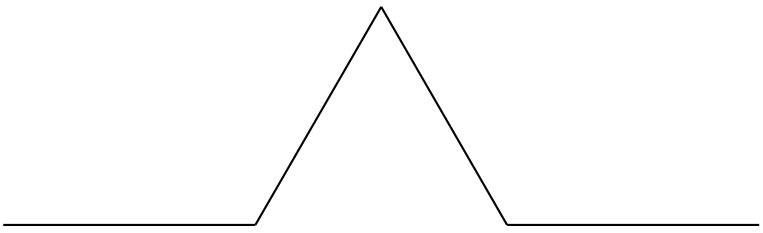
Aufgabe 5.3: Fraktale (5 Punkte)

Als Kochsche Kurve bezeichnet man eine spezielle Kurve, deren Funktion überall stetig aber nirgends differenzierbar ist. Eine Näherung an diese Kurve kann mit einem rekursiven Algorithmus wie folgt gezeichnet werden. Durch die Wahl einer bestimmten Rekursionstiefe wird die Genauigkeit der Näherung bestimmt.

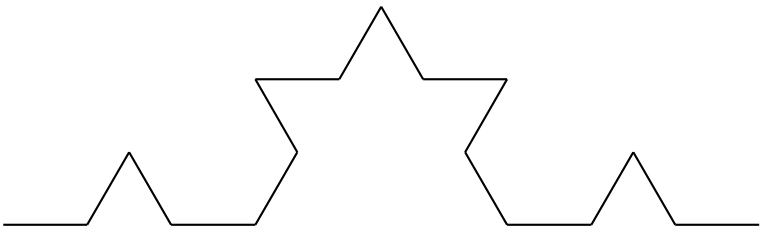
Bei Rekursionstiefe 0 ist die Näherung N_0 eine gerade Linie. Bei Rekursionstiefe $t > 0$ setzt sich die Näherung N_t aus vier gleich langen Teilen zusammen, die alle aus der Näherung N_{t-1} bestehen: zeichne N_{t-1}, drehe um $60°$ nach links, zeichne N_{t-1}, drehe $120°$ nach rechts, zeichne N_{t-1}, drehe $60°$ nach links, zeichne N_{t-1}.

Rekursionstiefe 0:

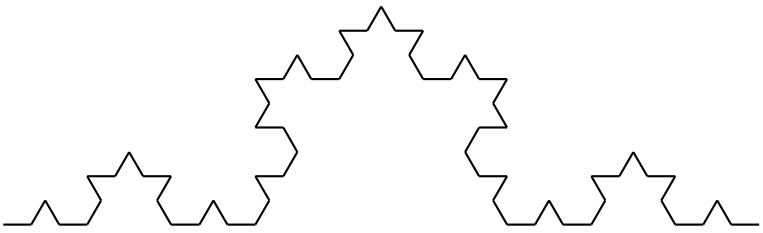
Rekursionstiefe 1:



Rekursionstiefe 2:



Rekursionstiefe 3:



Auf dem Übungsblatt 4 haben Sie gelernt mit abstrakten Datenobjekten umzugehen. Das Paket Adalogo ist ein Beispiel für eine deutlich umfangreichere Anwendung dieses Konzepts. Das Paket bietet Funktionalität um eine Schildkröte über eine Zeichenfläche zu bewegen. Die Schildkröte kann sich um ihre eigene Achse drehen und vorwärts laufen. Außerdem ist an der Schildkröte ein Stift befestigt, der auf die Zeichenfläche aufgesetzt und hochgehoben werden kann. Läuft die Schildkröte mit aufgesetztem Stift vorwärts, so zeichnet sie eine gerade Linie entlang ihres Wegs.

Schreiben Sie ein Programm „Snowflake“, gehen Sie folgendermaßen vor:

1. Lassen Sie Ihr Programm mit Hilfe von Adalogo ein gleichseitiges Dreieck zeichnen. Die Seitenlänge sollte ihrem Programm durch das 1. Kommandozeilen-Argument übergeben werden.

2. Schreiben Sie eine rekursive Prozedur, die als **in**-Parameter eine Seitenlänge (als Float-Wert) sowie eine Rekursionstiefe (als Natural-Wert) übergeben bekommt. Diese Prozedur zeichnet die oben beschriebene Näherung an die Kochsche Kurve mit Hilfe von Adalogo.

3. Ersetzen Sie die Seiten Ihres gleichseitigen Dreiecks aus Teil 1 durch Kochsche Kurven. Die anfängliche Rekursionstiefe sollte durch das 2. Kommandozeilen-Argument bestimmt werden.

4. Behandeln Sie den Fall, dass nur ein oder kein Kommandozeilen-Argument angegeben wurde. Verwenden Sie in diesem Fall die Standardwerte Rekursionstiefe = 3 und Seitenlänge = 200.0.

Hinweise:

- Verwenden Sie zum Testen kleine Rekursionstiefen (0 – 3). Rechnen Sie damit, dass Ihr Programm für größere Tiefen extrem langsam wird.

- Die Kommandozeilen-Argumente werden von Ada als Strings zur Verfügung gestellt. Verwenden Sie die Attribute Integer'Value (S) bzw. Float'Value (S) um einen String S in einen Wert des Typs Integer bzw. Float umzuwandeln.

Hinweise zu Adalogo:

- Adalogo kann von der Übungswebseite in einem zip-Archiv bezogen werden. Dieses Archiv wurde für das Betriebssystem Linux im Grundstudiumspool der Informatik erstellt und es funktioniert nur dort. Unter Windows ist die benötigte Software GtkAda nicht installiert. Sollte im Grundstudiumspool ein Problem auftreten, fragen Sie möglichst frühzeitig bei der Übungsleitung nach! Sie können die benötigten Pakete auch auf einem privaten Rechner installieren, allerdings kann die Übungsleitung hier keine Unterstützung bei Problemen anbieten.

- Als erstes muss Ihr Programm das Unterprogramm Turtle_Reset aufrufen. Danach folgen die Aufrufe zur Bewegung der Schildkröte. Lesen Sie die Dokumentation in der Paketspezifikation des Pakets Adalogo und den Userguide von Adalogo.

- Zum Schluss muss Ihr Programm das Unterprogramm Draw aufrufen. Draw öffnet auf dem Bildschirm ein Fenster und zeichnet Ihre Zeichnung. Der Benutzer kann die Zeichnung betrachten und das Fenster schließen. Erst danach wird die Programmausführung nach dem Aufruf von Draw (d. h. in Ihrem Programm) fortgesetzt.