Friedrich-Schiller-Universität Jena **Fakultät für Mathematik und Informatik** **Institut für Informatik** **Lehrstuhl für Programmiersprachen und Compiler**	**Höhere Programmierung** **SS 2001**	**Praktikumblatt 3** **Ausgabe: 30.04.2001** **Termin: 03.05.2001**

Aufgabe 1: Umfangs- und Volumenberechnung

Es soll ein Programm zur Berechnung von Umfang und Volumen einer Kugel entwickelt werden. Der Wert für den Radius wird vom Benutzer vorgegeben. Benutzen sie für die Aufgabe den Fixpunkttyp und eine Genauigkeit von drei Stellen hinter dem Komma. Wie zuvor sollen die Anwendungsbereiche der einzelnen Berechnungen und die Ausnahmebehandlung mit berücksichtigt werden.
In dieser Aufgabe geht es um das Rechnen mit Festkommazahlen. Die Typdefinition sieht wie folgt aus:

```
TYPE <typname> IS DELTA <schrittweite> RANGE <untergrenze> .. <obergrenze>;
```

oder

```
TYPE <typname> IS DELTA <schrittweite> DIGITS <gesamtstellenzahl>
    [RANGE <untergrenze> .. <obergrenze>]
```

Die <schrittweite> gibt die Schrittweite an, <untergrenze> und <obergrenze> sind die Grenzwerte des Wertebereichs, die <gesamtstellenzahl> in der zweiten Typdefinition ist die Anzahl der Stellen vor und nach dem Komma. Als Beispiel dient ein Datentyp WasserstandTyp der auf zwei Nachkommastellen genau angegeben werden soll und im Bereich 0 bis 20 liegen soll.

```
TYPE WasserstandsTyp IS DELTA 0.01 RANGE 0.0 .. 20.0;
```

Sobald Sie das Programm erstellen und versuchen eine Fixtyp-Variable als Variable vom Typ FLOAT auf dem Bildschirm auszugeben, werden sie feststellen, dass FLOAT und Festkommatyp Variablen nicht kompatibel sind. Hier soll eine einfache Art der Ausgabe angegeben werden, die auch für alle anderen skalaren Datentypen genutzt werden kann.

Umwandlung einer Fixtypzahl in einen String.

Jeder skalare Datentyp hat verschiedene hilfreiche Attribute. Die Attribute Integer'First und Integer'Last haben wir bereits kennen gelernt. Mit dem Attribut `<fixtyp>'Image (<fixtypvariable>)` wird der Wert der Variable <fixtypvariable> in einen String umgewandelt. Dabei gibt das Attribut `<fixtyp>'Aft` an wie viele Nachkommastellen umgewandelt werden. Der Rest wird einfach abgeschnitten. Dieser String kann dann mit der üblichen Textausgabe

```
Ada.Text_Io.Put_Line(<fixtyp>'Image(<fixtypvariable>));
```

am Bildschirm ausgegeben werden.

Aufgabe 2: Fallunterscheidung

Es ist die Minimumberechnung von drei Zahlen zu realisieren. Der Benutzer soll drei Zahlen vom Typ Float eingeben und das Programm gibt dann den kleinsten der drei Werte als Minimum wieder aus. Das Programm ist auf drei verschiedene Arten zu realisieren. Zum einen mit geschachtelten, mit und ohne ELSIF, und zum anderen ohne geschachtelte IF-Anweisung.

Bisher wurden einfache, sequentielle Programme gefordert. Das Programm wurde im korrekten Fall Zeile für Zeile abgearbeitet. In manchen Situationen werden jedoch Fallunterscheidungen

notwendig. Die einfachste Fallunterscheidung ist die

```
IF <bedingung> THEN <anweisungen> ELSE <anweisungen> END IF;
```

Zunächst können mit einer `IF` Anweisung nur zwei Fälle unterschieden werden. Müssen mehr Fälle unterschieden werden, dann müssen mehrere `IF` Anweisungen geschachtelt werden.

```
IF X>0 THEN
    IF X>5 THEN
          <anweisungen>
    ELSE
          <anweisungen>
    END IF;
ELSE
    IF X>-5 THEN
          <anweisungen>
    ELSE
          <anweisungen>
    END IF;
END IF;
```

oder

```
IF X>5 THEN <anweisungen>
ELSE
     IF X>0 THEN <anweisungen>
     ELSE
         IF X>-5 THEN <anweisungen>
         ELSE
              <anweisungen>
         END IF;
     END IF;
END IF;
```

Eine solche Schachtelung kann sehr unübersichtlich und komplex werden, wenn der Anweisungsteil größer wird oder ebenfalls noch weitere Fallunterscheidungen enthält. Die Grammatik von Ada erlaubt eine einfachere und effizientere Möglichkeit, eine solche Fallunterscheidung zu realisieren.

```
IF <bedingung> THEN <anweisungen>
ELSIF <bedingung> THEN <anweisungen>
ELSIF <bedingung> THEN <anweisungen>
ELSE <anweisungen>
END IF;
```

Das obige Beispiel kann dann wie folgt umgeschrieben werden:

```
IF X>5 THEN
    <anweisungen>
ELSIF X>0 THEN
    <anweisungen>
ELSIF X>-5 THEN
    <anweisungen>
ELSE
    <anweisungen>
END IF;
```