Professor Dr. F. Schweiggert   Abteilung Angewandte Informationsverarbeitung   29. Januar 2001
Johannes Mayer   Blatt 11

Allg. Informatik für WiWi (WS 2000/2001)

Abgabetermin: 1. Februar 2001

1 Rechnen mit Brüchen (30 Punkte)

Nachdem Sie ja wissen, dass das Rechnen mit REALs nicht unbedingt zum exakten Ergebnis führt, überlegen Sie sich einen Ausweg. Ja, rationale Zahlen sind in Oberon noch nicht implementiert. Das ist eine schöne Aufgabe, um auch Prozeduren kennenzulernen. Sie bekommen einen Teil eines Oberon-Programms, das ein Taschenrechner für rationale Zahlen werden soll. Implementieren Sie zunächst die Prozeduren, die noch nicht implementiert sind (wie in den Kommentaren beschrieben). Danach müssen sie noch das Hauptprogramm für den Taschenrechner schreiben. Dieser Liest Zeile für Zeile von der Standardeingabe bis zum Dateiende oder bis ein Fehler auftritt. In der ersten Zeile steht der erste Bruch, mit dem begonnen werden soll. In der darauffolgenden Zeile steht ein Operator „+“, „-“, „*“ oder „/“. Danach folgt wieder eine Zeile mit einem Bruch, der auf den bisherigen (hier: Startwert aus der ersten Zeile) addiert, von ihm subtrahiert, zu ihm multipliziert oder durch den er dividiert werden soll. Danach folgt wieder eine Zeile mit einem Operator und eine Zeile mit einem Bruch und so weiter. Jeweils nach Eingabe einer Zahl geben Sie den bisher berechneten Wert aus (als Bruch und REAL-Zahl; siehe folgendes Beispiel). Brüche werden in der Form `zaehler/nenner` eingegeben (Bsp.: `1/3`, `-5/10`). Die Berechnungen erfolgen hier in der Reihenfolge, in der die Operatoren eingegeben werden. (Es gilt nicht Punkt vor Strich!) (Hinweis: Sie können davon ausgehen, dass die Eingabe korrekt ist.)
Beispiel für eine Ausführung dieses Programms (Ausgaben sind durch „`>>`“ gekennzeichnet):

```
turing$ BruchRechner
1/2
>> 1/2 (= 5.000000D-0001)
+
2/5
>> 9/10 (= 9.000000D-0001)
-
1/6
>> 11/15 (= 7.333333D-0001)
*
3/4
```

```
>> 11/20 (= 5.500000D-0001)
/
7/2
>> 11/70 (= 1.571429D-0001)
^D
turing$
```

Der Taschenrechner hat bei obigem Beispiel folgende Berechnung durchgeführt:

$$\left(\left(\left(\frac{1}{2}+\frac{2}{5}\right)-\frac{1}{6}\right)*\frac{3}{4}\right)/\frac{7}{2}$$

Das folgende Grundgerüst von *BruchRechner.om* ist Ihnen vorgegeben:

```
MODULE BruchRechner;

  IMPORT Read, Write, Streams;

  TYPE Bruch = RECORD
                 zaehler,
                 nenner   : LONGINT;
               END;

  VAR op : CHAR;
      acc : Bruch;

  PROCEDURE GGT(a, b: LONGINT): LONGINT;
  BEGIN
    IF (a = 0) OR (b = 0) THEN
      Write.StringS(Streams.stderr, "GGT: Parameter gleich 0");
      Write.LnS(Streams.stderr);
      HALT(1);
    END;

    WHILE a # b DO
      IF a > b THEN
        a := a - b;
      ELSE
        b := b - a;
      END;
    END;

    RETURN a;
  END GGT;

  PROCEDURE KGV(a, b: LONGINT): LONGINT;
  BEGIN
    RETURN a * (b DIV GGT(a, b));
```

```
  END KGV;

  PROCEDURE Kuerze(VAR erg: Bruch);
  (** kuerzt den Bruch erg; ist der Zaehler 0, so soll
   ** der Nenner den Wert 1 bekommen *)

  PROCEDURE Addiere(VAR erg: Bruch; a, b: Bruch);
  (** addiert a und b; das GEKUERZTE Ergebnis steht in erg **)

  PROCEDURE Subtrahiere(VAR erg: Bruch; a, b: Bruch);
  (** subtrahiert b von a; das GEKUERZTE Ergebnis steht in erg **)

  PROCEDURE Multipliziere(VAR erg: Bruch; a, b: Bruch);
  (** multipliziert a und b; das GEKUERZTE Ergebnis steht in erg **)

  PROCEDURE Dividiere(VAR erg: Bruch; a, b: Bruch);
  (** dividiert a und b; das GEKUERZTE Ergebnis steht in erg **)

  PROCEDURE KonvertiereBruch(a: Bruch): REAL;
  (** konvertiert a in eine REAL-Zahl und gibt diese zurueck **)

  PROCEDURE ReadBruch(VAR erg: Bruch);
  (** liest einen Bruch der Form [0-9]+/[0-9]+ (Bsp.: "1/3", "-10/15", ...)
   ** von der Standardeingabe ein und gibt diesen in GEKUERZTER
   ** Form in erg zurueck *)

  PROCEDURE WriteBruch(a: Bruch);
  (** gibt den Bruch a in der Form [0-9]+/[0-9]+ (Bsp.: "1/3", "-3/4", ...)
   ** auf die Standardausgabe aus *)

BEGIN
  ReadBruch(acc);  (* ersten Bruch einlesen *)
  Read.Ln;         (* alles bis zum Zeilenende weglesen *)

  (** hier soll acc als Bruch und als REAL-Zahl ausgegeben werden **)

  Read.Char(op);   (* Operator einlesen *)
  WHILE ~ Streams.stdin.eof DO
    Read.Ln;       (* danach alle Zeichen bis (inkl.) Zeilenende weglesen *)

    (** hier soll zunaechst der Bruch b eingelesen werden; dann
     ** entsprechend dem Wert von op berechne:
     **    acc := acc + b oder acc := acc - b oder acc := acc * b
     **    oder acc := acc / b (!!! VORSICHT: symbolische Schreibweise !!!)
     ** bzw. gib Fehler aus, falls keiner der vier Operatoren ist
     ** danach Ausgabe von acc als Bruch UND als REAL-Zahl **)
```

```
    Read.Char(op); (* Operator einlesen *)
  END;
END BruchRechner.
```

Viel Erfolg!