Aufgabe 4

Ausdrücke lassen sich in Infix-Notation und in Postfix-Notation angeben. Bei der Postfix-Notation steht ein Operator hinter seinen beiden Operanden. Klammern und Vorrangregeln werden nicht benötigt. Die folgende Tabelle enthält vier jeweils gleichwertige Ausdrücke in Infix- und Postfix-Notation:

Infix-Notation	Postfix-Notation
3-4	34-
(5+3)*2	53+2*
5+3*2	532*+
(5+3)*(6-9)	53+69-*

Zur Ermittlung des Wertes eines Ausdrucks in Postfix-Notation wird ein Stapel verwendet. Das Vorgehen wird an dem Ausdruck 53+69-* erläutert.
Zuerst wird ein leerer Stapel angelegt. Dann wird der Ausdruck von links nach rechts Zeichen für Zeichen bearbeitet. Es werden die Zahlen 5 und 3 gestapelt. Der Operator + führt dazu, dass die Zahlen 3 und 5 entstapelt und dann addiert werden. Die Summe 8 wird gestapelt. Nun werden die Zahlen 6 und 9 gestapelt. Der Operator - führt dazu, dass die Zahlen 9 und 6 entstapelt und dann von einander subtrahiert werden. Die Differenz -3 wird gestapelt. Der Operator * führt dazu, dass die Zahlen -3 und 8 entstapelt und dann multipliziert werden. Das Produkt -24 wird gestapelt.
Da der Ausdruck nun vollständig abgearbeitet ist, wird die Zahl -24 entstapelt. Der Stapel ist nun leer. Das Ergebnis -24 wird ausgegeben.

Entwerfen und implementieren Sie ein Programm in Oberon oder Turbo Pascal, das den Wert eines eingelesenen Ausdrucks in Postfix-Notation ermittelt und ausgibt!
Testen Sie das Programm an den vier Beispielen der Tabelle! Die Tests sind zu dokumentieren!

Beachten Sie die folgenden Festlegungen:

- Zulässige Operatoren sind +, - und *. Als Operanden können 0, 1, 2, 3, 4, 5, 6, 7, 8 oder 9 angegeben werden.
- Vom Nutzer des Programms wird ein korrekter Ausdruck in Postfix-Notation eingegeben.
- Das Programm hat den ADT Liste zu importieren und zu verwenden. Die Spezifikation des ADT Liste ist im Anhang gegeben.

Entwurf:
Als Variablen werden benötigt:

- *Zahl1, Zahl2 und Ergebnis: Datentyp ganze Zahl,*
- *Eingabe: Datentyp Zeichen - Einlesen erfolgt über In.Char(Eingabe)*
- *Stapel: Datentyp adtliste.Liste*

Die Zeichenfolge wird Zeichen für Zeichen eingelesen. Falls es sich um einen Operanden handelt erfolgt die Konvertierung in eine ganze Zahl welche sodann gestapelt wird. Beim Einlesen eines Operators werden die letzten beiden Zahlen entstapelt und aus ihnen das Ergebnis dieser Operation berechnet. Vor allem bei der Subtraktion muss dabei die Reihenfolge der Operanden beachtet werden. Das Ergebnis wird erneut gestapelt.
Als Abbruchzeichen für die Eingabe wird das Doppelkreuz ‚#' verwendet. Mit dem Einlesen dieses Zeichens wird die Eingabe beendet, das letzte im Stapel verbliebene Element entstapelt und das Gesamtergebnis des Ausdrucks ausgegeben.

Stapeln: adtliste.Einfuegen oder adtliste.Anhaengen
Entstapeln: adtliste.HoleEintrag und adtliste.Loeschen

Quelltexte:

Aufgabe4.Tool

```
Edit.Open Aufgabe4.Mod

System.Free Postfix~

Postfix.berechnen 3 4 - #

Loesung:
=========
Es werden so lange einzelne Zeichen eingelesen,
bis ein „#" im Eingabestrom gefunden wird.

Leerzeichen werden bei der Eingabe zugelassen -
das erhoeht deren Uebersichtlichkeit.

Der Inhalt des Stapels wird bei jedem
verarbeiteten Zeichen angezeigt.
```

Aufgabe4.Mod

```
MODULE Postfix;

IMPORT In,Out,adtliste;

VAR
  Stapel:adtliste.Liste;
  Eingabe:CHAR;
  Zahl1,Zahl2,Ergebnis,i:INTEGER;

PROCEDURE berechnen*;

BEGIN
  In.Open;
  adtliste.Erzeugen(Stapel);
  Ergebnis:=0;
  REPEAT
   In.Char(Eingabe);
  UNTIL Eingabe#' ';
     (* Mit dieser Repeat-Schleife werden
     Leerzeichen aus der Eingabe herausgefiltert.
     *)

  WHILE Eingabe#'#' DO
    CASE Eingabe OF

    "0".."9":
    adtliste.AnhaengenElement(Stapel,ORD(Eingabe)
    -ORD('0'));
    (* Umwandlung Zeichen -> ganze Zahl *)

    |'+':
    adtliste.GeheLetztes(Stapel);
    adtliste.HoleEintrag(Stapel,Zahl2);
    adtliste.LoeschenElement(Stapel);
    adtliste.GeheLetztes(Stapel);
    adtliste.HoleEintrag(Stapel,Zahl1);
    adtliste.LoeschenElement(Stapel);
    Ergebnis:= Zahl1+Zahl2;
    adtliste.AnhaengenElement(Stapel,Ergebnis)
```

```
   |'*':
   adtliste.GeheLetztes(Stapel);
   adtliste.HoleEintrag(Stapel,Zahl2);
   adtliste.LoeschenElement(Stapel);
   adtliste.GeheLetztes(Stapel);
   adtliste.HoleEintrag(Stapel,Zahl1);
   adtliste.LoeschenElement(Stapel);
   Ergebnis:= Zahl1*Zahl2;
   adtliste.AnhaengenElement(Stapel,Ergebnis)

   |'-':
   adtliste.GeheLetztes(Stapel);
   adtliste.HoleEintrag(Stapel,Zahl2);
   adtliste.LoeschenElement(Stapel);
   adtliste.GeheLetztes(Stapel);
   adtliste.HoleEintrag(Stapel,Zahl1);
   adtliste.LoeschenElement(Stapel);
   Ergebnis:= Zahl1-Zahl2;
   adtliste.AnhaengenElement(Stapel,Ergebnis)

   END; (* Ende von Case *)

   (* Die folgenden Befehle dienen der
   Kontrolle, sie koennen weg gelassen werden.
   *)
   adtliste.GeheErstes(Stapel);
   WHILE ~adtliste.ListenEnde(Stapel) DO
      adtliste.HoleEintrag(Stapel, Zahl1);
      Out.Int(Zahl1, 6);
      adtliste.GeheNaechstes(Stapel)
   END;
   adtliste.HoleEintrag(Stapel, Zahl1);
   Out.Int(Zahl1, 6);
   Out.Ln;

   (* Ende der Befehle zur Ausgabe des Stapels
   *)

   REPEAT
    In.Char(Eingabe);
   UNTIL Eingabe#' ';
 END; (* Ende der WHILE-Schleife *)

(* die folgenden Befehle dienen der Ausgabe des
Ergebnisses und dem Leeren des Stapels *)
```

```
adtliste.HoleEintrag(Stapel,Ergebnis);
  adtliste.LoeschenElement(Stapel);
  Out.String('Das Ergebnis ist: ');
  Out.Int(Ergebnis,5);
  Out.Ln
 END berechnen;

 BEGIN
   Out.Open;
 END Postfix.
```

Protokoll:

Eingabe	***Ausgabe***
34-	-1
53+2*	16
532*+	11
53+69-*	-24

20 BE