Aufgabe 1 - Transformation arithmetischer Ausdrücke

**ATG:**

Text, letter

Description automatically generated

*Ich hoffe es ist einiger maßen leserlich :D*

**Zeitaufwand: ~**30min

**Code:**

program InfixPrefix;

const

  eofCh = Chr(0);

type

  Symbol = (

    eofSy,

    errSy,

    plusSy, minusSy, timesSy, divSy,

    leftParSy, rightParSy,

    numberSy, identSy

    );

var

  line: string;         (\* input sequence \*)

  ch: char;             (\* current character \*)

  chNr: integer;        (\* pos of ch \*)

  sy: Symbol;           (\* current symbol \*)

  numberVal: integer;   (\* numerical value if sy is a numberSy \*)

  numberValStr: string; (\* numerical value as string if sy is a numberSy \*)

  identStr: string;     (\* ident string value if sy is a identSy \*)

  success: boolean;     (\* syntax correct \*)

(\* SCANNER \*)

procedure NewChar;

begin

  if(chNr < Length(line)) then

  begin

    Inc(chNr);

    ch := line[chNr];

  end else ch := eofCh;

end;

procedure NewSy;

begin

  while(ch = ' ') do NewChar;

  case ch of

    eofCh: sy := eofSy;

    '+':

    begin sy := plusSy; NewChar; end;

    '-':

    begin sy := minusSy; NewChar; end;

    '\*':

    begin sy := timesSy; NewChar; end;

    '/':

    begin sy := divSy; NewChar; end;

    '(':

    begin sy := leftParSy; NewChar; end;

    ')':

    begin sy := rightParSy; NewChar; end;

    '0'..'9':

    begin

      sy := numberSy;

      numberval := 0;

      while((ch >= '0') and (ch <= '9')) do

      begin

        numberval := numberVal \* 10 + Ord(ch) - Ord('0');

        NewChar;

      end;

      Str(numberVal, numberValStr);

    end;

    'a'..'z', 'A'..'Z', '\_':

    begin

      sy := identSy;

      identStr := '';

      while((ch in ['a'..'z','A'..'Z','\_','0'..'9'])) do

      begin

        identStr := identStr + ch;

        NewChar;

      end;

    end;

  else

    sy := errSy;

  end;

end;

(\* Parser \*)

procedure S; forward;

procedure Expr(var e: string); forward;

procedure Term(var t: string); forward;

procedure Fact(var f: string); forward;

procedure S;

var

  e: string;

begin

  success := true;

  Expr(e); if not success then exit;

  (\* sem \*) writeln(e); (\* end sem \*)

  if(sy <> eofSy) then

  begin

    success := false;

    exit;

  end;

end;

procedure Expr(var e: string);

var

  t: string;

begin

  Term(e); if not success then exit;

  while(sy = plusSy) or (sy = minusSy) do

    case sy of

      plusSy:

      begin

        NewSy;

        Term(t); if not success then exit;

        (\* sem \*) e := '+ ' + e + ' ' + t; (\* end sem \*)

      end;

      minusSy:

      begin

        NewSy;

        Term(t); if not success then exit;

        (\* sem \*) e := '- ' + e + ' ' + t; (\* end sem \*)

      end;

    end;

end;

procedure Term(var t: string);

var

  f: string;

begin

  Fact(t); if not success then exit;

  while(sy = timesSy) or (sy = divSy) do

    case sy of

      timesSy:

      begin

        NewSy;

        Fact(f); if not success then exit;

        (\* sem \*) t := '\* ' + t + ' ' + f; (\* end sem \*)

      end;

      divSy:

      begin

        NewSy;

        Fact(f); if not success then exit;

        (\* sem \*) t := '/ ' + t + ' ' + f; (\* end sem \*)

      end;

    end;

end;

procedure Fact(var f: string);

begin

  case sy of

    numberSy:

    begin

      (\* sem \*) f := numberValStr; (\* end sem \*)

      NewSy;

    end;

    identSy:

    begin

      (\* sem \*) f := identStr; (\* end sem \*)

      NewSy;

    end;

    leftParSy:

    begin

      NewSy;

      Expr(f); if not success then exit;

      if(sy <> rightParSy) then

      begin success := false; Exit; end;

      NewSy;

    end;

  else

    success := false;

  end;

end;

(\* Main \*)

begin

  write('expr > '); readln(line);

  while(line <> '') do

  begin

    chNr := 0;

    NewChar;

    NewSy;

    S;

if not success then writeln('syntax error');

    write('expr > '); readln(line);

  end;

end.

**A lot of tests:**

**Test Case 1:**

Input: 3+4\*5/(6-2)

Expected Output: + 3 / \* 4 5 - 6 2



**Test Case 2:**

Input: (2+3)\*(4+5)

Expected Output: \* + 2 3 + 4 5



**Test Case 3:**

Input: 2\*(3+4)+5/6

Expected Output: + \* 2 + 3 4 / 5 6

A screenshot of a computer

Description automatically generated with medium confidence

**Test Case 4:**

Input: 5 + ((1 + 2) \* 4) - 3

Expected Output: - + 5 \* + 1 2 4 3



**Test Case 5:**

Input: 3 \* (4 - 2) / (5 + 1)

Expected Output: / \* 3 - 4 2 + 5 1



**Test Case 6:**

Input: 1 + 2 + 3 + 4 + 5

Expected Output: + + + + 1 2 3 4 5

A screenshot of a computer

Description automatically generated with medium confidence

**Test Case 7:**

Input: (1 + 2) + (3 + 4) + 5

Expected Output: + + + 1 2 + 3 4 5

A screenshot of a computer

Description automatically generated with low confidence

**Test Case 8:**

Input: a + b \* c - d / e

Expected Output: - + a \* b c / d e

A screenshot of a computer

Description automatically generated with medium confidence

**Test Case 9:**

Input: 32

Expected Output: 32



**Test Case 10:**

Input: a

Expected Output: a



**Test Case 11:**

Input: (3)

Expected Output: 3



**Test Case 12:**

Input: 2 \* (3 + )

Expected Output: invalid

A picture containing text

Description automatically generated

**Test Case 13:**

Input: 2 ++ 3

Expected Output: invalid



**Test Case 14:**

Input: ()

Expected Output: invalid

