Aufgabe 1 - MidiPascal

Lösungsidee:
Man nimmt den MiniPascal Compiler der Übung und fügt beim lexikalischen Analysator die neuen 5 Symbole zum Enum hinzu und beim Scanner beim Erkennen eines Idents die Zusatzoption der 5 neuen Symbole. Beim Parser in der Statement Prozedur fügt man beim Case wieder die Zusatzoptionen der neuen Symbole hinzu mit deren Grammatik und Semantischen Aktionen, die in der Angabe gegeben sind.

Zeitaufwand: ~15min

Code (Ausschnitte):

```
unit MPLex;

interface

type
  Symbol = (
    emptySy, eofSy, errSy,
    numberSy, identSy,
    semicolonSy, colonSy, commaSy, periodSy, assignSy,
    plusSy, minusSy, timesSy, divSy,
    leftParSy, rightParSy,
    programSy,
    varSy, integerSy,
    ifSy, elseSy, thenSy, whileSy, doSy,
    readSy, writeSy,
    beginSy, endSy
    );

…

procedure NewSy;
begin
  sy := emptySy;
  repeat
    while((ch = ' ') or (ch = tabCh)) do NewCh;

    syLnr := chLnr;
    syCnr := chCNr;

    case ch of
      eofCh: sy := eofSy;
      '+':
```

```
begin sy := plusSy; NewCh; end;
'-':
begin sy := minusSy; NewCh; end;
'*':
begin sy := timesSy; NewCh; end;
'/':
begin sy := divSy; NewCh; end;
'(':
begin sy := leftParSy; NewCh; end;
')':
begin sy := rightParSy; NewCh; end;
';':
begin sy := semicolonSy; NewCh; end;
':':
begin
  sy := colonSy; NewCh;
  if(ch = '=') then
  begin
    sy := assignSy; NewCh;
  end;
end;
'.':
begin sy := periodSy; NewCh; end;
',':
begin sy := commaSy; NewCh; end;
'0'..'9':
begin
  sy := numberSy;
  numberval := 0;
  while((ch >= '0') and (ch <= '9')) do
  begin
    numberval := numberVal * 10 + Ord(ch) - Ord('0');
    NewCh;
  end;
end;
'a'..'z', 'A'..'Z', '_':
begin
  identStr := '';
  while((ch in ['a'..'z','A'..'Z','_','0'..'9'])) do
  begin
    identStr := identStr + UpCase(ch);
    NewCh;
  end;
  if(identStr = 'PROGRAM') then
    sy := programSy
  else if(identStr = 'VAR') then
    sy := varSy
  else if(identStr = 'READ') then
```

```
          sy := readSy
        else if(identStr = 'WRITE') then
          sy := writeSy
        else if(identStr = 'BEGIN') then
          sy := beginSy
        else if(identStr = 'END') then
          sy := endSy
        else if(identStr = 'INTEGER') then
          sy := integerSy
        else if(identStr = 'IF') then
          sy := ifSy
        else if(identStr = 'ELSE') then
          sy := elseSy
        else if(identStr = 'THEN') then
          sy := thenSy
        else if(identStr = 'WHILE') then
          sy := whileSy
        else if(identStr = 'DO') then
          sy := doSy
        else sy := identSy;
      end;

    else sy := errSy;
    end;

  until(sy <> emptySy);
end;
```

```
unit MPC_SS;

…

procedure Stat;
var
  destId: string;
  addr, addr1, addr2: integer;
begin
  case sy of
    identSy:
    begin
      (*sem*)
      destId := identStr;
      if (not IsDecl(destId)) then SemErr('variable not declared') else
Emit2(LoadAddrOpc, AddrOf(destId));
      (*endsem*)
      NewSy;
      if (SyIsNot(assignSy)) then Exit;
      NewSy;
```

```
      Expr; if (not success) then Exit;
      (*sem*)
      if (IsDecl(destId)) then Emit1(StoreOpc);
      (*endsem*)
    end;
    readSy:
    begin
      NewSy;
      if (SyIsNot(leftParSy)) then Exit;
      NewSy;
      if (SyIsNot(identSy)) then Exit;
      (*sem*)
      if (not IsDecl(identStr)) then SemErr('variable not declared') else
Emit2(ReadOpc, AddrOf(identStr));
      (*endsem*)
      NewSy;
      if (SyIsNot(rightParSy)) then Exit;
      NewSy;
    end;
    writeSy:
    begin
      NewSy;
      if (SyIsNot(leftParSy)) then Exit;
      NewSy;
      Expr; if (not success) then Exit;
      (*sem*) Emit1(WriteOpc); (*endsem*)
      if (SyIsNot(rightParSy)) then Exit;
      NewSy;
    end;
    beginSy:
    begin
      NewSy;
      StatSeq; if not success then exit;
      if SyIsNot(endSy) then exit;
      NewSy;
    end;
    ifSy:
    begin
      NewSy;
      if SyIsNot(identSy) then exit;
      (*sem*)
      if not IsDecl(identStr) then SemErr('variable not declared');
      Emit2(LoadValOpc, AddrOf(identStr));
      Emit2(JmpZOpc, 0); (*0 as dummy address*)
      addr := CurAddr - 2;
      (*endsem*)
      NewSy;
      if SyIsNot(thenSy) then exit;
```

```
      NewSy;
      Stat; if not success then exit;
      while (sy = elseSy) do
      begin
        (*sem*)
        Emit2(JmpOpc, 0); (*0 as dummy address*)
        FixUp(addr, CurAddr);
        addr := CurAddr - 2;
        (*endsem*)
        NewSy;
        Stat; if not success then exit;
      end;
      (*sem*) FixUp(addr, CurAddr); (*endsem*)
    end;
    whileSy:
    begin
      NewSy;
      if SyIsNot(identSy) then exit;
      (*sem*)
      if not IsDecl(identStr) then SemErr('variable not declared');
      addr1 := CurAddr;
      Emit2(LoadValOpc, AddrOf(identStr));
      Emit2(JmpZOpc, 0); (*0 as dummy address*)
      addr2 := CurAddr - 2;
      (*endsem*)
      NewSy;
      if SyIsNot(doSy) then exit;
      NewSy;
      Stat; if not success then exit;
      (*sem*) Emit2(JmpOpc, addr1); FixUp(addr2, CurAddr); (*endsem*)
    end;
  end;
end;
```

Test:

Midipascal Test Programm, dass die Fakultät einer eingegebenen Zahl berechnet und ausgibt.

FAKU.mp U

UE6 > hu > FAKU.mp

```
1   PROGRAM SVP;
2     VAR
3       f, n: INTEGER;
4   BEGIN
5     Read(n);
6     f := n; n := n - 1;
7     WHILE n DO BEGIN
8       f := n * f;
9       n := n - 1;
10    END;
11    WRITE(f);
12  END.
```

```
MiniPascal source file > faku.mp
Parsing started ...
file compiled successfully

code interpretation started ...
var@1 > 5
120
... code interpretation ended
```

Die Fakultät von 5 ist 120.