

PM

Prüfungs-Musteraufgabe

PM.1 Parser mit rekursivem Abstieg

Zur Sprache, deren Syntax durch die EBNF-Regeln

```
Expr    = [ "+" | "-" ] Term { ( "+" | "-" ) Term }.  
Term    = Factor { ( "*" | "/" ) Factor }.  
Factor  = ident | "(" Expr ")".
```

gegeben ist, ist ein Parser mit rekursivem Abstieg zu entwerfen. Stellen Sie die **Aufrufbeziehungen** der benötigten Prozeduren in einem Diagramm dar und skizzieren Sie die **Algorithmen** der Prozeduren mit den unten angegebenen Vereinbarungen!

Entwurfsregeln: Siehe Hug: Module, Klassen, Verträge, S. 270f.

*Prozeduren: Eine Startprozedur *Analyze* und zu jeder EBNF-Regel *X* eine Prozedur namens *ReadX*.*

Aufrufbeziehungen



Zu benutzende Abfragen und Aktionen

symbol	Abfrage: zuletzt gelesenes Symbol mit möglichen Werten plus, minus, times, divide, leftParenthesis, rightParenthesis, ident, illegal, endOfInput
next	Aktion: liest nächstes Symbol und weist symbol den passenden Wert zu
first (E)	Abfrage: Menge der Startsymbole des EBNF-Ausdrucks E
error	Aktion: meldet Syntaxfehler

Algorithmen

```
Analyze:  
  next;  
  IF symbol # endOfInput THEN  
    ReadExpr  
  END  
  
ReadExpr:  
  IF symbol IN {plus, minus} THEN  
    next  
  END  
  ReadTerm  
  WHILE symbol IN {plus, minus} DO  
    next  
    ReadTerm  
  END
```

ReadTerm:

```
ReadFactor  
WHILE symbol IN {times, divide} DO  
    next  
    ReadFactor  
END
```

ReadFactor:

```
IF symbol = ident THEN  
    next  
ELSIF symbol = leftParenthesis THEN  
    next  
    ReadExpr  
    IF symbol = rightParenthesis THEN  
        next  
    ELSE  
        error  
    END  
ELSE  
    error  
END
```