Context-Free Grammar for Mini Language

Program Structure

program → declaration_list statement_list

Declarations

declaration_list → declaration declaration_list | ε

declaration → id_list ":" type ";"

id_list → IDENTIFIER ("," IDENTIFIER)*

type → "integer" | "real"

Statements

statement_list → statement statement_list | ε

statement → assignment ";"

assignment → IDENTIFIER ":=" expression

Expressions (Operator Precedence)

expression → term (("+" | "-") term) *

term → factor (("*" | "/") factor)*

factor → base ("^" base)*

base → "(" expression ")" | IDENTIFIER | NUMBER

Terminal Symbols

NUMBER → INTEGER | REAL

INTEGER → [0-9]+

REAL → [0-9]+\.[0-9]+

IDENTIFIER → [a-zA-Z_][a-zA-Z0-9_]*

Operator Precedence (Highest to Lowest)

Parentheses: ( )

Exponentiation: ^ (right-associative)

Multiplication/Division: * / (left-associative)

Addition/Subtraction: + - (left-associative)

Key Features

LL(1) Grammar - Suitable for recursive descent parsing

No Left Recursion - Eliminated for top-down parsing

Precedence Climbing - Clear operator hierarchy

Error Recovery - Built-in error detection points

Parse Example

```
x, y: integer;
result: real;
result := (x + y) * 2 ^ 3;
```