Analisador Sintático - Gramática LL(1)

1. Gramática

S -> EXPR

EXPR -> (OPERAND OPERAND OPERATOR)

| (if EXPR then EXPR else EXPR)

| (for INT EXPR)

| (INT RES)

| (NUM MEM)

| (MEM)

| NUM

OPERAND -> EXPR | NUM | MEM

OPERATOR -> + | - | * | / | % | ^ | | | < | > | == | != | <= | >=

NUM -> INT | REAL

INT -> [0-9]+

REAL -> [0-9]+.[0-9]+ | [0-9]+e[+-]?[0-9]+

2. Conjuntos FIRST

FIRST(S) = { (, INT, REAL }

FIRST(EXPR) = { (, INT, REAL }

FIRST(OPERAND) = { (, INT, REAL, MEM }

FIRST(OPERATOR) = { +, -, *, /, %, ^, |, <, >, ==, !=, <=, >= }

FIRST(NUM) = { INT, REAL }

3. Conjuntos FOLLOW

FOLLOW(S) = { $ }

FOLLOW(EXPR) = {), then, else, $, OPERATOR }

FOLLOW(OPERAND) = { (, INT, REAL, MEM, OPERATOR }

FOLLOW(OPERATOR) = {) }

FOLLOW(NUM) = {), MEM, RES, OPERATOR }

4. Tabela de Derivação LL(1)

M[S, (] = EXPR

M[S, INT] = EXPR

M[S, REAL] = EXPR

M[EXPR, (] = (PARSE_PAREN_EXPR

M[EXPR, INT] = NUM

M[EXPR, REAL] = NUM

M[PARSE_PAREN_EXPR, if] = if EXPR then EXPR else EXPR)

M[PARSE_PAREN_EXPR, for] = for INT EXPR)

M[PARSE_PAREN_EXPR, INT] = CHECK_SECOND

M[PARSE_PAREN_EXPR, REAL] = CHECK_SECOND

M[PARSE_PAREN_EXPR, MEM] = CHECK_TYPE

M[PARSE_PAREN_EXPR, OPERAND] = OPERAND OPERAND OPERATOR)

M[CHECK_SECOND, RES] = INT RES)

M[CHECK_SECOND, MEM] = NUM MEM)

M[CHECK_SECOND, default] = OPERAND OPERAND OPERATOR)

M[CHECK_TYPE,)] = MEM)

M[CHECK_TYPE, default] = OPERAND OPERAND OPERATOR)

M[OPERAND, (] = EXPR

M[OPERAND, INT] = NUM

M[OPERAND, REAL] = NUM

M[OPERAND, MEM] = MEM

M[NUM, INT] = INT

M[NUM, REAL] = REAL

5. Notas sobre a gramática

Esta gramática implementa a Notação Polonesa Reversa (RPN) onde:

1. As operações binárias seguem o formato: (OPERAND OPERAND OPERATOR)
2. Suporta expressões condicionais: (if CONDIÇÃO then EXPR_VERDADEIRO else EXPR_FALSO)
3. Suporta loops for: (for CONTADOR EXPRESSÃO)
4. Implementa comandos especiais:
    - (N RES) - Recupera o resultado de N linhas anteriores
    - (V MEM) - Armazena um valor V na memória
    - (MEM) - Recupera o valor armazenado na memória

A gramática está em conformidade com os requisitos da atividade avaliativa e suporta todas as operações necessárias:

- Operações aritméticas: +, -, *, /, %, ^, |
- Operações de comparação: <, >, ==, !=, <=, >=
- Estruturas de controle: if-then-else, for