Anhembi Morumbi

A3 - Teoria da computação e compiladores

Integrantes:

Abraão Rocha de Souza – RA: 125111357411

Henrique Barbosa - RA: 125111368512

Kaique Gino Santiago – RA: 125111377775

Luiz Henrique Rodrigues de Brito – RA: 125111344683

Marcus Gustavo Oliveira Sigrist Romão – RA: 125111372918

São Paulo

2024

Sumário

[Tokens 3](#_Toc168509005)

[Regra sintática 4](#_Toc168509006)

[Exemplos 9](#_Toc168509007)

[Link github 10](#_Toc168509008)

# Tokens

Nesse projeto foram utilizados tokens aritméticos, relacionais, lógicos, atribuição, delimitadores, separadores, espaçador para espaços, quebras de linhas, tabulação e Keywords para as palavras reservadas, para o token espaçador foi utilizado o skip para que quando o lexer encontrar esse token ele seja descartado na análise léxica, procurando pelo próximo token que será passado ao parser.

lexer grammar Token;

INT: 'int';

DOUBLE: 'double';

STRING: 'string';

IF: 'if';

ELSE: 'else';

FOR: 'for';

WHILE: 'while';

PRINTF: 'printf';

SCANF: 'scanf';

IGUAL: '=';

PONTOVIRGULA: ';';

LPAREN: '(';

RPAREN: ')';

LBRACE: '{';

RBRACE: '}';

VIRGULA: ',';

MAIS: '+';

MENOS: '-';

MULT: '\*';

DIVISAO: '/';

E\_COMERCIAL: '&';

OP\_RELACIONAL: '>'|'<'|'>='|'<='|'!='|'==';

OP\_LOGICO: '&&' | '||';

LETRAS: [a-zA-Z\_][a-zA-Z0-9\_]\*;

NUMEROS: [0-9]+;

INCREMENTO: '++';

DECREMENTO: '--';

STRING\_LITERAL: '"' .\*? '"';

WS: [ \t\r\n]+ -> skip;

NUMEROS\_DECIMAL: [0-9]+('.'[0-9]+)?;

# Regra sintática

Foi implementado regras para a estrutura condicional if, regras para estrutura de repetição utilizando o for e o while, regras para expressões matemáticas básicas, definições dos três tipos de variáveis que são string, int e double, além dos três tipos de variáveis foi implementado também as atribuições, porém pela falta do compilador não possível identificar se a variável foi declarada antes de realizar uma atribuição.

* programa: Define que um programa é uma sequência de zero ou mais declarações.
* declaracoes: Onde é feita a especificação dos diferentes tipos de declarações e estruturas que podem aparecer no programa, podendo ser declaração de variáveis, atribuição de valores, interações, estrutura de entrada e saída, estrutura condicional e estrutura de repetição.
* expressao: Define uma expressão que pode ser uma expressão matemática, relacional utilizando recursividade permitindo criar expressões mais complexas.
* expressaoMatematica: Define uma expressão matemática, envolvendo operadores de matemática básica, usando recursividade para poder construir expressões matemáticas complexas de forma aninhada e com uso de parênteses para que a gramática respeite a precedência das operações e a associatividade.
* expressaoRelacional: Define uma expressão relacional, usando recursividade para construir expressões relacionais mais complexas
* expressaoLogico: Define uma expressão lógica, usando recursividade para construir expressões lógicas mais complexas
* As expressões matemáticas, relacional e logico foram feitas fatoração evitando problemas ocasionados pela recursividade a esquerda.

grammar gramatica;

import Token;

// Definição da gramática

programa: declaracoes\*;

declaracoes:

declaracaoVariavel

| estruturaIf

| estruturaWhile

| estruturaFor

| estruturaPrintf

| estruturaScanf

| atribuicaoVariavel

| interacao

| interacaoString;

declaracaoVariavelDouble: DOUBLE LETRAS (IGUAL (NUMEROS\_DECIMAL|expressaoMatematica))? PONTOVIRGULA;

declaracaoVariavelInt: INT LETRAS (IGUAL (NUMEROS|expressaoMatematica))? PONTOVIRGULA;

declaracaoVariavelString: STRING LETRAS (IGUAL STRING\_LITERAL)? PONTOVIRGULA;

declaracaoVariavel: declaracaoVariavelDouble | declaracaoVariavelInt | declaracaoVariavelString;

estruturaIf: IF LPAREN expressao RPAREN LBRACE declaracoes+ RBRACE (ELSE LBRACE declaracoes+ RBRACE)?;

estruturaWhile: WHILE LPAREN expressao RPAREN LBRACE declaracoes+ RBRACE;

atribuicaoFor: (INT|DOUBLE|STRING)? LETRAS (IGUAL (NUMEROS | NUMEROS\_DECIMAL| STRING\_LITERAL))?;

estruturaFor:

FOR LPAREN (atribuicaoFor|LETRAS) PONTOVIRGULA expressao PONTOVIRGULA (interacao|interacaoString) RPAREN LBRACE declaracoes+ RBRACE;

leituraVar:E\_COMERCIAL LETRAS;

estruturaPrintf:PRINTF LPAREN expressao (VIRGULA LETRAS)? RPAREN PONTOVIRGULA;

estruturaScanf:SCANF LPAREN expressao (VIRGULA leituraVar)? RPAREN PONTOVIRGULA;

atribuicaoVariavel: LETRAS IGUAL expressao PONTOVIRGULA;

interacao: LETRAS (INCREMENTO|DECREMENTO|((MAIS|MENOS)IGUAL NUMEROS)) PONTOVIRGULA?;

interacaoString: LETRAS ((MAIS|MENOS)IGUAL STRING\_LITERAL|LETRAS) PONTOVIRGULA?;

expressao:

NUMEROS

| NUMEROS\_DECIMAL

| STRING\_LITERAL

| LETRAS

| expressaoMatematica

| relacional

| logico

| LPAREN expressao RPAREN

;

relacional: expressaoRelacional (OP\_RELACIONAL expressaoRelacional)+;

logico: expressaoLogico (OP\_LOGICO expressaoLogico)+;

expressaoRelacional:

| NUMEROS

| NUMEROS\_DECIMAL

| STRING\_LITERAL

| LETRAS

| LPAREN expressaoRelacional RPAREN

;

expressaoLogico:

expressaoMatematica

| relacional

| NUMEROS

| NUMEROS\_DECIMAL

| LETRAS

| LPAREN expressaoLogico RPAREN

;

expressaoMatematica:

term ((MAIS|MENOS) term)\*;

term:

fator ((MULT|DIVISAO) fator)\*;

fator:

NUMEROS

|NUMEROS\_DECIMAL

| LPAREN expressaoMatematica RPAREN;

# Exemplos

Exemplo 1: operações matemáticas

Texto

Descrição gerada automaticamente

Exemplo 2: Estrutura while

Texto

Descrição gerada automaticamente

Exemplo 3: Estrutura de entrada e saída.

Texto

Descrição gerada automaticamente

Exemplo 4: Estrutura for e condicional (if e else)

Texto

Descrição gerada automaticamente

# Link github

<https://github.com/AbraaoRSouza/A3_Compiladores.git>