CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO

IMPLEMENTAÇÃO COMPILADOR

ANALISADOR SINTÁTICO

AUTOR: ALAN FERREIRA LEITE SANTOS
MARCOS JUNIO DA SILVA XAVIER
SAMUEL FILIPE DOS SANTOS
ORIENTADOR: KECIA ALINE MARQUES FERREIRA

2 de abril de 2021, BELO HORIZONTE

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO

Alan Ferreira Leite Santos

Marcos Junio da Silva Xavier
Samuel Filipe dos Santos

IMPLEMENTAÇÃO COMPILADOR PARTE II - ANALISADOR SINTÁTICO

Trabalho realizado em consonância com os princípios teóricos aprendidos durante o semestre a fim de criar, por etapas, um compilador para uma determinada linguagem proposta pela orientadora.
Orientador: Kecia Aline Marques Ferreira

2 de abril de 2021, Belo Horizonte

RESUMO

Na disciplina de Compiladores, ofertada pelo CEFET-MG, iremos construir um compilador ao longo do semestre, este, dividido em etapas. Na primeira etapa, entregamos a implementação do analisador léxico e a Tabela de Símbolos em Java. Nesta segunda etapa, estamos entregando a analise do programa levando em consideração a derivação da linguagem.Tais conhecimentos obtidos com a disciplina orientados pela Dr Kecia Ferreira, para que seja possível, utilizamos como base os conceitos abordados pelo Aho, em seu livro : ”Compiladores Princípios ,Técnicas e Ferramentas”

Palavras Chaves: java,compilador

ABSTRACT

In the course of Compilers, offered by CEFET-MG, we will build a compiler throughout the semester, this one, divided into stages. In the first step, we delivered the implementation of the lexical analyzer and the Symbol Table in Java. In this second stage, we are delivering the analysis of the program taking into account the language derivation. Such knowledge obtained with the discipline guided by Dr Kecia Ferreira, to make it possible, we use as a basis the concepts addressed by Aho, in his book: ”Compilers Principles, Techniques and Tools ”

Key-words: java,compiler

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS

$\backslash n$	Quebra de Linha
(	Abertura de Parenteses
)	Fechamento de parenteses
$*$	Operador de Multiplicação
$+$	Operador de Adição
$-$	Operador de Subtração
$/$	Operador de Divisão
$:=$	Operador de Atribuição
;	Indicador de Fim de Linha
$<$	Menor que
$<=$	Menor ou igual a
$<>$	Diferente
$>$	Maior que
$>=$	Maior ou igual a
%	Representa o inicio de um comentário

Sumário

1. INTRODUÇÃO

Um compilador é um programa de computador (ou um grupo de programas) que, a partir de um código fonte escrito em uma linguagem compilada, cria um programa semanticamente equivalente, porém escrito em outra linguagem, código objeto. Classicamente, um compilador traduz um programa de uma linguagem textual facilmente entendida por um ser humano para uma linguagem de máquina , específica para um processador e sistema operacional. Atualmente, porém, são comuns compiladores que geram código para uma máquina virtual que é, depois, interpretada por um interpretador. Ele é chamado compilador por razões históricas; nos primeiros anos da programação automática, existiam programas que percorriam bibliotecas de sub-rotinas e as reunia, ou compilava, as sub-rotinas necessárias para executar uma determinada tarefa.

O nome ”compilador”é usado principalmente para os programas que traduzem o código fonte de uma linguagem de programação de alto nível para uma linguagem de programação de baixo nível (por exemplo, Assembly ou código de máquina). Contudo alguns autores citam exemplos de compiladores que traduzem para linguagens de alto nível como C. Para alguns autores um programa que faz uma tradução entre linguagens de alto nível é normalmente chamado um tradutor, filtro ou conversor de linguagem. Um programa que traduz uma linguagem de programação de baixo nível para uma linguagem de programação de alto nível é um descompilador. Um programa que faz uma tradução entre uma linguagem de montagem e o código de máquina é denominado montador (assembler). Um programa que faz uma tradução entre o código de máquina e uma linguagem de montagem é denominado desmontador (disassembler). Se o programa compilado pode ser executado em um computador cuja CPU ou sistema operacional é diferente daquele em que o compilador é executado, o compilador é conhecido como um compilador cruzado.

2. PALAVRAS RESERVADAS

Para que o compilador seja executado de forma correta, existem algumas regrinhas a serem seguidas, uma delas é a definição de palavras reservadas, essas, são palavras que não devem ser utilizadas em nenhum outro momento a não ser o que foi pre-determinado pelos programadores da linguagem. Como por exemplo: No caso deste compilador, existe a palavra stop, a qual significa que quando o compilador ler esse token, significa que o programa chegou ao final, logo, em nenhum outro local esse token deve ter outro significado a não ser o significado que já foi determinado. Não se pode atribuir uma outra função a uma palavra reservada.

Tabela 1 – Relação das Palavras Reservadas da Linguagem

Palavra Reservada	Significado
init	Inicio do Programa
stop	Indica a finalização do programa
is	Usado para determinador o tipo de uma variável
integer	Tipo Inteiro para variável
string	Determina a variável como tipo string
real	Determina a variável como tipo Real
if	Determina o inicio de um bloco de código sob uma condicional
begin	Deternia o inicio de uma condição do if
end	Determina o fim da condição do if
else	Determina a contraposição da condição do if
end else	Indica o Fim da condição do Else if
do	Determina a entrada de um laço
while	determina a condição para o laço Do
read	Prepara para ler e reservar a próxima variável a ser lida
write	Determina que ira imprimir na tela a literal que virá a seguir
not	Determina a negação do valor booleano de uma expressão
or	Determina a soma binaria entre dois valores
and	Determina a multiplicação binaria entre dois valores

Fonte: Documentação Fornecida pela Orientadora o qual pode ser consultado no diretório do projeto

3. MELHORIAS GRADATIVAS

Para iniciar o desenvolvimento dessa segunda parte do trabalho que faz parte do desenvolvimento do compilador, tivemos que consertar erros que foram relatados na parte anterior (Analisador Léxico) pela nossa orientadora Kecia. Foram pontuados os seguintes erros:

- O resultado do Teste 1 mostrado não corresponde a ele. -0,25.
- Teste 4: não reconheceu o erro de comentário não fechado. -0,25
- Teste 2: ”_valor”tem que ser reportado como erro. -0,25
- Faltou o Teste 5. -0,25

Esses erros passaram desapercebidos e foram corrigidos para essa entrega do trabalho.

4. FORMA DE UTILIZAÇÃO

Para que seja possível a utilização deste programa, basta entrar no terminal e navegar até o diretório: ”TP_Compilador/src/”, e então, digitar o seguinte comando :

```
javac Main.java
java Main
```

Listing 4.1: Entrada terminal

Acima podemos ver duas linhas de comando, a primeira serve para que possa compilar a classe Main.java e então, é criado o executável e assim, podemos acessar a classe a partir da linha seguinte, java Main .
Definimos qual arquivo de teste que o programa deve compilar dentro da classe Main, no seguinte trecho de código:

```
public class Main {
  public static void main(String[] args) {
    ArrayList<Token> tokens = new ArrayList<Token> ();
    Lexer L = null;
    int line = -5;
    try {
      L = new Lexer("codigos_teste/corretos/Teste1.txt");
      L.adicionapalavras();//Inicia adicionando palavras reservadas
      System.out.println("**** Tokens lidos ****");
      // Apenas para entrar no laço
      Token T = new Token(0, line);
      while (T.tag != Tag.EOF) {
        try {
          T = L.scan();
          if(T.tag == Tag.EOF)
            break;
          T.imprimeToken(T);
          tokens.add(T);
          line = T.line;
        } catch (InvalidTokenException | IOException e) {
          System.out.println(e.getMessage());
          try {
            L.readch();
          } catch (IOException e1) {
            e1.printStackTrace();
          }
        }
      }
      line++;
      tokens.add(new Token(Tag.EOF, line));
      //L.imprimirTabela();
```

```
32        Parser P = new Parser(tokens);
33        System.out.println("\n\n\n**** Inicio Parser ****");
34        P.init();
35      } catch (FileNotFoundException e) {
36        e.printStackTrace();
37      }
38    }
39  }
40
```

Listing 4.2: Indicando arquivo para teste

Na linha 7 (sete) do trecho do código destacado acima, mostra como determinamos qual programa nosso compilador irá testar. Veja que criamos uma pasta chamada testes para que seja incluídos somente os arquivos de teste do programa. Foram criados 10 arquivos de teste.

5. DOCUMENTAÇÃO

O Analisador Sintático tem a função de solicitar ao Analisador léxico o próximo Tokens desde o inicio do programa ate o final, e apos a cada solicitação, verificar se o Token lido esta em consonância com o que foi determinado pela gramatica do programa.
O Projeto foi desenvolvido em conjunto com as plataformas Visual Studio Code e Apache Netbeans, a principio todo o código seria criado somente ”a mão”criando as pastas e arquivos manualmente, porém, devido a algumas facilidades que o Netbeans oferece, o projeto foi alterado em alguns parâmetros para que seja possível abrir no Netbeans. Uma dessas facilidades é a opção de geração da Documentação do projeto, assim o fizemos e esta documentação gerada automaticamente pelo Netbeans se encontra no diretório: ”/TP_Compilador/Projeto/dist/javadoc/”. Para esta implementação foi acatado o modelo abordado pelo autor do livro base da disciplina: Aho
.

5.1. TABELA FRIST-FOLLOW E LL(1)

Para evitar o retrocesso métodos baseados em tabela são utilizados. Para auxiliar na construção das tabelas sintáticas deve serutilizado as funções first(α) e follow(A)

5.1.1. Frist

Defina First(α), onde α é qualquer cadeia de símbolos da gramática, como sendo o conjunto de símbolos terminais que iniciam as cadeias derivadas de α. Se α → * ε, então ε está no First(α)

5.1.2. Calculando Frist(X)

Para calcular o First(X) para todo símbolo X da gramática, aplique as seguintes regras até que não haja mais terminais ou e que possam ser acrescentados a algum dos conjuntos First.

- Se X é um símbolo terminal, então First(X)= X
- Se X é um símbolo não-terminal e X → Y 1 Y 2 ..Y k é uma produção para k ≥ 1, então acrescente a ao First(X) se, para algum i, a estiver em First(Yı), e ε estiver em todos os First(Y),...,First(Yı-1); ou seja,Y ...Yı-1 → ε. Se ε está em First(Y j) para todo j=1, 2, …, k então ε está em First(X).
- Se X → e é uma produção de X então ε está em First(X).

5.2. FOLLOW(A)

Defina Follow(A), para o não-termianl A, como sendo o conjunto de terminais a que podem aparecer imediatamente à direita de A em uma forma sentencial; ou seja, o conjunto de terminais

a tais que exista uma derivação na forma S → αAaβ, para algum α, β.

5.2.1. Calculando Follow(A)

Para calcular o Follow(A) para todos os nãoterminais da gramática aplique as seguintes regras até que nada mais possa ser acresncentado a nenhum dos conjuntos Follow

- Coloque $ (símbolo de final de cadeia) em Follow(S), onde S é o símbolo inicial da gramática.
- Se houver uma produção A → αBβ, então tudo no First(β) exceto ε está em Follow(B).
- Se houver uma produção A → αB, ou uma produção A → αBβ onde Firts(β) contém ε , então inclua Follow(A) ao Follow(B).

5.3. LL(1)

Análise LL(1)

- Conceitualmente, o analisador LL(1) constrói uma derivação mais à esquerda para o programa, partindo do símbolo inicial
- A cada passo da derivação, o prefixo de terminais da forma sentencial tem que casar com um prefixo da entrada
- Caso exista mais de uma regra para o não-terminal que vai gerar o próximo passo da derivação, o analisador usa o primeiro token após esse prefixo para escolher qual regra usar
- Esse processo continua até todo o programa ser derivado ou acontecer um erro (o prefixo de terminais da forma sentencial não casa com um prefixo do programa)

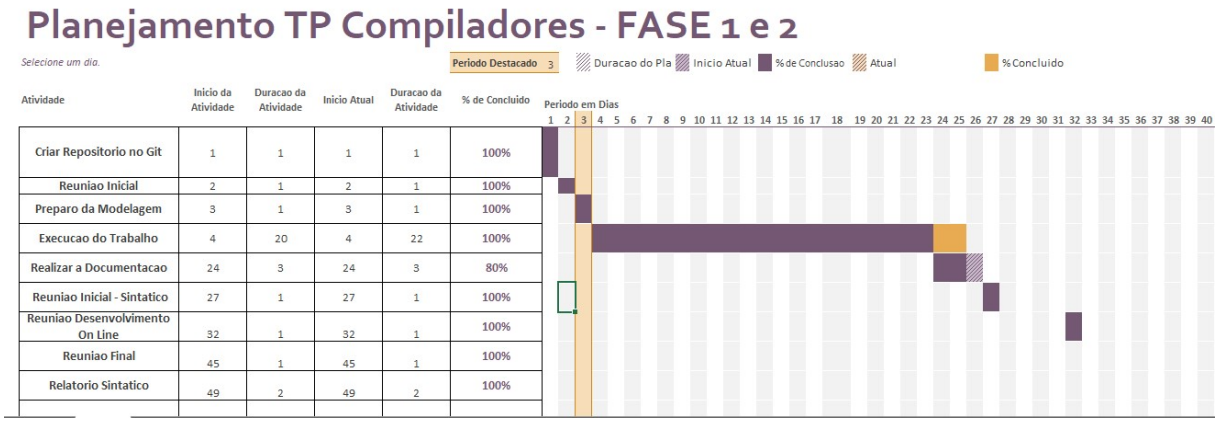
Assim, foi criado as tabelas de Frist e Follow da Gramatica e estao dispostas no arquivo Criado pelo Programa Microsoft Excel. O Arquivo encontra-se na pasta raiz deste trabalho,pode ser encontrado pelo nome : ”AnaliseSintatica_FristFollow_LL1”

6. PLANEJAMENTO

Para que todo projeto possa ser bem sucedido, devem se analisar, projetar e estudar minuciosamente e fazer o levantamento de todo os requisitos necessários para o projeto. Apartir das matérias de Modelagem e Desenvolvimento de Sistenas e Engenharia de Software, foi possível adquirir conhecimentos para um bom planejamento do trabalho. Como esse trabalho demonstrou ser extenso e complexo, decidimos por nos planejarmos a fim de evitar que possíveis erros de implementação fosse acarretado durante a fase de Desenvolvimento. Assim, fizemos uma reunião inicial para decidir os pontos importantes e decidir pontos sobre implementação do código. Utilizamos um organizador de tarefas no Excel para acompanhar as tarefas.Tal diagrama sera utilizado para as próximas etapas também. Veja abaixo, como ficou a organização do planejamento deste trabalho na figura abaixo.

Figura 1 – Planejamento Analise TP Compiladores

(a) Diagrama Planejamento



Atividade	Inicio da Atividade	Duracao da Atividade	Inicio Atual	Duracao da Atividade	% de Concluido
Criar Repositorio no Git	1	1	1	1	100%
Reuniao Inicial	2	1	2	1	100%
Preparo da Modelagem	3	1	3	1	100%
Execucao do Trabalho	4	20	4	22	100%
Realizar a Documentacao	24	3	24	3	80%
Reuniao Inicial - Sintatico	27	1	27	1	100%
Reuniao Desenvolvimento On Line	32	1	32	1	100%
Reuniao Final	45	1	45	1	100%
Relatorio Sintatico	49	2	49	2	100%

Fonte: Desenvolvedores do Código

Tal arquivo se encontra dentro do diretório deste projeto criado pelo Programa Microsoft Excel, podendo ser acesso pelo caminho: ”TP_Compilador/Planejamento/

7. TESTES REALIZADOS

Para testar todo o compilador construído, foram propostos sete exemplos de teste, abordando algumas possibilidade de programas a serem compilados. Apartir desses arquivos podemos visualizar os seguintes retorno do compilador :

7.1. TESTE 1

```

init
a, b, $c, is integer;
result is real;
write("Digite o valor de a:"
read (a);
write("Digite o valor de c:"
read (c);
b := 10
result := (a * c)/(b + 5 - 345);
write("O resultado e:");
write(result);
stop
```

Listing 7.1: Teste1.txt

Para o código acima o compilador obteve a seguinte saida :

```
**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, a >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, b >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, c >
Token Consumido(2): < virgula >
Error in: identList; Error(2): Token n?čo esperado:< is_decl >
Fim de arquivo inesperado.
```

Listing 7.2: Saida para o Codigo de teste : Teste1.txt

NO exemplo acima, a declaração da ultima variável foi declarada errada, dessa forma, o programa é abortado.

7.1.1. Correção Do Teste 1

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

```
**** Tokens lidos ****
< init_program >
< identifier, a >
< virgula >
< identifier, b >
< virgula >
Error  na Linha(2): Token '$' Nao esperado
< identifier, c >
< is_decl >
< integer_type >
< ponto_virgula >
< identifier, result >
< is_decl >
< real_type >
< ponto_virgula >
< begin >
< write >
< abre_parent >
< literal, "Digite o valor de a:" >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, a >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Digite o valor de c:" >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, c >
< fecha_parent >
< ponto_virgula >
< identifier, b >
< assign >
< num, 10 >
< ponto_virgula >
< identifier, result >
< assign >
< abre_parent >
```

```
< identifier, a >
< mult_mulop >
< identifier, c >
< fecha_parent >
< div_mulop >
< abre_parent >
< identifier, b >
< soma_addop >
< num, 5 >
< menos_addop >
< num, 345 >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "O resultado e:" >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, result >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, a >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, b >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, c >
Token Consumido(2): < is_decl >
Token Consumido(2): < integer_type >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < identifier, result >
Token Consumido(3): < is_decl >
Token Consumido(3): < real_type >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < begin >
Token Consumido(5): < write >
Token Consumido(5): < abre_parent >
Token Consumido(5): < literal, "Digite o valor de a:" >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
```

```
Token Consumido(6): < read >
Token Consumido(6): < abre_parent >
Token Consumido(6): < identifier, a >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < write >
Token Consumido(7): < abre_parent >
Token Consumido(7): < literal, "Digite o valor de c:" >
Token Consumido(7): < fecha_parent >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < read >
Token Consumido(8): < abre_parent >
Token Consumido(8): < identifier, c >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < identifier, b >
Token Consumido(9): < assign >
Token Consumido(9): < num, 10 >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < identifier, result >
Token Consumido(10): < assign >
Token Consumido(10): < abre_parent >
Token Consumido(10): < identifier, a >
Token Consumido(10): < mult_mulop >
Token Consumido(10): < identifier, c >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < div_mulop >
Token Consumido(10): < abre_parent >
Token Consumido(10): < identifier, b >
Token Consumido(10): < soma_addop >
Token Consumido(10): < num, 5 >
Token Consumido(10): < menos_addop >
Token Consumido(10): < num, 345 >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < ponto_virgula >
Token Consumido(11): < write >
Token Consumido(11): < abre_parent >
Token Consumido(11): < literal, "O resultado e:" >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < ponto_virgula >
Token Consumido(12): < write >
Token Consumido(12): < abre_parent >
Token Consumido(12): < identifier, result >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
```

Listing 7.3: Saida Correta para o Codigo de teste : Teste1.txt

7.2. TESTE 2

```
init
write("Entre com o valor de a:");
read (a);
b_1 := a * a;
write("O valor de b1 e:");
write (b_1);
b_2 = b + a/2 * (a + 5);
write("O valor de b1 e:");
Write (b1);
stop
```

Listing 7.4: Teste2.txt

Para o código acima o compilador obteve a seguinte saida :

```

**** Tokens lidos ****
< identifier, a >
< virgula >
Error  na Linha(1): Token '_' Nao esperado
< identifier, b_1 >
< virgula >
< identifier, b_2 >
Error  na Linha(1): Token ':' Nao esperado
< integer_type >
< ponto_virgula >
< init_program >
< write >
< abre_parent >
< literal, "Entre com o valor de a:" >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, a >
< fecha_parent >
< ponto_virgula >
< identifier, b_1 >
< assign >
< identifier, a >
< mult_mulop >
< identifier, a >
< ponto_virgula >
< write >
< abre_parent >
< literal, "O valor de b1 e:" >
```

```
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, b_1 >
< fecha_parent >
< ponto_virgula >
< identifier, b_2 >
< equal_relop >
< identifier, b >
< soma_addop >
< identifier, a >
< div_mulop >
< num, 2 >
< mult_mulop >
< abre_parent >
< identifier, a >
< soma_addop >
< num, 5 >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "O valor de b1 e:" >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, b1 >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Error in: program; Error(1): Token n?čo esperado:< identifier, a >
```

Listing 7.5: Saida para o Codigo de teste : Teste2.txt

No Exemplo acima, foi reportado um Token não esperado, pois se esperava o identificador init para inicio de qualquer programa.

7.2.1. Correção Do Teste 2

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

Listing 7.6: Saida Correta para o Codigo de teste : Teste2.txt

7.3. TESTE 3

```
% Programa de Teste Calculo de idade %
INIT
cont is int;
media, idade, soma is integer;
begin
cont := 5;
soma := 0;
do
write("Altura:" );
read (altura);
soma := soma+altura;
cont := cont - 1;
while(cont > 0)
media := soma / qts
write("Media: ");
write (media);
STOP
```

Listing 7.7: Teste3.txt

Para o código acima o compilador obteve a seguinte saída :

```
**** Tokens lidos ****
< init_program >
< identifier, cont >
< is_decl >
< identifier, int >
< ponto_virgula >
< identifier, media >
< virgula >
< identifier, idade >
< virgula >
< identifier, soma >
< is_decl >
< integer_type >
< ponto_virgula >
< begin >
< identifier, cont >
< assign >
< num, 5 >
< ponto_virgula >
```

```
< identifier, soma >
< assign >
< num, 0 >
< ponto_virgula >
< do >
< write >
< abre_parent >
< literal, "Altura:" >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, altura >
< fecha_parent >
< ponto_virgula >
< identifier, soma >
< assign >
< identifier, soma >
< soma_addop >
< identifier, altura >
< ponto_virgula >
< identifier, cont >
< assign >
< identifier, cont >
< menos_addop >
< num, 1 >
< ponto_virgula >
< while >
< abre_parent >
< identifier, cont >
< greater_than_relop >
< num, 0 >
< fecha_parent >
< identifier, media >
< assign >
< identifier, soma >
< div_mulop >
< identifier, qts >
< write >
< abre_parent >
< literal, "Media: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, media >
< fecha_parent >
```

```
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(3): < init_program >
Token Consumido(4): < identifier, cont >
Token Consumido(4): < is_decl >
Error in: type; Error(4): Token n?čo esperado:< identifier, int >
Fim de arquivo inesperado.
```

Listing 7.8: Saída para o Código de teste : Teste3.txt

No exemplo acima , o programa reportou um erro pois foi definido um tipo de variável inexistente na gramatica para essa linguagem. O tipo de variável int não foi modelado para essa gramatica.

7.3.1. Correção Do Teste 3

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

```
**** Tokens lidos ****
< init_program >
< identifier, cont >
< virgula >
< identifier, qts >
< is_decl >
< integer_type >
< ponto_virgula >
< identifier, media >
< virgula >
< identifier, idade >
< virgula >
< identifier, soma >
< is_decl >
< integer_type >
< ponto_virgula >
< begin >
< identifier, cont >
< assign >
< num, 5 >
< ponto_virgula >
< identifier, soma >
< assign >
< num, 0 >
< ponto_virgula >
```

```
< identifier, qts >
< assign >
< num, 1 >
< ponto_virgula >
< do >
< write >
< abre_parent >
< literal, "Altura:" >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, altura >
< fecha_parent >
< ponto_virgula >
< identifier, soma >
< assign >
< identifier, soma >
< soma_addop >
< identifier, altura >
< ponto_virgula >
< identifier, cont >
< assign >
< identifier, cont >
< menos_addop >
< num, 1 >
< ponto_virgula >
< while >
< abre_parent >
< identifier, cont >
< greater_than_relop >
< num, 0 >
< fecha_parent >
< ponto_virgula >
< identifier, media >
< assign >
< identifier, soma >
< div_mulop >
< identifier, qts >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Media: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
```

```
< identifier, media >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(3): < init_program >
Token Consumido(4): < identifier, cont >
Token Consumido(4): < virgula >
Token Consumido(4): < identifier, qts >
Token Consumido(4): < is_decl >
Token Consumido(4): < integer_type >
Token Consumido(4): < ponto_virgula >
Token Consumido(5): < identifier, media >
Token Consumido(5): < virgula >
Token Consumido(5): < identifier, idade >
Token Consumido(5): < virgula >
Token Consumido(5): < identifier, soma >
Token Consumido(5): < is_decl >
Token Consumido(5): < integer_type >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < begin >
Token Consumido(7): < identifier, cont >
Token Consumido(7): < assign >
Token Consumido(7): < num, 5 >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < identifier, soma >
Token Consumido(8): < assign >
Token Consumido(8): < num, 0 >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < identifier, qts >
Token Consumido(9): < assign >
Token Consumido(9): < num, 1 >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < do >
Token Consumido(11): < write >
Token Consumido(11): < abre_parent >
Token Consumido(11): < literal, "Altura:" >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < ponto_virgula >
Token Consumido(12): < read >
Token Consumido(12): < abre_parent >
Token Consumido(12): < identifier, altura >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
```

```
Token Consumido(13): < identifier, soma >
Token Consumido(13): < assign >
Token Consumido(13): < identifier, soma >
Token Consumido(13): < soma_addop >
Token Consumido(13): < identifier, altura >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < identifier, cont >
Token Consumido(14): < assign >
Token Consumido(14): < identifier, cont >
Token Consumido(14): < menos_addop >
Token Consumido(14): < num, 1 >
Token Consumido(14): < ponto_virgula >
Token Consumido(15): < while >
Token Consumido(15): < abre_parent >
Token Consumido(15): < identifier, cont >
Token Consumido(15): < greater_than_relop >
Token Consumido(15): < num, 0 >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < ponto_virgula >
Token Consumido(16): < identifier, media >
Token Consumido(16): < assign >
Token Consumido(16): < identifier, soma >
Token Consumido(16): < div_mulop >
Token Consumido(16): < identifier, qts >
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < write >
Token Consumido(17): < abre_parent >
Token Consumido(17): < literal, "Media: " >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < write >
Token Consumido(18): < abre_parent >
Token Consumido(18): < identifier, media >
Token Consumido(18): < fecha_parent >
Token Consumido(18): < ponto_virgula >
```

Listing 7.9: Saida Correta para o Codigo de teste : Teste3.txt

7.4. TESTE 4

```
init
% Outro programa de teste
i, j, k, @total is integer;
nome is string
write("Digite o seu nome: ");
read(nome);
```

```
write("Digite um valor inteiro: );
read (I);
k := i * (5-i * 50 / 10;
j := i * 10;
k := i* j / k;
k := 4 + a $;
write(nome);
write(", os números gerados sao: ");
write(i);
write(j);
write(k);
```

Listing 7.10: Teste4.txt

Para o codigo acima o compilador obteve a seguinte saida :

```
**** Tokens lidos ****
< init_program >
Error(2): coment?ąrio n?čo fechado



**** Inicio Parser ****
Token Consumido(1): < init_program >
Fim de arquivo inesperado.
```

Listing 7.11: Saida para o Codigo de teste : Teste4.txt

No código acima, não foi fechado o comentário, assim, erro para iniciar o programa, pois era esperado o % para fim de comentário.

7.4.1. Correção Do Teste 4

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

```


**** Tokens lidos ****
< init_program >
< identifier, i >
< virgula >
< identifier, j >
< virgula >
< identifier, k >
< virgula >
Error  na Linha(3): Token '@' Nao esperado
< identifier, total >
< is_decl >
```

```
14 < integer_type >
15 < ponto_virgula >
16 < identifier, nome >
17 < is_decl >
18 < string_type >
19 < ponto_virgula >
20 < begin >
21 < write >
22 < abre_parent >
23 < literal, "Digite o seu nome: " >
24 < fecha_parent >
25 < ponto_virgula >
26 < read >
27 < abre_parent >
28 < identifier, nome >
29 < fecha_parent >
30 < ponto_virgula >
31 < write >
32 < abre_parent >
33 < literal, "Digite um valor inteiro: " >
34 < fecha_parent >
35 < ponto_virgula >
36 < read >
37 < abre_parent >
38 < identifier, i >
39 < fecha_parent >
40 < ponto_virgula >
41 < identifier, k >
42 < assign >
43 < abre_parent >
44 < abre_parent >
45 < identifier, i >
46 < mult_mulop >
47 < num, 5 >
48 < fecha_parent >
49 < menos_addop >
50 < abre_parent >
51 < identifier, i >
52 < mult_mulop >
53 < num, 50 >
54 < fecha_parent >
55 < fecha_parent >
56 < div_mulop >
57 < num, 10 >
58 < ponto_virgula >
59 < identifier, j >
60 < assign >
```

```
61 < identifier, i >
62 < mult_mulop >
63 < num, 10 >
64 < ponto_virgula >
65 < identifier, k >
66 < assign >
67 < abre_parent >
68 < identifier, i >
69 < mult_mulop >
70 < identifier, j >
71 < fecha_parent >
72 < div_mulop >
73 < identifier, k >
74 < ponto_virgula >
75 < identifier, k >
76 < assign >
77 < num, 4 >
78 < soma_addop >
79 < identifier, a >
80 Error  na Linha(13): Token '$' Nao esperado
81 < ponto_virgula >
82 < write >
83 < abre_parent >
84 < identifier, nome >
85 < fecha_parent >
86 < ponto_virgula >
87 < write >
88 < abre_parent >
89 < literal, ", os n?žmeros gerados sao: " >
90 < fecha_parent >
91 < ponto_virgula >
92 < write >
93 < abre_parent >
94 < identifier, i >
95 < fecha_parent >
96 < ponto_virgula >
97 < write >
98 < abre_parent >
99 < identifier, j >
100 < fecha_parent >
101 < ponto_virgula >
102 < write >
103 < abre_parent >
104 < identifier, k >
105 < fecha_parent >
106 < ponto_virgula >
107 < stop_program >
```

```
108
109
110
111 **** Inicio Parser ****
112 Token Consumido(1): < init_program >
113 Token Consumido(3): < identifier, i >
114 Token Consumido(3): < virgula >
115 Token Consumido(3): < identifier, j >
116 Token Consumido(3): < virgula >
117 Token Consumido(3): < identifier, k >
118 Token Consumido(3): < virgula >
119 Token Consumido(3): < identifier, total >
120 Token Consumido(3): < is_decl >
121 Token Consumido(3): < integer_type >
122 Token Consumido(3): < ponto_virgula >
123 Token Consumido(4): < identifier, nome >
124 Token Consumido(4): < is_decl >
125 Token Consumido(4): < string_type >
126 Token Consumido(4): < ponto_virgula >
127 Token Consumido(5): < begin >
128 Token Consumido(6): < write >
129 Token Consumido(6): < abre_parent >
130 Token Consumido(6): < literal, "Digite o seu nome: " >
131 Token Consumido(6): < fecha_parent >
132 Token Consumido(6): < ponto_virgula >
133 Token Consumido(7): < read >
134 Token Consumido(7): < abre_parent >
135 Token Consumido(7): < identifier, nome >
136 Token Consumido(7): < fecha_parent >
137 Token Consumido(7): < ponto_virgula >
138 Token Consumido(8): < write >
139 Token Consumido(8): < abre_parent >
140 Token Consumido(8): < literal, "Digite um valor inteiro: " >
141 Token Consumido(8): < fecha_parent >
142 Token Consumido(8): < ponto_virgula >
143 Token Consumido(9): < read >
144 Token Consumido(9): < abre_parent >
145 Token Consumido(9): < identifier, i >
146 Token Consumido(9): < fecha_parent >
147 Token Consumido(9): < ponto_virgula >
148 Token Consumido(10): < identifier, k >
149 Token Consumido(10): < assign >
150 Token Consumido(10): < abre_parent >
151 Token Consumido(10): < abre_parent >
152 Token Consumido(10): < identifier, i >
153 Token Consumido(10): < mult_mulop >
154 Token Consumido(10): < num, 5 >
```

```
155 Token Consumido(10): < fecha_parent >
156 Token Consumido(10): < menos_addop >
157 Token Consumido(10): < abre_parent >
158 Token Consumido(10): < identifier, i >
159 Token Consumido(10): < mult_mulop >
160 Token Consumido(10): < num, 50 >
161 Token Consumido(10): < fecha_parent >
162 Token Consumido(10): < fecha_parent >
163 Token Consumido(10): < div_mulop >
164 Token Consumido(10): < num, 10 >
165 Token Consumido(10): < ponto_virgula >
166 Token Consumido(11): < identifier, j >
167 Token Consumido(11): < assign >
168 Token Consumido(11): < identifier, i >
169 Token Consumido(11): < mult_mulop >
170 Token Consumido(11): < num, 10 >
171 Token Consumido(11): < ponto_virgula >
172 Token Consumido(12): < identifier, k >
173 Token Consumido(12): < assign >
174 Token Consumido(12): < abre_parent >
175 Token Consumido(12): < identifier, i >
176 Token Consumido(12): < mult_mulop >
177 Token Consumido(12): < identifier, j >
178 Token Consumido(12): < fecha_parent >
179 Token Consumido(12): < div_mulop >
180 Token Consumido(12): < identifier, k >
181 Token Consumido(12): < ponto_virgula >
182 Token Consumido(13): < identifier, k >
183 Token Consumido(13): < assign >
184 Token Consumido(13): < num, 4 >
185 Token Consumido(13): < soma_addop >
186 Token Consumido(13): < identifier, a >
187 Token Consumido(13): < ponto_virgula >
188 Token Consumido(14): < write >
189 Token Consumido(14): < abre_parent >
190 Token Consumido(14): < identifier, nome >
191 Token Consumido(14): < fecha_parent >
192 Token Consumido(14): < ponto_virgula >
193 Token Consumido(15): < write >
194 Token Consumido(15): < abre_parent >
195 Token Consumido(15): < literal, ", os n?žmeros gerados sao: " >
196 Token Consumido(15): < fecha_parent >
197 Token Consumido(15): < ponto_virgula >
198 Token Consumido(16): < write >
199 Token Consumido(16): < abre_parent >
200 Token Consumido(16): < identifier, i >
201 Token Consumido(16): < fecha_parent >
```

```
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < write >
Token Consumido(17): < abre_parent >
Token Consumido(17): < identifier, j >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < write >
Token Consumido(18): < abre_parent >
Token Consumido(18): < identifier, k >
Token Consumido(18): < fecha_parent >
Token Consumido(18): < ponto_virgula >
```

Listing 7.12: Saida Correta para o Codigo de teste : Teste4.txt

7.5. TESTE 5

```
init
i, j, k, @total is integer;
nome is string
write("Digite o seu nome: ");
read(nome);
write("Digite um valor inteiro: );
read (I);
k := i * (5-i * 50 / 10;
j := i * 10;
k := i* j / k;
k := 4 + a $;
write(nome);
write(", os números gerados sao: ");
write(i);
write(j);
write(k);
```

Listing 7.13: Teste5.txt

Para o codigo acima o compilador obteve a seguinte saida :

```

**** Tokens lidos ****
< init_program >
< identifier, i >
< virgula >
< identifier, j >
< virgula >
< identifier, k >
< virgula >
Error  na Linha(2): Token '@' Nao esperado
```

```
< identifier, total >
< is_decl >
< integer_type >
< ponto_virgula >
< identifier, nome >
< is_decl >
< string_type >
< write >
< abre_parent >
< literal, "Digite o seu nome: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, nome >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
Error  na Linha(6): Token '"' Nao esperado
< read >
< abre_parent >
< identifier, i >
< fecha_parent >
< ponto_virgula >
< identifier, k >
< assign >
< identifier, i >
< mult_mulop >
< abre_parent >
< num, 5 >
< menos_addop >
< identifier, i >
< mult_mulop >
< num, 50 >
< div_mulop >
< num, 10 >
< ponto_virgula >
< identifier, j >
< assign >
< identifier, i >
< mult_mulop >
< num, 10 >
< ponto_virgula >
< identifier, k >
< assign >
< identifier, i >
```

```
< mult_mulop >
< identifier, j >
< div_mulop >
< identifier, k >
< ponto_virgula >
< identifier, k >
< assign >
< num, 4 >
< soma_addop >
< identifier, a >
Error  na Linha(10): Token '$' Nao esperado
< ponto_virgula >
< write >
< abre_parent >
< identifier, nome >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, ", os n?žmeros gerados sao: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, i >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, j >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, k >
< fecha_parent >
< ponto_virgula >


**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, i >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, j >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, k >
```

```
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, total >
Token Consumido(2): < is_decl >
Token Consumido(2): < integer_type >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < identifier, nome >
Token Consumido(3): < is_decl >
Token Consumido(3): < string_type >
Error in: declList; Error(4): Token n?čo esperado:< write >
Fim de arquivo inesperado.
```

Listing 7.14: Saida para o Codigo de teste : Teste5.txt

No Código acima, não foi declarado o begin. Assim o comando write foi dado como inesperado, pois era esperado o identificador begin.

7.5.1. Correção Do Teste 5

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

```


**** Tokens lidos ****
< init_program >
< identifier, i >
< virgula >
< identifier, j >
< virgula >
< identifier, k >
< virgula >
Error  na Linha(3): Token '@' Nao esperado
< identifier, total >
< is_decl >
< integer_type >
< ponto_virgula >
< identifier, nome >
< is_decl >
< string_type >
< ponto_virgula >
< begin >
< write >
< abre_parent >
< literal, "Digite o seu nome: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
```

```
28 < identifier, nome >
29 < fecha_parent >
30 < ponto_virgula >
31 < write >
32 < abre_parent >
33 < literal, "Digite um valor inteiro: " >
34 < fecha_parent >
35 < ponto_virgula >
36 < read >
37 < abre_parent >
38 < identifier, i >
39 < fecha_parent >
40 < ponto_virgula >
41 < identifier, k >
42 < assign >
43 < abre_parent >
44 < abre_parent >
45 < identifier, i >
46 < mult_mulop >
47 < num, 5 >
48 < fecha_parent >
49 < menos_addop >
50 < abre_parent >
51 < identifier, i >
52 < mult_mulop >
53 < num, 50 >
54 < fecha_parent >
55 < fecha_parent >
56 < div_mulop >
57 < num, 10 >
58 < ponto_virgula >
59 < identifier, j >
60 < assign >
61 < identifier, i >
62 < mult_mulop >
63 < num, 10 >
64 < ponto_virgula >
65 < identifier, k >
66 < assign >
67 < abre_parent >
68 < identifier, i >
69 < mult_mulop >
70 < identifier, j >
71 < fecha_parent >
72 < div_mulop >
73 < identifier, k >
74 < ponto_virgula >
```

```
< identifier, k >
< assign >
< num, 4 >
< soma_addop >
< identifier, a >
Error  na Linha(13): Token '$' Nao esperado
< ponto_virgula >
< write >
< abre_parent >
< identifier, nome >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, ", os n?žmeros gerados sao: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, i >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, j >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, k >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(3): < identifier, i >
Token Consumido(3): < virgula >
Token Consumido(3): < identifier, j >
Token Consumido(3): < virgula >
Token Consumido(3): < identifier, k >
Token Consumido(3): < virgula >
Token Consumido(3): < identifier, total >
Token Consumido(3): < is_decl >
Token Consumido(3): < integer_type >
```

```
122 Token Consumido(3): < ponto_virgula >
123 Token Consumido(4): < identifier, nome >
124 Token Consumido(4): < is_decl >
125 Token Consumido(4): < string_type >
126 Token Consumido(4): < ponto_virgula >
127 Token Consumido(5): < begin >
128 Token Consumido(6): < write >
129 Token Consumido(6): < abre_parent >
130 Token Consumido(6): < literal, "Digite o seu nome: " >
131 Token Consumido(6): < fecha_parent >
132 Token Consumido(6): < ponto_virgula >
133 Token Consumido(7): < read >
134 Token Consumido(7): < abre_parent >
135 Token Consumido(7): < identifier, nome >
136 Token Consumido(7): < fecha_parent >
137 Token Consumido(7): < ponto_virgula >
138 Token Consumido(8): < write >
139 Token Consumido(8): < abre_parent >
140 Token Consumido(8): < literal, "Digite um valor inteiro: " >
141 Token Consumido(8): < fecha_parent >
142 Token Consumido(8): < ponto_virgula >
143 Token Consumido(9): < read >
144 Token Consumido(9): < abre_parent >
145 Token Consumido(9): < identifier, i >
146 Token Consumido(9): < fecha_parent >
147 Token Consumido(9): < ponto_virgula >
148 Token Consumido(10): < identifier, k >
149 Token Consumido(10): < assign >
150 Token Consumido(10): < abre_parent >
151 Token Consumido(10): < abre_parent >
152 Token Consumido(10): < identifier, i >
153 Token Consumido(10): < mult_mulop >
154 Token Consumido(10): < num, 5 >
155 Token Consumido(10): < fecha_parent >
156 Token Consumido(10): < menos_addop >
157 Token Consumido(10): < abre_parent >
158 Token Consumido(10): < identifier, i >
159 Token Consumido(10): < mult_mulop >
160 Token Consumido(10): < num, 50 >
161 Token Consumido(10): < fecha_parent >
162 Token Consumido(10): < fecha_parent >
163 Token Consumido(10): < div_mulop >
164 Token Consumido(10): < num, 10 >
165 Token Consumido(10): < ponto_virgula >
166 Token Consumido(11): < identifier, j >
167 Token Consumido(11): < assign >
168 Token Consumido(11): < identifier, i >
```

```
169 Token Consumido(11): < mult_mulop >
170 Token Consumido(11): < num, 10 >
171 Token Consumido(11): < ponto_virgula >
172 Token Consumido(12): < identifier, k >
173 Token Consumido(12): < assign >
174 Token Consumido(12): < abre_parent >
175 Token Consumido(12): < identifier, i >
176 Token Consumido(12): < mult_mulop >
177 Token Consumido(12): < identifier, j >
178 Token Consumido(12): < fecha_parent >
179 Token Consumido(12): < div_mulop >
180 Token Consumido(12): < identifier, k >
181 Token Consumido(12): < ponto_virgula >
182 Token Consumido(13): < identifier, k >
183 Token Consumido(13): < assign >
184 Token Consumido(13): < num, 4 >
185 Token Consumido(13): < soma_addop >
186 Token Consumido(13): < identifier, a >
187 Token Consumido(13): < ponto_virgula >
188 Token Consumido(14): < write >
189 Token Consumido(14): < abre_parent >
190 Token Consumido(14): < identifier, nome >
191 Token Consumido(14): < fecha_parent >
192 Token Consumido(14): < ponto_virgula >
193 Token Consumido(15): < write >
194 Token Consumido(15): < abre_parent >
195 Token Consumido(15): < literal, ", os n?žmeros gerados sao: " >
196 Token Consumido(15): < fecha_parent >
197 Token Consumido(15): < ponto_virgula >
198 Token Consumido(16): < write >
199 Token Consumido(16): < abre_parent >
200 Token Consumido(16): < identifier, i >
201 Token Consumido(16): < fecha_parent >
202 Token Consumido(16): < ponto_virgula >
203 Token Consumido(17): < write >
204 Token Consumido(17): < abre_parent >
205 Token Consumido(17): < identifier, j >
206 Token Consumido(17): < fecha_parent >
207 Token Consumido(17): < ponto_virgula >
208 Token Consumido(18): < write >
209 Token Consumido(18): < abre_parent >
210 Token Consumido(18): < identifier, k >
211 Token Consumido(18): < fecha_parent >
212 Token Consumido(18): < ponto_virgula >
```

Listing 7.15: Saida Correta para o Codigo de teste : Teste5.txt

7.6. TESTE 6

```
init
a, b, c, maior is integer;
write("Digite uma idade: ");
read(a);
write("Digite outra idade: ");
read(b);
write("Digite mais uma idade: ");
read(c;
maior := 0;
if ( a>b and a>c )
maior := a;
else
if (b>c)
maior := b;
else
maior := c;
write("Maior idade: ");
write(maior);
end
```

Listing 7.16: Teste6.txt

Para o codigo acima o compilador obteve a seguinte saida :

```
**** Tokens lidos ****
< init_program >
< identifier, a >
< virgula >
< identifier, b >
< virgula >
< identifier, c >
< virgula >
< identifier, maior >
< is_decl >
< integer_type >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Digite uma idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, a >
< fecha_parent >
< ponto_virgula >
```

```
< write >
< abre_parent >
< literal, "Digite outra idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, b >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Digite mais uma idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, c >
< ponto_virgula >
< identifier, maior >
< assign >
< num, 0 >
< ponto_virgula >
< if >
< abre_parent >
< identifier, a >
< greater_than_relop >
< identifier, b >
< and_mulop >
< identifier, a >
< greater_than_relop >
< identifier, c >
< fecha_parent >
< identifier, maior >
< assign >
< identifier, a >
< ponto_virgula >
< else >
< if >
< abre_parent >
< identifier, b >
< greater_than_relop >
< identifier, c >
< fecha_parent >
< identifier, maior >
< assign >
< identifier, b >
```

```
< ponto_virgula >
< else >
< identifier, maior >
< assign >
< identifier, c >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Maior idade: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, maior >
< fecha_parent >
< ponto_virgula >
< end >



**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, a >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, b >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, c >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, maior >
Token Consumido(2): < is_decl >
Token Consumido(2): < integer_type >
Token Consumido(2): < ponto_virgula >
Error in: identList; Error(3): Token n?čo esperado:< write >
Fim de arquivo inesperado.
```

Listing 7.17: Saida para o Codigo de teste : Teste6.txt

No Código acima, não foi declarado o begin. Assim o comando write foi dado como inesperado, pois era esperado o identificador begin.

7.6.1. Correção Do Teste 6

Foram feitas as devidas correções no código acima para adequar as regras da gramatica da linguagem .Assim, obtemos o seguinte resultado.

```


**** Tokens lidos ****
```

```
< init_program >
< identifier, a >
< virgula >
< identifier, b >
< virgula >
< identifier, c >
< virgula >
< identifier, maior >
< is_decl >
< integer_type >
< ponto_virgula >
< begin >
< write >
< abre_parent >
< literal, "Digite uma idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, a >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Digite outra idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, b >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< literal, "Digite mais uma idade: " >
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, c >
< fecha_parent >
< ponto_virgula >
< identifier, maior >
< assign >
< num, 0 >
< ponto_virgula >
< if >
```

```
51 < abre_parent >
52 < abre_parent >
53 < identifier, a >
54 < greater_than_relop >
55 < identifier, b >
56 < fecha_parent >
57 < and_mulop >
58 < abre_parent >
59 < identifier, a >
60 < greater_than_relop >
61 < identifier, c >
62 < fecha_parent >
63 < fecha_parent >
64 < begin >
65 < identifier, maior >
66 < assign >
67 < identifier, a >
68 < ponto_virgula >
69 < end >
70 < else >
71 < begin >
72 < if >
73 < abre_parent >
74 < identifier, b >
75 < greater_than_relop >
76 < identifier, c >
77 < fecha_parent >
78 < begin >
79 < identifier, maior >
80 < assign >
81 < identifier, b >
82 < ponto_virgula >
83 < end >
84 < else >
85 < begin >
86 < identifier, maior >
87 < assign >
88 < identifier, c >
89 < ponto_virgula >
90 < end >
91 < end >
92 < write >
93 < abre_parent >
94 < literal, "Maior idade: " >
95 < fecha_parent >
96 < ponto_virgula >
97 < write >
```

```
< abre_parent >
< identifier, maior >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, a >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, b >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, c >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, maior >
Token Consumido(2): < is_decl >
Token Consumido(2): < integer_type >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < begin >
Token Consumido(4): < write >
Token Consumido(4): < abre_parent >
Token Consumido(4): < literal, "Digite uma idade: " >
Token Consumido(4): < fecha_parent >
Token Consumido(4): < ponto_virgula >
Token Consumido(5): < read >
Token Consumido(5): < abre_parent >
Token Consumido(5): < identifier, a >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < write >
Token Consumido(6): < abre_parent >
Token Consumido(6): < literal, "Digite outra idade: " >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < read >
Token Consumido(7): < abre_parent >
Token Consumido(7): < identifier, b >
Token Consumido(7): < fecha_parent >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < write >
Token Consumido(8): < abre_parent >
Token Consumido(8): < literal, "Digite mais uma idade: " >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < read >
```

```
145 Token Consumido(9): < abre_parent >
146 Token Consumido(9): < identifier, c >
147 Token Consumido(9): < fecha_parent >
148 Token Consumido(9): < ponto_virgula >
149 Token Consumido(10): < identifier, maior >
150 Token Consumido(10): < assign >
151 Token Consumido(10): < num, 0 >
152 Token Consumido(10): < ponto_virgula >
153 Token Consumido(11): < if >
154 Token Consumido(11): < abre_parent >
155 Token Consumido(11): < abre_parent >
156 Token Consumido(11): < identifier, a >
157 Token Consumido(11): < greater_than_relop >
158 Token Consumido(11): < identifier, b >
159 Token Consumido(11): < fecha_parent >
160 Token Consumido(11): < and_mulop >
161 Token Consumido(11): < abre_parent >
162 Token Consumido(11): < identifier, a >
163 Token Consumido(11): < greater_than_relop >
164 Token Consumido(11): < identifier, c >
165 Token Consumido(11): < fecha_parent >
166 Token Consumido(11): < fecha_parent >
167 Token Consumido(11): < begin >
168 Token Consumido(12): < identifier, maior >
169 Token Consumido(12): < assign >
170 Token Consumido(12): < identifier, a >
171 Token Consumido(12): < ponto_virgula >
172 Token Consumido(13): < end >
173 Token Consumido(13): < else >
174 Token Consumido(13): < begin >
175 Token Consumido(14): < if >
176 Token Consumido(14): < abre_parent >
177 Token Consumido(14): < identifier, b >
178 Token Consumido(14): < greater_than_relop >
179 Token Consumido(14): < identifier, c >
180 Token Consumido(14): < fecha_parent >
181 Token Consumido(14): < begin >
182 Token Consumido(15): < identifier, maior >
183 Token Consumido(15): < assign >
184 Token Consumido(15): < identifier, b >
185 Token Consumido(15): < ponto_virgula >
186 Token Consumido(16): < end >
187 Token Consumido(16): < else >
188 Token Consumido(16): < begin >
189 Token Consumido(17): < identifier, maior >
190 Token Consumido(17): < assign >
191 Token Consumido(17): < identifier, c >
```

```
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < end >
Token Consumido(19): < end >
Token Consumido(20): < write >
Token Consumido(20): < abre_parent >
Token Consumido(20): < literal, "Maior idade: " >
Token Consumido(20): < fecha_parent >
Token Consumido(20): < ponto_virgula >
Token Consumido(21): < write >
Token Consumido(21): < abre_parent >
Token Consumido(21): < identifier, maior >
Token Consumido(21): < fecha_parent >
Token Consumido(21): < ponto_virgula >
```

Listing 7.18: Saida Correta para o Codigo de teste : Teste6.txt

7.7. TESTE 7

```
% Programa de Teste Calculo de idade %
init
  cont_ is integer;
  media, idade, soma_ is integer;
begin
  cont_ := 5;
  soma := 0;
  do
    write(Altura: );
    read (altura);
    soma := soma + altura;
    cont_ := cont_ - 1;
  while(cont_ > 0);
  write(Media: );
  write (soma / qtd);
stop
```

Listing 7.19: Teste7.txt

Para o codigo acima o compilador obteve a seguinte saida :

```
**** Tokens lidos ****
< init_program >
< identifier, cont_ >
< is_decl >
< integer_type >
< ponto_virgula >
< identifier, media >
< virgula >
```

```
< identifier, idade >
< virgula >
< identifier, soma_ >
< is_decl >
< integer_type >
< ponto_virgula >
< begin >
< identifier, cont_ >
< assign >
< num, 5 >
< ponto_virgula >
< identifier, soma >
< assign >
< num, 0 >
< ponto_virgula >
< do >
< write >
< abre_parent >
Error  na Linha(9): Token 'â' Nao esperado
Error  na Linha(9): Token '?' Nao esperado
Error  na Linha(9): Token '?' Nao esperado
< identifier, Altura >
Error  na Linha(9): Token ':' Nao esperado
Error  na Linha(9): Token '?' Nao esperado
Error  na Linha(9): Token '?' Nao esperado
< fecha_parent >
< ponto_virgula >
< read >
< abre_parent >
< identifier, altura >
< fecha_parent >
< ponto_virgula >
< identifier, soma >
< assign >
< identifier, soma >
< soma_addop >
< identifier, altura >
< ponto_virgula >
< identifier, cont_ >
< assign >
< identifier, cont_ >
< menos_addop >
< num, 1 >
< ponto_virgula >
< while >
< abre_parent >
< identifier, cont_ >
```

```
< greater_than_relop >
< num, 0 >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
Error  na Linha(14): Token 'â' Nao esperado
Error  na Linha(14): Token '?' Nao esperado
Error  na Linha(14): Token '?' Nao esperado
< identifier, media >
Error  na Linha(14): Token ':' Nao esperado
Error  na Linha(14): Token 'â' Nao esperado
Error  na Linha(14): Token '?' Nao esperado
Error  na Linha(14): Token '?' Nao esperado
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, soma >
< div_mulop >
< identifier, qtd >
< fecha_parent >
< ponto_virgula >
< stop_program >



**** Inicio Parser ****
Token Consumido(2): < init_program >
Token Consumido(3): < identifier, cont_ >
Token Consumido(3): < is_decl >
Token Consumido(3): < integer_type >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < identifier, media >
Token Consumido(4): < virgula >
Token Consumido(4): < identifier, idade >
Token Consumido(4): < virgula >
Token Consumido(4): < identifier, soma_ >
Token Consumido(4): < is_decl >
Token Consumido(4): < integer_type >
Token Consumido(4): < ponto_virgula >
Token Consumido(5): < begin >
Token Consumido(6): < identifier, cont_ >
Token Consumido(6): < assign >
Token Consumido(6): < num, 5 >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < identifier, soma >
```

```
Token Consumido(7): < assign >
Token Consumido(7): < num, 0 >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < do >
Token Consumido(9): < write >
Token Consumido(9): < abre_parent >
Token Consumido(9): < identifier, Altura >
Token Consumido(9): < fecha_parent >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < read >
Token Consumido(10): < abre_parent >
Token Consumido(10): < identifier, altura >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < ponto_virgula >
Token Consumido(11): < identifier, soma >
Token Consumido(11): < assign >
Token Consumido(11): < identifier, soma >
Token Consumido(11): < soma_addop >
Token Consumido(11): < identifier, altura >
Token Consumido(11): < ponto_virgula >
Token Consumido(12): < identifier, cont_ >
Token Consumido(12): < assign >
Token Consumido(12): < identifier, cont_ >
Token Consumido(12): < menos_addop >
Token Consumido(12): < num, 1 >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < while >
Token Consumido(13): < abre_parent >
Token Consumido(13): < identifier, cont_ >
Token Consumido(13): < greater_than_relop >
Token Consumido(13): < num, 0 >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < write >
Token Consumido(14): < abre_parent >
Token Consumido(14): < identifier, media >
Token Consumido(14): < fecha_parent >
Token Consumido(14): < ponto_virgula >
Token Consumido(15): < write >
Token Consumido(15): < abre_parent >
Token Consumido(15): < identifier, soma >
Token Consumido(15): < div_mulop >
Token Consumido(15): < identifier, qtd >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < ponto_virgula >
```

Listing 7.20: Saida para o Codigo de teste : Teste.txt

O Código acima esta escrito conforme a gramatica da linguagem . Não foi reportado nenhum erro.

7.8. TESTE 8

```
init
a, b is integer;
begin
if ( a > b) begin
write ("O maior numero e: ");
write (a);
end else begin
write ("O menor numero e: ");
write (b);
end
stop
```

Listing 7.21: Teste8.txt

Para o codigo acima o compilador obteve a seguinte saida :

```
**** Tokens lidos ****
< init_program >
< identifier, a >
< virgula >
< identifier, b >
< is_decl >
< integer_type >
< ponto_virgula >
< begin >
< if >
< abre_parent >
< identifier, a >
< greater_than_relop >
< identifier, b >
< fecha_parent >
< begin >
< write >
< abre_parent >
< literal, "O maior numero e: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, a >
< fecha_parent >
< ponto_virgula >
```

```
< end >
< else >
< begin >
< write >
< abre_parent >
< literal, "O menor numero e: " >
< fecha_parent >
< ponto_virgula >
< write >
< abre_parent >
< identifier, b >
< fecha_parent >
< ponto_virgula >
< end >
< stop_program >



**** Inicio Parser ****
Token Consumido(1): < init_program >
Token Consumido(2): < identifier, a >
Token Consumido(2): < virgula >
Token Consumido(2): < identifier, b >
Token Consumido(2): < is_decl >
Token Consumido(2): < integer_type >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < begin >
Token Consumido(4): < if >
Token Consumido(4): < abre_parent >
Token Consumido(4): < identifier, a >
Token Consumido(4): < greater_than_relop >
Token Consumido(4): < identifier, b >
Token Consumido(4): < fecha_parent >
Token Consumido(4): < begin >
Token Consumido(5): < write >
Token Consumido(5): < abre_parent >
Token Consumido(5): < literal, "O maior numero e: " >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < write >
Token Consumido(6): < abre_parent >
Token Consumido(6): < identifier, a >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < end >
Token Consumido(7): < else >
Token Consumido(7): < begin >
```

```
74 Token Consumido(8): < write >
75 Token Consumido(8): < abre_parent >
76 Token Consumido(8): < literal, "O menor numero e: " >
77 Token Consumido(8): < fecha_parent >
78 Token Consumido(8): < ponto_virgula >
79 Token Consumido(9): < write >
80 Token Consumido(9): < abre_parent >
81 Token Consumido(9): < identifier, b >
82 Token Consumido(9): < fecha_parent >
83 Token Consumido(9): < ponto_virgula >
84 Token Consumido(10): < end >
```

Listing 7.22: Saida para o Codigo de teste : Teste8.txt

O Código acima foi criado seguindo as regras de derivação da gramatica da linguagem. Não foram encontrados erros.

8. METODOLOGIA

Como dito anteriormente, este trabalho está sendo desenvolvido em consonância com os conceitos ensinados pela orientadora e pelo Autor Aho, pelo livro : Compiladores, Técnicas e ferramentas.

```
1  package analisador_sintatico;
2
3  import analisador_lexico.Tag;
4  import analisador_lexico.Token;
5  import java.util.ArrayList;
6  import java.util.Hashtable;
7
8  public class Parser {
9      private Token tok;
10     private int tag;
11     private int i;
12     private int line;
13     private int curType;
14     private int resultExprType;
15     private ArrayList<Token> tokens = new ArrayList<Token> ();
16
17     public Parser(ArrayList<Token> tokens){
18         this.tokens=tokens;
19         i=0;
20         tok=tokens.get(i);
21         tag=tok.tag;
22         line=tok.line;
23         curType=Tag.VOID;
24         resultExprType=Tag.VOID;
25     }
26
27     public void init(){
28         program();
29     };
30
31     private void advance(){
32         i++;
33         tok=tokens.get(i);
34         tag=tok.tag;
35         line=tok.line;
36     }
37
38     private void error(String funcaoDoErro){
39         if(tag==Tag.EOF) {
40             if(line==-4)
```

```
41                  System.out.println("Arquivo de entrada vazio");
42              return;
43          }
44          System.out.print(funcaoDoErro+";"+" Error(" + line + "): Token
    não esperado:"); //debug
45          tok.imprimeToken(tok);
46          while(tag!=Tag.EOF)
47              advance();
48      }
49
50      /* Seta o tipo basico com que se esta trabalhando em uma expressao
    de atribuicao */
51      public void setCurType(int tipo){
52          curType = tipo;
53      }
54
55      public void resetResultExprType(){
56          resultExprType = Tag.VOID;
57      }
58
59      private void eat(int t){
60          if(tag==t){
61              //Checa se a tag e um tipo basico
62              if(tag == Tag.INT || tag == Tag.STR){
63                  setCurType(tag);
64              }
65              //Caso ; deve resetar o resultado esperado de uma expressao
66              if(tag == Tag.PV){
67                  resetResultExprType();
68              }
69              System.out.print("Token Consumido("+line+"): ");
70              tok.imprimeToken(tok);
71              advance();
72          }
73          else { error("Error in: eat"); }
74      }
75
76      private void program(){
77          switch(tag) {
78              //G:: program ::= init [decl-list] begin stmt-list stop
79              case Tag.INIT:
80                  eat(Tag.INIT); declList();
81                  if (tag == Tag.BEGIN) {
82                      eat(Tag.BEGIN); stmtList();
83
84                  } else if (tag == Tag.EOF) {
85                      System.out.println("Fim de arquivo inesperado.");
```

```
86
87                  } else {
88                      eat(Tag.STOP);
89                  }
90                  break;
91              default:
92                  error("Error in: program");
93          }
94      }
95
96      private void declList(){
97          //G:: decl-list ::= decl ";" { decl ";"}
98          decl();
99          switch(tag) {
100             case Tag.PV:
101                 eat(Tag.PV);
102                 if(tag == Tag.BEGIN){
103                     break;
104                 }else{
105                     declList();
106                 }
107                 break;
108             default:
109                 error("Error in: declList");
110         }
111     }
112
113     private void decl(){
114         identList();
115         switch(tag) {
116             //G:: decl ::= ident-list is type
117             case Tag.IS:
118                 eat(Tag.IS); type();
119                 break;
120             case Tag.VRG:
121                 eat(Tag.VRG); decl();
122                 break;
123             default:
124                 error("Error in: decl");
125         }
126     }
127
128     private void identList(){
129         switch(tag) {
130             //G:: ident-list ::= identifier {"," identifier}
131             case Tag.ID:
132                 eat(Tag.ID);
```

```
133                 break;
134             case Tag.VRG:
135                 eat(Tag.VRG); eat(Tag.ID);
136                 break;
137             default:
138                 error("Error in: identList");
139         }
140     }
141 
142     private void type(){
143         switch(tag) {
144             //G:: type ::= int
145             case Tag.INT:
146                 eat(Tag.INT);
147                 break;
148             //G:: type ::= string
149             case Tag.STR:
150                 eat(Tag.STR);
151                 break;
152             //G:: type ::= real
153             case Tag.REAL:
154                 eat(Tag.REAL);
155                 break;
156             default:
157                 error("Error in: type");
158         }
159     }
160 
161     private void stmtList(){
162         switch(tag) {
163             //G:: stmt-list ::= stmt ";" { stmt ";" }
164             case Tag.IF:
165                 stmt(); stmtList();
166                 break;
167             case Tag.ID:
168             case Tag.DO:
169             case Tag.READ:
170             case Tag.WRITE:
171                 stmt(); eat(Tag.PV); stmtList();
172                 break;
173             case Tag.WHILE:
174                 break;
175             case Tag.STOP:
176                 break;
177             case Tag.END:
178                 break;
179             default:
```

```
180                 error("Error in: stmtList");
181             }
182         }
183
184         private void stmt(){
185             switch(tag) {
186                 //G:: stmt ::= assign-stmt
187                 case Tag.ID:
188                     assignStmt();
189                     break;
190                 //G:: stmt ::= if-stmt
191                 case Tag.IF:
192                     ifStmt();
193                     break;
194                 //G:: stmt ::= while-stmt
195                 case Tag.DO:
196                     doStmt();
197                     break;
198                 //G:: stmt ::= read-stmt
199                 case Tag.READ:
200                     readStmt();
201                     break;
202                 //G:: stmt ::= write-stmt
203                 case Tag.WRITE:
204                     writeStmt();
205                     break;
206                 default:
207                     error("Error in: stmt");
208             }
209         }
210
211         private void assignStmt(){
212             switch(tag) {
213                 //G:: assign-stmt ::= identifier ":=" simple_expr
214                 case Tag.ID:
215                     eat(Tag.ID); eat(Tag.PPV); simpleExpr();
216                     break;
217                 default:
218                     error("Error in: assignStmt");
219             }
220         }
221
222         private void ifStmt(){
223             switch(tag) {
224                 //G:: if-stmt ::= if "(" condition ")" begin stmt-list end
        else begin stmt-list end
225                 case Tag.IF:
```

```
226                 eat(Tag.IF); eat(Tag.AP); condition(); eat(Tag.FP); eat(
    Tag.BEGIN); stmtList(); eat(Tag.END);
227
228             case Tag.ELSE:
229                 eat(Tag.ELSE); eat(Tag.BEGIN); stmtList(); eat(Tag.END);
230                 break;
231             default:
232                 error("Error in: ifStmt");
233         }
234     }
235
236     private void doStmt(){
237         switch(tag) {
238             //G:: do-stmt ::= do stmt-list do-suffix.
239             case Tag.DO:
240                 eat(Tag.DO); stmtList(); doSufix();
241                 break;
242             default:
243                 error("Error in: doStmt");
244         }
245     }
246
247     private void doSufix(){
248         switch(tag) {
249             //G:: stmt-sufix ::= while "(" condition ")"
250             case Tag.WHILE:
251                 eat(Tag.WHILE); eat(Tag.AP); condition(); eat(Tag.FP);
252                 break;
253             default:
254                 error("Error in: doSufix");
255         }
256     }
257
258     private void readStmt(){
259         switch(tag) {
260             //G:: read-stmt ::= read "(" identifier ")"
261             case Tag.READ:
262                 eat(Tag.READ); eat(Tag.AP); eat(Tag.ID); eat(Tag.FP);
263                 break;
264             default:
265                 error("Error in: redStmt");
266         }
267     }
268
269     private void writeStmt(){
270         switch(tag) {
271             //G:: write-stmt ::= write "(" writable ")"
```

```
272             case Tag.WRITE:
273                 eat(Tag.WRITE); eat(Tag.AP); writable(); eat(Tag.FP);
274                 break;
275             default:
276                 error("Error in: writeStmt");
277         }
278     }
279 
280     private void writable() {
281         simpleExpr();
282     }
283 
284     private void condition(){
285         switch(tag) {
286             //G:: expression ::= simple-expr | simple-expr relop simple-
    expr
287             case Tag.ID:
288             case Tag.NUM:
289             case Tag.LIT:
290             case Tag.AP:
291             case Tag.NOT:
292             case Tag.MIN:
293                 simpleExpr();
294                 switch(tag){
295                     case Tag.GT:
296                     case Tag.LT:
297                     case Tag.GE:
298                     case Tag.LE:
299                     case Tag.NE:
300                     case Tag.EQ:
301                         relop(); simpleExpr();
302                         break;
303                     case Tag.FP:
304                         break;
305                     default:
306                         error("Error in: condition, 'simple-expr relop
    simple-expr'");
307                 }
308                 break;
309             default:
310                 error("Error in: condition, 'simple-expr'");
311         }
312     }
313 
314     private void simpleExpr(){
315         switch(tag) {
316             //G:: simple-expr ::= term | simple-expr addop term
```

```
317             case Tag.ID:
318             case Tag.NUM:
319             case Tag.LIT:
320             case Tag.AP:
321             case Tag.NOT:
322             case Tag.MIN:
323                 term();
324                 switch(tag){
325                     case Tag.MIN:
326                     case Tag.SUM:
327                         addop(); term();
328                         break;
329                     case Tag.PV:
330                     case Tag.FP:
331                     case Tag.GT:
332                     case Tag.LT:
333                     case Tag.GE:
334                     case Tag.LE:
335                     case Tag.NE:
336                     case Tag.EQ:
337                         break;
338                     default:
339                         error("Error in: simpleExpr, 'simple-expr addop
    term'");
340                     }
341                 break;
342             default:
343                 error("Error in: simpleExpr, 'term'");
344         }
345     }
346 
347     private void term(){
348         switch(tag) {
349             //G:: term ::= factor-a | term mulop factor-a
350             case Tag.ID:
351             case Tag.NUM:
352             case Tag.LIT:
353             case Tag.AP:
354             case Tag.NOT:
355             case Tag.MIN:
356                 factorA(); term();
357                 break;
358             case Tag.PV:
359             case Tag.FP:
360             case Tag.SUM:
361             case Tag.GT:
362                 break;
```

```
363             case Tag.MUL:
364             case Tag.DIV:
365             case Tag.AND:
366                 mulop(); factorA();
367                 break;
368             default:
369                 error("Error in: termA");
370         }
371     }
372 
373 
374     private void factorA(){
375         switch(tag) {
376             //G:: factor-a ::= factor
377             case Tag.ID:
378             case Tag.NUM:
379             case Tag.LIT:
380             case Tag.AP:
381                 factor();
382                 break;
383             //G:: factor-a ::= not factor
384             case Tag.NOT:
385                 eat(Tag.NOT); factor();
386                 break;
387             //G:: factor-a ::= "-"  factor
388             case Tag.MIN:
389                 eat(Tag.MIN); factor();
390                 break;
391             default:
392                 error("Error in: factorA");
393         }
394     }
395 
396     private void factor(){
397         switch(tag) {
398             //G:: factor ::= identifier
399             case Tag.ID:
400                 eat(Tag.ID);
401                 break;
402             //G:: factor ::= constant
403             case Tag.NUM:
404             case Tag.LIT:
405                 constant();
406                 break;
407             //G:: factor ::= "(" expression ")"
408             case Tag.AP:
409                 eat(Tag.AP); condition(); eat(Tag.FP);
```

```
410                 break;
411             default:
412                 error("Error in: factor");
413         }
414     }
415
416     private void relop(){
417         switch(tag) {
418             //G:: relop ::= "="
419             case Tag.EQ:
420                 eat(Tag.EQ);
421                 break;
422             //G:: relop ::= ">"
423             case Tag.GT:
424                 eat(Tag.GT);
425                 break;
426             //G:: relop ::= "<"
427             case Tag.LT:
428                 eat(Tag.LT);
429                 break;
430             //G:: relop ::= "<>"
431             case Tag.NE:
432                 eat(Tag.NE);
433                 break;
434             //G:: relop ::= ">="
435             case Tag.GE:
436                 eat(Tag.GE);
437                 break;
438             //G:: relop ::= "<="
439             case Tag.LE:
440                 eat(Tag.LE);
441                 break;
442             default:
443                 error("Error in: relop");
444         }
445     }
446
447     private void addop(){
448         switch(tag) {
449             //G:: addop ::= "+"
450             case Tag.SUM:
451                 eat(Tag.SUM);
452                 break;
453             //G:: addop ::= "-"
454             case Tag.MIN:
455                 eat(Tag.MIN);
456                 break;
```

```
457             //G:: addop ::= "OR"
458             case Tag.OR:
459                 eat(Tag.OR);
460                 break;
461             default:
462                 error("Error in: addop");
463         }
464     }
465 
466     private void mulop(){
467         switch(tag) {
468             //G:: mulop ::= "*"
469             case Tag.MUL:
470                 eat(Tag.MUL);
471                 break;
472             //G:: mulop ::= "/"
473             case Tag.DIV:
474                 eat(Tag.DIV);
475                 break;
476             //G:: mulop ::= "AND"
477             case Tag.AND:
478                 eat(Tag.AND);
479                 break;
480             default:
481                 error("Error in: mulop");
482         }
483     }
484 
485     private void constant() {
486         switch (tag) {
487             //G:: constant ::= integer_const
488             case Tag.NUM:
489                 eat(Tag.NUM);
490                 break;
491             //G:: constant ::= literal
492             case Tag.LIT:
493                 eat(Tag.LIT);
494                 break;
495             default:
496                 error("Error in: constant");
497         }
498     }
499 }
```

Codigos/Parser.java

9. SUGESTÕES PARA TRABALHOS FUTUROS

Nesta etapa, finalizamos o processo de analise Sintática do Compilador (Etapa 2),assim, fica necessário implementar na próxima etapa o analisador Semântico e o Gerador de código. Para as próximas etapas, fica sugerido que devemos implementar a recuperação de Erros para que seja possível implementar um compilador mais eficiente e próximo da realidade.

10. REFERÊNCIAS

10.1. LIVROS

A. V. Aho, R. Sethi, J. D. Ullman Compiladores: Princípios, técnicas e ferramentas LTC - Livros Técnicos e Científicos Editora, 2013

10.2. PROGRAMAS

- Apache Netbeans
- Visual Studio
- Microsoft Office Excel

10.3. SITE

- Site : Overleaf Para criação do relatório