Trabalho Compiladores

Analisador Sintático - pasC

Professor: Gustavo Fernandes

Integrantes:
Marcos Magno de Carvalho
Marcos Vinicius Gonçalves Cunha

Package compiladores _____

- **compiler.syntatic**
  - **Classe**
    - **RunSyntatic**
      Essa classe iniciar o objeto Lexer e Parser.
      Atributos:
        lexer (object) : Um objeto da classe Lexer
        parser (object) : Um objeto da classe Parser.

      - **Métodos**
        - main()
          Esse método é responsável por iniciar o parser.

  - **Classe**
    - **Parser**
      Essa classe implementa o Parser
      Atributos:
        lexer (object) : Um objeto da classe Lexer.
        tag (object) : Um objeto da classe Tag_Type.
        token (String): Uma String que representa um token.

      - **Métodos**
        - advance()
          Esse método é responsável solicitar um novo token.
          Atributos:
            token (String): Uma String que representa um token.

        - sinaliza_erro(mensagem)
          Esse método é responsável por sinalizar erros encontrados.
          Parâmetros:
            mensagem (string): Um mensagem de erro

        - eat(recv_token)
          Esse método consumo um token.
          Parâmetros:
            recv_token (String): Uma String que representa um token.

        - start_parse()
          Esse método é responsável por consumir o primeiro token, que obrigatoriamente tem que ser um KW_PROGRAM.

- prog()
  Esse método é responsável por produzir o não terminal prog:
  prog -> "program" "id" body(1)

- body()
  Esse método é responsável por produzir o não terminal body:
  body -> decl-list "{" stmt-list "}" (2)

- decl_list()
  Esse método é responsável por produzir o não terminal decl_list:
  decl_lit - > decl ";" decl-list (3) | vazio(4)

- decl()
  Esse método é responsável por produzir o não terminal decl:
  decl-> type id-list (5)

- type()
  Esse método é responsável por produzir o não terminal type:
  type -> "num" (6) | "char" (7)

- id_list()
  Esse método é responsável por produzir o não terminal id_list:
  id-list -> "id" id_list' (8)

- id_list_linha()
  Esse método é responsável por produzir o não terminal id_list_linha:
  id-list' -> "," id_list (9)

- stmt_list()
  Esse método é responsável por produzir o não terminal stmt_list:
  stmp-list -> stmt ";" stmt-list (11) | vazio (12)

- stmt()
  Esse método é responsável por produzir o não terminal stmt:
  stmt -> assing-stmt (13) | if-stmt (14) | while-stmt (15) | read-stmt (16) | write-stmt (17)

- assing_stmt()
  Esse método é responsável por produzir o não terminal assing_stmt:
  assing-stmt -> "id" "=" simple_exprt (18)

- if_stmt()
  Esse método é responsável por produzir o não terminal if_stmt:
  if-stmt -> "if" "(" condition ")" "{" stmt-list "}" if_stmt' (19)

- if_stmt_linha()
  Esse método é responsável por produzir o não terminal if_stmt_linha:
  if-stmt_linha -> "else" "{" stmt-list "}" (20) | vazio (21)

- while_stmt()
  Esse método é responsável por produzir o não terminal while_stmt:
  while-stmt → stmt-prefix “{“ stmt-list “}” (23)

- stmt_prefix()
  Esse método é responsável por produzir o não terminal stmt_prefix:
  "while" "(" condition ")" (24)

- read_stmt()
  Esse método é responsável por produzir o não terminal read_stmt:
  "read" "id" (25)

- write_stmt()
  Esse método é responsável por produzir o não terminal write_stmt:
  write" writable

- writable()
  Esse método é responsável por produzir o não terminal writable:
  writable-> simple-expr (27) | "literal" (28)

- expression()
  Esse método é responsável por produzir o não terminal expression:
  expression-> simple-expr expression_linha (29)

- expression_linha()
  Esse método é responsável por produzir o não terminal expression_linha:
  expression_linha-> relop expression (30) | vazio (31)

- simple_expr_linha()
  Esse método é responsável por produzir o não terminal simple_expr_linha:
  simple_expr_linha -> term simple_expr (33) | vazio (34)

- term()
  Esse método é responsável por produzir o não terminal term:
  term -> factor_a term_linha (35)

- term_linha()
  Esse método é responsável por produzir o não terminal term_linha:
  term_linha -> mulop factor_a term_linha (36) | vazio (37)

- factor_a()
  Esse método é responsável por produzir o não terminal factor_a:
  factor_a -> factor (38) | not factor (39)

- factor()
  Esse método é responsável por produzir o não terminal factor:
  factor -> “id” (40) | constant (41) | “(“ expression “)” (42)

Report

Programa Correto 1:

Tokens

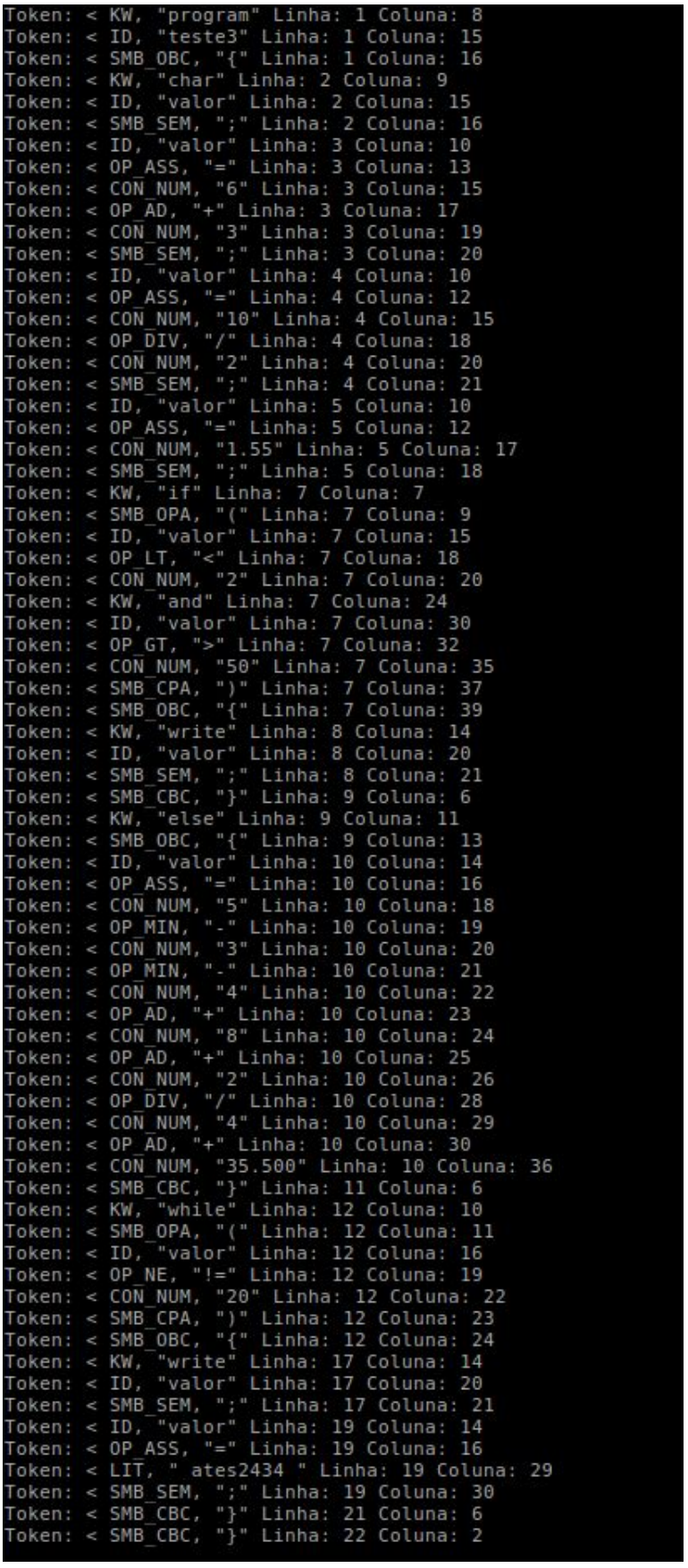
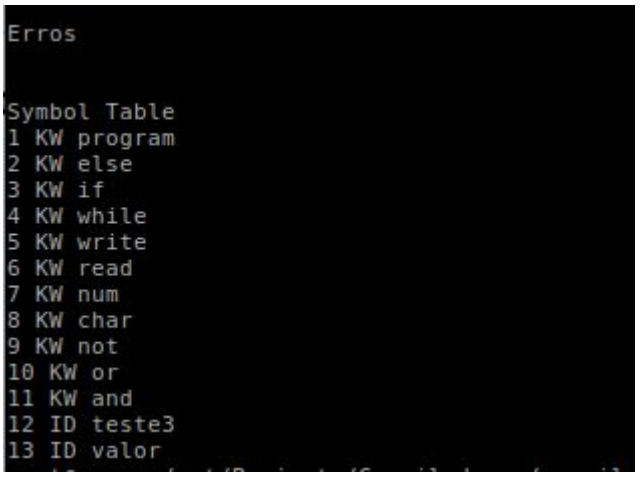


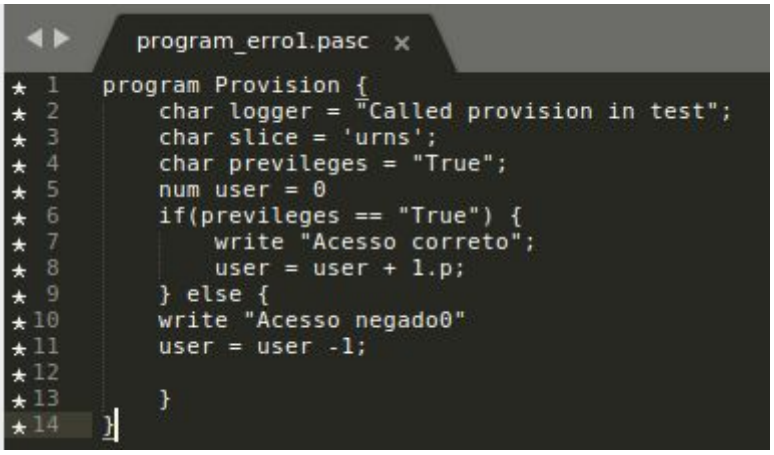
Tabela de Símbolos e Erros (nenhum encontrado).



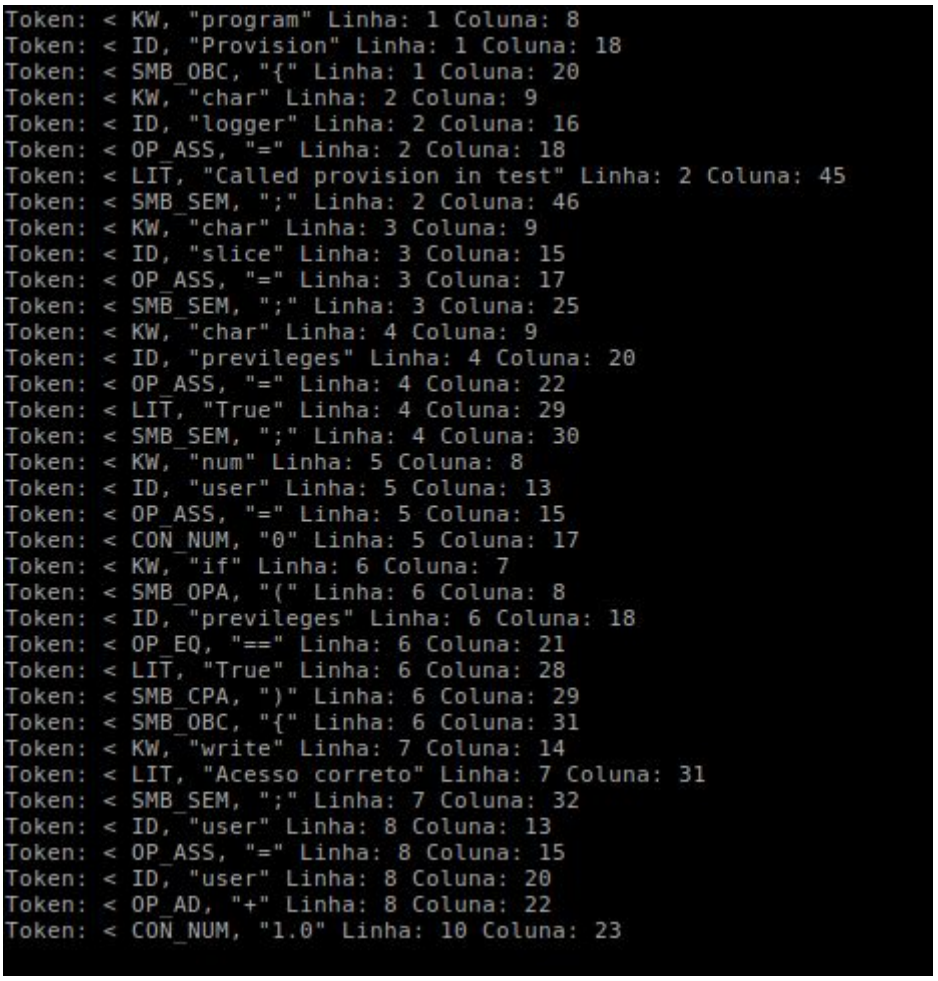
Programa com Erro 1

Este programa teve como objetivo testar o modo pânico entendido em sala de aula.

Na linha 8 coluna 23 existe um erro de formação de *digit*. Após detectar o erro o programa continua lendo os demais caracteres até encontrar um inteiro, o que de fato estava esperando receber. Isso acontece quando encontra o 0 após a palavra negado, na linha 10 coluna 26. Posteriormente é encontrado outro erro, na linha 10 coluna 27, quando se tenta formar um literal, na qual deve-se ser feito entre aspa duplas e não pode ter quebra de linha. Existe ainda o erro para se formar constante char, na linha 3, coluna 24.



Tokens


Token: < KW, "program" Linha: 1 Coluna: 8
Token: < ID, "Provision" Linha: 1 Coluna: 18
Token: < SMB_OBC, "{" Linha: 1 Coluna: 20
Token: < KW, "char" Linha: 2 Coluna: 9
Token: < ID, "logger" Linha: 2 Coluna: 16
Token: < OP_ASS, "=" Linha: 2 Coluna: 18
Token: < LIT, "Called provision in test" Linha: 2 Coluna: 45
Token: < SMB_SEM, ";" Linha: 2 Coluna: 46
Token: < KW, "char" Linha: 3 Coluna: 9
Token: < ID, "slice" Linha: 3 Coluna: 15
Token: < OP_ASS, "=" Linha: 3 Coluna: 17
Token: < SMB_SEM, ";" Linha: 3 Coluna: 25
Token: < KW, "char" Linha: 4 Coluna: 9
Token: < ID, "previleges" Linha: 4 Coluna: 20
Token: < OP_ASS, "=" Linha: 4 Coluna: 22
Token: < LIT, "True" Linha: 4 Coluna: 29
Token: < SMB_SEM, ";" Linha: 4 Coluna: 30
Token: < KW, "num" Linha: 5 Coluna: 8
Token: < ID, "user" Linha: 5 Coluna: 13
Token: < OP_ASS, "=" Linha: 5 Coluna: 15
Token: < CON_NUM, "0" Linha: 5 Coluna: 17
Token: < KW, "if" Linha: 6 Coluna: 7
Token: < SMB_OPA, "(" Linha: 6 Coluna: 8
Token: < ID, "previleges" Linha: 6 Coluna: 18
Token: < OP_EQ, "==" Linha: 6 Coluna: 21
Token: < LIT, "True" Linha: 6 Coluna: 28
Token: < SMB_CPA, ")" Linha: 6 Coluna: 29
Token: < SMB_OBC, "{" Linha: 6 Coluna: 31
Token: < KW, "write" Linha: 7 Coluna: 14
Token: < LIT, "Acesso correto" Linha: 7 Coluna: 31
Token: < SMB_SEM, ";" Linha: 7 Coluna: 32
Token: < ID, "user" Linha: 8 Coluna: 13
Token: < OP_ASS, "=" Linha: 8 Coluna: 15
Token: < ID, "user" Linha: 8 Coluna: 20
Token: < OP_AD, "+" Linha: 8 Coluna: 22
Token: < CON_NUM, "1.0" Linha: 10 Coluna: 23

Erros

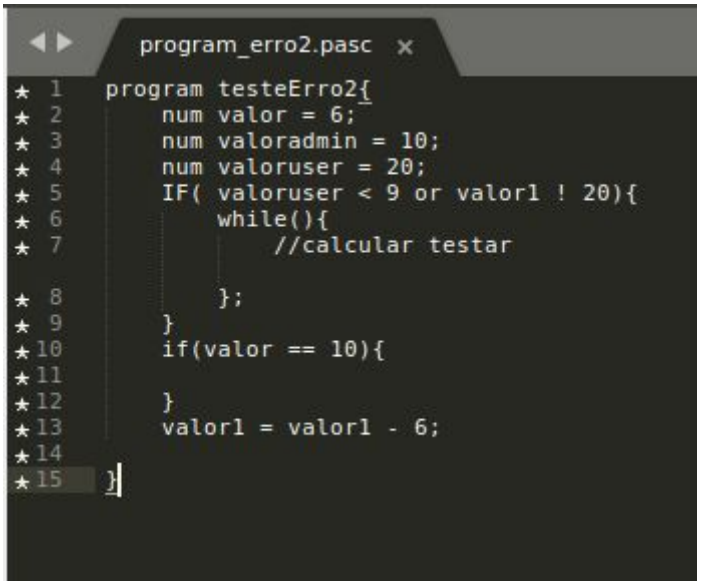
```
Erros
literal must contain only one character Linha: 3 Coluna: 24
Error encountered. Expected an integer  Linha: 8 Coluna: 26
Error encountered. Expected an integer  Linha: 8 Coluna: 27
Error encountered. Expected an integer  Linha: 9 Coluna: 1
Error encountered. Expected an integer  Linha: 9 Coluna: 5
Error encountered. Expected an integer  Linha: 9 Coluna: 6
Error encountered. Expected an integer  Linha: 9 Coluna: 7
Error encountered. Expected an integer  Linha: 9 Coluna: 8
Error encountered. Expected an integer  Linha: 9 Coluna: 9
Error encountered. Expected an integer  Linha: 9 Coluna: 10
Error encountered. Expected an integer  Linha: 9 Coluna: 11
Error encountered. Expected an integer  Linha: 9 Coluna: 12
Error encountered. Expected an integer  Linha: 9 Coluna: 13
Error encountered. Expected an integer  Linha: 10 Coluna: 1
Error encountered. Expected an integer  Linha: 10 Coluna: 5
Error encountered. Expected an integer  Linha: 10 Coluna: 6
Error encountered. Expected an integer  Linha: 10 Coluna: 7
Error encountered. Expected an integer  Linha: 10 Coluna: 8
Error encountered. Expected an integer  Linha: 10 Coluna: 9
Error encountered. Expected an integer  Linha: 10 Coluna: 10
Error encountered. Expected an integer  Linha: 10 Coluna: 11
Error encountered. Expected an integer  Linha: 10 Coluna: 12
Error encountered. Expected an integer  Linha: 10 Coluna: 13
Error encountered. Expected an integer  Linha: 10 Coluna: 14
Error encountered. Expected an integer  Linha: 10 Coluna: 15
Error encountered. Expected an integer  Linha: 10 Coluna: 16
Error encountered. Expected an integer  Linha: 10 Coluna: 17
Error encountered. Expected an integer  Linha: 10 Coluna: 18
Error encountered. Expected an integer  Linha: 10 Coluna: 19
Error encountered. Expected an integer  Linha: 10 Coluna: 20
Error encountered. Expected an integer  Linha: 10 Coluna: 21
Error encountered. Expected an integer  Linha: 10 Coluna: 22
Error encountered. Expected an integer  Linha: 10 Coluna: 23
Error encountered. Expected an integer  Linha: 10 Coluna: 24
Error encountered. Expected an integer  Linha: 10 Coluna: 25
Can not have line break : Linha: 11 Coluna: 1
Can not have line break : Linha: 12 Coluna: 1
Can not have line break : Linha: 13 Coluna: 1
Can not have line break : Linha: 14 Coluna: 1
Invalid Character expected " Linha: 14 Coluna: 3
```

Tabela de Símbolo

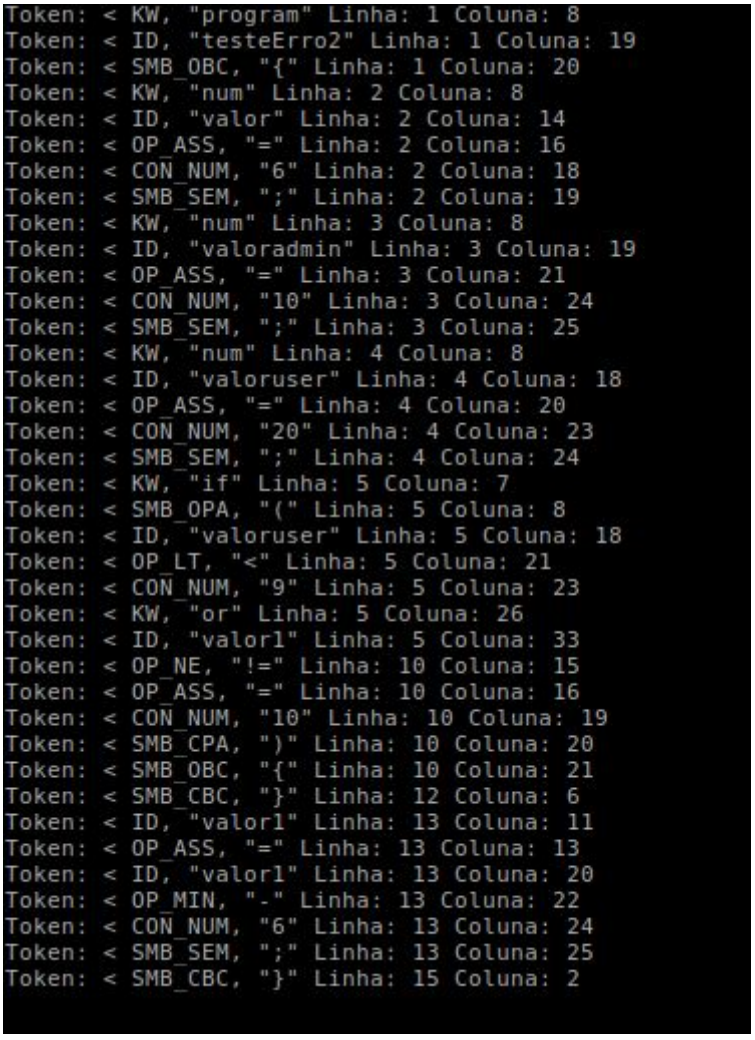
```
Symbol Table
1 KW program
2 KW else
3 KW if
4 KW while
5 KW write
6 KW read
7 KW num
8 KW char
9 KW not
10 KW or
11 KW and
12 ID Provision
13 ID logger
14 ID slice
15 ID previleges
16 ID user
```

Programa Erro 2

Neste programa testou-se o modo pânico na formação do operador != . Quando não se encontra o sinal de igual após o !, na linha 5 coluna 34, o analisador léxico ignora os demais caracter, até encontrar o sinal esperado (=), o que acontece na linha 10 coluna 15.



Tokens:



Erros:

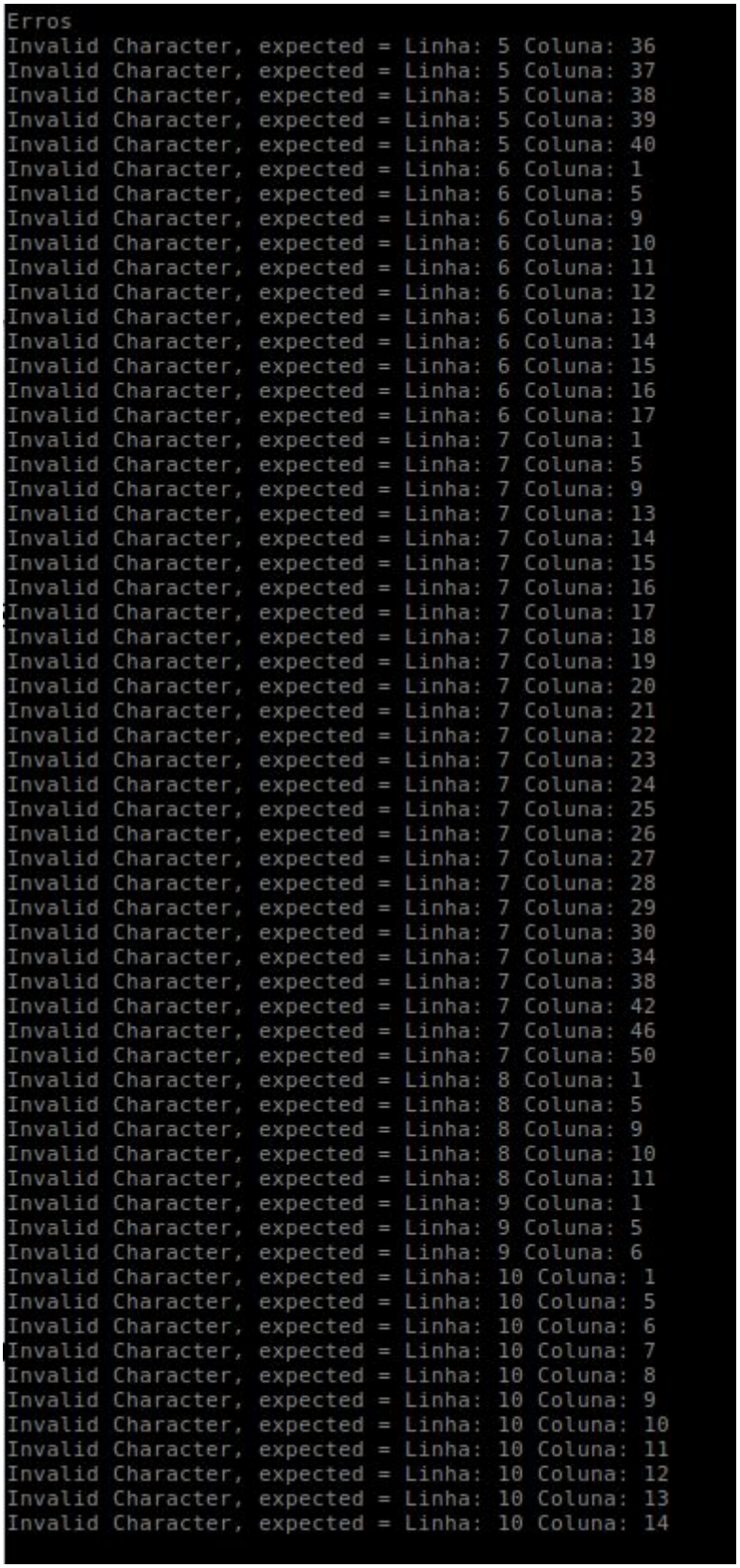


Tabela de Símbolo

```
Symbol Table
1 KW program
2 KW else
3 KW if
4 KW while
5 KW write
6 KW read
7 KW num
8 KW char
9 KW not
10 KW or
11 KW and
12 ID testeErro2
13 ID valor
14 ID valoradmin
15 ID valoruser
16 ID valor1
```

Programa Erro 3

Com este programa visa testar a formação de *digit*, construção de literal e comentário de várias linhas.

program_erro3.pasc

```
1  program erro2{
2      valor = 5;
3      if (valor < 5) {
4          valor = 10 * 5;
5          valor = 10.o;
6
7      } else {
8          valor = 5;
9      }
10
11     while(valor > 5) {
12     /* vamos testar o comentario de varias linhas.
13         para ver se funciona mesmo */
14     }
15
16     user = 'marcos'
17
18     if(user == "marcos"){
19     write "Marcos"
20     }else{
21         /* inicia o teste dos comentarios de varias linhas
22             na qual deve-se ignorar ate o fim de arquivo
23     }else if(valor1 != ' teste'){
24
25     else(valor == 'a'){
26
27     }
28
29 }
```

Tokens

```
Token: < KW, "program" Linha: 1 Coluna: 8
Token: < ID, "erro2" Linha: 1 Coluna: 14
Token: < SMB_OBC, "{" Linha: 1 Coluna: 15
Token: < ID, "valor" Linha: 2 Coluna: 10
Token: < OP_ASS, "=" Linha: 2 Coluna: 12
Token: < CON_NUM, "5" Linha: 2 Coluna: 14
Token: < SMB_SEM, ";" Linha: 2 Coluna: 15
Token: < KW, "if" Linha: 3 Coluna: 7
Token: < SMB_OPA, "(" Linha: 3 Coluna: 9
Token: < ID, "valor" Linha: 3 Coluna: 14
Token: < OP_LT, "<" Linha: 3 Coluna: 17
Token: < CON_NUM, "5" Linha: 3 Coluna: 19
Token: < SMB_CPA, ")" Linha: 3 Coluna: 20
Token: < SMB_OBC, "{" Linha: 3 Coluna: 22
Token: < ID, "valor" Linha: 4 Coluna: 14
Token: < OP_ASS, "=" Linha: 4 Coluna: 16
Token: < CON_NUM, "10" Linha: 4 Coluna: 19
Token: < OP_MUL, "*" Linha: 4 Coluna: 22
Token: < CON_NUM, "5" Linha: 4 Coluna: 24
Token: < SMB_SEM, ";" Linha: 4 Coluna: 25
Token: < ID, "valor" Linha: 5 Coluna: 14
Token: < OP_ASS, "=" Linha: 5 Coluna: 16
Token: < CON_NUM, "10.5" Linha: 8 Coluna: 18
Token: < SMB_SEM, ";" Linha: 8 Coluna: 19
Token: < SMB_CBC, "}" Linha: 9 Coluna: 6
Token: < KW, "while" Linha: 11 Coluna: 10
Token: < SMB_OPA, "(" Linha: 11 Coluna: 11
Token: < ID, "valor" Linha: 11 Coluna: 16
Token: < OP_GT, ">" Linha: 11 Coluna: 18
Token: < CON_NUM, "5" Linha: 11 Coluna: 20
Token: < SMB_CPA, ")" Linha: 11 Coluna: 21
Token: < SMB_OBC, "{" Linha: 11 Coluna: 23
Token: < SMB_CBC, "}" Linha: 14 Coluna: 6
Token: < ID, "user" Linha: 16 Coluna: 9
Token: < OP_ASS, "=" Linha: 16 Coluna: 11
Token: < KW, "if" Linha: 18 Coluna: 7
Token: < SMB_OPA, "(" Linha: 18 Coluna: 8
Token: < ID, "user" Linha: 18 Coluna: 12
Token: < OP_EQ, "==" Linha: 18 Coluna: 15
Token: < LIT, "marcos" Linha: 18 Coluna: 24
Token: < SMB_CPA, ")" Linha: 18 Coluna: 25
Token: < SMB_OBC, "{" Linha: 18 Coluna: 26
Token: < KW, "write" Linha: 19 Coluna: 10
Token: < LIT, "Marcos" Linha: 19 Coluna: 19
Token: < SMB_CBC, "}" Linha: 20 Coluna: 6
Token: < KW, "else" Linha: 20 Coluna: 10
Token: < SMB_OBC, "{" Linha: 20 Coluna: 11
```

Erros e Tabela de Símbolo

```
Erros
Error encountered. Expected an integer  Linha: 5 Coluna: 21
Error encountered. Expected an integer  Linha: 5 Coluna: 22
Error encountered. Expected an integer  Linha: 6 Coluna: 1
Error encountered. Expected an integer  Linha: 7 Coluna: 1
Error encountered. Expected an integer  Linha: 7 Coluna: 5
Error encountered. Expected an integer  Linha: 7 Coluna: 6
Error encountered. Expected an integer  Linha: 7 Coluna: 7
Error encountered. Expected an integer  Linha: 7 Coluna: 8
Error encountered. Expected an integer  Linha: 7 Coluna: 9
Error encountered. Expected an integer  Linha: 7 Coluna: 10
Error encountered. Expected an integer  Linha: 7 Coluna: 11
Error encountered. Expected an integer  Linha: 7 Coluna: 12
Error encountered. Expected an integer  Linha: 7 Coluna: 13
Error encountered. Expected an integer  Linha: 8 Coluna: 1
Error encountered. Expected an integer  Linha: 8 Coluna: 5
Error encountered. Expected an integer  Linha: 8 Coluna: 9
Error encountered. Expected an integer  Linha: 8 Coluna: 10
Error encountered. Expected an integer  Linha: 8 Coluna: 11
Error encountered. Expected an integer  Linha: 8 Coluna: 12
Error encountered. Expected an integer  Linha: 8 Coluna: 13
Error encountered. Expected an integer  Linha: 8 Coluna: 14
Error encountered. Expected an integer  Linha: 8 Coluna: 15
Error encountered. Expected an integer  Linha: 8 Coluna: 16
Error encountered. Expected an integer  Linha: 8 Coluna: 17
literal must contain only one character Linha: 16 Coluna: 20
Invalid Character expected */ Linha: 29 Coluna: 3
```