

Universidade de São Paulo

Instituto de Ciências Matemáticas e de Computação

SCC0605 - Teoria de Computação e Compiladores
Analisador Léxico

Felipe Tetzner 9897870

Junho
2020

Sumário

Autômatos do Analisador Léxico	3
Autômato para Comentários	4
Autômato para Operadores e Símbolos	5
Autômato para Identificadores	6
Autômato para Números Inteiros e Reais	7
Implementação	8
Estruturas Auxiliares	8
Pares de Buffers com Sentinelas	8
Tabela de Palavras e Símbolos Reservados	9
Resultados	10
Exemplo Apresentado na Especificação	10
Exemplo com Conjunto de Erros Léxicos para a Linguagem P - -	12
Instruções de Compilação	15
Referências	16

Autômatos do Analisador Léxico

O Analisador Léxico é composto por um conjunto de quatro Autômatos independentes. Os Autômatos para reconhecimento de Comentários, Operadores e Símbolos Especiais, Identificadores e Números - Inteiros e Reais - estão representados em Figura 1, Figura 2,

Figura 3 e Figura 4, respectivamente.

A separação em quatro Autômatos distintos é uma escolha arbitrária que torna mais clara a funcionalidade de cada componente do Analisador Léxico. Assim, esses Autômatos poderiam ser unificados em apenas um simplesmente agrupando seus estados iniciais individuais em um estado inicial único. Desta forma, seus estados estão nomeados em uma sequência numérica indicando a natureza componente deles de um todo - Analisador Léxico.

Quando uma cadeia não é reconhecida por nenhum dos Autômatos um erro de caractere inesperado é reportado. Ao colapsar os estados iniciais podemos modelar isto como um estado extra de erro. Esta estratégia não é demonstrada aqui porque a transição que levaria a esse estado de erro envolveria o alfabeto de cada um dos Autômatos independentes, derrotando o propósito de separação dos Autômatos.

Observação: os Autômatos foram desenvolvidos utilizando a ferramenta JFlap. As Máquinas de Moore nesta ferramenta não possuem designação de estados finais. Assim, as saídas e ações dos estados estão determinadas textualmente. Essa escolha foi tomada para deixar de maneira clara quais são os estados finais de cada Autômato que indicam a aceitação de cadeias ou estados de erro.

Autômato para Comentários

O Autômato para Comentários reconhece comentários e consome espaços em branco, tabulações e quebras de linha. O estado 3 representa um estado de erro em que um comentário foi iniciado e não finalizado, indicando um fim inesperado do arquivo analisado. A finalização do arquivo é feita pela leitura de um caractere especial EOF - END OF FILE.

Saídas:

Estado 2: não há

Estado 3: “Erro: fim de arquivo inesperado na linha n”, onde n é a linha em do arquivo em que o erro ocorreu

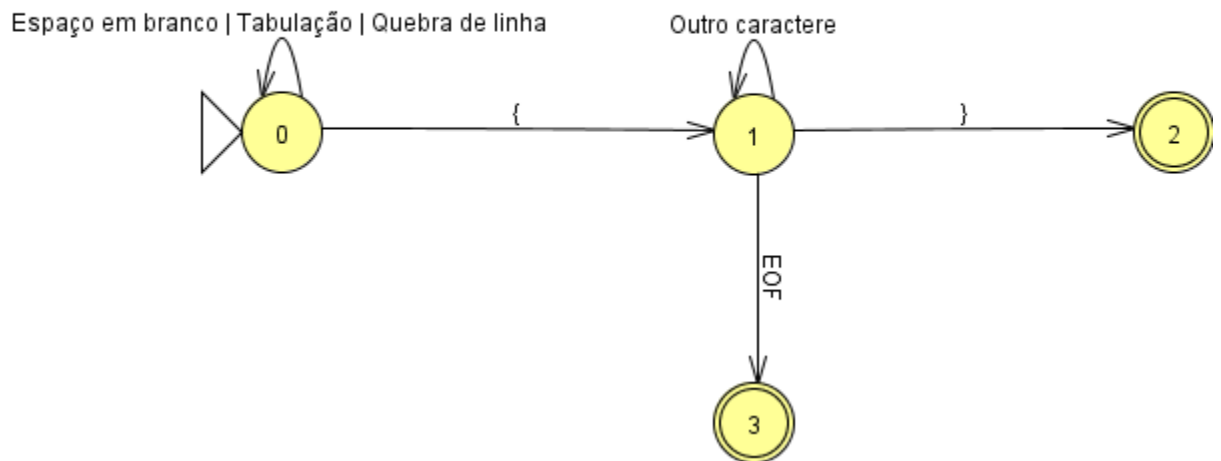


Figure 1. Autômato para Comentários.

Autômato para Operadores e Símbolos

O Autômato para Operadores e Símbolos reconhece os operadores relacionais, matemáticos e símbolos especiais da Linguagem P - . Os estados atingidos através de uma transição “Outro Caractere” - estados 8, 12 e 15 - têm uma ação associada, onde o caractere lido é devolvido à fonte de dados. Esses caracteres são necessários para o reconhecimento de determinados operadores que constituem outros operadores, como ocorre com os operadores “<” e “<=”.

Quando cada estado final é atingido, a cadeia é buscada na Tabela de Palavras e Símbolos Reservados(ver seção de implementação) para obter seu token associado. Essa busca é especialmente importante no estado 9, que condensa diversos operadores e reduz o número de estados do Autômato.

As saídas dos estados finais são os tokens associados aos símbolos especiais

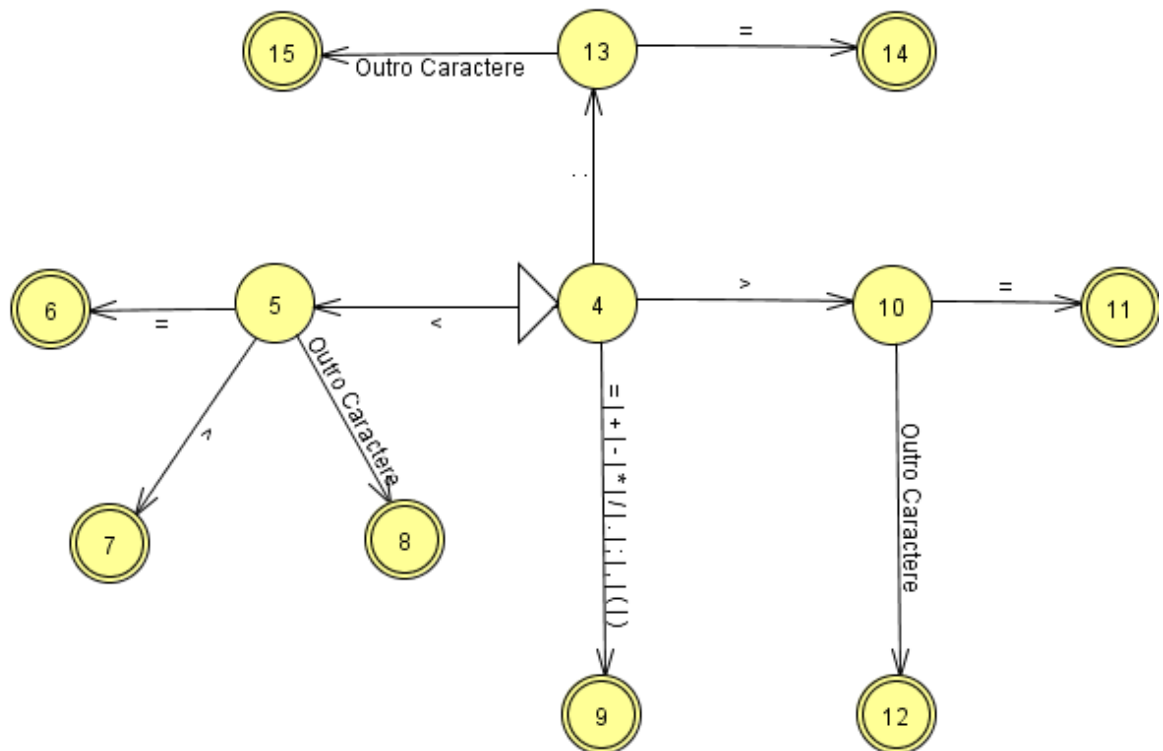


Figure 2. Autômato para Operadores e Símbolos

Autômato para Identificadores

O Autômato para Identificadores reconhece os identificadores e palavras reservadas da Linguagem P - -.

O estado 18 representa o reconhecimento da cadeia e, quando atingido, a cadeia é buscada na Tabela de Palavras e Símbolos Reservados. Se a busca for bem sucedida, a cadeia é retornada e seu token associado é fornecido pela Tabela de Palavras e Símbolos Reservados. Caso contrário, a cadeia é classificada como um Identificador.

O estado 19 representa um estado de erro em que um caractere que não pertence à Linguagem P - - foi lido da fonte de dados. Nesse caso, temos uma má formação de um identificador. O caractere inválido é isolado e a cadeia retornada é a cadeia sem o caractere inválido. Um erro é reportado para esse identificador. O caractere isolado não é reconhecido por nenhum Autômato e é classificado como Caractere Inesperado na próxima chamada do Analisador Léxico.

Assim como no Autômato anterior, a transição “Outro Caractere” leva à devolução deste à fonte de dados. Isto também ocorre na transição entre os estados 17 e 18.

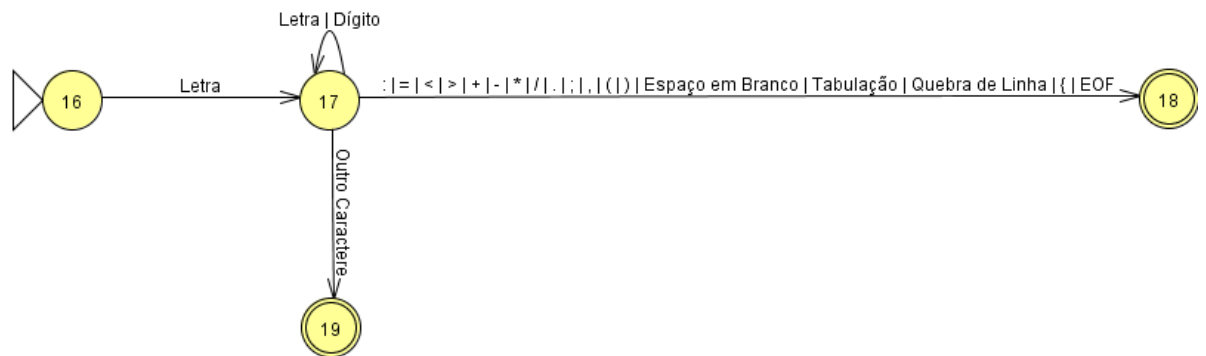


Figure 3. Autômato para Identificadores

Autômato para Números Inteiros e Reais

O Autômato para Números Inteiros e Reais reconhece os números aceitos pela Linguagem P - -. Assim como nos casos anteriores, as transições do tipo “Outro Caractere” levam à devolução do caractere lido à fonte de dados.

O estado 23 é o estado de aceitação de um Número Inteiro, enquanto que os Números Reais são reconhecidos no estado 26.

O estado 27 representa um estado de erro e má formação de um Número Real. Este é isolado e a Análise Léxica continua do caractere após “.”.

Saídas

Estado 23: n, “Número Inteiro”, onde n é a cadeia reconhecida

Estado 26: n, “Número Real”, onde n é a cadeia reconhecida

Estado 27: “Erro: número real mal formado na linha n”, onde n é a linha em que o erro ocorreu

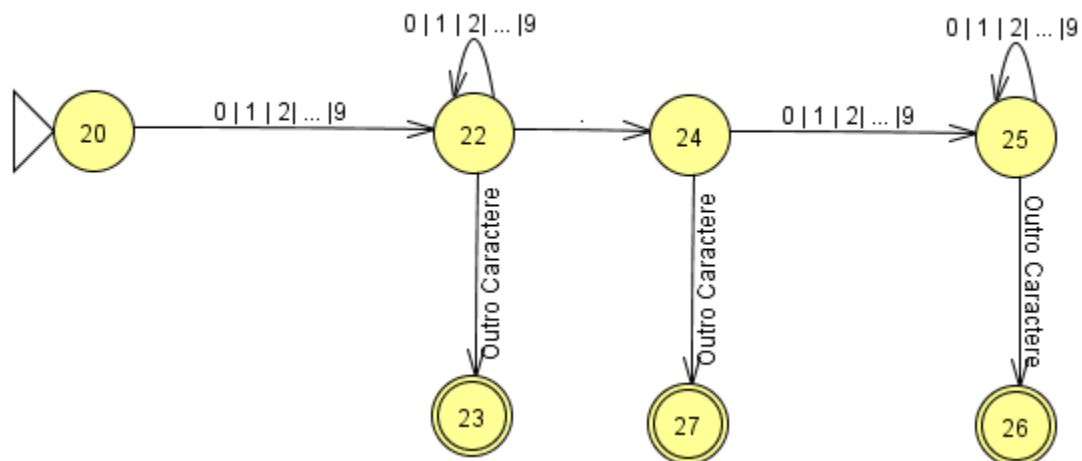


Figure 4. Autômato para Números Inteiros e Reais

Implementação

Como, essencialmente, os quatro Autômatos são apenas um único Autômato, o Analisador Léxico foi implementado como um Autômato e uma única função de transição com os estados implementados como em Figura 1, Figura 2, Figura 3 e Figura 4. Porém, há uma função auxiliar - `falhar()` - que determina qual o estado inicial do próximo Autômato quando a cadeia atual não é reconhecida por um Autômato deles. Esse processo continua até que a cadeia seja rejeitada por todos os Autômatos e, conseqüentemente, rejeitada pelo Analisador Léxico.

Cada estado tem pelo menos um transição associada e, possivelmente, ações a serem executadas, como discutido anteriormente - tratamento de erros, controle de fluxo de dados, busca na Tabela de Palavras e Símbolos Reservados e aceitação de cadeias.

Para esta implementação, foram utilizadas duas estruturas auxiliares para otimizar processos de busca e fluxo de dados. Para tornar as buscas mais eficientes, uma Tabela de Palavras e Símbolos Reservados foi desenvolvida. Além disso, um Buffer para o arquivo de entrada foi criado, reduzindo acessos a disco.

Estruturas Auxiliares

Pares de Buffers com Sentinelas

Como um controle do fluxo de dados é necessário, temos muitos acessos ao arquivo de dados durante a execução das transições do Analisador Léxico.

Para otimizar os processos de leitura e devolução dos caracteres do arquivo de entrada, um Buffer foi implementado. Sua estrutura está representada na Figura 5. Nela temos um par de Buffers delimitados por dois caracteres especiais chamados Sentinelas.

O Apontador adiante `F` indica o próximo caractere a ser lido no Buffer, enquanto que o Apontador de Início indica o começo da cadeia considerada na iteração atual do Analisador Léxico. Em cada leitura obtemos um caractere e o Apontador `F` move em frente.

Ao atingir um Sentinela, o Buffer seguinte é recarregado e o Apontador `F` começa a leitura desse Buffer. Um cuidado especial tem que ser tomado pois, na devolução de caracteres ao Buffer, o Apontador `F` retrocede e não podemos recarregar um Buffer sem ler todo seu conteúdo antes - caso em que `F` retrocede até atingir o Início do Buffer `I` ou o primeiro Sentinela.

Em um estado final, a cadeia retornada é delimitada pelos Apontadores `B` e `F`.

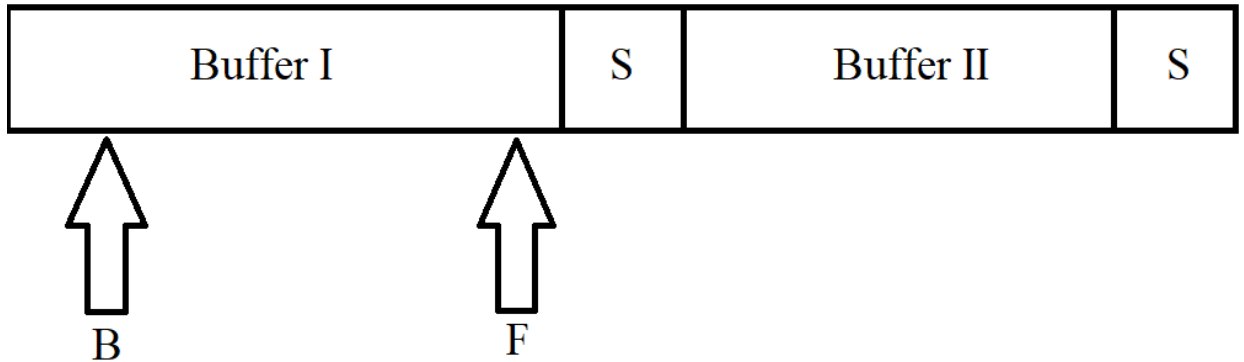


Figure 5. Estrutura do Par de Buffers com Sentinelas

Tabela de Palavras e Símbolos Reservados

A Tabela de Palavras e Símbolos Reservados foi implementada como uma Tabela Hash para otimizar as buscas de símbolos especiais e palavras reservadas. A função Hash desenvolvida mapeia cada um destas cadeias especiais na Tabela Hash univocamente, sem colisões. Porém não há como garantir que outras cadeias definidas pelas regras de formação de Identificadores não tenham o mesmo Valor Hash que uma palavra ou símbolo especial.

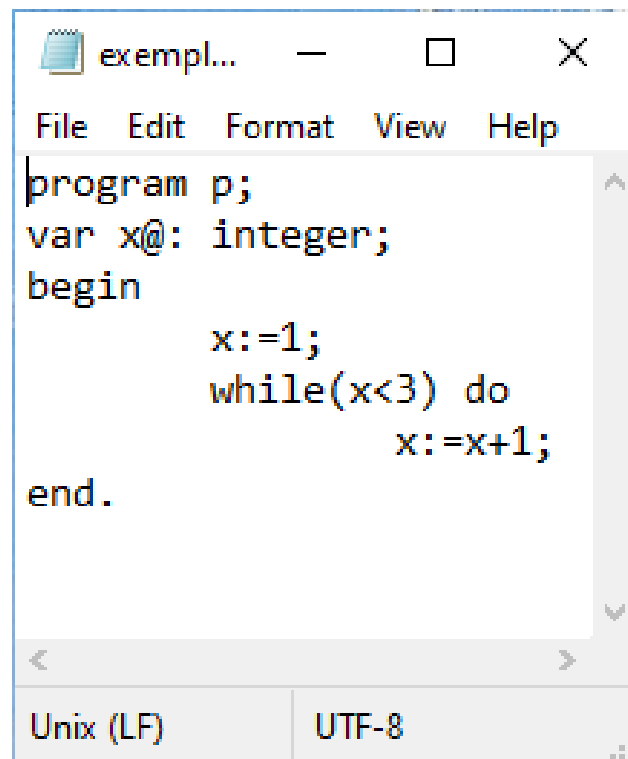
Para resolver isso, cada entrada da Tabela de Palavras e Símbolos Especiais consiste de um Par cadeia token. Assim, se houver algum tipo de colisão de um Identificador com uma palavra ou símbolo especial, basta comparar com esta cadeia especial. Embora não seja uma Função Hash perfeita ela é simples e eficiente devido ao conjunto limitado de palavras e símbolos especiais. Desta forma, temos uma computação de um Valor Hash e uma comparação em detrimento de n comparações para a determinação do token associado à uma cadeia, onde n é o número de palavras e símbolos especiais.

Resultados

A seguir são apresentados dois exemplos de Arquivos de Entrada e seus respectivos Arquivos de Saída, resultados da Análise Léxica.

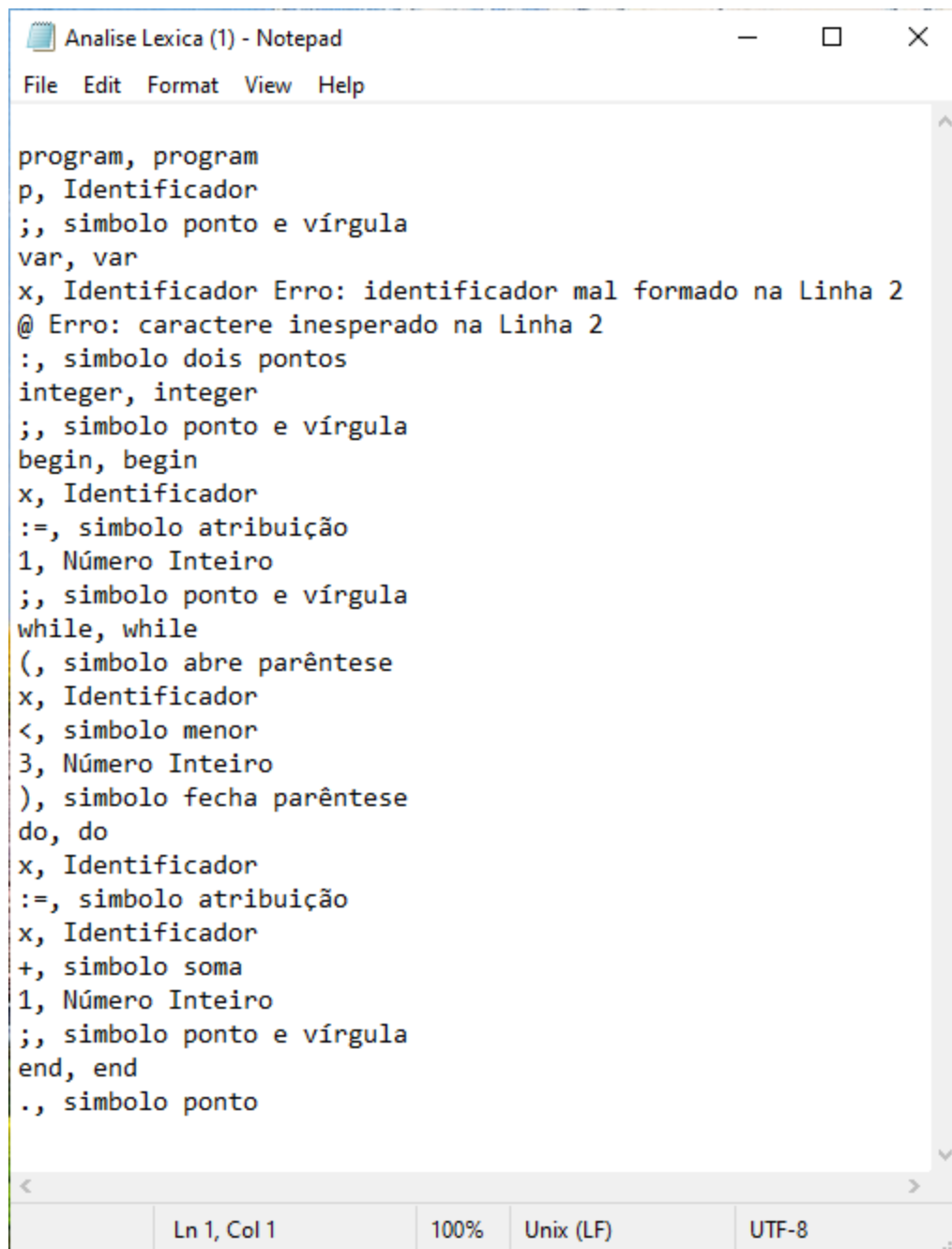
Exemplo Apresentado na Especificação

Este exemplo é o mesmo apresentado na especificação dada do Analisador Léxico. A Figura 6 contém o Arquivo de Entrada e a Figura 7 a saída do Analisador Léxico no Arquivo de Saída.



```
program p;  
var x@: integer;  
begin  
    x:=1;  
    while(x<3) do  
        x:=x+1;  
    end.  
end.
```

Figure 6. Exemplo encontrado na Especificação do Analisador Léxico



```
Analise Lexica (1) - Notepad
File Edit Format View Help

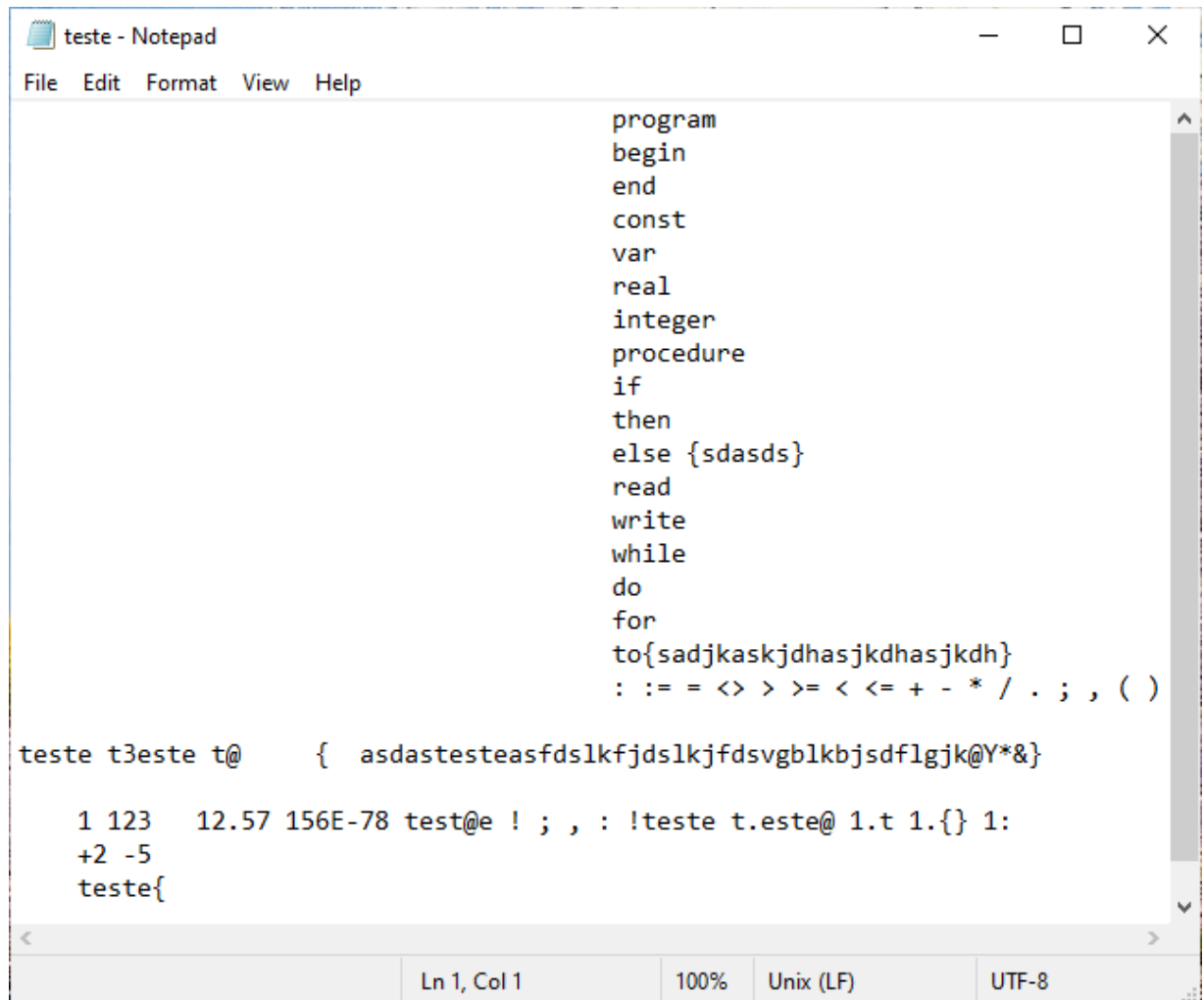
program, program
p, Identificador
;, simbolo ponto e vírgula
var, var
x, Identificador Erro: identificador mal formado na Linha 2
@ Erro: caractere inesperado na Linha 2
:, simbolo dois pontos
integer, integer
;, simbolo ponto e vírgula
begin, begin
x, Identificador
:=, simbolo atribuição
1, Número Inteiro
;, simbolo ponto e vírgula
while, while
(, simbolo abre parêntese
x, Identificador
<, simbolo menor
3, Número Inteiro
), simbolo fecha parêntese
do, do
x, Identificador
:=, simbolo atribuição
x, Identificador
+, simbolo soma
1, Número Inteiro
;, simbolo ponto e vírgula
end, end
., simbolo ponto

Ln 1, Col 1 100% Unix (LF) UTF-8
```

Figure 7. Resultado do Exemplo dado

Exemplo com Conjunto de Erros Léxicos para a Linguagem P - -

Este exemplo demonstra o reconhecimento de palavras e símbolos reservados, espaços em branco, tabulações, quebras de linha e comentários. Além disso, ilustra o tratamento de alguns erros léxicos tais como caracteres inválidos, identificadores e números mal formados e comentários não encerrados. A Figura 8 contém o Arquivo de Entrada e Figura 9 e Figura 10 a saída do Analisador Léxico no Arquivo de Saída.

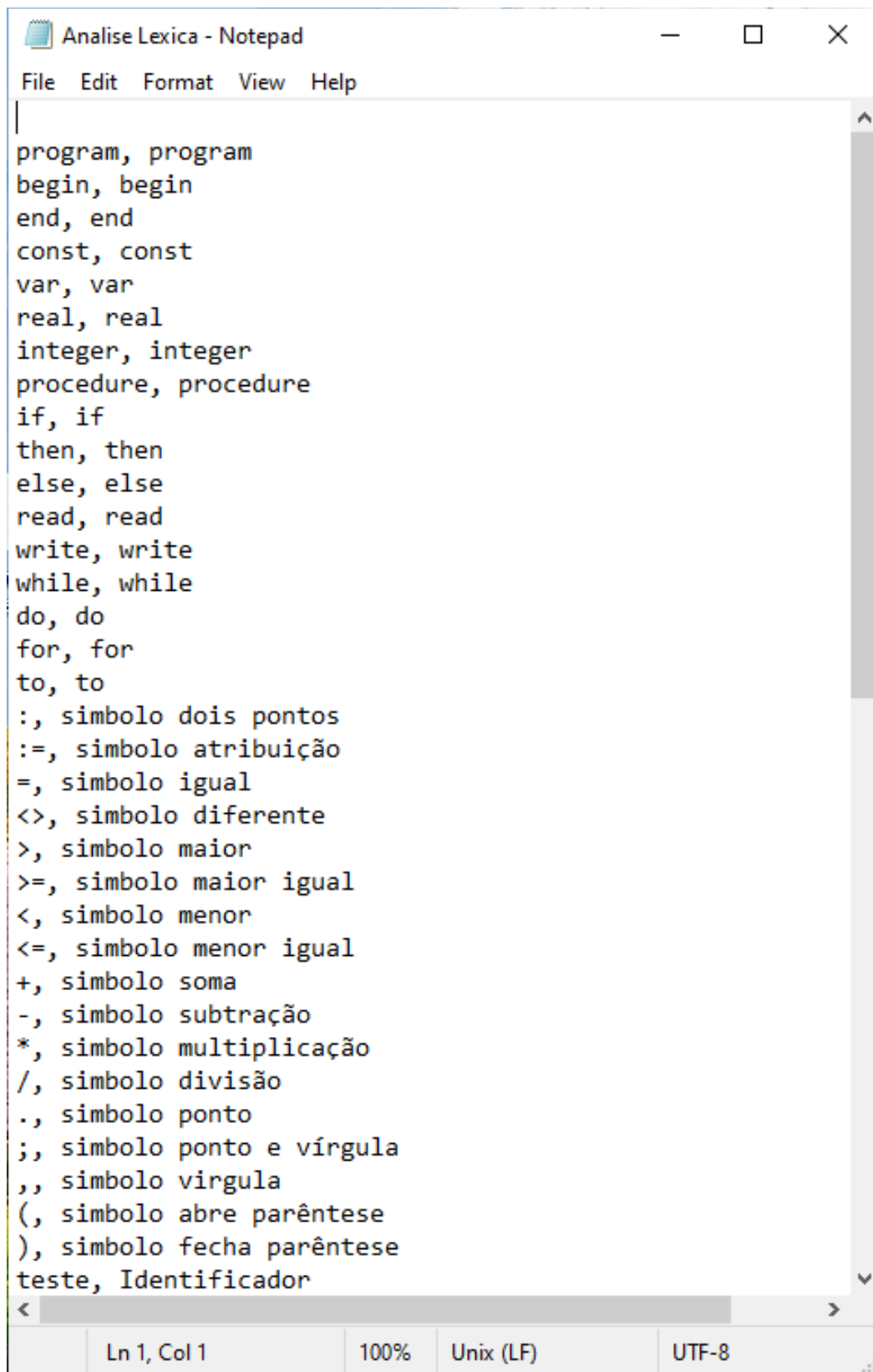


```
program
begin
end
const
var
real
integer
procedure
if
then
else {sdasds}
read
write
while
do
for
to{sadjkaskjdhasjkdhasjkdh}
: := = < > >= < <= + - * / . ; , ( )

teste t3este t@    { asdastesteasfdslkfjdsldkjdsvglkbljdsdflgjk@Y*&}

1 123    12.57 156E-78 test@e ! ; , : !teste t.este@ 1.t 1.{ } 1:
+2 -5
teste{
```

Figure 8. Arquivo com uma conjunto de Erros Léxicos para a Linguagem P - -

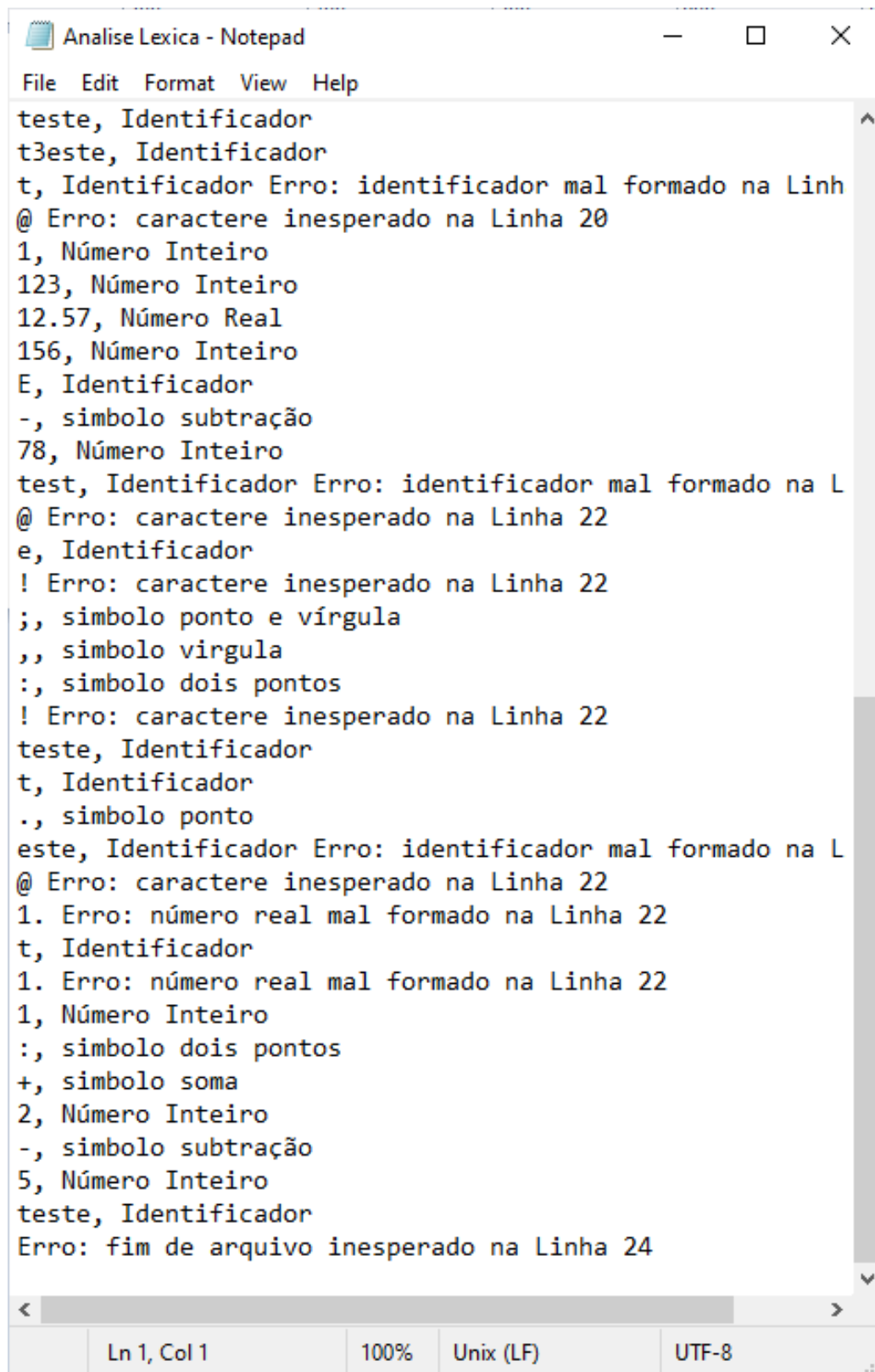


Analise Lexica - Notepad

File Edit Format View Help

```
program, program
begin, begin
end, end
const, const
var, var
real, real
integer, integer
procedure, procedure
if, if
then, then
else, else
read, read
write, write
while, while
do, do
for, for
to, to
:, simbolo dois pontos
:=, simbolo atribuição
=, simbolo igual
<>, simbolo diferente
>, simbolo maior
>=, simbolo maior igual
<, simbolo menor
<=, simbolo menor igual
+, simbolo soma
-, simbolo subtração
*, simbolo multiplicação
/, simbolo divisão
., simbolo ponto
;, simbolo ponto e vírgula
,, simbolo virgula
(, simbolo abre parêntese
), simbolo fecha parêntese
teste, Identificador
```

Ln 1, Col 1 100% Unix (LF) UTF-8



```
Analise Lexica - Notepad
File Edit Format View Help
teste, Identificador
t3este, Identificador
t, Identificador Erro: identificador mal formado na Linh
@ Erro: caractere inesperado na Linha 20
1, Número Inteiro
123, Número Inteiro
12.57, Número Real
156, Número Inteiro
E, Identificador
-, símbolo subtração
78, Número Inteiro
test, Identificador Erro: identificador mal formado na L
@ Erro: caractere inesperado na Linha 22
e, Identificador
! Erro: caractere inesperado na Linha 22
;, símbolo ponto e vírgula
,, símbolo virgula
:, símbolo dois pontos
! Erro: caractere inesperado na Linha 22
teste, Identificador
t, Identificador
., símbolo ponto
este, Identificador Erro: identificador mal formado na L
@ Erro: caractere inesperado na Linha 22
1. Erro: número real mal formado na Linha 22
t, Identificador
1. Erro: número real mal formado na Linha 22
1, Número Inteiro
:, símbolo dois pontos
+, símbolo soma
2, Número Inteiro
-, símbolo subtração
5, Número Inteiro
teste, Identificador
Erro: fim de arquivo inesperado na Linha 24
Ln 1, Col 1 100% Unix (LF) UTF-8
```

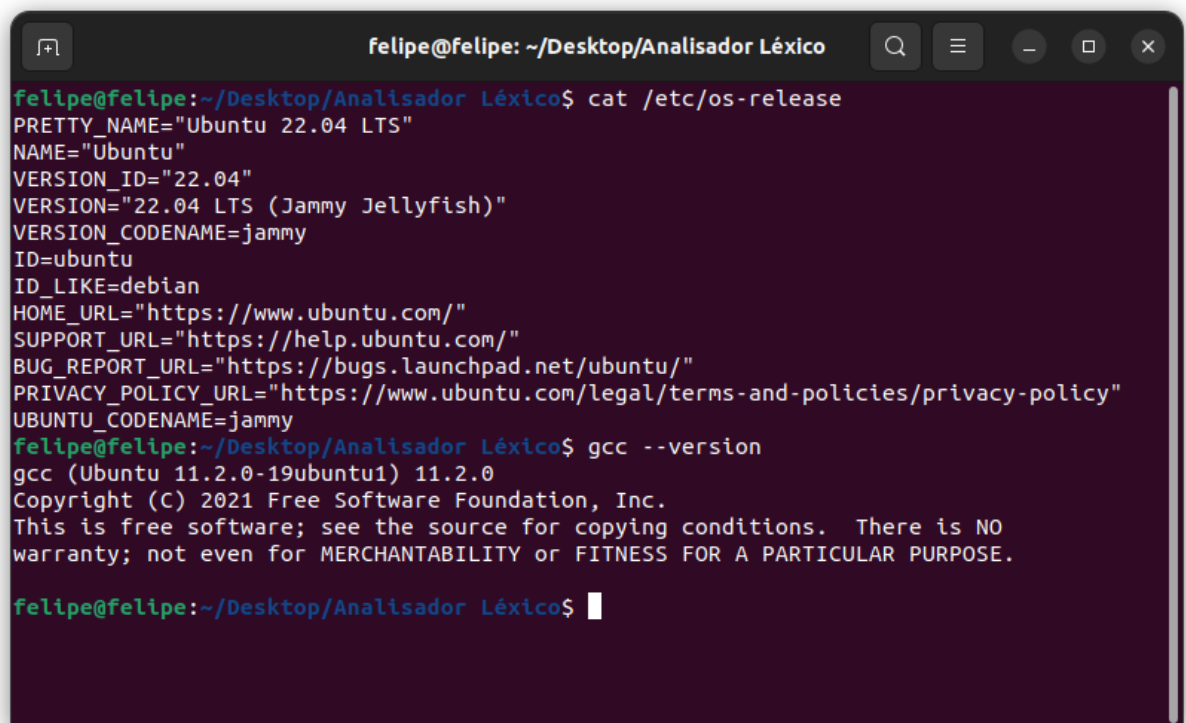
Figure 10. Análise Léxica do arquivo da Figura 08 parte 2

Instruções de Compilação

Todo o código foi desenvolvido na Linguagem C e compilado utilizando o gcc em um ambiente Linux Ubuntu, especificados na Figura 11. Nesse contexto, para compilar e rodar o programa basta executar os comandos:

```
gcc main.c -o main
./main
```

Ao executá-lo, este pedirá o caminho de um Arquivo de Entrada para executar a Análise Léxica. Ao terminar, uma mensagem de finalização é exibida e uma opção de impressão do Arquivo de Saída com os pares de cadeia token e respectivos erros léxicos é fornecida.

A terminal window titled 'felipe@felipe: ~/Desktop/Analizador Léxico' with standard Ubuntu window controls. The terminal shows the output of 'cat /etc/os-release' and 'gcc --version'.

```
felipe@felipe:~/Desktop/Analizador Léxico$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
felipe@felipe:~/Desktop/Analizador Léxico$ gcc --version
gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

felipe@felipe:~/Desktop/Analizador Léxico$
```

Figure 11. Versões do Sistema Operacional e Compilador utilizados

Referências

A. V. Aho, R. Sethi, J. D. Ullman

Compiladores: Princípios, técnicas e ferramentas

LTC - Livros Técnicos e Científicos Editora, 1995