Universidade Estadual Paulista
Relatório

Manual de uso - Analisador léxico

Laiza Lima Peria
Lara Cesquini Stopa

São José do Rio Preto
2024

Introdução

Este manual oferece uma orientação detalhada sobre o uso e configuração de um analisador léxico, uma ferramenta crucial na transformação de caracteres de entrada em tokens durante a compilação de linguagens de programação. O documento aborda o alfabeto da linguagem aceita, as expressões regulares que a definem e instruções de uso do analisador, sendo voltado para desenvolvedores, estudantes e profissionais que buscam compreender e aplicar a análise lexical de forma eficiente em seus projetos.

Alfabeto

O alfabeto utilizado inclui os seguintes caracteres:

- **Dígitos numéricos:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **Letras maiúsculas:** A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
- **Letras minúsculas:** a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
- **Caracteres acentuados:**
  - Acentos agudos: é, í, ó, ú, Á, É, Í, Ó, Ú
  - Acentos graves: à, è, ì, ò, ù, À, È, Ì, Ò, Ù
  - Acentos circunflexos: â, ê, î, ô, û, Â, Ê, Î, Ô, Û
  - Til: ã, õ, Ã, Õ

Além do alfabeto, são definidos os seguintes conjuntos de caracteres:

- **Espaços:** [' '\t\r](inclui espaço em branco, tabulação e retorno de carro)
- **Operadores matemáticos:** [-*/+%] (inclui os operadores de subtração, multiplicação, divisão, adição e módulo)
- **Operadores relacionais:** [><=] (inclui os operadores de maior que, menor que e igual a)
- **Separadores:** [[ ]( ){ },;#!&|'_:.](inclui colchetes, parênteses, chaves, vírgula, ponto e vírgula, cerquilha, símbolos de e comercial, barra vertical, apóstrofo, sublinhado, dois pontos e ponto final)

Expressões regulares

As expressões regulares ajudam a identificar e classificar diferentes tipos de tokens e caracteres em um texto, facilitando o processo de análise léxica.

- **{ESPACO}:** Evita que espaços em branco sejam processados ou contados como tokens.

- **{DIGITO}+{ID}:** Identifica erros ao declarar variáveis que começam com números, como ‘123abc‘.
- **{ID}{DIGITO}*{ID}*:** Reconhece identificadores válidos, ou seja, que comecem com letras.
- **{DIGITO}+:** Identifica e valida números inteiros.
- **{DIGITO}+”.”{DIGITO}+:** Identifica e valida números reais, que contêm um ponto decimal.
- **{OPERADORES}:** Reconhece operadores matemáticos.
- **{SEPARADORES}:** Reconhece separadores.
- **{RELACIONAIS}:** Reconhece operadores relacionais.
- **{DIGITO}+”,”{DIGITO}+:** Reconhece e marca números com vírgula como inválidos.
- **”:** Reconhece aspas duplas.
- **\:** Reconhece contra barra.
- **”//”[\n]*:** Identifica comentários de uma linha iniciados por ‘//‘, seguidos por qualquer texto até o final da linha.
- **é|É|à|À|á|Á:** Reconhece identificadores que contêm caracteres acentuados.
- **\n:** Incrementa a contagem de linhas quando um caractere de nova linha é encontrado.

Além disso, qualquer coisa que não seja uma regra válida é declarado como token inválido.

Instruções de uso

Instalar o flex

Primeiramente, é necessário instalar o Flex, que é uma ferramenta para geradores de analisadores léxicos. Para isso, é preciso inserir o seguinte comando em um terminal Linux:

```
sudo apt-get install flex
```

Compilar o arquivo .l (Analisador)

Depois de ter o Flex instalado, é preciso compilar o arquivo de definição do analisador léxico (.l). Supondo que o nome do arquivo seja lex.l, o comando no terminal será:

```
flex lex.l
```

Compilar o arquivo gerado com o GCC

Em seguida, é necessário compilar o arquivo gerado (lex.yy.c) usando o GCC para criar um executável. Insira o seguinte comando no terminal:

```
gcc lex.yy.c -o nome_do_executavel
```

O comando -o nome_do_executavel especifica o nome do executável que será criado, podendo substituí-lo por qualquer nome desejado.

Executar o analisador com um arquivo de texto para teste

Por fim, basta executar o analisador gerado com um arquivo de texto de teste. Supondo que o nome do arquivo de entrada seja teste.txt. O comando será:

```
./nome_do_executavel < teste.txt
```

Aqui, ./nome_do_executavel é o nome do executável gerado, e ¡ teste.txt redireciona o conteúdo do arquivo de teste para o analisador.