Analisador Léxico

Heitor de Lima Belém[160123950]

Universidade de Brasília cic@unb.com

1 Introdução

Este trabalho tem como objetivo a apresentação da análise léxica, que é a primeira fase do processo de compilação de um programa. Para a abordagem prática, com o intuito de fixar os conhecimentos obtidos do livro base da disciplina [ALSU07], foi proposto o desenvolvimento de um analisador léxico para uma nova linguagem, *C-IPL*, baseada nos princípios da linguagem *C*. A motivação que levou à criação da *C-IPL*, bem como os benefícios trazidos por ela e o processo de análise léxica desenvolvido no trabalho serão abordados ao longo do relatório.

2 Motivação

Listas são estruturas de dados que implementam uma coleção de valores, assim como os *arrays* presentes na linguagem C. Entretanto, as listas possuem operações especiais de acesso, adição e remoção de itens, que podem variar de linguagem para linguagem. A partir disso, foi proposta a criação de uma nova linguagem, baseada em C, que abordasse a implementação de listas, bem como de suas operações de acesso, adição e remoção, de maneira que a utilização dessas estruturas fosse facilitada por tais operações.

3 Descrição da análise léxica

Para implementar o analisador léxico da linguagem proposta, foi utilizado o *software FLEX* (*Fast Lexical Analyzer*), uma ferramenta geradora de programas que reconhecem padrões léxicos em texto [Est]. A estrutura de um arquivo reconhecido pelo *FLEX*, que possui a extensão .l, é dividida em três partes:

1. Definições
   Definições de funções, constantes, variáveis globais e inclusão de bibliotecas. Para este trabalho, foram criadas 3 variáveis globais: *errors_count*, *line_idx* e *column_idx*, que representam, respectivamente, a quantidade de erros obtidos durante a análise e o número da linha e coluna atual.
2. Regras
   Aqui são escritas as expressões regulares que vão procurar padrões no arquivo juntamente com as ações a serem tomadas ao encontrar tais padrões.
3. Código
   Nesta seção, encontra-se o código da função principal do arquivo com extensão *.l*, é aqui que será colocado o código gerado pelo *FLEX*.

4 Descrição dos arquivos de teste

Os arquivos utilizados para verificar o funcionamento do analisador léxico desenvolvido no trabalho estão no subdiretório */tests*. Nesse diretório, estão 4 arquivos com o prefixo *success_*, que representam os casos em que a análise léxica não identifica nenhum erro durante o processo. Já os 2 arquivos com o prefixo *wrong_* englobam os casos em que o analisador identifica erros léxicos na análise.

Os erros apresentados para cada arquivo estão identificados abaixo:

- *wrong_ex1.c*

```
ERROR: Invalid token '549a' at line: 2 column: 9
ERROR: Invalid token '1$bas' at line: 3 column: 11
ERROR: Unexpected character '#' at line: 4 column: 10
ERROR: Unexpected character '\' at line: 6 column: 13
ERROR: Unexpected character '~' at line: 9 column: 11
```

- *wrong_ex2.c*

```
ERROR: Unexpected character '~' at line: 2 column: 15
ERROR: Unexpected character '^' at line: 8 column: 10
ERROR: Unexpected character '[' at line: 28 column: 17
ERROR: Unexpected character ']' at line: 28 column: 19
```

5 Compilação e execução do programa.

Requisitos para compilação: software *FLEX* e o compilador *GCC*. Com isso devidamente instalado, é possível prosseguir para os seguintes passos.

- No diretório raiz do projeto, executar os comandos:

```
$ make
$ ./tradutor ./tests/<nome_do_arquivo>.c
```

Referências

[ALSU07] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques, Tools*. Pearson/Addison Wesley, 2nd edition, 2007.

[Est] W. Estes. Flex: Fast lexical analyser generator. https://github.com/westes/flex. Online; acessado 24 de Fevereiro de 2021.

A Gramática

⟨*program*⟩ ::= ⟨*function_def*⟩
| ⟨*function_def*⟩ ⟨*program*⟩
| ⟨*multi_statments*⟩

⟨*function_def*⟩ ::= ⟨*type*⟩ ⟨*identifier*⟩ '(' ⟨*function_parameters*⟩ ')' ⟨*multi_statments*⟩

⟨*function_parameters*⟩ ::= ⟨*type*⟩ ⟨*identifier*⟩ ',' ⟨*function_parameters*⟩
| ⟨*type*⟩ ⟨*identifier*⟩

⟨*multi_statments*⟩ ::= '{' ⟨*single_statment*⟩ '}'

⟨*single_statment*⟩ ::= ⟨*for*⟩
| ⟨*conditional*⟩
| ⟨*multi_statments*⟩
| ⟨*variable_declaration*⟩
| ⟨*assign_statment*⟩
| ⟨*filter_list_expression*⟩
| ⟨*map_list_expression*⟩
| ⟨*push_list_expression*⟩
| ⟨*single_statment*⟩

⟨*for*⟩ ::= 'for' '(' ⟨*assign_expression*⟩ ';' ⟨*logical_expression*⟩ ';'
⟨*identifier*⟩ ⟨*unary_operators*⟩ ')' ⟨*single_statment*⟩
| 'for' '(' ⟨*assign_expression*⟩ ';' ⟨*logical_expression*⟩ ';'
⟨*assign_expression*⟩ ')' ⟨*single_statment*⟩

⟨*conditional*⟩ ::= 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*single_statment*⟩ 'else'
⟨*single_statment*⟩
| 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*multi_statments*⟩ 'else'
⟨*single_statment*⟩
| 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*multi_statments*⟩ 'else'
⟨*multi_statments*⟩
| 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*single_statment*⟩ 'else'
⟨*multi_statments*⟩
| 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*single_statment*⟩
| 'if' '(' ⟨*logical_expression*⟩ ')' ⟨*multi_statments*⟩

⟨*filter_list_expression*⟩ ::= ⟨*identifier*⟩ '=' ⟨*identifier*⟩ '<<' ⟨*identifier*⟩ ';'

⟨*map_list_expression*⟩ ::= ⟨*identifier*⟩ '=' ⟨*identifier*⟩ '>>' ⟨*identifier*⟩ ';'

⟨*push_list_expression*⟩ ::= ⟨*identifier*⟩ '=' ⟨*identifier*⟩ ':' ⟨*identifier*⟩ ';'

⟨*assign_expression*⟩ ::= ⟨*identifier*⟩ ⟨*assign_operators*⟩ ⟨*values*⟩ ';'
| ⟨*logical_expression*⟩ '||' ⟨*comparison_expression*⟩
| ⟨*comparison_expression*⟩

⟨*logical_expression*⟩ ::= ⟨*logical_expression*⟩ '&&' ⟨*comparison_expression*⟩
| ⟨*logical_expression*⟩ '||' ⟨*comparison_expression*⟩
| ⟨*comparison_expression*⟩

⟨*comparison_expression*⟩ ::= ⟨*comparison_expression*⟩ ⟨*comparison_operators*⟩
⟨*aritmetic_expression*⟩
| ⟨*aritmetic_expression*⟩

⟨*aritmetic_expression*⟩ ::= ⟨*aritmetic_expression*⟩ ⟨*aritmetic_operators*⟩ ⟨*aritmetic_expression*⟩
| ⟨*values*⟩
⟨*return*⟩ ::= '`return`' ⟨*values*⟩ '`;`'
⟨*io_expression*⟩ ::= '`read`''`(`'⟨*identifier*⟩'`)`''`;`'
| '`write`''`(`'⟨*string*⟩ | ⟨*logical_expression*⟩'`)`''`;`'
| '`writeln`''`(`'⟨*string*⟩ | ⟨*logical_expression*⟩'`)`''`;`'
⟨*values*⟩ ::= ⟨*constant*⟩ | ⟨*identifier*⟩
⟨*variable_declaration*⟩ ::= ⟨*type*⟩ ⟨*identifier*⟩ '`;`'
⟨*constant*⟩ ::= ⟨*integer*⟩ | ⟨*float*⟩ | ⟨*nil*⟩
⟨*type*⟩ ::= ⟨*simple_type*⟩ | ⟨*list_type*⟩
⟨*assign_operators*⟩ ::= '`=`' | '`+=`' | '`-=`' | '`*=`' | '`/=`'
⟨*unary_operators*⟩ ::= '`++`' | '`--`'
⟨*aritmetic_operators*⟩ ::= '`+`' | '`-`' | '`*`' | '`/`'
⟨*comparison_operators*⟩ ::= '`>`' | '`<`' | '`>=`' | '`<=`' | '`==`' | '`!=`'
⟨*identifier*⟩ ::= (⟨*character*⟩|'`_`')(⟨*character*⟩|⟨*digit*⟩|'`_`')*
⟨*nil*⟩ ::= NIL
⟨*string*⟩ ::= ¨'.* ¨

⟨*float*⟩ ::= ⟨*digit*⟩*'`.`'⟨*digit*⟩+
⟨*integer*⟩ ::= ⟨*digit*⟩+
⟨*list_type*⟩ ::= ⟨*simple_type*⟩ list
⟨*simple_type*⟩ ::= int | float
⟨*character*⟩ ::= [a-zA-Z]
⟨*digit*⟩ ::= [0-9]