Análise Léxica da Linguagem C-IPL

Lucas Dalle Rocha
17/0016641

Universidade de Brasília - UnB
170016641@aluno.unb.br

1 Motivação

A disciplina propõe a implementação das etapas de análise léxica, sintática, semântica e de geração de código intermediário para um subconjunto da linguagem C, denominado *C-IPL*, com o intuito de ressaltar a importância o processo de tradução de uma linguagem em alto nível para linguagem de máquina, bem como o impacto do desenvolvimento do projeto no sistema como um todo. Assim, pela linguagem *C-IPL* busca-se facilitar a manipulação de listas para programas escritos em C, visto que são utilizadas frequentemente para armazenamento de dados em sequência.

2 Descrição

A etapa de análise léxica visa atribuir lexemas válidos para dada linguagem, além de estabelecer a tabela de símbolos para satisfazer a comunicação com o analisador sintático.

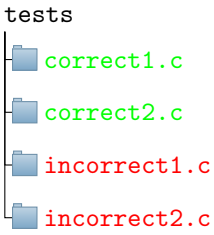
Dito isso, houve o uso de variáveis com a finalidade de exibir a linha e coluna em que o lexema analisado se encontra, bem como expressões regulares para buscar padrões que condizem com a descrição da linguagem *C-IPL*, de modo a atribuir *tokens* válidos para identificadores, dígitos, tipos de dado, palavras reservadas, números inteiros e reais, operadores aritméticos, relacionais, lógicos, e as operações sobre listas, implementadas pela linguagem *C-IPL*. Ademais, comentários não são tratados como lexemas, *strings* são sinalizadas como um único *token* e os símbolos de pontuação pareada também serão lexemas.

Dessa forma, quaisquer *lexemas* que não foram representados pelas expressões regulares são considerados erros léxico, assim como comentários multilinhas e *strings*, ambos não finalizados. Por fim, delimitadores como ponto e vírgula serão aprofundados na etapa de análise sintática, em conjunto com a tabela de símbolos, que será composta pelos atributos de identificadores (nome e tipo) e constantes numéricas (valores inteiros ou reais) armazenados em um tipo de lista.

3 Testes

Existem dois arquivos de teste que se encontram corretos (representados pela cor verde abaixo), de acordo com a análise léxica, uma vez que não apresentam

lexemas irreconhecíveis pela linguagem. Em contrapartida, existem outros dois arquivos de teste que possuem erros (representados pela cor vermelha abaixo), que serão listados abaixo. Todos os testes encontram-se na pasta *tests*.



1. incorrect1.c apresenta erros de lexemas não reconhecidos pela linguagem, como [$ \ .], nas linhas 2, 3 e 4, respectivamente. Ademais, o erro léxico de comentário multilinha não finalizado encerra o programa previamente.
2. incorrect2.c similarmente possui lexemas não reconhecidos pela *C-IPL*, tais como [@ ~#], nas linhas 1, 5 e 8, respectivamente. Por inicializar uma *string*, na linha 9, e não finalizá-la, ocorre o encerramento do programa igualmente.

4 Instruções para compilação e execução

Para a compilação e execução, é requerido a utilização do sistema operacional Ubuntu 20.04.2 LTS, além do *gcc* versão 11.1.0, *flex* versão 2.6.4 e GNU Make 4.2.1. O programa pode ser compilado pela utilização do *Makefile*, com o seguinte comando em seu terminal, ou manualmente, apresentado em seguida:

```
$ make

$ flex src/lexical.l && gcc-11 -o tradutor src/main.c
```

Por fim, para execução do analisador léxico nos arquivos teste:

```
$ ./tradutor tests/<file>.c
```

Referências

[ALSU06] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 2nd edition, 2006.

[Hec21] Robert Heckendorn. A grammar for the c- programming language. *http://marvin.cs.uidaho.edu/Teaching/CS445/c-Grammar.pdf*, 2021. [Online; Last accessed 19-August-2021].

[LMB92] John R. Levine, Tony Mason, and Doug Brown. *lex & yacc*. O'Reilly & Associates, Inc., 2nd edition, 1992.

[PEM16] Vern Paxson, Will Estes, and John Millaway. Lexical analysis with flex, for flex 2.6.2. *https://westes.github.io/flex/manual/*, 2016. [Online; Last accessed 19-August-2021].

A Estruturação do *token*

Para estruturação do *token*, isto é, seus padrões, nomes e atributos, foi utilizado como referência a figura 3.12 do livro-texto [ALSU06], abordado na página 130.

Lexema	**Nome do *token***	**Atributo**
Quaisquer espaços em branco	–	–
Qualquer *id*	**id**	Ponteiro p/ tabela de símbolos
Qualquer *integer_number*	**integer**	Ponteiro p/ tabela de símbolos
Qualquer *real_number*	**real**	Ponteiro p/ tabela de símbolos
if	**if**	–
else	**else**	–
for	**for**	–
return	**return**	–
read	**input**	–
write, writeln	**output**	–
NIL	**NIL**	–
int, float	**simple**	–
int list, float list	**compound**	–
<, <=, >, >=, ==, ! =	**relop**	LT, LE, GT, GE, EQ, NE
+, -, *, /	**ariop**	ADD, SUB, MUL, DIV
||, &&	**logop**	OR, AND
=	**assign**	ASSIGN
:, ?, %, >>, <<	**listop**	CTOR, HOP, TDTOR, MAP, FLTR
!	**ambiguous**	NOT or TOP

Tabela 1. Padrões de *tokens*, seus nomes e atributos.

B Gramática da linguagem

Para a montagem da gramática, foi utilizado como referência [Hec21], a fim de estimar a transição da análise léxica para a sintática. As adaptações foram feitas para que a gramática seja compatível com a linguagem *C-IPL*.

1. *initial* → *declaration_list*
2. *declaration_list* → *declaration_list decl* | *decl*
3. *decl* → *var_declaration* | *func_declaration*
4. *var_declaration* → *data_type* **ID**;
5. *data_type* → **INT** | **FLOAT** | **INT_LIST** | **FLOAT_LIST**
6. *func_declaration* → *data_type* **ID** (*params*) *stmts*
7. *params* → *param_list* | **NIL**
8. *param_list* → *param_list*, *param* | *param*
9. *param* → *var_declaration*
10. *stmts* → {*stmt_list*}
11. *stmt_list* → *stmt stmt_list* | *stmt*
12. *stmt* → *var_declaration* | *return_stmt* | *select_stmt* | *iter_stmt* | *io_stmt* | *exp_stmt* | *list_related_stmt*
13. *list_related_stmt* → *constr_exp* | *opr_exp* | *destr_exp* | *func_exp*
14. *constr_exp* → *assign_exp* **LIST_CONSTRUCTOR ID**;
15. *opr_exp* → **LIST_OPERATOR ID**;
16. *destr_exp* → **LIST_DESTRUCTOR ID**;
17. *func_exp* → *assign_exp* **LIST_FUNCTIONS ID**;
18. *exp_stmt* → *assign_exp* | *simple_exp*
19. *assign_exp* → **ID** = *exp_stmt*
20. *select_stmt* → **IF** (*simple_exp*) *stmt* | **IF** (*simple_exp*) *stmt* **ELSE** *stmt*
21. *iter_stmt* **FOR** (*assign_exp*; *simple_exp*; *assign_exp*;) *stmt*
22. *return_stmt* → **RETURN** *exp_stmt*;
23. *io_stmt* → *in_stmt* | *out_stmt*
24. *in_stmt* → **READ** (**ID**);
25. *out_stmt* → **WRITE** (**STRING_LITERAL**); | **WRITELN** (**STRING_LITERAL**);
26. *simple_exp* → *simple_exp* **LOGICAL_OR** *and_exp* | *and_exp*
27. *and_exp* → *and_exp* **LOGICAL_AND** *not_exp* | *not_exp*
28. *not_exp* → **LOGICAL_NOT** *not_exp* | rel_exp
29. *rel_exp* → *sum_exp relop sum_exp* | *sum_exp*
30. *relop* → <= | < | > | >= | == |! =
31. *sum_exp* → *sum_exp sumop mul_exp* | *mul_exp*
32. *sumop* → + | -
33. *mul_exp* → *mul_exp mulop factor* | *factor*
34. *mulop* → * | /
35. *factor* → **ID** | (*exp_stmt*) | *func_call* | *const*
36. *func_call* → **ID** (*params*)
37. *const* → **CONST_INT** | **CONST_FLOAT** | **NIL**

C Léxico da linguagem

Para a esquematização do léxico em expressões regulares, foi utilizado como referência o livro a respeito do analisador léxico [LMB92], bem como o manual do flex [PEM16], disponibilizado *online*.

Macro do token	**Definição *regex***
comment_line	"//".*
delim	[\t\r]
ws	[delim]+
newline	[\n]
brace_opening & brace_closing	"{" "}"
bracket_opening & bracket_closing	"[" "]"
parenthese_opening & parenthese_closing	"(" ")"
letter	[A-Za-z_]
digit	[0-9]
keywords	"if"|"else"|"for"|"return"
data_type	"int list"|"int"|"float list"|"float"
input_command	"read"
output_command	"write"|"writeln"
nil_constant	"NIL"
id	{letter}({letter}|{digit})*
integer_number	-?{digit}+
real_number	-?{integer_number}(\.{digit}+)(E[+-]?{digit}+)?
arithmetic_operators	"+"|"-"|"*"|"/"
relational_operators	"<"|"<="|">"|">="|"=="|"! ="
logical_operators	"||"|"&&"
assignment	"="
separator	","
flow_control	";"
list_constructor	":"
list_operator	"?"
list_destructor	"%"
list_functions	">>"|"<<"
string_literal	\"([^"\\\n]|\\.)*\"
ambiguous_operator	!

Tabela 2. Tabela de lexemas e suas regras.