Implementação de Analisador Léxico para a Linguagem C - Trabalho de Compiladores

Pedro Henrique Serpa, Vitor Bernstorff Clemes

Departamento de Ciência da Computação
Universidade Estadual de Santa Catarina (UDESC) – Joinville, SC – Brazil

pedroserpah@gmail.com, vitorbc.oficial@gmail.co

1. Contextualização

O trabalho compreende por ser um analisador léxico, categorizando tokens do código exemplo de acordo com a Forma Normal de Backus-Naur.

2. Trabalho Desenvolvido

Foi criado um programa em LEX para fazer a leitura de um código em C que cataloga e imprime uma lista com todos os tokens encontrados naquele exemplo.

OBS: O analisador léxico criado não é um analisador da linguagem C. Ele meramente cataloga os símbolos encontrados no arquivo exemplo (codigo-exemplo.c).

3. Implementação

Na implementação em LEX, cada símbolo é estruturado da seguinte maneira:

```
typedef struct symbol{
       int id;
       int row;
       int column;
       char content[100];
       char *type;

   } symbol;
```

Cada símbolo é composto por um id (identificador único do símbolo), row (linha em que o símbolo foi encontrado), column (coluna em que o símbolo foi encontrado), content

(conteúdo armazenado) e type (um identificador daquele token que será usado em todos os outros símbolos os quais possuírem o mesmo token).

Para a tabela, notou-se necessário guardar todos os tokens que o arquivo possa vir a conter. Portanto, foi alocado uma lista de symbol para armazenar esses.

```
   symbol **symbolsTable;
```

Esses comandos são utilizados na compilação do trabalho

```
flex --noyywrap lex.lex
gcc lex.yy.c -lfl -o lex

./lex código-exemplo.c > out
```

A compilação e execução do programa ocorrem com uma única linha de comando:

```
./project.sh
```

A saída encontrada é a seguinte (disponível no arquivo out):

TABELA DE SÍMBOLOS

id	type	row	column	value
0	DIRECTIVE	1	1	#include
1	LIBRARY	1	10	<stdio.h>
2	DIRECTIVE	2	0	#include
3	LIBRARY	2	9	<stdlib.h>
4	DIRECTIVE	3	0	#include
5	LIBRARY	3	9	<locale.h>
6	DEFINE	4	0	#define
7	IDENTIFIER	4	8	TAM
8	INTEGER	4	12	10
9	TYPE	6	0	int
10	IDENTIFIER	6	4	main
11	PARENTHESES	6	8	(
12	PARENTHESES	6	9	)
13	QUOTE	6	11	{
14	IDENTIFIER	7	4	setlocale
15	PARENTHESES	7	13	(
16	IDENTIFIER	7	14	LC_ALL
17	COMMA	7	20	,
18	STRING	7	22	
19	PARENTHESES	7	24	)
20	SEMICOLON	7	25	;
21	TYPE	8	4	int
22	IDENTIFIER	8	8	numeros
23	BRACKET	8	15	[
24	IDENTIFIER	8	16	TAM
25	BRACKET	8	19	]
26	SEMICOLON	8	20	;
27	TYPE	9	4	int

28	IDENTIFIER	9	8	i
29	COMMA	9	9	,
30	IDENTIFIER	9	11	aux
31	COMMA	9	14	,
32	IDENTIFIER	9	16	contador
33	SEMICOLON	9	24	;
34	IDENTIFIER	11	4	printf
35	PARENTHESES	11	10	(
36	STRING	11	11	Entre com dez números para preencher o array, e pressione enter após digitar cada um:\n
37	PARENTHESES	11	102	)
38	SEMICOLON	11	103	;
39	IDENTIFIER	12	4	for
40	PARENTHESES	12	8	(
41	IDENTIFIER	12	9	i
42	OPERATOR	12	11	=
43	INTEGER	12	13	0
44	SEMICOLON	12	14	;
45	IDENTIFIER	12	16	i
46	OPERATOR	12	18	<
47	IDENTIFIER	12	20	TAM
48	SEMICOLON	12	23	;
49	IDENTIFIER	12	25	i
50	OPERATOR	12	26	=+
51	OPERATOR	12	27	=+
52	PARENTHESES	12	28	)
53	QUOTE	12	30	{
54	IDENTIFIER	13	8	scanf
55	PARENTHESES	13	13	(
56	STRING	13	14	%d
57	COMMA	13	18	,

58	SEMICOLON	13	20	&
59	IDENTIFIER	13	21	numeros
60	BRACKET	13	28	[
61	IDENTIFIER	13	29	i
62	BRACKET	13	30	]
63	PARENTHESES	13	31	)
64	SEMICOLON	13	32	;
65	QUOTE	14	4	}
66	IDENTIFIER	15	4	printf
67	PARENTHESES	15	10	(
68	STRING	15	11	Ordem atual dos itens no array:\n
69	PARENTHESES	15	46	)
70	SEMICOLON	15	47	;
71	IDENTIFIER	16	4	for
72	PARENTHESES	16	8	(
73	IDENTIFIER	16	9	i
74	OPERATOR	16	11	=+
75	INTEGER	16	13	0
76	SEMICOLON	16	14	;
77	IDENTIFIER	16	16	i
78	OPERATOR	16	18	<
79	IDENTIFIER	16	20	TAM
80	SEMICOLON	16	23	;
81	IDENTIFIER	16	25	i
82	OPERATOR	16	26	=+
83	OPERATOR	16	27	=+
84	PARENTHESES	16	28	)
85	QUOTE	16	30	{
86	IDENTIFIER	17	4	printf
87	PARENTHESES	17	10	(

88	STRING	17	11	%4d
89	COMMA	17	16	,
90	IDENTIFIER	17	18	numeros
91	BRACKET	17	25	[
92	IDENTIFIER	17	26	i
93	BRACKET	17	27	]
94	PARENTHESES	17	28	)
95	SEMICOLON	17	29	;
96	QUOTE	18	4	}
97	IDENTIFIER	20	4	for
98	PARENTHESES	20	8	(
99	IDENTIFIER	20	9	contador
100	OPERATOR	20	18	=+
101	INTEGER	20	20	1
102	SEMICOLON	20	21	;
103	IDENTIFIER	20	23	contador
104	OPERATOR	20	32	<
105	IDENTIFIER	20	34	TAM
106	SEMICOLON	20	37	;
107	IDENTIFIER	20	39	contador
108	OPERATOR	20	47	=+
109	OPERATOR	20	48	=+
110	PARENTHESES	20	49	)
111	QUOTE	20	51	{
112	IDENTIFIER	21	8	for
113	PARENTHESES	21	12	(
114	IDENTIFIER	21	13	i
115	OPERATOR	21	15	=
116	INTEGER	21	17	0
117	SEMICOLON	21	18	;

118	IDENTIFIER	21	20	i
119	OPERATOR	21	22	<
120	IDENTIFIER	21	24	TAM
121	OPERATOR	21	28	-
122	INTEGER	21	30	1
123	SEMICOLON	21	31	;
124	IDENTIFIER	21	33	i
125	OPERATOR	21	34	=+
126	OPERATOR	21	35	=+
127	PARENTHESES	21	36	)
128	QUOTE	21	38	{
129	IDENTIFIER	22	12	if
130	PARENTHESES	22	15	(
131	IDENTIFIER	22	16	numeros
132	BRACKET	22	23	[
133	IDENTIFIER	22	24	i
134	BRACKET	22	25	]
135	OPERATOR	22	27	>
136	IDENTIFIER	22	29	numeros
137	BRACKET	22	36	[
138	IDENTIFIER	22	37	i
139	OPERATOR	22	39	=+
140	INTEGER	22	41	1
141	BRACKET	22	42	]
142	PARENTHESES	22	43	)
143	QUOTE	22	45	{
144	IDENTIFIER	23	16	aux
145	OPERATOR	23	20	=
146	IDENTIFIER	23	22	numeros
147	BRACKET	23	29	[

148	IDENTIFIER	23	30	i
149	BRACKET	23	31	]
150	SEMICOLON	23	32	;
151	IDENTIFIER	24	16	numeros
152	BRACKET	24	23	[
153	IDENTIFIER	24	24	i
154	BRACKET	24	25	]
155	OPERATOR	24	27	=
156	IDENTIFIER	24	29	numeros
157	BRACKET	24	36	[
158	IDENTIFIER	24	37	i
159	OPERATOR	24	39	=+
160	INTEGER	24	41	1
161	BRACKET	24	42	]
162	SEMICOLON	24	43	;
163	IDENTIFIER	25	16	numeros
164	BRACKET	25	23	[
165	IDENTIFIER	25	24	i
166	OPERATOR	25	26	=+
167	INTEGER	25	28	1
168	BRACKET	25	29	]
169	OPERATOR	25	31	=
170	IDENTIFIER	25	33	aux
171	SEMICOLON	25	36	;
172	QUOTE	26	12	}
173	QUOTE	27	8	}
174	QUOTE	28	4	}
175	IDENTIFIER	30	4	printf
176	PARENTHESES	30	10	(
177	STRING	30	11	\nElementos do array em ordem crescente:\n

178	PARENTHESES	30	55	)
179	SEMICOLON	30	56	;
180	IDENTIFIER	31	4	for
181	PARENTHESES	31	8	(
182	IDENTIFIER	31	9	i
183	OPERATOR	31	11	=
184	INTEGER	31	13	0
185	SEMICOLON	31	14	;
186	IDENTIFIER	31	16	i
187	OPERATOR	31	18	<
188	IDENTIFIER	31	20	TAM
189	SEMICOLON	31	23	;
190	IDENTIFIER	31	25	i
191	OPERATOR	31	26	=+
192	OPERATOR	31	27	=+
193	PARENTHESES	31	28	)
194	QUOTE	31	30	{
195	IDENTIFIER	32	8	printf
196	PARENTHESES	32	14	(
197	STRING	32	15	%4d
198	COMMA	32	20	,
199	IDENTIFIER	32	22	numeros
200	BRACKET	32	29	[
201	IDENTIFIER	32	30	i
202	BRACKET	32	31	]
203	PARENTHESES	32	32	)
204	SEMICOLON	32	33	;
205	QUOTE	33	4	}
206	IDENTIFIER	34	4	printf
207	PARENTHESES	34	10	(

208	STRING	34	11	\n
209	PARENTHESES	34	15	)
210	SEMICOLON	34	16	;
211	IDENTIFIER	35	4	return
212	INTEGER	35	11	0
213	SEMICOLON	35	12	;
214	QUOTE	36	0	}