SCC0605 Teoria da Computação e Compiladores

Prof. Dr. Thiago A.S Pardo

Trabalho 1: Analisador Léxico

Nome:	*Número USP:*
Carla Nunes	8479343
Leandro A. Silva	9805341
Marilene A. Garcia	10276974

1 Decisões do Projeto

O grupo optou por desenvolver o projeto na linguagem de programação C++ devido a alguns fatores. Entre eles pode-se citar a familiaridade dos membros da equipe com esta linguagem, que possui um bom desempenho, o que pode otimizar o tempo da análise léxica considerando que o algoritmo é complexo, visto que possui vários laços de repetição e estruturas condicionais. Além do fato da linguagem ser orientada a objetos o que tornou implementação mais eficiente e consumindo menos memória.

Para implementação dos autômatos foi usada uma solução ligeiramente diferente das que foram apresentadas no material. Inicialmente, a função principal no arquivo "main.cpp" recebe o arquivo texto com o código na linguagem P- -, realiza um pequeno tratamento para separar os tokens e então os envia, um por um, para uma função que realiza a análise de tokens. Nessa função é identificado se ele é um número, identificador, símbolo ou palavra reservada. Ou, caso ele não seja válido, é retornado o erro correspondente, que pode ser de número ou identificador irregular, de caractere não permitido ou um erro léxico inesperado.

Vale ressaltar, que comentários são descartados pelo analisador léxico e um tratamento de erro sintático já foi implementado que é o caso de comentários abertos mas não fechados. Além de ser importante notar que não é feito um pré-processamento, ou seja, o arquivo é lido conforme a analise léxica avança e caso seja necessário interromper o processo, não haverá desperdício de tempo ou recursos.

Por fim, pode-se notar que foi decidido a criação de bibliotecas para melhor organizar o código e também para facilitar a próxima etapa do projeto - analisador sintático. Com isso, tem-se três arquivos distintos:
- O arquivo **"main.cpp"** contém a função principal que coordena todo o procedimento, abrindo e fechando os arquivos de entrada e saída e também chamando as funções do analisador léxico da biblioteca.
- O arquivo **"lexico.cpp"** contém a definição das classes e as funções referentes aos autômatos da biblioteca léxica.
- O arquivo **"lexico.h"** define as funções da biblioteca.

Todos os códigos estão comentados, facilitando o entendimento.

2 Estrutura de Análise

As tabelas 1 e 2 apresentam a forma como os tokens foram mapeados.

Tabela 1: Tabela do Mapeamento

Token	Cadeia
id	a-z A-Z
num_int	0-9
num_real	0-9.0-9
simb_program	program
simb_real	real
simb_end	end
simb_begin	begin
simb_var	var
simb_else	else
simb_read	read
simb_write	write
simb_if	if
simb_then	then
simb_while	while
simb_do	do
simb_for	for
simb_to	to
simb_procedure	procedure
simb_integer	integer
simb_soma	+
simb_sub	-
simb_mult	*
simb_div	/
simb_maior	>
simb_menor	<
simb_apar	(
simb_fpar	)
simb_dp	:
simb_pv	;
simb_virg	,
simb_igual	=
simb_atri	:=
simb_maig	>=
simb_dif	<>
simb_meig	<=

Tabela 2: Tabela de Tokens

Token	Expressão Regular
id	a-z A-Z
num int	0-9
num real	0-9.0-9
palavras reservadas	program end begin var else read write if then while do for to procedure integer real
pontuação	+ - / * > < <> <= >= := ( ) : ; =
caracteres não imprimíveis	Espaço em branco, tabulação, nova linha

3 Autômato

A Figura 1 é um esboço do autômato que faz o reconhecimento dos tokens e ignora o conteúdo dos comentários, ou seja, as cadeias dentro de { }.

As ações para cada estado final são:

Estado q2: se busca tabela(palavras reservadas) = verdadeiro
então retornar(nome da palavra reservada)
senão retorna(id)

Estado q4: retornar(numero inteiro)

Estado q8: retornar(numero real)

Estado q9: retornar("erro lexico: identificador irregular")

Estado q11: retornar("erro lexico: número irregular")

Estado q15: retornar("erro lexico: caractere não permitido")

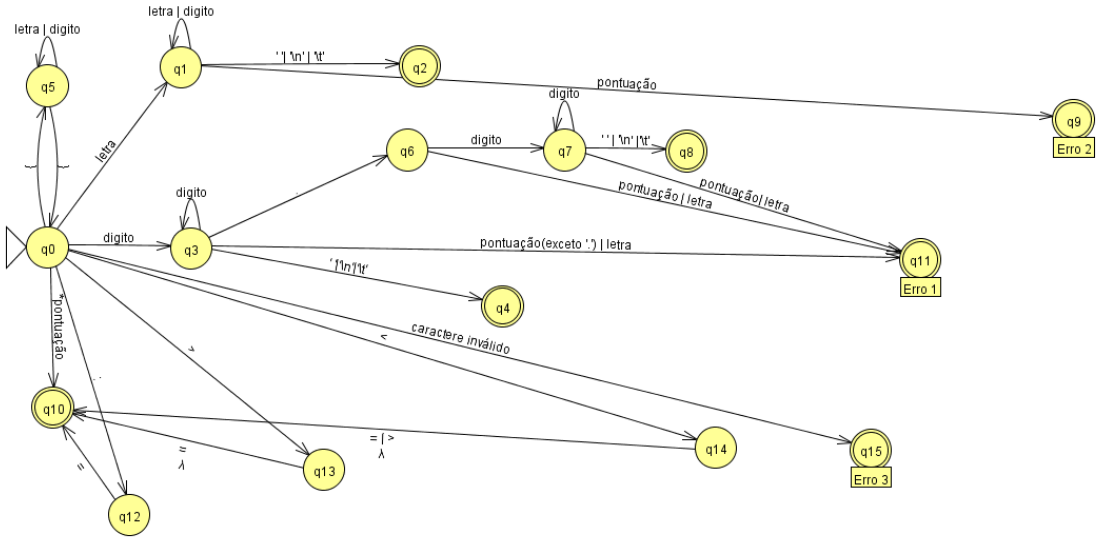

Figura 1: Autômato Finito do Projeto

4 Tratamento de Erros

Três tipos de erro podem ser identificados no analisador léxico:
- Erro 1: número irregular, com letra ou pontuação inválida
- Erro 2: identificador irregular, com pontuação
- Erro 3: caractere inválido inserido

Além desses erros, também é tratado o erro de abertura do arquivo de entrada ("erro na abertura do arquivo") e um erro sintático de não fechamento de parênteses de comentários ("comentário não finalizado").

5 Exemplo de Execução

5.1 Comandos no terminal

Para compilar e executar nosso programa, deve-se estar na pasta onde estão os códigos e executar os seguintes comandos (no Linux):

```
g++ lexico.cpp main.cpp -o exe
./exe (numero do arquivo teste)_input.txt
```

No arquivo zip enviado, existem 6 arquivos teste que tem exemplos de programa em P−−, com e sem erros.

5.2 Programa de entrada (com erros e irregularidades)

```
program program;
var x@a:integer;
{comentario aleatorio 5 @ _ asa}
begin
        1x:=1.578;
        while(x*<=3) do
                if$ (x<>2) then
                        x::=x+12.5%1;
end.
```

5.3 Arquivo de saída

```
program, simb_program
program, simb_program
;, simb_pv
var, simb_var
x@a, erro lexico: identificador irregular
:, simb_dp
integer, simb_integer
;, simb_pv
begin, simb_begin
1x, erro lexico: numero irregular
:=, simb_atri
1.578, num_real
;, simb_pv
while, simb_while
(, simb_apar
x, id
*, simb_mult
<=, simb_meig
3, num_int
), simb_fpar
do, simb_do
if$, erro lexico: identificador irregular
(, simb_apar
x, id
<>, simb_dif
2, num_int
), simb_fpar
then, simb_then
```

```
x, id
:, simb_dp
:, simb_dp
=, simb_igual
x, id
+, simb_soma
12.5%1, erro lexico: numero irregular
;, simb_pv
end, simb_end
., simb_pont
```

6 Conclusão

O projeto do analisador léxico funcionou como o esperado. As cadeias estão sendo analisadas conforme descrito na gramática da linguagem P−−, além de serem mapeadas pelos autômatos. Os possíveis erros léxicos são identificados e tratados. Logo, o código do projeto está eficiente e modularizado de forma que seja possível avançar nas próximas etapas, para a construção do compilador.

Referências

1º) Repositório do trabalho: Disponível em:
`https://github.com/CarlaNunes/Analisador-L-xico`
Último acesso em 23 de Abril de 2020

2º) Notas de aula SCC0605- Teoria de Computação e Compiladores: Disponível em:
`https://ae4.tidia-ae.usp.br/portal/site/0dd38d17-9cb7-413c-a860-2c158b8ebaad/page/d3ff2027-a933-4750-b90e-6ba56e2f591f`

Último acesso em 24 de Abril de 2020

3º) JFLAP: Programa usado para a confecção do Autômato Disponível em: `http://www.jflap.org/`
Último acesso em 24 de Abril de 2020