SCC0605 Teoria da Computação e Compiladores

SAMIRA CRISTINA VASCONCELOS MENDES
MATHEUS ARATAQUE UEMA
NELSON CALSOLARI NETO

Desenvolvimento de um analisador léxico para linguagem P--

SÃO CARLOS
2020

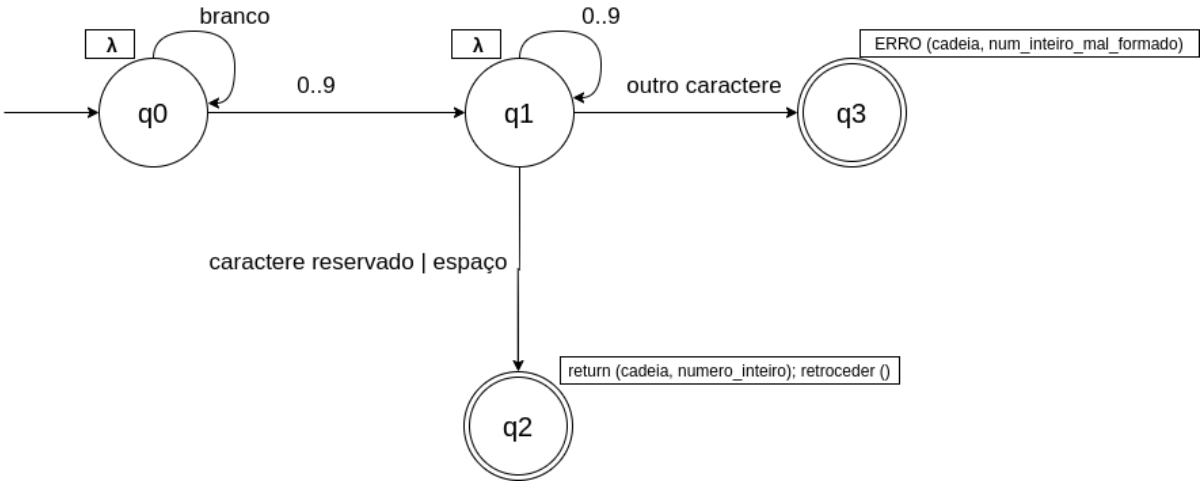
1. AUTÔMATOS PROJETADOS

Seguindo as especificações da linguagem P--, inicialmente foram projetados, de forma individual, os autômatos para identificação de números inteiros, números reais, comentários, identificadores e palavras reservadas, sendo os dois últimos implementados num mesmo autômato.

O termo "caractere reservado" usado na transição de alguns autômatos a seguir, refere-se a qualquer caractere que faça parte da tabela lado. Essa decisão foi tomada com base em símbolos reservados da linguagem P-- que serão utilizados na identificação muitas vezes do término de certas cadeias.

=
<
>
+
-
*
/
:
;
,
.
(
)

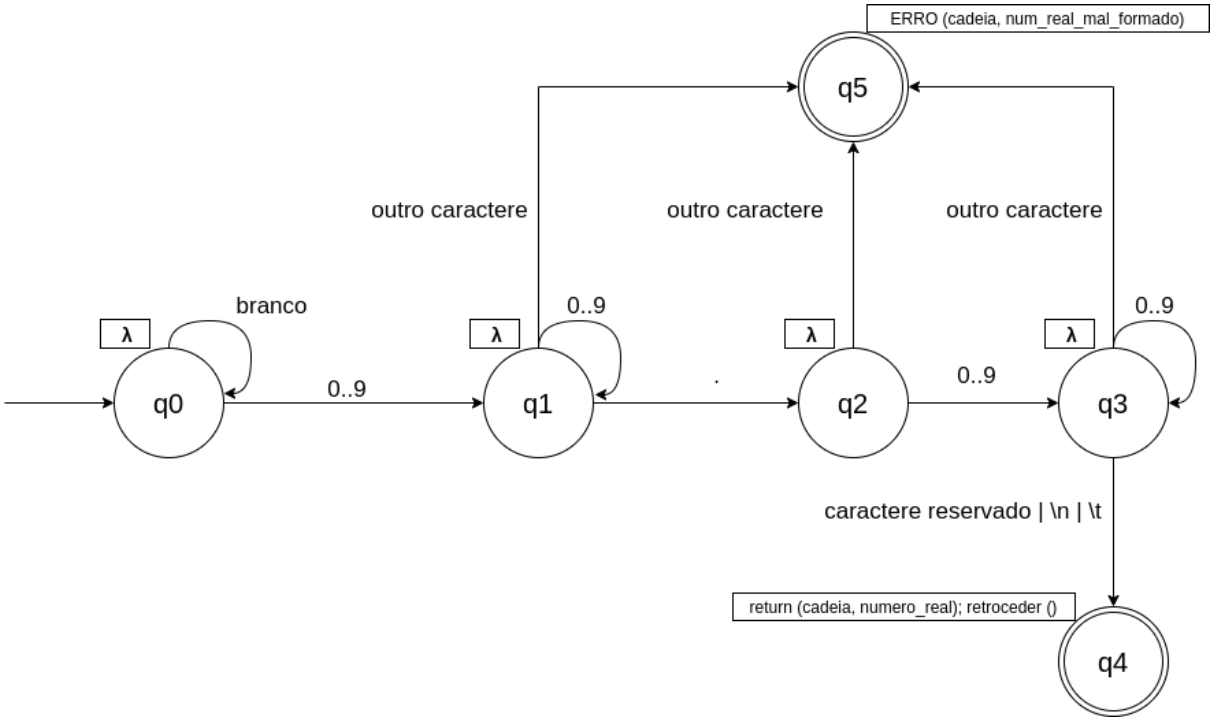
1.1 NÚMEROS INTEIROS



Um "outro caractere" aqui é definido aqui como qualquer caractere diferente de um número, caractere reservado ou espaço, como por exemplo, uma letra. Essa decisão foi

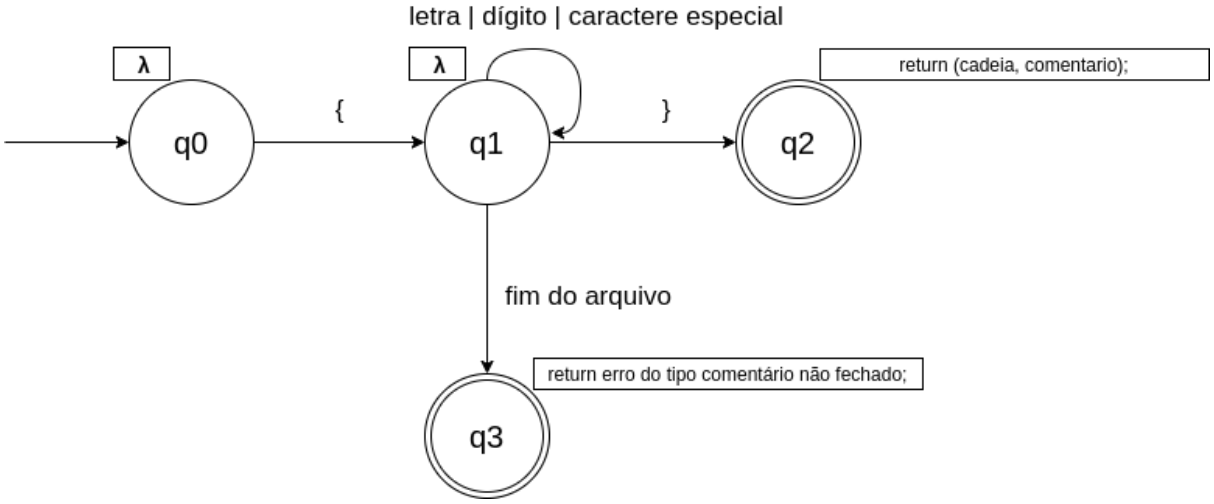
tomada de forma que números escrito como “2a3” ou “23a” sejam interpretados como números inteiros mal formados pelo analisador léxico.

1.2 NÚMEROS REAIS



“Outro caractere” é definido da mesma forma como foi definido em números inteiros; erros em relação a formação dos números são reportados de forma análoga.

1.3 COMENTÁRIOS



Qualquer cadeia escrita entre chaves é ignorada por se tratar de um comentário e por consequência ser irrelevante para a compilação. Na implementação é assegurado também que os comentários se restrinjam a uma linha de código.

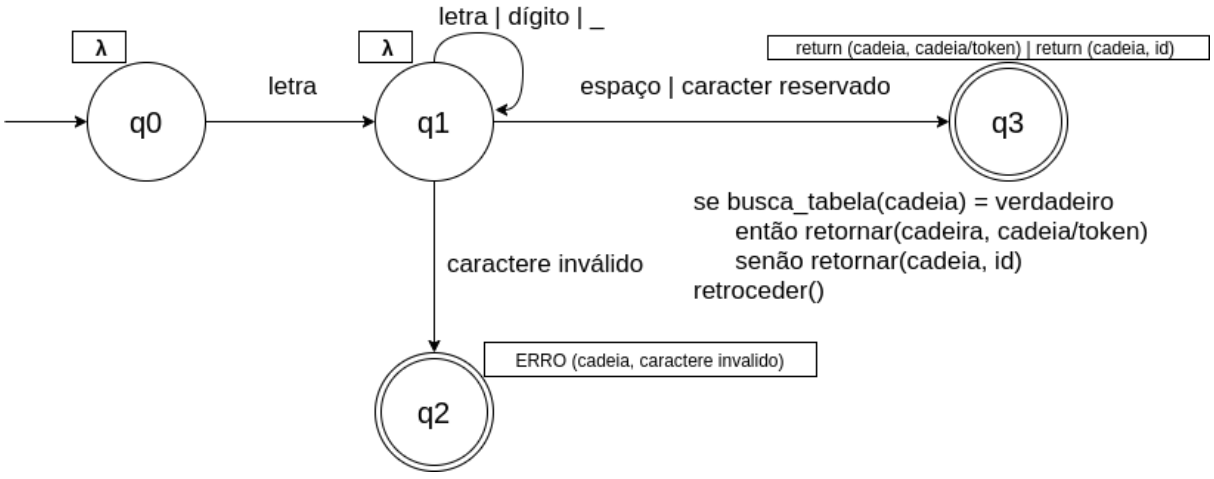
1.4 PALAVRAS RESERVADAS E IDENTIFICADORES

Diferentemente dos autômatos anteriores, foi tomada a decisão de projeto de usar o mesmo autômato para reconhecer palavras reservadas da linguagem e identificadores. Essa decisão foi tomada com base em sugestão dada em aula, já que é mais eficiente permitir que o autômato para identificadores reconheça a cadeia, e ao final, verifique numa tabela se essa faz parte do conjunto de palavras reservadas.A tabela de palavras reservadas usada na implementação pode ser conferida abaixo.

Program
Begin
End
Const
Var
Real

Integer
Procedure
Else
Read
Write
While

Do
For
If
Then
Numero_int
Numero_real



Como decisão de projeto foi definido que um identificador deve ser formado por letras ou dígitos numéricos, sendo o primeiro caractere obrigatoriamente uma letra, o que é assegurado pelo autônomo acima.

Um “caractere inválido” pode ser definido aqui como um caractere diferente de espaço, caracteres reservados, letras ou dígitos numéricos.

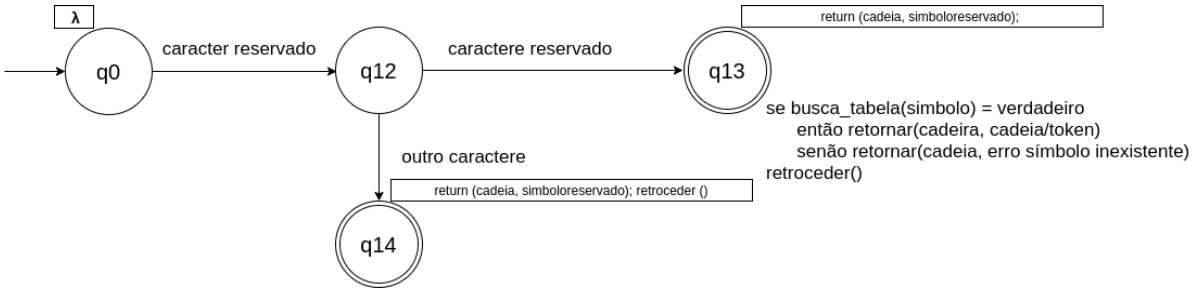
1.5 SÍMBOLOS RESERVADOS

Optou-se por criar um autômato específico para o reconhecimento de símbolos nomeados como símbolos reservados, esses foram listados na tabela abaixo. Perceba que os símbolos reservados são compostos basicamente pelos caracteres reservados, já listados acima, e símbolos formados por dois caracteres reservados.

=
<>
>=
<=
>
<

+
-
*
/
:=
;

:
.
,
(
)



A inserção de um segundo caractere reservado força a verificação desse símbolo numa tabela onde só são os aceitos símbolos <>, >=, <=< :=, uma forma de evitar que símbolos não interpretados pela linguagem como +=, por exemplo, sejam aceitos.

Para solucionar este caso uma verificação simples do tamanho da cadeia impede símbolos como <>> e =+.

Um “outro caractere” aqui pode ser definido aqui como qualquer caractere diferente de um caractere reservado.

1.6 AUTÔMATO GERAL

O autômato usado para codificação do projeto foi basicamente a junção de todos autômatos comentados anteriormente, com um tratamento espaço em branco e com a inclusão de um estado que aponta a inserção de um caractere inválido, que é basicamente qualquer caractere que seja diferente de letras, dígitos numéricos, chaves e caracteres reservados. O autômato geral será anexado ao arquivo.zip e enviado juntamente com o código-fonte.

O tratamento de erros do analisador léxico implementado se propôs a ser um pouco mais informativo. Através da criação de estados de erros, o usuário tem acesso a informações a respeito de números inteiros e reais mal formados, inserção de caracteres inválidos e símbolos inexistentes, assim como a notificação de erros quanto a comentários em aberto.

2. IMPLEMENTAÇÃO DOS AUTÔMATOS

A implementação dos autômatos foi realizada dividindo-se em dois processos de execução. Primeiro, se reconhece a cadeia que será analisada. Para isso, lê-se a posição corrente do arquivo até encontrar um caractere que não seja espaço branco, quebra de linha ou fim do arquivo. Então a partir desse primeiro caractere, seleciona-se dentre um dos possíveis autômatos que ele pode ser:

1. Autômato de números inteiros e reais;
2. Autômato de palavras reservadas e identificadores;
3. Autômato de comentários;
4. Autômato de símbolos;

Daí, parte-se então para a análise da cadeia que pode ser finalizada de duas maneiras. Ou caso seja identificado um caractere inválido para aquele autômato, ou caso seja atendida uma condição de finalize o autômato selecionado. Feito isso, realiza-se uma análise da cadeia e então, ela é escrita no arquivo de saída.txt.

3. INSTRUÇÕES PARA COMPILAR O CÓDIGO-FONTE

Para compilar o programa, basta entrar no arquivo analisador_lexico e digitar no terminal:

gcc main.c − o main

Em seguida, para execução do arquivo, deve-se colocar também o arquivo o qual se deseja realizar a análise léxica. Foram deixados alguns arquivos de exemplos que podem ser utilizados para essa finalidade. Dessa forma, basta seguir o exemplo a seguir:

./main programa_teste_*1.txt*

Foi deixado um programa_teste_1.txt e programa_teste_2.txt para mais exemplos de execução.

Após realizar a execução, o arquivo estará presente no arquivo *saida.txt.*

Por fim, segue um exemplo de execução:

programa_teste_1.txt

Entrada:	Saída:

program p;
var x@: integer;
begin
  x:=1;
  while (x<3) do
  x:=x+1;
end.
	Analise lexica:
	program, comando_reservado_program
	p, identificador
	;, comando_reservado_ponto_virgula
	var, comando_reservado_var
	x, identificador
	@:, simbolo_nao_reconhecido
	integer, comando_reservado_integer
	;, comando_reservado_ponto_virgula
	begin, comando_reservado_begin
	x, identificador
	:=, comando_reservado_atribuicao
	1, num_int
	;, comando_reservado_ponto_virgula
	while, comando_reservado_while
	(, comando_reservado_abre_parenteses

	x, identificador <, comando_reservado_menor 3, num_int), comando_reservado_fecha_parenteses do, comando_reservado_do x, identificador :=, comando_reservado_atribuicao x, identificador +, comando_reservado_mais 1, num_int ;, comando_reservado_ponto_virgula end, comando_reservado_end ., comando_reservado_ponto

programa_teste_2.txt

Entrada	Saida

programa nome2;
{exemplo 2}
var a:real;
var b:integer;
proceduer nomep(x: real);
var a,c: integer;
begin
read(c, a);
if a<x+c then
begin
a:=c+x;
write(a);
end
else c:=a+x;
end;
begin {programa principal}
read(b);
nomep(b);
end.
	Analise lexica:
	programa, identificador
	nome2, identificador
	;, comando_reservado_ponto_virgula
	{exemplo 2}, comentario
	var, comando_reservado_var
	a, identificador
	:, comando_reservado_dois_pontos
	real, comando_reservado_real
	;, comando_reservado_ponto_virgula
	var, comando_reservado_var
	b, identificador
	:, comando_reservado_dois_pontos
	integer, comando_reservado_integer
	;, comando_reservado_ponto_virgula
	proceduer, identificador
	nomep, identificador
	(, comando_reservado_abre_parenteses
	x, identificador
	:, comando_reservado_dois_pontos
	real, comando_reservado_real
	);, simbolo_nao_reconhecido
	var, comando_reservado_var
	a, identificador
	,, comando_reservado_virgula

	c, identificador :, comando_reservado_dois_pontos integer, comando_reservado_integer ;, comando_reservado_ponto_virgula begin, comando_reservado_begin read, comando_reservado_read (, comando_reservado_abre_parenteses c, identificador ,, comando_reservado_virgula a, identificador);, simbolo_nao_reconhecido if, comando_reservado_if a, identificador <, comando_reservado_menor x, identificador +, comando_reservado_mais c, identificador then, identificador begin, comando_reservado_begin a, identificador :=, comando_reservado_atribuicao c, identificador +, comando_reservado_mais x, identificador ;, comando_reservado_ponto_virgula write, comando_reservado_write (, comando_reservado_abre_parenteses a, identificador);, simbolo_nao_reconhecido end, comando_reservado_end else, comando_reservado_else c, identificador :=, comando_reservado_atribuicao a, identificador +, comando_reservado_mais x, identificador ;, comando_reservado_ponto_virgula end, comando_reservado_end ;, comando_reservado_ponto_virgula begin, comando_reservado_begin {programa principal}, comentario read, comando_reservado_read (, comando_reservado_abre_parenteses b, identificador);, simbolo_nao_reconhecido nomep, identificador (, comando_reservado_abre_parenteses b, identificador

	);, simbolo_nao_reconhecido end, comando_reservado_end ., comando_reservado_ponto