

Reconhecimento de Tokens

Uma das técnicas de reconhecimento de tokens é o uso de diagramas de transição que são a representação de um autômato finito. Um diagrama de transição é um grafo orientado em que os nodos representam estados do autômato finito. Dependendo a forma como forem construídos estes diagramas podem corresponder a um autômato finito determinístico ou não-determinístico. Caso haja a necessidade de usar um autômato finito não-determinístico, ao implementar, o mesmo deve ser transformado em um autômato finito determinístico equivalente.

A Figura 1 apresenta um exemplo para um diagrama de transição que reconhece o token **identificador** em uma linguagem típica de programação (inicia com um caractere, podendo ser seguido por letras e ou caracteres intercalados):



Figura 1 – Diagrama de transição para reconhecimento de um identificador.

No diagrama da Figura 1, os estados são representados por círculos, e o estado final por um círculo duplo. Os caracteres lidos são representados sobre os arcos que identificam as transições do autômato finito. Nesse diagrama “outros” significa qualquer outro caractere que não seja letra ou dígito. Este caractere indica o fim do identificador. Esse estado final (o “3”) retorna o token e o insere na tabela de símbolos.

O “*” sobre o estado “3” indica que o último caractere lido deve ser devolvido à entrada pois pertence ao próximo token a ser reconhecido. Assim, ele deve ser “devolvido” para ser tratado junto ao próximo lexema.

Outro exemplo é mostrado na Figura 2, um diagrama de transição para reconhecer tokens de operadores relacionais:

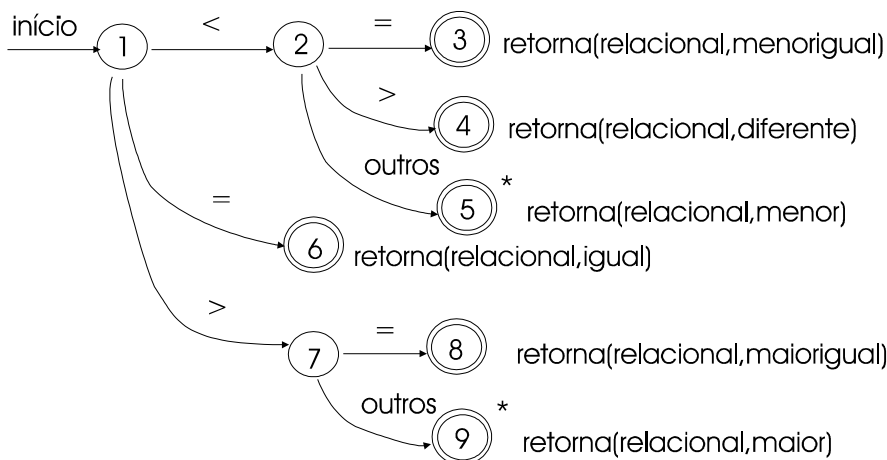


Figura 2 – Diagrama de estado para reconhecimento de operadores relacionais.

A estrutura básica de um analisador léxico reflete a estrutura de um comando de múltipla seleção: o SWITCH/CASE. Por exemplo, o reconhecimento dos operadores relacionais mostrados acima seria implementado como o código apresentado na Figura 3.

<pre> estado := 1; final := false; for(; notfinish ;) switch estado { case 1 : lecaracter(x); switch x { case '<' : estado := 2; break; case '=' : estado := 6; break; case '>' : estado := 7; } case 2 : lecaracter(x); switch x { case '=' : estado := 3; break; case '>' : estado := 4; break; default : devolve(x); estado := 5; } break; case 3 : final := true; retorna(relacional,menorigual) break; </pre>	<pre> case 4 : final := true; retorna(relacional,diferente) break; case 5 : final := true; retorna(relacional,menor) break; case 6 : final := true; retorna(relacional,igual) break; case 7 : lecaracter(x); if x = '=' then estado := 8 else { devolve(x); estado := 9; } break; case 8 : final := true; retorna(relacional,maiorigual) break; case 9 : final := true; retorna(relacional,maior) } </pre>
---	--

Figura 3 – Pseudo-código para implementação do diagrama de transição apresentado na Figura 2.

Exercícios

1. Construa expressões regulares para reconhecer:
 - 1.1. Identificadores, os quais devem iniciar por uma letra, podendo ser seguidos de letras e dígitos intercalados.
 - 1.2. Números inteiros, formados por um ou mais dígitos
 - 1.3. Comentários, delimitados por /* e */
2. Desenhe o autômato das expressões regulares definidas no Exercício 1.
3. Implemente um código para o autômato desenvolvido no Exercício 2.

Respostas

1. Construa expressões regulares para reconhecer:

1.1. Identificadores, os quais devem iniciar por uma letra, podendo ser seguidos de letras e dígitos intercalados.

```
BRANCO [ \t\n ]
LETRA [ a-zA-Z ]
DIGITO [ 0-9 ]
{LETRA} [ {LETRA} | {DIGITO} ] *
```

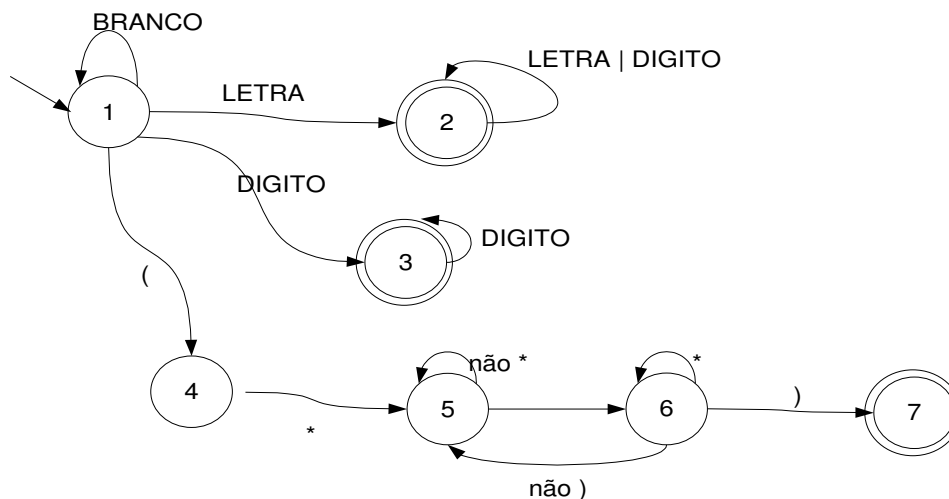
1.2. Números inteiros, formados por um ou mais dígitos

```
[ NUM ] +
```

1.3. Comentários, delimitados por /* e */

```
"\*" . "*" /"
```

2. Desenhe o autômato das expressões regulares definidas no Exercício 1.



3. Implemente um código para o autômato desenvolvido no Exercício 2.

```
for( car = getchar() ; car == " " || car == "\t" || car == "\n" ; car = getchar() );

if( isalpha(car) ) {
    for( car = getchar() ; isalpha(car) || isdigit(car) ; car = getchar() );
    printf("Encontrei um identificador");
}
if( isdigit(car) ){
    for( car = getchar() ; isdigit(car) ; car = getchar() );
    printf("Encontrei um numero");
}
if( car = "/" ) {
    car = getchar();
    if( car == "*" )
        do {
            for( car = getchar() ; car != "*" ; car = getchar() );
            car = getchar();
        } while( car != "/" )
}

/* para qquer outra situação, erro */
```