Trabalho Prático de Linguagens Formais e Autômatos

Matheus Ávila Moreira de Paula

2025

1 Introdução

Um analisador léxico é a primeira parte de um compilador e é responsável por reconhecer *tokens*, e seus valores associados, no código escrito pelo programador. Um analisador léxico pode ser visto como um **Autômato Finito Determinístico**. O trabalho prático (TP) deve ser realizado em grupos de até 3 pessoas e consiste na implementação de um **Analisador Léxico simplificado**. O trabalho poderá ser feito em C++ ou python3. As próximas seções deste documento descrevem as funcionalidades exigidas no trabalho, critérios de avaliação e alguns direcionamentos para o trabalho.

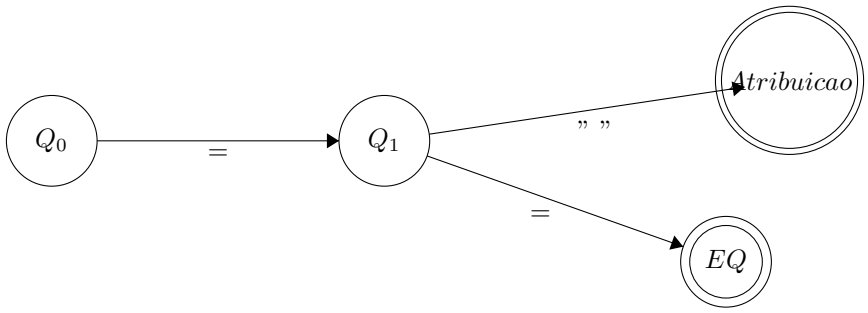
1.1 Descrição do Trabalho

O trabalho consiste em implementar uma parte do analisador léxico. Esta parte que deverá ser implementada é o reconhecedor de *tokens*. *Tokens* são identificadores únicos definidos durante o desenvolvimento de uma linguagem de programação e tem como objetivo facilitar o desenvolvimento da próxima etapa do compilador.

A entrada do programa deverá ser um arquivo de texto, cujo nome deve ser espeficidado como parâmetro pela linha de comando, composto por um código escrito em uma sintaxe semelhante ao C, mas limitada aos caracteres descritos na tabela 1. A saída deverá ser um novo arquivo contendo a conversão do arquivo lido em *tokens*. Assuma que cada trecho do código fonte que será convertido em um único *token* está separado por um espaço em branco(" "). Observe que essa suposição tem como objetivo facilitar a criação do AFD já que o espaço em branco poderá ser usado como marcador do fim do *token*. Um exemplo deste uso é exibido no AFD abaixo.

Table 1: Tabela de tokens da linguagem C simplificada

Token	Texto Lido	Token	Texto Lido
LParenteses	(	SUM	+
RParenteses	)	SUB	-
LChave	{	MULT	*
RChave	}	DIV	/
Lcolchete	[	RESTO	%
RColchete	]	INTDEF	int
EQ	==	FLOATDEF	float
Atribuicao	=	AND	&&
GEQ	>=	OR	||
LEQ	<=	NUM_INT	$(digitos)^+$
GT	>	NUM_FLOAT	$(digitos)^*.(digitos)^+$
LT	<	DIF	!=
NEG	!	CHAR_TYPE	char
Virgula	,	BOLL_TYPE	bool
PVirgula	;	RETURN	return
IF	If	VAR	VAR(letras,digitos)$^+$



A lista de *Tokens* que devem ser utilizados para o reconhecimento do código fonte está descrita na tabela 1

1.2 Exemplo de execução

Entrada do programa: int VARa = VARb - 4;

Saída referente a essa entrada: INTDEF VAR Atribuicao VAR SUB NUM_INT

2 Critérios de Avaliação

- Modularização;
- Legibilidade (nomes de variáveis significativos, código bem formatado, uso de comentários);

- Documentação;
- Funcionamento correto do analisador léxico;
- Leitura e escrita de acordo com o especificado no trabalho;

3 Algumas Dicas

Antes de implementar o AFD faça um desenho do autômato como um guia. A partir das transições reconheça o alfabeto de entrada do seu AFD. Cuidado com *tokens* que possuem o mesmo início na coluna "texto lido"! Eles devem seguir para os mesmos estados, caso contrário estará sendo criado um não-determinismo no autômato. Por exemplo, ao ler o caractere "=" será necessário ler outro caractere para decidir se o *token* que está sendo formado é um EQ ou Atribuicao. Use o AFD exibido na seção 1.1 como guia.

3.1 Submissão

Os arquivos do trabalho devem ser submetidos no Portal Didático da disciplina. A documentação deve conter o nome completo de cada integrante do grupo.