UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Faculdade de Ciências e Tecnologia

Departamento de Matemática e Computação

Bacharelado em Ciência da Computação

Disciplina: Compiladores – 02/2017

Projeto Parte 1 - Analisador Léxico para LALG

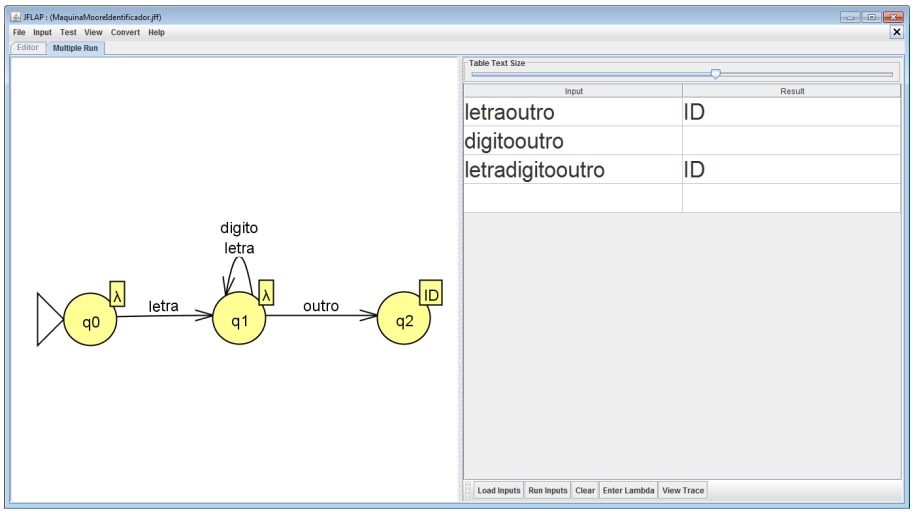
1. Quais são as funções do analisador léxico nos compiladores/ interpretadores?

2. Defina formalmente, através de expressões regulares sobre o conjunto de caracteres ASCII, a sintaxe de cada um dos tipos de lexemas a serem extraídas do texto-fonte pelo analisador léxico, bem como de cada um dos espaçadores e comentários. Ex:

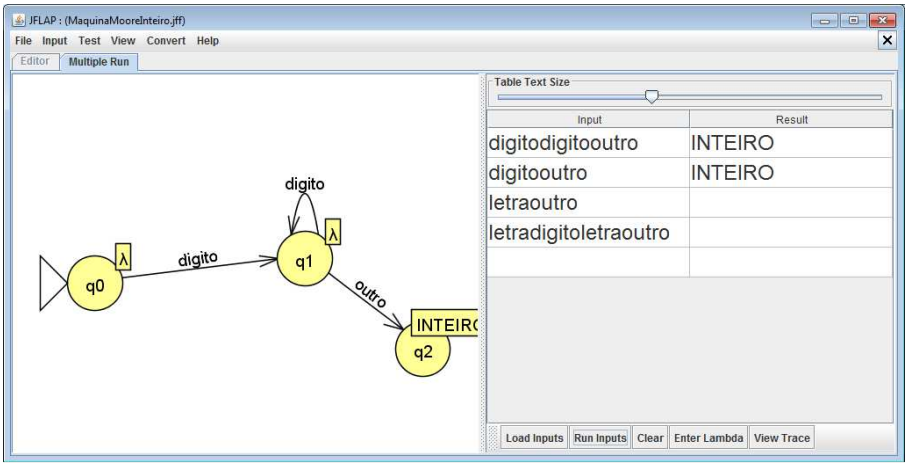
Token	Expressão regular
IDENTIFICADOR	[a-zA-Z] ([a-zA-Z] | [0-9]) *
PALAVRA_RESERVADA	(if | then | else | end | while | do | ...)
SIMBOLOS_ESPECIAIS	(> | < | <= | >= | := | ; |...)
SIMBOLOS_INVALIDOS	(@ | # | & | ...)
COMENTÁRIOS	{} | //

3. Converta cada uma das expressões regulares em um autômato finito com saída nos estados (*Moore Machine* - JFlap), que emita como saída a lexema encontrada ao abandonar cada um dos estados finais para iniciar o reconhecimento de mais uma lexema do texto.

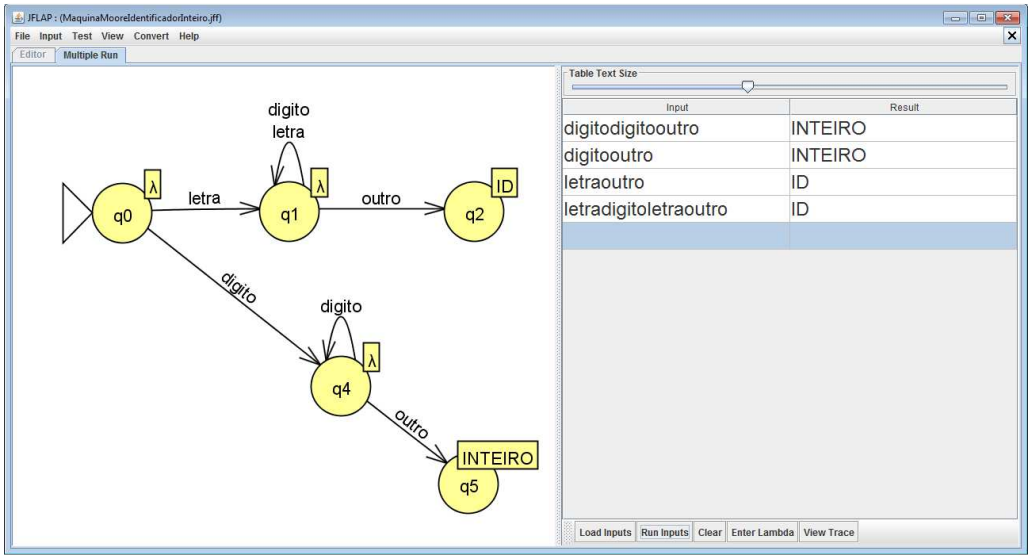
Ex: Máquina de *Moore* que retorna um **identificador.** Obs: a transição **outro** indica, por exemplo, um espaço em branco, uma quebra de linha, um símbolo especial, etc.



Ex: Máquina de *Moore* que retorna um **inteiro**



4. Crie uma Máquina de Moore (função *combinate automata* do JFlap) que aceite todos os tokens da linguagem LALG a partir de um mesmo estado inicial, mas que apresente um estado final diferenciado para cada uma delas. Ex: máquina que retorna apenas **identificador** e **inteiro** – gerado a partir da função *combinate automata*.

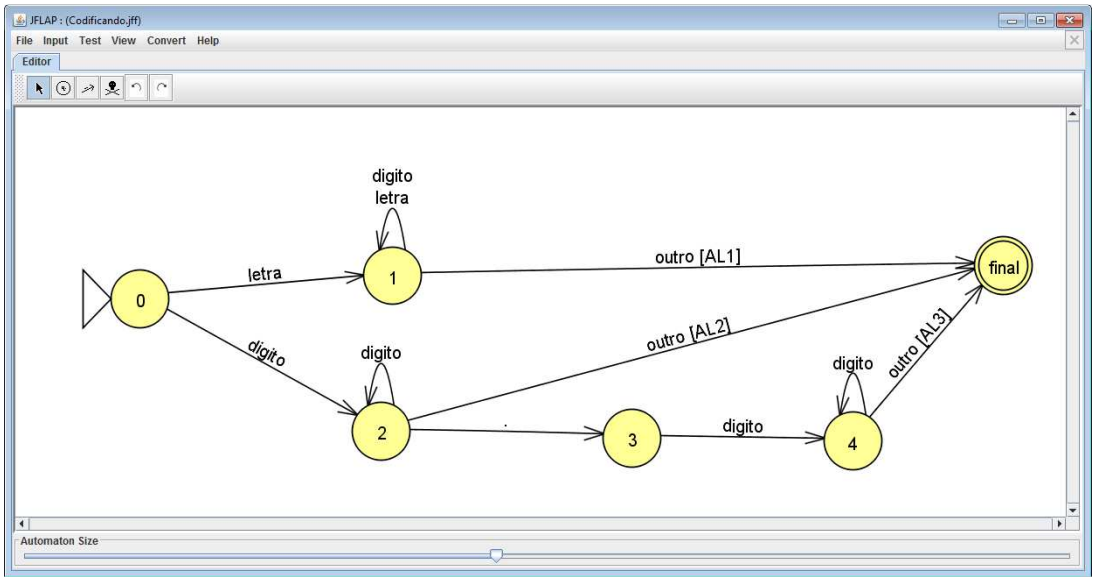


5. Converta o autômato finito em uma sub-rotina, escrita na linguagem de programação de sua preferência. Não se esqueça que o final de cada lexema é determinado ao ser encontrado espaço, quebra de linha ou uma outra lexema. Esse símbolo não pode ser perdido, devendo-se, portanto, tomar os cuidados de programação que forem necessários para reprocessá-los, apesar de já terem sido lidos pelo autômato. Ex: autômato parcial que reconhece AL1, AL2 e AL3, descritos a seguir:

AL1: Verifica se a lexema é um identificador ou palavra reservada
AL2: Verifica se a lexema é um número inteiro
AL3: Verifica se a lexema é um número real
AL?: ...

Obs: (-) caso seja encontrado um símbolo inválido, o mesmo deverá ser informado e a leitura deverá prosseguir;

(--) em caso de comentários no código, os caracteres devem ser descartados e, após encontrar o fim do comentário, continuar a extração.



Codificando:

```
Caso estado == 0
      Se caractere lido ∈ letra
            Então  lexema = lexema + caractere lido
                   vá_para estado 1
            Senão se caractere lido ∈ digito
                  Então lexema = lexema + caractere lido
                        vá_para estado 2
                  Senão {tratar demais estados...}

Caso estado == 1
      Se caractere lido ∈ letra ou dígito e não chegou ao final da lexema(\n, espaço)
            Então lexema = lexema + caractere lido
                  Se chegou ao final da lexema (encontrou \n, espaço)
                        Então busca lexema na tabela de palavras reservadas
                              Se encontrou
                                    Então retorna PALAVRA_RESERVADA
                                    Senão retorna IDENTIFICADOR

Caso estado == 2
      Se caractere lido ∈ dígito e não chegou ao final da lexema(\n)
            Então lexema = lexema + caractere lido
                  Se chegou ao final da lexema (encontrou \n)
                    Então verifica se o inteiro está no limite da linguagem (overflow)
                        Se está no limite
                              Então retorna INTEIRO
                              Senão retorna ERRO_NUM_INTEIRO

      Se caractere lido == . //é um número real
            Então lexema = lexema + caractere lido
                  vá_para estado 3

Caso estado == 3
      Se caractere lido ∈ dígito e não chegou ao final da lexema(\n)
            Então lexema = lexema + caractere lido
                  vá_para estado 4

Caso estado == 4
      Se chegou ao final da lexema (encontrou \n)
            Então verifica se o real está no limite da linguagem (overflow)
            Se está no limite
                Então retorna REAL
                Senão retorna ERRO_NUM_REAL
```

6. Crie um programa principal, **com interface gráfica,** que chame repetidamente a sub-rotina construída, e a aplique sobre um arquivo do tipo texto contendo o texto-fonte a ser analisado. Após cada chamada, esse programa principal deve imprimir a lexema extraída, o token (classificação de acordo com a ER), a coluna inicial e final, a linha e descreva o escopo (se existir). Faça o programa parar quando o programa principal receber do analisador léxico uma lexema especial indicativo da ausência de novas lexemas no texto de entrada. **Obs: serão disponibilizados códigos-fonte para a realização de testes.**

Exemplo de código:

```
program programa1;
begin
    @aux=10;
end.
```

Lexema lida	Token	Coluna inicial	Coluna final	Linha
program	PALAVRA_RESERVADA_PROGRAM	0	7	0
programa1	IDENTIFICADOR	8	...	...
begin	PALAVRA_RESERVADA_BEGIN	...	...	...
@	SIMBOLO_INVALIDO_@	...	...	...
aux	IDENTIFICADOR	...	...	...
...	...	...	...	...

7. Relate o funcionamento do analisador léxico construído, incluindo no relatório: descrição teórica do programa; descrição da sua estrutura; descrição de seu funcionamento; descrição dos testes realizados e das saídas obtidas.

Data e forma de Entrega:

- data limite: 02/10/17
- enviar o relatório (formato pdf) e código-fonte (com executável) em um único arquivo compactado para o email: olivete@fct.unesp.br
- executar os testes e apresentar para o professor até a data limite.