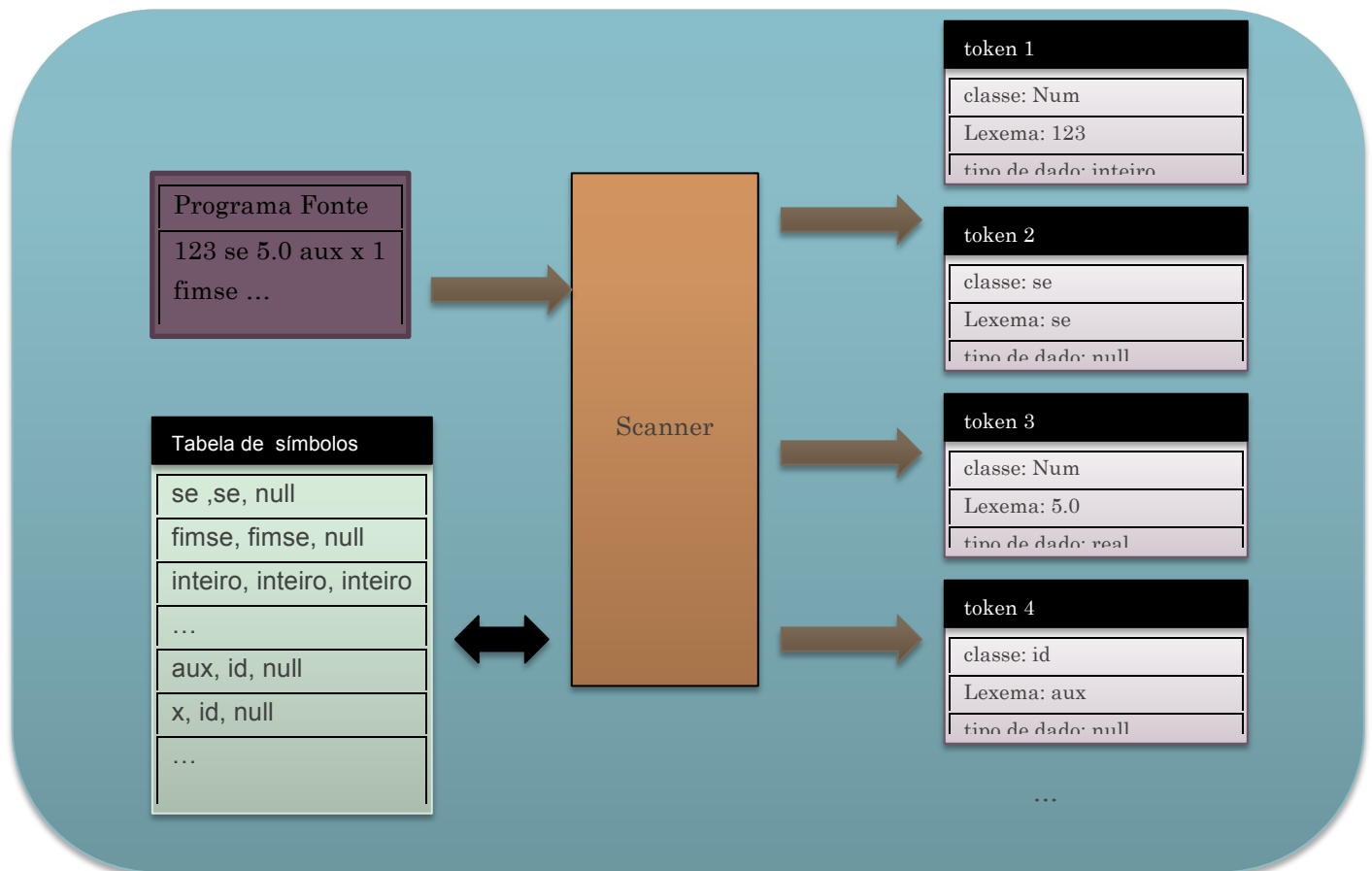


## COMPILADORES – TRABALHO 1 – T1

# Estudo de Caso: Analisador Léxico e Tabela de Símbolos



## 1 . Descrição

Desenvolvimento da atividade prática Trabalho 1 (T1) – Analisador léxico e tabela de símbolos. O valor dessa atividade é 10,0 e compõe a média de aprovação na disciplina conforme definido no plano de curso.

Verifique as regras gerais aplicáveis à todos os trabalhos T1, T2 e T3 do semestre no arquivo disponível na Plataforma Turing: [https://turing.inf.ufg.br/pluginfile.php/92669/mod\\_resource/content/1/Descricao-GERAL%20-%20Trabalho%20Compiladores.pdf](https://turing.inf.ufg.br/pluginfile.php/92669/mod_resource/content/1/Descricao-GERAL%20-%20Trabalho%20Compiladores.pdf)

## 2 - Entregáveis

1 – (T1.1) Atividade 1 complementar ao Trabalho 1 - Entregar na data determinada pelo professor, EXCLUSIVAMENTE via plataforma Turing, o Autômato Finito Determinístico que reconhecerá os padrões definidos na TABELA 1 conforme solicitado na Atividade Complementar T1.1. Essa atividade é INDIVIDUAL e vale 1,0 na nota final do trabalho T1. Não será computada nota de atividade T1.1 entregue após a data determinada para a entrega.

2 – (Código de implementação do T1) - Entregar na data determinada pelo professor, EXCLUSIVAMENTE via plataforma Turing, O CÓDIGO desenvolvido para o analisador léxico e tabela de símbolos a ser descrito nas seções abaixo.

## 3 - Notas

1 – T1.1. = 1,0

2 – A entrega e arguição oral terão o valor total de 9,0.

3 – Nota total T1= Nota T1.1 + Nota avaliação T1.

## 4 – O que fazer?

O programa a ser desenvolvido deverá estar de acordo com as decisões de projeto definidas abaixo e será avaliado pelo professor em relação a cada critério estabelecido. Portanto, leia com atenção.

Desenvolver um programa computacional na linguagem escolhida que implemente:

- 1- Uma estrutura composta heterogênea (nó, registro,...) **TOKEN** :
  - a. Seu nome será **TOKEN** - armazenará, no momento apropriado da análise a classificação da palavra e seus atributos;
  - b. Possuirá três campos (seus atributos):
    - i. **classe** (armazenará a classificação do lexema reconhecido);
    - ii. **lexema** (armazenará a palavra computada);
    - iii. **tipo** armazenará:
      - Inteiro: Caso seja identificada uma constante inteiro;
      - Real: Caso seja identificada uma constante real;
      - literal: Caso seja identificada uma constante literal;
      - NULO caso a classe não se enquadre nos itens anteriores.
- 2- Uma **TABELA DE SÍMBOLOS**:
  - a. Que armazenará EXCLUSIVAMENTE palavras-chaves da linguagem e também tokens ID identificados no fonte pelo scanner durante o processo de análise.
  - b. Poderá ser utilizada uma estrutura de dados do tipo lista ou *hash* para sua composição.
  - c. Cada item da tabela será um nó do tipo **TOKEN** definido no item 1 acima.: token, lexema, tipo.
  - d. As operações a serem realizadas para manipulação da Tabela de Símbolos são: Inserção e Atualização, procedimentos a serem implementados.
  - e. Poderão ser utilizadas estruturas disponíveis em bibliotecas da linguagem escolhida para a implementação do projeto.
  - f. Ao iniciar o programa, a tabela de símbolos deverá ser preenchida com todas as PALAVRAS RESERVAS da linguagem, TABELA 2. Para cada nó **TOKEN** da tabela, campos classe e lexema receberão o mesmo valor, ou seja, a mesma palavra. O campo tipo:
    - i. Para palavras reservadas associadas à tipo de dados (inteiro, literal e real) assumirá o mesmo valor do lexema;
    - ii. As demais palavras reservadas receberão o valor do tipo =NULO.
- 3- Uma **função SCANNER** que:
  - Possua o cabeçalho: **TOKEN SCANNER (parâmetros de entrada)**
    - i. A função SCANNER retornará um único TOKEN a cada chamada. TOKEN é o retorno de SCANNER, uma variável conforme definido em 1;
    - ii. SCANNER é o nome do procedimento;

- iii. Parâmetros de entrada serão definidos pelo programador para ajustar a leitura do arquivo para palavra por palavra;
- Implementará a máquina reconhecedora de padrões projetada no AFD definido na seção 3-1.
  - i. Efetuará a leitura do texto fonte caractere a caractere, esses caracteres são as entradas para o AFD (tabela de transição) que, partindo do estado inicial, realiza uma transição de estado a cada reconhecimento de símbolo até que o estado final seja alcançado.
  - i. Ao encontrar um estado final, uma cadeia será reconhecida, dessa forma a máquina para, preenche uma estrutura do tipo TOKEN (item 1) com a classe, lexema e TIPO.
    - a. Se a classe for NUM, preencher o campo tipo com “inteiro” ou “real”, dependendo do tipo da constante e retornar o TOKEN para quem invocou o SCANNER.
    - b. Se a classe for lit preencher o campo tipo com “literal” e retornar o TOKEN para quem invocou o SCANNER.
    - c. Se a classe for **IDENTIFICADOR**, verificar se o mesmo se encontra na tabela de símbolos.
      - Se o lexema **existir** na TABELA DE SÍMBOLOS, retornar na função SCANNER o TOKEN que está na tabela de símbolos ;
      - Se o lexema **não existir** na TABELA DE SÍMBOLOS, inserí-lo e retornar na função SCANNER o TOKEN.
    - d. Se o TOKEN reconhecido for **diferente de IDENTIFICADOR, NUM e LIT, preencher o campo TIPO com NULO e** retornar o TOKEN para quem invocou o SCANNER.
- Se a sequência de caracteres lida do fonte for classificada pelo autômato como ERRO, retornar na função SCANNER o token ERRO com um número que identifique o tipo de erro ocorrido. Exemplo TOKEN: classe – ERRO1, lexema: erro1 e tipo: Nulo, ou ERRO2, ...

4- Uma função **ERROR** que:

- a. Receberá o número do erro identificado e mostrará na tela o tipo do erro léxico encontrado e a linha e a coluna do texto fonte na qual o erro ocorreu. Exemplo de mensagem a ser emitida na tela: “ERRO1 – Caractere inválido na linguagem , linha 2, coluna 1”.
- b. O(A)s aluno(a)s deverão mapear todos os tipos de erros léxicos possíveis dentro do escopo deste projeto.

5- Um programa **PRINCIPAL** no qual :

- a. Haverá abertura do arquivo fonte;
- b. Invocação do SCANNER (dentro de um estrutura de repetição) para que retorne um TOKEN por chamada;

- c. A cada TOKEN retornado pelo SCANNER, emitirá uma mensagem como o exemplo abaixo:  
Classe: Num, Lexema: 123, Tipo: NULL
- d. Cada token erro retornado pelo SCANNER, deverá ser impresso na tela como no exemplo: “  
ERRO1 – Caractere inválido na linguagem , linha 2, coluna 1”
- e. O programa só finalizará ao realização toda a leitura do Código Fonte.

## 5 – Resultado final do Scanner

O Scanner deverá ler todo o texto fonte, realizando todas as tarefas especificadas na seção 2. O resultado será a emissão na tela de todos os TOKENs reconhecidos. Observe o desenho da FIGURA 1 abaixo, nela é apresentada uma amostra da saída do programa a ser desenvolvido em T1.

Figura 1 – Resultado do Scanner.

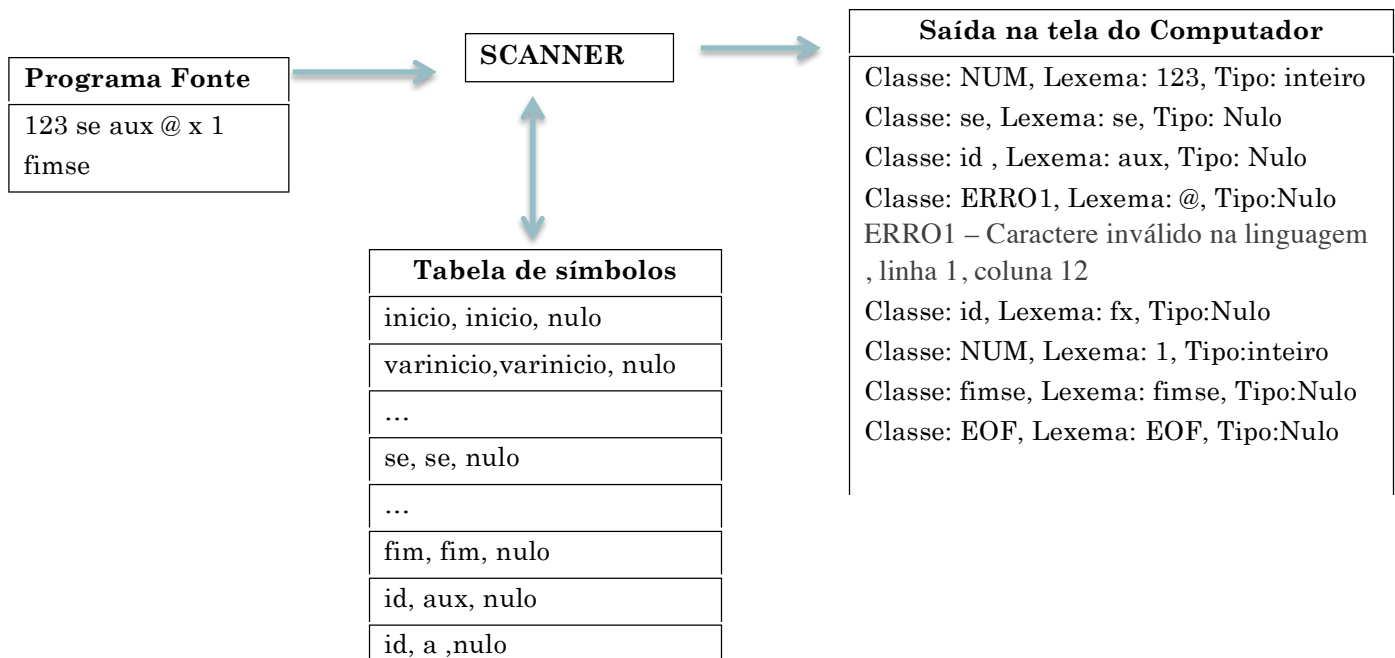


TABELA 1 – Tokens a serem reconhecidos pelo analisador Léxico para a linguagem MGol.

token	Significado	Características/ Padrão
<b>Num</b>	Constante numérica INTEIRA ou REAL	$D^+(\backslash.D^+)?((E e)(+ -)?D^+)?$
<b>Lit</b>	Constante literal	" . * "
<b>id</b>	Identificador	$L(L D _)*$
<b>Comentário</b>	Ignorar comentários, ou seja, reconhecer mas não retornar o token.	{ . * }
<b>EOF</b>	Final de Arquivo	\$   Flag da linguagem (EOF é um único símbolo)
<b>OPR</b>	Operadores relacionais	<, >, >=, <=, =, <>
<b>RCB</b>	Atribuição	<-
<b>OPM</b>	Operadores aritméticos	+, -, *, /
<b>AB_P</b>	Abre Parênteses	(
<b>FC_P</b>	Fecha Parênteses	)
<b>PT_V</b>	Ponto e vírgula	;
<b>ERRO</b>	Qualquer símbolo diferente de qualquer palavra definida.	
<b>Vir</b>	Vírgula	,
	tabulação, espaço em branco, salto de linha	Reconhecidos e ignorados.

TABELA 2 – Palavras reservadas da linguagem MGol a ser reconhecida pelo Analisador Léxico.

Token	Significado
<b>inicio</b>	Delimita o início do programa
<b>varinicio</b>	Delimita o início da declaração de variáveis
<b>varfim</b>	Delimita o fim da declaração de variáveis
<b>escreva</b>	Imprime na saída padrão
<b>leia</b>	Lê da saída padrão
<b>se</b>	Estrutura condicional
<b>entao</b>	Elemento de estrutura condicional
<b>fimse</b>	Elemento de estrutura condicional
<b>repita</b>	Elemento de estrutura de repetição
<b>fimrepita</b>	Elemento de estrutura de repetição
<b>fim</b>	Delimita o fim do programa
<b>inteiro</b>	Tipo de dado inteiro
<b>literal</b>	Tipo de dado literal
<b>real</b>	Tipo de dado real

## 6 – Código para teste

```
inicio
  varinicio
    A literal;
    B inteiro;
    D inteiro;
    C real;
  varfim;
  escreva "Digite B";
  leia B;
  escreva "Digite A:";
  leia A;
  se(B>2)
  entao
    se(B<=4)
    entao
      escreva "B esta entre 2 e 4";
    fimse
  fimse

  B<-B+1;
  B<-B+2;
  B<-B+3;
  D<-B;
  C<-5.0;
  repita (B<5)
    B<-B+1;
  fimrepita
  escreva "\nB=\n";
  escreva D;
  escreva "\n";
  escreva C;
  escreva "\n";
  escreva A;
fim
```