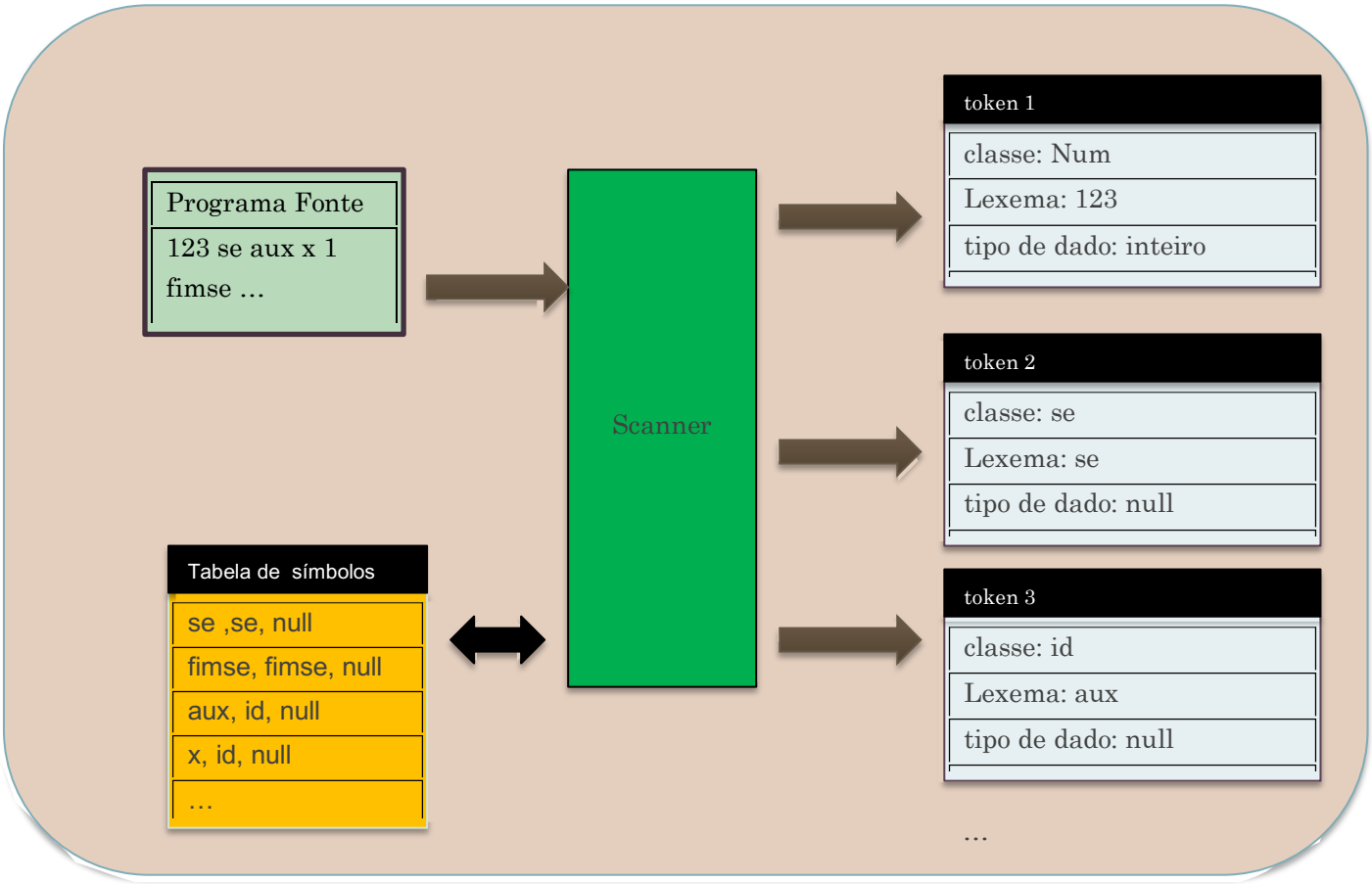
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

COMPILADORES – TRABALHO 1 – T1

Analisador Léxico



Profa. Dra. Deborah Fernandes

1. Descrição

A atividade prática Trabalho 1 (T1) – Analisador Léxico em Compiladores é um componente para a avaliação e desenvolvimento dos conhecimentos desenvolvidos nas disciplinas ofertadas para Ciência da Computação e Engenharia de Computação - Compiladores e Compiladores 1. O valor dessa atividade é 10,0 e compõe a média de aprovação na disciplina conforme definido no plano de curso.

As regras gerais para o desenvolvimento dos três trabalhos T1, T2 e T3 estão descritas do documento **"Regras gerais para T1, T2 e T3"** disponível no módulo "**Trabalho PRÁTICO"** na plataforma Turing.

2 - Entregáveis

1. (Atividade Complementar T1.1) – Valor =1,0 - 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.
2. (Código implementado para o T1) - Entregar na data determinada pelo professor, EXCLUSIVAMENTE via plataforma Turing, o CÓDIGO desenvolvido para o analisador léxico, cujas funcionalidades estão descritas nas seções abaixo.
3. Nota total do trabalho T1 = Nota T1.1 + Nota T1.
4. SOBRE a entrega:
    - Caso seja realizado em duplas, apenas um componente deverá entregá-lo na plataforma.
    - O NOME do código deverá seguir o padrão: T1-NomeAluno1-NomeAluno2-20232.extensão. **Exemplo**: T1-DeborahFernandes-FulanoPrado-20232.c .
    - Se o código possuir vários arquivos, entregar em .zip.
    - A entrega e arguição oral terão o valor total de 9,0.

3 – 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, classe ...) denominada **TOKEN.** Esta estrutura armazenará, no momento apropriado da análise, a classificação da palavra e seus atributos. Ela possuirá três campos (os atributos):

a. **Classe**: armazenará a classificação do lexema reconhecido;
b. **Lexema**: armazenará a palavra computada;
c. **Tipo**: armazenará o tipo de dado do lexema quando for possível determiná-lo nesta análise (inteiro, real ou literal) ou NULO em casos que serão definidos abaixo.

2- Uma **estrutura de dados** (*hash table*, *map*, lista,...) denominada **TABELA DE SÍMBOLOS**:

a. Armazenará, **EXCLUSIVAMENTE,** tokens **ID** e palavras reservadas da linguagem e reconhecido no programa fonte pelo *scanner* durante o processo de análise.
b. Cada item da tabela será um nó do tipo TOKEN como definido no item 1.
c. As operações a serem realizadas para manipulação da Tabela de Símbolos são: Inserção e Busca e Atualização.
d. Estruturas de dados, disponíveis em bibliotecas da linguagem escolhida, podem ser utilizadas.
e. Ao iniciar o programa, a tabela de símbolos deverá ser preenchida com todas as PALAVRAS RESERVAS da linguagem disponíveis na TABELA 2. Os campos classe, lexema e tipo serão preenchidos com a palavra reservada.

3- Uma **função SCANNER** que:

a. Possua o cabeçalho: **token SCANNER (parâmetros de entrada)**
    - Esta função retornará um único TOKEN (definido no item 1) a cada chamada;
    - SCANNER é o nome do procedimento;
    - Parâmetros de entrada serão definidos pelo programador para ajustar a leitura do arquivo fonte para palavra por palavra;
b. Implemente a máquina reconhecedora de padrões projetada no AFD definido na atividade complementar T1.1.
c. Efetue a leitura do texto fonte caractere por caractere. Partindo do estado inicial do AFD, após a leitura do caractere, consulta-se a tabela de transições e realiza-se uma transição de estado. Essa mudança de estados é realizada até que um estado final seja alcançado ou que não haja possibilidade de transição, os caracteres são unidos para a formação de uma palavra (lexema). Ao encontrar um estado final, uma cadeia de caracteres será reconhecida por um padrão (classe). Nesse momento, são preenchidos os campos de um novo no TOKEN. Associado ao estado final temos uma **classe**, a palavra reconhecida é o **lexema**, o campo **tipo** será preenchido conforme:
    - Se a classe = NUM, sendo uma constante numérica inteira, **Tipo** = “inteiro”, se real, **Tipo** = “real. O token é retornado por SCANNER.
    - Se a classe = LIT, **Tipo**= “literal” e retornar o TOKEN para quem invocou o SCANNER. O token é retornado por SCANNER.

- Se a classe = **ID**, preencher Tipo = NULO. Verificar se o lexema deste TOKEN está na **tabela de símbolos**:
  - Se estiver, retornar na função SCANNER o TOKEN que está na tabela de símbolos;
  - Se não estiver, inserir o novo TOKEN na TABELA DE SÍMBOLOS e retorná-lo na função SCANNER.
- Se a classe = ERRO:
  - Emitir, na saída padrão, a descrição do tipo do erro (mensagem para o programador com o tipo do erro identificado) seguida da linha e coluna (do código fonte) nas quais o erro ocorreu.
  - O(A)(s) aluno(a)(s) deverão mapear todos os tipos de erros léxicos possíveis dentro do escopo deste projeto.
  - Exemplo de mensagem a ser emitida na saída: "ERRO LÉXICO – Caractere inválido na linguagem, linha 2, coluna 1".
  - Retornar, na função SCANNER, o TOKEN com os campos classe=ERROR, lexema=NULO e tipo=NULO.
- Se a classe for caractere em branco, espaço, salto de linha, tabulação ou comentário, o scanner reconhece e ignora, reinicia o processo para um novo TOKEN.
- Se a classe for **diferente das anteriores,** preencher o campo TIPO com NULO e retornar o TOKEN na função SCANNER.

d. Um programa **PRINCIPAL** que:
- Efetuará a abertura do arquivo fonte;
- Conterá uma estrutura de repetição que:
  - Invocará a função SCANNER para que retorne um TOKEN por chamada;
  - A cada TOKEN retornado pelo SCANNER:
    a. Se casse = ERRO, ignorar, pois já foi tratado dentro da função SCANNER;
    b. Caso contrário, emitir mensagem como no exemplo: **Classe: Num, Lexema: 123, Tipo: NULL**
  - O loop só finalizará após a leitura de todo o Código Fonte.

TABELA 1 – Símbolos do alfabeto da linguagem MGOL.

Definições	Significado
Dígitos	{0,1,2,3,4,5,6,7,8,9}
Letras (maiúsculas e minúsculas)	{A, B, ...,Z ,a ,..., z}
Demais caracteres	{ ,(vírgula), ;(ponto e vírgula), :(dois pontos), .(ponto), !, ?, \, * ,+ ,- , / , (,), {, }, [,] ,< , >, ='(aspas simples), " (aspas duplas), _(underline)}

TABELA 2 – Tokens a serem reconhecidos pelo analisador Léxico para a linguagem MGol.

Token	Significado	Características/ Padrão
Num	Constante numérica (reconhecer inteiro e real em estados diferentes)	$(D^+(\backslash.D^+)?)$ | (D(\.D⁺)? $((E\|e)(+\|-)?D^+)$?)
Lit	Constante literal	" .* "
id	Identificador	$L(L\|D\|\_)^*$
Comentário	Texto dentro de { }	{ . *}
EOF	Final de Arquivo	Flag da linguagem (EOF é um único símbolo)
OPR	Operadores relacionais	<, >, >= , <= , =, <>
RCB	Atribuição	<-
OPM	Operadores aritméticos	+ , -, *, /
AB_P	Abre Parênteses	(
FC_P	Fecha Parênteses	)
PT_V	Ponto e vírgula	;
VIR	Vírgula	,
ERRO	Qualquer símbolo que não faça parte do alfabeto ou palavra diferente de qualquer padrão definido.	
esp	Espaço em branco	Espaço em branco (reconhecer e ignorar)
tab	Espaço de Tabulação	Espaço de Tabulação (reconhecer e ignorar)
salt	Salto de Linha	Salto de linha (reconhecer e ignorar)

TABELA 3 – Palavras reservadas da linguagem MGol a ser reconhecida pelo Analisador Léxico.

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

4 – Resultado final do Scanner

O *Scanner* deverá ler todo o texto fonte realizando todas as tarefas especificadas na seção 4. O resultado esperado é a emissão na saída padrão de todos os TOKENs reconhecidos. Observe o desenho da FIGURA 1 abaixo, nela é apresentada uma amostra da saída do programa a ser desenvolvido para o T1.

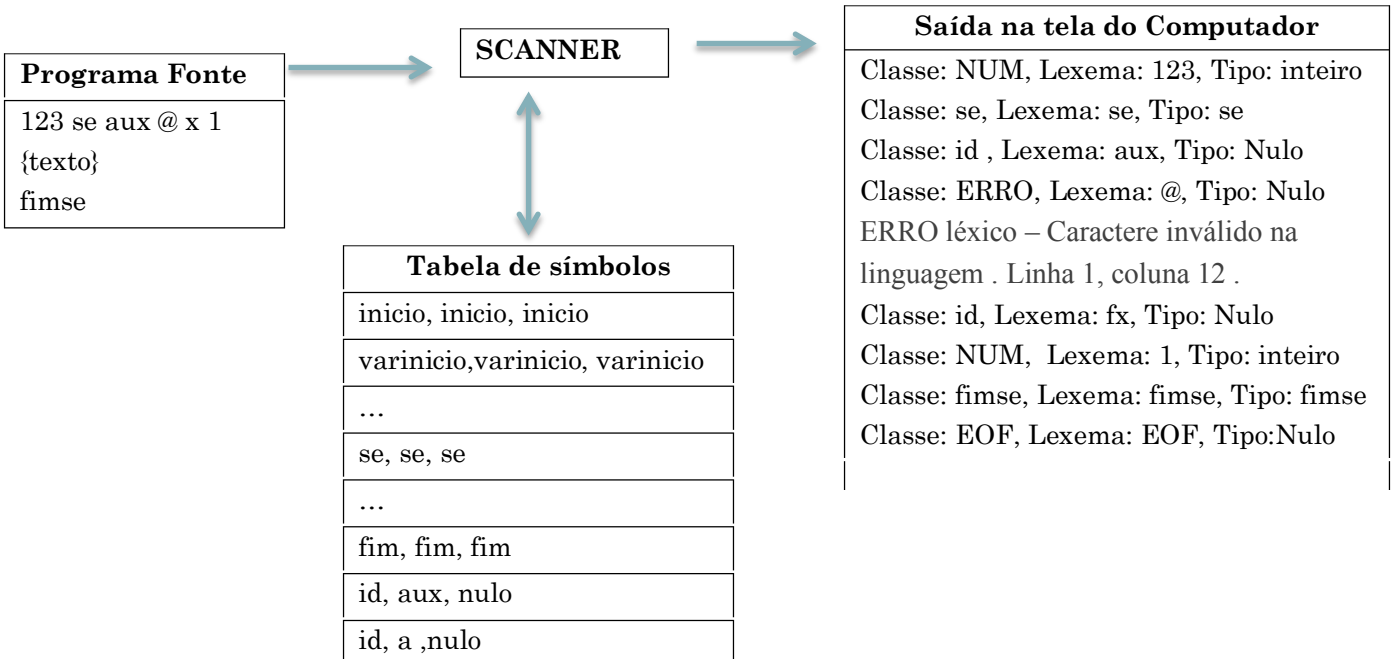


Figura 1 – Resultado do Scanner.

5 – Arquivo Sugestão para teste

Programa fonte em linguagem Mgol: FONTE.ALG.

```
inicio
    varinicio
        literal A,B;
        inteiro B;
        inteiro D;
        real C ;
    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)
      C<-B+2;
      escreva C;
      B<-B+1;
    fimrepita
    escreva "\nB=\n";
    escreva D;
    escreva "\n";
    escreva C;
    escreva "\n";
    escreva A;
fim
```

FIGURA 2 – Código fonte em linguagem MGOL (Fonte.alg).