

# TRABALHO 1 DE LABORATÓRIO

## Estudo de Caso: Implementação analisador Léxico para linguagem MGOL

Esta atividade prática em Compiladores é um componente para a avaliação e desenvolvimento dos conhecimentos envolvidos na disciplina Compiladores. O valor dessa atividade é 10,0 e compõe a média de aprovação na disciplina conforme plano de curso.

Prof. Dra. Deborah Silva Alves Fernandes – UFG/INF

Goiânia, março, 2018.

# 1. INTRODUÇÃO

O trabalho introduzido nesse documento busca a realização de atividade prática em Compiladores e compõe a nota T1 das atividades avaliativas expostas no plano de curso.

A disciplina de compiladores preocupa-se em estudar técnicas e teorias para a construção de um compilador. Para tal, durante o semestre investigar-se-á seus componentes sobre aspectos teóricos e práticos em um estudo de caso. Esse estudo envolverá o desenvolvimento de um compilador que recebe como entrada um arquivo fonte na uma linguagem Mgol, realiza a fase de análise e síntese gerando um arquivo objeto em linguagem C. O arquivo final deverá ser compilável em compilador C, ou seja, o código gerado deverá estar completo para compilação e execução.

## 2. ATIVIDADE PRÁTICA T1

### 2.1. Regras DO TRABALHO

1. Trabalho individual ou em dupla;
2. O trabalho (códigos fonte e executáveis) será entregue via moodle na data definida pelo professor (para cada dia de atraso serão descontados 0,3 por dia até o dia de apresentação).
3. As apresentações serão realizadas nos dias e horários definidos pelo professor (dentro dos horários de aula regulares da disciplina).
4. O professor arguirá o aluno quanto a questões sobre o desenvolvimento do trabalho.
5. Em caso de duplas, o professor escolherá a qualquer momento da apresentação, quem responderá a pergunta a ser realizada. A nota será a mesma para ambos os alunos.
6. O aluno poderá escolher a linguagem de programação que será utilizada para desenvolver o trabalho. Portanto, é de responsabilidade do aluno que no dia da apresentação todo o aparato para execução do trabalho esteja disponível.
7. A evolução do trabalho será acompanhada pela professora durante as aulas até o dia da entrega.
8. Cópias de trabalhos de colegas ou de semestres anteriores terão nota 0,0.
9. Durante a apresentação o professor poderá questionar quaisquer itens relacionados ao trabalho, a análise léxica e tabela de símbolos implementada.
10. Para ser apresentado, o programa deverá estar executando e com as principais funcionalidades implementadas e funcionando.

### 2.2. Atividade a ser desenvolvida

Desenvolver um programa computacional na linguagem escolhida que implemente:

1. Um analisador léxico que reconheça a tabela de tokens disponíveis na Tabela1 para linguagem Mgol.
  - a. A cada leitura de lexema e reconhecimento do token gerar uma estrutura com três campos (token,lexema,tipo) que guardarão as informações do token reconhecido.
2. Para tal, desenvolver um autômato finito determinístico em forma de diagrama e tabela de transições que reconheça a linguagem regular expressa através da Tabela1.
  - a. Desenvolver o AFD em papel e entregar assinado no dia da apresentação;
  - b. Implementar o AFD na forma de tabela de transição (não usar se/senão ou case);
3. Desenvolver uma tabela de Símbolos, utilizar estrutura de dados (sugestão – Tabela *hash*) para armazenar as palavras chave da linguagem expostas na Tabela 2 e os identificadores reconhecidos no fonte pelo analisador léxico .
  - a. Campos a serem armazenados: token, lexema, tipo.
  - b. Algumas definições:
    - i. Token – Classe de palavra identificada.

- ii. Lexema – é a palavra lida do texto do programa
  - iii. Tipo – Tipo do token, quando não souber o tipo deixar o campo vazio.
  - iv. ERRO – Um token que indica que o analisador léxico encontrou um lexema fora do padrão especificado.
4. O programa deverá ler como entrada o programa fonte apresentado na Figura 1 e mostrar na tela o token reconhecido seguido de seu lexema e tipo, quando possível. Qualquer elemento diferente dos tokens definidos, não será reconhecido pelo programa e este retornará na tela ERRO.
  5. Ao encontrar um ERRO o analisador para e emitirá uma mensagem na tela com o tipo de erro encontrado, a linha e coluna onde o erro está.
  6. **O programa deverá conter uma função ou módulo que, quando invocado retornará apenas o token e seus atributos , ou seja, um token por cada chamada.**
  7. Para a apresentação o programa deverá ler o texto fonte e retornar todos os tokens e seus atributos na tela.
  8. Comentários, espaço em branco, tabulação e salto de linha deverão ser reconhecidos porém ignorados.

Tabela 1 – Tokens a serem reconhecidos pelo analisador Léxico para a linguagem ALG.

Token	Significado	Características	Atributos
Num	Constante numérica	$D^+(\backslash.D^+)?(E(+ -)?D^+)?$	Token, Tipo e lexema
Literal	Constante literal	" . * "	Token, Tipo e lexema
id	Identificador	$L(L D _)*$	Token, Tipo, Lexema
Comentário	Ignorar comentários, ou seja, reconhecer mas não retornar o token.	{ . * }	
EOF	Final de Arquivo	Flag da linguagem	Token
OPR	Operadores relacionais	<, >, >=, <=, =, <>	Token, lexema
RCB	Atribuição	<-	Token, lexema
OPM	Operadores aritméticos	+, -, *, /	Token, lexema
AB_P	Abre Parênteses	(	Token, lexema
FC_P	Fecha Parênteses	)	Token, lexema
PT_V	Ponto e vírgula	;	Token, lexema
ERRO	Qualquer coisa diferente de qualquer símbolo token e palavra-chave definida.		Token, descrição do erro, linha e coluna onde o erro ocorreu.

Tabela 2 – Palavras-chave da linguagem ALG 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

senao	Elemento de estrutura condicional
fimse	Elemento de estrutura condicional
fim	Delimita o fim do programa
Inteiro	Tipo de dado
literal	Tipo de dado
real	Tipo de dado

### 3. Programa fonte a ser lido

O analisador léxico deverá ler o programa fonte a ser disponibilizado em FONTE.ALG e imprimir na tela o reconhecimento realizado para cada sequência de símbolos reconhecida: token, lexema e tipo e palavras reservadas da linguagem. O FONTE.ALG deverá ter o conteúdo apresentado na Figura 1.

Figura 1 – Programa fonte a ser lido pelo analisador léxico.

```

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;

  escreva "\nB=\n";
  escreva D;
  escreva "\n";
  escreva C;
  escreva "\n";
  escreva A;

fim

```

### 4. Produto Final

Ao final de todos os três trabalhos práticos da disciplina nos quais serão aplicadas as técnicas adquiridas em sala (fases de análise e síntese) teremos como sistema e resultado do estudo de caso, um pequeno compilador que, utilizando dos tokens reconhecidos (Tabela 1), as palavras da linguagem definidas na Tabela 2 e das demais fases de análise e síntese a serem implementadas posteriormente, compilará o programa fonte em linguagem Mgol: FONTE.ALG em PROGRAMA.C da figura 2.

Figura 2 – Programa objeto a ser gerado pelo compilador ao final de todos os trabalhos da disciplina (PROGRAMA.C).

```
#include<stdio.h>

typedef char literal[256];
void main(void)
{
    /*----Variaveis temporarias----*/
    int T0;
    int T1;
    int T2;
    int T3;
    int T4;
    /*-----*/
    literal A;
    int B;
    int D;
    double C;

    printf("Digite B");
    scanf("%d",&B);
    printf("Digite A:");
    scanf("%s",A);
    T0=B>2;
    if(T0)
    {
        T1=B<=4;
        if(T1)
        {
            printf("B esta entre 2 e 4");
        }
    }
    T2=B+1;
    B=T2;
    T3=B+2;
    B=T3;
    T4=B+3;
    B=T4;
    D=B;
    C=5.0;
    printf("\nB=\n");
    printf("%d",D);
    printf("\n");
    printf("%lf",C);
    printf("\n");
    printf("%s",A);
}
```