

Prof. Dr. Diego Raphael Amancio  
diego@icmc.usp.br

### Trabalho 1 - Analisador léxico

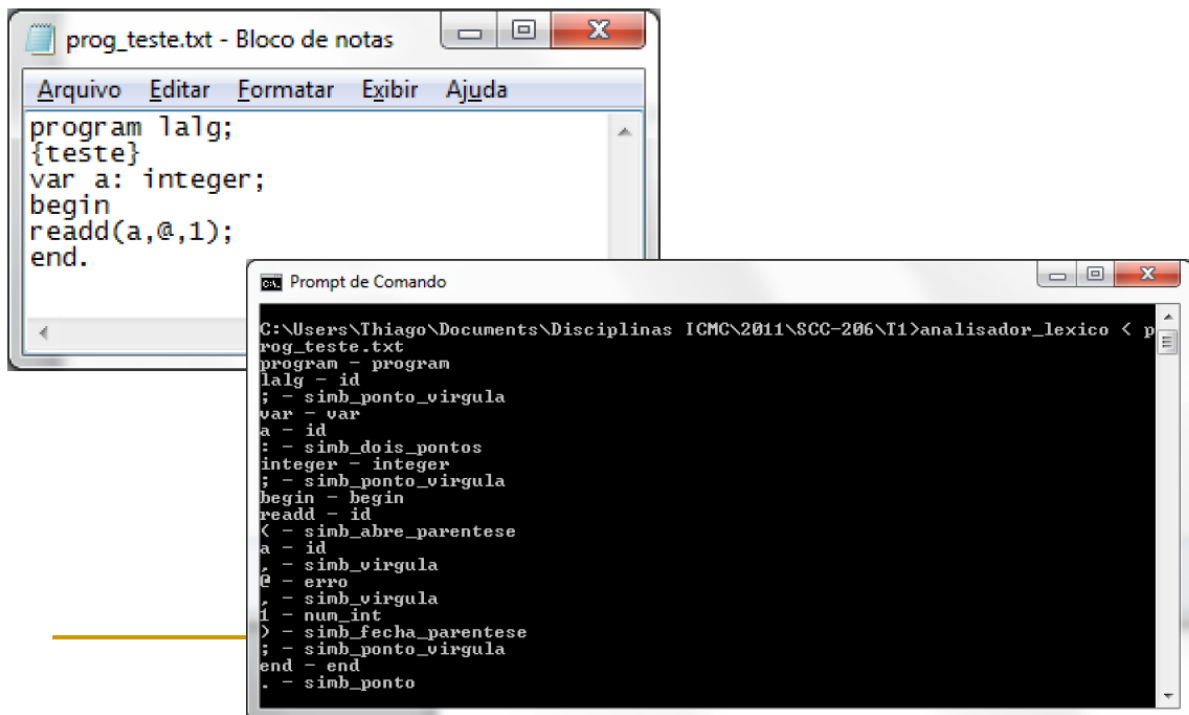
**Especificação:** desenvolver o analisador léxico para a linguagem LALG, com tratamento de erro. Como exemplo, considere as seguintes entradas e saídas:

<u>Entrada: programa-fonte em LALG</u>	<u>Saída (na tela e em arquivo txt)</u>
<pre> program lalg; {entrada} var a: integer; begin readd(a, @, 1); end. </pre>	<pre> program - program lalg - id ;-; var - var a - id ;-; integer - integer ;-; begin - begin readd - id ( - ( a - id ;-; @ - erro ;-; 1 - num ) - ) ;-; end - end ;-; </pre>

As seguintes tarefas devem ser desenvolvidas neste trabalho prático:

1. Modelar a tarefa do analisador léxico: tokens possíveis, expressões regulares utilizadas, formas de tratamento de erros (ver slides das aulas).
2. Buscar e estudar o lex/flex ou JavaCC: note que quase todos os livros de compiladores têm apresentações dessas ferramentas; também há muitos tutoriais na Web (alguns estão disponíveis no site da disciplina). **O grupo pode decidir por implementar o analisador de maneira manual, sem o uso das ferramentas mencionadas.**
3. Gerar o analisador léxico usando o lex/flex ou javaCC: o grupo deve incorporar no lex/flex/javaCC a geração de uma função principal que analise todo o arquivo de entrada, chamando o analisador léxico várias vezes, o qual, a cada chamada, deve retornar um único par <cadeia,token>. Note que esta função será substituída posteriormente pelo analisador sintático.

Seguem mais alguns exemplos:

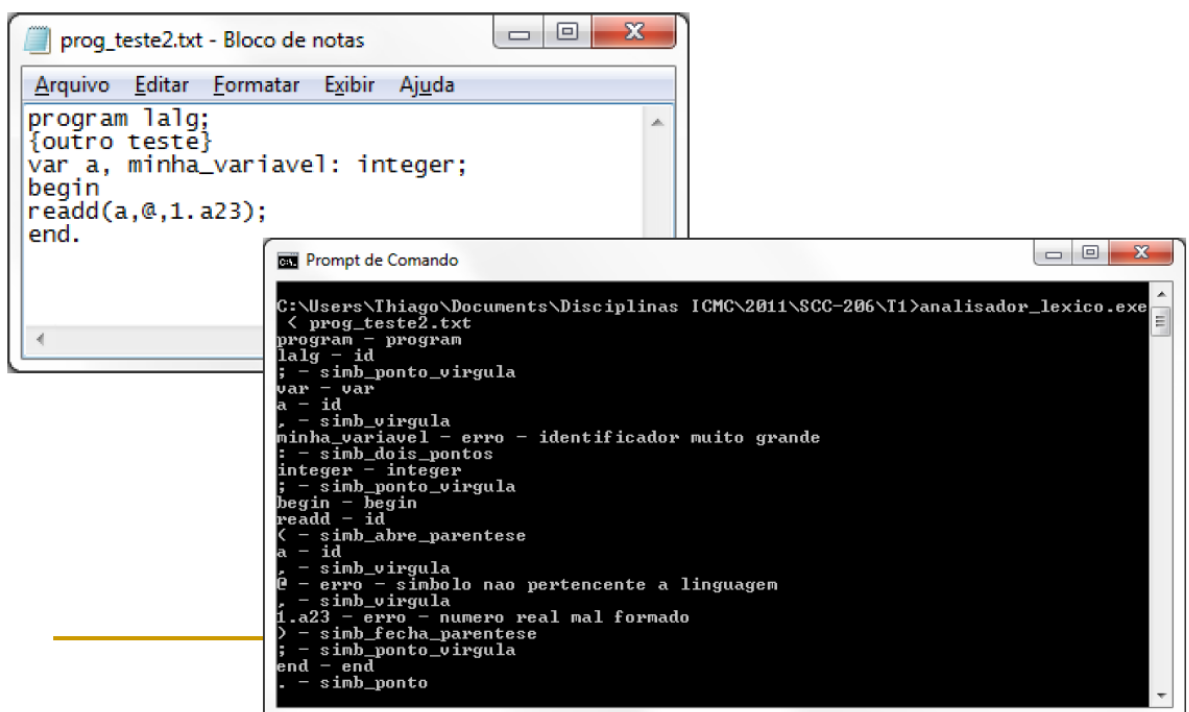


The screenshot shows a Notepad window titled "prog\_teste.txt - Bloco de notas" with the following code:

```
program lalg;  
{teste}  
var a: integer;  
begin  
  readd(a,@,1);  
end.
```

Below it is a Command Prompt window titled "Prompt de Comando" showing the output of the lexical analyzer for the same code:

```
C:\Users\Thiago\Documents\Disciplinas ICMC\2011\SCG-206\T1>analizador_lexico < prog_teste.txt  
program - program  
lalg - id  
; - simb_ponto_virgula  
var - var  
a - id  
: - simb_dois_pontos  
integer - integer  
; - simb_ponto_virgula  
begin - begin  
readd - id  
< - simb_abre_parentese  
a - id  
, - simb_virgula  
@ - erro  
; - simb_virgula  
1 - num_int  
> - simb_fecha_parentese  
; - simb_ponto_virgula  
end - end  
. - simb_ponto
```



The screenshot shows a Notepad window titled "prog\_teste2.txt - Bloco de notas" with the following code:

```
program lalg;  
{outro teste}  
var a, minha_variavel: integer;  
begin  
  readd(a,@,1.a23);  
end.
```

Below it is a Command Prompt window titled "Prompt de Comando" showing the output of the lexical analyzer for the same code, including error messages:

```
C:\Users\Thiago\Documents\Disciplinas ICMC\2011\SCG-206\T1>analizador_lexico.exe  
< prog_teste2.txt  
program - program  
lalg - id  
; - simb_ponto_virgula  
var - var  
a - id  
, - simb_virgula  
minha_variavel - erro - identificador muito grande  
: - simb_dois_pontos  
integer - integer  
; - simb_ponto_virgula  
begin - begin  
readd - id  
< - simb_abre_parentese  
a - id  
, - simb_virgula  
@ - erro - simbolo nao pertencente a linguagem  
; - simb_virgula  
1.a23 - erro - numero real mal fornado  
> - simb_fecha_parentese  
; - simb_ponto_virgula  
end - end  
. - simb_ponto
```

O grupo deve tomar as seguintes decisões de projeto:

1. <palavra\_reservada,palavra\_reservada> ou <palavra\_reservada,simb\_palavra\_reservada>.  
Para facilitar o entendimento, não utilize códigos numéricos para os tokens.

2. Implementação da tabela de palavras reservadas: escolha da estrutura de dados e da função de busca. Note que a busca deve ser eficiente.

3. Como lidar com erros? Erros genéricos ou mais específicos?

**Entrega:** submissão de arquivo zip/rar no Tidia (um membro do grupo deve submeter no escaninho) até o dia 28/4/2019.

### **O que entregar?**

- Especificação/listagem do analisador léxico na linguagem lex/flex/javacc;
- Código fonte produzido e executável;
- Relatório sucinto informando os membros do grupo (número USP), decisões de projeto e justificativas, descrição da especificação do analisador léxico na linguagem lex/flex/javacc, passo a passo para compilar o analisador léxico e executá-lo além de um ou mais exemplos de execução.

**Prazo de entrega:** 28/4/2019 até meia noite. A cada dia de atraso, um ponto a menos. Se cópia identificada, zero para todos os grupos envolvidos.

Itens a serem avaliados:

**10% da nota:** Clareza e completude do relatório pedido, especificação criado em lex ...

**80% da nota:** Análise léxica em si, com tratamento de erros. A avaliação será realizada com base em casos de teste.

**10% da nota:** questões de implementação que incluem acesso à tabela de palavras reservadas, presença de programa principal executando o analisador léxico várias vezes, tratamento de comentários, etc.

Dica: desenvolvam o trabalho com calma e atenção, aprimorando a especificação do lex.. e avaliando os impactos na análise léxica de casos reais.