



UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

SCC0217 Linguagens de programação e compiladores

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:

Entrada (programa fonte em LALG):

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

Saída (na tela e em arquivo txt)

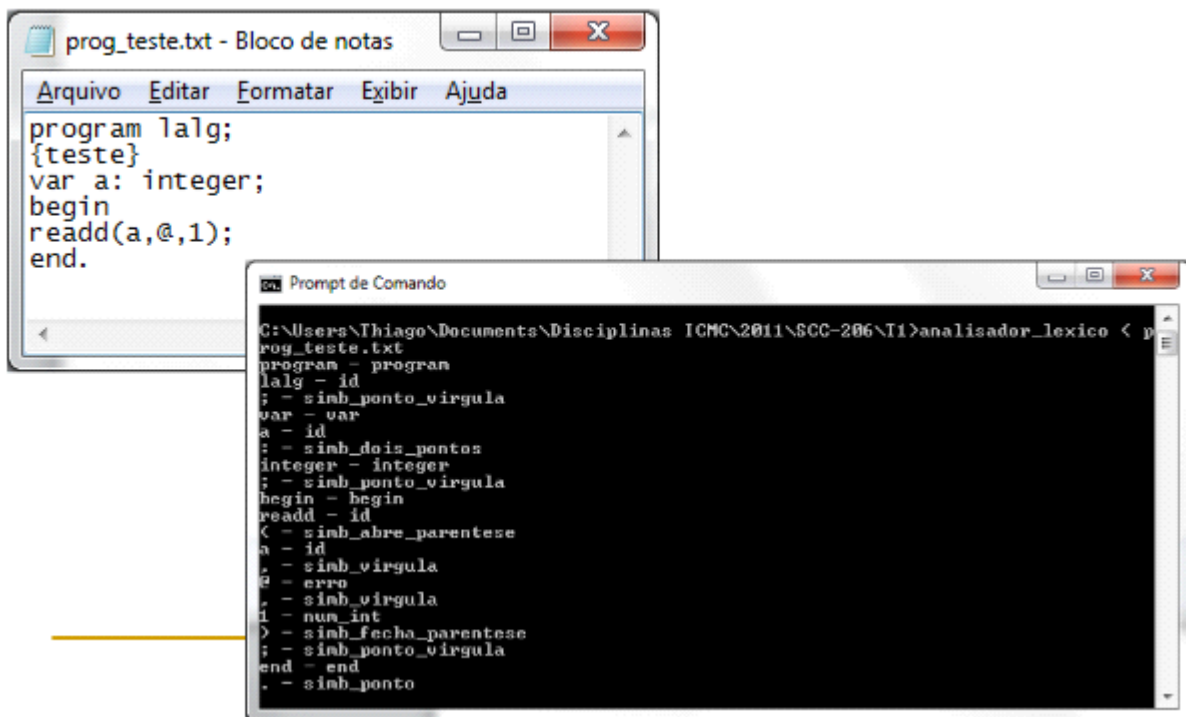
```
program - programação
lalg - id
; - ;
var - var
a - id
: - :
integer - integer
; - ;
begin - begin
read - id
( - (
a - id
, - ,
@ - erro
, - ,
1 - num
) - )
```

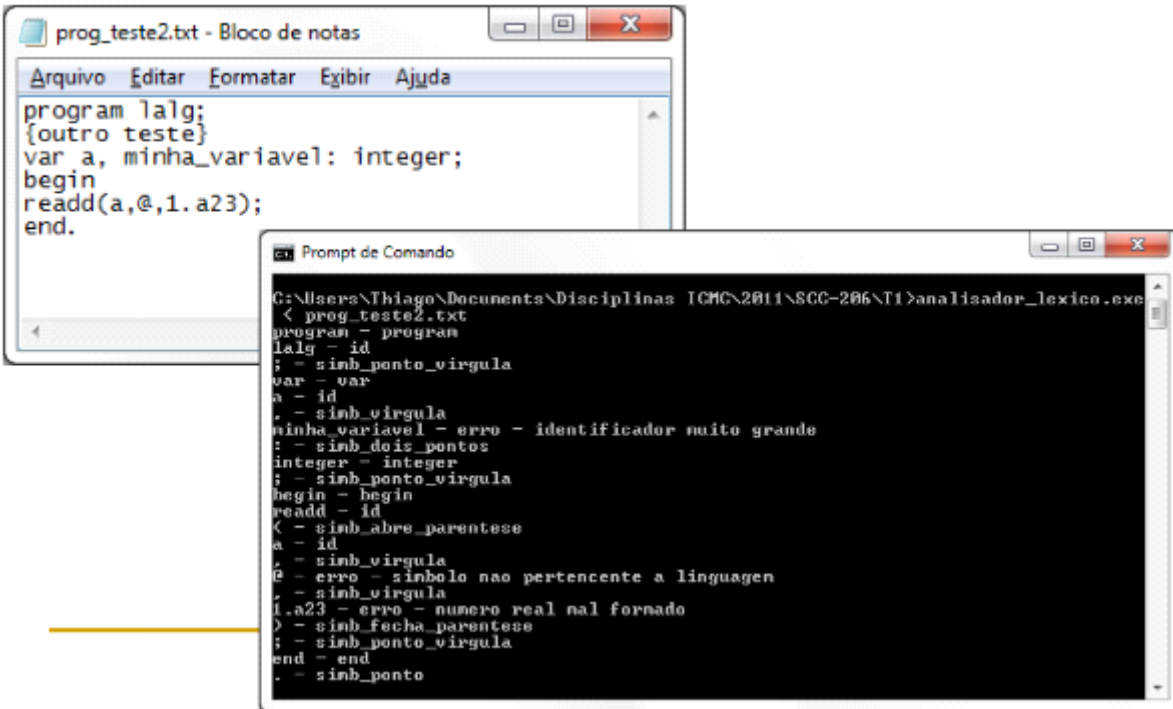
```
--;  
end - end  
.-.
```

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, JavaCC ou outro: 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). Opcionalmente, há a opção de desenvolver o trabalho sem o uso dessas ferramentas.
3. Gerar o analisador léxico: o grupo deve incorporar 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.

Alguns exemplos:





```
prog_teste2.txt - Bloco de notas
Arquivo Editar Formatar Exibir Ajuda
program lalg;
{outro teste}
var a, minha_variavel: integer;
begin
  readd(a,@,1. a23);
end.
```

```
Prompt de Comando
C:\Users\Thiago\Documents\Disciplinas ICMC\2011\SCC-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 nal 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 run.codes até o dia 30/4/2017 (até 23h59).

O que entregar?

- Especificação/listagem do analisador léxico na linguagem lex/flex/javacc/etc...;
- 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/etc..., passo a passo para compilar o analisador léxico e executá-lo além de um ou mais exemplos de execução.

Prazo de entrega: 30/4/2017 até antes da 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.

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.