

---

# Trabalho 1: analisador léxico para LALG

- Especificação
  - Desenvolver o analisador léxico para a linguagem LALG, com tratamento de erro



# Trabalho 1: analisador léxico para LALG

## ■ Exemplo

Entrada: programa-fonte  
em LALG

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

Saída (na tela e em  
arquivo txt)

```
program - program  
lalg - id  
; - ;  
var - var  
a - id  
: - :  
integer - integer  
: - :  
begin - begin  
readd - id  
( - (  
a - id  
, - ,  
@ - erro  
, - ,  
1 - num  
) - )  
: - :  
end - end  
. - .
```

---

# Trabalho 1: analisador léxico para LALG

## ■ Especificação

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 o JavaCC
    - Atenção: quase todos os livros de Compiladores têm apresentações dessas ferramentas; também há muitos tutoriais na web
      - Faz parte do trabalho aprender usar esse ferramental
  3. Gerar o analisador léxico usando o Lex/Flex ou o JavaCC
    - 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>
      - Essa função será substituída posteriormente pelo analisador sintático
-

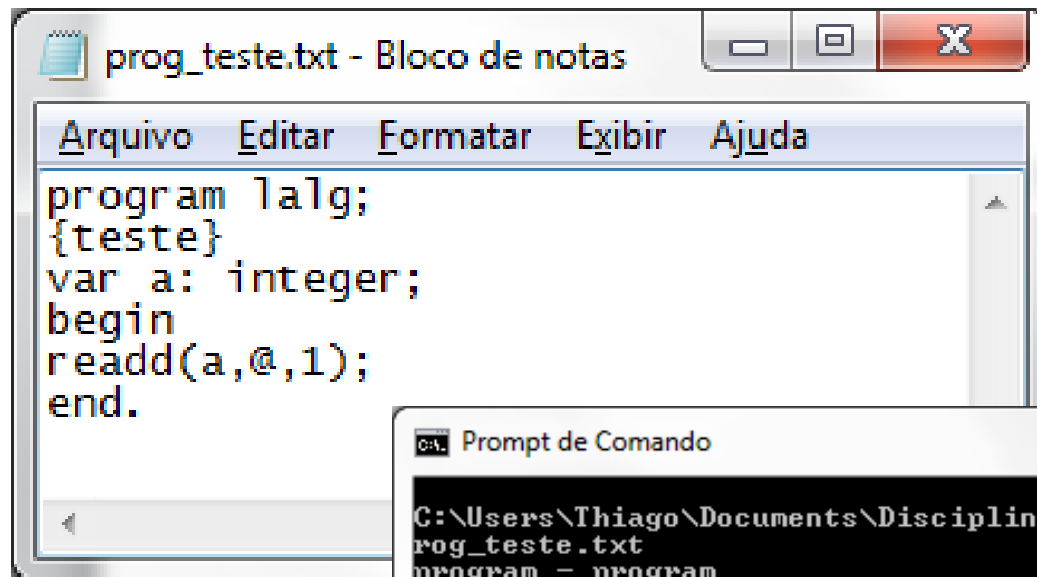
---

# Trabalho 1: analisador léxico para LALG

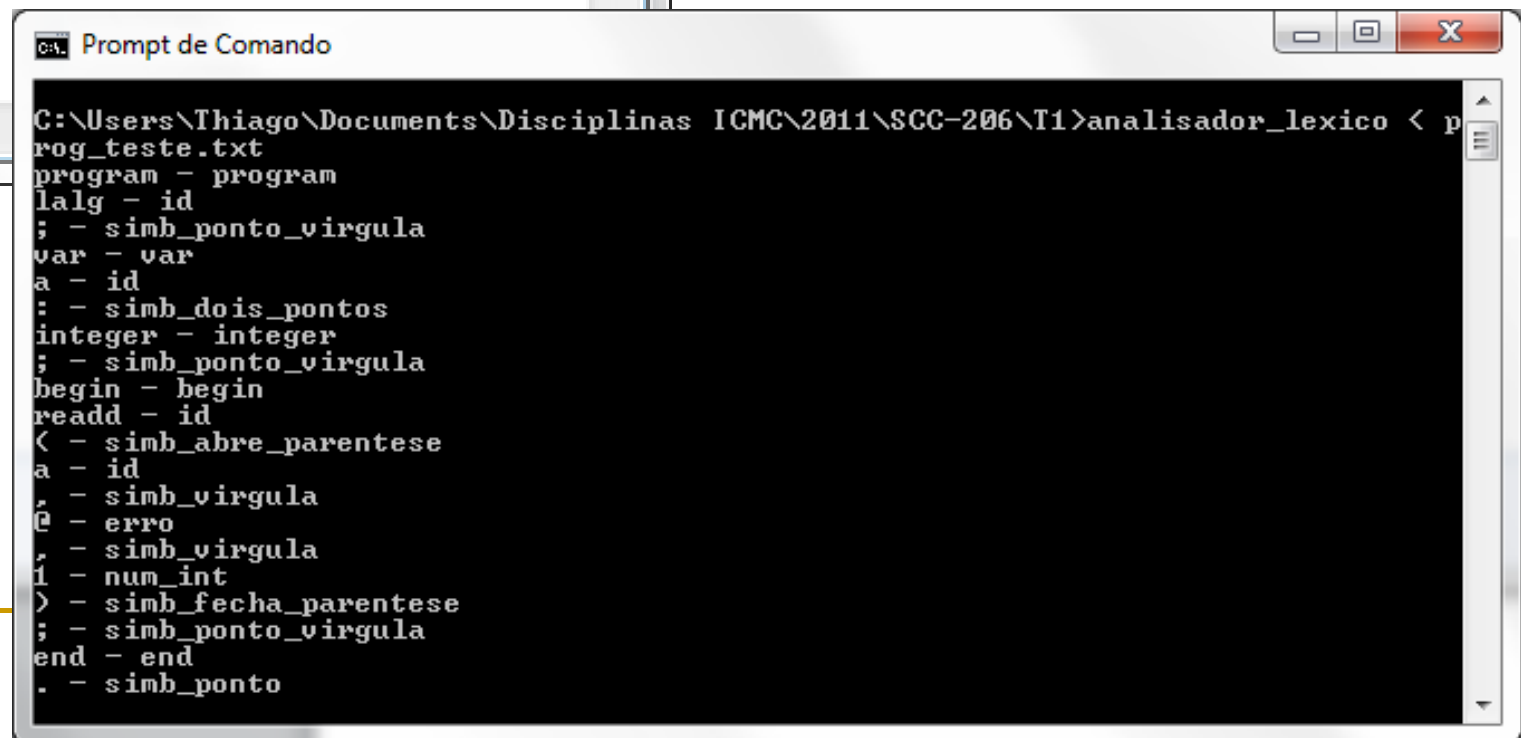
- Decisões de projeto e de implementação
    - Quais os tokens para as cadeias
      - <palavra\_reservada, palavra\_reservada> ou <palavra\_reservada, simb\_palavra\_reservada>?
        - Não usem códigos numéricos para os tokens!
    - Como implementar a tabela de palavras reservadas
      - Estrutura de dados, busca
      - Lembrem-se: busca deve ser eficiente
    - Como lidar com os erros
      - Erros genéricos? Erros mais específicos?
-

# Trabalho 1: analisador léxico para LALG

- Exemplo: tratamento de erros com mensagens genéricas



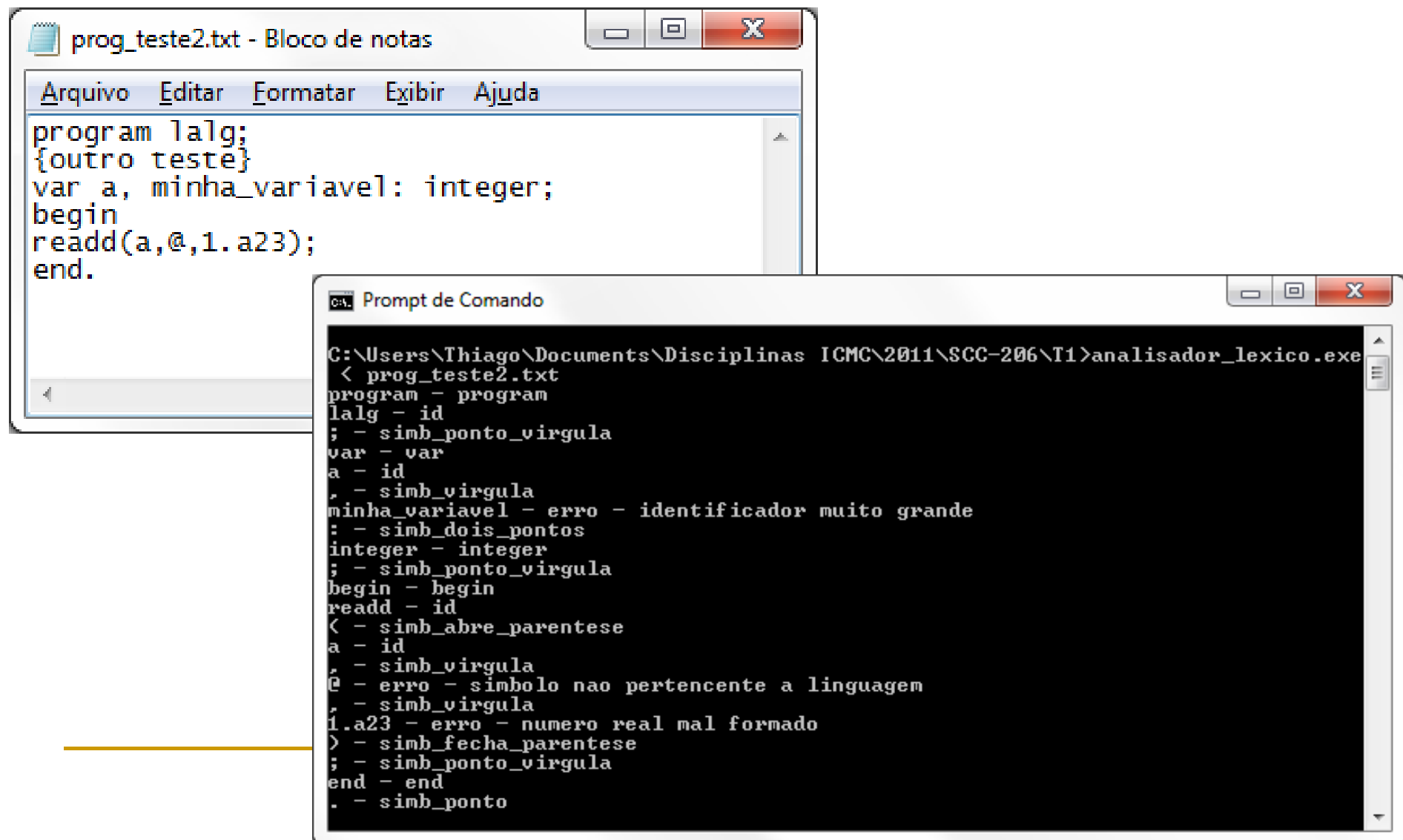
```
program lalg;  
{teste}  
var a: integer;  
begin  
  readd(a,@,1);  
end.
```



```
C:\Users\Thiago\Documents\Disciplinas ICMC\2011\SCC-206\T1>analisador_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
```

# Trabalho 1: analisador léxico para LALG

- Outro exemplo: tratamento de erros com mensagens mais específicas



The image shows two windows. The top window is a Notepad application titled 'prog\_teste2.txt - Bloco de notas'. It contains the following code:

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

The bottom window is a Command Prompt titled 'Prompt de Comando'. It shows the execution of the 'analisador\_lexico.exe' program, which has been run on 'prog\_teste2.txt'. The output lists the tokens and error messages generated by the lexical analyzer:

```
C:\Users\Thiago\Documents\Disciplinas ICMC\2011\SCC-206\T1>analisador_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 formado  
> - simb_fecha_parentese  
; - simb_ponto_virgula  
end - end  
. - simb_ponto
```

---

# Trabalho 1: analisador léxico para LALG

- Grupos de 3 alunos, no máximo
  - Entrega
    - Submissão de arquivo zip/rar em “Atividade” no Tidia
      - Especificação/listagem do analisador léxico na linguagem do Lex/Flex/JavaCC
      - Código-fonte produzido
      - Executável (para evitar problemas, alterem o nome, se necessário: de **arq.exe** para **arq.ex**, por exemplo)
      - Relatório *sucinto* e *objetivo* (5 páginas, aproximadamente) contendo
        - Identificação dos membros do grupo (nome e número USP)
        - Decisões de projeto, justificativas
        - Descrição da especificação do analisador léxico na linguagem do Lex/Flex/JavaCC
        - Passo a passo para compilar o analisador léxico e executá-lo
        - Exemplo de execução
    - Data de entrega: **08/04** até meia noite
      - A cada dia de atraso, 1 ponto a menos
  - Se cópia de trabalhos detectada: zero para todos os grupos envolvidos!
-

---

# Trabalho 1: analisador léxico para LALG

- Itens a serem avaliados
    - Clareza e completude do relatório pedido, especificação criada em Lex/Flex/JavaCC (~10% da nota)
    - Análise léxica em si, com tratamento de erros (~80% da nota, pelo menos)
      - Avaliação com base em casos de teste
        - Dica: criem vários casos de teste e testem seus programas
    - Questões de implementação (~10% da nota)
      - Acesso à tabela de palavras reservadas (sugere-se o uso de *hashing*)
      - Presença do programa principal executando o analisador léxico várias vezes
      - Tratamento de comentários
      - Etc.
  - Dica: desenvolvam o trabalho com calma e atenção, aprimorando a especificação no Lex/Flex/JavaCC e avaliando os impactos na análise léxica de casos reais
-