IV.2 – Aspectos Léxicos Convencionais

- **Classes de símbolos**
  - **Genéricos**
    - Token genérico / Lei de formação bem definida
    - Podem possuir limitações de tamanho e/ou valor
    - Possuem valor semântico – o token deve ser acompanhado por uma informação adicional: a representação/valor do símbolo reconhecido (lexeme)
    - Exemplos:
      - **Constantes**
        - **numéricas (inteira, real, ...)**
        - **literais (cadeias / sequências de caracteres)**
      - **Identificadores**
        - **Nome de variáveis, constantes, métodos, etc...**
  - **Específicos**
    - Token específico para cada símbolo
    - não possuem valor semântico
    - Exemplos:
      - **Símbolos especiais**
        - **simples, duplos, triplos , ...**
        - **pontuação, operadores, separadores ...**
      - **palavras reservadas**
        - **casos especiais de identificadores**
  - **Sem valor sintático**
    - Não constituem token
    - Devem ser reconhecidos, porém ignorados
    - Exemplos:
      - **Comentários**
      - **Espaços em branco**
      - **Caracteres de controle**

IV.3 – Especificação dos aspectos Léxicos

- **Informal – descrição textual**
- **Formal – Usando GR, A.F. e/ou E.R**

Escolha do Mecanismo de Especificação

- **Depende da forma de implementação do A. Léxico**
  - **Quando o AL é gerado automaticamente**
    - **O gerador usado determina o mecanismo de especificação (geralmente E.R. ou variações)**
    - **Exemplos de Geradores:**
      - **LEX, GALS, Micro-LEX, GALEX**
  - **Quando o AL é programado manualmente**
    - **Qualquer mecanismo**
    - **Especificação usada como documentação**
    - **Normalmente usa-se:**
      - **E.R. para notação**
      - **A.F. como base para reconhecimento**
      - **Descrição Informal (textual) para exceções**

IV.4 – Analisador Léxico

IV.4.1 - Funções

- Ler o programa fonte
- Agrupar caracteres em itens léxicos (tokens)
  - **Identificadores / Palavras Reservadas**
  - **Constantes (numéricas e literais)**
  - **Símbolos especiais (pontuação, operadores, etc)**
- Ignorar "brancos", comentários e car. de controle
- Detectar e diagnosticar erros léxicos
  - **Símbolos (caracteres) inválidos**
  - **Identificadores, constantes, literais etc. mal formados**
  - **Tamanho/valor inválido de constantes**
- Exemplo:

Programa Fonte	Tokens Reconhecidos	
program exemplo;	program	1 - PR
	exemplo	2 - ID
	;	3 - SEsp
var a, b : integer;	var	4 - PR
	a	2 - ID
	,	5 - SEsp
...	...	...
begin	begin	6 - PR
(* Inicio do programa *)		
b := 2.5 + a;	b	2 –ID
escreva ('tot = ', B);	:=	7 – SEsp duplo
...	2.5	8 – Cte Real
	...	...
end.	end	9 – PR
	.	10 - SEsp

IV.4.2 - Estratégias de Implementação do AL

- **Procedimento Independente**
  - Analisa o programa fonte inteiro
  - Gera arquivo (lista) de TOKENS
  - Constitui um PASSO do processo de Compilação

- **Função do Analisador Sintático**
  - A cada ativação, reconhece um símbolo e retorna o token correspondente (juntamente com sua representação literal e sua localização dentro do programa fonte)

IV.4.3 – Especificação Léxica da LSI-13/2

Identificadores

- **caracteres válidos - letras, digitos e os caracteres especiais: "@", "#" e "_"**
- **regras de formação:**
  - **começa com letra ou com "@"**
  - **não pode terminar com "@", "#" e "_"**
  - **não possui caracteres especiais consecutivos**
  - **não possui limite de tamanho**

Palavras reservadas

casos especiais de identificadores

- programa, var, caracter, cadeia, procedimento, inicio, fim, inteiro, booleano, funcao, se, entao, senao, leia, escreva, ou, e, nao, falso, verdadeiro, de, faca, real, vetor, enquanto

OBS.: Esta relação deverá ser atualizada quando da especificação sintática da LSI-13/2

Constantes numéricas

- Inteiras e reais sem sinal (com ou sem parte exponencial)
- Parte exponencial composta por: "E" ou "e", sinal opcional ("+" ou "-") e pelo menos 1 dígito
- Tokens distintos para constantes inteiras e reais
- Constantes inteiras – números sem ponto decimal
- Constantes reais – números com ponto decimal (no início, no fim ou no meio)

Constantes literais

- Usar ‘ (caracter aspa) como delimitador
- Sem limite de tamanho
- No meio de um string, o caracter ‘ (aspa) deve ser representado por duas aspas simples justapostas
  Ex. ‘pato d’’agua’
- Permitir continuação em outra linha
- Literal não fechado – erro léxico
- Literais podem conter quaisquer caracteres (mesmo os caracteres inválidos para outros fins)

Comentários de linha

- **Começa com “//”**
- **Termina no final da linha**

Comentários de bloco

- **notação: qualquer sequência de caracteres entre os delimitadores /* e */**
- **não analisar sequências de caracteres internas**
- **sem limite de tamanho**
- **comentário não fechado = erro léxico**

Símbolos especiais

(lista a ser atualizada posteriormente)

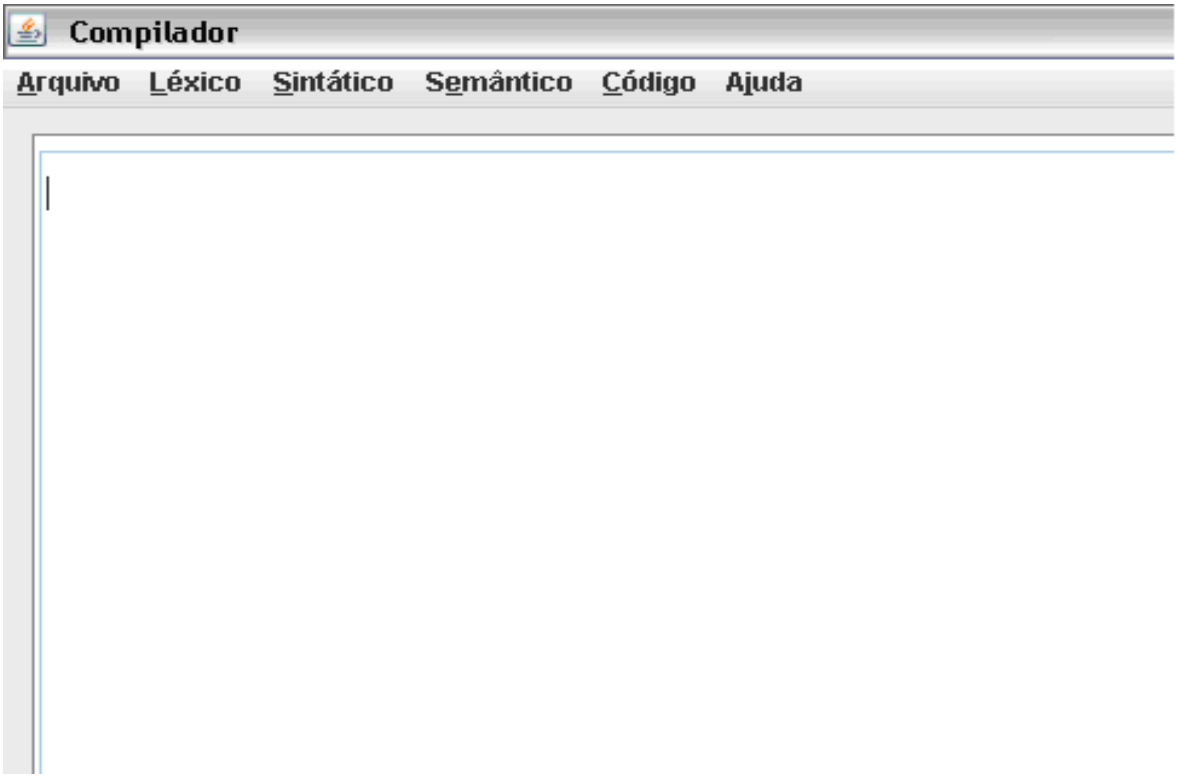
- **Token específico para cada símbolo**
  - **Simples: ; , . > < = ( ) [ ] + - * / :**
  - **Duplos: := .. <> <= >=**

IV.4.4 – Implementação do Anal. Léxico

- **Especificação Formal**
  - Através de E.R.
- **Implementação**
  - uso de um Gerador Automático (GALS)
  - interface gráfica (para integrar as várias fases)
- **Estratégia – Livre escolha**
  - procedimento independente
  - função do Analisador Sintático
- **Linguagem para implementação**
  - livre escolha
    - GALS gera Java, C++ e Delphi
- **Equipes : 2 alunos**
- **Prazo de entrega**
  - junto com o analisador sintático (a ser definido)
- **Leitura complementar**
  - capítulo sobre análise léxica de, pelo menos, um dos livros indicados na bibliografia da disciplina.

IV.4.5 – Implementação do Compilador

- Criar um Módulo de Integração, usando Interface Gráfica, conforme exemplo abaixo:



- **Observações gerais:**
  - **Permitir a edição, leitura e gravação de programas (permitir qualquer extensao)**
  - **Permitir a ativação independente de cada analisador**
  - **Fornecer mensagens de erro em uma janela separada**
  - **Posicionar o cursor na posição do token que causou o erro**
  - **Substituir as mensagens de erro padrão do gerador por mensagens personalizadas adequadas**
  - **Documentar o programa fonte adequadamente**
  - **Registrar data/autor em todos os módulos/classes**

LEX – Gerador de Analisadores Léxicos
(Lesk & Schmith, 1975)

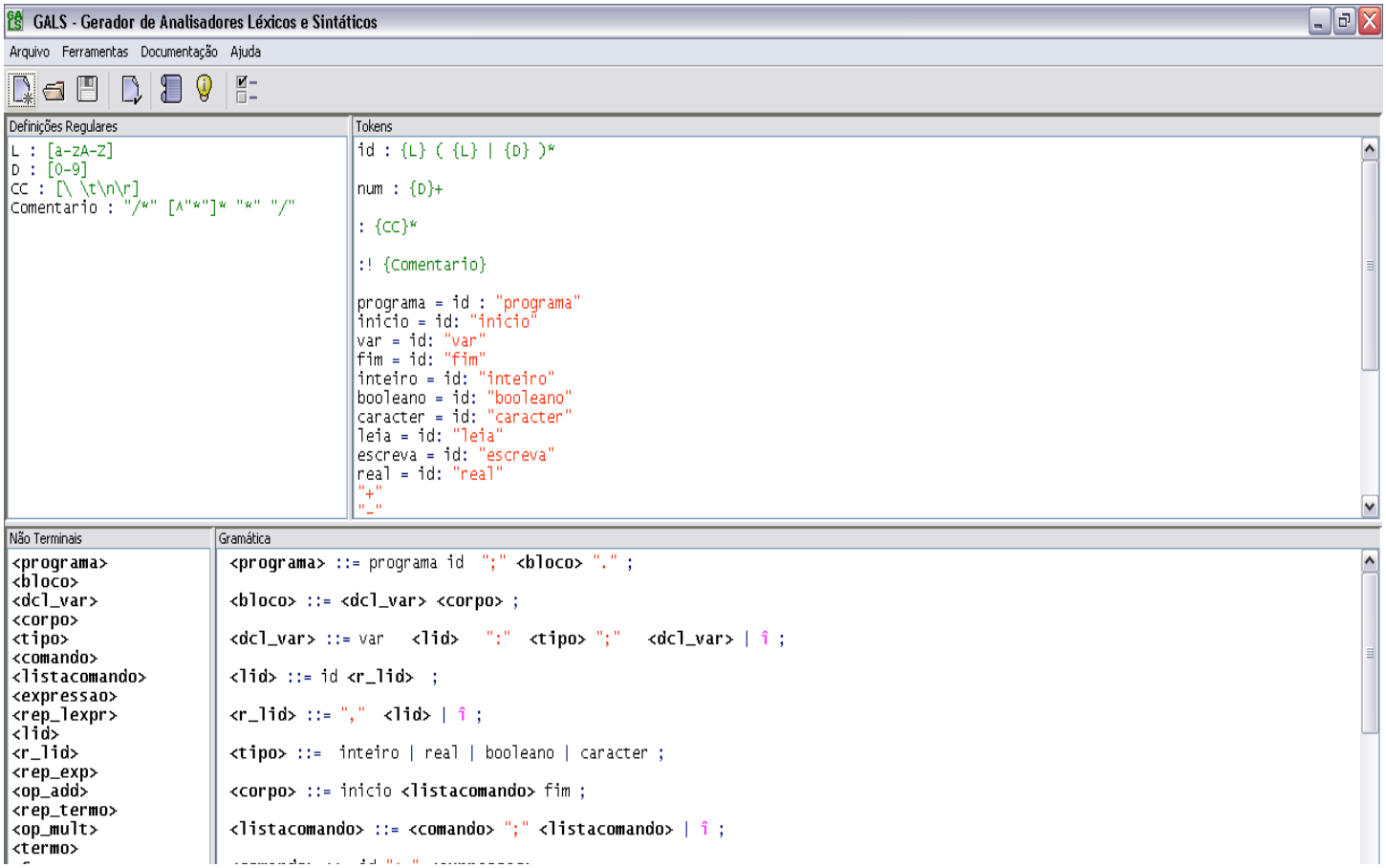
- **Linguagem de especificação:**

  Definições
  %%
  Regras de tradução
  %%
  Procedimento auxiliares

- **Exemplo de uma especificação:**

```
letra A | B | C | ... | Z   ou [A-Z]

digito      0 | 1 | 2 | ... | 9    ou [0-9]

%%

=       return (1)

:=      return (2)

(       return (3)

        : : :

letra {letra | digito}  {Se epal-reservada (id, token)
                         então return (token)
                         senao insere-ts(id); return(20)});

dígito {digito}   {insere-tc (num); return (21)}
%%
epal-reservada (...)
insere-ts (...)
insere-tc (...)
```

GALS


GALS - Gerador de Analisadores Léxicos e Sintáticos
Arquivo Ferramentas Documentação Ajuda
Definições Regulares
L : [a-zA-Z]
D : [0-9]
CC : [\ \t\n\r]
Comentario : "/*" [^"*"]* "*" "/"
Tokens
id : {L} ({L} | {D})*
num : {D}+
: {CC}*
:! {Comentario}
programa = id : "programa"
inicio = id: "inicio"
var = id: "var"
fim = id: "fim"
inteiro = id: "inteiro"
booleano = id: "booleano"
caracter = id: "caracter"
leia = id: "leia"
escreva = id: "escreva"
real = id: "real"
"+"
"-"
Não Terminais
<programa>
<bloco>
<dcl_var>
<corpo>
<tipo>
<comando>
<listacomando>
<expressao>
<rep_lexpr>
<lid>
<r_lid>
<rep_exp>
<op_add>
<rep_termo>
<op_mult>
<termo>
Gramática
<programa> ::= programa id ";" <bloco> "." ;
<bloco> ::= <dcl_var> <corpo> ;
<dcl_var> ::= var <lid> ":" <tipo> ";" <dcl_var> | î ;
<lid> ::= id <r_lid> ;
<r_lid> ::= "," <lid> | î ;
<tipo> ::= inteiro | real | booleano | caracter ;
<corpo> ::= inicio <listacomando> fim ;
<listacomando> ::= <comando> ";" <listacomando> | î ;