Ferramenta para Apoio ao Ensino de Introdução à Programação

Acadêmica:
Karly Schubert Vargas
Orientadora:
Joyce Martins

Roteiro

- Introdução
  - Objetivos
- Ensino no computador
- Linguagens de programação
- Compiladores
- Trabalhos correlatos
- Requisitos
- Especificação da ferramenta e da linguagem
- Implementação e Operacionalidade da ferramenta
- Critérios de qualidade
- Resultados
- Conclusão
  - Extensões

Introdução

- Ensino de programação
  - Despertar a criatividade
  - Desenvolver a lógica do aluno
- Representação do algoritmo
  - Fluxograma
  - Portugol
- Uso do papel

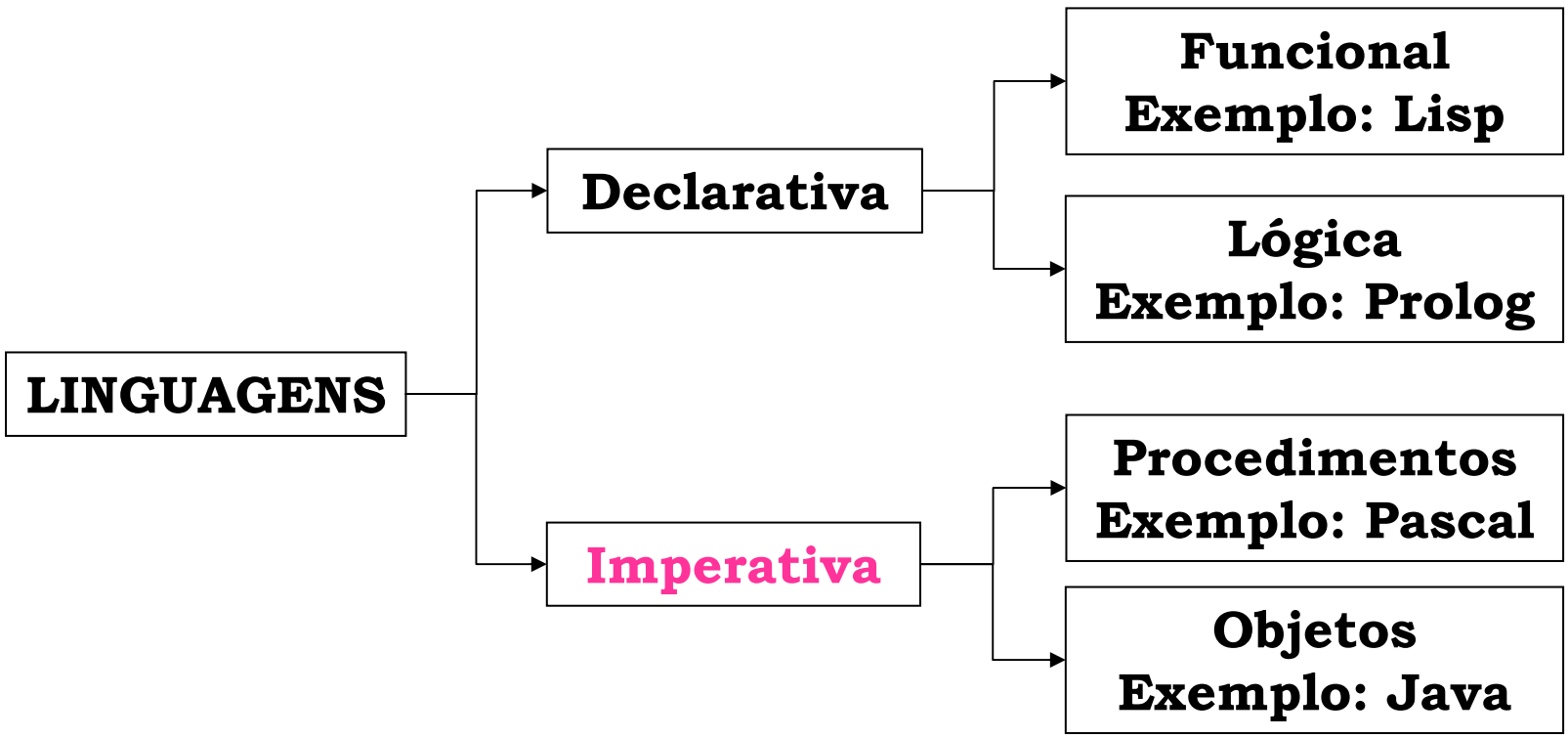
Objetivos

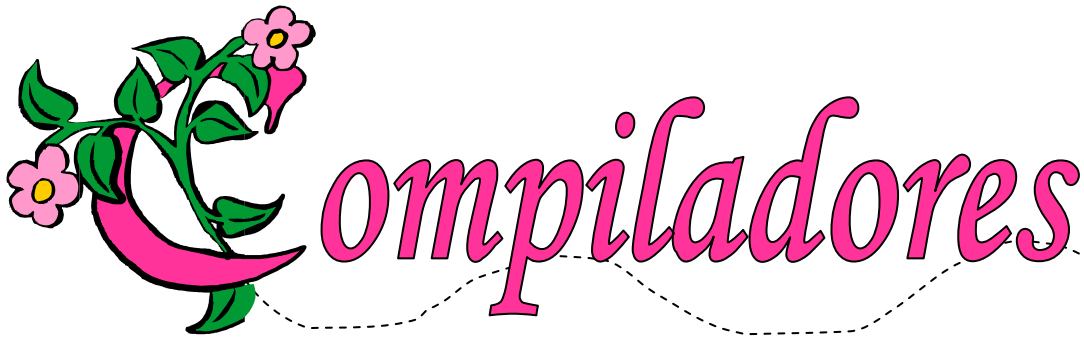
- **Disponibilizar analisadores léxico, sintático e semântico;**
- **Efetuar detecção/tratamento de erros, emitindo mensagens capazes de auxiliar a correção dos algoritmos elaborados;**
- **Possibilitar a execução dos algoritmos passo a passo, com opção para visualizar os valores das variáveis declaradas;**
- **Utilizar os padrões de qualidade para o desenvolvimento de softwares educacionais.**

Ensino no computador

- **Ensino assistido por computador**
  - **Tutorial**
- **Aprendizado socialmente distribuído**
  - **Internet**
- **Ambientes interativos de aprendizagem**
  - **Modelagem/Simulação**
  - **Ambientes de programação**
    - **Prolog, Logo e Pascal**

Linguagens de Programação





Programa fonte

↓

Análise léxica	**Definições regulares**	`letra: [a-z]` `identificador:(letra)+`
Análise sintática	**BNF**	`<item> → identificador` `| numero`
Análise semântica	**Esquemas de tradução**	`<item> → identificador #1 | numero #2`
Geração de código	**Notação polonesa**	`(a+b)*c → ab+c*`

↓

Programa objeto

Trabalhos correlatos

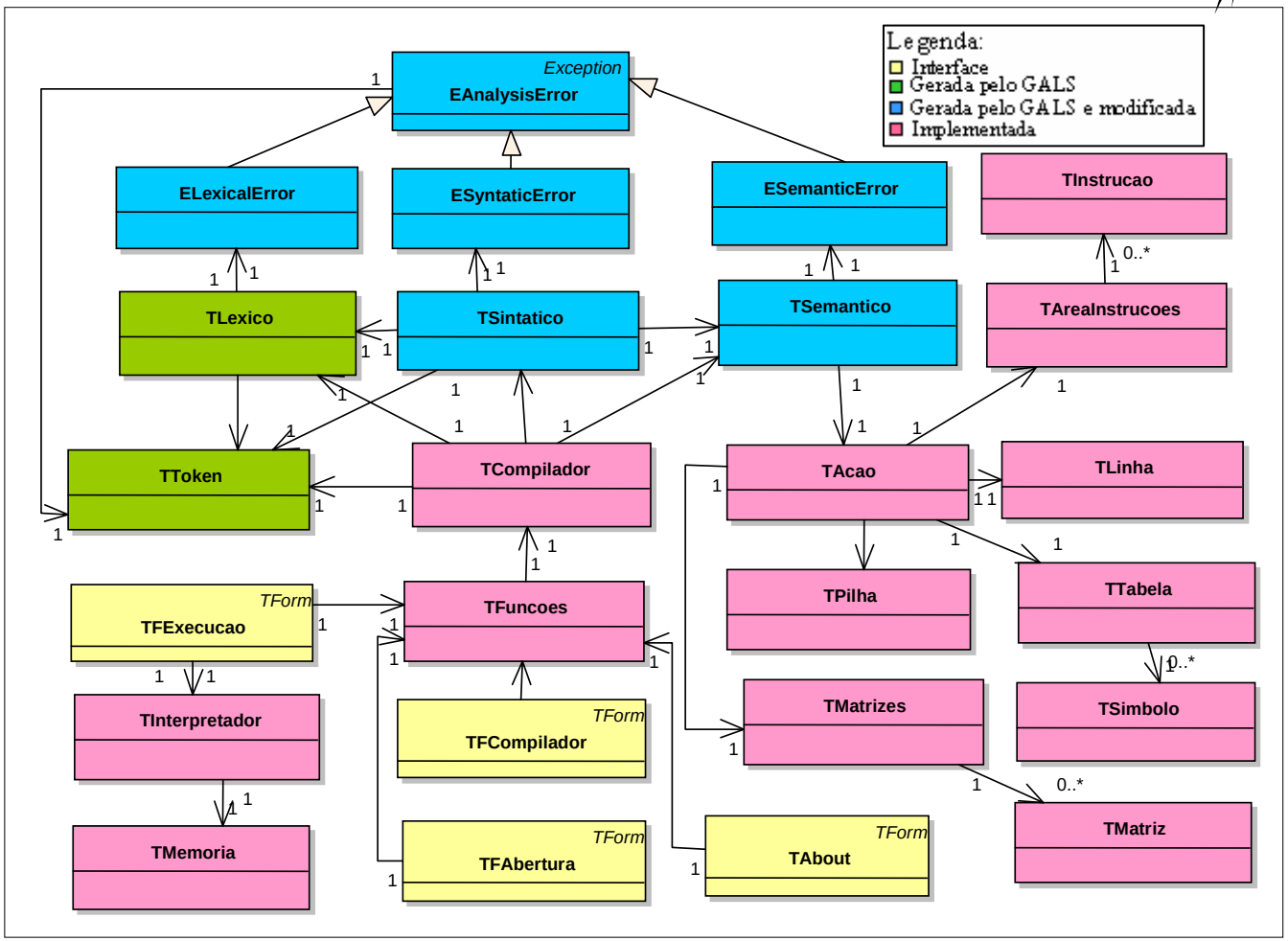
- **Ambiente de Apoio ao Aprendizado de Programação (AMBAP)**
- **Ambiente de Simulação e Animação de Algoritmos (ASA)**
- **Software para o auxílio ao aprendizado de algoritmos**
- **Ferramenta de apoio ao ensino de algoritmos (CIFluxProg)**

Requisitos

- **Editor de texto**
- **Possuir uma interface seguindo padrões de qualidade**
- **Compilar algoritmos em português e estruturados**
- **Informar erros encontrados**
- **Gerar código intermediário**
- **Executar o código intermediário, com opção passo a passo**

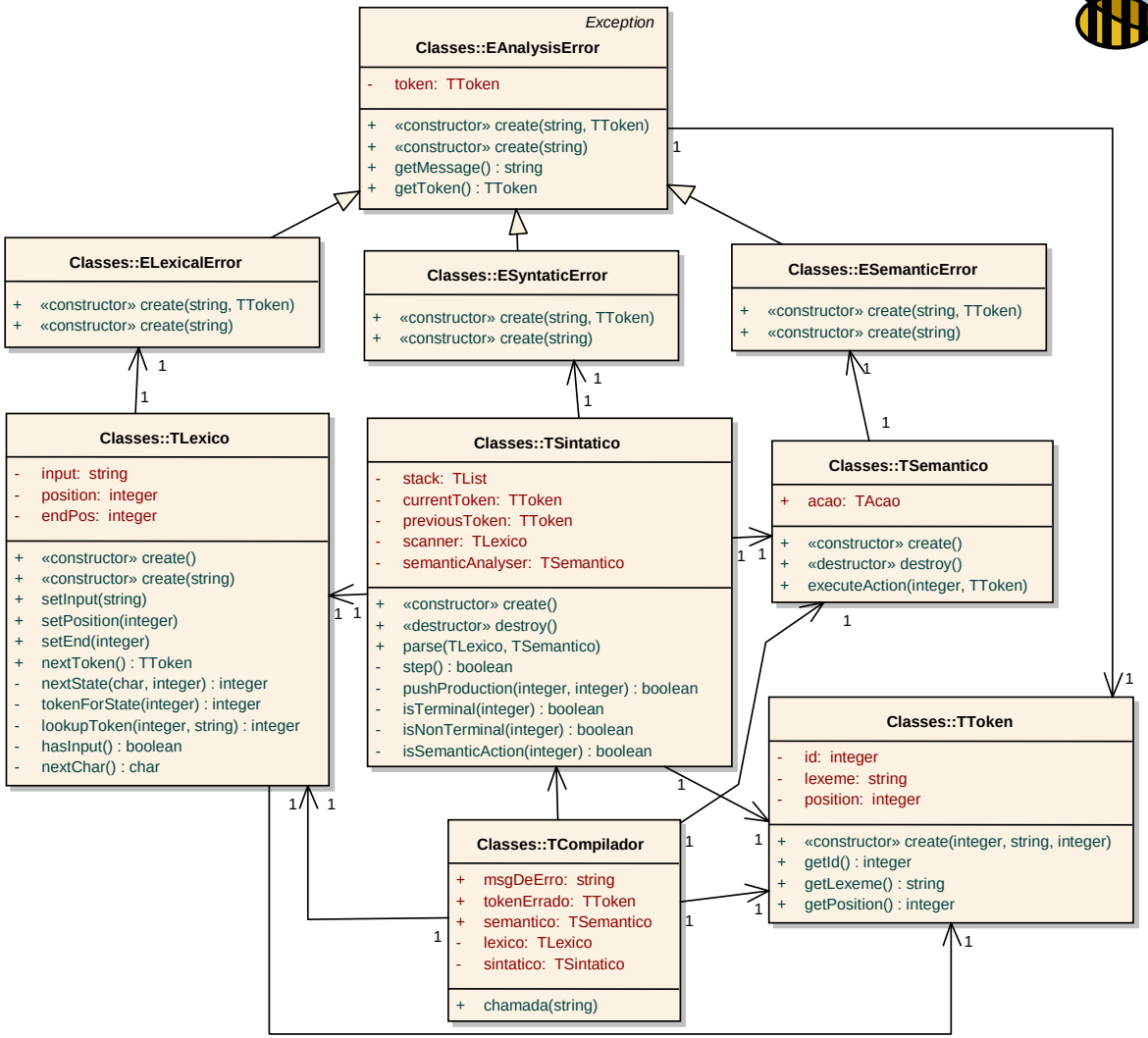
Especificação da ferramenta

Diagrama de Classes



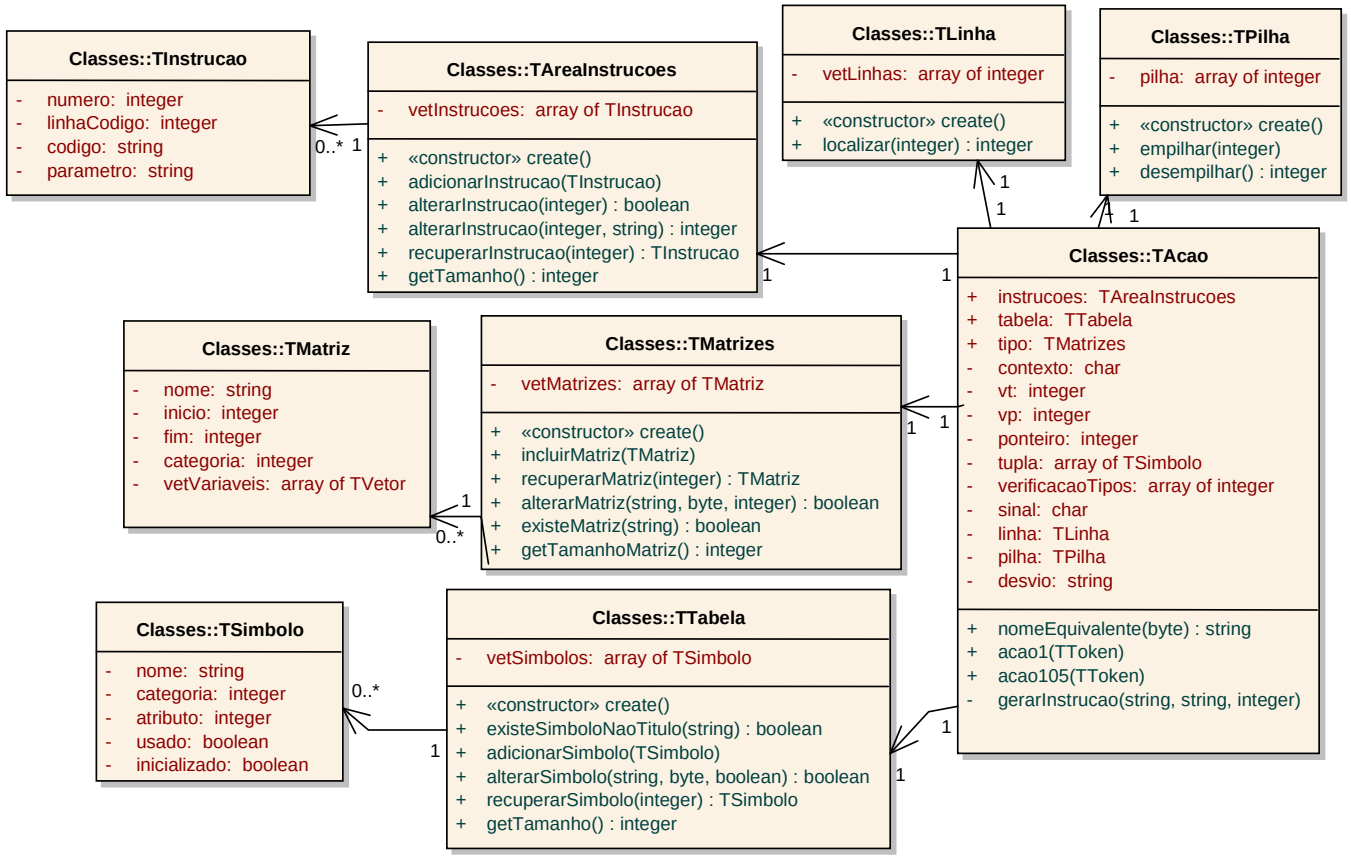
Especificação da ferramenta

Classes de Compilação



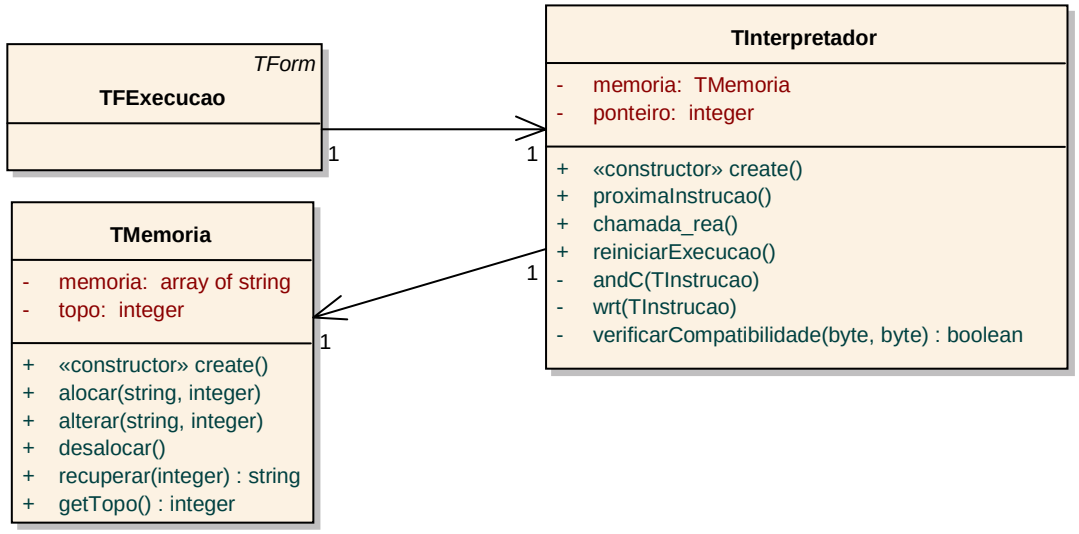
Especificação da ferramenta

Classes da análise semântica



Especificação da ferramenta

Classes de Execução



Especificação da linguagem

Especificação dos tokens

```
CONSTANTE_INTEIRO : {DIGITO}+
CONSTANTE_REAL : {DIGITO}+ "." {DIGITO}+
CONSTANTE_LITERAL : ' ([^'\n])* '
COMENTARIO : "{" [^"}"]*"}"
IDENTIFICADOR : {LETRA}({LETRA}|{DIGITO}|"_" )*
```

Especificação sintática

```
<algoritmo> → algoritmo IDENTIFICADOR ;
                    <declaracao_tipo>
                    <declaracao_variaveis>
              inicio
                  <lista_comandos>
              fim .
```

Especificação semântica

<algoritmo> → algoritmo IDENTIFICADOR **#1** ;
<declaracao_tipo>
<declaracao_variaveis>
inicio **#2**
<lista_comandos>
fim. **#3**

ação#1: Adicionar o nome do algoritmo na tabela de símbolos.

ação#2: Informar a linha onde iniciam os comandos do algoritmo.

ação#3: Gerar a instrução STP (final do algoritmo).
Informar se alguma variável não foi utilizada ou inicializada.

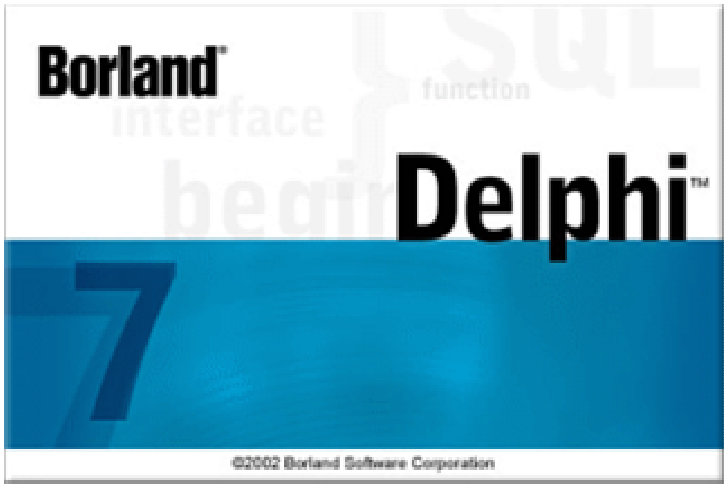
Especificação da linguagem

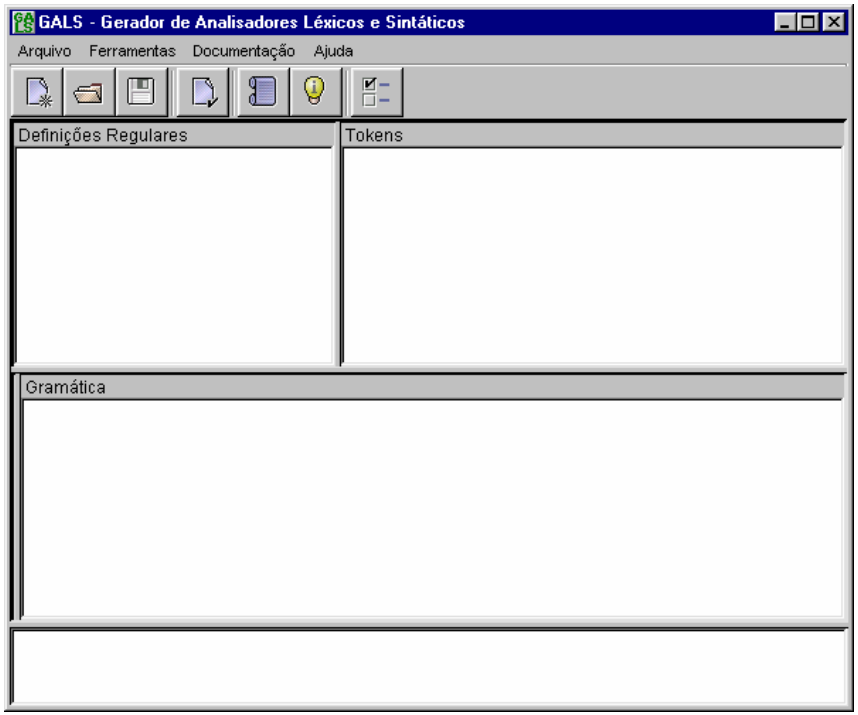
Linguagem intermediária

ALGORITMO	CÓDIGO INTERMEDIÁRIO
ALGORITMO oi_mundo ; **INÍCIO** **ESCREVA** ('oi mundo!'); **FIM** .	0 LDS 'oi mundo!' 3 1 WRT 0 3 2 STP 0 4

CÓDIGO	PARÂMETRO	FUNCIONAMENTO
LDS	constante	Alocar uma posição na memória. Armazenar a constante passada por parâmetro nesta posição.
STP	0	Encerrar a execução.
WRT	0	Escrever na tela o valor armazenado na última posição da memória. Desalocar a última posição da memória.

Implementação





DELPHI 7

GALS

Implementação

ação#3: Gerar a instrução STP (final do algoritmo).

Informar se alguma variável não foi utilizada ou inicializada.

```
procedure TAcao.acao3(token : TToken);
  ...
  begin
    gerarInstrucao('STP','0', token.getPosition);
    for ind:= 0 to tabela.getTamanho-1 do begin
      if tabela.recuperarSimbolo(ind).getUsado <> true then
        aviso:=  'AVISO: A variável "' + tabela.recuperarSimbolo(ind).getNome
                 + '" não foi utilizada. Ela é necessária?'
      else
      if (tabela.recuperarSimbolo(ind).getInicializado <> true) then begin
        aviso:=  'AVISO: A variável "' + tabela.recuperarSimbolo(ind).getNome
                 + '" não foi inicializada.';
        ...
      end;
    end;
  end;
```

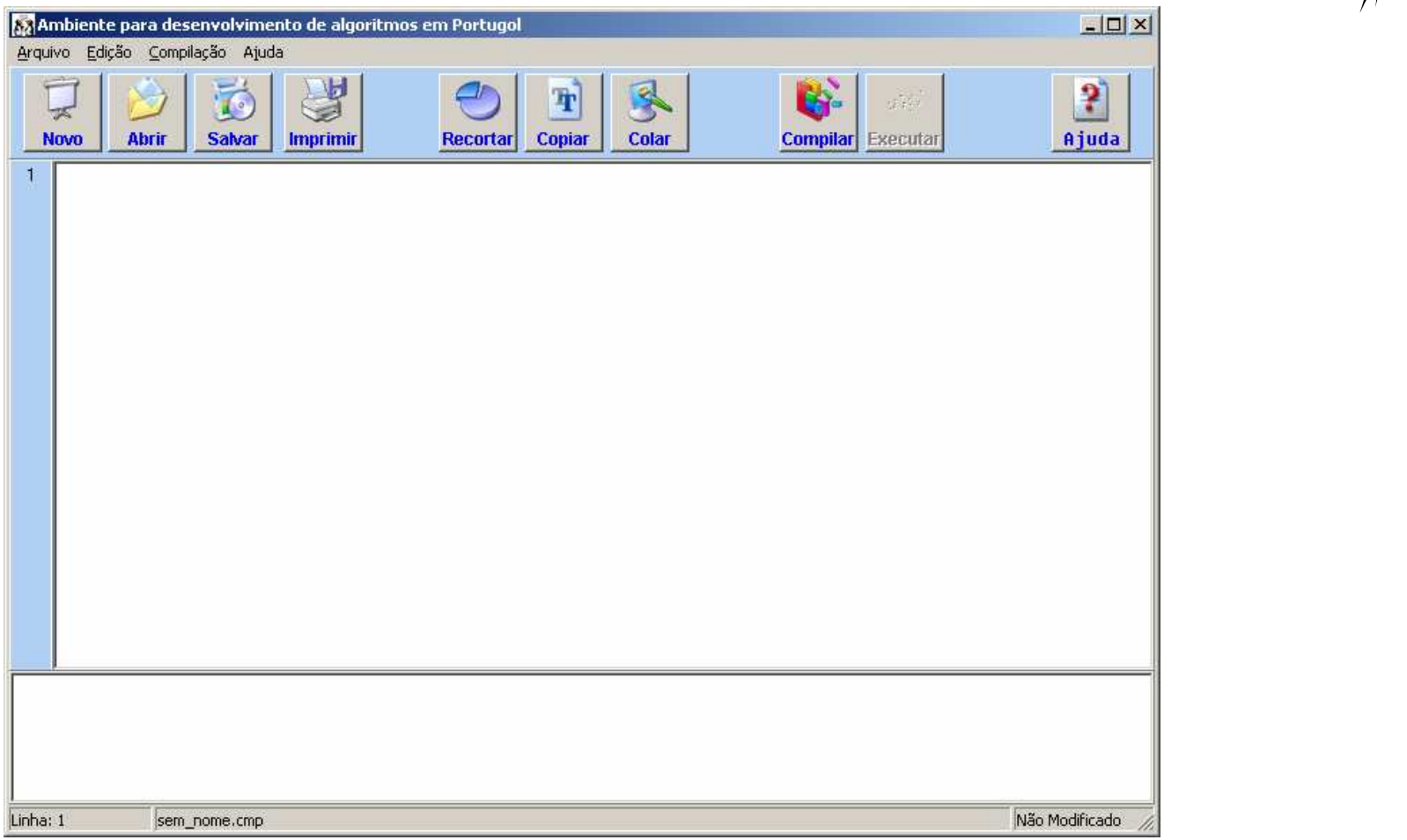
Implementação

LDS	constante	Alocar uma posição na memória. Armazenar a constante passada por parâmetro nesta posição.

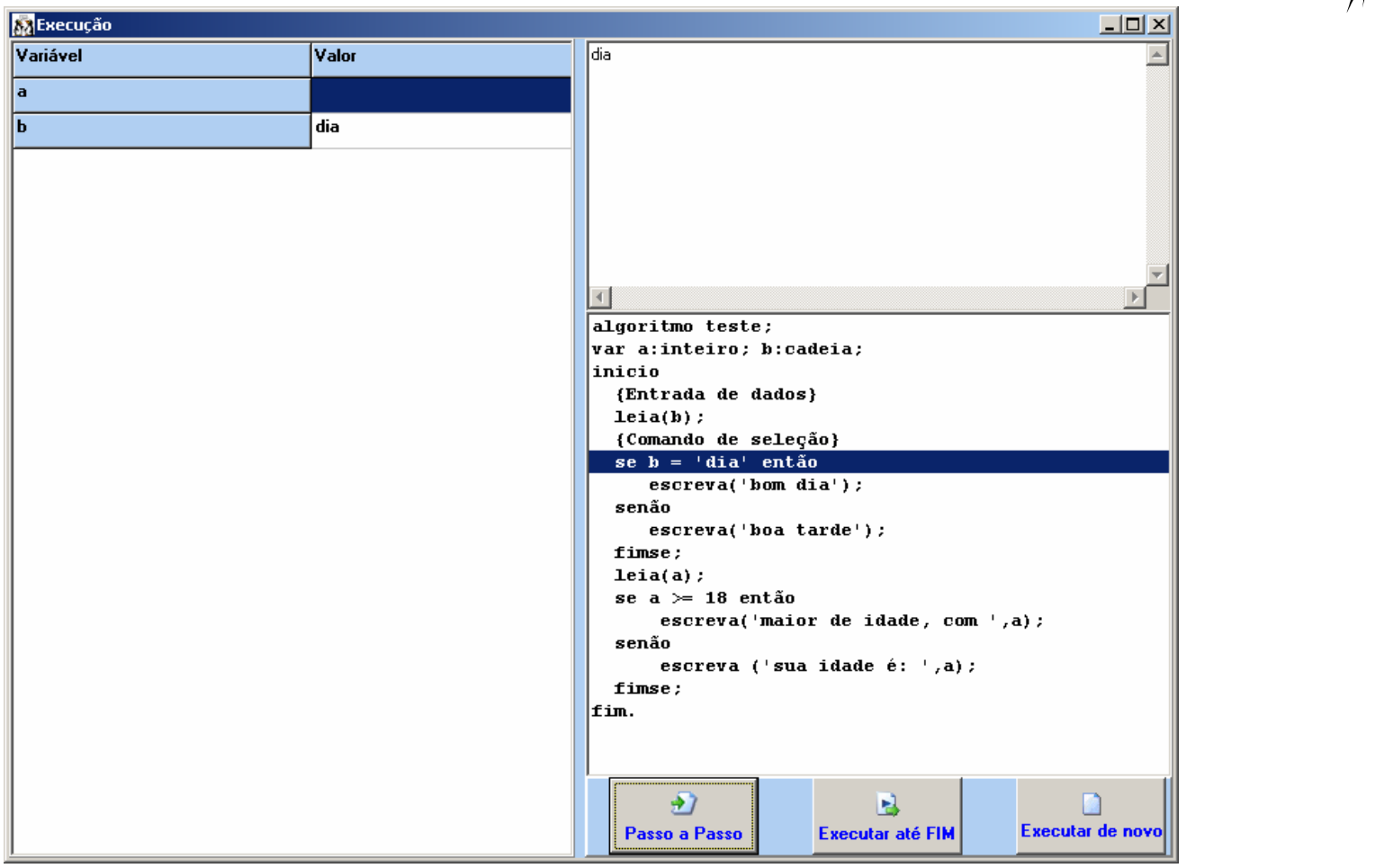
```
 procedure TInterpretador.lds(instrucao: TInstrucao);
  ...
  begin
    try
      constante_sem_aspas:=instrucao.getParametro;
      memoria.alocar(constante_sem_aspas,memoria.getTopo);
      ponteiro:=ponteiro+1;
      proximaInstrucao;
    except
      funcao.imprimirTela('ERRO DURANTE A EXECUÇÃO: Impossível carregar
constante
                          do tipo CADEIA.');
    end;
  end;
```

Operacionalidade

Operacionalidade



Operacionalidade



1 (Ajuda de programação)

Arquivo Editar Indicador Opções Ajuda

Conteúdo | Índice | Voltar | Imprimir | << | >>

AJUDA SOBRE PORTUGOL

Algoritmo
Declaração de Tipo
Declaração de Variáveis
Tipos primitivos
Comando de Atribuição (:=)
Comando de Entrada de dados ou Leitura (Leia)
Comando de Saída de dados ou Escrita (Escreva)
Comando de Seleção (Se-Senão)
Comando de Seleção (Escolha-Caso)
Comando de Repetição (Enquanto)
Comando de Repetição (Repita-Até)
Comando de Repetição (Para-Faça)

5 (Ajuda de programação)

Arquivo Editar Indicador Opções Ajuda

Conteúdo | Índice | Voltar | Imprimir | << | >>

COMANDO DE ENTRADA DE DADOS OU LEITURA (LEIA)

Como escrever?

```
LEIA ( <lista de variáveis> ) ;
```

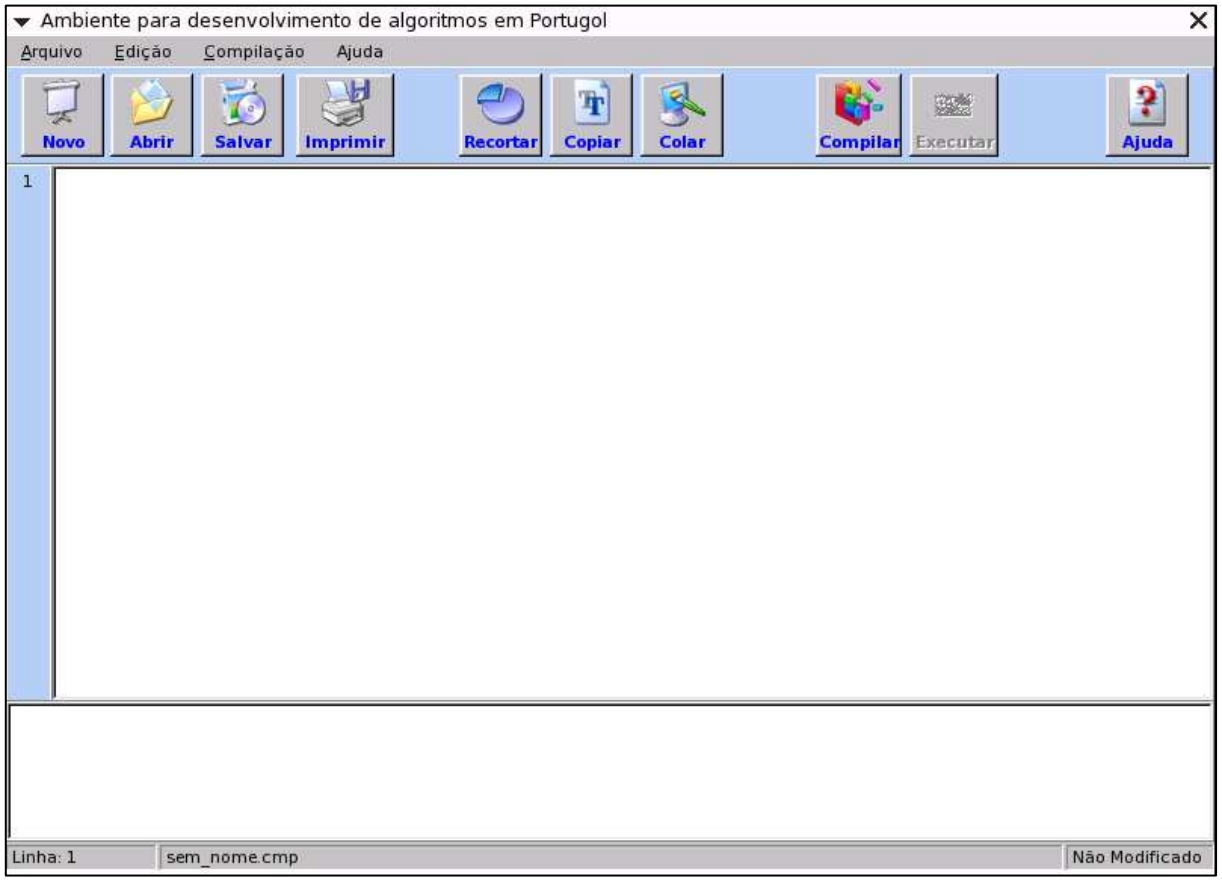
onde:
<lista de variáveis> : contém a relação dos nomes (ou **identificador**es) das variáveis nas quais serão armazenados os valores digitados. As variáveis só podem armazenar valores do tipo declarado, conforme a declaração de variáveis.

EXEMPLO: quando da execução do algoritmo abaixo, deverão ser digitados primeiramente valores do tipo **CADEIA** que serão armazenados nas variáveis *nome01* e *nome02*. E quando da execução do segundo comando **LEIA**, deverão ser digitados valores do tipo **REAL** para *salário01* e *salário02*.

```
ALGORITMO com_entrada_de_dados ;
VAR
    salário01 , salário02 , média : REAL ;
    nome01, nome02 : CADEIA ;
INÍCIO
    ESCREVA ( 'digite o nome de dois funcionários:' );
    LEIA ( nome01 , nome02 );
    ESCREVA ( 'digite o salário de cada funcionário:' );
    LEIA ( salário01 , salário02 );
    média := ( salário01 + salário02 ) / 2 ;
    ESCREVA ( 'o salário de ', nome01, 'é: ', salário01 );
    ESCREVA ( 'o salário de ', nome02, 'é: ', salário02 );
    ESCREVA ( 'a média aritmética dos salários é:', média );
FIM .
```

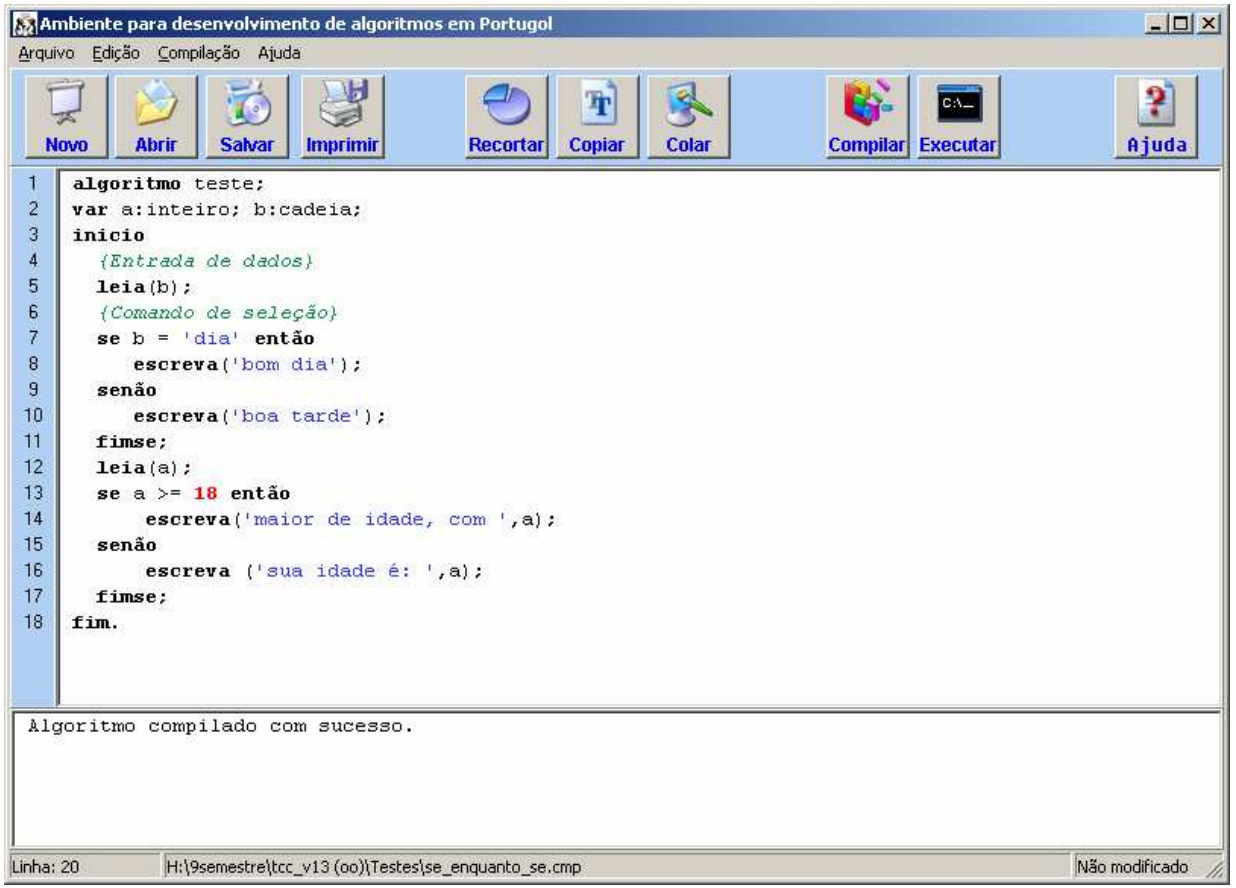
Operacionalidade

```
wine Compilador.exe
```



Critérios de Qualidade

- Possui ajuda;
- Informa erros;
- É adequada ao currículo;
- Possui execução passo a passo;
- Não perde informações;
- Fácil leitura;
- Desperta interesse;
- Faz uso de cores.



Resultados

- **Aceitação por parte dos alunos e do professor de Introdução à Programação (BCC/FURB)**
- **Trabalhos correlatos**
  - **Interface amigável (AMBAP)**
  - **Portugol (ASA)**
  - **Digitação de algoritmos (Software para o auxílio ao aprendizado de algoritmos)**
  - **Tratamento de erros (CIFluxProg)**

Conclusão

- **Desafio do ensino de programação**
- **Ambiente interativo de aprendizagem personalizado**
- **Diagnóstico de erros**
- **Experiências não possíveis em sala de aula**

Extensões

- **Mensagens e tratamento de erros**
- **Registros**
- **Matrizes multidimensionais**
- **Subrotinas (funções e procedimentos)**
- **Versão para Linux**

Não é digno de saborear o mel, aquele que se afasta da colméia com medo das picadelas das abelhas.

Willian Shakespeare