

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Projeto Tira-Teima

Documentação

Índice

Introdução.....	3
O Projeto Tira-Teima	3
Configuração e Execução do Tira-Teima	4
Rodando o Tira-Teima como uma aplicação.....	4
Rodando o Tira-Teima como uma Applet	4
Rodando o Tira-Teima como uma Applet Escondida	5
O Arquivo de Comandos do Tira-Teima	6
Considerações Gerais	6
Estrutura do Arquivo de Comandos.....	7
Definindo Passos	7
Comando <i>Não Faz Nada</i>	7
Declaração de Variáveis	8
Variáveis de Tipos Simples	8
Variáveis de Tipos Definidos Pelo Usuário.....	8
Arrays e Matrizes	9
Criando Várias Variáveis ao Mesmo Tempo	9
Definindo Novos Tipos	9
Atribuição de Valores a Variáveis	10
Trabalhando com o Console	10
Trabalhando com Arquivos	10
Declarando um Arquivo (Simbólico)	11
Associação Arquivo Físico-Simbólico	11
Abrindo um Arquivo para Leitura	11
Abrindo um Arquivo para Escrita	11
Fechando um Arquivo	12
Lendo e Escrevendo em Arquivos	12
Comentários	12

Introdução

Este documento descreve os procedimentos de configuração e uso do projeto Tira-Teima, bem como a sintaxe de seus arquivos de comandos. Este documento não descreve detalhes de arquitetura e implementação do projeto Tira-Teima, sendo necessário consultar a documentação específica, disponível gratuitamente, bem como o código fonte do Tira-Teima.

Esta documentação é destinada a qualquer professor, instrutor ou aluno que deseje utilizar o Tira-Teima na elaboração de tutoriais e/ou na simulação de algoritmos simples.

O Projeto Tira-Teima

O Tira-Teima é um visualizador iterativo de algoritmos que permite ao usuário executar o código passo a passo e verificar o valor das variáveis e dos arquivos em cada passo. Ele foi projeto para auxiliar no ensino de disciplinas introdutórias de computação, como Computação Básica ou Introdução à Ciência da Computação.

Na sua versão atual, o Tira-Teima lê um arquivo de comandos que geram a visualização desejada. É planejado, no entanto, para futuras versões, que esses comandos possam ser gerados automaticamente a partir de um programa escrito em uma linguagem de programação convencional, como C ou Pascal.

A Figura 1 abaixo ilustra a janela do Tira-Teima ao executar um algoritmo. Note que o ambiente possui seções de visualização de código fonte, variáveis, arquivos e console:

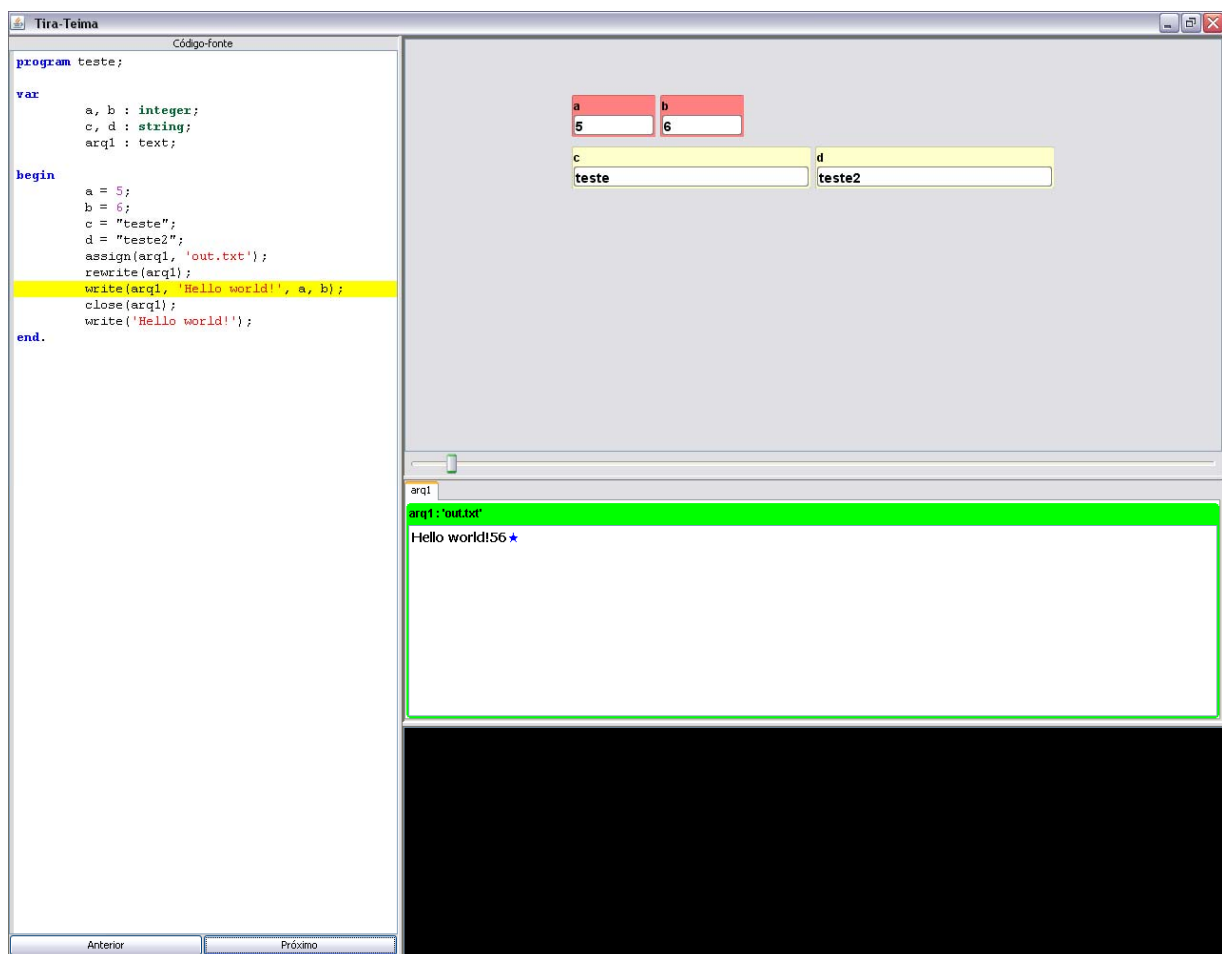


Figura 1

Configuração e Execução do Tira-Teima

O projeto Tira-Teima foi todo escrito em Java e, portanto, requer que o JRE (Java Runtime Enviroment) esteja instalado no sistema. Para rodar a última versão do Tira-Teima, é preciso ter instalado o JRE 1.5 ou superior. Além disso, é recomendado um mínimo de 128 MB de memória livre para rodar o sistema.

O Tira-Teima foi projetado para rodar tanto como uma aplicação quanto como uma Applet. Na biblioteca do Tira-Teima (*tirateima.jar*) já estão disponibilizadas classes para esses dois modos de execução.

Em qualquer um dos modos de execução, o Tira-Teima espera receber como parâmetros os nomes de dois arquivos: um *arquivo de código fonte* e um *arquivo de comandos*.

O arquivo de código fonte pode estar escrito em qualquer linguagem de programação, no entanto, o Tira-Teima só oferece no momento suporte ao *highlighting* da linguagem Pascal. O arquivo de comandos contém as instruções para gerar a visualização. A estrutura e sintaxe desse arquivo é descrita nas seções seguintes.

Rodando o Tira-Teima como uma aplicação

Para rodar o Tira-Teima localmente como uma aplicação basta simplesmente rodar o arquivo jar como uma aplicação Java, usando o JRE. É possível também rodá-lo pelo prompt do sistema, com o seguinte comando:

```
java -jar <caminho para o "tirateima.jar">
```

Alternativamente, pose-se acesar explicitamente a classe que roda o Tira-Teima como uma aplicação. Assumindo-se que o *jar* do Tira-Teima foi adicionado ao classpath, pode-se fazer:

```
java tirateima.main.Programa
```

Ou, ainda:

```
java -classpath tirateima.jar tirateima.main.Programa
```

Nos procedimentos descritos anteriormente, nenhum parâmetro foi passado para o Tira-Teima. Se isso acontecer, o Tira-Teima vai abrir sem nenhum programa carregado e mostrará uma mensagem de advertência ao usuário. Para passar os parâmetros para o Tira-Teima, é preciso complementar o comando:

```
java -jar tirateima.jar <arquivo de código fonte> <arquivo de comandos>
```

Caso o algoritmo a ser visualizado acesse arquivos no modo de leitura, é preciso que esses arquivos estejam no mesmo diretório do Tira-Teima. Mais detalhes sobre arquivos podem ser encontrados na seção de comandos do Tira-Teima.

Rodando o Tira-Teima como uma Applet

Ao rodar como applet, o Tira-Teima espera um parâmetro definindo o modo como a applet será mostrada na tela: integrada ao navegador, com uma janela própria, ou escondida.

Quando integrada ao navegador, a applet do Tira-Teima será rodada como qualquer outra applet Java: como um objeto na página.

No modo de janela, por sua vez, será criada uma janela separada que conterà a interface do Tira-Teima. Se a página que contém a applet for fechada, a janela criada também será fechada.

No modo escondido, nada será mostrado até que o método *abrirEntrada* da applet seja chamado, quando, então, uma nova janela será aberta. Esse último modo é bastante útil para controlar dinamicamente o Tira-Teima por meio de uma linguagem de script, como o JavaScript, por exemplo.

Para definir o modo de execução do Tira-Teima, é preciso passar um parâmetro *modo* que pode assumir os seguintes valores:

- *applet*: para o modo de applet integrada ao navegador
- *janela*: para o modo de janela
- *escondido*: para o modo escondido

No modo *janela*, pode-se ainda passar dois parâmetros adicionais: *largura* e *altura*, para definir as dimensões iniciais da janela aberta.

Além da definição do modo de execução, é preciso passar os nomes do arquivo fonte e do arquivo de comandos, por meio dos parâmetros *arq_fonte* e *arq_texto*, respectivamente. No caso da applet, esses parâmetros são obrigatórios. O endereço dos arquivos de código fonte e de comandos devem ser relativos ao diretório onde está a página no servidor. Preferencialmente, esses arquivos devem estar no mesmo diretório que a página.

Se o Tira-Teima precisar acessar algum arquivo em modo de leitura, o arquivo em questão deve estar no mesmo diretório que a página.

Abaixo segue um exemplo de arquivo HTML que abre o Tira-Teima no modo de janela:

```
<html>
  <head>
    <title>Tira-Teima</title>
  </head>
  <body>
    <applet archive="tirateima.jar" code="tirateima.main.Applet">
      <param name="modo" value="janela">
      <param name="arq_fonte" value="teste.pas">
      <param name="arq_texto" value="teste.txt">
    </applet>
  </body>
</html>
```

Rodando o Tira-Teima como uma Applet Escondida

A classe *tirateima.main.AppletEscondida* foi criada como uma alternativa ao modo escondido da classe Applet. Ela permite que a janela do Tira-Teima seja mostrada ao se clicar um botão, sem que haja necessidade de integrar código JavaScript à página.

Basicamente, a classe *AppletEscondida* fica integrada ao browser na forma de um botão que, quando apertado, ativa a janela do Tira-Teima. Para definir o texto que aparece no botão, basta setar o parâmetro *nome*.

Abaixo segue um exemplo de uso:

```
<html>
  <head>
    <title>Tira Teima</title>
  </head>
  <body>
    <applet
      archive="tirateima.jar"
      code="tirateima.main.AppletEscondida"
      width=120 height=40>
      <param name="arq_fonte" value="programa504.pas">
      <param name="arq_texto" value="roteiro504.txt">
      <param name="largura" value="800"> <!-- largura da janela -->
      <param name="altura" value="600"> <!-- altura da janela -->
      <param name="nome" value="Exemplo 504"> <!-- texto no botão -->
    </applet>
  </body>
</html>
```

O Arquivo de Comandos do Tira-Teima

Considerações Gerais

O arquivo de comandos do Tira-Teima é um arquivo texto comum, com uma série de instruções cuja sintaxe será definida a seguir.

Recomenda-se que o arquivo só use caracteres ASCII padrão. Caso seja necessário o uso de caracteres especiais, como os caracteres acentuados do português, o arquivo deve utilizar o *character encoding* padrão do sistema em que o Tira-Teima irá rodar.

Para fins de entendimento das regras a seguir, são feitas as seguintes definições:

- Um *identificador* ou *nome* é qualquer seqüências de letras, dígitos e caractere de sublinhado ‘_’, iniciada por letra ou caractere de sublinhado.
- Um *literal-caractere* (ou apenas *caractere*, quando não houver ambigüidade) é um caractere ou uma seqüência de escape entre aspas simples. O caractere não pode ser aspa simples, contra-barra ou quebra de linha. As seqüências de escape aceitas são listadas na Tabela 1. As aspas não fazem parte do valor do *literal-caractere*.
- Uma *string* é uma seqüência de caracteres entre aspas duplas. A seqüência não pode conter aspas duplas, contra-barras ou quebras de linha. As seqüências de escape aceitas são listadas na Tabela 1. As aspas não fazem parte do valor da *string*.

Seqüência de Escape	Caractere
\"	Aspa dupla (seqüência válida somente em strings)
\'	Aspa simples (seqüência válida somente em caracteres)
\\	Contra-barra
\t	Tabulação
\n	Nova linha
\r	Retorno de Carro

Tabela 1 - Seqüências de Escape Válidas

- Um *número inteiro* é qualquer seqüência de dígitos, com ou sem um sinal inicial. Os valores serão interpretados como números decimais.
- Um *número real* é qualquer seqüência de zero ou mais dígitos, seguida de um ponto decimal e um ou mais dígitos. Adicionalmente, pode haver um sinal na frente do número e uma indicação de expoente na base dez na forma (*e* ou *E*)(*número inteiro*). Ex: +314.1592e-2.
- Uma *constante* é uma string, um caractere, um número inteiro ou um número real.

Ao final da explicação de cada comando, é mostrada uma expressão que resume a sintaxe do mesmo.

Essas expressão segue as seguintes regras:

- Termos entre < e > devem ser substituídos por um valor correspondente, sem os sinais < e >.
- Termos ou expressões entre colchetes ([e]) são opcionais e podem ou não ser substituídos.
- Termos seguidos de ... podem ser repetidos indefinidamente.
- Todos os sinais de pontuação tais como vírgula, ponto-e-vírgula e ponto são obrigatórios, quando forem substituídos.

Estrutura do Arquivo de Comandos

O Tira-Teima utiliza a noção de *passo*. O passo é a unidade básica de navegação do algoritmo. Cada passo é traduzido em um estado do Tira-Teima, ou seja, as mudanças na visualização são sempre feitas entre dois passos consecutivos.

Do ponto de vista do arquivo de comandos, um passo nada mais é que um conjunto de um ou mais comandos do Tira-Teima. Se existir mais de um comando para um determinado passo, todos os comandos serão executados e será salvo somente o resultado dessa execução, sem que sejam vistas as modificações intermediárias entre os comandos. Isso é útil para simular funções e procedimentos complexos, sem que o usuário final veja os processos intermediários.

Portanto, o arquivo de comandos do Tira-Teima é uma seqüência de passos. Em que cada passo poderá conter um ou mais comandos.

Para todas as regras abaixo, os espaços em branco e quebras de linha serão ignorados. Além disso, as palavras-chave são insensíveis ao caso, ou seja, *LiNe* e *line* são a mesma coisa.

Definindo Passos

O Tira-Teima assume que cada passo do algoritmo está relacionado a uma linha do programa original (código fonte), portanto, todos os passos são iniciados pelo número da linha que deve ser destacada no código fonte.

Para tornar o programa mais inteligível, pode-se preceder o número de linha com a palavra chave *line*. Indicações de linhas negativas significam que nenhuma linha deve ser destacada.

Logo após o número de linha, deve vir o comando ou bloco de comandos a ser executado. Para agrupar comandos em bloco, basta colocá-los entre chaves.

Portanto, um passo do Tira-Teima é da seguinte forma:

[line] <número da linha> <comando>

ou

[line] <número da linha> {<comando> [<comando> ...]}

E um arquivo de comandos do Tira-Teima é um conjunto de passos:

<passo> [<passo> ...]

Comando Não Faz Nada

Para apenas destacar a linha, sem fazer nada, basta colocar um ponto-e-vírgula logo depois da indicação de linha:

[line] <número da linha>;

Declaração de Variáveis

A noção de variável no Tira-Teima é a mesma que a de qualquer linguagem de programação imperativa. Ao encontrar uma declaração de variável, o Tira-Teima criará um objeto gráfico para representá-la. Seguem abaixo as regras para declarar variáveis.

Variáveis de Tipos Simples

Os *tipos simples* são tipos intrínsecos ao Tira-Teima. Estão listados na tabela abaixo:

Tipo	Descrição
int	Valores inteiros com sinal Equivalente ao tipo <i>int</i> de Java.
real	Valores reais Equivalente ao tipo <i>double</i> de Java.
string	Seqüências de caracteres Equivalente ao tipo <i>String</i> de Java.
char	Caracteres Equivalente ao tipo <i>char</i> de Java.
boolean	Valores booleanos <i>true</i> e <i>false</i> Equivalente ao tipo <i>boolean</i> de Java

Tabela 2 - Tipos básicos do Tira-Teima

Para declarar uma variável X, do tipo inteiro, faça:

```
int X;
```

Em geral,

```
<tipo> <nome da variável>;
```

Não podem existir duas variáveis com mesmo nome num mesmo escopo. A definição de escopo é vista na seção de funções.

Variáveis de Tipos Definidos Pelo Usuário

Um tipo definido pelo usuário é um registro, cuja definição é explicada mais a frente. De qualquer forma, um registro é o equivalente a uma *struct* de C ou um *record* de Pascal.

Portanto, assumindo que exista um registro chamado *Pessoa*, para criar uma variável do tipo *Pessoa*, faça:

```
Pessoa <nome da variável>;
```

Em geral,

```
<nome do tipo> <nome da variável>;
```


Arrays e Matrizes

Para criar arrays e matrizes, adicione um índice logo após o nome da variável. Esse índice, que vem entre colchetes, indica a dimensão do array. Por exemplo, para criar um array de inteiros com 5 elementos:

```
int meu_array[5];
```

Para matrizes, basta colocar dois índices. Exemplo:

```
string matrix[5, 10];
```

Criando Várias Variáveis ao Mesmo Tempo

Para criar várias variáveis ao mesmo tempo, basta separar os nomes das variáveis com vírgula:

```
<tipo> var1, var2, var3;
```

Portanto, para declarar variáveis em geral, segue a regra (onde *tipo* pode ser um tipo básico ou o nome de um tipo definido pelo usuário):

```
<tipo> <nome>[[<índice>[, <índice>]]], <nome>[[<índice>[, <índice>]] ...];
```

Definindo Novos Tipos

Como em qualquer outra linguagem, um tipo definido pelo usuário é um registro, ou seja, um conjunto de campos. A sintaxe para definição de tipos é a seguinte:

```
record <nome do novo tipo> {
    <tipo1> <campo1>;
    <tipo2> <campo2>;
    ...
    <tipoN> <campoN>;
}
```

Por exemplo, definindo o tipo Pessoa:

```
record Pessoa{ string nome; int idade;}
```

As regras de criação de variáveis também valem para os campos de um registro, ou seja:

- O tipo de um campo pode ser outro registro, desde que ele tenha sido declarado previamente
- Um campo pode ser um array ou uma matriz, bastando que se especifique o índice
- Pode-se declarar vários campos ao mesmo tempo, fazendo-se: <tipo> <campo1>, <campo2>, ...;

Em geral:

```
record <nome> {
    <tipo> <nome>[[<índice>[, <índice>]]], <nome>[[<índice>[, <índice>]] ...];
    [<tipo> <nome>[[<índice>[, <índice>]]], <nome>[[<índice>[, <índice>]] ...]; ...]
}
```

Atribuição de Valores a Variáveis

Pela própria natureza limitada do Tira-Teima, só se pode atribuir constantes a variáveis, ou seja, não se pode atribuir o valor de uma variável a outra. Portanto, a sintaxe geral da atribuição é:

<variável> = <constante>;

Note que variável não é necessariamente uma variável simples, mas pode ser um campo de um registro ou um elemento de um vetor ou matriz. Portanto, seguem as seguintes regras para definir recursivamente uma variável:

- Se a variável *v* é um registro, então *v.<um campo de v>* também é uma variável válida
- Se a variável *v* é um array ou matriz, então *v[<índice de um elemento de v>]* também é uma variável válida

Resta apenas garantir que a constante seja do mesmo tipo de *v*. Não é possível atribuir uma string para um caractere ou vice-versa, ou uma string para um inteiro ou real. A exceção é a atribuição de um número inteiro para um real.

Em geral:

<nome>[<<índice> ou .<nome>> ...] = <constante>

O valor é atribuído sempre à variável no escopo o mais local possível. Essa questão é tratada na seção de funções.

Trabalhando com o Console

O console do Tira-Teima simula o prompt do sistema operacional. Na versão atual, o console é somente de saída e não lê entradas do usuário. O comando de escrita do Tira-Teima é idêntico ao comando *write* do Pascal. Portanto:

write([<variável ou constante>] [,<variável ou constante> ...]);

Note que a lista de parâmetros pode ser vazia, mas, diferentemente de Pascal, os parênteses são obrigatórios.

Para se acessar campos de registros ou elementos de arrays ou matrizes, segue-se a mesma regra do ponto e do índice, respectivamente.

Pode-se também usar o comando *writeln* com a diferença que esse comando adiciona uma nova linha após escrever os valores.

Trabalhando com Arquivos

A manipulação de arquivos do Tira-Teima foi toda baseada em Pascal. Portanto, os comandos e conceitos a seguir se comportam da mesma maneira que nessa linguagem.

Note que o suporte a arquivos binários no Tira-Teima foi descontinuado, logo, sempre que o termo *arquivo* for usado, ele refere-se a arquivos texto.

Um comentário adicional é que, conforme explicado na seção de configuração e execução, se um arquivo for aberto para leitura, ele deve existir fisicamente e estar presente no mesmo diretório do Tira-Teima.

Declarando um Arquivo (Simbólico)

Para declarar um novo arquivo, siga a mesma sintaxe da declaração de variáveis usando o tipo *text*.

Apesar da sintaxe igual, existem duas restrições para variáveis do tipo arquivo: não podem existir arrays ou matrizes de arquivos e arquivos não podem ser campos de arrays.

Portanto em geral:

```
text <arq1>, <arq2>, ..., <arqN>;
```

ou, mais formalmente,

```
text <nome>[, <nome> ...];
```

Associação Arquivo Físico-Simbólico

Para associar o arquivo simbólico a um arquivo físico, use o comando *assign*:

```
assign(<nome>, <string>;
```

Onde a string contém o nome do arquivo físico. Esse nome deve, necessariamente, ser uma constante; ou seja, não se pode pegar o nome do arquivo físico de uma variável.

Resta lembrar que o comando *assign* não abre o arquivo, apenas associa o nome.

Abrindo um Arquivo para Leitura

Para abrir um arquivo exclusivamente para leitura, use o comando *reset*:

```
reset(<nome>;
```

O nome se refere à variável *text*, ou seja, ao arquivo simbólico. Para que seja aberto, o arquivo simbólico de estar associado a um arquivo físico e este arquivo físico deve existir, poder ser acessado e estar no mesmo diretório do Tira-Teima.

Abrindo um Arquivo para Escrita

Para abrir um arquivo exclusivamente para escrita, use o comando *rewrite*:

```
rewrite(<nome>;
```

O nome se refere à variável *text*, ou seja, ao arquivo simbólico. Para que seja aberto, o arquivo simbólico de estar associado a um arquivo físico, que não precisa existir.

Os arquivos abertos para escrita só existem em memória e seus dados não são preservados após a execução.

Fechando um Arquivo

Para fechar um arquivo, use o comando *close*:

```
rewrite(<nome>);
```

O nome se refere à variável *text*, ou seja, ao arquivo simbólico. O arquivo simbólico permanece associado ao arquivo físico depois de fechado.

Lendo e Escrevendo em Arquivos

Para se escrever em um arquivo, também usa-se o comando *write*. No entanto, o primeiro parâmetro é o nome do arquivo simbólico. Portanto:

```
write(<nome>, [<variável ou constante>] [<variável ou constante> ...]);
```

O comando *writeln* também funciona da mesma forma para arquivos.

Para se ler dados de arquivos, deve-se usar os comandos *read* e *readln*. Devido à ausência de strings de tamanho fixo no Tira-Teima, esses comandos têm uma sintaxe levemente diferente de *write* e *writeln*. A seguir, estão as regras.

Só pode ser lida uma variável por vez. Dado o tipo da variável, o Tira-Teima vai inferir o tipo de dado a ser lido. Só podem ser lidas variáveis (que incluem campos de registros e elementos de arrays e matrizes) dos tipos básicos, à exceção de *boolean*, ou seja, *int*, *real*, *string* e *char*.

Para o tipo *string*, pode ser passado um parâmetro adicional definindo quantos caracteres devem ser lidos. Se esse parâmetro não for informado, será lido até a primeira quebra de linha.

Para os tipos *int* e *real*, todos os espaços em branco (inclusive quebras de linha) antes do primeiro dígito serão ignorados.

Em geral:

```
read(<nome arq>, <nome var>[, <número de caracteres, se var for string>]);
```

O comando *readln* também só lê uma variável. No entanto, após ler os dados, ele vai ignorar todos os caracteres até a primeira quebra de linha que encontrar. Portanto,

```
readln(<nome arq>, <variável string>, <número de caracteres X>);
```

vai ler X caracteres e ignorar até o final da linha.

```
readln(<nome arq>, <variável string>);
```

é o mesmo que *read*.

Comentários

Se um caractere *#* for encontrado, todos os caracteres até a próxima quebra de linha serão ignorados. Ou seja:

```
#isso é um comentário  
line 1 int x, y; #cria x e y
```