1 Arquivos

Um arquivo pode ser visto como uma sequência de caracteres, como uma string; a diferença entre strings e arquivos é a maneira como são armazenados:

- Uma string é uma região da memória RAM do computador, usada (normalmente) dentro de um único programa. Como o acesso a qualquer posição de memória é simples e rápido, podemos selecionar qualquer posição da string, como já vimos:
  ```
  a <- "uma string"
  mostra a[4]  // mostrara' 's', porque a[4]='s'
  ```
- Um arquivo é uma sequência de caracteres que fica armazenada em disco ou outra forma de memória não-volátil (HD, DVD, pendrive etc). O acesso a muitas destas mídias é lento, especialmente quando se quer selecionar uma posição qualquer do arquivo. Por isso, normalmente usamos arquivos de maneira diferente de strings: usarems um arquivo como uma sequência de caracteres que lemos do início ao fim, sem retrocesso, ou como uma sequência de caracteres que gravamos, sem podermos apagar algo que já foi escrito.

Antes de usar um arquivo, precisamos "abrí-lo". Faremos isto com o comando `abre`:

```
arquivo arq
arq <- abre "dados.txt"
```

O trecho de código acima faz o seguinte:

-

Usaremos os mesmos comandos de leitura e escrita que já definimos para teclado e tela (mostra e leia).

```
arquivo arq
arq <- abre "meuarquivo.txt"
mostra (arq) "uma linha"
para i em (0..5):
    mostra (arq) i
fecha arq
```

Este agoritmo criará um arquivo `meuarquivo.txt`, e escreverá nele o conteúdo:

```
uma linha
1
2
3
4
5
```

O próximo exemplo usa leitura e gravação em arquivos:

```
arquivo entrada, saida
entrada <- abre "arquivo-entrada.txt"
saida <- abre "arquivo-saida.txt"

leia (entrada) n
vetor(real, n) vet

para i em (0..n):
    leia (entrada) vet[i]

para i em (0..n):
    mostra (saida) i, 2.0 * vet[i]
fecha entrada
fecha saida
```

Este programa lerá de um arquivo "arquivo-entrada.txt"um inteiro n. Depois, lerá n números reais e os armazenará em um vetor. Em seguida, escreverá em outro arquivo (arquivo-saida.txt) uma tabela com o índice (i) de cada valor, e o dobro do valor na posição i do vetor:
arquivo-entrada.txt:

```
3
1.0
2.5
4.0
```

arquivo-saida.txt:

```
0 2.0
1 5.0
2 8.0
```

1.1 Em Java

Em Java, declaramos variáveis do tipo arquivo:

```
File arqentra = new File("meuarquivo.txt");
File arqsai   = new File("outroarquivo.txt");
```

Mas precisamos também criar um Scanner se o arquivo for de saída (e depois poderemos usar `sc.next(), sc.nextLine(), sc.nextInt(), sc.nextDouble()`:

```
Scanner sc = new Scanner(arqentra);
```

Se o arquivo for de entrada, primeiro é necessário criar alguma variável (neste exemplo, "s") do tipo `FileOutputStream` a partir do arquivo, e depois uma outra ("sai", neste exemplo) do tipo `PrintStream`. Assim poderemos usar `sai.printf(), sai.print(), sai.println()`:

```
FileOutputStream s = new FileOutputStream (arqsai);
PrintStream sai = new PrintStream (s, true);
```

Para um exemplo completo em Java, veja o link na página do curso.