Universidade Federal Fluminense
Luís Felipe Ignácio Cunha

# Maratona 4 – Manipulação de arquivos texto

1. **Análise de palavras em arquivo texto**

   (a) **Objetivo:** Implementar um programa em Python, utilizando subprogramações, que leia o nome de um arquivo texto e dois valores inteiros que definem um intervalo de tamanho de palavras. O programa deve exibir o conteúdo do arquivo, construir um dicionário de palavras com contagem de ocorrências e apresentar esse dicionário ordenado alfabeticamente.

   (b) **Dados de entrada:**

   - Na primeira linha da entrada padrão, o nome de um arquivo texto. Cada linha desse arquivo pode conter zero ou mais palavras.
   - Na segunda linha da entrada padrão, dois números inteiros, chamados `tamanhoMinimo` e `tamanhoMaximo`.

   (c) **Saída esperada:** O programa deve:

   i. Mostrar na saída padrão o conteúdo completo do arquivo informado, preservando os caracteres acentuados, antecedido por um cabeçalho indicando o nome do arquivo.

   ii. Produzir um dicionário em que:
   - As chaves são palavras convertidas para letras maiúsculas;
   - Todas as vírgulas presentes no texto devem ser removidas antes do processamento das palavras;
   - Apenas palavras cujo comprimento (número de caracteres) esteja no intervalo

   $$\texttt{tamanhoMinimo} \leq \text{comprimento da palavra} \leq \texttt{tamanhoMaximo}$$

   devem ser consideradas.

   iii. Exibir o dicionário de palavras produzido no item anterior, juntamente com a contagem de ocorrências de cada palavra, em ordem alfabética, no formato:

   ```
   PALAVRA ocorreu X vez/vezes
   ```

   seguindo o modelo apresentado no teste a seguir.

   (d) **Observação:** O arquivo texto referido pode conter trechos com acentuação. A saída deve preservar esses caracteres ao imprimir o conteúdo do arquivo.

   (e) **Exemplo de teste:**
   **Entrada:**

```
carta.txt
5 7
```

**Saída correspondente:**

```
Conteúdo do Arquivo: carta
Rua Nascimento Silva, cento e sete
Você ensinando pra Elizete
As canções de canção do amor demais

Lembra que tempo feliz, ai, que saudade
Ipanema era só felicidade
Era como se o amor doesse em paz

Nossa famosa garota nem sabia
A que ponto a cidade turvaria
Esse Rio de amor que se perdeu

Mesmo a tristeza da gente era mais bela
E além disso se via da janela
Um cantinho de céu e o Redentor

É, meu amigo, só resta uma certeza
É preciso acabar com essa tristeza
É preciso inventar de novo o amor

Dicionário de Palavras com Contagem de Ocorrências:
ACABAR ocorreu 1 vez
AMIGO ocorreu 1 vez
CANÇÃO ocorreu 1 vez
CANÇÕES ocorreu 1 vez
CENTO ocorreu 1 vez
CERTEZA ocorreu 1 vez
CIDADE ocorreu 1 vez
DEMAIS ocorreu 1 vez
DISSO ocorreu 1 vez
DOESSE ocorreu 1 vez
ELIZETE ocorreu 1 vez
FAMOSA ocorreu 1 vez
FELIZ ocorreu 1 vez
GAROTA ocorreu 1 vez
GENTE ocorreu 1 vez
IPANEMA ocorreu 1 vez
JANELA ocorreu 1 vez
LEMBRA ocorreu 1 vez
MESMO ocorreu 1 vez
NOSSA ocorreu 1 vez
PERDEU ocorreu 1 vez
PONTO ocorreu 1 vez
```

```
PRECISO ocorreu 2 vezes
RESTA ocorreu 1 vez
SABIA ocorreu 1 vez
SAUDADE ocorreu 1 vez
SILVA ocorreu 1 vez
TEMPO ocorreu 1 vez
```

2. **Remoção de linhas com números primos em arquivo texto**

   (a) **Objetivo:** Implementar um programa em Python, utilizando subprogramações, que leia o nome de um arquivo texto e processe seu conteúdo linha a linha. Cada linha do arquivo contém zero ou mais números inteiros separados por espaços em branco. O programa deve exibir o conteúdo original do arquivo, remover todas as linhas que contenham pelo menos um número primo e, em seguida, exibir novamente o conteúdo do arquivo após as eventuais remoções.

   (b) **Dados de entrada:**

   - Na primeira linha da entrada padrão, o nome de um arquivo texto. Cada linha desse arquivo pode conter zero ou mais números inteiros separados por um ou mais espaços.

   (c) **Saída esperada:** O programa deve:

   i. Mostrar na saída padrão o conteúdo completo do arquivo informado, antecedido por um cabeçalho que indique o nome do arquivo.
   ii. Processar o arquivo de forma a remover todas as linhas que contenham pelo menos um número primo.
   iii. Exibir novamente o conteúdo do arquivo, após as remoções, também precedido por um cabeçalho adequado.

   Um número inteiro é considerado primo se, e somente se, for maior que 1 e tiver como únicos divisores positivos o número 1 e ele próprio.

   (d) **Restrição:** Assuma que, em geral, o arquivo não pode ser mantido integralmente na memória principal. O programa deve manter em memória apenas a linha que está sendo processada em cada momento.

   (e) **Exemplo de teste:**
   **Entrada:**

   ```
   numeros.txt
   ```

   **Conteúdo do arquivo numeros:**

   ```
   4 6 8
   5 10 12
   9 21 15
   2 4 6
   ```

   **Saída correspondente:**

```
Conteúdo do Arquivo numeros:
4 6 8
5 10 12
9 21 15
2 4 6

Conteúdo do Arquivo numeros após eventuais remoções:
4 6 8
9 21 15
```

Nesse exemplo, as linhas que contêm os números 5 e 2 são removidas por apresentarem pelo menos um número primo.

3. **Detecção de células com vizinhos maiores em matriz inteira**

   (a) **Objetivo:** Implementar um programa em Python, utilizando subprogramações, que leia da entrada padrão o nome de um arquivo texto contendo uma matriz bidimensional de números inteiros. O programa deve identificar e exibir todas as posições da matriz cujos valores sejam estritamente menores que *todos* os seus vizinhos existentes. São considerados vizinhos as células adjacentes horizontal, vertical e diagonalmente.

   (b) **Dados de entrada:**

   - Na entrada padrão, o nome de um arquivo texto.
   - O arquivo referido contém, em cada linha, um ou mais valores inteiros separados por um ou mais espaços em branco.
   - Cada linha do arquivo representa uma linha da matriz; o número de colunas é determinado pela quantidade de inteiros em cada linha.

   (c) **Saída esperada:** O programa deve:

   i. Ler o nome do arquivo, abrir o arquivo e interpretar seu conteúdo como uma matriz de inteiros.

   ii. Para cada célula da matriz, verificar todos os vizinhos existentes:
   - mesma linha, colunas anterior e posterior (quando existirem);
   - mesma coluna, linhas anterior e posterior (quando existirem);
   - células diagonais adjacentes (superior esquerda, superior direita, inferior esquerda, inferior direita), quando existirem.

   iii. Para cada célula cujo valor seja estritamente menor que o valor de *todos* os seus vizinhos existentes, imprimir uma linha na forma:

   ```
   Linha X, Coluna Y, Valor = V
   ```

   onde `X` é o número da linha, `Y` é o número da coluna (ambos iniciando em 1), e `V` é o valor armazenado nessa célula.

   (d) **Observações:**

   - A matriz pode ter qualquer dimensão válida (número de linhas $\geq 1$ e número de colunas $\geq 1$).
   - Nos elementos de borda e de canto, apenas os vizinhos que existirem devem ser considerados na comparação.

- Recomenda-se percorrer a matriz utilizando índices de linha e coluna, de forma a facilitar o acesso aos vizinhos.

(e) **Exemplos de teste:**

**Exemplo 1**

**Conteúdo do arquivo** `testeUm.txt`**:**

```
10 20 15
13 18 21
-4 19 92
7  9  5
```

**Entrada:**

```
testeUm.txt
```

**Saída correspondente:**

```
Linha 1, Coluna 1, Valor = 10
Linha 1, Coluna 3, Valor = 15
Linha 3, Coluna 1, Valor = -4
Linha 4, Coluna 3, Valor = 5
```

**Exemplo 2**

**Conteúdo do arquivo** `testeDois.txt`**:**

```
-9 20 15 13 18 21
-4 19 92 7  9  5
```

**Entrada:**

```
testeDois.txt
```

**Saída correspondente:**

```
Linha 1, Coluna 1, Valor = -9
Linha 2, Coluna 4, Valor = 7
Linha 2, Coluna 6, Valor = 5
```

4. **Contagem recursiva de ocorrências e posições em matriz**

(a) **Objetivo:** Implementar um programa em Python, utilizando subprogramações e recursividade, que leia da entrada padrão as dimensões de uma matriz de inteiros, seguido dos seus elementos, e um valor alvo $x$. O programa deve contar, por meio de uma função recursiva, quantas vezes $x$ aparece na matriz e **listar todas as posições** (linha e coluna) onde essas ocorrências acontecem.

(b) **Dados de entrada:**

- Na primeira linha da entrada padrão, dois inteiros positivos $L$ e $C$, representando o número de linhas e colunas da matriz.
- Em seguida, $L$ linhas, cada uma contendo $C$ números inteiros separados por espaço.
- Na última linha, o número inteiro $x$ a ser buscado.

(c) **Saída esperada:** O programa deve:

i. Exibir a matriz lida, linha a linha.

ii. Utilizar uma função recursiva para percorrer toda a matriz e:
   - contar quantas vezes $x$ aparece;
   - registrar as posições onde $x$ ocorre.

iii. Exibir o resultado no formato:

```
x ocorre N vez(es) na matriz
```

seguido da lista de posições, no formato:

```
Linha i, Coluna j
```

onde as posições são numeradas a partir de 1.

(d) **Restrições:**

- A busca deve ser implementada por uma função **recursiva**, percorrendo a matriz por índices.
- Não é permitido utilizar `count()`.

(e) **Exemplo de teste:**

**Entrada:**

```
3 4
1 2 3 4
5 1 1 0
-1 2 1 7
1
```

**Saída correspondente:**

```
Matriz lida:
1 2 3 4
5 1 1 0
-1 2 1 7

1 ocorre 4 vez(es) na matriz
Posições:
Linha 1, Coluna 1
Linha 2, Coluna 2
Linha 2, Coluna 3
Linha 3, Coluna 3
```