Exercícios – Lista 3

1) Faça um algoritmo que leia de um arquivo chamado "**Turma.txt**" o nome dos alunos de uma turma e as suas três notas. Seu algoritmo deve ter como saída um arquivo chamado "**MediasAlunos.txt**" que contém os nomes dos alunos e suas respectivas médias (com duas casas decimais).

- Exemplo de arquivos:

Turma.txt

```
Bruna  8.5  6.7  4.5
Carlos  3.2  5.8  5.9
Fabiana  8.0  9.2  4.1
Igor  6.3  2.1  7.8
Larissa  7.4  9.2  4.4
Marcos  2.7  6.7  8.8
```

MediasAlunos.txt

```
Bruna 6.57
Carlos 4.97
Fabiana 7.10
Igor 5.40
Larissa 7.00
Marcos 6.07
```

- Obs.:

Para declarar uma variável que receba uma palavra:

```
char palavra[50];
```

Para ler uma palavra de um arquivo:

```
fscanf(arqLeitura, "%s", &palavra);
```

Para escrever uma palavra em um arquivo:

```
fprintf(arqEscrita, "%s", palavra);
```

- Utilize as dicas desta observação para a leitura e escrita do nome do aluno.

2) Suponha que cada aluno faça três provas, como mostra cada linha do arquivo "**Notas.txt**". Faça um algoritmo para gerar o arquivo "**Situacao.txt**", onde cada linha contenha a média final do aluno e sua situação: "A" – Aprovado (média igual ou superior a 5.0) ou "R" – Reprovado (média inferior a 5.0).

- Exemplo de arquivos:

Notas.txt

```
7.5   8.5   7.8
8.4   9.2   6.8
9.1  10.0   9.5
4.0   5.2   4.6
5.7   3.4   4.3
4.3   6.0   5.8
```

Situacao.txt

```
7.9   A
8.1   A
9.5   A
4.6   R
4.5   R
5.4   A
```

3) Faça um algoritmo que leia de um arquivo chamado "**Medias.txt**" o nome dos alunos e as suas respectivas médias. Seu algoritmo deve ter como saída um arquivo chamado "**Aprovados.txt**", que contém os nomes dos alunos com média igual ou superior a 7.0, e outro arquivo chamado "**Reprovados.txt**", que contém os nomes dos alunos com média inferior a 7.0.

- Exemplo de arquivos:

Medias.txt

```
Bruna  6.7
Carlos  5.8
Fabiana  8.0
Igor  2.1
Larissa  9.2
Marcos  8.8
```

Aprovados.txt

```
Fabiana  8.0
Larissa  9.2
Marcos  8.8
```

Reprovados.txt

```
Bruna  6.7
Carlos  5.8
Igor  2.1
```

4) Considere um arquivo chamado "**Matriz4x4.txt**" que armazena uma matriz quadrada de ordem 4 com números inteiros. Faça um algoritmo para ler esta matriz do arquivo e criar outro arquivo chamado "**Transposta.txt**" que conterá a matriz transposta da matriz lida.

- **Obs.:** Utilize um **procedimento** para gerar a matriz transposta.

- Exemplo de arquivos:

Matriz4x4.txt

```
6 1 0 4
8 4 9 6
5 8 2 0
0 7 4 2
```

Transposta.txt

```
6 8 5 0
1 4 8 7
0 9 2 4
4 6 0 2
```

- **Obs.:** O seguinte código lê de um arquivo (apontado pelo ponteiro arqLeitura) uma matriz como a apresentada acima:

```
for( i=0; i<4; i++ ){
    for( j=0; j<4; j++ ){
        fscanf(arqLeitura, "%d", &matriz[i][j]);
    }
}
```

Observe que o código lê apenas um número de cada vez, mesmo o arquivo tendo 4 números em cada linha.

5) Considere um arquivo chamado "**Distancias.txt**" que armazena (em cada linha) o nome de uma cidade (sem espaços) e a distância (em km) entre a mesma e Campos dos Goytacazes. Imagine que um usuário deseja saber quantos litros de combustível e quanto custaria abastecer tais litros para ele **chegar a cada uma** das cidades partindo de Campos. Faça um algoritmo para ler (pelo teclado) o consumo do veículo do usuário (em km/l), o valor do litro de combustível e depois gerar um arquivo chamado "**Gastos.txt**" informando (em cada linha) o nome da cidade, quantos litros serão necessários para a viagem e o valor para abastecer tais litros.

- Exemplo de arquivo: **Distancias.txt**

```
SaoJoaodaBarra 40
Vitoria 217
RiodeJaneiro 372
Guarapari 173
```

6) Faça um algoritmo para ler 10 números inteiros a partir de um arquivo chamado "**Numeros.txt**" e depois imprima-os na tela em ordem *crescente*.

- Obs.: Para facilitar, leia os números do arquivo, armazene-os em um vetor e depois ordene este vetor. Seu algoritmo deve ter um **procedimento** para ordenar o vetor e outro para imprimir o vetor antes e depois da ordenação.

- Exemplo de arquivo: **Numeros.txt**

```
7
3
8
11
9
0
10
2
1
4
```

7) Considere um arquivo chamado "**Distancias.txt**" que armazena (em cada linha) o nome de uma cidade (sem espaços) e a distância (em km) entre a mesma e Campos dos Goytacazes. Imagine que um usuário deseja saber quantos litros de combustível e quanto custaria abastecer tais litros para ele **chegar a uma** das cidades partindo de Campos. Assim, faça um algoritmo para ler (pelo teclado) a cidade de destino, o consumo do veículo do usuário (em km/l), o valor do litro de combustível e depois gerar um arquivo chamado "**Gastos.txt**" informando o nome da cidade de destino, quantos litros serão necessários para a viagem e o valor para abastecer tais litros.

- **Obs.:** Utilize a função **strcmp(str1,str2)** da biblioteca **string.h** para comparar os nomes das cidades. Esta função retorna 0 (zero) se **str1** é igual a **str2**.

- **Exemplo de arquivos:**

Distancias.txt

```
SaoJoaodaBarra  40
Vitoria  217
RiodeJaneiro  372
Guarapari  173
```

Gastos.txt

```
Vitoria  21.7litros  R$43.40
```

8) Considere um arquivo de entrada chamado "**Cidades.txt**" que armazena (em cada linha) o nome de uma cidade (sem espaços) e o seu número de habitantes. Faça um algoritmo que leia o arquivo de entrada e gere um arquivo chamado "**Saida.txt**" contendo o nome da cidade mais populosa seguida do seu número de habitantes.

9) Faça um algoritmo para gerar uma matriz 10 x 10 aleatoriamente com números de 0 até 39, com exceção dos elementos da diagonal principal, que devem ser gerados aleatoriamente com números de 1 até 50. A matriz gerada deve ser armazenada no arquivo "**Matriz10x10.txt**".

10) Considere um arquivo chamado "**Matriz4x5.txt**" que armazena uma matriz de tamanho 4 x 5 de números inteiros. Faça um algoritmo para ler essa matriz do arquivo, trocar a primeira linha com a quarta e imprimir a nova matriz no arquivo chamado "**MatrizAlterada.txt**". A matriz que receberá os valores lidos do arquivo deve ser criada na função main( ) e **não deve ser utilizada uma matriz auxiliar** para realizar a troca das linhas.

Utilize três **procedimentos**: um para ler a matriz do arquivo de entrada, outro para realizar a troca das linhas e um terceiro para imprimir no arquivo de saída a matriz alterada.

11) Considere um arquivo chamado "**Matriz4x5.txt**" que armazena uma matriz de tamanho 4 x 5 de números inteiros. Faça um algoritmo para ler essa matriz do arquivo, trocar a segunda coluna com a quinta e imprimir a nova matriz no arquivo chamado "**MatrizAlterada.txt**". A matriz que receberá os valores lidos do arquivo deve ser criada na função main( ) e **não deve ser utilizada uma matriz auxiliar** para realizar a troca das colunas.

Utilize três **procedimentos**: um para ler a matriz do arquivo de entrada, outro para realizar a troca das colunas e um terceiro para imprimir no arquivo de saída a matriz alterada.

12) Considere um arquivo chamado "**Matriz5x6.txt**" que armazena uma matriz de tamanho 5 x 6 de números inteiros. Faça um algoritmo para ler essa matriz do arquivo, realizar a soma dos elementos da segunda, quarta e sexta colunas e imprimir o resultado da soma no arquivo chamado "**Soma.txt**". A matriz que receberá os valores lidos do arquivo deve ser criada na função main( ).

Utilize uma **função** para realizar o cálculo da soma (retornando esse valor para a função main( )) e dois **procedimentos**: um para ler a matriz do arquivo de entrada e outro para imprimir no arquivo de saída o resultado da soma.

13) Considere um arquivo chamado "**Cidades.txt**" que armazena em cada linha o nome de uma cidade e o seu número de habitantes. Faça um algoritmo para ler esse arquivo e criar um arquivo chamado "**Populacao.txt**" contendo o nome da cidade mais populosa seguida pelo seu número de habitantes.

- Obs.: A função **strcpy(str1,str2)** da biblioteca **string.h** copia uma string para outra. Ela copia a string **str2** para a string **str1**.

14) Considere um arquivo chamado "**Pessoas.txt**" que armazena em cada linha o nome de uma pessoa e o seu ano de nascimento. Faça um algoritmo para ler esse arquivo e criar dois outros arquivos: um chamado "**Maiores.txt**", que contém em cada linha o nome e a idade das pessoas maiores de idade (idade $\geq$ 18 anos) e outro chamado "**Menores.txt**", que contém em cada linha o nome e a idade das pessoas menores de idade (idade < 18 anos).

15) Faça um algoritmo para imprimir o número de palavras (de no máximo 20 caracteres) presentes no arquivo "**Texto.txt**". Considere que o arquivo não possui qualquer número, somente palavras.

16) Considere um arquivo chamado "**Palavras.txt**" que armazena em cada linha uma palavra de no máximo 15 caracteres. Faça um algoritmo para ler pelo teclado uma palavra (também de no máximo 15 caracteres) e imprimir o número de vezes que essa palavra aparece no arquivo.

- Obs.: Utilize a função **strcmp(str1,str2)** da biblioteca **string.h** para comparar as palavras. Esta função retorna 0 (zero) se **str1** é igual a **str2**.