

Tema: Recuperação 02

Atividade: Arquivos, arranjos e matrizes

INSTRUÇÕES:

- Os exercícios extras são opcionais, mas recomendados para todos.

.

O objetivo é rever aplicações de métodos, arquivos e grupos de dados.

- Desenvolver um método para cada enunciado abaixo.

- Cada método deverá conter, em seu cabeçalho, como comentário

(`/** e */`), a documentação essencial:

nome e matrícula,

identificação, objetivo, parâmetros e condições especiais,

se houver, e relatório de testes (exemplos de valores usados e condições testadas).

SUGESTÃO: Montar um menu para a escolha do método a ser testado
(ver modelo em Lista00.cpp).

Testes deverão ser realizados e os valores usados deverão
ser guardados no final do programa como comentários (`/* e */`).

O uso de recursão é opcional; se desejar utilizá-lo,

fazer também a implementação da forma não-recursiva.

Os métodos devem, preferencialmente, usar os modelos
bibliotecas `array.h` e `matrix.h`.

Para os exercícios abaixo considera a ampliação das bibliotecas
para arranjos (`array.h`) e matrizes (`matrix.h`) definidas com estruturas (**`struct`**).

01.) FAZER um programa para:

- definir uma função para determinar se os valores
no arranjo estão em ordem decrescente;

- ler arranjo do arquivo DADOS1.TXT;

- se não estiverem, colocá-los em ordem decrescente,
antes de regravar os dados no arquivo DECRESCENTE.TXT,
colocando a quantidade de elementos na primeira linha do arquivo.

DICA: Para ordenar usar o princípio de troca dos elementos vizinhos
que não estiverem ordenados, e testar novamente usando a função,
até que não seja necessário fazer mais trocas.

02.) FAZER um programa para:

- definir uma função para receber o arranjo como parâmetro e inverter a ordem de seus elementos;
 - ler arranjo do arquivo DECRESCENTE.TXT;
 - gravar o arranjo invertido no arquivo INVERTIDOS.TXT, colocando a quantidade de dados elementos na primeira linha.
- DICA: Trocar o último com o primeiro e prosseguir até a metade.

Exemplo:

```
// arranjo1 = { 4, 3, 2, 1 };  
arranjo_inverter ( arranjo2, arranjo1 ); // [ 1, 2, 3, 4 ]
```

03.) FAZER um programa para:

- definir uma função para receber um arranjo e um valor inteiro como parâmetros, e achar o maior valor que não ultrapasse o segundo parâmetro;
 - ler arranjo do arquivo DADOS1.TXT;
 - receber e mostrar o segundo maior valor usando a função definida;
- DICA: Usar o primeiro valor do arranjo como referência inicial para o valor inicial do maior de todos, quando o procurar por outra função.

Exemplo:

```
arranjo_ler ( "DADOS1.TXT", tabela );  
valor = arranjo_SegundoMaior ( tabela, arranjo_maiorDeTodos( tabela ) );  
tela <- valor
```

04.) FAZER um programa para:

- ler um arranjo do arquivo DADOS1.TXT;
- ler outro arranjo do arquivo DADOS2.TXT;
- filtrar e mostrar os elementos comuns aos dois arranjos, sem repetições;
- gravar o resultado no arquivo FILTRADOS.TXT, colocando a quantidade de dados únicos na primeira linha.

Exemplo:

```
arranjo_ler ( "DADOS1.TXT", arranjo_1 );  
arranjo_ler ( "DADOS2.TXT", arranjo_2 );  
arranjo_filtrar ( arranjo3, arranjo1, arranjo2 );  
arranjo_gravar ( "FILTRADOS.TXT", arranjo3 );
```

05.) FAZER um programa para:

- ler cadeias de caracteres do arquivo BINARIOS1.TXT;
- considerar válidos apenas os valores iguais a zero ou a um;
- converter e armazenar em um arranjo de inteiros (**int**);
- supondo serem dígitos de um número binário, convertê-los para decimal mediante o uso de uma função.

Exemplo:

```
// binario = { 1, 0, 1, 1 } // o último estará na menor potência  
int x = arranjo_paraDecimal ( binario ); // x = 11
```

06.) FAZER um programa para:

- ler dados para matrizes do arquivo MATRIZ1.TXT;
- ler um número inteiro (N) para indicar a quantidade de linhas e colunas de uma matriz quadrada;
- montar, mostrar e gravar no arquivo MATRIZ2.TXT uma matriz com a característica abaixo (tridiagonal crescente).

Exemplo:

```
1 2 0 0
3 4 5 0
0 6 7 8
0 0 9 10
```

07.) FAZER um programa para:

- ler dados para matrizes do arquivo MATRIZ1.TXT;
- ler um número inteiro (N) para indicar a quantidade de linhas e colunas de uma matriz quadrada;
- montar, mostrar e gravar no arquivo MATRIZ2.TXT uma matriz com a característica abaixo (tridiagonal decrescente).

Exemplo:

```
10 8 0 0
9 7 5 0
0 6 4 2
0 0 3 1
```

08.) FAZER um programa para:

- ler matriz do arquivo MATRIZ3.TXT;
- definir uma função lógica para verificar se a matriz lida apresenta a característica abaixo (potências por linha).

Exemplo:

```
1 1 1 1
1 2 4 8
1 3 9 27
1 4 16 64
```

09.) FAZER um programa para:

- ler matriz do arquivo MATRIZ4.TXT;
- definir uma função lógica para verificar e testar se a matriz lida apresenta a característica abaixo (potências por coluna).

Exemplo:

1	1	1	1
1	2	3	4
1	4	9	16
1	8	27	64

10.) FAZER um programa para:

- ler do arquivo DADOS3.TXT:
 - um número inteiro (N) para indicar a quantidade de supermercados cujos preços de produtos serão avaliados;
 - o nome e o código (**int**) de cada supermercado;
 - ler o preço de um produto de cada supermercado;
- calcular o preço médio desse produto;
- informar pelo menos dois supermercados com preços inferiores à média.