**UFF – Universidade Federal Fluminense**
**TIC – Instituto de Computação**
**TCC – Departamento de Ciência da Computação**

**TCC 00.309 | Programação de Computadores II | Turma A-1 | 2016.2**
**Professor: Leandro A. F. Fernandes**

Lista de Exercícios Sobre

# Arquivos

## Questão 1

Escreve dois programas. O primeiro pede para o usuário informar os dados de um aluno (matrícula, nome e CR, dos tipos inteiro, texto e real, respectivamente) via entrada padrão e salva esses dados em um arquivo texto. Cada registro de aluno ocupará 3 linhas. O usuário sinaliza o fim da leitura informando matrícula igual a 0. O segundo programa abre o arquivo texto previamente salvo, lê as informações e escreve o resultado da leitura na saída padrão.

## Questão 2

Escreve dois programas. O primeiro pede para o usuário informar os dados de um aluno (matrícula, nome e CR, dos tipos inteiro, texto e real, respectivamente) via entrada padrão e salva esses dados em um arquivo binário sequencial. O segundo programa abre o arquivo binário previamente salvo, lê as informações e escreve o resultado da leitura na saída padrão.

Utilize os programas desenvolvido nas questões 1 e 2 e dê como entrada os dados dos mesmos alunos. Feito isso, compare o tamanho dos arquivos gerados. Compare, também, o conteúdo dos arquivos após abri-los em algum editor de texto simples, como o Bloco de Notas do Windows.

## Questão 3

Escreva um programa que lê o conteúdo de um arquivo binário e copie para outro, byte a byte. Porém, implemente o programa de tal modo que seja utilizada *buffering* na escrita. Ao rodar seu programa, certifique-se de ter escolhido um arquivo vem grande para ser copiado (por exemplo, o ISO de um CD), e defina um buffer relativamente pequeno (por exemplo, 10 bytes). Durante a execução, utilize o modo debug do NetBeans para verificar que o arquivo de destino aumentará de tamanho apenas a cada dez iterações do laço principal em sua implementação. Por quê?

## Questão 4

Implemente um programa que gere uma versão criptografada de um arquivo texto trocando cada caractere de código ASCII $j$ pelo caractere de código ASCII $j + k$, onde $k$ é um parâmetro especificado pelo usuário. Fique atento para não gerar códigos ASCII fora da faixa permitida. Como sugestão, utilize como entrada o arquivo gerado na questão 1.

Utilize o arquivo fornecido pelo professor como exemplo de entrada para seu programa.

## Questão 5

Escrever um programa que aceita a seguinte sintaxe em sua invocação:

```
java -jar SeuPrograma -compactar <arquivo_origem> <arquivo_destino>

java -jar SeuPrograma -descompactar <arquivo_origem> <arquivo_destino>
```

Quando a primeira linha de comando é usada, seu programa deverá utilizar fluxo filtrado de compressão para compactar o arquivo indicado em `<arquivo_origem>` e salvá-lo em `<arquivo_destino>`. Com a segunda linha, seu programa fará o processo inverso: receberá como entrada o arquivo indicado em `<arquivo_origem>`, descompactará seu conteúdo e salvará o resultado em `<arquivo_destino>`. Teste com arquivos de texto e imagem presentes em seu computador para verificar se a compressão e descompressão funciona corretamente.

## Questão 6

Considere dois arquivos, “itens1.dat” e itens2.dat”, contendo registros sobre itens de estoque de um supermercado. Cada registro contém o nome do produto, preço, marca e data de validade. Em ambos os arquivos os registros estão ordenados pelo nome do produto. Escreva um programa que leia os dois arquivos e gere um terceiro formado pela combinação dos dois anteriores de modo que os registros continuem ordenados pela chave nome. Os arquivos não devem ser lidos na íntegra para a memória principal. Sua tarefa é fazer tal programa de modo que no máximo um registro de cada arquivo esteja na memória durante sua execução.

Resolva esse exercício de duas maneiras: na primeira o arquivo será textual. Na segunda, o conteúdo será binário. Os arquivos de entrada deverão ser escritos por programas implementados pelo aluno para este fim.

## Questão 7

Escreva um programa que seja capaz de ler uma imagem de dimensões 256x256 em formato Raw (arquivo binário contendo apenas uma sequência de bytes indicando um tom de cinza entre 0 e 255) e gere seu histograma. O histograma é um mapa que associa a cada tom o número de suas ocorrências na imagem.

Utilize o arquivo fornecido pelo professor como exemplo de entrada para seu programa.

## Questão 8

Listas lineares sequenciais são fascinantes, pois permitem acesso randômico a qualquer elemento armazenado. Assumindo que você terá uma coleção de elementos compostos por um valor inteiro e dois valores reais, implemente uma classe capaz de gerenciar uma lista linear sequencial completamente contida em um arquivo de acesso randômico. Ou seja, nesta classe, os elementos não poderão estar armazenados na memória principal. Em programação, essa estratégia de gerência de memória é conhecida como *Out-of-Core*.

## Questão 9 ** Só para quem se garante!

Escreva uma classe especializada em carregar arquivos “.tga”, um formato bastante popular para armazenamento de texturas em jogos de videogame antigos. Estude o formato a partir de sua especificação (disponível em http://www.gamers.org/dEngine/quake3/TGA.txt), porém, restrinja sua implementação para "Color map type" igual a 0 (*image file contains no color map*), *Image type* igual a 2 (*uncompressed true-color image*) e 24 bits de precisão por pixel no espaço de cores RGB. Uma vez carregada a imagem, salve-a em três arquivos, um para cada canal R, G e B, com formato Raw (arquivo binário contendo apenas uma sequência de bytes indicando um tom entre 0 e 255 de cada canal). Utilize sua ferramenta de edição de imagens predileta para verificar se os arquivos Raw gerados estão corretos.