Relatório Trabalho prático ED:

Disciplina: Estruturas de Dados (GAC - 108)
Alunos: Cesar Augusto Pires e Gabriel Aguiar Alves e Silva
Universidade Federal de Lavras - UFLA, 28 de maio de 2023.

Sumário:

Introdução:

A primeira parte do trabalho prático de ED consiste em:

- Realizar a leitura de um banco de dados em CSV, cujos campos estão separados por “,”, e composto por 14 colunas;
- Geração um arquivo binário com os dados do arquivo CSV, para que possamos fazer as manipulações;
- Após isso, deveríamos fazer a leitura do arquivo binário e algumas manipulações diretamente no arquivo binário, sendo elas:
    - Adição de elementos no arquivo em uma posição específica escolhida pelo usuário;
    - Visualizar os registros entre duas posições específicas também definidas pelo usuário;
    - Alterar os dados de um registo em uma posição específica a ser definida pelo usuário;
    - Trocar dois registros de posição, sendo as posições informadas pelo usuário;
    - Imprimir todos os registros do arquivo binário na ordem em que estão armazenados;
- Para além disso, o grupo também grupo uma função responsável por gerar um arquivo “.txt” com todos os dados do arquivo binário, sendo essa uma funcionalidade extra do programa.

Desenvolvimento:

Para o desenvolvimento do programa seguimos alguns conceitos da programação orientada a objetos, de modo que a construção do programa ficou da seguinte forma:

- Utilizamos o arquivo ***‘main.cpp’*** para declaração das variáveis principais, chamada da função ***‘conversor.h’*** e chamada da função ***‘menu.h’***;
- ***conversor.h*** é a parte do programa responsável por realizar a conversão do arquivo CSV para binário, por meio da classe conversor e suas funções:
    - **iniciarProcesso()** serve somente para mostrar que o processo de conversão foi iniciado, ou não (imprimindo uma mensagem ao usuário,

caso haja algum erro na hora de abrir o arquivo), nesse caso chamando a função **finalizarProcesso()**;

- **finalizarProcesso()** função existente apenas para fechar os arquivos de leitura e escrita;
- **processar()** Primeiramente, inicia um processo necessário para a conversão. Em seguida, lê cada linha de um arquivo de entrada. Para cada linha, realiza o processamento utilizando um objeto chamado ProcessadorLinha. Durante o processamento, são geradas estatísticas de transferência de propriedades, as quais são armazenadas em uma estrutura apropriada. Essas estatísticas são então escritas no arquivo de saída. O contador de linhas é incrementado a cada processamento de linha. Durante a execução, a função imprime um ponto no console para indicar o progresso do processamento.

- ***menu.h*** é responsável por fazer fazer a chamada de todas as funções que são realizadas pelo programa;
- ***operacoes.h*** arquivo que contém as funções responsáveis por realizar todas as operações requisitadas para o projeto:
  - **operacoes()** essa é a função responsável por abrir os arquivos tanto de entrada, quanto de saída;
  - **imprimir()** responsável por chamar a função que irá imprimir os campos do arquivo para o usuário. É chamada a função ***'imprimirNaTela'*** do arquivo **PropertyTransferStatisticsClass.h**, que realiza o procedimento de mostrar os campos;
  - **imprimirNoArquivo()** função responsável por fazer a impressão dos dados do arquivo binário no arquivo ".txt". A função ***'imprimirNoArquivo'*** que realiza esse procedimento também está presente no arquivo **PropertyTransferStatisticsClass.h**.
  - **busca()** essa função é responsável por fazer a impressão de uma posição específica do arquivo, utilizando uma função anterior **imprimir();**
  - **busca(*parâmetros*)** essa função tem o intuito de, na impressão dos dados em um intervalo especificado pelo usuário, buscar os campos

que correspondem às posições informadas. Também utiliza a função **imprimir()**;

- **trocaPosicao()** é a função que irá realizar a troca dos valores de duas posições, informadas pelo usuário, dentro do arquivo com o uso de uma variável auxiliar. Após a troca, é feita a reescrita das mesmas posições dentro do arquivo binário;
- **insereNaPosicao()** essa função é responsável por fazer a adição de um novo registro em uma posição qualquer, fornecida pelo usuário, através da chamada de ***'novoDado()'***;
- **alterarResgistroPosicao()** essa função é responsável por fazer a substituição dos campos, através da chamada da função ***'novoDado()'*** em uma posição específica, determinada pelo usuário, por valores que serão informados pelo mesmo;
- **novoDado()** essa função, através da chamada da função ***'novaPropertyTransferStatistics()'*** do arquivo **propertyTransferStatisticsClass.h,** solicitar os valores de cada campo do registro, que serão inseridos no campo especificado anteriormente, ao usuário;
- **transformaEmTxt()** essa é a função que é responsável por salvar os dados do arquivo binário em um arquivo txt. A gravação dos dados é feita por meio da chamada da função ***'imprimirNoArquivo()'*** citada anteriormente.

- ***processadorLinha.h*** esse é o arquivo que contém as funções responsáveis pelo tratamento do arquivo CSV, bem como a separação correta dos campos e suas atribuições as respectivas variáveis:
  - **tratarLinha()** é a função responsável por fazer a trativa da linha do arquivo, separando o campo analisado do restante da linha, tenho a "," como parâmetro para a separação;
  - **tratarPorcentagem()** função responsável por fazer a tratativa dos valores que são representados em porcentagem dentro do arquivo, tendo como parâmetro o "%";
  - **processarLinha()** é a função principal do arquivo, a qual é responsável por atribuir os campos lidos do arquivo a suas respectivas variáveis;

- **PropertyTransferStatisticsClass.h** o referido arquivo possui apenas 3 funções:
    - **novaPropertyTransferStatistics()** função que é responsável por obter os dados do usuário que serão utilizados nas funções ***'alterarResgistroPosicao()'*** e ***'insereNaPosicao()'***;
    - **imprimirNaTela()** é a função que é responsável pela impressão dos valores de cada campo na tela de execução do programa, sendo os mesmo separados por "|" que foi o parâmetro de separação definido pelo grupo. Essa função é utilizada na função ***'imprimir()'***;
    - **imprimirNoArquivo()** responsável por fazer a gravação campo a campo, utilizando o parâmetro de separação "|" (definido pelo grupo) entre os mesmos, no arquivo txt que é gerado através da função '***transformaEmTxt()'***;
- **PropertyTransferStatisticsStruct.h** nesse arquivo é onde foi criada a struct que irá armazenar todos os campos lidos do arquivo CSV. Para a declaração das 14 colunas o grupo optou por fazer da seguinte forma:
    - O primeiro campo **'id'** do tipo **int**, que é o campo responsável por pegar a posição da linha dentro do arquivo;
    - Os campos presentes no arquivo são: **'seriesReference', 'period', 'dataValue', 'status', 'units', 'magnitude', 'subject', 'periodicity', 'group', 'seriesTitle1', 'seriesTitle2', 'seriesTitle3', 'seriesTitle4'** e **'seriesTitle5'**. Ambos foram declarados como **vetor** de **char** com número de posições limitados em **256 caracteres**, pois são campos que possuem valores como datas, nome de cidades, identificação de série que contém uma sequência alfa-numérica de caracteres, etc. Devido a isso, optamos por utilizar um vetor de caracteres para os campos presentes.

Execução do Programa

Para executar o programa basta certificar que tenha o arquivo PropertyTransferStatistics.csv no mesmo diretório do arquivo main e dos outros arquivos .h.

Linux

Para executar o programa no linux, após a compilação do programa basta executar no terminal o seguinte comando:

```
Unset
./main
```

Já no windows, basta executar o .exe gerado após a compilação

***Conclusão*:**

O primeira parte do projeto prático desenvolvido pelo grupo, além de atender todas as solicitações exigidas pelos docentes da disciplina, ainda apresentou um bônus que é a possibilidade de criação de um arquivo **txt** através da conversão do arquivo binário. Além de utilizar diversos conceitos aprendidos em sala de aula ao longo da disciplina, como a criação de classes. Houve também a apresentação de um conteúdo extra-curricular que é a separação do programa em arquivos de extensão **“.h”**, responsáveis por deixar a execução do programa ainda mais legível e estruturada.