BACHARELADO EM CIENCIA DA COMPUTAÇÃO

ENZO VELO

LUCAS GOMES COLOMBO

RAFAEL BRUNINI pereira

RELATORIO DA PRIMEIRA ETAPA DO PROJETO PRÁTICO DA DISCIPLINA ESTRUTURA DE DADOS

Manipulação de arquivos binários

LAVRAS – MG

2022

ENZO VELO

LUCAS GOMES COLOMBO

RAFAEL BRUNINI

RELATORIO DA PRIMEIRA ETAPA DO PROJETO PRÁTICO DA DISCIPLINA ESTRUTURA DE DADOS

Manipulação de arquivos binários

Projeto prático apresentado à Universidade federal de Lavras, como requisito para a aprovação na matéria estrutura de dados

Professor: Renato Ramos da Silva

LAVRAS – MG

2022

Sumário

1. INTRODUÇÃO

Aqui será apresentada a primeira etapa do projeto prático da disciplina Estrutura de Dados, no qual foram feitos códigos para a conversão de um arquivo csv para binário, e códigos para a realizar a manipulação deste mesmo arquivo binário.

No decorrer deste trabalho serão apresentados os objetivos, a metodologia utilizada pelos integrantes do grupo, o funcionamento e bibliotecas do código, e as conclusões acerca do trabalho.

2. OBJETIVOS

A ideia principal do projeto foi criar um sistema que realizasse a conversão de um arquivo no formato csv para o formato binário, e ainda permitisse a alteração e verificação desses dados dentro do arquivo binário.

O trabalho tem como objetivo principal a aplicação pratica dos assuntos abordados na matéria até o momento, assim como estimular a criatividade e incentivar o uso de ferramentas que ampliem o conhecimento do aluno.

3. METODOLOGIA

Na seção serão apresentados todos os métodos e ferramentas utilizadas neste trabalho. Começaremos apresentando as estruturas de dados e bibliotecas utilizadas, e por fim o funcionamento do sistema.

3.1. Bibliotecas

Além da biblioteca iostream, foram utilizadas as bibliotecas fstream para facilitar a manipulação de arquivos (tanto de texto quanto binário), a string para facilitar a manipulação de vetores de char, a cstring para manipulação de regiões na memória e também para a manipulação de vetores de char.

3.2. Funcionamento

No programa a função main será responsável exclusivamente por permitir a entrada de dados feitas pelo usuário, e por consequência realizará todas as chamadas das funções presentes no programa, sendo elas as funções de ler todos os registros, ler todos os registros entre duas posições, alterar a posição de dois dados, inserir um dado em uma posição especifica, alterar os dados de uma posição e a de deletar um dado do programa.

Sempre que uma função for chamada no main ela será executada no arquivo “funcçoes.cpp” utilizando como parâmetros os dados inseridos na main, além dos dados do arquivo “funções.hpp” que contem a struct com todos os dados do arquivo csv.

3.2.1. funções

Ao iniciar o código será pedido ao usuário que o mesmo insira uma opção válida, são estas: 1, 2, 3, 4, 5, 6, 7.

Ao selecionar a opção **1** de **ler registros**, a função será chamada na main, e executada no arquivo “funções.cpp” e dessa forma o programa irá exibir todos os elementos do arquivo. Para que isso ocorra há um vetor de char de uma única posição

que irá percorrer todo o arquivo e exibir um elemento de cada vez no terminal, trazendo apenas um dado por vez para a memória principal.

Se a opção escolhida for a **2** de **ler entre duas posições específicas,** o código irá pedir ao usuário que insira entre quais posições ele deseja visualizar os dados e logo depois chamará a função “alterarPos” que será executada no arquivo “funções.cpp”. Dentro desse arquivo, o código iniciará um laço de repetição que começará na primeira posição inserida e percorrerá todos os elementos através de um vetor de char que exibirá cada um dos dados pelos quais ele passar, até que ele chegue na condição de parada (segunda posição inserida).

Quando a opção **3** de **alterar posição** for selecionada, o usuário deverá digitar as posições dos dados deseja trocar, e após fazer isso a função de “alterarPos” será chamada e executada no arquivo “funções.cpp”. Nela serão criadas structs auxiliares que receberão os valores das posições digitadas, e logo depois terão seus valores trocados, por fim os dados serão gravados de volta na struct principal.

Caso a opção **4** de **inserção** seja escolhida, será pedido ao usuário para escolher qual posição ele deseja adicionar esse dado. Após a posição ser selecionada o usuário ainda devera digitar os novos dados a serem adicionados no código, sendo esses dados os mesmos referentes a cada campo da struct, e logo após isso a função “inserirNovoRegistro” será chamada.

Para fazer a inserção cada dado será passado através de uma struct auxiliar para a função de inserção, que terá um laço para percorrer o arquivo de trás para frente copiando todos os dados da esquerda para direita afim de “abrir” mais um espaço na struct principal. Quando o laço chegar ao fim (encontrar a posição de inserção) os novos dados digitados pelo usuário serão inseridos e todos os outros dados terão sido passados para frente.

Em caso de a opção **5** de **alterar dados** for escolhida**,** será pedido ao usuário que digite em qual posição ele deseja alterar os dados, após isso o usuário ainda devera digitar os novos dados a serem adicionados no código, sendo esses dados os mesmos referentes a cada campo da struct, quando o usuário terminar a inserção a função “alterarDados” será chamada e executada no arquivo “funções.cpp”. Nesse arquivo será criado uma struct auxiliar para exibir os dados da opção onde ocorrerá a troca de dados, e logo após isso esses dados serão trocados pelos digitados pelo usuário e novamente exibidos no terminal.

Para a opção **6** de “**deletar”** foi atribuído ao final de cada struct um valor lógico que indicará se esse dado é ou não válido no momento, quando esse valor estiver como verdadeiro o dado será exibido sem nenhum problema, mas caso algum dado tenha sido “deletado”, ao tentar exibi-lo o usuário recebera uma mensagem dizendo que o dado foi deletado.

Quando o usuário quiser deletar algum dado ele devera digitar a posição que ele deseja deletar o dado, após isso a função “deletar” será chamada e executada no arquivo “funções.cpp”. Nessa função o dado receberá o valor lógico para indicar a exclusão do dado.

É importante lembrar que apesar do nome da função, ela não deleta os dados do programa, ela apenas faz a exclusão lógica dos mesmos através da struct.

Ao final de todas as opções o usuário será direcionado de volta para o menu até que decida sair apertando a opção **7** de **sair do programa**.

Se qualquer outra opção for selecionada uma mensagem de erro será exibida, e o usuário será enviado de volta para o menu.

3.2.2. makefile

O makefile é basicamente um conjunto de instruções para o compilador dizendo a ordem de como cada coisa deve ser executada

Foi realizado um arquivo makefile para facilitar e otimizar o código, tornando-o mais eficiente e prático para o uso geral.

4. Conclusão

O trabalho engloba assuntos relacionados às principais matérias do curso de ciência da computação tidas até o momento.

A necessidade de converter um arquivo para o formato binário e depois manipula-lo sem trazer todos os dados para a memoria principal motivou a busca por diversos conhecimentos que eram necessários para a resolução do problema, dentre eles o aprofundamento do estudo sobre manipulação de arquivos de forma mais eficiente e prática, a utilização do método “get” para o tratamento de virgular e quebras de linha no meio do arquivo, além de diversos outros estudos para o melhoramento do código como o makefile.

Por fim, a criação e implementação do projeto foi bem sucedida. Toda a conversão de um arquivo csv para binário se mostrou eficiente, todas as funções de manipulação se mostraram altamente funcionais, a conversação entre os 3 códigos feitos (o “main.cpp”, o “funções.cpp”, e o “funções.hpp”), ocorreu de forma bem prática, e a implementação do makefile tornou tudo ainda mais eficiente.