UNIVERSIDADE FEDERAL DE LAVRAS – UFLA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DAVI HERMÓGENES SIQUEIRA

GUSTAVO SOARES SILVA

RAFAEL FURTADO MORAIS

RELATÓRIO DO PROJETO PRÁTICO

ETAPA 1

ESTRUTURA DE DADOS

MINISTRADA PELOS DOCENTES

RENATO RAMOS DA SILVA E

JOAQUIM QUINTEIRO UCHOA

Lavras – MG

2022

O seguinte relatório tem como objetivo esclarecer os métodos utilizados no desenvolvimento da primeira etapa do projeto prático, bem como descrever as estruturas utilizadas e a lógica geral do programa.

Relatório:

O projeto foi desenvolvido utilizando três arquivos, um somente para converter a base sorteada para binário, outro para manipulação do arquivo binário, e um cabeçalho (header) que contém a estrutura com os seis campos contidos no arquivo CSV juntamente com uma função para visualizar os registros.

Conversão:

No geral, o sistema de conversão lê um carácter por vez e o concatena em uma string, e dependendo de em qual campo esteja dentro do switch, ele irá converter os dados concatenados na string para o tipo especifico de dado e armazenar no registro e, no último campo, ele converte o último dado e o grava no arquivo binário.

Detalhadamente, a conversão funciona da seguinte maneira, é definido uma variável booleana para verificar se foi lida alguma aspa, e caso o seja, a variável ‘aspasDuplas’ que inicia seu valor em verdadeiro, é alterada para falso e somente retorna para verdadeiro quando ler outra aspa. Isso se fez necessário pois havia virgula entre aspas dentro da base sorteada, e para o sistema funcionar corretamente a virgula teria de ser considerada apenas fora das aspas para representar uma mudança de campo, visto que dentro das aspas ela se constitui apenas parte da frase.

Após entrar na condição para mudança de campo, o valor da variável ‘campo’ é aumentado em um, e dependendo de seu valor ela entrará dentro de uma condição (case), que converterá a string concatenada para

o tipo de dado reservado para aquele campo. No último campo ele irá gravar o registro no arquivo binário. Por fim, a string com os valores concatenados é esvaziada, para que possa ler os próximos valores.

O arquivo "csvToBin.cpp" deve ser O PRIMEIRO a ser executado para que seja gerado o arquivo binário que será manipulado pelo programa "manipularBinario.cpp". Caso ocorra algum erro em relação a ter mais de uma função main() basta comentar a função main() do arquivo que não está sendo executado.

Funções principais e auxiliares:

Todos os métodos para manipulação estão contidos no arquivo "manipularBinario.cpp". Houve uma tentativa de separar as funções em um outro arquivo, mas isso se mostrou inviável, o que acabou nos forçando a implementa-las no arquivo principal.

Adicionar:

A função **adicionar** recebe como parâmetro a posição em que gostaria de inserir o novo elemento. Após receber o novo elemento, todos os registros até a posição desejada são deslocados para frente. Isso foi implementado utilizando uma variável auxiliar, que recebe o último elemento do arquivo e o regrava uma posição à frente. Isso é feito até a posição desejada. Nesta posição (posição a ser inserido) terá um elemento duplicado. O elemento a ser adicionado é gravado onde o primeiro duplo se encontra.

Imprimir em partes:

A função **imprimir em partes** recebe dois parâmetros, ponto inicial e final, e, utilizando essas informações, a função posiciona o arquivo no ponto inicial (o arquivo é posicionado corretamente porque o ponto inicial é multiplicado pelo tamanho do registro). Utilizando o início do arquivo como referência, é implementada uma repetição contada para iterar do elemento inicial ao final. Cada registro individual é lido e passado para a variável do tipo registro denominada ‘linha’, que em seguida chama a função contida na estrutura (registro), denominada ‘mostrarCampos’.

Imprimir tudo:

A função **imprimir tudo** mostra todos os campos a partir do início.

Inverter posições:

A função de **inverter** faz uso de duas variáveis auxiliares. Os valores das posições a serem invertidas são passados para as variáveis, e regravados invertendo as posições.

Alterar:

A função de **alterar** simplesmente grava o registro do usuário na posição desejada, sobrescrevendo o registro anterior.

Funcionamento das funções auxiliares:

O projeto faz uso de três funções auxiliares, são elas número Elementos, novo Elemento e Limitar entrada.

nElementos:

As funções **nElementos** simplesmente retornam a quantidade de registros presentes no arquivo binário e foram implementadas utilizando sobrecarga de métodos, para que seja possível a utilização tanto com fstream, ifstream e sem parâmetro quando a função é chamada sem ter um arquivo aberto.

Novo Elemento:

A função **novoElemento** foi implementada separadamente para fins de organização, e permite que o usuário insira os campos do registro. O campo ‘id’ é recebido por parâmetro porque, caso essa função tenha sido chamada pela função **alterar**, o ‘id’ é mantido, e caso tenha sido chamada pela função **inserir**, o ‘id’ recebe o número de registros do arquivo. Foi empregado um método para evitar inputs errados e além do limite predefinido de caracteres.

Limitar Entrada:

As funções **limitarEntrada** foi sobrecarregada para inteiros e vetor de char. Para o vetor de char, é feito um 'getline()' para uma variável string auxiliar e verifica se ela excede a quantidade programada, e se o fizer, irá aparecer uma mensagem pedindo para digitar novamente até que o tamanho seja menor que o programado.

Já a sobrecarga para o inteiro é garantir que o usuário não digite um valor inválido para as posições, como por exemplo números negativos, letras ou um número acima do número de posições do arquivo.

Menu:

O menu é autoexplicativo, simplesmente pega a opção do usuário e, dependendo de qual for, chamará um dos métodos implementados.

Esclarecimentos e conclusão:

Nossa abordagem para a organização do código estava em dividir em quatro grandes blocos. O primeiro arquivo contém a conversão do CSV para binário, o segundo seria um arquivo de implementação de todas as funções existentes, terceiro o arquivo principal (main) contendo menu e construção de todo o necessário para a interação do usuário e por fim o arquivo cabeçalho que uniria o bloco das funções e suas chamadas na função principal, baseando-se na ideia de esconder a implementação do usuário e deixá-lo preocupado apenas com a utilização das funções (similar com o que acontece nas bibliotecas).

Porém, ao executar nosso trabalho em outros ambientes de desenvolvimento, nos deparamos com o fato de que estes não compilavam os quatro arquivos em conjunto estragando todo trabalho. Após pesquisas descobrimos que, onde estávamos desenvolvendo, o arquivo *makefile* era criado automaticamente a parte para integração, o

que não acontecia em outros ambientes. Houve uma tentativa de criar um arquivo *makefile* que fizesse tal integração, porém não foi possível implementar, tanto pela falta conhecimento quanto pelo prazo. Nossa solução foi jogar todas as funções dentro do arquivo principal, deixando o cabeçalho apenas por definir a estrutura e a função de mostrar registros.

No início do desenvolvimento do projeto, houve uma tentativa de se fazer uso de vetores dinâmicos de carácteres (char), para que não houvesse espaço desnecessário com vetores estáticos. Contudo, não é possível armazenar variável dinâmica em arquivo tipado. Por conta disso a implementação de vetor estático foi necessária.

Considerações finais:

Por fim, o processo de desenvolvimento da primeira parte do projeto foi de bastante aprendizado. Ao longo do projeto foram necessárias várias mudanças e pesquisas. Novos problemas foram surgindo, nos obrigando a alterar o código a fim de lidar com as demandas da nova situação. Ao nosso ver o código ficou enxuto, porém um pouco extenso, visto que criamos condições para lidar com possíveis erros que o usuário possa vir a cometer, um exemplo disso são as funções de limitar entrada.