Universidade Federal de Lavras

Instituto de Ciências Exatas e Tecnológicas.

Departamento de Ciência da Computação

Disciplina: GAC108 - Estrutura de Dados

Relatório: Trabalho Prático.

Tratamento de Arquivos e Ordenação em Memória Secundária

Discentes:	Marcos Carvalho Ferreira -	202010202 -	Turma: 14B
	Willian Brandão de Souza -	202220233 -	Turma 14A

Lavras - MG

Data de entrega: 02 de Julho de 2023

SUMÁRIO:

Introdução

O trabalho em questão tem como objetivo desenvolver um sistema para a conversão de arquivos CSV em formato binário, bem como a implementação de operações de leitura e tratamento desses arquivos binários. Ademais, pretende-se explorar técnicas de ordenação em memória secundária para processar e classificar os dados armazenados nos arquivos binários. A base de dados sorteada para o desenvolvimento das aplicações foi: international-trade-june-2022-quarter-csv.

Durante a **primeira etapa** do trabalho foram desenvolvidas duas aplicações: em que o primeiro é responsável pela conversão do arquivo CSV para binário, enquanto o segundo se dedica à leitura, tratamento, manipulação e outras operações sobre os dados do arquivo binário resultante.

Durante a **segunda etapa** do projeto, foi desenvolvida outra aplicação em que o foco está na ordenação dos dados armazenados no arquivo binário gerado na etapa anterior. Foi atribuído ao nosso grupo dois atributos-chave para a ordenação: Product_type e Value, sendo o primeiro atributo considerado o ordenador primário e o segundo o secundário em caso de empates. O método de ordenação escolhido foi o Multiway MergeSort. Para avaliar o resultado do método escolhido foi enviado outra aplicação que consiste numa variação da segunda aplicação da etapa anterior em que é possível realizar manipulações e outras operações sobre o arquivo binário ordenado.

Durante todo o trabalho, um de nossos objetivos principais foi o desenvolvimento de um sistema com uma interface clara e intuitiva para o uso das operações de conversão, leitura, tratamento e ordenação de arquivos binários que tratasse a maioria dos erros de digitação do usuário.

Desenvolvimento

Primeira Etapa

- ***csv2bin.cpp:***

É declarado o registro DadosCSV, que representa os dados do arquivo CSV. Cada atributo do registro corresponde a um campo do registro no arquivo, sendo eles:

- **Time_ref** (valor tipo inteiro);
- **Account** (vetor de 8 caracteres);
- **Code** (vetor de 8 caracteres);
- **Country_code** (vetor de caracteres 26);
- **Product_type** (vetor de 10 caracteres);
- **Value** (valor tipo float);
- **Status** (vetor de 1 caracteres).

Em resumo, a aplicação consiste de um loop em que o código lê um arquivo CSV, extrai os campos de cada registro, converte os campos para os tipos apropriados citados anteriormente e, em seguida, salva os registros no arquivo binário “arqBin.bin”. As funções utilizadas para criar toda a lógica de extração dos dados são:

- **find()**: Encontra a posição final do campo atual com base no delimitador.
- **substr()**: Extrai o campo da linha, com base nas posições encontradas.
- **erase()**: Remove o campo processado da linha.

Alguns dos valores de “Value” estão vazios na base de dados. Nesses casos o sistema considera como valor 0.

Esse é o primeiro passo para preparar os dados para as operações de leitura, tratamento e ordenação descritas na segunda parte do projeto.

- ***manipulaArqBin.cpp:***

Essa aplicação realiza as operações sobre arquivo binário “arqBin.bin”. Ele inclui funções para manipular os dados no arquivo binário, como inserir um novo registro em uma posição específica, visualizar uma faixa de registros, alterar os dados de um registro em uma posição específica, trocar as posições de dois

registros e imprimir todos os registros. Além disso, o código possui um sistema de tratamento de erros de digitação e incoerência dos dados digitados pelos usuários.

O código utiliza a mesma estrutura do código “csv2bin.cpp” chamada **DadosCSV** para representar os registros de dados no arquivo binário.

Segue uma breve explicação das principais funções do código:

- **main():** A função principal que lida com a interação do usuário e executa a operação selecionada a partir da função auxiliar selecao().
- **trocarPosicao():** Troca as posições de dois registros no arquivo binário.
- **insereNaPosicao():** Insere um novo registro em uma posição específica no arquivo binário.
- **lerFaixaRegistros():** Lê e imprime uma faixa de registros do arquivo binário.
- **alterarDadosCSV():** Modifica os dados de um registro em uma posição específica.
- **imprimirTodos():** Imprime todos os registros do arquivo binário.
- **calculaNPosicoes():** Calcula o número de posições ocupadas no arquivo binário.
- **verificadorErro():** Realiza a verificação de erros para a entrada do usuário com base no tipo de dado esperado e no tamanho. Além disso, vale a pena citar sua função secundária **ehNumerico()** que verifica se uma sequência de caracteres é numérica, com opção de aceitar valores de ponto flutuante.

Segunda Etapa

Foi atribuído ao nosso grupo que a ordenação fosse em ordem decrescente e utilizasse os seguintes atributos-chave para comparação: Product_type e Value, sendo o primeiro atributo considerado o ordenador primário e o segundo o secundário em caso de empates. Para ordenação externa do arquivo binário seguindo essa especificação e manipulação do arquivo binário ordenado foi criado os seguintes códigos:

- ***sortFile.cpp:***

A aplicação “sortFile.cpp” é a implementação de um algoritmo de ordenação externa que utiliza o método de classificação chamado **Multi-Way MergeSort**. O objetivo é transformar um arquivo binário “arqBin.bin” em um outro arquivo binário ordenado “arqBinOrdenado.bin”. Segue uma explicação da metodologia do código:

Antes de explicar o código é necessário chamar a atenção para as seguintes coisas: O código utiliza a mesma estrutura do código “csv2bin.cpp” chamada **DadosCSV** para representar os registros de dados no arquivo binário; Para atender a especificação de ordenação foi definida uma sobrecarga do operador “**>**” para a estrutura **DadosCSV**. Isso permite que os registros sejam comparados com base no atributo **"Product_type"** e, em caso de empate, no atributo **"Value"**. Essa sobrecarga é necessária para a ordenação dos dados.

Primeiramente a função main é o ponto de entrada do programa. Ela fará abertura do arquivo binário de entrada “arqBin.bin” e realiza as seguintes etapas:

- Verifica se o arquivo foi aberto com sucesso.
- Chama a função **criarArqsTemp** para criar os arquivos temporários.
- Chama a função **multiwayMergeSort** para ordenar os arquivos temporários e gerar o arquivo final ordenado.

O algoritmo de ordenação **MergeSort** é utilizado para ordenar o vetor de registros durante a função **criarArqsTemp**. Essa função se chama recursivamente para repartir o vetor ao meio até que possua apenas um elemento. Após isso, é chamada a função auxiliar **merge** que será responsável pela ordenação dos vetores

divididos no **mergesort** comparando elemento a elemento dos vetores divididos e unindo-os em apenas um vetor ordenado.

A função **criarArqsTemp** é responsável por criar os arquivos temporários a partir do arquivo binário de entrada desordenado. Ela recebe como parâmetros o arquivo a ser dividido em arquivos temporários ordenados a quantidade máxima de registros por arquivo. A função lê os registros do arquivo de entrada e os armazena em um vetor. Quando o tamanho máximo de um arquivo temporário é alcançado, o vetor é ordenado usando o algoritmo mergesort e os registros são salvos em um arquivo temporário separado (o próprio código define os nome dos arquivos da forma: “arqTemp1.txt”). Essa função retorna o número total de arquivos temporários criados.

A classe **maxHeap** é definida juntamente com a classe **noh** que representa um registro a ser ordenado, essas classes serão utilizadas na função **multiwayMergeSort**. A heap é usada para **identificar o maior** dado e realizar a ordenação dos dados. A classe possui os métodos característicos para correção da estrutura da heap (**arruma**, **corrigeDescendo**, **corrigeSubindo**), inserir um dado na heap (**insere**) e retirar a raiz da heap (**retiraRaiz**).

A função **multiwayMergeSort** realiza a etapa de ordenação externa usando o método de classificação Multi-Way MergeSort. Essa função recebe por parâmetro o nome do arquivo de saída que será criado e o total de arquivos temporários. A função irá criar (ou sobrescrever) o arquivo de saída com o nome passado por parâmetro. Para iniciar o processo de ordenação é criado um vetor com os arquivos temporários para leitura, um vetor do tipo noh e uma **maxHeap**. Logo em seguida é feita a leitura do primeiro elemento de cada arquivo para o vetor tipo noh, para que possa ser associado o index do arquivo de origem e o noh é inserido no **maxHeap**. Após isso, a raiz do **maxHeap** é retirada e escrita no arquivo de saída e um novo elemento do mesmo arquivo que o da raiz (objetivo do index) é adicionado ao **maxHeap**, que é “arrumado”, esse processo será repetido enquanto todos os arquivos temporários não estiverem vazios. Por fim, todos os arquivos temporários são excluídos e o arquivo de saída finalmente estará ordenado.

- ***manipulaArqBinOrdenado.cpp:***

A aplicação “manipulaArqBinOrdenado.cpp” é uma adaptação da aplicação “manipulaArqBin.cpp”. Ela é uma cópia da aplicação da etapa passada, possuindo as mesmas funções implementadas anteriormente. Sua única modificação é que ela abrirá por padrão o arquivo binário “arqBinOrdenado.bin” (linha de código 13) e possibilitará avaliar o resultado da ordenação pelo método Multi-way MergeSort.

Conclusão

O trabalho proposto pelos professores de Estrutura de Dados oferece uma oportunidade de explorar técnicas avançadas de manipulação e processamento de dados em memória secundária, além de aprimorar a compreensão e aplicação de algoritmos de ordenação eficientes. Ao final do trabalho podemos concluir que o grupo adquiriu conhecimentos sólidos sobre a conversão e manipulação de arquivos binários, bem como sobre a implementação e avaliação de algoritmos de ordenação em memória secundária.

Referências:

- https://en.wikipedia.org/wiki/K-way_merge_algorithm
- https://www.youtube.com/watch?v=kdBg79KBBuw&list=PLKoAQTS8rBFp0ZRb_9fhEyY3nryg5iixU
- https://www.youtube.com/watch?v=XI7AHQIQzls
- https://www.youtube.com/watch?v=0Whr9aOqgrI&t=649s
- https://www.tutorialandexample.com/external-merge-sort-in-cpp
- https://www.tutorialspoint.com/external-sorting-with-example-in-cplusplus#:~:text=External%20Sorting%20is%20a%20category,secondary%20memory%20(%20hard%20disk)
- https://www.geeksforgeeks.org/external-sorting/