BEATRIZ FERNANDES TEIXEIRA
DÊNIS DE SOUZA CORDEIRO
RONALD DE SOUZA GALDINO

TRABALHO PRÁTICO:
PARTE 2

Informações

Grupo**:** 4
Base de dados : data_athlete_game.7z

Descrição parte 1

Primeiramente, ao executar o código, é chamada automaticamente uma função(conversão()), que lê os dados do arquivo .csv em strings, chama outra função escritor que converte essas strings jogando-as em uma estrutura com vetores de char de tamanhos estabelecidos conforme a necessidade do arquivo através da função strcpy() ,e, em seguida retorna uma variável do tipo da estrutura para a função conversão, esta por sua vez escreve os dados em um arquivo binário.

Posteriormente, é impresso o menu para escolher qual procedimento realizar, com as opções de inserir novo atleta, visualizar os registros entre duas posições, alterar o registro em uma posição específica, imprimir todos os registros, trocar dois registros de posição e sair do programa, implementado dentro de um while com switch e case para realizar a interação com o usuário.

Caso seja escolhido inserir novo atleta, é pedido ao usuário que digite a posição em que o novo registro será inserido e os dados desse novo atleta, logo em seguida é chamada a função escritor para transformar os dados inseridos em uma variável no tipo da estrutura, logo após é chamada a função para inserir esse atleta na base de dados.

Caso seja escolhido visualizar registros entre duas posições, é pedido ao usuário para digitar as posições dentre as quais ele deseja que os registros sejam visualizados, e, é chamada a função listar atletas que recebe as posições inseridas como parâmetros, realiza uma busca, e quando encontra, imprime os dados até a posição desejada.

Caso seja escolhido alterar os dados de um registro em uma posição específica, é pedido para o usuário inserir a posição a ser alterada e os dados desse novo registro, novamente chamada a função escritor para retornar uma variável do tipo da estrutura,e, em seguida essa variável recebida é passada como parâmetro da função alterar registro para promover a alteração.

Caso seja escolhido imprimir todos os registros,é chamada a função imprimir registros, que enquanto o arquivo puder ser lido realiza a impressão dos dados nele contidos.

Caso seja escolhido trocar dois registros de posição, é solicitado ao usuário que digite as posições em que a troca de dados deverá ser feita, em seguida é chamada a função trocar atletas com as posições inseridas sendo os parâmetros, posicionando no arquivo e efetuando a troca.

Caso seja escolhido sair do programa, a repetição é finalizada e a execução do programa encerrada.

Descrição parte 2:

Ordenação: Name (1º), Id (2º), crescente.

Ao executar o arquivo “ordenacao.cpp” o programa chama inicialmente uma função, presente no arquivo “conversao.cpp” que converte o arquivo CSV em binário. Inicialmente foi definido que o número de registros de cada arquivo temporário seria de 1000 (variável “n” no arquivo “ordenacao.cpp”), isto é 126kb por arquivo temporário criado, já que cada registro ocupa 126 bytes. A variável “k” foi definida pelo valor de 600 porque a velocidade de processamento da ordenação de cada arquivo de cada buffer foi satisfatória, cada buffer de cada arquivo gerado ocupando um tamanho de 75,6 kbytes. Logo após a conversão é chamada a função "mergeSortExterno" que tem o arquivo em binário como parâmetro. Esse método de ordenação externa utiliza a técnica de intercalação, por meio da qual pequenos blocos são ordenados individualmente e posteriormente combinados em blocos maiores e ordenados, até resultar em um único bloco do tamanho do arquivo totalmente ordenado.

Os blocos a serem ordenados são criados pela função “criaArquivosOrdenados”, que pega o arquivo principal e divide em partes já ordenadas, de acordo com a capacidade da RAM previamente definida por 126kb, salvando essas partes em diferentes arquivos temporários. Os arquivos temporários são lidos pelo uso de buffers, que estão contidos na estrutura “arquivo”, de forma que permitem o armazenamento dos primeiros registros de cada arquivo (já ordenado).

A função “merge” cria um vetor com o endereço dos arquivos menores ordenados através do método “procuraMenor”, que retorna o menor registro de cada posição de cada buffer que constitui cada objeto da estrutura “arquivo” e preenche os buffers em outro método auxiliar “preencheBuffer” e vai salvando os registros pelo método “salvaOrdenado”. Observamos que alguns dos nomes do arquivo iniciam-se com aspas ou espaços, então criamos um métodos para ignorar esses caracteres, o “retornaPrimLetra”.

A função “retornaTamanhoarq” foi feita para retornar o tamanho dos blocos ordenados individualmente.