Projeto 1: Arquivos - Compactação e Recuperação dinâmica

O objetivo do projeto é gerenciar um sistema de arquivos que gerência a locação de veículos de uma dada empresa. O sistema armazena as seguintes informações:

- Código do Cliente (CodCli)
- Código do Veículo (CodVei)
- Nome do Cliente
- Nome do Veículo
- Número de Dias

A chave primária é composta pela composição "CodCli+CodVei". Para simplificar, considere que um cliente pode alugar uma única vez um determinado carro (i.e., CodCli+CodVei será sempre único). O arquivo a ser criado deve ser **binário** de **registros e campos de tamanho variável**, com **1 byte** no início do registro como **indicador de tamanho do registro**, e com **campos separados** pelo caractere '**|**'.

Código do Cliente	Código do Veículo	Nome do Cliente	Nome do Veículo	Número de Dias
11 caracteres (fixo)	7 caracteres (fixo)	50 caracteres (máximo)	50 caracteres (máximo)	int (fixo ou caracteres)

Ex.: <57 *1 byte*>12121212121|ABC1234|João da Silva|Chevrolet Agile 2010|2|

As seguintes funcionalidades deverão estar disponíveis:

1. Inserção
2. Pesquisa por chave primária, i.e., "CodCli+CodVel" (índice primário)
3. Carrega Arquivos (dependente da implementação)

1 - Inserção

Insere o registro no final do arquivo. Deve-se atualizar o arquivo de índice (vide funcionalidade 2).

Os dados a serem inseridos devem ser recuperados de um arquivo a ser fornecido no momento da execução do programa (vide funcionalidade 3).

2 - Pesquisa por chave primária

Para essa funcionalidade deve-se gerenciar um arquivo de índice que contenha a lista das chaves primárias, i.e., "CodCli+CodVel", presentes no arquivo de dados junto com o deslocamento (byte offset) necessário para acessar o registro de cada chave presente no arquivo. Assim, uma consulta deve primeiramente procurar a chave desejada neste novo arquivo e depois acessar diretamente o registro desejado no arquivo de dados. Os dados relacionados ao "CodCli+CodVel" pesquisado devem ser exibidos.

Os dados a serem pesquisados devem ser recuperados de um arquivo a ser fornecido no momento da execução do programa (vide funcionalidade 3).

Observações:

(1) a inserção de um registro requer a manipulação de 2 arquivos (dados e índice).
(2) A busca no índice primário pode ser feita sequencialmente ou por pesquisa binária. Logo o índice deve ser mantido ordenado, usar função de sort. Em memoria o índice pode ser mantido em um vetor de tamanho fixo (e.g., 25) ou lista encadeada.
(3) O índice deve ser mantido em memória principal e, em caso do programa ser interrompido inesperadamente, o índice deve ser recriado a partir do arquivo de dados. Desse modo, deve existir uma função que carrega o índice para a memória e uma que recria o índice quando necessário. Para criar/recriar o índice utilizem o Keysorting. Utilizar uma flag/campo no header do arquivo de índice para marcar se o mesmo está sincronizado com o arquivo de dados.

3 - Carrega Arquivos

A fim de facilitar os testes e avaliação do projeto, serão fornecidos dois arquivos:

a) "insere.bin"
b) "busca_p.bin"

O arquivo (a) conterá os dados a serem inseridos durante os testes. Não necessariamente todos os dados serão inseridos, isto é, está funcionalidade deve perguntar ao usuário qual registro deve ser inserido. Para tanto, uma sugestão é carregar o arquivo em memória (um vetor de struct, por exemplo) e ir acessando cada posição conforme as inserções vão ocorrendo.

O arquivo (b) conterá uma lista de chaves primarias, "CodCli+CodVei", a serem utilizados durante a pesquisa por chave primária (funcionalidade 2). A ideia é a mesma já descrita, ou seja, carregar o arquivo em memória (um vetor de struct, por exemplo) e ir acessando cada posição conforme as remoções vão ocorrendo.

Nesses arquivos os registros seguem uma organização de registro e campo de tamanho fixo (que difere da estrutura que deve ser usada no projeto). Ver códigos fonte fornecidos.

Observações gerais:

(1) Não criar o arquivo toda vez que o programa for aberto (fazer verificação). Isto é, o programa pode ser encerrado, e ao recomeçar deve continuar com o arquivo do estado que parou!

(2) O arquivo principal deve ser manipulado totalmente em memória secundária!

(3) Criar um pequeno menu para acessar cada uma das funcionalidades (1, 2 e 3, sendo que a 4 pode ser ativada ao iniciar o programa). Note que as funcionalidades podem ser executadas de forma aleatório de acordo com a necessidade do usuário. Por exemplo, 3 inserções → 1 busca → 1 inserção → fechar/abrir programa → 2 inserções → 1 busca.

(4) A avaliação terá uma dinâmica como o seguinte exemplo:

1. execute o programa
2. insira um registro – 3 (índice do arquivo insere.bin)
3. insira um registro – 5 (índice do arquivo insere.bin)
4. insira um registro – 1 (índice do arquivo insere.bin)
5. feche o programa
6. abra os arquivos no editor hexadecimal
7. execute o programa novamente
8. pesquisa um registro – 2 (índice do arquivo busca.bin)
9. insira um registro – 2 (índice do arquivo insere.bin)
10. feche o programa inesperadamente (fechar terminal)
11. abra os arquivos no editor hexadecimal
12. execute o programa novamente
13. insira um registro – 4 (índice do arquivo insere.bin)
14. pesquisa um registro – 3 (índice do arquivo busca.bin)
15. feche o programa
16. abra o arquivo no editor hexadecimal
17. …