

Exercício

O objetivo do exercício é gerenciar um sistema de streaming de filmes de uma dada empresa. O sistema armazena as seguintes informações: Código do Cliente (CodCli), Código do Filme (CodF), Nome do Cliente, Nome do Filme, e Gênero. A chave primária é composta pela composição “CodCli+CodF”. Para facilitar, suponha que um cliente pode assistir uma única vez um determinado filme. O arquivo a ser criado deve ser de registros e campos de tamanho variável, com um inteiro (4 bytes) no início do registro indicando o tamanho do registro, e com campos separados pelo caractere ‘#’.

Código do Cliente (CodCli)	Código do Filme (CodF)	Nome do Cliente	Nome do Filme	Gênero
int (a ser bufferizado)	int (a ser bufferizado)	50 caracteres (máximo)	50 caracteres (máximo)	50 caracteres (máximo)

Ex.: 601#1#João da Silva# Indiana Jones e a Última Cruzada#Aventura

As seguintes operações deverão estar disponíveis:

1. Inserção
2. Remoção
3. Compactação
4. Carrega Arquivos (dependente da implementação)

Inserção (1)

Ao adicionar um registro vocês terão que percorrer a lista de espaços disponíveis verificando se o novo registro se encaixa em algum dos espaços (vide Opção 2). Para tanto, usem a estratégia first-fit e, para facilitar, podem considerar fragmentação interna. Caso nenhum elemento da lista supra o espaço necessário para o novo registro, acrescente-o no final do arquivo. Os dados a serem inseridos devem ser recuperados de um arquivo a ser fornecido no momento da execução do programa (vide Opção 4).

Remoção (2)

Dada a chave de busca “CodCli+CodF” (recuperado de um arquivo a ser fornecido no momento da execução do programa (vide Opção 4)) realize a remoção do respectivo registro. A remoção deve ser feita diretamente no arquivo de dados. Para reaproveitar o espaço removido vocês terão que acrescentar no arquivo uma lista ligada entre os espaços disponíveis. Assim, vocês terão que acrescentar as seguintes informações no arquivo:

- (1) criem um registro cabeçalho e nele um campo que indica o offset para o primeiro elemento da lista.
- (2) ao remover um registro, substitua-o no arquivo por: <tamanho em bytes do registro removido>*<offset para o próximo elemento da lista>, onde * é um marcador indicando que este espaço está disponível (esse formato é apenas uma sugestão).
- (3) um novo espaço disponível deve ser acrescentado sempre no início da lista. Logo, vocês devem atualizar o offset do cabeçalho e guardar o seu antigo offset no novo elemento da lista.
- (4) o final da lista é indicado por -1 no campo offset para o próximo elemento.

Compactação (3)

A estratégia de remoção vai criar fragmentos (internos e externos). Reconstruam o arquivo, quando solicitado pelo usuário, compactando todos os registros e limpando esses fragmentos (internos e externos).

Carrega Arquivos (4)

A fim de facilitar os testes, serão fornecidos dois arquivos: (a) “insere.bin” e (b) “remove.bin”. O primeiro (a) conterá os dados a serem inseridos durante os testes (não necessariamente todos os dados serão inseridos). 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. Note que é possível encerrar a execução e recomeçar a execução, sendo necessário marcar, de algum modo, quantos registros já foram utilizados do mesmo.

Em relação a (b), o arquivo conterá uma lista de chaves, “CodCli+CodF”, a serem utilizados durante a remoção. 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. Note que é possível encerrar a execução e recomençar a execução, sendo necessário marcar, de algum modo, quantos registros já forma utilizados do mesmo.

Observações:

- (1) Não criar o arquivo toda vez que o programa for aberto (fazer verificação).
- (2) O arquivo deve ser manipulado totalmente em memória secundária!