Documentação externa da primeira parte do trabalho de Organização de Arquivos

09/05/2018

Universidade de São Paulo

Bacharelado em Ciências da Computação

Disciplina de Organização de Arquivos

Profa. Dra. Cristina Dutra de Aguiar Ciferri

Enzo Bustamante Junco Mendonca	9863437
Fábio Augusto Romanini Pinto	9791312
Ricardo A Araujo	9364890
Tiago Esperança Triques	9037713

Contents

Seção 1 - Leitura de dados

csv2bin

A função lê uma linha de cada vez do arquivo csv com freadline e separa cada campo com split e escreve os dados em outro arquivo binário (ambas funções implementadas em utils.c).
Uma mensagem de erro é printada caso o arquivo de entrada e/ou saída não consiga ser aberto.
Começamos escrevendo o registro de cabeçalho no arquivo de dados.
Para cada linha lida, separamos os campos de cada registro usando a função split.
Os dados então são escritos no arquivo de dados.
Primeiramente são escritos os campos de tamanho fixo.
Para os campos de tamanho variável, usamos indicador de tamanho.
Após escrever todos os dados, os arquivos são fechados e uma mensagem de sucesso é exibida na tela.

Como estamos usando registros de tamanho fixo, decidimos que o último campo do registro - 'prestadora' - ocupará o que sobra do tamanho do registro, ou seja, se o registro tem 87 bytes e os demais campos ocupam 50 bytes, o campo prestadora ocupará 87 - 50 = 37 bytes.
Apenas os primeiros bytes do campo realmente serão utilizados (de 2 a 4, no máximo), os demais serão espaço em branco.

Seção 2 - Recuperação de todos os registros

bin2out

A função recupera os dados de todos os registros - os que não foram removidos - do arquivo de dados 'output.dat' e os imprime de maneira organizada na tela.
O arquivo de dados é aberto para leitura.
Caso o arquivo não for encontrado, uma mensagem de erro é apresentada.
Como não iremos alterar o arquivo de dados, apenas ignoramos o registro de cabeçalho - pulamos 5 bytes do começo do arquivo.
Os dados dos registros começam a ser lidos.
Se o campo 'codINEP' de um registro for -1, esse registro foi removido.
Pulamos para o registro seguinte.
Caso o registro não tenha sido removido, ou seja, campo 'codINEP' diferente de -1, recuperamos os demais campos do registro.
Utilizando a função catReg (utils.c), os dados recuperados são exibidos na tela.
Toda memória alocada é liberada e o arquivo de dados é fechado.

Seção 3 - Busca por critério

bin2outGrep

Começamos abrindo o arquivo de dados para leitura e assim como na função bin2out iremos recuperando todos os campos dos registros armazenados.
Essa função recebe como parâmetro o campo e o valor a ser buscado.
Para cada registro recuperado, comparamos o valor do campo buscado com o do registro.
Caso for igual, imprimimos os dados do registro.
Repete-se até chegar no fim do arquivo de dados.
Caso a busca não encontre nenhum registro contendo o campo buscado ou caso o registro tenha sido removido, uma mensagem de erro é exibida na tela.

Para facilitar, usamos uma função de comparação que consegue comparar tanto números quanto strings.
Assim que chamamos a função bin2outGrep, já passamos como um dos parâmetros a função de comparação correta.

Seção 4 - Busca por RRN

bin2outRRN

Função para recuperar os dados de um registro específico, através de seu RRN.
Começamos abrindo o arquivo de dados e caso ocorra um erro na abertura desse arquivo, imprimir mensagem de erro.
Andamos no arquivo com a função fseek para o byte offset relativo ao registro buscado:
Checamos se o byte offset está nos limites do tamanho do arquivo.
Fazer a leitura dos campos dos registros:
- Caso o campo codINEP seja -1, o registro foi anteriormente removido. Informar ao usuário que o registro é inexistente e sair da função.
- Caso contrário recuperar os demais campos e imprimir o registro na tela.
Fechar o arquivo de dados.

Seção 5 - Remoção lógica de registros

bin2outTrash

Função que remove o registro especificado pelo RRN passado por parâmetro.
Caso o registro já tenha sido removido ou caso seja inexistente, uma mensagem de erro é exibida.
A remoção é apenas lógica - marcamos os registros removidos com -1 no campo codINEP.

Abrimos o arquivo de dados para leitura dos registros.
Checar se existe um registro com RRN que foi passado por parâmetro para a função: Se o tamanho do arquivo de dados for menor que (RRN * tamanho dos registros), imprimir mensagem informando que o registro é inexistente.
Caso o offset do registro esteja dentro dos limites do arquivo, pular para esta posição.
Ler o primeiro campo do registro - codINEP. Ele também nos informará se o registro é valido:
- Se o codINEP do registro for -1, ele já tinha sido removido anteriormente. Imprimir mensagem informando que o registro é inexistente.
- Se o codINEP for diferente de -1 o registro ainda não tinha sido removido. Voltamos 4 bytes para sobrescrever o campo codINEP com -1. Armazenamos o antigo topo da pilha nos próximos 4 bytes do registro.
Atualizar o registro de cabeçalho:
Voltamos para o início do arquivo e alteramos o campo topoPilha com o RRN do registro que acabou de ser removido.
Imprimir mensagem informando sucesso na remoção.

Seção 6 - Inserção de novos registros

add2bin

Começamos convertendo o argumento do console em seus respectivos campos.
Abrimos o arquivo corrente e verificamos se o arquivo existe, se não relatamos o erro.
Muda-se o status no cabeçalho e se copia o topo da pilha.
Lê se a pilha e verifica se ha espaços para serem reaproveitados.
Se houverem, inserimos o registro no espaço do ultimo registro removido.
Se não houverem, inserimos o registro ao final do arquivo.
Retornamos o status do cabeçalho, e escrevemos o novo topo da pilha.
Fechamos o arquivo e relatamos sucesso na inserção.

Seção 7 - Atualização dos campos do registro

updateBin

Função de update, recebe o argumento do console, busca o registro por RRN e o troca seus campos pelos campos do argumento.
Começamos convertendo o argumento do console em seus respectivos campos.
Abrimos o arquivo corrente e verificamos se:
- Ele existe;
- Se buscamos por um RRN plausível, dado o tamanho do arquivo.

Caso estas condições não sejam cumpridas relata-se o erro ao usuário.
Caso sejam, desloca-se o ponteiro de arquivo ate o RRN, onde verificamos se o registro existe ou foi removido, relatando erro neste ultimo caso.
No caso de nenhum desses erros ocorrerem, estamos aptos a trocar os campos antigos pelos campos, já convertidos do argumento.
Após a realização do update, relatamos sucesso e fechamos o arquivo.

Seção 8 - Compactação (desfragmentação)

binDefrag

Começamos renomeando o arquivo de dados fragmentado para output.dat.old
Essa operação poderá falhar caso não exista um arquivo de dados anteriormente.
Neste caso, imprimimos uma mensagem de erro.
Criamos um novo arquivo de dados, chamado output.dat e inicializamos o registro de cabeçalho.
Lê-se um registro do arquivo de dados fragmentado (output.dat.old): - Caso tenha sido removido, ou seja, caso o codINEP do registro seja -1, ignorar e pular para o próximo registro.
- Caso não tenha sido removido, recuperar os demais campos do registro e escrevê-los novamente no novo arquivo (output.dat).
Após recuperar todos os registros, atualizar o registro de cabeçalho, indicando que os dados estão consistentes.
Imprimir uma mensagem na tela informando que a compactação foi bem sucedida.

Seção 9 - Pilha dos RRNs removidos

recBin

Função que imprime a pilha de registros removidos.

Primeiro abrimos o arquivo e verificamos se existe. Caso contrário relata-se o erro.
Muda-se o status do arquivo em seu cabeçalho, guardando também o topo da pilha.
Caso a pilha esteja vazia, se relata o erro.
Caso contrário se imprime recursivamente todos os RRNs removidos do arquivo até que se chegue a base da pilha.
Após a impressão, retorna-se o status ao valor inicial e se fecha o arquivo.