Trabalho Prático (Parte I)

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Bacharelado em Ciências de Computação
Disciplina de Organização de Arquivos
Profa. Dra. Cristina Dutra de Aguiar Ciferri

Grupo 07:
10295311 - Alexis de Almeida Coutinho
9897988 - Guilherme Marchetto
9283607 - João Vitor Guino Rieswick
9779392 - Wallace Cruz de Souza

09/05/2018

Índice

Funcionalidade 1: Converte

Algoritmo:

- Abrir os arquivos de entrada e saída;
- Escrever os dados do cabeçalho no arquivo de saída;
- Enquanto não for o fim do arquivo:
- Armazenar os dados do arquivo de entrada em um vetor até que se encontre um ‘ponto e vírgula (‘;’) ou uma quebra de linha e guardar o tamanho do vetor;
- Encontrou um ponto e vírgula ou uma quebra de linha, incrementar o contador de campos;
    - Caso o valor do contador seja 1: inserir o conteúdo e o tamanho do vetor na região de prestadora de serviço no registro;
    - Caso o valor do contador seja 2: inserir o conteúdo do vetor na região de data no registro;
    - Caso o valor do contador seja 3: inserir o conteúdo do vetor na região de código INEP no registro;
    - Caso o valor do contador seja 4: inserir o conteúdo e o tamanho do vetor na região de escola do registro;
    - Caso o valor do contador seja 5: inserir o conteúdo e o tamanho do vetor na região de município no registro;
    - Caso o valor do contador seja 6: inserir o conteúdo do vetor na região de UF no registro;
    - Se o último byte lido for uma quebra de linha e se o registro já possuir o tamanho correto, escreve o registro no arquivo de saída;
    - Caso contrário corrige o tamanho do registro e escreve no arquivo de saída;
- Se a leitura do arquivo de entrada chegou ao fim, atualiza o cabeçalho e fecha os arquivos

Como pode-se observar pelo algoritmo, foi executada uma leitura de arquivo com registros de tamanho variável, campos híbridos (registro com campos de tamanho fixo e campos de tamanho variável) e delimitadores de campo (representado pelo ponto e vírgula) e registros (representado pela quebra de linha).

Funcionalidade 2: Imprime arquivo

Algoritmo:

- Abrir o arquivo de dados;
- Posicionar o “cursor” de leitura após o cabeçalho;
- Enquanto não for o fim do arquivo:
    - Ler 4 bytes (campo INEP) do arquivo e testar se é um registro não removido:
- Se for um registro removido: mover o cursor para o próximo registro;
- Senão, ler os demais campos e armazenar em um registro;
- Imprimir os campos INEP, data e UF;
- Testar se o primeiro indicador de tamanho é igual a 0:

    - Se for igual imprimir somente o indicador;
    - Senão, imprimir o indicador e o nome da escola;
- Testar se o segundo indicador de tamanho é igual a 0:
    - Se for igual imprimir somente o indicador;
    - Senão, imprimir o indicador e o nome da escola;
- Testar se o terceiro indicador de tamanho é igual a 0:
    - Se for igual imprimir somente o indicador;
    - Senão, imprimir o indicador e o nome da prestadora do serviço;
- Se a leitura do arquivo chegou ao fim, fecha-se o arquivo.

Foi utilizado máscaras com o tamanho definido na impressão para que fosse impresso somente a parte desejada do vetor, que possui o tamanho da maior variável para que posso ser utilizado de forma generalizada.

Funcionalidade 3: Busca por valor do campo

Algoritmo:

- Verificar se campo de busca é ‘codINEP’
    - Se for, converter o argumento valor de string para número. Atribuir o ponteiro de função adequado à comparação do código INEP.
    - Se não, continuar.
- Se o campo não é código INEP, então o valor é uma string, portanto posso ‘remover’ as aspas da string valor. Para isso, incremento o ponteiro da string em uma posição e substituo o último caractere por \0.
- Comparo a string campo com cada uma das restantes possibilidades e atribuo o ponteiro de função adequadamente.
- Percorro todo o arquivo de dados, registro por registro, e verifico se o registro não está removido e o parâmetro valor coincide com o valor de cada registro.
    - Se sim, imprimo o registro inteiro e desmarco a variável ‘nfound’.
- Se não foi encontrado nenhum registro (‘nfound’ não foi desmarcado), imprimo a mensagem adequada.

Funcionalidade 4: Busca por RRN

Algoritmo:

- Abre o arquivo de dados;
- Checa se o RRN inserido está contido no arquivo através da seguinte fórmula:
- (TAMANHO DO ARQUIVO - TAMANHO DO CABEÇALHO)/TAMANHO DO REGISTRO <= RRN
- Se está contido no arquivo, posiciona o leitor no inicio do registro de RRN correspondente.
- Checa se o mesmo ja foi removido:
- Lê-se 4 bytes e, se são equivalentes a -1, já foi removido. Imprime "Registro inexistente." e fecha o arquivo.

- Se não, imprime os dados do registro na formatação especificada.
- Se não está contido no arquivo, imprime "Registro inexistente." e fecha o arquivo.

Funcionalidade 5: Remoção dinâmica por RRN

Algoritmo:

- Abrir o arquivo de dados;
- Avaliar se o RRN é válido;
- Se não for válido, imprime “Registro Inexistente” e fecha o arquivo;
- Se for válido:
    - Posicionar o cursor no início do registro do RRN dado;
    - Checar se o arquivo já foi removido:
        - Ler o inteiro e testar se é igual a -1:
        - Se for igual, imprime “ Registro Inexistente” e fecha o arquivo;
        - Senão, deslocar para o próximo registro removido (tamanho do registro * topo + cabeçalho) e pular o primeiro campo que representa a remoção do registro;
        - Ler 4 bytes, que indica o próximo registro removido;
- Fechar o arquivo;

O RRN é validado a partir de tamanho do arquivo sem cabeçalho dividido pelo tamanho do registro, assim temos a quantidade total de registros no arquivo e portanto se o RRN for menor ou igual que essa quantidade é um valor válido.

Funcionalidade 6: Inserção dinâmica

Algoritmo:

- Analisa se a data ou o UF recebido como argumentos são nulos;
- Caso algum seja nulo sua variável é preenchida com ‘0’ na quantidade correta;
- Abre o arquivo de dados;
- Recupera o valor de topo no cabeçalho;
    - Se existir um registro logicamente removido:
        - Posiciona o cursor de leitura nesse registro;
        - Recupera o RRN do próximo registro logicamente removido;
        - Atualiza o cabeçalho com o novo RRN;
    - Senão, posiciona o cursor de leitura no final do arquivo;
- Na posição definida (fim do arquivo ou em um registro logicamente removido):
    - Escreve os campos de tamanho fixo;
    - Escreve o tamanho do nome e o nome da escola;
    - Escreve o tamanho do nome e o nome do município;
    - Escreve o tamanho do nome e o nome da prestadora do serviço;
    - Se o registro não completou o tamanho correto, preenche o restante com espaço;
- Fecha o arquivo;
- E imprime “Registro inserido com sucesso”.

Como nessa função recebe-se todos conteúdos a serem inseridos como strings, foi escolhido utilizar ponteiros para char no lugar do registro de escrita, evitando-se assim a necessidade de argumentos extras com a informação do tamanho de cada campo variável. Utilizando-se da capacidade de definir a quantidade de bytes a serem escritos da função fwrite, foi removido o byte delimitador das strings (‘\0’).

Funcionalidade 7: Atualização de campos por RRN

Algoritmo

- Abre o arquivo de dados;
- Checa se o RRN inserido está contido no arquivo através da seguinte fórmula:
- (TAMANHO DO ARQUIVO - TAMANHO DO CABEÇALHO)/TAMANHO DO REGISTRO <= RRN
- Se está contido no arquivo, posiciona o leitor no início do registro de RRN correspondente.

- Checa se o mesmo já foi removido:
- Lê-se 4 bytes e, se são equivalentes a -1, já foi removido. Imprime "Registro inexistente." e fecha o arquivo.
- Se não, atualiza os dados do registro na formatação especificada:
- Escreve os campos de tamanho fixo;
- Escreve o tamanho do nome e o nome da escola;
- Escreve o tamanho do nome e o nome do município;
- Escreve o tamanho do nome e o nome da prestadora do serviço;
- Se o registro não completou o tamanho correto (87 bytes), preenche o restante com 0;

- Se o RRN não está contido no arquivo, imprime "Registro inexistente." e fecha o arquivo.

Funcionalidade 8: Compactação dos dados

Algoritmo:

- Crio um arquivo temporário e escrevo no seu cabeçalho o status 0 (porque os dados ainda vão ser copiados) e a pilha vazia (-1).
- Percorro todo o arquivo de dados, registro por registro, e verifico se o registro não está removido.
    - Se não está, copio o registro inteiro para o arquivo temporário.
- Atualizo o status no cabeçalho do arquivo temporário para 1 já que a transferência já terminou.
- Substituir o arquivo de dados antigo com o temporário.

Funcionalidade 9: Recuperação por RRN de registro removido

Algoritmo:

- Abrir o arquivo de dados;
- Ler o topo da pilha no cabeçalho;

    - Se o topo for igual a -1:
    - Imprimir “Pilha vazia” e fechar o arquivo;
    - Senão, enquanto o topo for diferente de -1, fazer:
    - Imprimir topo;
    - Deslocar para o próximo registro removido (tamanho do registro * topo + cabeçalho), descolar os primeiros 4 bytes que representam a remoção do registro;
    - Ler 4 bytes, que indica o próximo registro removido;
- Fechar o arquivo;

Foi definido no projeto marcar com o inteiro -1 o início do registro removido, isso devido ao fato do primeiro campo ser código INEP que é não nulo e positivo. O algoritmo utiliza o próprio arquivo de dados como pilha, armazenando o RRN do próximo registro removido logo após a marcação de remoção.

Referências Bibliográficas e Observações

O programa deve ser compilado com make all.
O sistema operacional utilizado foi ubuntu 16.04.

Slides das aulas:
http://wiki.icmc.usp.br/images/3/33/SCC0215012018camposRegistros.pdf
http://wiki.icmc.usp.br/images/b/b1/SCC0215012016acesso.pdf
http://wiki.icmc.usp.br/images/1/16/SCC0215012016reaproveitamentoEspa%C3%A7o.pdf