

## ACH2024 Algoritmos e Estruturas de Dados II

Semestre 01/2014 - turmas 04 e 94

Prof. Ivandré Paraboni [ivandre@usp.br](mailto:ivandre@usp.br)

Estagiário PAE: Thiago Castro Ferreira [thiago.castro.ferreira@gmail.com](mailto:thiago.castro.ferreira@gmail.com)

### Exercícios básicos em arquivos

**Serialização de EDs** - faça você mesmo.

**Passo 1:** Escolha uma estrutura de dados dentre *lista sequencial* (i.e., vetor), *lista ligada dinâmica* (simples ou dupla), *árvore de busca binária*, *grafo em listas de adjacências* ou *grafo em matriz de adjacências*.

**Passo 2:** Escolha a operação de *leitura* (do arquivo para a memória) ou *escrita* (da ED em arquivo). Tipicamente a leitura é mais trabalhosa, pois envolve recriar a ED usando *malloc()* etc.

**Passo 3:** Escolha o tipo de arquivo: texto com delimitadores (use *fprintf/fscanf*) ou binário (use *fwrite/fread*).

**Passo 4:** Implemente o algoritmo correspondente à configuração escolhida (ED, operação, tipo de arquivo) e, em caso de dúvidas, teste no próprio compilador.

### Exercícios em arquivos simples (sem índices)

Considere arquivos de registros de tamanho fixo do tipo *REGISTRO* como segue:

```
typedef struct {
    int NroUSP; // chave primária
    int curso;
    int estado;
    int idade;
    bool valido; // para exclusão lógica
} REGISTRO;
```

Todos os arquivos são mantidos em ordem aleatória de chaves. Para os exercícios a seguir, não há nenhuma estrutura de índices disponível.

1. Reescreva um arquivo *arq1* em um novo arquivo *arq2* eliminando os registros inválidos.
2. Faça uma cópia invertida de *arq1* em um novo arquivo *arq2*, ou seja: copie o último registro (*n*) de *arq1* para o início de *arq2*, depois copie o registro *n-1* para a segunda posição etc.
3. Escreva uma função para inserir um novo registro *r* no arquivo, tomando cuidado para evitar chaves duplicadas.
4. Escreva uma função que, dada um *nroUSP* *X*, retorne o registro correspondente.
5. Escreva uma função para excluir todos os registros do curso *X*.
6. Escreva uma função para alterar o curso de um aluno de *nroUSP* *X* para o curso *Y*.
7. Implemente o procedimento de ordenação *KeySort*, que dado um arquivo *arq1* cria uma tabela temporária de chaves em memória (idêntica a uma tabela de índices primários) e então reescreve o arquivo em um novo arquivo de saída *arq2*, na ordem correta de chaves (exercício completo e altamente recomendável).

### Exercícios em tabelas de índices primários

Para os exercícios a seguir, considere os mesmos arquivos do tipo acima, e também a existência de uma tabela de índices primários mantida em memória e suas respectivas funções de manipulação.

*Bool inserirÍndice(Tabela, int nroUSP, int end) //inserção em tabela ordenada, retornando true/false*  
*int buscarEndereço(Tabela, int nroUSP) // retorna -1 se end não existe*  
*int excluirÍndice(Tabela, int nroUSP) // retorna o endereço excluído, ou -1 se não encontrar*

1. Escreva uma função para inserir um novo registro *r* no arquivo, tomando cuidado para evitar chaves duplicadas (use o índice).
2. Escreva uma função que, dada um *nroUSP X*, retorne o registro correspondente (use o índice).
3. Escreva uma função para excluir todos os registros do curso *X* (use o índice).
4. Escreva uma função para alterar o curso de um registro de *nroUSP X* para o curso *Y* (use o índice).
5. Escreva uma função para alterar o registro de *nroUSP X* para o *nroUSP Y* se possível (use o índice).

### Exercícios em tabelas de índices secundários

Nos exercícios a seguir considere ainda a existência de tabelas de índices secundários (e suas respectivas operações de inserção e exclusão) para os campos *int curso* e *int estado*, e duas funções adicionais que, dada uma chave secundária de interesse, retornam a lista ligada das chaves primárias a ela relacionadas:

*NO\* chavesCurso(int curso)*  
*NO\* chavesEstado(int estado)*

No caso dos exercícios de exclusão, certifique-se de que você entende porque a atualização de índices secundários (mas não do índice primário, que sempre precisa ser excluído) é opcional.

1. Escreva uma função para inserir um novo registro *r* no arquivo, tomando cuidado para evitar chaves duplicadas (verifique quais índices precisam ser atualizados).
2. Escreva uma função para exibir todos os registros do curso *X*.
3. Escreva uma função para excluir o registro de *nroUSP X* (verifique quais índices precisam ser atualizados).
4. Escreva uma função para excluir todos os registros do curso *X* (verifique quais índices precisam ser atualizados).
5. Escreva uma função para alterar a idade atual de um registro de *nroUSP X* para a idade *Y* (verifique quais índices precisam ser atualizados).
6. Escreva uma função para alterar o curso de um registro de *nroUSP X* para o curso *Y* (verifique quais índices precisam ser atualizados).
7. Escreva uma função para alterar o registro de *nroUSP X* para o *nroUSP Y* se possível (verifique quais índices precisam ser atualizados).

### Exercício de processamento cossequencial

Deseja-se ordenar um arquivo de 100MB utilizando-se a seguinte configuração: buffer de escrita de 250kb, registros contíguos ocupam posições contíguas em disco e memória interna  $MI = 4MB$ . Preencha a tabela abaixo com a quantidade de seeks e volume de bytes transferidos, e número *m* de intercalações. Não é necessário calcular tempos. Para facilitar arredondamentos, considere  $MB = 1$  milhão, e  $KB = 1$  mil.

Ordenação				intercalação de <i>m</i> = ____ vias			
Leitura		Escrita		Leitura		Escrita	
seek	transfer	seek	transfer	seek	transfer	seek	transfer