2ª LISTA DE EXERCÍCIOS

(Arquivos e Algoritmos Clássicos de Ordenação e Busca)

1) Elabore um programa que leia dois arquivos binários (*salbruto.bin* e *desc.bin*) contendo, respectivamente, os salários brutos e os descontos de até 100 funcionários. Para cada registro, deve-se calcular o salário líquido (sal. bruto – descontos) e gravá-lo no arquivo texto *salliq.txt*.

2) Suponha que exista dois arquivos texto (*nome.bin* e *notas.bin*) contendo, respectivamente, o nome e as notas de 0 a 10 (no máximo 5 por aluno) de um grupo de 70 alunos. Em cada arquivo, antes do conteúdo existe um número inteiro seqüencial (de 1 a 70) que corresponde ao código do aluno. Desenvolva um programa que leia estes arquivos e preencha um vetor de registros. Após a leitura, este programa deverá calcular a média do aluno, considerando a quantidade de notas existentes, e apresentá-la juntamente com o conceito "APROVADO" ou "REPROVADO".

   OBS: O aluno é aprovado se sua média for maior ou igual a 6.

3) Faça um programa que preencha um vetor com os nomes digitados pelo usuário (no máximo 100 nomes). Em seguida, permite a realização de várias pesquisas (utilizar o algoritmo de *busca seqüencial*) sobre o vetor resultante, retornando qual é a posição (ordem de entrada) do nome pesquisa ou uma mensagem explicativa quando ele não for encontrado. Tanto a entrada quanto a pesquisa serão finalizadas se o usuário digitar "FIM".

4) Elabore um programa que preencha um vetor com números inteiros digitados pelo usuário (no máximo 1000 números). Após a digitação, o vetor deverá ser ordenado e, em seguida, pesquisado inúmeras vezes (utilizar um algoritmo de busca em tabelas ordenadas). A cada pesquisa deverá ser retornado a posição (ordem de entrada) do número ou uma mensagem explicativa quando ele não for encontrado. Tanto a entrada quanto a pesquisa serão finalizadas se o usuário digitar -999.

5) Faça um programa que leia o arquivo texto *agenda.txt*, contendo o CPF, nome e telefone de no máximo 1000 pessoas e grave-o em um vetor. Após a leitura, o programa deverá ordenar este vetor através do método *select sort* e gravá-lo no arquivo texto *cadastro.txt*.

6) Faça um programa que leia o arquivo texto *cadastro.txt*, gravado no exercício anterior, e grave os registros em um vetor. Este programa deverá permitir inúmeras pesquisas ao vetor através do CPF, finalizando quando o usuário digitar um CPF negativo.

7) Elabore um programa que leia dois arquivos texto (*cod.txt* e *nome.txt*) contendo, respectivamente, os códigos de matrícula e os nomes dos alunos de Processamento de Dados; e gere um vetor com os dados dos alunos. Ao final, o vetor deve ser ordenado:

   a) pela matrícula (utilizar o método *bubble sort*), e gravado no arquivo binário *cod.bin*; e

   b) pelo nome (utilizar o método *insertion sort*), e gravado no arquivo binário *nome.bin*.

   OBS: O total de alunos matriculados é inferior a 80 e os nomes têm no máximo 60 caracteres.

8) Elabore um programa para pesquisar alunos matriculados no curso de Processamento de Dados. As pesquisas sejam feitas pelo nome ou pelo código de matrícula, de acordo com a opção escolhida pelo usuário. Para cada uma das opções, o programa deverá abrir o arquivo binário correspondente (gravado pelo programa do exercício anterior), carregar um vetor de registros e permitir repetidas pesquisas, até que o usuário digite um número negativo para matrícula ou "FIM" para o nome (neste caso deve-se voltar ao menu principal). Em cada pesquisa, deverão ser

apresentados os dados do aluno ou, se o registro não for encontrado, uma mensagem explicativa. Para sair do programa, o usuário deverá escolher a opção "Fim" do menu principal.

OBS: A pesquisa por código deverá utilizar o algoritmo de busca seqüencial para tabelas ordenadas. A pesquisa por nome deverá empregar o algoritmo de busca binária.

9) Construa um programa que leia um arquivo texto, cujo nome é passado como parâmetro de entrada, e carregue um vetor de registros de no máximo 100 registros com o seguinte formato:

- Matrícula: um número inteiro;
- Nome: uma string com no máximo 50 caracteres
- Sexo: somente o código de referência (F – Feminino ou M – Masculino)
- CPF: um número inteiro (deverá ser usado um inteiro longo – *long int*);
- Curso: uma string com no máximo 30 caracteres
- Telefone: uma string com no máximo 20 caracteres

OBS: Caso o arquivo não exista, deve-se continuar o programa com o vetor vazio.

Após a leitura, o programa deverá apresentar um menu com as seguintes opções: *inclusão, consulta, exclusão* e *fim do programa*. De acordo com a opção escolhida pelo usuário (opções inválidas devem ser tratadas), deve-se realizar os seguintes procedimentos:

a) *Inclusão*:

- Solicitar ao usuário uma nova matrícula.
- Verificar se a matrícula digitada já existe. Se existir, emitir uma mensagem explicativa e voltar ao passo anterior. Caso contrário, permitir o cadastro do novo aluno.
- Verificar se o usuário deseja cadastrar um novo registro. Se afirmativo, repetir os passos anteriores. Caso contrário, encerrar o módulo e retornar ao menu.

b) *Consulta*:

- Solicitar ao usuário a matrícula desejada.
- Verificar se a matrícula digitada já existe. Se existir, mostrar todos os dados do aluno (o sexo deve ser apresentado por extenso). Caso contrário, emitir uma mensagem explicativa de registro inexistente.
- Verificar se o usuário deseja realizar uma nova consulta. Se afirmativo, repetir os passos anteriores. Caso contrário, encerrar o módulo e retornar ao menu.

c) *Exclusão*:

- Solicitar ao usuário a matrícula desejada.
- Verificar se a matrícula digitada já existe. Se não existir, emitir uma mensagem explicativa de registro inexistente. Caso contrário, mostrar todos os dados do aluno (o sexo deve ser apresentado por extenso) e solicitar a confirmação da exclusão. Se confirmada, remover o registro. Se não, ir para o próximo passo.
- Verificar se o usuário deseja remover outro registro. Se sim, repetir os passos anteriores. Caso contrário, encerrar o módulo e retornar ao menu.

OBS: Remover um registro consiste em deslocar para esquerda todos registros subseqüentes e decrementar a quantidade de registros.

d) *Fim do Programa*:

- Ordenar o vetor de registros em ordem crescente (utilizar o método *quicksort*).
- Gravar o vetor ordenado no arquivo passado como parâmetro de entrada do programa. Se ele já existir, deve ser substituído. Senão, deve ser criado.
- Encerra-se o programa