Prova-05

Prof. Msc. Elias Batista Ferreira
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Cadastro de produtos (++)

 (++)

Um comerciante deseja criar um banco de dados primitivo para seu negócio. Ele deseja salvar em arquivo uma lista de itens em estoque. O software que ele deseja deve permitir a inclusão de itens sempre ao final do arquivo. Crie um programa que armazene *n* `structs ItemEstoque` no final de um arquivo binário chamado "estoque.bin". A estrutura de cada item é dada abaixo.

```
struct ItemEstoque {
   float valor;       // valor de 1 unidade do produto
   char[50] nomeItem;    // nome do produto
   int quantidade;       // quantidade em estoque
};
```

O programa deve apresentar o menu de opções:

```
   "1 - cadastrar itens"
   "2 - listar itens"
   "3 - sair"
```

Caso o usuário escolha a opção 1, o programa deve ler a quantidade de *n* itens a serem cadastrados, ler as informações de *n* **`struct`** `ItemEstoque` e retornar ao menu de opões.

Caso o usuário escolha a opção 2, o programa deve apresentar todos os itens presentes no arquivo seguindo o seguinte exemplo:

```
Item 1 -------------------------
valor: 10,99
nomeItem: caneta
quantidade: 10

Item 2 -------------------------
valor: 1,99
nomeItem: apontador
quantidade: 0
```

Caso usuário escolha a opção 3, o programa deve encerrar. Como todos os itens são armazenados em arquivo, ao reiniciar o programa o usuário poderia optar pela opção 2 e visualizar todos os itens cadastrados.

O arquivo deverá ser aberto ou criado no início do programa. O arquivo só poderá ser fechado quando o usuário escolher a opção "sair"no menu.

Observações

Neste exercícios você poderá usar a função `fseek`. Essa função possui o seguinte protótipo:

```
int fseek ( FILE * stream, long int offset, int origin );
```

Em caso de sucesso, a função `fseek` retorna o valor 0.

- `stream` é o arquivo
- `offset` é a quantidade de bytes que o ponteiro de leitura/escrita do arquivo deve ser deslocado
- `origin` é a referência de deslocamento. Sendo `SEEK_SET` o início do arquivo, `SEEK_CUR` a posição atual do ponteiro de leitura/escrita do arquivo e `SEEK_END` o fim do arquivo.

Por exemplo, se você deseja posicionar o ponteiro de leitura/escrita do arquivo `arq` para o 8º *byte*, a chamada da função fica: `fseek( arq, 8, SEEK_SET);`. Caso você deseja posicionar no início do arquivo a chamada poderia ser: `fseek(arq, 0, SEEK_SET);` e `fseek(arq, 0, SEEK_END);` para posicionar no final do arquivo. A chamada `fseek(arq, -2, SEEK_CUR);` retorna 2 *bytes* a partir da posição corrente do ponteiro de leitura/escrita.

2 Vetor econômico (+++)

 (+++)

Suponha o problema de armazenar uma quantidade indefinida de números reais de modo ordenado ocupando o mínimo de memória. Dada a sequência 1.0, 0.1, 20.0, 3.0 , no primeiro momento o número 1.0 deve ocupar a posição 0 do vetor v que ainda não existe. Nesse caso, como somente um número compõe o vetor, é razoável declarar v com 1 elemento apenas. Na segunda inserção, o número 0.1 ocupa a posição v[0] e o número 1.0 passa a ocupar a posição v[1] do vetor, o que exige um redimensionamento do vetor para 2 elementos e a movimentação de valores. Com a entrada do número 20.0, o vetor passa a ter 3 elementos e o valor 20.0 ocupa a posição v[2]. Com a entrada do número 3.0, o vetor precisa ser redimensionado para 4, o valor 3.0 passa a ocupar a posição v[2] e o valor 20.0 é transferido para a posição v[3].

Suponha agora o problema de verificar se um número X está armazenado no vetor. O procedimento é varrer o vetor a partir da posição v[0] até que o seu conteúdo seja menor ou igual a X.

Escreva um programa em linguagem C que permite esse tipo de armazenamento ordenado. O programa deve permitir a inserção e a pesquisa de números. Para isso, o programa deve conter um vetor de números reais que será redimensionado à medida que novos números são inseridos.

O vetor é representado pela estrutura dada no código abaixo. Sua tarefa é implementar as funções declaradas no código.

```c
/**
 * Estrutura que representa um vetor de inteiros.
 */
typedef struct {
   int * d; /** ponteiro para o vetor de inteiros */
   int n;   /** quantidade de elementos inseridos no vetor */
} Vet;

/**
 * Cria uma estrutura Vet com o vetor de elementos devidamente alocado
 * @param n quantidade de elementos a ser alocada
 * @return ponteiro para a estrutura Vet criada
 */
Vet * vet_new( int n );

/**
 * Libera a memória ocupada por vet
 * @param vet Ponteiro para a estrutura Vet
 */
void vet_free( Vet * vet );

/**
 * Função que insere o valor k na estrutura vet de modo ordenado crescente.
 * @param vet Ponteiro para a estrutura Vet
 * @param k valor a ser inserido no vetor
 */
void vet_insert( Vet * vet, int k );

/**
 * Função que verifica se o valor k está presente na estrutura vet.
 * @param vet Ponteiro para a estrutura Vet
 * @param k valor a ser pesquisado no vetor
 * @return posição do vetor em que k aparece pela primeira vez.
 *         Caso k não esteja presente, a função retorna -1.
 */
int vet_search( Vet * vet, int k );
```