TAD, Ordenação e Busca Binária – Mala Direta

Uma empresa de mala direta utiliza um sistema informatizado para auxiliar no gerenciamento de informações sobre os destinatários de suas malas diretas. Esse sistema usa um *TAD Destinatario*. A empresa em questão trabalha apenas com pessoas físicas.

O *TAD Destinatario*, implementado no arquivo *Destinatario.c*, tem a seguinte interface (arquivo *Destinatario.h*):

```
typedef struct destinatario Destinatario;
```

onde se destacam as seguintes funções:

Função	Objetivo
`Destinatario *dest_criarDestinatario (char *nome, int d, int m, int a, char sexo, int enderCep, char *enderNum, char *enderCompl, int instrucao, int profissao);`	Recebe os dados de um destinatário (nome, data de nascimento, sexo, endereço, nível de instrução e profissão) e retorna um ponteiro para um TAD Destinatario que contém estes dados.
`int dest_obterCep (Destinatario *d);`	Retorna o CEP do destinatário. O ponteiro para o destinatário é recebido como parâmetro.
`char *dest_obterNome (Destinatario *d);`	Retorna o nome do destinatário. O ponteiro para o destinatário é recebido como parâmetro. O nome do destinatário é representado como uma cadeia de caracteres.
`void dest_imprimirUmDestinatario (Destinatario *d);`	Imprime, em uma linha, os dados de um destinatário. O ponteiro do destinatário é recebido como parâmetro.

O catálogo dos destinatários de mala direta é implementado como um vetor de ponteiros para o TAD *Destinatario*. As funções específicas do catálogo devem ser implementadas no arquivo *catalogo.c* cujos protótipos das funções devem constar do arquivo *catalogo.h*.

Usando o *TAD Destinatario* disponibilizado:

1. Implemente o arquico *catalogo.h*;
2. Implemente o módulo *catalogo.c*;
3. Complete o código relativo às lacunas assinaladas no módulo *aplicacao.c*.

As funções específicas do catálogo de destinatários, a serem implementadas no arquivo *catalogo.c*, cujas declarações aparecem no arquivo *catalogo.h*, são:

1. Listar os dados de um determinado número de destinatários existentes no catálogo (vetor de ponteiros para TAD *Destinatario*). O parâmetro *`pos_inicial`* indica o índice do primeiro destinatário a ter seus dados listados e o parâmetro *`qtd`* indica quantos destinatários deverão ser listados a partir da posição inicial informada (*`pos_inicial`*). Considere que sempre são recebidos parâmetros válidos.

   ```
   void listarCatalogo (Destinatario **vd, int qtd, int pos_inicial);
   ```

2. Ordenar o vetor de ponteiros para TAD *Destinatario* em ordem decrescente de CEP e em ordem crescente de nome:

   ```
   void ordenarCatalogo (Destinatario **vd, int n);
   ```

3. Usando busca binária, retornar o índice do primeiro destinatário (considerando nomes em ordem alfabética) que tenha o CEP informado e a quantidade de destinatários associados a este CEP, através do ponteiro *qtd_dest*. Caso o CEP desejado não apareça no catálogo, a função deve retornar -1 como índice e 0 como quantidade de destinatários:

```
int buscarDestinatario(Destinatario **vd, int n, int cep, int *qtd_dest);
```

A aplicação (programa principal) implementada no módulo *aplicacao.c*, usando serviços do catálogo de destinatários e funções do *TAD Destinatario*, deve fazer o seguinte:

1. Listar todos os produtos do catálogo de destinatários;
2. Ordenar o catálogo de destinatários em ordem decrescente de CEP e, para o mesmo CEP, em ordem alfabética de nome;
3. Listar todos os produtos do catálogo de destinatários, após a ordenação;
4. Ler um CEP do teclado;
5. Buscar o primeiro destinatário associado a esse CEP, considerando que os nomes estão em ordem alfabética;
6. Listar os dados de todos os destinatários que estejam associados ao CEP informado.

OBS:
1. Só podem ser usadas técnicas de ordenação apresentadas no curso (**NÃO pode usar *qsort***)
2. Para a busca, tem que ser utilizada a técnica de busca binária (**NÃO pode usar *bsearch***)

Listagem do código disponibilizado

```
/*-----------------*/
/* Destinatario.h  */
/*-----------------*/

typedef struct destinatario Destinatario;

Destinatario *dest_criarDestinatario (char *nome, int d, int m, int a, char
sexo, int enderCep, char *enderNum, char *enderCompl, int instrucao, int
profissao);
int     dest_obterCep (Destinatario *d);
char   *dest_obterNome (Destinatario *d);
void    dest_imprimirUmDestinatario (Destinatario *d);
void    dest_liberarDestinatario (Destinatario *d);
```

```c
/*-----------------*/
/* aplicacao.c         */
/*-----------------*/

/* Preencha o trecho a seguir com os includes necessarios */
/******************/


/******************/

#define N 10

int main(void)
{
      int i;
      Destinatario *vpd[N];
      vpd[0] = dest_criarDestinatario("tio patinhas", 13, 1, 1930,
'm', 1000, "120", "casa", 0, 0);
      vpd[1] = dest_criarDestinatario("harry potter", 25, 12, 1990,
'm', 7000, "1", "fundos", 0, 0);
      vpd[2] = dest_criarDestinatario("pato donald", 6, 6, 1950,
'm', 1000, "120", "fundos", 0, 0);
      vpd[3] = dest_criarDestinatario("pica-pau", 7, 5, 1960,
'm', 9700, "356", "apto 102", 0, 0);
      vpd[4] = dest_criarDestinatario("clarabela", 6, 11, 1950,
'f', 700, "102", "lote 15", 0, 0);
      vpd[5] = dest_criarDestinatario("margarida", 25, 10, 1962,
'f', 1000, "10", "casa", 0, 0);
      vpd[6] = dest_criarDestinatario("power ranger", 7, 3, 2000,
'm', 20000, "670", "gruta azul", 0, 0);
      vpd[7] = dest_criarDestinatario("huguinho", 11, 11, 1987,
'm', 1000, "530", "casa B", 0, 0);
      vpd[8] = dest_criarDestinatario("luizinho", 11, 11, 1987,
'm', 1000, "530", "casa B", 0, 0);
      vpd[9] = dest_criarDestinatario("sininho", 1, 1, 2000,
'f', 0005, "6", "casa", 0, 0);

      /* Preencha o trecho a seguir com
         a funcionalidade solicitada no enunciado e
         as declaracoes que se facam necessarias     */
      /******************/



      /******************/

      for(i = 0; i < N; i++)
            dest_liberarDestinatario (vpd[i]);
      return 0;
}
```

```c
/*-----------------*/
/* Destinatario.c  */
/*-----------------*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "Destinatario.h"
```

```c
struct tdata {
	int dia, mes, ano;
} ;
typedef struct tdata Tdata;

struct tendereco {
	char numero[31];
	char complemento [51];
	int  cep;
} ;
typedef struct tendereco Tendereco;

struct destinatario
{
	char    destNome [101];
	char    destSexo;
	int           destProfissao;
	int           destInstrucao;
	Tdata   destNasc;
	Tendereco destEndereco;
};

Destinatario *dest_criarDestinatario (char *nome, int dia, int m, int a, char
sexo, int enderCep, char *enderNum, char *enderCompl, int instrucao, int
profissao)
{
	Destinatario *d;
	d = (Destinatario *) malloc (sizeof(Destinatario));
	if (d==NULL)
		exit (1);
	strcpy(d->destNome, nome);
	d->destSexo = sexo;
	d->destProfissao = profissao;
	d->destInstrucao = instrucao;
	d->destNasc.dia = dia;
	d->destNasc.mes = m;
	d->destNasc.ano = a;
	d->destEndereco.cep = enderCep;
	strcpy(d->destEndereco.numero, enderNum);
	strcpy(d->destEndereco.complemento, enderCompl);
	return d;
}

int dest_obterCep (Destinatario *d)
{
	return d->destEndereco.cep;

}
char *dest_obterNome (Destinatario *d)
{
	return d->destNome;
}

void dest_imprimirUmDestinatario (Destinatario *d)
{
	printf("destinatario %-30s: %c, %d, %d, %s - %s\n",
		d->destNome, d->destSexo, d->destProfissao, d->destInstrucao,
		d->destEndereco.numero, d->destEndereco.complemento);
}

void dest_liberarDestinatario (Destinatario *d)
{
	free(d);
}
```