TAD - Produto

Uma empresa de vendas pela Internet utiliza um *TAD Produto* em seus sistemas.

O TAD *Produto* tem a seguinte interface (arquivo *Produto.h*):

```
typedef struct produto Produto;
```

onde se destacam as seguintes funções:

Função	Objetivo
`Produto *prod_criarProduto (char *nome, int cod, float preco, char *categoria);`	Recebe os dados de um produto (nome, código, preço e categoria) e retorna um ponteiro para um *TAD Produto* que contém estes dados.
`char *prod_obterCategoria (Produto *p);`	Retorna a categoria do produto. O ponteiro para o produto é recebido como parâmetro. A categoria é representada como uma cadeia de caracteres.
`float prod_obterPreco(Produto *p);`	Retorna o preço do produto. O ponteiro para o produto é recebido como parâmetro.
`char *prod_obterNome (Produto *p);`	Retorna o nome do produto. O ponteiro para o produto é recebido como parâmetro. O nome do produto é representado como uma cadeia de caracteres.
`void prod_imprimirUmProduto (Produto *p);`	Imprime, em uma linha, os dados de um produto. O ponteiro do produto é recebido como parâmetro.

Para guardar os vários produtos de seu catálogo, a empresa quer usar um vetor de ponteiros para o *TAD Produto* e deseja que seja implementado um módulo especial (arquivo *catalogo.c*) com os seguintes serviços:

1. Montar dinamicamente o vetor de ponteiros para TAD *Produto* de tamanho *n* e preenche-lo com os dados existentes nos vetores *vetProduto* e *vetPreco* (ambos com tamanho *n*). O código do produto corresponde a sua posição no vetor *vetProduto*, começando em 1 (o 1º produto do vetor tem código 1, o 2º tem código 2 e assim sucessivamente):

   ```
   Produto **carregarVetor (Item *vetProduto, float *vetPreco, int n);
   ```

   onde *Item* corresponde ao tipo estruturado descrito abaixo.

   ```
   struct item
   {
         char nome_prod[51];
         char cat_prod [4];
   };
   typedef struct item Item;
   ```

2. Listar todos os produtos do vetor de ponteiros para TAD *Produto*:

   ```
   void listarVetor (Produto **vp, int n);
   ```

3. Ordenar o vetor de ponteiros para TAD *Produto* em ordem decrescente de categoria e ordem crescente de preço:

   ```
   void ordenarVetor (Produto **vp, int n);
   ```

4. Buscar, usando busca binária, o produto mais barato de uma determinada categoria informada, isto é, retorna o endereço do 1º produto que aparecer no vetor ordenado que seja da categoria informada, ou NULL, caso a categoria desejada não apareça no vetor:

   ```
   Produto *buscarProdutoMaisBarato(Produto **vp, int n, char *cat);
   ```

TAD - Produto

Os serviços que tratam do catálogo de produtos (vetor de ponteiros para o *TAD Produto*) devem ter suas interfaces descritas no arquivo *catalogo.h* e devem ser implementados no arquivo *catalogo.c*.

A empresa também deseja que seja implementada uma aplicação (um programa principal) no módulo *prog.c* que, usando serviços do catálogo de produtos e funções do *TAD Produto*, faça o seguinte:

1. Monte dinamicamente o catálogo de produtos (vetor de ponteiros para *TAD Produto*) a partir dos dados dos vetores de produtos e preços, já existentes;
2. Liste todos os produtos do catálogo de produtos;
3. Ordene o catálogo de produtos em ordem decrescente de categoria e, para a mesma categoria, em ordem crescente de preço;
4. Liste todos os produtos do catálogo **ordenado** de produtos;
5. Leia do teclado uma categoria;
6. Busque o produto mais barato da categoria informada;
7. Imprima o nome e preço desse produto.

Com essas considerações e usando o *TAD Produto* disponibilizado:

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

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

```
/*-----------------*/
/* Produto.h       */
/*-----------------*/

typedef struct produto Produto;

Produto *prod_criarProduto (char *nome, int cod, float preco, char *categoria);
char    *prod_obterCategoria (Produto *p);
float    prod_obterPreco(Produto *p);
char    *prod_obterNome (Produto *p);
void     prod_imprimirUmProduto (Produto *p);
void     prod_liberarProduto (Produto *p);
```

```
/*-----------------*/
/* prog.c          */
/*-----------------*/

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


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

#define N 13

int main(void)
{
      Item vetProdutos [N] = {
            {"Caneta BIC","Pap"},
            {"Nescau light","Ali"},
            {"Pijama Mena","Ves"},
            {"Caneta Parker ","Pap"},
            {"Panela pressao marmicoc","Uti"},
            {"T-Shirt Hering","Ves"},
            {"Bis","Ali"},
            {"Biscoito Maizena","Ali"},
            {"Borracha TK","Pap"},
            {"C Completo","Liv"},
            {"Conj talheres San Remo","Uti"},
            {"Trakinas","Ali"},
            {"CD Basf","Inf"} };
      float vetPrecos [N] =
            {1.50f, 4.5f, 53.00f, 78.0f, 145.99f,
             25f, 2.5f, 1.99f, 13f, 58f,
             25f, 2.5f, 5f};
      Produto **p;
      int i;

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



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

      for(i = 0; i < N; i++)
            prod_liberarProduto (p[i]);
      return 0;
}
```

```c
/*-----------------*/
/* Produto.c       */
/*-----------------*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "Produto.h"

struct produto
{
      int   codigoProduto;
      char  nomeProduto[51];
      char  categoriaProduto[4];
      float precoProduto;
};

Produto *prod_criarProduto (char *nome, int cod, float preco, char *categoria)
{
      Produto *p;

      p = (Produto *) malloc (sizeof(Produto));
      if (p==NULL)
            exit (1);

      strcpy(p->categoriaProduto, categoria);
      strcpy(p->nomeProduto, nome);
      p->codigoProduto = cod;
      p->precoProduto = preco;

      return p;
}

char *prod_obterCategoria (Produto *p)
{
      return p->categoriaProduto;
}

float prod_obterPreco(Produto *p)
{
      return p->precoProduto;
}

char *prod_obterNome (Produto *p)
{
      return p->nomeProduto;
}

void  prod_imprimirUmProduto (Produto *p)
{
      printf("produto: %s - %d - %s - %5.2f",
            p->categoriaProduto, p->codigoProduto, p->nomeProduto, p->precoProduto);
}

void  prod_liberarProduto (Produto *p)
{
      free(p);
}
```