(2021-1) P1-Ex03

Desenvolver um TAD para representar um estoque de produtos. O TAD deve gerenciar as seguintes informações do produto:

- **descrição: descrição atribuída ao estoque**
- **vetor: vetor de endereços dos produtos que pertencem ao estoque**

As funcionalidades a ser implementadas são:

- **Criar um estoque**
- **Destruir um estoque**
- **Anexar um produto ao estoque**
- **Remover um produto do estoque pela descrição**
- **Imprimir a lista de produtos do estoque no seguinte padrão:**

  [(1,produtoA,1.0),(2,produtoB,2.0),(3,produtoC,3.0)]

TAREFA

Desenvolver as funções contidas no arquivo tad_estoque.c. Submeta essa arquivo no run.codes

Observação importante sobre o TAD Produto

Você poderá utilizar todas as funções do TAD Produto descritas no arquivo tad_produto.h. Lembre-se que os dados da struct estão encapsulados. Portanto, você somente poderá acessar os dados do produto por meio das funções disponibilizadas.

Além disso, você não terá acesso ao código correspondente a implementação do TAD produto. Estou disponibilizando o arquivo compilado da implementação tad_produto.o

Arquivos

tad_estoque.h

```c
#ifndef _TAD_ESTOQUE_
#define _TAD_ESTOQUE_

#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "tad_produto.h"

/*****************************************
 * DADOS
*****************************************/
typedef struct estoque Estoque;
```

```c
/*****************************************
 * OPERAÇÕES
*****************************************/

/**
 * Cria dinamicamente um estoque de produtos e preenche seus atributos
 * Parâmetro descricao: Descrição do estoque a ser criado
 * RETORNO: endereço da struct Estoque criada
 */
Estoque* estoque_criar(char* descricao);

/**
 * Desaloca a struct Estoque criada
 * Parâmetro endEstoque: Endereço da variável que contém o ponteiro para o Estoque a ser desalocado.
 * A função também deve limpar o lixo de memória no endereço da variável recebido por parâmetro
 */
void estoque_destruir(Estoque** endEstoque);

/**
 * Insere um produto no final do vetor
 * Parâmetro e: Ponteiro para a struct Estoque
 * Parâmetro p: Ponteiro para a struct Produto
 * RETORNO: true se a inserção ocorreu com sucesso e false caso contrário
 */
bool estoque_anexar(Estoque* e, Produto* p);
```

```c
 /**
  * Remove um produto com a descricao especificada
  * Parâmetro e: Ponteiro para a struct Estoque
  * Parâmetro descricao: Descrição do produto a ser removido
  * RETORNO: índice do produto removido ou -1 caso contrário
  */
 int estoque_remover_elemento(Estoque* e, char* descricao);

 /**
  * Imprime a lista de produtos no estoque com o seguinte padrao
  * [(1,produtoA,5.5),(2,produtoB,7.7),(3,produtoC,2.1)]
  * Parâmetro e: Ponteiro para a struct Estoque
  */
 void estoque_imprimir(Estoque* e);

 #endif
```

tad_estoque.c

```c
#include "tad_estoque.h"

#define TAM 100

struct estoque{
    char descricao[50];
    Produto* vetor[TAM];
    int qtde;
};


Estoque* estoque_criar(char* descricao){

}
void estoque_destruir(Estoque** endEstoque){

}

bool estoque_anexar(Estoque* e, Produto* p){

}

int estoque_remover_elemento(Estoque* e, char* descricao){

}
```

```c
void estoque_imprimir(Estoque* e){
    // printf("[");

    // printf("(%d,%s,%.2f)", ..., ..., ...);

    // printf("]");
}
```

main.c

Faça os seus testes aqui.

```c
#include "tad_estoque.h"
#include "tad_produto.h"


int main(){

    Estoque* e = estoque_criar("estoque");
    Produto* p1 = produto_criar(1, "produtoA", 1.0);
    Produto* p2 = produto_criar(2, "produtoB", 2.0);
    Produto* p3 = produto_criar(3, "produtoB", 3.0);

    estoque_anexar(e, p1);
    estoque_anexar(e, p2);
    estoque_anexar(e, p3);

    estoque_imprimir(e);

    return 0;
}
```

makefile

```
all:
  gcc -c tad_estoque.c
  gcc main.c tad_estoque.o tad_produto.o -o main
  ./main
```