

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

#define MAX_NOME_PRODUTO 50
typedef int TIPOCHAVE;

typedef struct PRODUTO {
    TIPOCHAVE chave;
    char nome[MAX_NOME_PRODUTO];
    int quantidade;
    float preco;
} PRODUTO;

typedef struct aux {
    PRODUTO prod;
    struct aux *prox;
} ELEMENTO;

ELEMENTO;
ELEMENTO *PONT;

typedef struct {
    PONT topo;
} PILHA;

// Protótipos das funções
void inserir(PILHA *P);
void verificarTamanho(PILHA *P);
void consultarElemento(PILHA *P, const CHAR *nome);
```

```
Void inserirElementoComID (PILHA * P, TIPOCHAVE id, const char  
* nome, int quantidade, float preco);  
Void alterarElemento (PILHA * P, const char * nome);  
Void excluirElemento (PILHA * P);  
Void salvoElemento (PILHA * P);  
Void reiniciarEstoque (PILHA * P);  
Void corregerEstoque (PILHA * P);
```

// Função principal

```
int main () {
```

```
PILHA Estoque;
```

```
char nomeProduto [MAX_NOME_PRODUTO];  
char nomePesquisa [MAX_NOME_PRODUTO];  
char opcaoApagar;  
int IDProduto;  
int quantProduto;  
float precoProduto;  
  
int operao;
```

~~criarEstoque~~

```
criarEstoque (&Estoque);
```

```
WHILE (1) {
```

Print ("\\n===== Controle de Estoque Hardw & Driphards  
===== \\n");

```
printf ("1. Inserir Produto\n");
printf ("2. Excluir Produto\n");
printf ("3. Alterar Produto\n");
printf ("4. Exibir Produto\n");
printf ("5. Consultar Produto\n");
printf ("6. Limpar estoque\n");
printf ("7. Salvar Estoques\n");
printf ("8. Corrigir estoques\n");
printf ("0. Sair\n\n");
printf("GTD PROD ESTE :%i\n", verificarEstoque(&estoque));
printf ("Escolha uma opção:");
scanf ("%d", &opcao);
```

switch (opcao) {

case 1:

```
    printf ("ID do produto: ");
    scanf ("%d", &IDProduto);
    printf ("Nome do produto: ");
    scanf ("%[^ \n]", nomeProduto);
    printf ("Quantidade: ");
    scanf ("%d", &quantProduto);
    printf ("Preço: ");
    scanf ("%f", &precoProduto);
    inserirEstoqueComID (&estoque, IDProduto, nomeProduto,
    quantProduto, precoProduto);
    break;
```

case 2:

```
if (verificarTomanhotoEstq (& estoque) == 0) {  
    printf ("\nEstoque vazio!\n");  
    break;  
}
```

```
} else {
```

```
    printf ("Confirmar? Esta ação irá pagar o  
ultimo elemento do estoque (%d): ");  
    scanf ("%d", & opcaoPagar);  
    if (opcaoPagar (& opcaoPagar, "s") == 0) {  
        excluirElementoEstq (& estoque);  
    }
```

```
}
```

```
}
```

```
break;
```

case 3:

```
if (verificarTamanhotoEstq (& estoque) == 0) {  
    printf ("\nEstoque vazio!\n");  
    break;  
}
```

```
} else {
```

```
    printf ("\nQual produto deseja modificar? ");  
    scanf ("%s", nomeProduto);  
    alterarElementoEstq (& estoque, nomeProduto);
```

```
}
```

```
break;
```

case 4:

```
if (verificarTamanhotoEstq (& estoque) == 0) {  
    printf ("\nEstoque vazio!\n");  
    break;  
}
```

```
} else {
```

consultarElemento(&estoque);

3

break;

case 5:

if (verificarTamanhosEstq(&estoque) == 0) {  
 printf("\nEstoque vazio!\n");  
 break;

3 else {

printf("Nome do produto: ");

scanf("%s", nomePesquisa);

consultarElemento(&estoque, nomePesquisa);

3

break;

case 6:

if (verificarTamanhosEstq(&estoque) == 0) {  
 printf("\nEstoque vazio!\n");  
 break;

} else {

reiniciarEstq(&estoque);

3

break;

case 7:

salvarEstq(&estoque);

printf("\nEstoque salvo!");

break;

case 8:

corregarEstq(&estoque);

break;

case 9:

printf("\nPrograma encerrado.\n");



```
    return &;
```

```
default:
```

```
    printf("Opção inválida. Tente novamente. (n);  
    break;
```

```
}
```

```
}
```

```
return &;
```

```
}
```

```
Void criarEstq (PILHA *P) { P->topo = NULL; }
```

```
PONT end = P->topo;
```

```
printf ("\nProdutos: ) m");
```

```
while (end != NULL) {
```

```
    printf ("m > %s < \n", end->prod.nome);
```

```
    printf ("ID: %d \n", end->prod.chave);
```

```
    printf ("NOME: %s \n", end->prod.nome);
```

```
    printf ("QUANT: %d \n", end->prod.quantidade);
```

```
    printf ("PREÇO: R$ %f \n \n", end->prod.preco);
```

```
    end = end->prox;
```

```
}
```

```
printf ("\n) m");
```

```
}
```

```
Void consultarElemento (PILHA *P, const char *nome) {
```

```
PONT end = P->topo;
```

```
while (end != NULL) {
```

```
if (strcmp(end->prod.nome, nome) == 0) {
```

```
    printf ("\nProduto: \n");
```

```
    printf ("m > %s < \n", end->prod.nome);
```



```
    printf("ID: %i\n", end->prod. chave);  
    printf("NAME: %s\n", end->prod. nome);  
    printf("QUANT: %d\n", end->prod. quantidade);  
    printf("PRECO: R$%f\n", end->prod. preco);  
    return;  
}  
end = end->prox;
```

```
}  
printf("Produto não encontrado\n");
```

```
Void inserirElementoComID(PILHA *P, TIPOCHAVE id, const  
* nome, int quantidade, float preco) {  
    PONT novoProd = (PONT) malloc(sizeof(ELEMENTO));  
    novoProd->prod. chave = id;  
    strcpy(novoProd->prod. nome, nome);  
    novoProd->prod. quantidade = quantidade;  
    novoProd->prod. preco = preco;  
    novoProd->prox = P->topo;  
    P->topo = novoProd;
```

```
}  
Void inserirElementoComID(PILHA *P, TIPOCHAVE id, con-  
char * nome, int quantidade, float preco) {  
    PONT novoProd = (PONT) malloc(sizeof(ELEMENTO));  
    novoProd->prod. chave = id;  
    strcpy(novoProd->prod. nome, nome);  
    novoProd->prod. quantidade = quantidade;  
    novoProd->prod. preco = preco;  
    novoProd->prox = P->topo;  
    P->topo = novoProd;
```

```
Koid alterarElemento (PILHA *p, const char * nome) {  
    PONT end = p-> topo;  
    int opcaoMudar;  
    char novoNome[MAX_NAME_PROTO];  
    int novoQuant;  
    float novoPreco;
```

```
    while (end != NULL) {  
        if (strcmp(end->prod.nome, nome) == 0) {  
            while (opcaoMudar != 4) {  
                printf ("\n") nO que deseja alterar? \n");  
                printf ("1. Nome \n");  
                printf ("2. Quantidade \n");  
                printf ("3. Preco \n");  
                printf ("4. Sair \n");  
                printf (">> ");  
                scanf ("%i", &opcaoMudar);  
                switch (opcaoMudar) {
```

case 1:

```
                    printf ("\n") nNovo nome: ");  
                    scanf ("%s", novoNome);  
                    strcpy(end->prod.nome, novoNome);  
                    break;
```

case 2:

```
                    printf ("\n") nNova quantidade: ");  
                    scanf ("%d", &novoQuant);  
                    end->prod.quantidade = novoQuant;  
                    break;
```

case 3:

```
printf ("\nNovo preço: ");
scanf ("%f", & novoPreco);
end -> prox_preco = novoPreco;
break;
```

case 4:

```
printf ("\n");
return;
```

default:

```
printf ("Opção inválida. Tente novamente.");
break;
```

}

}

break;

}

end = end -> prox;

}

```
printf ("\nProduto não encontrado (%s)");
}
```

Void excluirElemento (PILHA \* P) {

PONT end = P -> topo;

end -> prox = P -> topo -> prox;

PONT apagar = P -> topo;

P -> topo = P -> topo -> prox;

free (apagar);

}

```

Void salvarEstq (PILHA * P) {
    char nomeArquivo [50];
    printf ("Insira o nome do arquivo: ");
    scanf ("%99s", nomeArquivo);
    getch ();
    nomeArquivo [strcspn (nomeArquivo, "\n")]=0;
    strcat (nomeArquivo, ".txt");
    FILE *file = fopen (nomeArquivo, "w");
    if (file == NULL) {
        printf ("Erro ao abrir o arquivo.\n");
        return;
    }

    PONT end = P-> topo;
    while (end != NULL) {
        fprintf (file, "Produto: %s, Quantidade: %.
        Preco: %.2f\n", end->prod.nome, end->prod.quantidade,
        end->prod.preco);
        end = end->prox;
    }

    fclose (file);
    reinicializarEstq (P);
}

```

```
Void reinicializarEstq (PILHA * P) {
```

```
    PONT qfragar;
```

```
    PONT posicao != NULL) {
```

```
        qfragar = posicao;
```

```
        posicao = posicao -> prox;
```

```
        free (qfragar);
```

```
}
```

```
    P-> topo = NULL;
```

```
}
```

```
Void corregerEstq (PILHA * P) {
```

```
    char nomeArquivo[100];
```

```
    printf ("Insira o nome do arquivo a ser  
corrigido ( incluindo a extensão .txt): ");
```

```
    scanf ("% 99s", nomeArquivo);
```

```
    getch();
```

```
FILE *file = fopen (nomeArquivo, "r");
```

```
if (file == NULL) {
```

```
    printf ("Erro ao abrir o arquivo.\n");  
    return;
```

```
}
```

```
reinicializarEstq (P);
```

```
PROTO prot;
```

```
TPOCHAVE id = Q;
```

while (frecoy (f16, "Produto: %49[^,], Quantidade:  
%d, Preço: %F\n", prod.nome, &prod.quantidade,  
&prod.preco) == 3) {

Port novoProd = (PORT) malloc (sizeof(ELEMENTO));  
novoProd->prod.chave = id++;

strcpy (novoProd->prod.nome, prod.nome);  
novoProd->prod.Preco = prod.Preco;

novoProd->prox = p->Topo;

p->Topo = novoProd;

}

Eclose (f16);

printf ("Estoque corrigido com sucesso!\n");

}