Fatec Ipiranga – Pastor Éneas Tognini – ADS 2 Semestre Vespertino

Disciplina: Programação Estruturada e Modular **| Prof.** Carlos Verissimo

Objetivo: Documento com análise crítica e programa refatorado para N2-7

Autor: Kauan Torres

Data: 23/11/2024

Análise crítica código 13

O programa está funcionando, cumprindo com os requisitos, utilizou-se Struct e imprimiu os dados do produto com ponteiros, além de realizar uma função para cada CRUD. Todas as funcionalidades estão funcionais caso siga o ‘’caminho feliz’’, porém, existem erros e falhas caso siga caminhos alternativos. Há uma série de melhorias que podem ser feitas para o código ficar mais limpo e melhor estruturado, ajustando algumas falhas que possam ser causadas por caminhos alternativos que o usuário siga.

1. O programa está modularizado, com funções para cada opção, entretanto as funções não estão com comentários marcados mostrando a modularização:
   ANTES:

```c
void cadastrarProduto(Produto *listaProdutos, int *contadorProduto){
    Produto *produto = &listaProdutos[*contadorProduto];
    produto -> ID = *contadorProduto + 1;
    printf("\nDigite o nome do produto: ");
    scanf(" %[^\n]", produto -> nomeProduto);
    printf("Digite a quantidade em estoque do produto (Somente numeros naturais): ");
    scanf(" %i", &produto -> quantidadeEmEstoque);
    while(produto -> quantidadeEmEstoque < 0){
        printf("A quantidade digitada esta errada. So eh possivel existir numeros positivos d
        printf("Digite novamente: ");
        scanf(" %d", &produto -> quantidadeEmEstoque);
    }
    printf("Digite o valor do produto: ");
    scanf(" %lf", &produto -> valorDoProduto);
    while(produto -> valorDoProduto < 0){
        printf("O valor digitado esta errado. So eh possivel adicionar valores positivos para
        printf("Digite novamente: ");
        scanf(" %d", &produto -> quantidadeEmEstoque);
    }

    (*contadorProduto)++;
    printf("Produto adicionado com sucesso!\n\n");
}

void imprimirDados(Produto *listaProdutos){
    printf("\nID: %d\n", listaProdutos -> ID);
    printf("Nome: %s\n", listaProdutos -> nomeProduto);
    printf("Quantidade em estoque: %d\n", listaProdutos -> quantidadeEmEstoque);
    printf("Valor do produto: %.2f\n\n", listaProdutos -> valorDoProduto);
}
```

O que dificulta uma visualização mais limpa do código, então, o ideal para a modularização seria a utilização de comentários maiores, como no exemplo abaixo, que separa em módulos comentados cada função:
DEPOIS:

```
/*-------------------------------------------------------------*
| Módulo - Modulo cadastro produto                             |
*-------------------------------------------------------------*/

void cadastrarProduto(Produto *listaProdutos, int *contadorProduto){
    Produto *produto = &listaProdutos[*contadorProduto];
    produto -> ID = *contadorProduto + 1;
    printf("\nDigite o nome do produto: ");
    scanf(" %[^\n]", produto -> nomeProduto);
    printf("Digite a quantidade em estoque do produto (Somente numeros naturais): ");
    scanf(" %i", &produto -> quantidadeEmEstoque);
    while(produto -> quantidadeEmEstoque < 0){
        printf("A quantidade digitada esta errada. So eh possivel existir numeros positivos
        printf("Digite novamente: ");
        scanf(" %d", &produto -> quantidadeEmEstoque);
    }
    printf("Digite o valor do produto: ");
    scanf(" %lf", &produto -> valorDoProduto);
    while(produto -> valorDoProduto < 0){
        printf("O valor digitado esta errado. So eh possivel adicionar valores positivos par
        printf("Digite novamente: ");
        scanf(" %d", &produto -> quantidadeEmEstoque);
    }

    (*contadorProduto)++;
    printf("Produto adicionado com sucesso!\n\n");
}

/*-------------------------------------------------------------*
| Módulo - Modulo imprimir dados                               |
*-------------------------------------------------------------*/

void imprimirDados(Produto *listaProdutos){
    printf("\nID: %d\n", listaProdutos -> ID);
    printf("Nome: %s\n", listaProdutos -> nomeProduto);
    printf("Quantidade em estoque: %d\n", listaProdutos -> quantidadeEmEstoque);
    printf("Valor do produto: %.2f\n\n", listaProdutos -> valorDoProduto);
}
```

---

2. Na seleção da opção, o programa traz diversos printf e scanf direto na main, o que poderia ir direto para a função, fazendo com o código fique mais encapsulado. **ANTES:**

```
switch(opcao){
    case 1: cadastrarProduto(listaProdutos, &contadorProduto);
            break;

    case 2: printf("\nDigite o ID do produto que será alterado: ");
            scanf(" %d", &ID);
            alterarProduto(listaProdutos, ID, contadorProduto);
            break;

    case 3: printf("\nDigite o ID do produto que deseja consultar: ");
            scanf(" %d", &ID);
            consultarProduto(listaProdutos, ID, contadorProduto);
            break;

    case 4: printf("\nDigite o ID do produto que deseja excluir: ");
            scanf(" %d", &ID);
            excluirProduto(listaProdutos, ID, &contadorProduto);
            break;

    case 5: imprimirLista(listaProdutos, contadorProduto);
            break;
```

Atualizado, ficando assim só a chamada da função:

DEPOIS:

```
switch(opcao){
    case 1: cadastrarProduto(listaProdutos, &contadorProduto);
            break;

    case 2:
            alterarProduto(listaProdutos, contadorProduto);
            break;

    case 3:
            consultarProduto(listaProdutos, contadorProduto);
            break;

    case 4:
            excluirProduto(listaProdutos, &contadorProduto);
            break;

    case 5: imprimirLista(listaProdutos, contadorProduto);
            break;
```

---

3. Ao visualizar a lista de produtos quando não há nenhum produto cadastrado, mesmo assim a lista aparece:

   ANTES:

```
------------- Lista de todos os produtos cadastrados -------------


------ * MENU * ------
1. Cadastrar produto
2. Alterar dados do produto
3. Consultar produto
4. Excluir Produto
5. Consultar lista de produtos
6. Vender produto
7. Dar desconto a um produto
8. Encerrar o programa
Escolha uma opcao:
```

Podendo ser alterado para uma mensagem dizendo que não há produtos cadastrados, isto na função imprimirlista, através de uma verificação.

DEPOIS:

```
Nenhum produto cadastrado no momento.

------ * MENU * ------
1. Cadastrar produto
2. Alterar dados do produto
3. Consultar produto
4. Excluir Produto
5. Consultar lista de produtos
6. Vender produto
7. Dar desconto a um produto
8. Encerrar o programa
Escolha uma opcao:
```

Da mesma forma, quando tenta-se vender um item que não existe, mesmo assim ele dá a opção para digitar a quantidade que será vendida, o que pode ser anulado caso não haja produto. Também havia um erro que quando colocava para comprar um estoque maior que o item, ele dizia que não havia estoque o suficiente, mas também mostrava a mensagem “ID x não encontrado”, as duas ao mesmo tempo. Alterando essas opções, fica assim:

```
Digite o ID do produto que deseja excluir: 1
Produto com o ID 1 nao encontrado.
```

---

4. O código não está comentado, o que é importante para outros desenvolvedores conseguirem entender sua aplicação, e para esta tarefa solicitei ajuda da IA para ressaltar os principais pontos do código.
5. No momento de alterar algum produto, caso o usuário queira alterar somente o valor, ele terá que alterar novamente o nome, quantidade e valor, então seria interessante deixar uma opção para o usuário selecionar o que ele quer alterar.
   ANTES:

```
Dados atuais do produto:

ID: 1
Nome: mouse
Quantidade em estoque: 23
Valor do produto: 2.00


Digite o nome do produto: teclado
Digite a quantidade em estoque do produto: 2
Digite o valor do produto: 1
Produto alterado com sucesso!
```

DEPOIS:

```
Dados atuais do produto:

ID: 2
Nome: teclado
Quantidade em estoque: 5
Valor do produto: 1.00


O que você deseja alterar?
1. Nome
2. Quantidade em estoque
3. Valor
Digite a opção desejada (1/2/3): 3
Digite o novo valor do produto: 1
Produto alterado com sucesso!
```

---

6. No momento da listagem dos produtos, ele não mostra valor, somente ID e nome, seria interessante já mostrar todas as informações do produto.
   ANTES:

```
------------ Lista de todos os produtos cadastrados ------------

ID: 1       Nome: mouse
ID: 2       Nome: teclado
```

   DEPOIS:

```
------------ Lista de todos os produtos cadastrados ------------

ID: 1
Nome: mouse
Quantidade em estoque: 2
Valor: 1.00

ID: 2
Nome: teclado
Quantidade em estoque: 23
Valor: 55.00
```