Vinicius Leon Melo Tavares | PEM Desafio N2-7 (Análise Crítica) | 22/11/2024

O código examinado funciona como esperado no “caminho feliz”, porém algumas brechas puderam ser observadas durante a investigação. Neste documento, também há levantamentos na perspectiva de manutenção e desenvolvimento.

Modularização

- O código apresenta uma modularização satisfatória, com funções separadas para operações CRUD e outras funcionalidades específicas, como listagem e posicionamento de produtos. Contudo, há pontos que demandam aprimoramento:

- Observou-se um uso desnecessário de `goto` para retornar a pontos anteriores no fluxo. Isso foi corrigido com a substituição por estruturas de repetição (`while` ou `do-while`), melhorando a legibilidade e manutenção.

- A `struct produto` não segue convenções de nomenclatura comuns em C. Sugere-se o uso de `Produto` (CamelCase) ou um formato mais padronizado como `produto_t` para indicar ser um tipo.

- A ausência de `typedef` exige o uso explícito da palavra-chave `struct`, o que aumenta a verbosidade. A inclusão de `typedef struct` simplificaria as declarações. - A utilização de macros para definir limites em tempo de compilação (e.g., tamanho de nomes e número máximo de produtos) está correta. Entretanto, a diretiva `INICIO` é redundante, pois representa uma constante que dificilmente será alterada.

- A função `menuOpcoes` centraliza o tratamento de entradas inválidas, mas deixa o `switch-case` no `main` dependente de lógica dispersa. A unificação do tratamento seria mais clara e consistente.

- Funções individuais tratam seus próprios erros, mas a `main` tenta capturá-los desnecessariamente. Isso cria redundância e aumenta a complexidade do código.

Percentual de adesão: 80%

Elementos Conceituais

- A estrutura `produto` encapsula corretamente os dados essenciais (código, nome, quantidade e preço), atendendo ao requisito.

- Contudo, o uso de `float` para preços contradiz a necessidade de maior precisão, que poderia ser alcançada com `double`.

- Variáveis globais, como `ch` para limpar buffers, são desnecessárias. A utilização de ponteiros em algumas funções é válida, mas nem sempre eficiente, como ao passar o número de produtos via ponteiro.

- Aritmética de ponteiros para acessar elementos de vetores torna o código menos intuitivo. Substituí-la por índices melhora a legibilidade.

- A variável `loop` no `main` é desnecessária, pois alternativas como `while (1)` ou `do-while` com condições de quebra são mais concisas.

- O método para marcar exclusões, movendo produtos excluídos para o início do vetor e deslocando os demais, é ineficiente para grandes volumes de dados. Adicionar um atributo que indique o estado ativo do produto seria mais eficiente, mesmo com maior uso de memória.

- Funções como `strcpy` são usadas de forma redundante, quando atribuições diretas poderiam ser realizadas.

Percentual de adesão: 40%

Elementos de Negócio

- Não há validação robusta para evitar entradas inválidas, como caracteres em campos numéricos, ou para tratar buffers excedentes no uso de `fgets`.

- A função `incluirProduto` gera automaticamente o código do produto, mas o requisito especifica que deveria ser definido pelo usuário. Também falta validação para garantir unicidade do código.

- Implementada de forma funcional na `alterarProduto`, permitindo a edição seletiva de atributos.

- Implementada na `excluirProduto`, mas com a lógica ineficiente de reorganização do array mencionada anteriormente.

- A função `comprarProduto` implementa corretamente a validação de estoque antes de realizar a venda.

- O sistema limita o cadastro a 500 produtos e exige confirmações explícitas em operações críticas, como exclusões e vendas.

Percentual de adesão: 80%