- Introdução

Grupo: João Manoel Bezerra Soares

Objetivo do Trabalho:
Este trabalho foi desenvolvido com o objetivo de criar um sistema de gestão de estoque. O sistema permite o armazenamento, atualização e consulta de produtos em um estoque de um mercado fictício.

Escolha do Tema:
O tema "Gestão de Estoque" foi escolhido por sua relevância prática e por permitir a implementação de diversas operações de manipulação de dados, podendo escalonar de um sistema simples para um sistema com diversas implementações complexas.

- Descrição do Programa

Funcionalidades do Sistema:
O sistema desenvolvido oferece as seguintes funcionalidades:

- **Processar Compra:** Reduz a quantidade de um produto no estoque conforme a quantidade comprada.
- **Processar Reabastecimento:** Aumenta a quantidade de um produto no estoque conforme a quantidade recebida.
- **Ver Produtos do Estoque:** Exibe uma lista de todos os produtos com suas respectivas quantidades.
- **Editar Produto:** Permite alterar o nome e a quantidade de um produto específico.
- **Adicionar Novo Produto:** Adiciona um novo produto ao estoque, caso haja espaço disponível no estoque.
- **Remover Produto:** Remove um produto específico do estoque.

Estrutura do Código:
O código foi organizado em diversas funções para modularizar as operações. Cada função é responsável por uma tarefa específica, o que facilita a manutenção e a expansão do sistema.

Módulos e Funções:

- **Funções de Menu:** `Menu()` e `voltar()` exibem opções para o usuário e retornam ao menu principal.
- **Funções de Produto:** `escolherProduto()` e `Mostrarprodutos()` permitem selecionar e exibir produtos.
- **Funções de Operações:** `processarCompra()`, `processarReabastecimento()`, `editarProduto()`, `adicionarProduto()`, e `removerProduto()` executam as operações principais do sistema.

- ## Dificuldades Enfrentadas

Durante o desenvolvimento do sistema de gestão de estoque, algumas dificuldades foram encontradas, principalmente relacionadas à estruturação dos dados e à organização do código em funções.

1. Manipulação de Variáveis para Armazenar Dados de Produtos:

Inicialmente, encontrei dificuldades ao tentar criar uma variável que pudesse armazenar tanto o nome de um produto (uma string) quanto a quantidade disponível (um inteiro). A linguagem C, por ser uma linguagem antiga, não oferece suporte direto para estruturas de dados compostas como em linguagens mais modernas. A solução para este problema foi a utilização de uma `struct`, que permitiu agrupar diferentes tipos de dados sob uma mesma entidade.

A `struct Produtos` foi criada para encapsular o nome do produto e sua quantidade em uma única unidade de dados. Esta abordagem facilitou o gerenciamento dos produtos no estoque, permitindo que o vetor de produtos armazenasse tanto as strings (nomes) quanto os inteiros (quantidades) de maneira eficiente e organizada.

```
struct Produtos {
    char nome[50];
    int quantidade;
};
```

Também criei a variável N para ser o número máximo de produtos que cabem no estoque, fazendo assim o vetor Produtos array[N], assim posso aumentar o número de produtos que cabem no estoque sem problemas

2. Estruturação do Código em Funções:

Outro desafio enfrentado foi a divisão e organização do código em funções distintas para melhorar a modularidade e a clareza do programa. A linguagem C exige uma abordagem disciplinada para a organização de funções, especialmente quando se trata de um sistema com múltiplas operações, como é o caso deste projeto.

- **Funções de Menu:**
  As funções de menu, como `Menu()` e `voltar()`, foram criadas para gerenciar a interação do usuário com o sistema, pois a interface do terminal não é muito agradável para usuários normais, além de que para o tipo de interface escolhida (daquelas digite 1 para isso, digite 2 para aquilo) que usar o switch case, as funções deixam o código muito mais limpo e legível. A função `Menu()` apresenta as opções disponíveis, enquanto a função `voltar()` retorna o usuário ao menu principal após

a execução de uma operação. Estas funções são simples, mas essenciais para garantir uma navegação intuitiva e organizada dentro do programa.

- **Funções de Produto:**
  As funções relacionadas aos produtos, como `escolherProduto()` e `Mostrarprodutos()`, foram criadas para facilitar a seleção e visualização dos itens no estoque. A função `escolherProduto()` ajuda a manter o código principal mais limpo e focado, delegando tarefas específicas a funções menores e mais manejáveis. A função `Mostrarprodutos()` É uma função principal, mas que não manipula os itens então não deixei ela como uma de operação, porém ela é de extrema importância para a compreensão do usuário.
- **Funções de Operações:**
  As funções de operações, incluindo `processarCompra()`, `processarReabastecimento()`, `editarProduto()`, `adicionarProduto()`, e `removerProduto()`, são responsáveis por implementar as funcionalidades principais do sistema. Essas funções encapsulam a lógica da manipulação dos itens, deixando o código funcional para ser um gestor de estoque, além de que elas são

3. Implementação de um banco de dados simples:

****Atenção****

Para o banco de dados funcionar, pegue o arquivo estoque.txt e coloque no mesmo diretório que o programa será executado.

Obs: Junto do arquivo .exe do programa.

Embora a implementação de um banco de dados externo não fosse algo que agregasse nas funções do código, senti a necessidade de incluir uma solução para a persistência de dados. A ideia de um sistema de gestão de estoque que perdesse todas as informações ao ser encerrado me deixou desconfortável. Para resolver esse problema, optei por implementar um sistema de armazenamento utilizando arquivos de texto, o que permitiu que os dados do estoque fossem salvos e carregados de uma sessão para outra.

1. Função `CarregarEstoque()`:

A função `CarregarEstoque()` foi implementada para ler os dados do arquivo `estoque.txt` ao iniciar o programa. Ela abre o arquivo em modo de leitura e processa cada linha, populando o vetor de produtos com os dados armazenados. Se o arquivo não puder ser aberto, o sistema inicia com um estoque vazio, mas a função permite que o sistema recupere o estado anterior do estoque sempre que o programa for reiniciado.

```
void SalvarEstoque(struct Produtos array[], int p) {

    FILE *file = fopen("estoque.txt", "w");

    if (file == NULL) {
```

```
 puts("Erro ao abrir o arquivo estoque.txt para escrita.");

return;}

for (int i = 0; i < p; i++){

fprintf(file, "%s;%d\n", array[i].nome, array[i].quantidade); }

fclose(file);}
```

Com essa função, o sistema é capaz de restaurar o estoque tal como estava antes de ser fechado, garantindo a continuidade das operações e a integridade dos dados.

2. Função `SalvarEstoque()`:

A função `SalvarEstoque()` foi criada para complementar a `CarregarEstoque()` e escrever os dados do estoque em um arquivo de texto (`estoque.txt`). Cada vez que uma alteração é feita no estoque, como a adição ou remoção de um produto, esta função é chamada para garantir que as mudanças sejam refletidas no arquivo. O arquivo é aberto em modo de escrita, e os dados são armazenados no formato `nome_do_produto;quantidade`, garantindo que cada linha do arquivo corresponda a um produto do estoque.

```
void CarregarEstoque(struct Produtos array[], int *p) {

FILE *file = fopen("estoque.txt", "r");

if (file == NULL) {

puts("Erro ao abrir o arquivo estoque.txt para leitura."); *p = 0;

 return;

}

*p = 0;

while (fscanf(file, "%49[^;];%d\n", array[*p].nome, &array[*p].quantidade) == 2) {

(*p)++;}

fclose(file);}
```

Essa função proporciona ao sistema a capacidade de manter um histórico persistente do estoque, o que aumenta significativamente a utilidade e a robustez do programa.

Impacto da Implementação:

A inclusão dessas funções de persistência de dados eleva o nível do sistema, proporcionando maior segurança e confiabilidade. Essa implementação, embora não

obrigatória, demonstra um cuidado adicional com a qualidade e a usabilidade do sistema, garantindo que o estoque gerenciado não seja perdido entre as sessões de uso.

- Conclusão

Resultados Obtidos e Análise Crítica:

O sistema de gestão de estoque foi implementado com sucesso, atendendo a todos os requisitos da disciplina. Ele é capaz de realizar operações básicas de estoque, como adição, remoção e atualização de produtos, além de garantir a persistência dos dados através do armazenamento em um arquivo.

Reflexão sobre o Desenvolvimento:

Ao longo do desenvolvimento, ficou evidente a importância de modularizar o código e estruturar corretamente os dados para manter o sistema organizado e eficiente. O uso de `struct` foi crucial para superar a limitação inicial de manipulação de variáveis compostas, e a implementação de funções específicas para cada operação permitiu que o código se mantivesse limpo e de fácil manutenção.

A inclusão de um sistema de persistência de dados foi uma adição significativa, não apenas tecnicamente, mas também em termos de usabilidade e robustez do sistema. Essa decisão, embora não fosse obrigatória, elevou a qualidade do sistema e refletiu uma preocupação com a experiência do usuário.