Atividade Final de **Estrutura de Dados II**

Alunos: Gustavo Terra e Carlos Ernesto.
Cenário escolhido: Sistema de Cadastro e Busca de Produtos.
Linguagem utilizada: C.

1. Problema:

O cenário requer o desenvolvimento de um sistema para gerenciar o catálogo de produtos de uma loja, assegurando eficiência e precisão na administração das mercadorias. O programa deve oferecer as seguintes funcionalidades:

1. **Cadastro de Produtos:** Inserir produtos no sistema atribuindo-lhes um identificador único (ID), nome, categoria e valor.
2. **Busca Rápida por ID ou pela Categoria:** Localizar produtos rapidamente por meio de seu ID ou buscar produtos pertencentes a uma determinada categoria.
3. **Remoção de um Produto:** Possibilitar a exclusão de um produto específico do catálogo.
4. **Organização e Ordenação:** Apresentar os produtos de maneira ordenada, seja pelo nome ou pelo preço, usando dois métodos de ordenação diferentes.

   Além disso, por se tratar de um possível catálogo muito extenso, é preciso criar um gerenciamento de colisões.

2. Implementação da Solução:

2.1 Tabela Hash:

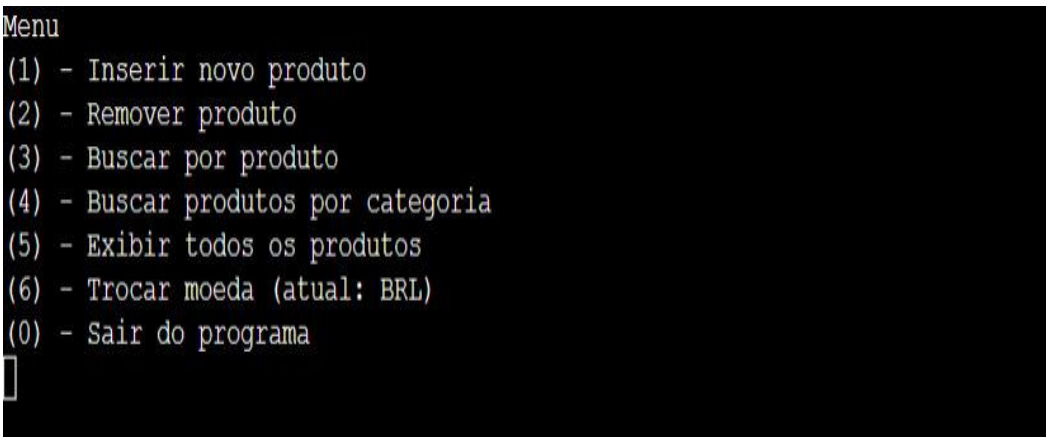
A tabela hash foi escolhida como estrutura principal para armazenar os produtos, pois oferece uma busca eficiente (tempo esperado de O(1)). A função hash utilizada foi simples:

```
21 int funcao_hash(int id) {
22     return id % TAMANHO_HASH;
23 }
```

Essa função usa o **resto da divisão** do ID do produto pelo tamanho da tabela para determinar o índice.

2.2 Funcionalidades Implementadas:

1. **Menu interativo:**



2. **Inserção**:
   A função `inserir` adiciona um novo produto na tabela. Caso a posição calculada pela função hash já esteja ocupada, utiliza-se sondagem linear para encontrar outra posição. Segue um exemplo do processo de inserção:

3. **Remoção**:
   A função `remover` percorre a tabela até encontrar o produto correspondente ao ID fornecido e o remove.

4. **Busca por ID**:
   A função `buscar` percorre a tabela hash para encontrar o produto correspondente ao ID desejado.

5. **Busca por Categoria**:
   A função `buscaPorCategoria` lista os produtos pertencentes a uma categoria específica.

6. **Conversão de Moeda**:
   A função `conversaoMoeda` converte o valor do produto para diferentes moedas suportadas (BRL, USD, CNY, EUR, ARG). Obs: Os valores das conversões são baseados na cotação do dia 30/11/2024.

7. **Exibição**:
   A função `exibir` lista todos os produtos armazenados na tabela. Segue um exemplo da função exibir:

2.3 Métodos de Ordenação:

Dois métodos foram implementados para ordenar os produtos:

- **Quick Sort** (`ordenacaoQsort`):
  Utiliza a função nativa `qsort` da linguagem C para realizar a ordenação baseada no valor dos produtos.

- **Selection Sort** (`ordenacaoSelectionSort`):
  Implementação manual do algoritmo Selection Sort, que percorre o array para encontrar o menor valor e o coloca na posição correta.

2.4 Gerenciamento de Colisões

As colisões são gerenciadas por **sondagem linear**, onde, o método garante que, ao inserir um novo elemento cuja posição calculada já está ocupada, o sistema encontrará a próxima posição disponível de maneira sequencial. A sondagem foi aplicada junto da função inserir.

```c
void inserir(TabelaHash *tabela, Produto* produto) {
    int indice = funcao_hash(produto->id);
    while (tabela->tabela[indice] != NULL) {
        if (indice == TAMANHO_HASH) {
            printf("Erro! Limite TAMANHO_HASH atingido...\n");
        }
        indice++;
    }
    tabela->tabela[indice] = produto;
}
```

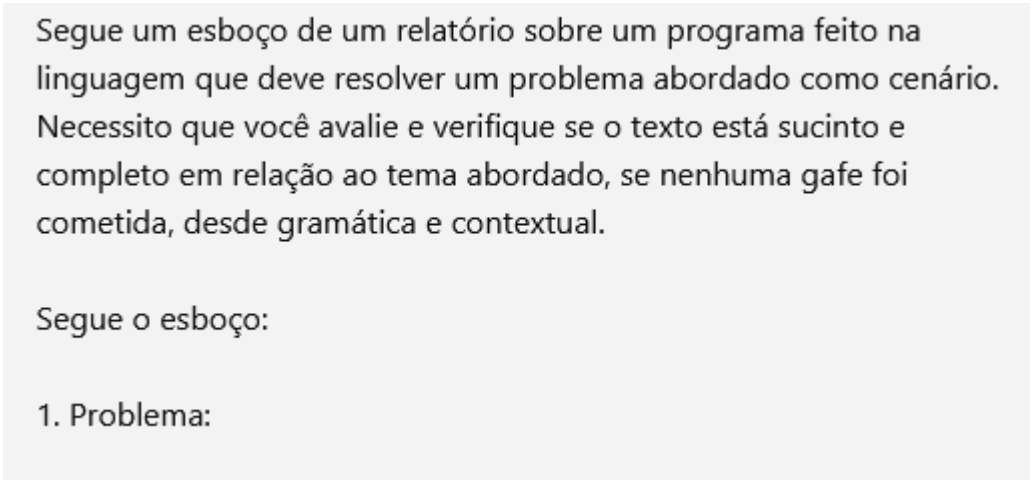
2.5 Comparação de Métodos de Ordenação

De acordo com a comparação feita:

A ordenação por **Quick Sort** é mais eficiente, com tempo esperado de O(n log n), enquanto o **Selection Sort** apresenta complexidade O(n^2). Isso é perceptível ao ordenar uma grande quantidade de produtos.

3. Uso de I.A na Confecção da Atividade:

A inteligência artificial foi utilizada para auxiliar a formação do relatório, no qual deveria deixar o texto fornecido mais sucinto e completo, de uma maneira que evitasse textos longos e que não abordasse o que não foi solicitado. Segue o print de um prompt utilizado:



4. Referências:

1.https://www.w3schools.com/c/ref_stdlib_qsort.php