Projeto Final — Estrutura de Dados I

RedeFarma

Valor: 2,0 pontos • Equipe: até 3 integrantes

Entrega: via Moodle (data 24/11/2025)

Apresentações – 25, 26/11 e 01,02 e 03/12/25

1) Descrição Geral

A RedeFarma é composta por várias filiais (lojas). Todas compartilham o mesmo catálogo de produtos (mesmos códigos e descrições), porém cada filial mantém seu próprio estoque com quantidades que podem variar de uma loja para outra. O sistema — em C, usando alocação dinâmica com estruturas não lineares (listas encadeadas) — deve permitir ao cliente montar um carrinho a partir do catálogo, atualizando o total automaticamente a cada inserção ou remoção de item. Ao finalizar a compra, o sistema deve verificar todas as filiais e:

- Identificar as filiais que atendem 100% do carrinho (ou seja, para cada item, a quantidade disponível na filial é ≥ à quantidade solicitada).
- Para cada filial que não atende tudo, listar item a item o que falta, indicando código/descrição e a quantidade faltante (quantidade solicitada − quantidade disponível).
- O sistema deve permitir que o cliente escolha qualquer filial para efetuar a retirada, independentemente de ela atender ou não 100% do carrinho na verificação inicial.
- Se a filial escolhida atender 100% do carrinho: A venda é confirmada, e o estoque da filial é imediatamente abatido.
- Se a filial escolhida NÃO atender 100% do carrinho:
- O sistema deve executar o Ajuste Automático do Carrinho (conforme detalhado na Seção 6.4): o pedido é automaticamente reduzido para incluir apenas os itens que a filial possui em estoque.
- Se, após o ajuste, o carrinho não ficar vazio, a venda é confirmada apenas para os itens restantes e o estoque da filial é abatido.
- Se, após o ajuste, o carrinho ficar vazio, a venda é cancelada, e o cliente é notificado.

EXEMPLO: CARRINHO: 2001 X 2, 5001 X 4, 2201 X 5

Filial	Código Produtos		
	2001	5001	2201
101	5	10	6
202	3	9	5
208	1	30	4

Aptas (100%) : 101, 202

Não apta: 208 – falta 2001 (falta 1) e 2201 (falta 1)

2) Requisitos Técnicos (Obrigatórios)

- Linguagem C, leitura/escrita de arquivos .txt.
- Alocação dinâmica e estruturas encadeadas (listas).
- Modularizar utilizando funções; uma função por opção de menu.
- Relatórios de saída em .txt (pedido e diagnóstico de atendimento por filial).
- Desalocar toda memória dinâmica ao finalizar a execução do programa.

3) Estruturas de Dados (exemplo base)

```
typedef struct Produto {
   int   codigo;
   char  descricao[64];
   float preco;
   struct Produto *prox; //se julgar necessário pode ser duplamente encadeada
} Produto;


typedef struct ItemEstoque {
   int codigo_produto;
   int quantidade;
   struct ItemEstoque *prox;
} ItemEstoque;
```

```
typedef struct Filial {
    int  id_filial;
    char nome[40];
    ItemEstoque *estoque;     // lista encadeada
    struct Filial *prox;
} Filial;


typedef struct ItemCarrinho {
    int   codigo_produto;
    int   qtd;
    float preco_unit;         // copiado do catálogo no momento da inserção
    struct ItemCarrinho *prox;
} ItemCarrinho;


typedef struct Carrinho {
    ItemCarrinho *itens;
    float total;              // atualizado a cada inserção/remoção
} Carrinho;
```

4) Formatos de Entrada (.txt)

- filiais_v2.txt — id_filial nome cidade
- produtos_v2.txt — codigo descricao preco
- estoques_v2.txt — id_filial codigo_produto quantidade
- carrinhos_clientes_v2.txt — comandos de simulação (ver README_v2)

Sempre use ' ' espaço em branco como separador. Não inclua cabeçalhos nas linhas de dados.

5) Funcionalidades (Menu Mínimo)

- Carregar dados (filiais, produtos, estoques por filial) dos arquivos .txt.
- Pesquisar por código e/ou descrição.
- Carrinho: inserir, remover, listar (subtotal por item e total geral atualizado em tempo real).
- Verificar disponibilidade por filial (sem abater estoque): filiais 100% aptas e faltas por filial.
- Escolher filial e finalizar compra: abater estoque, registrar venda e emitir relatórios.
  - O sistema deve sempre priorizar o fluxo de Ajuste Automático do Carrinho (Seção 6.4) para gerenciar a finalização em filiais que não atendem 100% do pedido original. O objetivo é sempre finalizar a venda com o máximo de itens que a filial escolhida possuir em estoque.
- Relatórios: estoque por filial (ordenado) e diagnóstico atual.

6) Regras de Cálculo e Verificação

6.1 Total do carrinho (atualização em tempo real)

Inserção: total ← total + (preco_unit * qtd_inserida)
Remoção: total ← total − (preco_unit * qtd_removida)

- Se a quantidade do item ficar 0, remova a linha do carrinho.
- Invariantes: total ≥ 0 e itens duplicados no carrinho devem ser consolidados (somar quantidades).

6.2 Verificação por filial (sem consumir estoque)

Para cada filial F, verifique item a item do carrinho:
quantidade disponível do produto de código cod na filial F.
quantidade faltante do produto de cod na filial F.

Definição:

- Filial apta (100%): faltantes = 0 para todos os itens.
- Filial não apta: existe algum item com faltantes > 0.
  → Produzir diagnóstico de faltas: pares (código, quantidade_faltante).

Importante: a verificação não abate estoque; é apenas um “ensaio” para informar disponibilidade e faltas.

6.3 Apresentação ao cliente (antes de fechar)

Liste todas as filiais aptas (100%) para o carrinho atual.
Para cada filial não apta, apresente o relatório de faltas (código/descrição e quantidade faltante).

6.4 Escolha de filial e ajuste automático do carrinho

Se o cliente escolher uma filial não apta (há faltas), o programa deve ajustar automaticamente o carrinho para aquilo que a filial consegue atender:

Para cada item (cod):

- Calcule faltantes
- Se faltante > 0, reduza a quantidade do item no carrinho em faltante.
  Caso a quantidade restante fique 0, remova o item do carrinho.
- Recalcule o total após todos os ajustes.
- Depois do ajuste:
  - Se o carrinho ficar vazio, cancele a venda e informe: “Nenhum item restante para retirada nesta filial”.
  - Caso contrário, reverifique a filial escolhida com o carrinho ajustado (deve estar 100% apta agora).

6.5 Finalização (abatimento e persistência)

Somente após a filial estar confirmada e o carrinho estar 100% apto para ser finalizado, abata o estoque da filial escolhida.
Gere os relatórios (pedido e diagnóstico) e salve os arquivos de saída/estoque atualizados.

6.6 Tratamento de exceções e consistência

- Código inexistente → rejeitar operação e exibir mensagem clara.
- Quantidade inválida (≤ 0 ou overflow) → rejeitar.
- Concorrência: sempre reverificar disponibilidade imediatamente antes de abater estoque.
- Integridade: proíba totais negativos, linhas com quantidade zero e duplicação de chaves no carrinho.

7) Gramática dos Comandos (carrinhos_clientes_v2.txt)

NOVO_CARRINHO <id>

ADD <codigo_produto> <quantidade>

REMOVE <codigo_produto> <quantidade>

VERIFICAR_ATENDIMENTO

FINALIZAR <id_filial>

FIM

8) Casos de Teste Mínimos

- Carrinho atendido por múltiplas filiais.
- Carrinho não atendido por nenhuma filial (listar faltas por filial).
- Inserir/remover itens com consolidação de duplicados e total correto.
- Finalização exata (estoque zera).
- Busca por código com resultados e 'sem resultados'.
- Liberação de memória dinâmica.

9) Observações

- Não abata estoque na verificação; somente ao finalizar.
- Faça comentários em funções mais complexas (headers .h e fontes .c).