



UNIVERSIDADE VEIGA DE ALMEIDA

CURSO DE CIÊNCIA DA COMPUTAÇÃO (Turno Noturno)

Bases de programação.

Wallace Calisto da Silva Santos - 1250115639

Caio Barbosa Galvão - 1250100534

Ana Beatriz da Silva Pinto - 1250109558

Guilherme Brazil Nascimento - 1250203978

Gabriel Bittencourt - 1250111508

RIO DE JANEIRO 2025

Wallace Calisto da Silva Santos - 1250115639

Caio Barbosa Galvão - 1250100534

Ana Beatriz da Silva Pinto - 1250109558

Guilherme Brazil Nascimento - 1250203978

Gabriel Bittencourt - 1250111508

Avaliação A4 - Resolução de Problema [Loja “Flaviozon”]

**Trabalho acadêmico apresentado à
Universidade Exemplo como requisito
parcial para obtenção do título de
Bacharel em Ciência da Computação.**

Orientador: Flavio Maggessi Viola.

RIO DE JANEIRO

2025

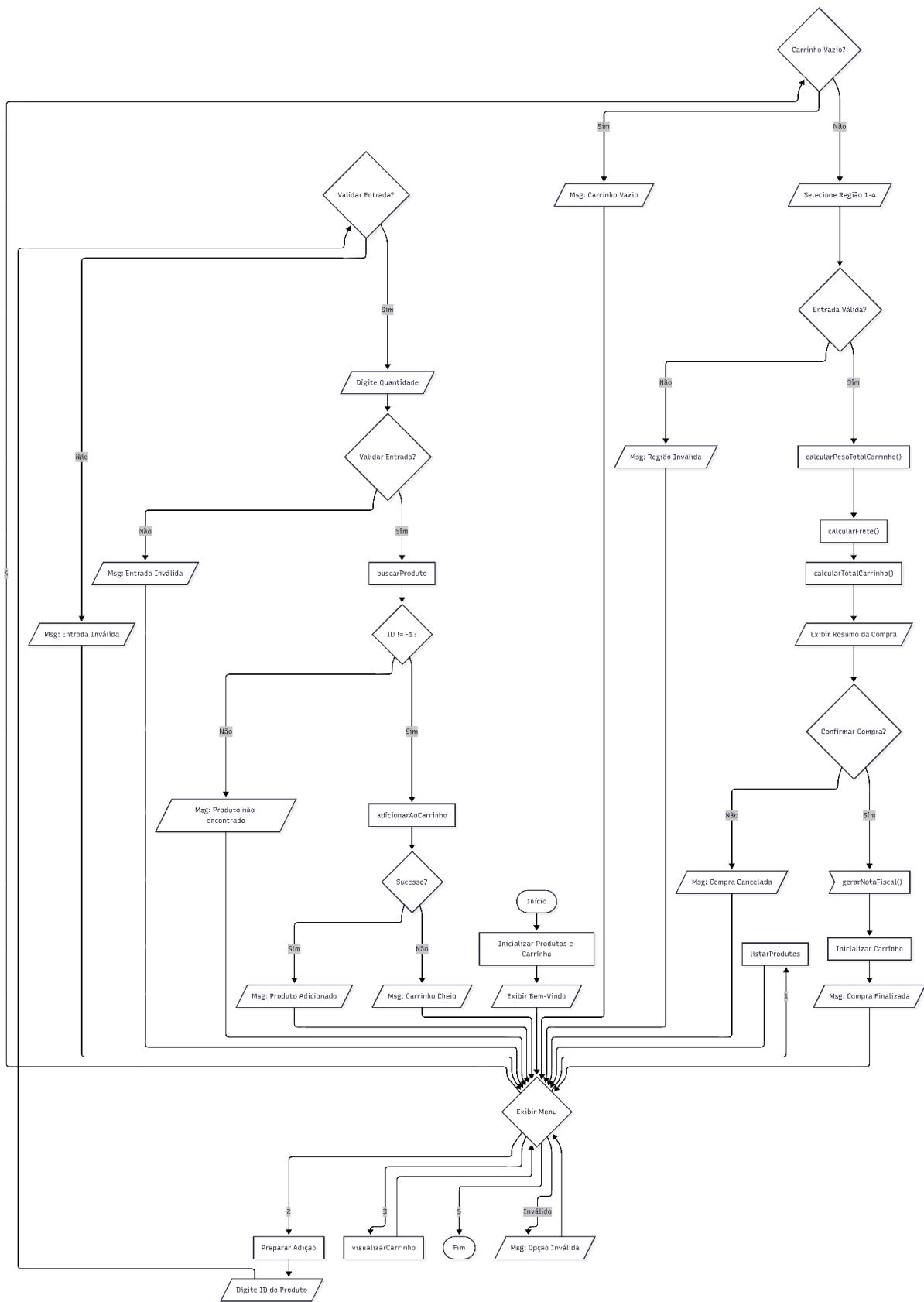
Introdução

O objetivo deste trabalho é desenvolver uma solução computacional para o sistema de e-commerce fictício denominado "Flaviozon". O problema proposto consiste na implementação de um software em linguagem C capaz de simular o fluxo completo de uma compra online, desde a seleção de produtos até a emissão da nota fiscal.

Para atender aos requisitos do sistema, o programa foi estruturado para realizar o gerenciamento de um catálogo de produtos pré-definidos (contendo ID, nome, preço e peso) e a manipulação de um carrinho de compras, permitindo a adição e contagem de itens.

Um dos desafios centrais do problema é a lógica de logística, onde o algoritmo deve calcular o valor do frete considerando duas variáveis: a região de entrega selecionada pelo usuário (Sul, Sudeste, Norte ou Nordeste) e o peso total dos itens acumulados no carrinho. Por fim, o sistema deve consolidar todas as informações da transação (subtotal, frete e total final) e gerar um arquivo de texto (.txt) simulando uma Nota Fiscal, contendo a data e a hora da compra.

Esta solução visa aplicar conceitos fundamentais de programação estruturada, como manipulação de registros (structs), vetores, ponteiros, modularização por funções e gravação de arquivos.



Código fonte:

flaviozon.h

```
#ifndef FLAVIOZON_H
#define FLAVIOZON_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

/*
=====
=====
*           CONSTANTES E MACROS
*
=====
===== */

/** Tamanho maximo para nomes de produtos e strings gerais */
#define MAX_NOME 100

/** Quantidade maxima de itens distintos que o carrinho suporta */
#define MAX_ITENS 50

/*
=====
=====
*           ESTRUTURAS DE DADOS
*
=====
===== */

/** 
 * @brief Enumeração para as regiões de entrega disponíveis.
 * Utilizado para calculo de frete.
 */
typedef enum { SUL = 1, SUDESTE = 2, NORTE = 3, NORDESTE = 4 } Regiao;
```

```

/** 
 * @brief Representa um Produto no catalogo da loja.
 */
typedef struct {
    int id;           // Identificador unico do produto
    char nome[MAX_NOME]; // Nome comercial do produto
    float preco;      // Preco unitario em Reais (R$)
    float peso;        // Peso unitario em Kg
} Produto;

/** 
 * @brief Representa um item selecionado dentro do carrinho de compras.
 * Conecta um produto a uma quantidade desejada.
 */
typedef struct {
    Produto produto; // Copia dos dados do produto selecionado
    int quantidade; // Quantidade do item
} ItemCarrinho;

/** 
 * @brief Estrutura principal do Carrinho de Compras.
 * Mantem o estado atual da sessao de compra do usuario.
 */
typedef struct {
    ItemCarrinho itens[MAX_ITENS]; // Array fixo de itens
    int qtdProdutos; // Numero atual de itens distintos no array
(0 a MAX_ITENS)
} Carrinho;

/*
=====
=====
*                               PROTOTIPOS DE FUNCOES
*
=====
===== */

```

```
/*
-----
-----
 * Módulo: PRODUTOS (Responsavel pelo catalogo)
 *
-----
----- */

/***
 * @brief Inicializa o catalogo de produtos simulado.
 *
 * DEVE SER IMPLEMENTADO EM: produtos.c
 *
 * Detalhes da Implementacao:
 * - Esta funcao deve preencher um vetor estatico interno
 * (local ao arquivo
 * produtos.c) com pelo menos 6 produtos pre-definidos.
 * - Exemplos de produtos: "Robô Gigante", "Pato de Borracha",
 * "Livro C".
 * - Deve ser chamada no inicio da execucao (na main) para
 * preparar os dados.
 *
 * > Dica: Dentro de produtos.c, declare: `static Produto
catalogo[6];`
 *           Na função, faça: `catalogo[0].id = 1;
strcpy(catalogo[0].nome, "Item
* 1"); ...`
```

*/

void inicializarProdutos();

```
/***
 * @brief Lista todos os produtos disponiveis no catalogo
 * formatados na tela.
 *
 * DEVE SER IMPLEMENTADO EM: produtos.c
 *
 * Detalhes da Implementacao:
```

```

    * - Percorre o catalogo interno e imprime os dados de cada
    produto.
    * - Formato sugerido: "ID | Nome | Preco | Peso".
    * - Deve exibir os precos com duas casas decimais (Ex: R$ 29.90).
    */
void listarProdutos();

/** 
 * @brief Busca um produto pelo seu ID unico.
 *
 * DEVE SER IMPLEMENTADO EM: produtos.c
 *
 * @param id O ID do produto a ser buscado.
 * @return Produto Retorna a estrutura do produto se encontrado.
 *         Se nao encontrar, deve retornar um Produto com ID = -1 ou similar
 *         para indicar erro.
 *
 * > Dica: Crie uma variavel `Produto erro; erro.id = -1;` e
 * retorne ela caso o
 * loop acabe sem achar o ID.
 */
Produto buscarProduto(int id);

/*
-----
-----
 * Módulo: CARRINHO (Responsavel pela sessao de compra)
 *
-----
----- */

/** 
 * @brief Inicializa a estrutura do carrinho de compras.
 *
 * DEVE SER IMPLEMENTADO EM: carrinho.c
 *

```

```

* Detalhes da Implementacao:
* - Define a quantidade de produtos (qtdProdutos) como 0.
* - Limpa qualquer lixo de memoria no array de itens se
necessario.
*
* @param c Ponteiro para a estrutura Carrinho a ser zerada.
*
* > Dica: O parametro é um ponteiro (`Carrinho *c`) para que a
alteracao
* persista fora da funcao. Use a seta (`->`) para acessar os
campos:
* `c->qtdProdutos = 0;`
*/
void inicializarCarrinho(Carrinho *c);

/**
* @brief Adiciona um produto ao carrinho ou incrementa sua
quantidade se ja
* existir.
*
* DEVE SER IMPLEMENTADO EM: carrinho.c
*
* Detalhes da Implementacao:
* 1. Verifica se o produto ja existe no array `c->itens`.
*      - Se SIM: Incrementa a quantidade desse item com
`quantidade`.
*      - Se NAO: Verifica se ha espaco no array (< MAX_ITENS).
*          - Se houver espaco, adiciona o novo produto na
posicao
* `c->qtdProdutos` e incrementa o contador `c->qtdProdutos`.
*
* @param c Ponteiro para o carrinho.
* @param p O produto a ser adicionado.
* @param quantidade A quantidade a ser adicionada.
* @return int Retorna 1 se sucesso, 0 se erro (ex: carrinho
cheio).
*

```

```

    * > Dica: Use um loop `for` de 0 até `c->qtdProdutos` para ver
    se o item ja
    * existe. Se não existir, verifique `if (c->qtdProdutos <
MAX_ITENS)` antes de
    * adicionar.
    */
int adicionarAoCarrinho(Carrinho *c, Produto p, int
quantidade);

/**
 * @brief Exibe o conteudo atual do carrinho.
 *
 * DEVE SER IMPLEMENTADO EM: carrinho.c
 *
 * Detalhes da Implementacao:
 * - Imprime cabecalho "--- SEU CARRINHO ---".
 * - Lista cada item: Nome, Preco Unitario, Quantidade,
Subtotal do item (Preco
* * Qtd).
 * - Este modulo NAO calcula o total final nem frete, apenas
lista o conteudo.
 * - Se vazio, imprimir "Carrinho vazio".
 *
 * @param c Carrinho a ser visualizado.
 *
 * > Dica: Aqui o parametro é `Carrinho c` (sem ponteiro) pois
apenas leremos os
 * dados, sem modificar. Acesse com ponto:
`c.itens[i].produto.nome`.
 */
void visualizarCarrinho(Carrinho c);

/**
 * @brief Permite alterar a quantidade de um item ja existente
no carrinho.
 *
 * DEVE SER IMPLEMENTADO EM: carrinho.c (OPCIONAL/EXTRA)
 *

```

```

* Detalhes da Implementacao:
* - Busca o produto no carrinho pelo ID.
* - Atualiza a quantidade para o novo valor.
* - Se a nova quantidade for 0, pode remover o item
(deslocando o array).
*
* @param c Ponteiro para o carrinho.
* @param idProduto ID do produto a ser alterado.
* @param novaQuantidade Nova quantidade desejada.
* @return int 1 se sucesso, 0 se produto nao encontrado.
*/
int atualizarQuantidadeCarrinho(Carrinho *c, int idProduto,
int novaQuantidade);

/**
* @brief Calcula o preco total dos itens no carrinho (Sem
Frete).
*
* DEVE SER IMPLEMENTADO EM: carrinho.c
*
* Detalhes da Implementacao:
* - Soma (Preco * Quantidade) de todos os itens.
*
* @param c Carrinho a ser calculado.
* @return float O valor total em Reais.
*/
float calcularTotalCarrinho(Carrinho c);

/**
* @brief Calcula o peso total de todos os produtos no carrinho
(para calculo de
* frete).
*
* DEVE SER IMPLEMENTADO EM: carrinho.c (Funcao auxiliar para o
modulo Frete)
*
* Detalhes da Implementacao:
* - Soma (Peso * Quantidade) de todos os itens.

```

```

/*
 * @param c Carrinho para calculo.
 * @return float Peso total em Kg.
 */
float calcularPesoTotalCarrinho(Carrinho c);

/*
-----
-----
 * Módulo: FRETE (Responsavel pela logistica)
 *
-----
-----
 */

/** 
 * @brief Calcula o valor do frete baseado na regiao e no peso
total.
*
* DEVE SER IMPLEMENTADO EM: frete.c
*
* Tabela de Precos (Baseada no PDF/README):
* - SUL (1): Leve(<=2kg)=30.00, Pesado(>2kg)=50.00
* - SUDESTE (2): Leve(<=2kg)=25.00, Pesado(>2kg)=45.00
* - NORTE (3): Leve(<=2kg)=35.00, Pesado(>2kg)=55.00
* - NORDESTE (4): Leve(<=2kg)=40.00, Pesado(>2kg)=60.00
*
* @param regiao Enum da regiao escolhida pelo usuario.
* @param pesoTotal Peso total somado do carrinho.
* @return float Valor do frete em Reais.
*
* > Dica: Use `if/else` ou `switch` para checar a `regiao` e
dentro de cada
* caso, verifique se `pesoTotal > 2.0` para decidir o preco.
*/
float calcularFrete(Regiao regiao, float pesoTotal);

/**
 * @brief Auxiliar para obter o nome da regiao em string.

```

```

* util para impressao no relatorio final.
*
* @param regiao Enum da regiao.
* @return const char* Nome da regiao (ex: "Sudeste").
*/
const char *obterNomeRegiao(Regiao regiao);

/*
-----
-----
* Módulo: RELATORIO (Responsavel pelo fechamento e arquivos)
*
-----
----- */

```

/**

- * **@brief** Gera a nota fiscal final e salva em arquivo.
- *
- * **DEVE SER IMPLEMENTADO EM:** relatorio.c
- *
- * **Detalhes da Implementacao:**
 - * 1. Obtem a data/hora atual do sistema (*time.h*).
 - * 2. Calcula uma data estimada de entrega (ex: Data Atual + X dias).
 - * 3. Cria um arquivo de saida (sugestao: "NF_<random>.txt" ou "resumo_compra.txt").
 - * 4. Escreve no arquivo:
 - * - Cabecalho FLAVIOZON.
 - * - Detalhes de cada item comprado.
 - * - Subtotal, Valor do Frete e Total Final.
 - * - Dados de entrega (Regiao, previsao).
 - * 5. Exibe uma mensagem de sucesso na tela com o nome do arquivo gerado.
 - *
 - * **@param** c O carrinho finalizado.
 - * **@param** regiao A regiao de entrega selecionada.
 - * **@param** valorFrete O custo calculado do frete.
 - * **@param** valorTotal O custo total da compra (Itens + Frete).

```

/*
 * > Dica: Use `FILE *f = fopen("nome_arquivo.txt", "w");` 
 *           Se `f == NULL`, deu erro ao criar. Se nao, use
 `fprintf(f,
 * "texto...");` Ao final, obrigatorio usar `fclose(f);` 
 */
void gerarNotaFiscal(Carrinho c, Regiao regiao, float
valorFrete,
                      float valorTotal);

#endif // FLAVIOZON_H

```

main.c

```

/*
=====
=====
* Arquivo: main.c
* Objetivo: Menu principal e fluxo de execução do programa.
*
* Autoria:
*   Nome:      ANA BEATRIZ DA SILVA PINTO
*   Matrícula: 1250109558
*   Data:      2025-12-06
*
=====
===== */

#include "flaviozon.h"

// Função auxiliar para limpar o buffer do teclado
void limparBufferEntrada() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF)
        ;
}

int main() {
    Carrinho meuCarrinho;

```

```
int opcao = 0;

inicializarProdutos();

inicializarCarrinho(&meuCarrinho);

printf("=====\\n");
printf("      Bem-vindo ao Flaviozon!  \\n");
printf("=====\\n");

while (opcao != 5) // funções
{
    printf("\nMenu Principal:\\n");
    printf("1 - Listar Produtos\\n");
    printf("2 - Adicionar Carrinho\\n");
    printf("3 - Visualizar Carrinho\\n");
    printf("4 - Finalizar Compra\\n");
    printf("5 - Sair\\n");

    if (scanf("%d", &opcao) != 1) {

        limparBufferEntrada();
        printf("Entrada inválida!\\n");
        continue;
    }

    switch (opcao) {
    case 1:
        listarProdutos(); // produto.c
        break;

    case 2: {

        int id, quantidade;

        printf("\\nDigite o ID do produto: ");
        if (scanf("%d", &id) != 1) {
```

```

        limparBufferEntrada();
        printf("Entrada inválida!\n");
        break;
    }

    printf("Digite a quantidade: ");
    if (scanf("%d", &quantidade) != 1) {
        limparBufferEntrada();
        printf("Entrada inválida!\n");
        break;
    }

    Produto p = buscarProduto(id);

    if (p.id == -1) {
        printf("Produto não encontrado!\n");
    } else {
        int ok = adicionarAoCarrinho(&meuCarrinho, p,
quantidade);
        if (ok)
            printf("Produto adicionado ao carrinho!\n");
        else
            printf("Carrinho cheio! Não foi possível
adicionar.\n");
    }
} break;

case 3: // ver carrinho
    visualizarCarrinho(meuCarrinho);
    break;

case 4: // finalizar a compra
{
    if (meuCarrinho.qtdProdutos == 0) {
        printf("\nSeu carrinho está vazio!\n");
        break;
}

```

```

int regiao;

printf("\nSelecione sua região para calcular o
frete:\n");
printf("1 - Sul\n");
printf("2 - Sudeste\n");
printf("3 - Norte\n");
printf("4 - Nordeste\n");
printf("Digite a opção: ");

int regiaoOpcão;

if (scanf("%d", &regiaoOpcão) != 1 || regiaoOpcão < 1 ||
    regiaoOpcão > 4) {
    printf("Região inválida!\n");
    limparBufferEntrada();
    break;
}

regiao = (Regiao)regiaoOpcão;

float pesoTotal =
calcularPesoTotalCarrinho(meuCarrinho);
float frete = calcularFrete(regiao, pesoTotal);
float totalItens = calcularTotalCarrinho(meuCarrinho);
float totalFinal = totalItens + frete;

printf("\n===== RESUMO DA COMPRA =====\n");
printf("Subtotal dos produtos: R$ %.2f\n", totalItens);
printf("Frete (%s): R$ %.2f\n", obterNomeRegiao(regiao),
frete);
printf("TOTAL FINAL: R$ %.2f\n", totalFinal);

int confirmar;
printf("\nDeseja finalizar a compra?\n1 - Sim\n2 -
Não\nEscolha: ");
scanf("%d", &confirmar);

```

```

    if (confirmar == 1) {
        gerarNotaFiscal(meuCarrinho, regiao, frete,
totalFinal);
        inicializarCarrinho(&meuCarrinho); // limpa carrinho
        printf("\nCompra finalizada com sucesso!\n");
    } else {
        printf("\nCompra cancelada.\n");
    }
} break;

default:
    printf("Opção inválida! Tente novamente.\n");
}
}

return 0;
}

```

produto.c

```

/*
=====
=====
 * Arquivo: produtos.c
 * Objetivo: Implementação do módulo de catálogo de produtos.
 *
 * Autoria:
 *   Nome: Gabriel Bittencourt
 *   Matrícula: 1250111508
 *   Data:      2025-12-13
 *
=====
===== */
#include "flaviozon.h"

static Produto catalogo[10];
static int total_produtos = 0;
//-----

```

```
// Inicializacao do Banco de Dados
//-----
void inicializarProdutos() {
    catalogo[0].id = 1;
    strcpy(catalogo[0].nome, "Jordan 4 Retro Black Cat");
    catalogo[0].preco = 1300.00;
    catalogo[0].peso = 1.2; // kg

    catalogo[1].id = 2;
    strcpy(catalogo[1].nome, "Jordan 1 Retro High OG");
    catalogo[1].preco = 1100.00;
    catalogo[1].peso = 0.9; // kg

    catalogo[2].id = 3;
    strcpy(catalogo[2].nome, "Jordan 1 Retro Low OG SP");
    catalogo[2].preco = 2100.00;
    catalogo[2].peso = 0.750; // kg

    catalogo[3].id = 4;
    strcpy(catalogo[3].nome, "Jordan 5 Retro");
    catalogo[3].preco = 900.00;
    catalogo[3].peso = 1; // kg

    catalogo[4].id = 5;
    strcpy(catalogo[4].nome, "Jordan 4 Retro Metallic Gold
(Women's)");
    catalogo[4].preco = 1000.00;
    catalogo[4].peso = 1; // kg

    catalogo[5].id = 6;
    strcpy(catalogo[5].nome, "Jordan 4 Retro Mist Blue");
    catalogo[5].preco = 4000.00;
    catalogo[5].peso = 1; // kg

    catalogo[6].id = 7;
    strcpy(catalogo[6].nome, "Jordan 1 Low OG Obsidian UNC");
    catalogo[6].preco = 500.00;
    catalogo[6].peso = 1; // kg
```

```

    catalogo[7].id = 8;
    strcpy(catalogo[7].nome, "Nike Air Force 1 Low Kobe Bryant
Court Purple");
    catalogo[7].preco = 700.00;
    catalogo[7].peso = 1; // kg

    catalogo[8].id = 9;
    strcpy(catalogo[8].nome, "Nike Air Force 1 Low LX Leaf
Camo");
    catalogo[8].preco = 800.00;
    catalogo[8].peso = 1; // kg

    catalogo[9].id = 10;
    strcpy(catalogo[9].nome, "Nike Air Force 1 Low '07 LV8
Bred");
    catalogo[9].preco = 600.00;
    catalogo[9].peso = 1; // kg

    total_produtos = 10;
}

//-----
// Listagem de produtos
//-----
void listarProdutos() {
    printf("\n===== LISTA DE PRODUTOS =====\n");
    printf("Nome, Preco, Peso (kg)\n");

    for (int i = 0; i < total_produtos; i++) {
        printf("%03d | %s | R$ %8.2f | %8.2f kg\n",
catalogo[i].id,
            catalogo[i].nome, catalogo[i].preco,
catalogo[i].peso);
    }
}

//-----
// Busca por ID - Retorna ponteiro para um produto ou NULL

```

```
//-----  
  
Produto buscarProduto(int id) {  
    Produto erro;  
    erro.id = -1;  
    for (int i = 0; i < total_produtos; i++) {  
        if (catalogo[i].id == id) {  
            return catalogo[i];  
        }  
    }  
    return erro;  
}
```

carrinho.c

```
/*  
=====  
=====  
 * Arquivo: carrinho.c  
 * Objetivo: Implementação da lógica do carrinho de compras.  
 *  
 * Autoria:  
 *     Nome:      Wallace Calisto da Silva Santos  
 *     Matrícula: 1250115639  
 *     Data:      2025-12-13  
 *  
===== */  
#include <stdio.h>  
#include <string.h> // Necessario para o memset  
#include "flaviozon.h"  
  
//-----  
// Inicializa o carrinho limpando a memoria  
//-----  
void inicializarCarrinho(Carrinho *c) {  
    // Garante que o array comece zerado para evitar lixo de  
    memoria
```

```

        memset(c->itens, 0, sizeof(c->itens));
        c->qtdProdutos = 0;
    }

//-----
// Adiciona produto ou soma quantidade (Retorna 1 se ok, 0 se erro)
//-----
int adicionarAoCarrinho(Carrinho *c, Produto p, int quantidade) {
    int i;

    // Verifica se a quantidade e valida
    if (quantidade <= 0) {
        return 0; // Erro: Quantidade invalida
    }

    // Procura se o produto ja existe no carrinho
    for (i = 0; i < c->qtdProdutos; i++) {
        if (c->itens[i].produto.id == p.id) {
            // Se achou, so atualiza a quantidade
            c->itens[i].quantidade += quantidade;
            return 1; // Sucesso
        }
    }

    // Se nao achou, verifica se ainda cabe no carrinho
    if (c->qtdProdutos >= MAX_ITENS) {
        return 0; // Erro: Carrinho cheio
    }

    // Adiciona o produto novo na proxima posicao livre
    int posicao = c->qtdProdutos;
    c->itens[posicao].produto = p;
    c->itens[posicao].quantidade = quantidade;

    c->qtdProdutos++; // Aumenta o contador de itens

    return 1; // Sucesso
}

```

```

}

//-----
// Mostra a lista de produtos e o total
//-----
void visualizarCarrinho(Carrinho c) {
    int i;

    printf("\n===== SEU CARRINHO =====\n");

    if (c.qtdProdutos == 0) {
        printf("Carrinho vazio.\n");
    } else {
        printf("Item | Preco Unit. | Qtd | Subtotal\n");

        for (i = 0; i < c.qtdProdutos; i++) {
            Produto p = c.itens[i].produto;
            int qtd = c.itens[i].quantidade;
            float subtotal = p.preco * qtd;

            printf("%d. %s | R$ %.2f | %d | R$ %.2f\n",
                   i + 1, p.nome, p.preco, qtd, subtotal);
        }
    }

    // Calcula e mostra o total geral da compra
    float totalGeral = calcularTotalCarrinho(c);
    printf("-----\n");
    printf("TOTAL DA COMPRA: R$ %.2f\n", totalGeral);
    printf("=====*\n");
}

//-----
// Calcula o valor total em Reais (Soma tudo)
//-----
float calcularTotalCarrinho(Carrinho c) {
    float total = 0.0;
    int i;

```

```

        for (i = 0; i < c.qtdProdutos; i++) {
            total += (c.itens[i].produto.preco *
c.itens[i].quantidade);
        }

        return total;
    }

//-----
// Calcula o peso total em Kg (Para o Frete)
//-----
float calcularPesoTotalCarrinho(Carrinho c) {
    float pesoTotal = 0.0;
    int i;

    for (i = 0; i < c.qtdProdutos; i++) {
        pesoTotal += (c.itens[i].produto.peso *
c.itens[i].quantidade);
    }

    return pesoTotal;
}

//-----
// Atualiza quantidade ou remove se for zero
//-----
int atualizarQuantidadeCarrinho(Carrinho *c, int idProduto,
int novaQuantidade) {

    for (int i = 0; i < c->qtdProdutos; i++) {
        // Procura o produto pelo ID
        if (c->itens[i].produto.id == idProduto) {

            // Se for zero ou menos, remove do array
            if (novaQuantidade <= 0) {
                // Puxa os itens da frente para tras para tapar
                o buraco
                for (int j = i; j < c->qtdProdutos - 1; j++) {

```

```

        c->itens[j] = c->itens[j + 1];
    }
    c->qtdProdutos--; // Diminui o tamanho do
carrinho
}
else {
    // Se for maior que zero, apenas atualiza
    c->itens[i].quantidade = novaQuantidade;
}
return 1; // Sucesso
}
}
return 0; // Erro: Nao achou o produto
}

```

frete.c

```

/*
=====
=====
 * Arquivo: frete.c
 * Objetivo: Implementação do módulo de cálculo de frete.
 *
 * Autoria:
 *   Nome:      Guilherme Brazil Nascimento
 *   Matrícula: 1250203978
 *   Data:      2025-12-06
 *
=====
=====
 */
#include "flaviozon.h"

//
// FUNÇÕES DO MÓDULO FRETE
//

/***

```

```
* @brief Calcula o valor do frete com base na região e no peso
total.
*/
float calcularFrete(Regiao regiao, float pesoTotal) {

    // Validação de peso negativo
    if (pesoTotal < 0) {
        return 0.0;
    }

    float frete = 0;
    int isPesado = (pesoTotal > 2.0);

    switch (regiao) {
        case SUL:
            // <= 2kg: R$ 30,00 | > 2kg: R$ 50,00
            frete = isPesado ? 50.0 : 30.0;
            break;

        case SUDESTE:
            // <= 2kg: R$ 25,00 | > 2kg: R$ 45,00
            frete = isPesado ? 45.0 : 25.0;
            break;

        case NORTE:
            // <= 2kg: R$ 35,00 | > 2kg: R$ 55,00
            frete = isPesado ? 55.0 : 35.0;
            break;

        case NORDESTE:
            // <= 2kg: R$ 40,00 | > 2kg: R$ 60,00
            frete = isPesado ? 60.0 : 40.0;
            break;

        default:
            printf("[ERRO] Regiao invalida para calculo!\n");
            frete = -1.0;
    }
}
```

```

    return frete;
}

/** 
 * @brief Retorna o nome da região para ser usado na Nota
Fiscal.
*/
const char *obterNomeRegiao(Regiao regiao) {
    switch (regiao) {
    case SUL:
        return "Sul"; // Aspas retas corrigidas
    case SUDESTE:
        return "Sudeste";
    case NORTE:
        return "Norte";
    case NORDESTE:
        return "Nordeste";
    default:
        return "Regiao Invalida";
    }
}

```

relatorio.c

```

/*
=====
=====
* Arquivo: relatorio.c
* Objetivo: Implementação da geração de Nota Fiscal e arquivos.
*
* Autoria:
*   Nome:      Caio Barbosa Galvão
*   Matrícula: 1250100534
*   Data:      2025-12-06
*
=====
===== */

```

```
#include "flaviozon.h"

void gerarNotaFiscal(Carrinho c, Regiao regiao, float
valorFrete,
                      float valorTotal) {
    FILE *arquivo;
    char nomeArquivo[50];

    // Gera nome único simples baseado em timestamp ou randômico
    // Para simplificar, usaremos um estático ou randômico simples
    srand(time(NULL));
    int idNota = rand() % 10000;
    sprintf(nomeArquivo, "NF_%04d.txt", idNota);

    arquivo = fopen(nomeArquivo, "w");
    if (arquivo == NULL) {
        printf("Erro ao criar arquivo de nota fiscal!\n");
        return;
    }

    // Obter data atual
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);

    // Calcular "Previsão de Entrega" (Data atual + 5 dias)
    // Nota: Lógica simplificada de data
    struct tm tmEntrega = tm;
    tmEntrega.tm_mday += 5;
    mktime(&tmEntrega); // Normaliza a data

    fprintf(arquivo,
    "=====\\n");
    fprintf(arquivo, "          FLAVIOZON - NOTA FISCAL\\n");
    fprintf(arquivo,
    "=====\\n");
    fprintf(arquivo, "Data da Compra: %02d/%02d/%d %02d:%02d\\n",
    tm.tm_mday,
```

```

        tm.tm_mon + 1, tm.tm_year + 1900, tm.tm_hour,
tm.tm_min);
    fprintf(arquivo, "Previsao Entrega: %02d/%02d/%d\n",
tmEntrega.tm_mday,
            tmEntrega.tm_mon + 1, tmEntrega.tm_year + 1900);
    fprintf(arquivo, "Destino: %s\n", obterNomeRegiao(regiao));
    fprintf(arquivo,
"-----\n");
    fprintf(arquivo, "ITENS:\n\n");

    for (int i = 0; i < c.qtdProdutos; i++) {
        fprintf(arquivo,
            "[%03d] %-20s | Peso: %6.2fKg | Unit: R$ %7.2f |
Qtd: %d | Total: "
            "R$ %.2f\n",
            c.itens[i].produto.id, c.itens[i].produto.nome,
            c.itens[i].produto.peso, c.itens[i].produto.preco,
            c.itens[i].quantidade,
            c.itens[i].produto.preco * c.itens[i].quantidade);
    }

    fprintf(arquivo,
"-----\n");
    fprintf(arquivo, "Subtotal Itens: R$ %.2f\n", valorTotal -
valorFrete);
    fprintf(arquivo, "Frete:           R$ %.2f\n", valorFrete);
    fprintf(arquivo, "TOTAL FINAL:     R$ %.2f\n", valorTotal);
    fprintf(arquivo,
"=====*\n");
    fprintf(arquivo, "Obrigado pela preferencia!\n");

fclose(arquivo);

printf("\n[SUCESSO] Compra finalizada!\n");
printf("Nota Fiscal gerada em: %s\n", nomeArquivo);
}

```

```
=====
      Bem-vindo ao Flaviozon!
=====

Menu Principal:
1 - Listar Produtos
2 - Adicionar Carrinho
3 - Visualizar Carrinho
4 - Finalizar Compra
5 - Sair
Digite uma opção: 1

===== LISTA DE PRODUTOS =====
Nome, Preço, Peso (kg)
001 | Jordan 4 Retro Black Cat | R$ 1300.00 |    1.20 kg
002 | Jordan 1 Retro High OG | R$ 1100.00 |    0.90 kg
003 | Jordan 1 Retro Low OG SP | R$ 2100.00 |    0.75 kg
004 | Jordan 5 Retro | R$ 900.00 |    1.00 kg
005 | Jordan 4 Retro Metallic Gold (Women's | R$ 1000.00 |    1.00 kg
006 | Jordan 4 Retro Mist Blue | R$ 4000.00 |    1.00 kg
007 | Jordan 1 Low OG Obsidian UNC | R$ 500.00 |    1.00 kg
008 | Nike Air Force 1 Low Kobe Bryant Court Purple | R$ 700.00 |    1.00 kg
009 | Nike Air Force 1 Low LX Leaf Camo | R$ 800.00 |    1.00 kg
010 | Nike Air Force 1 Low '07 LV8 Bred | R$ 600.00 |    1.00 kg
```

```
Menu Principal:
1 - Listar Produtos
2 - Adicionar Carrinho
3 - Visualizar Carrinho
4 - Finalizar Compra
5 - Sair
Digite uma opção: 2
```

```
Digite o ID do produto: 001
Digite a quantidade: 1000
Produto adicionado ao carrinho!
```

```
Menu Principal:  
1 - Listar Produtos  
2 - Adicionar Carrinho  
3 - Visualizar Carrinho  
4 - Finalizar Compra  
5 - Sair  
Digite uma opção: 4
```

```
Selecione sua região para calcular o frete:  
1 - Sul  
2 - Sudeste  
3 - Norte  
4 - Nordeste  
Digite a opcao: 1
```

```
===== RESUMO DA COMPRA =====  
Subtotal dos produtos: R$ 1300000.00  
Frete (Sul): R$ 50.00  
TOTAL FINAL: R$ 1300050.00
```

```
Deseja finalizar a compra?  
1 - Sim  
2 - Não  
Escolha: 1
```

```
[SUCESSO] Compra finalizada!  
Nota Fiscal gerada em: NF_5351.txt
```

```
Menu Principal:  
1 - Listar Produtos  
2 - Adicionar Carrinho  
3 - Visualizar Carrinho  
4 - Finalizar Compra  
5 - Sair  
Digite uma opção: 3  
  
===== SEU CARRINHO =====  
Item | Preco Unit. | Qtd | Subtotal  
1. Nike Air Force 1 Low LX Leaf Camo | R$ 800.00 | 100 | R$ 80000.00  
-----  
TOTAL DA COMPRA: R$ 80000.00  
=====
```