**Relatório do Projeto: Programa de Estoque e Carrinho de Supermercado**

**Lista de Integrantes**

Júlio César Toscano Ximenes Júnior - 31214061

Maria Teresa Torres Santos - 31049796

Sâmi Gabriele de Andrade Carvalho - 31077960

Tito José de Castro Amorim - 31516106

**Introdução**

O programa criado tem como objetivo simular um estoque de um supermercado, onde o usuário pode adicionar novos itens ao estoque e também adicionar itens a um carrinho. Uma vez que o usuário tenha adicionado itens ao carrinho, é possível verificar quais itens foram adicionados, informações sobre os itens e o valor total dos itens contidos no carrinho.

**Resultados**

O programa consiste de 5 telas diferentes, cada uma com uma função diferente que vai guiar o usuário para onde ele deseja. Por ser um programa um pouco mais complexo, o planejamento foi realizado através de um fluxograma (fig. 1).

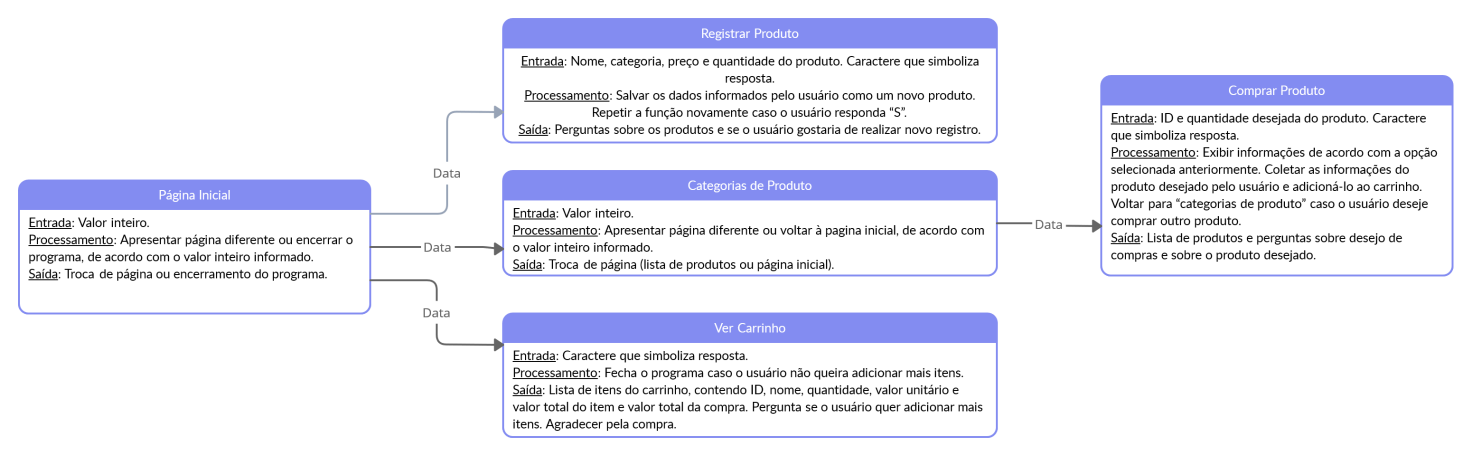


Figura 1

Para melhor visualização, o texto contido no fluxograma será inserido integralmente abaixo.

Página Inicial:

*Entrada*: Valor inteiro.

*Processamento*: Apresentar página diferente ou encerrar o programa, de acordo com o valor inteiro informado.

*Saída*: Troca de página ou encerramento do programa.

Registrar Produto:

*Entrada*: Nome, categoria, preço e quantidade do produto. Caractere que simboliza resposta.

*Processamento*: Salvar os dados informados pelo usuário como um novo produto. Repetir a função novamente caso o usuário responda “S”.

*Saída*: Perguntas sobre os produtos e se o usuário gostaria de realizar novo registro.

Categorias de Produto:

*Entrada*: Valor inteiro.

*Processamento*: Apresentar página diferente ou voltar à pagina inicial, de acordo com o valor inteiro informado.

*Saída*: Troca de página (lista de produtos ou página inicial).

Comprar Produto:

*Entrada*: ID e quantidade desejada do produto. Caractere que simboliza resposta.

*Processamento*: Exibir informações de acordo com a opção selecionada anteriormente. Coletar as informações do produto desejado pelo usuário e adicioná-lo ao carrinho. Voltar para “categorias de produto” caso o usuário deseje comprar outro produto.

*Saída*: Lista de produtos e perguntas sobre desejo de compras e sobre o produto desejado.

Ver Carrinho:

*Entrada*: Caractere que simboliza resposta.

*Processamento*: Fecha o programa caso o usuário não queira adicionar mais itens.

*Saída*: Lista de itens do carrinho, contendo ID, nome, quantidade, valor unitário e valor total do item e valor total da compra. Pergunta se o usuário quer adicionar mais itens. Agradecer pela compra.

A primeira tela é chamada de Página Inicial e está definida dentro da função main do nosso programa (fig. 2 e 3). Ela é responsável por informar ao usuário quais as tarefas que ele pode realizar e redirecionar ele para outras telas. Além disso, também alimentos o estoque do supermercado com itens fixos, que não dependem de entradas do usuário, através da função inventarioAtual. O menu inicial será repetido indefinidamente até que o usuário informe que quer sair do programa, ação que pode ser realizada na página inicial ou no carrinho de compras.



Figura 2

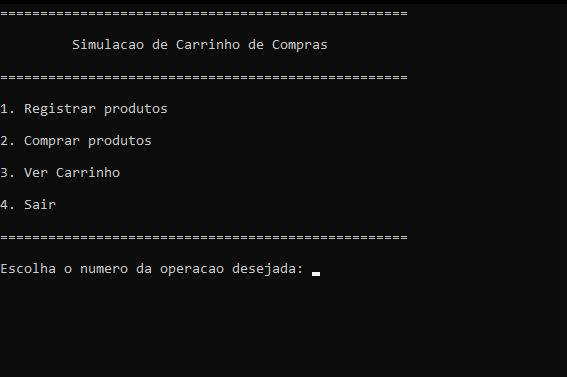


Figura 3

Após o usuário selecionar uma opção, que é validada através da função validarEscolha, o switch case executará a função correspondente ao número digitado. Para simplificar o processo de explicação, seguiremos as funções de cima para baixo.

A função de registrar produtos (fig. 4 e 5) recebe como parâmetro nossa struct de produtos é responsável por adicionar produtos ao estoque do supermercado. Ao ser executada, a função perguntará ao usuário o nome, categoria, preço e quantidade do produto para alimentação do estoque. A função realiza validação de entrada ao receber a categoria do produto, uma vez é que de extrema importância para o funcionamento do código que a categoria esteja escrita de forma correta.

Além disso, essa função também é responsável por aumentar dinamicamente o tamanho da nossa struct produtos. Ao final da função, perguntamos ao usuário se ele deseja adicionar outro produto ao inventário, passando a pergunta como parâmetro para a função simOuNao. Essa função recebe uma string como parâmetro, que deve ter como resposta Sim ou Não. Ela valida a entrada recebida (aceita o caractere ‘s’ ou ‘n’, maiúsculo ou minúsculo) e retorna o caractere correspondente maiúsculo. Dessa forma, conseguimos criar um loop de registro de produtos que só é interrompido quando o usuário responde que não deseja adicionar mais produtos.

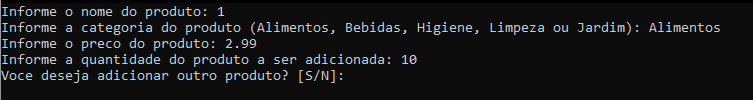


Figura 4

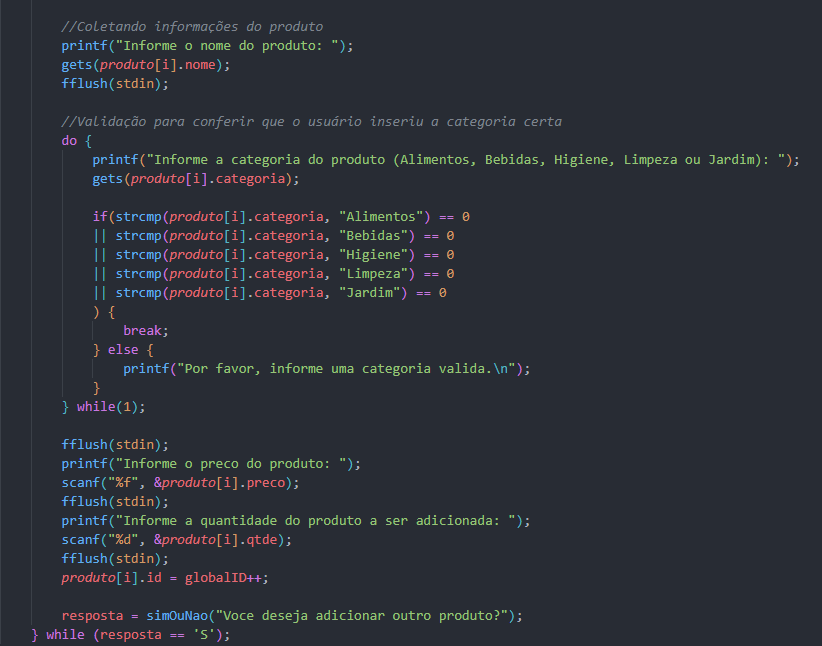


Figura 5

O ID dos produtos é atribuído utilizando uma variável global, que é utilizada em diferentes funções do nosso código e incrementada a cada produto adicionado. Dessa forma, todos os produtos terão IDs diferentes.

A próxima opção da nossa tela inicial é a de comprar produtos. Quando o usuário digita 2 na tela inicial, a função comprarProdutos é chamada. Porém, a primeira coisa que essa função faz é chamar a função paginaCategorias, então vamos começar por ela.

A função paginaCategorias (fig. 6 e 7) recebe como parâmetro nossa struct de produtos e imprime um menu na tela do usuário, perguntando qual categoria de produto ele gostaria de visualizar. A opção digitada é validada pela função validarEscolha. Os produtos foram categorizados para facilitar a procura de algo específico em uma lista volumosa.

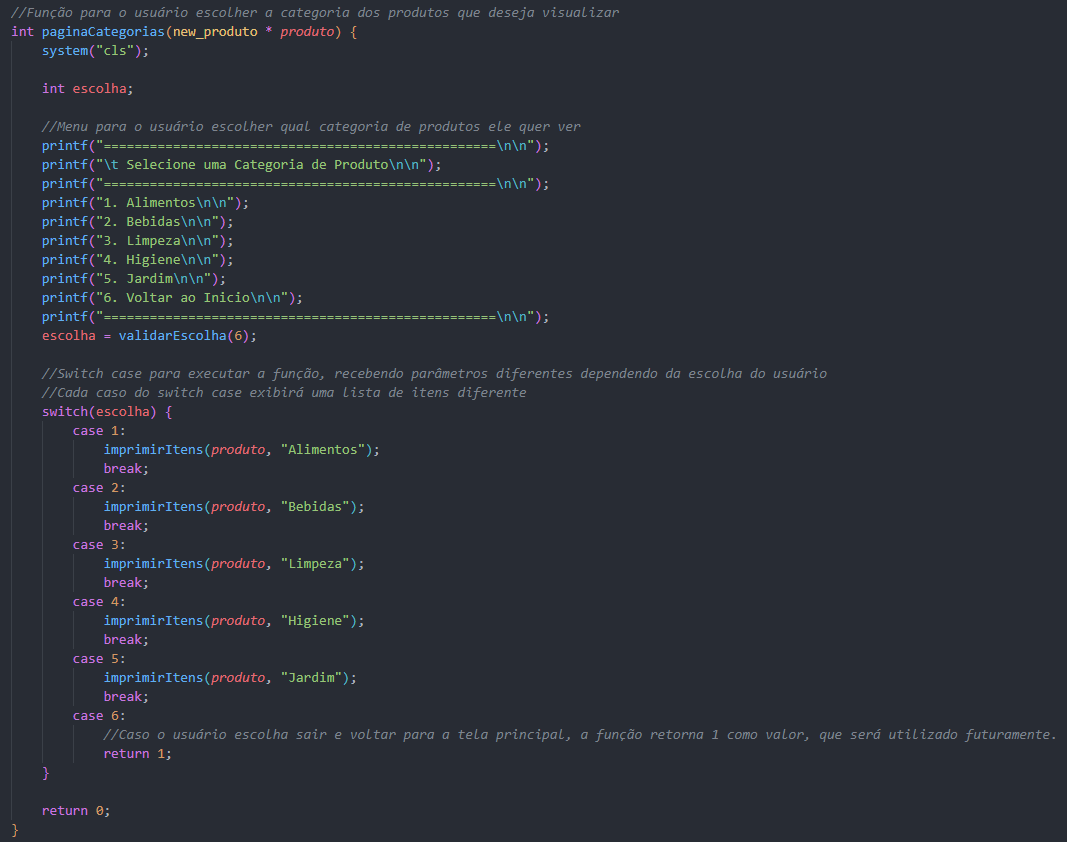


Figura 6

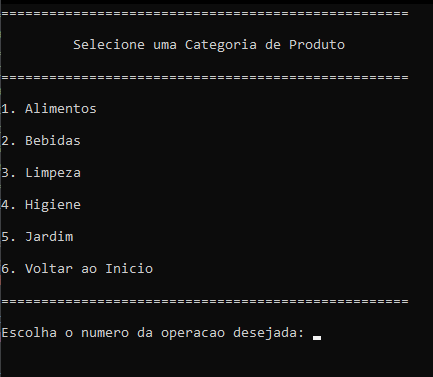


Figura 7

Ao digitar um número correspondente a uma categoria de produto uma nova função será chamada: a imprimirItens (fig. 8). Essa função recebe como parâmetro nossa lista de produtos e uma string, que é correspondente a uma categoria. Tendo a lista de produtos e uma categoria, ela compara a categoria de todos os produtos do nosso estoque e retorna informações apenas dos produtos cuja categoria corresponde à categoria passada como parâmetro.

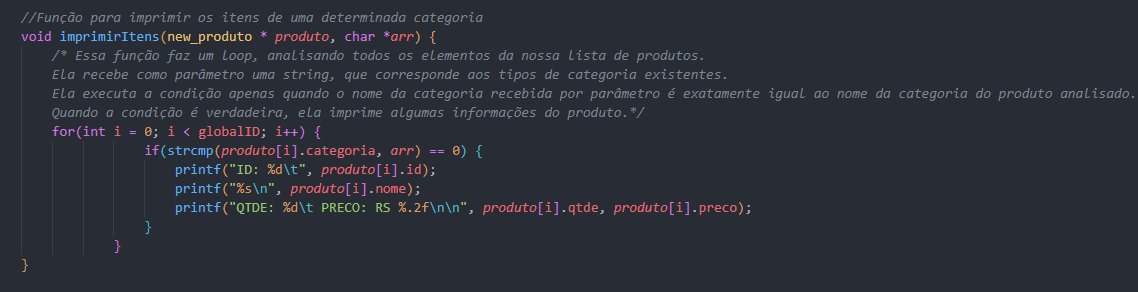


Figura 8

Além disso, a função paginaCategorias sempre retorna 0 ou 1. Ela retorna 0 caso uma categoria seja selecionada e retorna 1 caso a opção 6 (voltar ao início) seja selecionada. Uma vez que o usuário fez sua escolha no menu e que os itens estão impressos na tela, voltamos para a função comprarProdutos.

É importante ressaltar que a maior parte da função comprarProdutos (fig. 9 e 10) só é executada quando o retorno da função paginaCategorias é 0. Se o retorno de paginaCategorias for 1, isso significa que o usuário quer voltar ao menu inicial, então a função comprarProdutos não é executada. Caso o usuário escolha uma categoria de produto, a função comprarProdutos é executada normalmente e pergunta se o usuário gostaria de comprar algum item da lista.

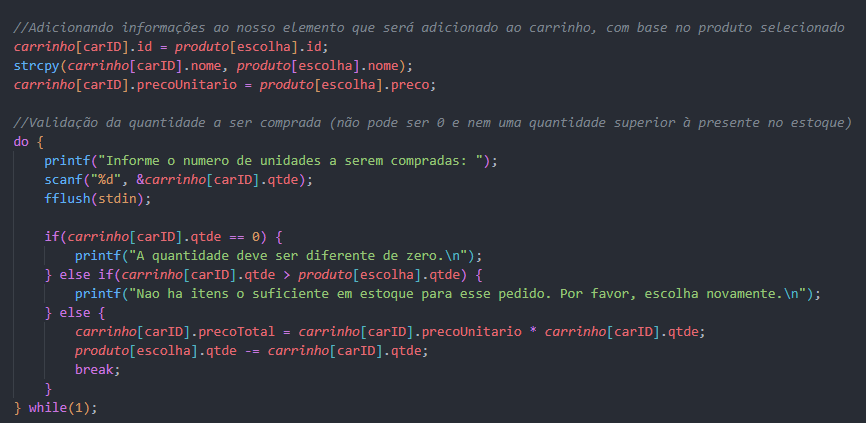


Figura 9

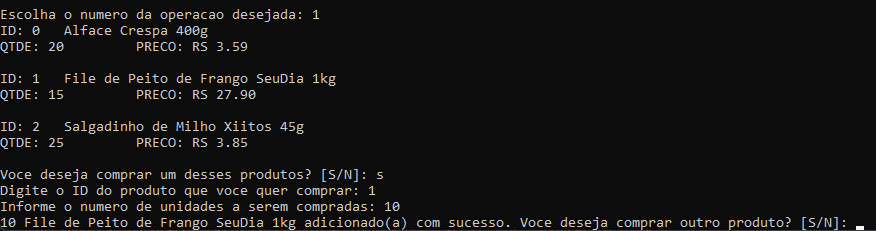


Figura 10

Assim como a função registrarProduto, a comprarProduto também é responsável por aumentar dinamicamente o tamanho da nossa struct carrinho caso ela não seja grande o suficiente. Utilizamos o ID do item para perguntar ao usuário qual item ele gostaria de comprar, e também pedimos a quantidade de itens desejados, que tem validação dentro da própria função (não pode ser igual a zero e não pode ser maior que a quantidade disponível em estoque). Sabendo essas respostas, podemos adicionar o item à nossa struct de carrinho, que será exibida posteriormente. Após salvar o item no carrinho, perguntamos se o usuário gostaria de comprar outro item. Usamos a função simOuNao para validar a resposta e, caso a resposta seja sim, a função comprarProduto é executada novamente. Caso contrário, voltamos para a página inicial.

A última função do nosso código é a verCarrinho (fig. 11 e 12), que recebe como parâmetro nossa struct carrinho. O usuário pode acessar essa página através da página inicial. A verCarrinho é relativamente simples em comparação às outras, uma vez que ela apenas exibe informações.

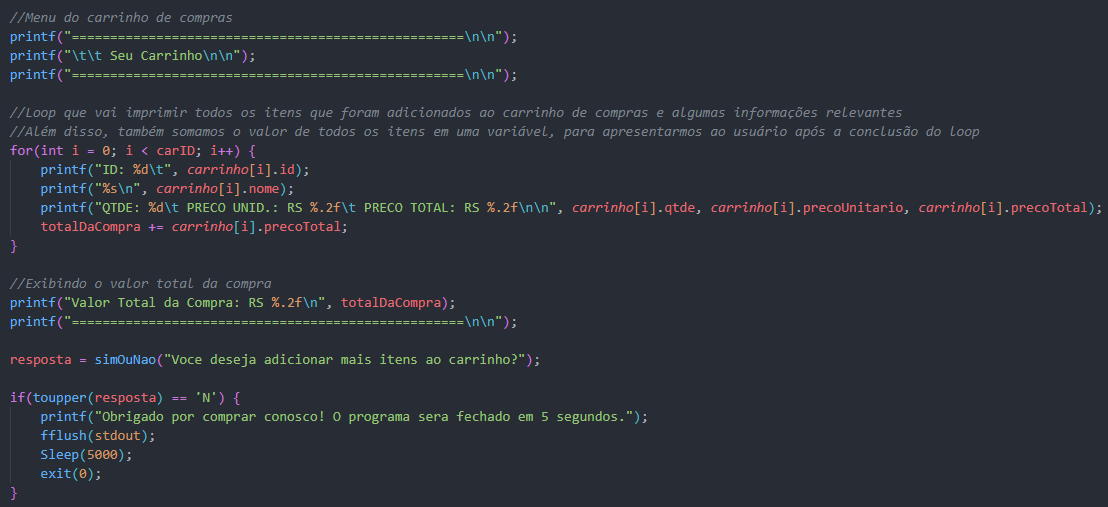


Figura 11

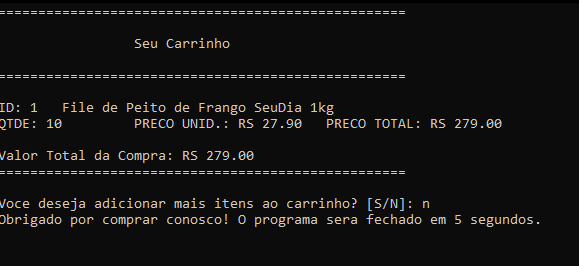


Figura 12

Ao ser chamada, ela faz um loop em nossa struct carrinho para exibir informações dos produtos que estão nela. Após isso, ela apresenta o valor total dos itens do carrinho e pergunta ao usuário se ele gostaria de adicionar mais itens. Caso a resposta seja sim, o programa volta à tela inicial, e caso seja não, ele agradece a preferência e informa que o programa será encerrado em 5 segundos.

As maiores dificuldades encontradas foram manter a lógica coerente ao longo do código, decidir quais códigos seriam transformados em funções e planejar o programa de forma que não fosse muito ambiciosa, uma vez que temos um prazo limitado para execução. Além disso, também sentimos dificuldades em alocar dinamicamente a lista de produtos e em como receber nossa struct como parâmetro da função. Em todos os casos, nos solucionamos nossas dificuldades através de pesquisa, testes, organização e discussões em equipe.

**Apêndice**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <windows.h>

//Estrutura que lida com o conteúdo de cada produto

typedef struct Produto {

int id;

char nome[200];

char categoria[10];

float preco;

int qtde;

} new\_produto;

//Estrutura que lida com o carrinho de compras

typedef struct Carrinho {

int id;

char nome[200];

float precoUnitario;

float precoTotal;

int qtde;

} new\_carrinho;

//Variáveis globais, usadas em múltiplas funções

int globalID = 0;

int carID = 0;

int prodSize = 100;

int carSize = 100;

//Declarações de funções pra o compilador não ficar zangado tentando adivinhar o tipo delas (e falhando miseravelmente)

void registrarProduto();

int paginaCategorias();

void imprimirItens();

void comprarProdutos();

void verCarrinho();

void inventarioAtual();

char simOuNao();

int validarEscolha();

int main() {

//Variáveis da função main

int escolha;

//Criando nossa struct de lista de produtos

new\_produto \* produto = malloc(prodSize \* sizeof(\*produto));

//Criando nossa struct de carrinho de compras

new\_carrinho \* carrinho = malloc(carSize \* sizeof(\*carrinho));

//Alimentando o estoque com opções fixas

inventarioAtual(produto);

//Repetição para a tela inicial fechar apenas quando o usuário decide sair do programa

while(1) {

//Função para limpar a tela do programa

system("cls");

int escolha;

//Menu da Página Inicial + opção para o usuário ir para outras páginas

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

printf("\t Simulacao de Carrinho de Compras\n\n");

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

printf("1. Registrar produtos\n\n");

printf("2. Comprar produtos\n\n");

printf("3. Ver Carrinho\n\n");

printf("4. Sair\n\n");

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

// printf("Escolha o numero da operacao desejada: ");

// scanf("%d", &escolha);

// fflush(stdin);

escolha = validarEscolha(4);

//Switch case para escolher qual função será executada através do input do usuário

switch(escolha) {

case 1:

registrarProduto(produto);

break;

case 2:

comprarProdutos(produto, carrinho);

break;

case 3:

verCarrinho(carrinho);

break;

case 4:

exit(0);

}

}

return 0;

}

//Função para o usuário registrar novos produtos

void registrarProduto(new\_produto \* produto) {

char resposta;

//Loop para registro de produtos. O usuário sai do loop apenas quando indica que não quer adicionar mais produtos.

do {

//Limpando a tela

system("cls");

//Definindo i como sendo igual à variável Global ID, para atribuirmos os valores ao índice correto

int i = globalID;

//Confirmando se o tamanho alocado a carrinho é o suficiente para receber outro elemento, e aumentando o tamanho caso não seja

if(globalID >= prodSize) {

prodSize \*= 1.5;

produto = (new\_produto \*) realloc(produto, prodSize \* sizeof(\*produto));

}

//Coletando informações do produto

printf("Informe o nome do produto: ");

gets(produto[i].nome);

fflush(stdin);

//Validação para conferir que o usuário inseriu a categoria certa

do {

printf("Informe a categoria do produto (Alimentos, Bebidas, Higiene, Limpeza ou Jardim): ");

gets(produto[i].categoria);

if(strcmp(produto[i].categoria, "Alimentos") == 0

|| strcmp(produto[i].categoria, "Bebidas") == 0

|| strcmp(produto[i].categoria, "Higiene") == 0

|| strcmp(produto[i].categoria, "Limpeza") == 0

|| strcmp(produto[i].categoria, "Jardim") == 0

) {

break;

} else {

printf("Por favor, informe uma categoria valida.\n");

}

} while(1);

fflush(stdin);

printf("Informe o preco do produto: ");

scanf("%f", &produto[i].preco);

fflush(stdin);

printf("Informe a quantidade do produto a ser adicionada: ");

scanf("%d", &produto[i].qtde);

fflush(stdin);

produto[i].id = globalID++;

resposta = simOuNao("Voce deseja adicionar outro produto?");

} while (resposta == 'S');

}

//Função para o usuário escolher a categoria dos produtos que deseja visualizar

int paginaCategorias(new\_produto \* produto) {

system("cls");

int escolha;

//Menu para o usuário escolher qual categoria de produtos ele quer ver

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

printf("\t Selecione uma Categoria de Produto\n\n");

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

printf("1. Alimentos\n\n");

printf("2. Bebidas\n\n");

printf("3. Limpeza\n\n");

printf("4. Higiene\n\n");

printf("5. Jardim\n\n");

printf("6. Voltar ao Inicio\n\n");

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

escolha = validarEscolha(6);

//Switch case para executar a função, recebendo parâmetros diferentes dependendo da escolha do usuário

//Cada caso do switch case exibirá uma lista de itens diferente

switch(escolha) {

case 1:

imprimirItens(produto, "Alimentos");

break;

case 2:

imprimirItens(produto, "Bebidas");

break;

case 3:

imprimirItens(produto, "Limpeza");

break;

case 4:

imprimirItens(produto, "Higiene");

break;

case 5:

imprimirItens(produto, "Jardim");

break;

case 6:

//Caso o usuário escolha sair e voltar para a tela principal, a função retorna 1 como valor, que será utilizado futuramente.

return 1;

}

return 0;

}

//Função para imprimir os itens de uma determinada categoria

void imprimirItens(new\_produto \* produto, char \*arr) {

/\* Essa função faz um loop, analisando todos os elementos da nossa lista de produtos.

Ela recebe como parâmetro uma string, que corresponde aos tipos de categoria existentes.

Ela executa a condição apenas quando o nome da categoria recebida por parâmetro é exatamente igual ao nome da categoria do produto analisado.

Quando a condição é verdadeira, ela imprime algumas informações do produto.\*/

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

if(strcmp(produto[i].categoria, arr) == 0) {

printf("ID: %d\t", produto[i].id);

printf("%s\n", produto[i].nome);

printf("QTDE: %d\t PRECO: RS %.2f\n\n", produto[i].qtde, produto[i].preco);

}

}

}

//Função para o usuário comprar produtos da categoria selecionada

void comprarProdutos(new\_produto \* produto, new\_carrinho \* carrinho) {

int escolha;

char resposta;

//Loop de compra de produtos: será realizado até que o usuário indique que não quer mais comprar produtos.

do {

//Renderizando os produtos na tela e condição que faz voltar para a tela inicial quando o usuário digita 6 como categoria escolhida

if(paginaCategorias(produto) == 1) {

break;

};

//Condição que checa se o carrinho tem espaço o suficiente para receber novos itens.

//Caso o carrinho não tenha espaço o suficiente, espaço adicional será realocado para ele.

if(carID >= carSize) {

carSize \*= 1.5;

carrinho = (new\_carrinho \*) realloc(carrinho, carSize \* sizeof(\*carrinho));

}

//Perguntando se o usuário gostaria de comprar algum produto e validando resposta

resposta = simOuNao("Voce deseja comprar um desses produtos?");

if(resposta == 'S') {

//Validação do ID inserido (tem que ser um ID que represente um produto)

do {

printf("Digite o ID do produto que voce quer comprar: ");

scanf("%d", &escolha);

fflush(stdin);

if(escolha >= globalID) {

printf("Por favor, escolha um ID valido.");

} else {

break;

}

} while(1);

//Adicionando informações ao nosso elemento que será adicionado ao carrinho, com base no produto selecionado

carrinho[carID].id = produto[escolha].id;

strcpy(carrinho[carID].nome, produto[escolha].nome);

carrinho[carID].precoUnitario = produto[escolha].preco;

//Validação da quantidade a ser comprada (não pode ser 0 e nem uma quantidade superior à presente no estoque)

do {

printf("Informe o numero de unidades a serem compradas: ");

scanf("%d", &carrinho[carID].qtde);

fflush(stdin);

if(carrinho[carID].qtde == 0) {

printf("A quantidade deve ser diferente de zero.\n");

} else if(carrinho[carID].qtde > produto[escolha].qtde) {

printf("Nao ha itens o suficiente em estoque para esse pedido. Por favor, escolha novamente.\n");

} else {

carrinho[carID].precoTotal = carrinho[carID].precoUnitario \* carrinho[carID].qtde;

produto[escolha].qtde -= carrinho[carID].qtde;

break;

}

} while(1);

printf("%d %s adicionado(a) com sucesso. ", carrinho[carID].qtde, carrinho[carID].nome);

carID++;

//Validando desejo de outra compra

resposta = simOuNao("Voce deseja comprar outro produto?");

if(resposta == 'N') {

break;

}

}

} while(1);

}

//Função para ver o carrinho

void verCarrinho(new\_carrinho \* carrinho) {

float totalDaCompra = 0;

char resposta;

system("cls");

//Menu do carrinho de compras

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

printf("\t\t Seu Carrinho\n\n");

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

//Loop que vai imprimir todos os itens que foram adicionados ao carrinho de compras e algumas informações relevantes

//Além disso, também somamos o valor de todos os itens em uma variável, para apresentarmos ao usuário após a conclusão do loop

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

printf("ID: %d\t", carrinho[i].id);

printf("%s\n", carrinho[i].nome);

printf("QTDE: %d\t PRECO UNID.: RS %.2f\t PRECO TOTAL: RS %.2f\n\n", carrinho[i].qtde, carrinho[i].precoUnitario, carrinho[i].precoTotal);

totalDaCompra += carrinho[i].precoTotal;

}

//Exibindo o valor total da compra

printf("Valor Total da Compra: RS %.2f\n", totalDaCompra);

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

resposta = simOuNao("Voce deseja adicionar mais itens ao carrinho?");

if(toupper(resposta) == 'N') {

printf("Obrigado por comprar conosco! O programa sera fechado em 5 segundos.");

fflush(stdout);

Sleep(5000);

exit(0);

}

}

//Função para validar respostas de S ou N

char simOuNao(char \*arr) {

char resposta;

do {

printf("%s [S/N]: ", arr);

scanf("%c", &resposta);

fflush(stdin);

if(toupper(resposta) != 'S' && toupper(resposta) != 'N') {

printf("Por favor, digite S ou N.\n");

}

} while(toupper(resposta) != 'S' && toupper(resposta) != 'N');

return toupper(resposta);

}

//Função que valida número de escolha para um menu, recebe como argumento o maior número do menu

int validarEscolha(int max) {

int escolha;

while(1) {

printf("Escolha o numero da operacao desejada: ");

scanf("%d", &escolha);

fflush(stdin);

if(escolha <= 0 || escolha > max) {

printf("Por favor, escolha um numero valido (de 1 a %d)\n", max);

} else {

return escolha;

}

}

}

//Registrando produtos fixos (que já estariam em inventário) para ter opções caso o usuário queira apenas comprar

void inventarioAtual(new\_produto \* produto) {

int i = 0;

//-------- categoria alimentos ----------

produto[i].id = globalID++;

strcpy(produto[i].nome, "Alface Crespa 400g");

strcpy(produto[i].categoria, "Alimentos");

produto[i].preco = 3.59;

produto[i++].qtde = 20;

produto[i].id = globalID++;

strcpy(produto[i].nome, "File de Peito de Frango SeuDia 1kg");

strcpy(produto[i].categoria, "Alimentos");

produto[i].preco = 27.90;

produto[i++].qtde = 15;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Salgadinho de Milho Xiitos 45g");

strcpy(produto[i].categoria, "Alimentos");

produto[i].preco = 3.85;

produto[i++].qtde = 25;

//-------- categoria bebidas ----------

produto[i].id = globalID++;

strcpy(produto[i].nome, "Cerveja Vishnu Puro Malte 350ml");

strcpy(produto[i].categoria, "Bebidas");

produto[i].preco = 2.99;

produto[i++].qtde = 29;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Vinho Fazenda da Preguica Bordo Meio Seco 750ml");

strcpy(produto[i].categoria, "Bebidas");

produto[i].preco = 20.88;

produto[i++].qtde = 7;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Gin Tancai London Dry 750 ml");

strcpy(produto[i].categoria, "Bebidas");

produto[i].preco = 125.99;

produto[i++].qtde = 17;

//-------- categoria limpeza ----------

produto[i].id = globalID++;

strcpy(produto[i].nome, "Amaciante Flor 1.8L");

strcpy(produto[i].categoria, "Limpeza");

produto[i].preco = 22.79;

produto[i++].qtde = 13;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Detergente Olhe Limao 400ml");

strcpy(produto[i].categoria, "Limpeza");

produto[i].preco = 11.49;

produto[i++].qtde = 43;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Desinfetante Olhe Lavanda 1L");

strcpy(produto[i].categoria, "Limpeza");

produto[i].preco = 13.29;

produto[i++].qtde = 27;

//-------- categoria higiene ----------

produto[i].id = globalID++;

strcpy(produto[i].nome, "Desodorante Aerosol Palomas MEN 150ml");

strcpy(produto[i].categoria, "Higiene");

produto[i].preco = 15.79;

produto[i++].qtde = 33;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Sabonete Phoebe Odor de Rosas 90g");

strcpy(produto[i].categoria, "Higiene");

produto[i].preco = 4.89;

produto[i++].qtde = 3;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Xampu Algodao Pureza Refrescante 325ml");

strcpy(produto[i].categoria, "Higiene");

produto[i].preco = 11.29;

produto[i++].qtde = 7;

//-------- categoria jardim ----------

produto[i].id = globalID++;

strcpy(produto[i].nome, "Vaso Concreto Palawa Joko Cinza 16x15cm");

strcpy(produto[i].categoria, "Jardim");

produto[i].preco = 52.80;

produto[i++].qtde = 12;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Adubo Esterco Bovino Cheiro Feliz 4kg");

strcpy(produto[i].categoria, "Jardim");

produto[i].preco = 9.50;

produto[i++].qtde = 2;

produto[i].id = globalID++;

strcpy(produto[i].nome, "Flor Orquidea Ludisia Pt08");

strcpy(produto[i].categoria, "Jardim");

produto[i].preco = 18.99;

produto[i++].qtde = 4;

}