**Abstract**

This code is an abstract implementation of a simple online gadget store using C programming language. The store allows users to view a list of available products, add products to their shopping cart, and then display and manage the contents of the cart. Here's a breakdown of the different parts of the code:

1. \*Structures:\*

- `struct Product`: Represents a product available in the store, with attributes such as ID, name, price, and a pointer to the next product.

- `struct CartItem`: Represents an item in the user's shopping cart, with attributes including ID, name, price, quantity, and a pointer to the next cart item.

2. \*Functions for Creating Items:\*

- `createProduct`: Creates a new `Product` instance with the provided details.

- `createCartItem`: Creates a new `CartItem` instance with the provided details.

3. \*Displaying Products:\*

- `displayProducts`: Displays the list of available products along with their IDs, names, and prices.

4. \*Removing Items from the Cart:\*

- `removeCartItem`: Allows users to remove a specific item from their shopping cart based on its ID.

5. \*Freeing Memory:\*

- `freeCart`: Frees the memory allocated for cart items when the shopping is done.

6. \*Displaying Cart and Shopping:\*

- `displayCart`: Displays the user's shopping cart, including the items, prices, and quantities. Allows users to remove items from the cart.

- `shop`: Implements the shopping process. Users can input product IDs and quantities they want to purchase, and the items are added to the cart.

7. \*Main Function:\*

- `main`: The main function initializes the list of available products and interacts with the user. It displays a welcome message, prompts the user for their name, displays available products, initiates the shopping process, and displays the final cart and total bill.

The code creates a basic interactive shopping experience where users can select products, specify quantities, and view their cart before making a purchase. It also handles memory allocation and deallocation for products and cart items to avoid memory leaks. Keep in mind that this code is a simplified example and lacks advanced features such as error handling, user authentication, persistence, and a more sophisticated user interface.

**Objectives**

The objectives of this code are to create a simple interactive program for an online gadget store. The program allows users to simulate a shopping experience by interacting with products, adding items to a cart, managing the cart, and eventually displaying the final bill. Here are the main objectives of this code:

1. \*Display Available Products:\*

- Allow users to see a list of available gadgets in the store.

- Display each product's ID, name, and price.

2. \*Shopping Process:\*

- Enable users to select products they want to purchase by entering the product's ID.

- Prompt users for the quantity they want to purchase.

- Calculate the total cost of the items being added to the cart.

3. \*Shopping Cart Management:\*

- Allow users to view the contents of their shopping cart.

- Display each cart item's ID, name, price, and quantity.

- Provide an option for users to remove items from the cart.

4. \*Total Bill Calculation:\*

- Calculate and display the total bill amount based on the prices and quantities of items in the cart.

5. \*Memory Management:\*

- Allocate memory dynamically for product and cart item structures.

- Free memory when cart items are removed and when the program exits.

6. \*User Interaction:\*

- Interact with the user by displaying messages, prompts, and menus.

- Prompt the user to enter their name for a personalized experience.

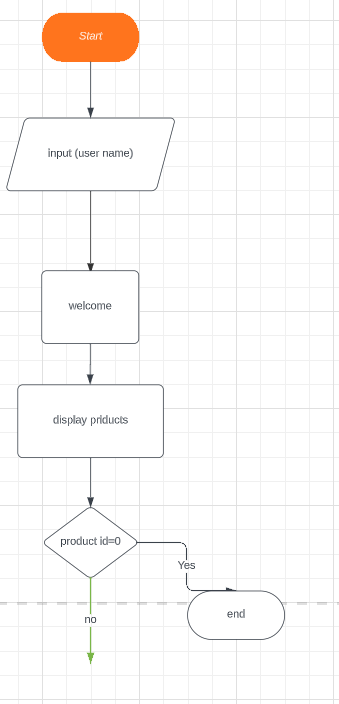
7. \*Simulation of Online Shopping:\*

- Simulate an online shopping experience, where users can browse products, add them to a cart, and check out.

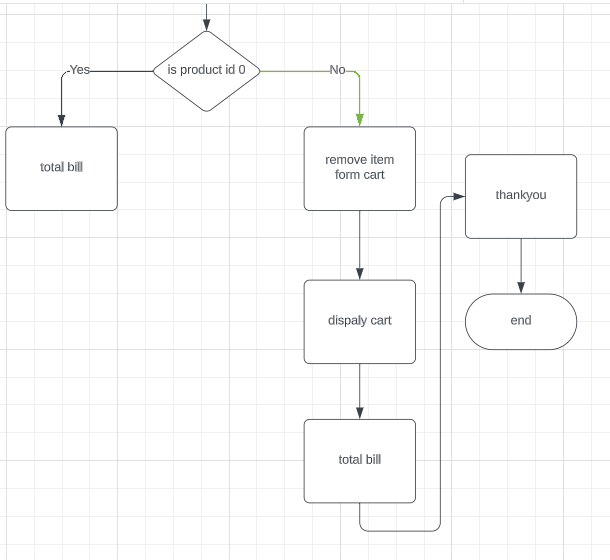
8. \*Simplified Structure:\*

- Provide a simplified demonstration of how an online store's backend might work, including product listing, cart management, and basic interactions.

It's important to note that while this code fulfills the stated objectives, it is a minimalistic representation and lacks several real-world features that a complete online store would require, such as user authentication, database integration, payment processing, error handling, and more advanced user interfaces.

**Flow chart**

****

****

**Explanation of code**

1. \*Include Statements:\*

- The code starts by including necessary header files for standard input/output and memory allocation.

2. \*Product and CartItem Structures:\*

- Two structures, `struct Product` and `struct CartItem`, are defined to represent products available in the store and items in the shopping cart, respectively.

- These structures store information like ID, name, price, quantity, and a pointer to the next item.

3. \*Create Product and CartItem Functions:\*

- `createProduct` and `createCartItem` functions are defined to create instances of `struct Product` and `struct CartItem`, respectively.

- These functions allocate memory, set the attributes, and return the newly created structures.

4. \*Display Products Function:\*

- `displayProducts` function displays the available products' details like ID, name, and price in a formatted table.

5. \*Remove CartItem Function:\*

- `removeCartItem` function allows removing a specific cart item by providing its ID.

- It searches for the item with the given ID and removes it from the cart.

6. \*Free Cart Function:\*

- `freeCart` function releases the memory allocated for cart items when they are no longer needed.

7. \*Display Cart Function:\*

- `displayCart` function displays the items in the shopping cart, including their details (ID, name, price, quantity).

- It allows the user to remove items from the cart by entering the item's ID.

8. \*Shop Function:\*

- `shop` function simulates the shopping process.

- Users can input product IDs and quantities they want to purchase.

- It calculates the total bill, adds items to the cart, and displays the cart.

9. \*Main Function:\*

- The `main` function initializes the list of available products.

- It prompts the user for their name and welcomes them to the store.

- It displays the available products using `displayProducts`.

- It initiates the shopping process using the `shop` function.

- After shopping is complete, it displays the final bill, thanks the user, and frees memory.

10. \*Initializing Product List:\*

- In the `main` function, a list of available products is initialized using the `createProduct` function.

- Product details (ID, name, price) are set for each product, and they are linked together.

11. \*User Interaction:\*

- Users are prompted to enter their name.

- Welcome message and product list are displayed.

- Users can select products and quantities to purchase.

12. \*Shopping Loop:\*

- The shopping process uses a loop where users can select items to add to their cart by providing the product ID and quantity.

- The loop continues until the user enters 0 to exit.

13. \*Display Cart and Total Bill:\*

- After shopping, the contents of the cart are displayed using the `displayCart` function.

- The total bill amount is calculated and displayed.

14. \*Freeing Memory:\*

- Memory allocated for cart items is freed using the `freeCart` function to prevent memory leaks.

15. \*End of Program:\*

- The program thanks the user for shopping and terminates.

Overall, this code implements a basic simulation of an online gadget store, allowing users to browse products, add them to their cart, manage the cart, and calculate the total bill. Keep in mind that this code is a simplified example and lacks error handling, database integration, and other features that a real online store would require.

**Implementation of code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Product

{

int id;

char name[50];

double price;

struct Product\* next;

};

struct CartItem

{

int id;

char name[50];

double price;

int quantity;

struct CartItem\* next;

};

struct Product\* createProduct(int id, const char\* name, double price)

{

struct Product\* newProduct = (struct Product\*)malloc(sizeof(struct Product));

newProduct->id = id;

strncpy(newProduct->name, name, sizeof(newProduct->name));

newProduct->price = price;

newProduct->next = NULL;

return newProduct;

}

struct CartItem\* createCartItem(int id, const char\* name, double price, int quantity)

{

struct CartItem\* newCartItem = (struct CartItem\*)malloc(sizeof(struct CartItem));

newCartItem->id = id;

strncpy(newCartItem->name, name, sizeof(newCartItem->name));

newCartItem->price = price;

newCartItem->quantity = quantity;

newCartItem->next = NULL;

return newCartItem;

}

void displayProducts(struct Product\* productList)

{

printf("Available gadgets are:\n");

printf("ID\tName\t\t\t\tPrice\n");

printf("----------------------------------------------------\n");

struct Product\* currentProduct = productList;

while (currentProduct != NULL) {

printf("%d\t%s\t\t%.2f\n", currentProduct->id, currentProduct->name, currentProduct->price);

currentProduct = currentProduct->next;

}

}

struct CartItem\* removeCartItem(struct CartItem\* cart, int itemId)

{

struct CartItem\* currentCartItem = cart;

struct CartItem\* prevCartItem = NULL;

while (currentCartItem != NULL) {

if (currentCartItem->id == itemId) {

if (prevCartItem == NULL) {

cart = currentCartItem->next;

} else {

prevCartItem->next = currentCartItem->next;

}

free(currentCartItem);

printf("Item removed from the cart.\n");

return cart;

}

prevCartItem = currentCartItem;

currentCartItem = currentCartItem->next;

}

printf("Item with ID %d not found in the cart.\n", itemId);

return cart;

}

void freeCart(struct CartItem\* cart)

{

struct CartItem\* currentCartItem = cart;

while (currentCartItem != NULL) {

struct CartItem\* temp = currentCartItem;

currentCartItem = currentCartItem->next;

free(temp);

}

}

void displayCart(struct CartItem\* cart)

{

printf("\nYour Cart:\n");

printf("ID\tName\t\t\t\t\tPrice\t\tQuantity\n");

printf("----------------------------------------------\n");

struct CartItem\* currentCartItem = cart;

while (currentCartItem != NULL) {

printf("%d\t%s\t\t%.2f\t\t%d\n", currentCartItem->id, currentCartItem->name, currentCartItem->price,

currentCartItem->quantity);

currentCartItem = currentCartItem->next;

}

int itemId;

printf("\nEnter the ID of the item you want to remove (0 to continue shopping): ");

scanf("%d", &itemId);

if (itemId != 0) {

cart = removeCartItem(cart, itemId);

displayCart(cart);

}

}

void shop(struct Product\* productList)

{

int itemId;

int quantity;

double totalPrice = 0.0;

struct CartItem\* cart = NULL;

printf("\nEnter the ID of the item you want to purchase (0 to exit): ");

scanf("%d", &itemId);

while (itemId != 0) {

struct Product\* currentProduct = productList;

while (currentProduct != NULL) {

if (currentProduct->id == itemId) {

printf("Enter the quantity you want to purchase: ");

scanf("%d", &quantity);

totalPrice += currentProduct->price \* quantity;

struct CartItem\* cartItem = createCartItem(currentProduct->id, currentProduct->name, currentProduct->price, quantity);

cartItem->next = cart;

cart = cartItem;

break;

}

currentProduct = currentProduct->next;

}

printf("\nEnter the ID of the next item you want to purchase (0 to exit): ");

scanf("%d", &itemId);

}

displayCart(cart);

printf("\nThank you for shopping with us!\n");

printf("Your total bill amount is: %.2f-/Rs\n", totalPrice);

freeCart(cart); // Free the memory of cart items before exiting

}

int main(void)

{

struct Product\* productList = createProduct(1, "samsung s24 ", 125000);

productList->next = createProduct(2, "apple 14 pro ", 149999);

productList->next->next = createProduct(3, "macbook m1 air", 98999);

productList->next->next->next = createProduct(4, "apple ipad air", 56000);

productList->next->next->next->next = createProduct(5, "apple VR ", 350000);

char name[50];

printf("Please enter your name:\n");

scanf("%s", name);

printf("\n\nHello %s, welcome to the BlipKart online gadget store\n", name);

displayProducts(productList);

shop(productList);

return 0;

}

**Output**

Please enter your name:

Vardhan

Hello Vardhan, welcome to the BlipKart online gadget store

Available gadgets are:

ID Name Price

----------------------------------------------------

1 samsung s24 125000.00

2 apple 14 pro 149999.00

3 macbook m1 air 98999.00

4 apple ipad air 56000.00

5 apple VR 350000.00

Enter the ID of the item you want to purchase (0 to exit): 1

Enter the quantity you want to purchase: 1

Enter the ID of the next item you want to purchase (0 to exit): 2

Enter the quantity you want to purchase: 1

Enter the ID of the next item you want to purchase (0 to exit): 3

Enter the quantity you want to purchase: 1

Enter the ID of the next item you want to purchase (0 to exit): 4

Enter the quantity you want to purchase: 1

Enter the ID of the next item you want to purchase (0 to exit): 0

Your Cart:

ID Name Price Quantity

----------------------------------------------

4 apple ipad air 56000.00 1

3 macbook m1 air 98999.00 1

2 apple 14 pro 149999.00 1

1 samsung s24 125000.00 1

Enter the ID of the item you want to remove (0 to continue shopping): 3

Item removed from the cart.

Your Cart:

ID Name Price Quantity

----------------------------------------------

4 apple ipad air 56000.00 1

2 apple 14 pro 149999.00 1

1 samsung s24 125000.00 1

Enter the ID of the item you want to remove (0 to continue shopping): 2

Item removed from the cart.

Your Cart:

ID Name Price Quantity

----------------------------------------------

4 apple ipad air 56000.00 1

1 samsung s24 125000.00 1

Enter the ID of the item you want to remove (0 to continue shopping): 0

Thank you for shopping with us!

Your total bill amount is: 429998.00-/Rs

Program ended with exit code: 0

Online

Shopping