#include <iostream>

#include <string>

using namespace std;

// Структура для представления элемента списка

struct Item {

string name;

int quantity;

double price;

Item\* next = nullptr; // Указатель на следующий элемент списка, инициализируем null

};

// Класс для работы со списком товаров

class ItemList {

private:

Item\* head = nullptr; // Указатель на первый элемент списка, инициализируем null

// Вспомогательная функция для удаления элемента списка

void deleteItem(Item\* item) {

delete item;

}

public:

// Деструктор для освобождения памяти, занятой списком

~ItemList() {

while (head) { // Пока есть элементы в списке

Item\* temp = head; // Сохраняем указатель на текущий элемент

head = head->next; // Перемещаем указатель на следующий элемент

deleteItem(temp); // Удаляем текущий элемент

}

}

// Метод для добавления элемента в начало списка

void addToFront(const string& name, int quantity, double price) {

head = new Item{name, quantity, price, head}; // Создаём новый элемент и устанавливаем его как голову списка

}

// Метод для добавления элемента в конец списка

void addToEnd(const string& name, int quantity, double price) {

Item\* newItem = new Item{name, quantity, price}; // Создаём новый элемент

if (!head) { // Если список пуст

head = newItem; // Устанавливаем новый элемент как голову списка

return;

}

Item\* temp = head; // Указатель для обхода списка

while (temp->next) // Пока есть следующий элемент

temp = temp->next; // Перемещаемся к следующему элементу

temp->next = newItem; // Добавляем новый элемент в конец списка

}

// Метод для добавления элемента после заданного элемента

void addAfter(const string& afterName, const string& name, int quantity, double price) {

Item\* temp = head; // Указатель для обхода списка

while (temp && temp->name != afterName) // Ищем элемент с заданным именем

temp = temp->next; // Перемещаемся к следующему элементу

if (temp) // Если нашли элемент

temp->next = new Item{name, quantity, price, temp->next}; // Вставляем новый элемент после найденного

}

// Метод для добавления элемента перед заданным элементом

void addBefore(const string& beforeName, const string& name, int quantity, double price) {

if (!head) return; // Если список пуст, выходим из метода

if (head->name == beforeName) { // Если нужно добавить перед первым элементом

addToFront(name, quantity, price); // Добавляем в начало списка

return; // Выходим из метода

}

Item\* temp = head; // Указатель для обхода списка

while (temp->next && temp->next->name != beforeName) // Ищем элемент перед заданным

temp = temp->next; // Перемещаемся к следующему элементу

if (temp->next) // Если нашли элемент

temp->next = new Item{name, quantity, price, temp->next}; // Вставляем новый элемент перед найденным

}

// Метод для удаления элемента по имени

void remove(const string& name) {

if (!head) return; // Если список пуст, выходим из метода

if (head->name == name) { // Если нужно удалить первый элемент

Item\* temp = head; // Сохраняем указатель на первый элемент

head = head->next; // Перемещаем голову на следующий элемент

deleteItem(temp); // Удаляем первый элемент

return;

}

Item\* temp = head; // Указатель для обхода списка

while (temp->next && temp->next->name != name) // Ищем элемент с заданным именем

temp = temp->next; // Перемещаемся к следующему элементу

if (temp->next) { // Если нашли элемент

Item\* toDelete = temp->next; // Сохраняем указатель на элемент, который нужно удалить

temp->next = toDelete->next; // Переподключаем указатели

deleteItem(toDelete); // Удаляем элемент

}

}

// Метод для отображения содержимого списка

void display() const {

for (Item\* temp = head; temp; temp = temp->next) { // Обходим список

cout << "Name: " << temp->name // Выводим название

<< ", Quantity: " << temp->quantity // Выводим количество

<< ", Price: " << temp->price << endl; // Выводим цену

}

}

};

// Функция для ввода данных

void inputItemData(string& name, int& quantity, double& price) {

cout << "Введите название товара: "; // Запрос названия

getline(cin, name); // Чтение названия

cout << "Введите количество: "; // Запрос количества

cin >> quantity; // Чтение количества

cout << "Введите цену: "; // Запрос цены

cin >> price; // Чтение цены

cin.ignore(); // Игнорируем символ новой строки после ввода числа

}

// Главная функция программы

int main() {

ItemList list; // Создание экземпляра списка

int choice; // Переменная для хранения выбора пользователя

do {

// Меню действий

cout << "\nМеню:\n"

<< "1. Добавить элемент в начало списка\n"

<< "2. Добавить элемент в конец списка\n"

<< "3. Добавить элемент после заданного\n"

<< "4. Добавить элемент перед заданным\n"

<< "5. Удалить элемент\n"

<< "6. Вывести содержимое списка\n"

<< "0. Выход\n"

<< "Выберите действие: ";

cin >> choice; // Чтение выбора пользователя

cin.ignore(); // Игнорируем символ новой строки

string name, afterName, beforeName; // Переменные для хранения данных

int quantity; // Переменная для хранения количества

double price; // Переменная для хранения цены

switch (choice) { // Обработка выбора пользователя

case 1:

inputItemData(name, quantity, price); // Ввод данных

list.addToFront(name, quantity, price); // Добавление в начало списка

break;

case 2:

inputItemData(name, quantity, price); // Ввод данных

list.addToEnd(name, quantity, price); // Добавление в конец списка

break;

case 3:

cout << "Введите имя элемента после которого добавить: ";

getline(cin, afterName); // Чтение имени элемента

inputItemData(name, quantity, price); // Ввод данных

list.addAfter(afterName, name, quantity, price); // Добавление после заданного

break;

case 4:

cout << "Введите имя элемента перед которым добавить: ";

getline(cin, beforeName); // Чтение имени элемента

inputItemData(name, quantity, price); // Ввод данных

list.addBefore(beforeName, name, quantity, price); // Добавление перед заданным

break;

case 5:

cout << "Введите имя элемента для удаления: ";

getline(cin, name); // Чтение имени элемента

list.remove(name); // Удаление элемента из списка

break;

case 6:

cout << "Содержимое списка:\n";

list.display(); // Вывод содержимого списка

break;

case 0:

cout << "Выход из программы.\n";

break;

default:

cout << "Неверный выбор. Попробуйте снова.\n";

}

} while (choice != 0); // Продолжаем, пока пользователь не выберет выход

return 0;

}