### 

Выполнил

студент группы КТбо1-2 И. Н. Акимов

Принял

доцент кафедры САиТ В. С. Лапшин

Таганрог 2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №2

по дисциплине «Основы алгоритмизации и программирования»

**«РАЗРАБОТКА ДИАГРАММЫ КЛАССОВ НА С++»**

Вариант 2

Оглавление

[1](#_Toc168689453)

[Техническое задание 3](#_Toc168689454)

[Цель задания 3](#_Toc168689455)

[Задача 3](#_Toc168689456)

[Ход работы 4](#_Toc168689457)

[Структура UML-диаграммы 4](#_Toc168689458)

[Написание программы 5](#_Toc168689459)

[Пример программы 5](#_Toc168689460)

[Вывод 6](#_Toc168689461)

[Листинг 7](#_Toc168689462)

[class.h 7](#_Toc168689463)

[class.cpp 7](#_Toc168689464)

[main.cpp 10](#_Toc168689465)

# Техническое задание

## Цель задания

Используя среду разработки MS Visual Studio, разработать и реализовать диаграмму классов на С++.

## Задача

Задачи индивидуального задания:

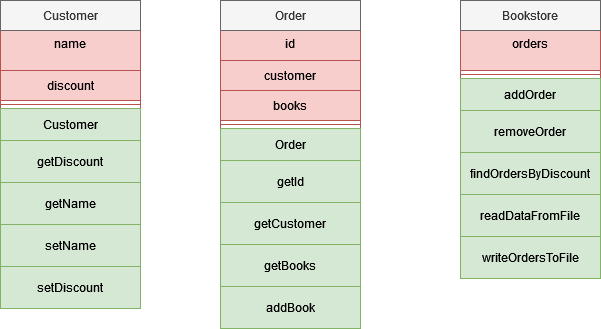
1. Разработать UML-диаграмму классов (в draw.io);
2. Реализовать структуру классов на С++ на основе созданной   
   UML-диаграммы;
3. Разработать методы контроля и редактирования параметров объектов.

4. Разработать программный код метода соответственно таблице индивидуальных заданий;

Заданием моего варианта стали классы, с помощью которых необходимо: Найти заказы, сделанные клиентами с заданными значениями скидки.

# Ход работы

## Структура UML-диаграммы

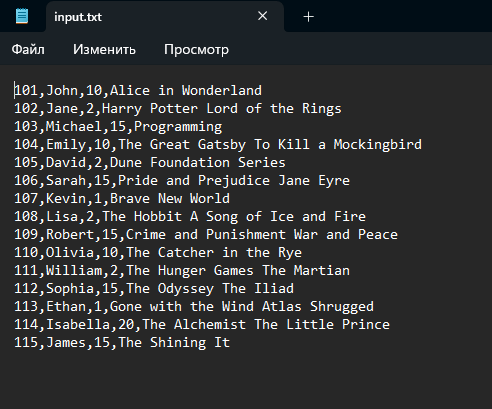


## Написание программы

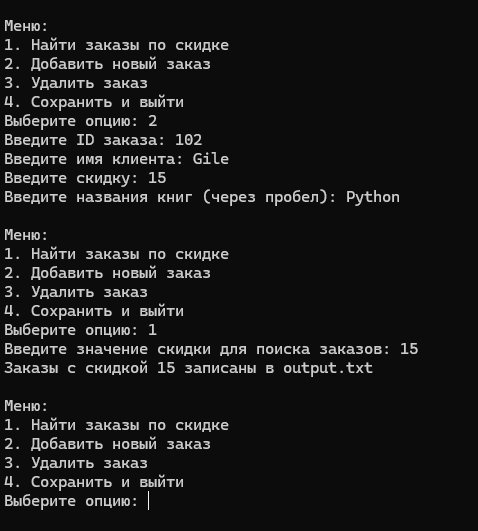
Реализация вышеуказанной диаграммы классов произведена стандартным для C++ способом: в class.h прописано объявление класса, его полей и методов; в class.cpp прописано определение всех методов классов; в main.cpp представлено диалоговое окно вызова методов класса для оперирования данными классов. Так же делаем проверку корректности введённых пользователем данных.

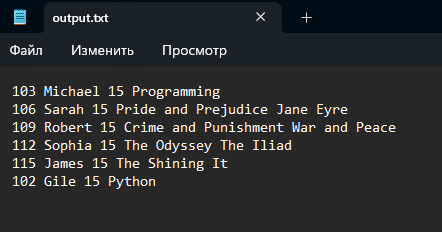
## Пример программы

Входные данные:



Работа программы:





# Вывод

В ходе индивидуального задания я научился разрабатывать UML-диаграммы в графическом приложении draw.io и вновь ознакомился с методами реализации классов на С++.

# Листинг

## class.h

#ifndef CLASS\_H

#define CLASS\_H

#include <string>

#include <vector>

using namespace std;

// Класс, представляющий клиента

class Customer

{

private:

string name; // Имя клиента

float discount; // Величина скидки клиента

public:

// Конструктор клиента

Customer(const string& name, float discount);

// Геттеры для имени и скидки

string getName() const;

float getDiscount() const;

// Сеттеры для имени и скидки

void setName(const string& newName);

void setDiscount(float newDiscount);

};

// Класс, представляющий заказ на книги

class Order

{

private:

int id; // Идентификатор заказа

Customer customer; // Клиент, сделавший заказ

vector<string> books; // Список заказанных книг

public:

// Конструктор заказа

Order(int id, const Customer& customer, const vector<string>& books);

// Геттеры для id, клиента и списка книг

int getId() const;

Customer getCustomer() const;

vector<string> getBooks() const;

// Метод для добавления книги к заказу

void addBook(const string& newBook);

};

// Класс, представляющий книжный магазин

class Bookstore

{

private:

vector<Order> orders; // Список заказов в книжном магазине

public:

// Методы для управления заказами

void addOrder(const Order& order); // Добавление заказа в список

void removeOrder(int orderId); // Удаление заказа по id

vector<Order> findOrdersByDiscount(float discount) const; // Поиск заказов по величине скидки

// Методы для работы с файлами

void readDataFromFile(const string& filename); // Чтение данных из файла

void writeOrdersToFile(const string& filename, const vector<Order>& orders, bool append = false) const; // Запись данных в файл

};

// Функции для взаимодействия с пользователем

void addOrderInteractively(Bookstore& bookstore); // Функция для интерактивного добавления заказа

void removeOrderInteractively(Bookstore& bookstore); // Функция для интерактивного удаления заказа

#endif // CLASS\_H

## class.cpp

#include "class.h"

#include <iostream>

#include <fstream>

#include <sstream>

#include <algorithm>

using namespace std;

// Конструктор клиента

Customer::Customer(const string& name, float discount)

: name(name), discount(discount) {}

// Геттер для величины скидки

float Customer::getDiscount() const

{

return discount;

}

// Геттер для имени клиента

string Customer::getName() const

{

return name;

}

// Сеттер для имени клиента с проверкой на пустое значение

void Customer::setName(const string& newName)

{

if (!newName.empty())

{

name = newName;

}

else

{

cerr << "Ошибка: имя не может быть пустым." << endl;

}

}

// Сеттер для величины скидки с проверкой на отрицательное значение

void Customer::setDiscount(float newDiscount)

{

if (newDiscount >= 0.0f)

{

discount = newDiscount;

}

else

{

cerr << "Ошибка: скидка не может быть отрицательной." << endl;

}

}

// Конструктор заказа

Order::Order(int id, const Customer& customer, const vector<string>& books)

: id(id), customer(customer), books(books) {}

// Геттер для id заказа

int Order::getId() const

{

return id;

}

// Геттер для клиента, сделавшего заказ

Customer Order::getCustomer() const

{

return customer;

}

// Геттер для списка заказанных книг

vector<string> Order::getBooks() const

{

return books;

}

// Метод для добавления книги к заказу

void Order::addBook(const string& newBook)

{

if (!newBook.empty())

{

books.push\_back(newBook);

}

else

{

cerr << "Ошибка: название книги не может быть пустым." << endl;

}

}

// Добавление заказа в книжный магазин

void Bookstore::addOrder(const Order& order)

{

orders.push\_back(order);

}

// Удаление заказа из книжного магазина по id

void Bookstore::removeOrder(int orderId)

{

orders.erase(remove\_if(orders.begin(), orders.end(), [orderId](const Order& order)

{ return order.getId() == orderId; }),

orders.end());

}

// Поиск заказов по величине скидки

vector<Order> Bookstore::findOrdersByDiscount(float discount) const

{

vector<Order> result;

for (const auto& order : orders)

{

if (order.getCustomer().getDiscount() == discount)

{

result.push\_back(order);

}

}

return result;

}

// Чтение данных из файла и заполнение заказов

void Bookstore::readDataFromFile(const string& filename)

{

ifstream infile(filename);

if (!infile.is\_open())

{

cerr << "Ошибка открытия файла " << filename << endl;

return;

}

orders.clear(); // Очищаем текущие заказы перед загрузкой новых данных

string line;

while (getline(infile, line))

{

// Используем stringstream для разбора строки по запятым

stringstream ss(line);

string segment;

vector<string> segments;

while (getline(ss, segment, ','))

{

segments.push\_back(segment);

}

// Проверяем, что мы успешно получили все необходимые сегменты данных

if (segments.size() < 4)

{

cerr << "Ошибка чтения данных из файла: недостаточно данных" << endl;

continue;

}

// Извлекаем данные из сегментов

int orderId = stoi(segments[0]);

string name = segments[1];

float discount = stof(segments[2]);

// Список заказанных книг начинается с третьего элемента в segments

vector<string> books;

for (size\_t i = 3; i < segments.size(); ++i)

{

books.push\_back(segments[i]);

}

// Создаем объекты и добавляем в хранилище заказов

Customer customer(name, discount);

Order order(orderId, customer, books);

addOrder(order);

}

infile.close(); // Закрываем файл после чтения

}

// Запись данных заказов в файл

void Bookstore::writeOrdersToFile(const string& filename, const vector<Order>& orders, bool append) const

{

ofstream outfile;

if (append)

{

outfile.open(filename, ios\_base::app); // Открыть файл для добавления в конец

}

else

{

outfile.open(filename); // Перезаписать файл

}

for (const auto& order : orders)

{

// Записываем id заказа, имя клиента, величину скидки и список книг через пробел

outfile << order.getId() << " " << order.getCustomer().getName() << " "

<< order.getCustomer().getDiscount() << " ";

for (const auto& book : order.getBooks())

{

outfile << book << " ";

}

outfile << endl;

}

}

// Функция для интерактивного добавления заказа

void addOrderInteractively(Bookstore& bookstore)

{

int orderId;

float discount;

string name, book;

vector<string> books;

cout << "Введите ID заказа: ";

while (!(cin >> orderId))

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cerr << "Ошибка: введите целое число." << endl;

cout << "Введите ID заказа: ";

}

cin.ignore(); // Очищаем буфер ввода

cout << "Введите имя клиента: ";

getline(cin, name);

cout << "Введите скидку: ";

while (!(cin >> discount) || discount < 0.0f)

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cerr << "Ошибка: введите положительное число." << endl;

cout << "Введите скидку: ";

}

cin.ignore(); // Очищаем буфер ввода

cout << "Введите названия книг (через пробел): ";

getline(cin, book);

istringstream iss(book);

while (iss >> book)

{

books.push\_back(book);

}

Customer customer(name, discount);

Order order(orderId, customer, books);

bookstore.addOrder(order);

}

// Функция для интерактивного удаления заказа по ID

void removeOrderInteractively(Bookstore& bookstore)

{

int orderId;

cout << "Введите ID заказа для удаления: ";

while (!(cin >> orderId))

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cerr << "Ошибка: введите целое число." << endl;

cout << "Введите ID заказа для удаления: ";

}

bookstore.removeOrder(orderId);

}

## main.cpp

#include <iostream>

#include "class.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian"); // Установка локали для вывода на русском языке

Bookstore bookstore;

bookstore.readDataFromFile("input.txt"); // Чтение данных из файла input.txt при запуске программы

bool running = true;

while (running)

{

// Вывод меню пользователю

cout << "\nМеню:\n"

<< "1. Найти заказы по скидке\n"

<< "2. Добавить новый заказ\n"

<< "3. Удалить заказ\n"

<< "4. Сохранить и выйти\n"

<< "Выберите опцию: ";

int choice;

while (!(cin >> choice))

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cerr << "Ошибка: введите целое число." << endl;

cout << "Выберите опцию: ";

}

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

switch (choice)

{

case 1:

{

float targetDiscount;

cout << "Введите значение скидки для поиска заказов: ";

while (!(cin >> targetDiscount) || targetDiscount < 0.0f)

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cerr << "Ошибка: введите положительное число." << endl;

cout << "Введите значение скидки для поиска заказов: ";

}

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

// Поиск заказов по введенной скидке

auto foundOrders = bookstore.findOrdersByDiscount(targetDiscount);

if (foundOrders.empty())

{

cout << "Заказы с скидкой " << targetDiscount << " не найдены" << endl;

}

else

{

// Записываем найденные заказы в файл output.txt

bookstore.writeOrdersToFile("output.txt", foundOrders, false);

cout << "Заказы с скидкой " << targetDiscount << " записаны в output.txt" << endl;

}

break;

}

case 2:

addOrderInteractively(bookstore); // Функция для интерактивного добавления нового заказа

break;

case 3:

removeOrderInteractively(bookstore); // Функция для интерактивного удаления заказа

break;

case 4:

// Записываем все заказы в файл input.txt перед завершением программы

bookstore.writeOrdersToFile("input.txt", bookstore.findOrdersByDiscount(-1), true);

running = false; // Выход из цикла, завершение работы программы

break;

default:

cerr << "Недопустимый выбор. Попробуйте снова." << endl;

}

}

return 0;

}