

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра базовой подготовки

**Расчетно-графическая работа**

Дисциплина: Языки программирования

Ф.И.О студента: Игнатова Вероника Алексеевна

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Инженерия программного обеспечения

Номер группы: ФИ- 202

Номер зачетной книжки: 220118

Проверил: Ковригин Алексей Викторович, кандидат педагогических наук, доцент

Новосибирск *2023*

**Содержание**

[1. Описание частей кода 3](#_Toc155809194)

[1.1. Описание класса Order 3](#_Toc155809195)

[1.2. Описание класса OrderManager 3](#_Toc155809196)

[1.3. Описание функции main 5](#_Toc155809197)

[2. Цельный код 7](#_Toc155809198)

# Описание частей кода

## Описание класса Order

Эта часть кода представляет класс Order с тремя приватными переменными: name, date и type, представляющими имя заказа, дату и тип заказа соответственно. Класс имеет конструктор, который инициализирует переменные name, date и type заданными значениями. Также класс содержит три открытых метода, которые возвращают значения приватных переменных: getName() -возвращает имя заказа, getDate() - возвращает дату заказа, и getType() - возвращает тип заказа.

С помощью этого класса можно создавать объекты заказов, и получать доступ к их свойствам с помощью методов класса.

// Определяем класс "Заказ"

class Order {

private:

std::string name;

std::string date;

std::string type;

public:

Order(std::string name, std::string date, std::string type) : name(name), date(date), type(type) {}

std::string getName() { return name; }

std::string getDate() { return date; }

std::string getType() { return type; }

};

## Описание класса OrderManager

Данный код содержит класс OrderManager, который работает с данными заказов. У класса OrderManager есть приватное свойство orders, которое является вектором объектов класса Order. Класс содержит методы addOrder для добавления заказа, removeOrder для удаления заказа по индексу, printOrders для вывода информации о заказах и saveToFile для сохранения информации о заказах в файл, определения номера последней строки и добавления новых строк с информацией о заказах. Метод loadFromFile загружает информацию о заказах из файла и создает новые объекты заказов на основе этой информации, добавляя их в вектор orders. Оба метода saveToFile и loadFromFile выводят сообщения об успешном или неуспешном выполнении операции.   
Этот класс предоставляет набор методов для управления заказами, добавления, удаления, печати всех заказов, сохранения и загрузки их из файла.

class OrderManager {

private:

std::vector<Order> orders;

public:

void addOrder(Order order) {

orders.push\_back(order);

}

void removeOrder(int index) {

if (index >= 0 && index < orders.size()) {

orders.erase(orders.begin() + index);

}

}

void printOrders() {

for (int i = 0; i < orders.size(); i++) {

std::cout << "Заказ " << i + 1 << ":" << std::endl;

std::cout << "Имя: " << orders[i].getName() << std::endl;

std::cout << "Дата: " << orders[i].getDate() << std::endl;

std::cout << "Тип: " << orders[i].getType() << std::endl;

std::cout << std::endl;

}

}

void saveToFile(std::string filename) {

std::ifstream inputFile(filename);

std::vector<std::string> lines;

std::string line;

if (inputFile.is\_open()) {

while (getline(inputFile, line)) {

lines.push\_back(line);

}

inputFile.close();

}

// Определить номер последней строки

int lastLineNumber = lines.size();

std::ofstream file(filename, std::ios::app);

if (file.is\_open()) {

for (int i = 0; i < orders.size(); i++) {

file << lastLineNumber + i + 1 << " Заказ: " << "Имя: " << orders[i].getName() << "," << "Дата: " << orders[i].getDate() << "," << "Тип: " << orders[i].getType() << std::endl;

}

file.close();

std::cout << "Сохранено в файл: " << filename << std::endl;

}

else {

std::cout << "Ошибка при открытии файла." << std::endl;

}

}

void loadFromFile(std::string filename) {

std::ifstream file(filename);

if (file.is\_open()) {

orders.clear();

std::string line;

while (std::getline(file, line)) {

std::string name, date, type;

size\_t pos1 = line.find(",");

size\_t pos2 = line.find(",", pos1 + 1);

name = line.substr(0, pos1);

date = line.substr(pos1 + 1, pos2 - pos1 - 1);

type = line.substr(pos2 + 1);

Order order(name, date, type);

orders.push\_back(order);

}

file.close();

std::cout << "Загружено из файла: " << filename << std::endl;

}

else {

std::cout << "Ошибка при открытии файла." << std::endl;

}

}

};

## Описание функции main

Данный код представляет собой функцию main, которая реализует применение класса OrderManager для управления заказами. У пользователя есть возможность выбирать действия по добавлению, удалению, выводу и сохранению заказов через консольное меню.

Код начинается с инициализации генератора случайных чисел srand(time(NULL)) и установки кодировки для корректного вывода кириллических символов (1251) с помощью функций SetConsoleCP и SetConsoleOutputCP. Затем создается объект orderManager класса OrderManager. Далее следует бесконечный цикл do-while, в котором пользователь может выбирать действия, вводя соответствующие цифры. С помощью оператора switch программа выполняет соответствующее действие, в зависимости от выбора пользователя. Например, для добавления нового заказа, программа запрашивает у пользователя имя, дату и тип заказа и добавляет его через метод addOrder. Аналогично для удаления заказа, вывода всех заказов, сохранения и загрузки из файла.

Цикл продолжается до тех пор, пока пользователь не выберет выход (ввод 0). Затем функция возвращает ноль, что означает успешное завершение программы.

int main() {

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

OrderManager orderManager;

int choice;

std::string name, date, type, filename;

do {

std::cout << "Выберите действие:\n";

std::cout << "1. Добавить заказ\n";

std::cout << "2. Удалить заказ\n";

std::cout << "3. Вывести все заказы\n";

std::cout << "4. Сохранить в файл\n";

std::cout << "5. Загрузить из файла\n";

std::cout << "0. Выход\n";

std::cout << "Ваш выбор: ";

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Введите имя: ";

std::cin >> name;

std::cout << "Введите дату: ";

std::cin >> date;

std::cout << "Введите тип: ";

std::cin >> type;

orderManager.addOrder(Order(name, date, type));

std::cout << "Заказ добавлен\n";

break;

case 2:

int index;

std::cout << "Введите индекс заказа: ";

std::cin >> index;

orderManager.removeOrder(index - 1);

std::cout << "Заказ удален\n";

break;

case 3:

orderManager.printOrders();

break;

case 4:

std::

cout << "Введите имя файла для сохранения: ";

std::cin >> filename;

orderManager.saveToFile(filename);

break;

case 5:

std::cout << "Введите имя файла для загрузки: ";

std::cin >> filename;

orderManager.loadFromFile(filename);

break;

case 0:

std::cout << "Выход\n";

break;

default:

std::cout << "Неверный выбор\n";

break;

}

std::cout << std::endl;

} while (choice != 0);

return 0;

}

# Цельный код

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <windows.h>

using namespace std;

// Определяем класс "Заказ"

class Order {

private:

std::string name;

std::string date;

std::string type;

public:

Order(std::string name, std::string date, std::string type) : name(name), date(date), type(type) {}

std::string getName() { return name; }

std::string getDate() { return date; }

std::string getType() { return type; }

};

// Определяем класс "Управление заказами"

class OrderManager {

private:

std::vector<Order> orders;

public:

void addOrder(Order order) {

orders.push\_back(order);

}

void removeOrder(int index) {

if (index >= 0 && index < orders.size()) {

orders.erase(orders.begin() + index);

}

}

void printOrders() {

for (int i = 0; i < orders.size(); i++) {

std::cout << "Заказ " << i + 1 << ":" << std::endl;

std::cout << "Имя: " << orders[i].getName() << std::endl;

std::cout << "Дата: " << orders[i].getDate() << std::endl;

std::cout << "Тип: " << orders[i].getType() << std::endl;

std::cout << std::endl;

}

}

void saveToFile(std::string filename) {

std::ifstream inputFile(filename);

std::vector<std::string> lines;

std::string line;

if (inputFile.is\_open()) {

while (getline(inputFile, line)) {

lines.push\_back(line);

}

inputFile.close();

}

// Определить номер последней строки

int lastLineNumber = lines.size();

std::ofstream file(filename, std::ios::app);

if (file.is\_open()) {

for (int i = 0; i < orders.size(); i++) {

file << lastLineNumber + i + 1 << " Заказ: " << "Имя: " << orders[i].getName() << "," << "Дата: " << orders[i].getDate() << "," << "Тип: " << orders[i].getType() << std::endl;

}

file.close();

std::cout << "Сохранено в файл: " << filename << std::endl;

}

else {

std::cout << "Ошибка при открытии файла." << std::endl;

}

}

void loadFromFile(std::string filename) {

std::ifstream file(filename);

if (file.is\_open()) {

orders.clear();

std::string line;

while (std::getline(file, line)) {

std::string name, date, type;

size\_t pos1 = line.find(",");

size\_t pos2 = line.find(",", pos1 + 1);

name = line.substr(0, pos1);

date = line.substr(pos1 + 1, pos2 - pos1 - 1);

type = line.substr(pos2 + 1);

Order order(name, date, type);

orders.push\_back(order);

}

file.close();

std::cout << "Загружено из файла: " << filename << std::endl;

}

else {

std::cout << "Ошибка при открытии файла." << std::endl;

}

}

};

int main() {

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

OrderManager orderManager;

int choice;

std::string name, date, type, filename;

do {

std::cout << "Выберите действие:\n";

std::cout << "1. Добавить заказ\n";

std::cout << "2. Удалить заказ\n";

std::cout << "3. Вывести все заказы\n";

std::cout << "4. Сохранить в файл\n";

std::cout << "5. Загрузить из файла\n";

std::cout << "0. Выход\n";

std::cout << "Ваш выбор: ";

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Введите имя: ";

std::cin >> name;

std::cout << "Введите дату: ";

std::cin >> date;

std::cout << "Введите тип: ";

std::cin >> type;

orderManager.addOrder(Order(name, date, type));

std::cout << "Заказ добавлен\n";

break;

case 2:

int index;

std::cout << "Введите индекс заказа: ";

std::cin >> index;

orderManager.removeOrder(index - 1);

std::cout << "Заказ удален\n";

break;

case 3:

orderManager.printOrders();

break;

case 4:

std::

cout << "Введите имя файла для сохранения: ";

std::cin >> filename;

orderManager.saveToFile(filename);

break;

case 5:

std::cout << "Введите имя файла для загрузки: ";

std::cin >> filename;

orderManager.loadFromFile(filename);

break;

case 0:

std::cout << "Выход\n";

break;

default:

std::cout << "Неверный выбор\n";

break;

}

std::cout << std::endl;

} while (choice != 0);

return 0;

}