Федеральное агентство связи

Сибирский Государственный Университет Телекоммуникаций и Информатики

Кафедра ТС и ВС

Лабораторная работа №1

По дисциплине: Объектно-ориентированное программирование

Выполнил: Крахмальный Кирилл Вячеславович

Группа ИКС433

Вариант: 12

Проверила: Нейдорф П.Я

Новосибирск, 2025 г

**Лабораторная работа №1**

**Тема: «Введение в объектно-ориентированное программирование. Простые классы»**

**Цель работы:** ознакомиться с понятиями класса и объекта, научиться создавать простые классы, их поля и методы.

**Задание:**

Создать собственный класс согласно варианту задания. Создаваемый класс должен содержать **несколько полей** (по выбору студента), типы которых соответствуют варианту задания (можете ввести дополнительные поля), **конструктор** для инициализации значений полей объекта, **методы** для изменения значений полей объекта и для получения этих значений, кроме того **метод для вывода** всей имеющейся информации об объекте.

| № Варианта | Суперкласс | Поля |
| --- | --- | --- |
| 12 | Места развлечений | Тип учреждения (театр, выставка…) Вместимость |

**Ход работы:**

1. **Программа на языке С++**

#include <iostream>

#include <string>

using namespace std;

class Events {

private:

string type;

int box;

public:

Events(string t, int b) {

type = t;

box = b;

}

void outputTo() {

cout << "Тип мероприятия: " << type << endl;

cout << "Вместимость: " << box << endl;

}

string get\_type() {

return type;

}

void set\_type(string t) {

type = t;

}

int get\_box() {

return box;

}

void set\_box(int b) {

box = b;

}

};

int main() {

Events Theatre("Опера", 1000);

Events \*Concert = new Events("Симфонический концерт", 500);

Theatre.outputTo();

Concert->outputTo();

Theatre.set\_box(Theatre.get\_box() + 100);

cout << "Вместимость театра: " << Theatre.get\_box() << "\n";

cout << "Вместимость концерта: " << Concert->get\_box() << endl;

Theatre.outputTo();

Concert->outputTo();

delete Concert;

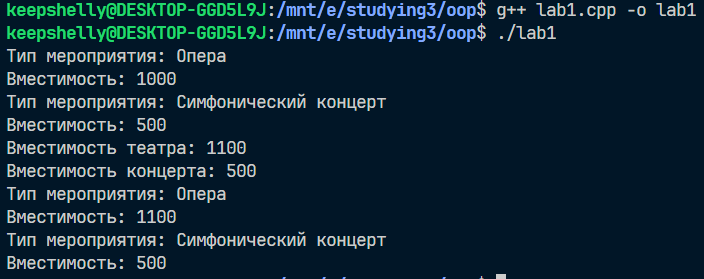
return 0;

}

**2. Комментарии к программе**

Класс события включает в себя тип мероприятия и его вместимость,через геттеры и сеттеры присваиваю поля к переменным конкретным, чтобы можно было их возвратить, в главной функции вывожу театр и концерт, где театр сделан динамически, и проверил динамику через добавление нового значения к вместимости театра. Опера поменялось в ходе работы на 1100(было 1000)

**3. Результаты выполнения программы**



**4. Ответы на контрольные вопросы**

## 1. Класс

**Класс** - это как чертёж или шаблон для создания объектов. Представьте, что класс - это рецепт приготовления блюда. В рецепте описано, какие ингредиенты нужны и какие действия выполнить, но самого блюда ещё нет. Класс определяет, какие данные будут хранить объекты и какие действия они смогут выполнять.

## 2. Объект

**Объект** - это конкретная вещь, созданная по чертежу класса. Если класс - это рецепт, то объект - это реальное блюдо, приготовленное по этому рецепту. Например, класс "Автомобиль" - это общее описание, а объекты - это конкретные машины: "мой синий Ford", "соседский красный Toyota".

## 3. Поля и методы класса

**Поля** - это характеристики или свойства объекта. Например, у объекта "Студент" поля: имя, возраст, средний балл. Это данные, которые объект хранит.

**Методы** - это действия, которые объект может выполнять. Например, у объекта "Студент" методы: "учиться", "сдавать экзамен", "получать оценку".

## 4. Статические поля и методы

**Статические поля** - это общая информация для всех объектов класса. Например, если у класса "Студент" есть статическое поле "количество студентов", то оно будет одинаковым для всех объектов и покажет общее число созданных студентов.

**Статические методы** - это действия, которые относятся ко всему классу, а не к конкретному объекту. Например, метод "посчитать общее количество студентов" - он работает со всей группой, а не с отдельным студентом.

## 5. Модификаторы доступа

Это правила, определяющие, кто может обращаться к полям и методам класса:

* **private (закрытый)** - доступно только внутри класса. Как личный дневник - только владелец может его читать.
* **protected (защищённый)** - доступно внутри класса и его "потомкам". Как семейные секреты - знают только члены семьи.
* **public (открытый)** - доступно всем. Как публичная информация - любой может её использовать.

## 6. Преимущества объектно-ориентированного программирования

**Инкапсуляция** - объединение данных и действий в одном объекте. Как автомобиль: вам не нужно знать, как работает двигатель, чтобы ездить - просто нажимаете педали.

**Наследование** - возможность создавать новые классы на основе существующих. Как наследование профессии в семье: сын врача может стать врачом, унаследовав знания, но добавив что-то своё.

**Полиморфизм** - способность объектов по-разному выполнять одни и те же действия. Как животные: все издают звуки, но кошка мяукает, а собака лает.

**Абстракция** - работа с объектами на высоком уровне, без погружения в детали. Как вождение автомобиля: вы думаете о направлении, а не о том, как работает каждый механизм.

## Главные преимущества:

* **Упрощение сложных систем** - разбиваем на понятные объекты
* **Повторное использование кода** - созданные классы можно использовать многократно
* **Лёгкость сопровождения** - изменения в одном объекте не ломают всю систему
* **Более ясная структура** - код отражает реальные предметы и процессы
* **Гибкость** - легко добавлять новую функциональность