МИНИСТЕРСТВО НАУКИ и высшего образования

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**Отчёт по лабораторной работе № 1**

**по учебной дисциплине «Системное программирование»**

**Тема: «Проектирование классов»**

Выполнил(а) студент(ка)

специальности 09.02.07

Информационные системы и

программирование

IV курса группы 42919/7

Побирчев Никита Андреевич

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург,

2024

**Лабораторная работа №1**

**Цель работы:**

Изучение структуры класса, механизм создания и использования, описание членов-данных класса и методов доступа к ним.

**Задание:**

- Необходимо создать массив объектов класса Прямоугольник с двумя свойствами, сторонами прямоугольника. Так же необходимо включить такие операции как: увеличение/уменьшение размера любой из сторон на определенный процент, вычисление периметра и площади, а также вычисление диагонали. Пользователь выбирает элемент массива и выполняемую операцию.

**Индивидуальное задание:**

- Необходимо создать 5 классов по теме Авиационная и космическая техника. Каждый из них должен содержать не менее 3 свойств, 3 операции. Имена классов – существительные.

**Ход работы:**

Создал класс Rectangle, инициализировал свойства его двух сторон (Рисунок 1).

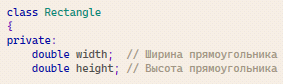


Рисунок 1 - Инициализация класса Rectangle

Добавил конструктор по умолчанию, а также конструктор запрашивающий все аргументы (Рисунок 2).



Рисунок 2 - Конструктор по умолчанию

Добавил методы вычисления периметра, площади и диагонали прямоугольника (Рисунок 3).

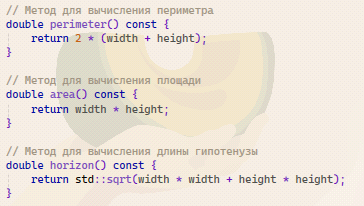


Рисунок 3 - Методы вычисления периметра, площади и диагонали прямоугольника

Добавил метод изменения сторон прямоугольника (Рисунок 4).

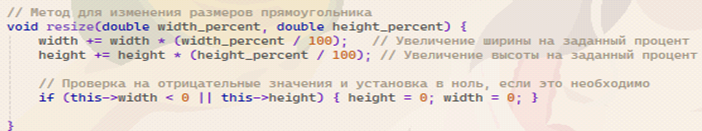


Рисунок 4 - Метод изменения сторон прямоугольника

Добавил метод для вывода информации о состоянии объекта в консоль (Рисунок 5)

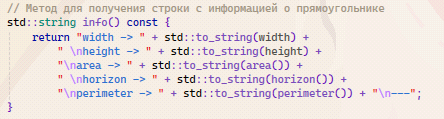


Рисунок 5 - Метод для вывода информации о состоянии объекта

Далее в main.cpp, пользователю необходимо ввести размер контейнера Цикл не завершится, пока пользователь не введёт размер контейнера корректно. (Рисунок 6)

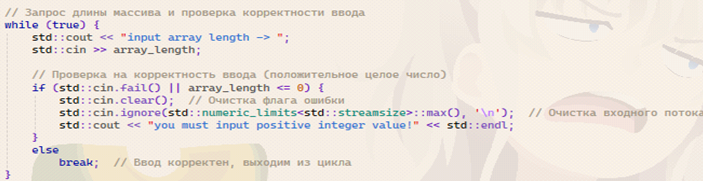


Рисунок 6 - Старт программы

После этого с помощью цикла for, осуществляется проверка корректности ввода, в случае успеха в качестве аргумента, инициализируется конструктор, в него передаются только что введенные поля, создается новые объект класса и с помощью метода push\_back(), сохраняется в контейнере (Рисунок 7).

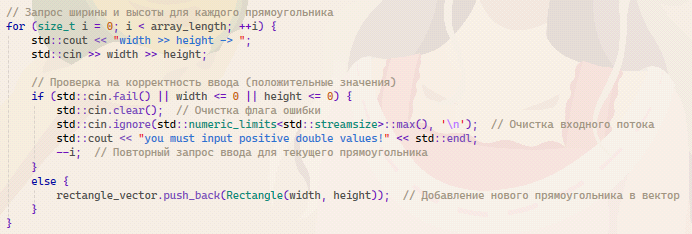


Рисунок 7 - Проверка корректности вводимых данных

В конце пользователь может изменить размер сторон любого прямоугольника на выбор, в случае если пользователь хочет выйти, он может ввести -1 (Рисунок 8).

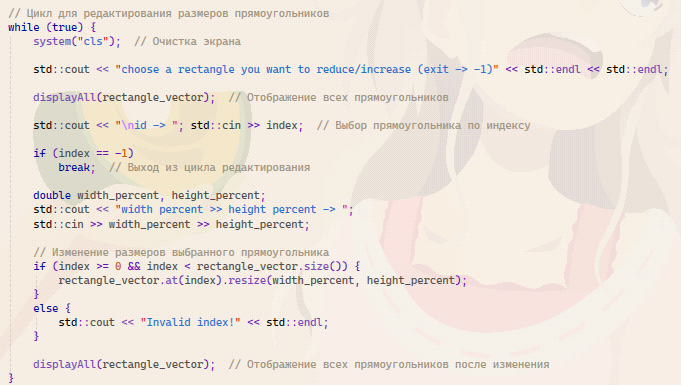


Рисунок 8 - Изменение сторон прямоугольника

Создал диаграмму классов (Рисунок 9).

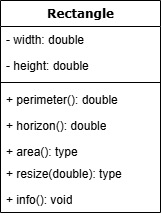


Рисунок 9 - Диаграмма классов

Индивидуальное задание:

Реализация классов Aircraft и SpaceShuttle (Рисунок 10).

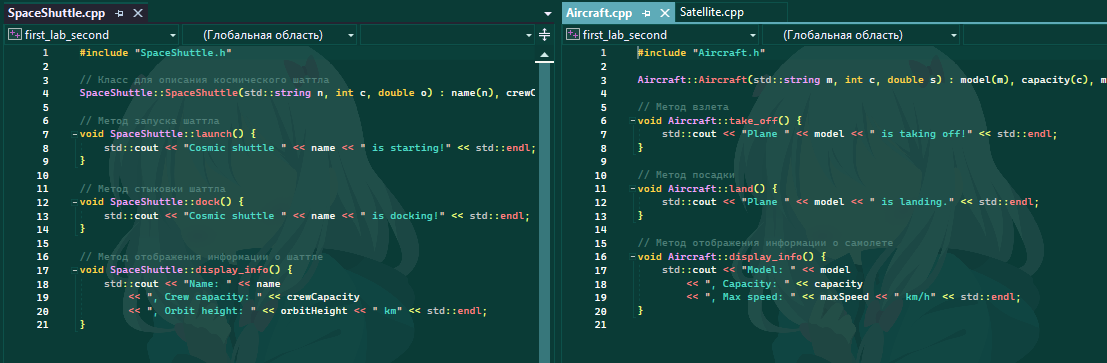


Рисунок 10 - Реализация классов Aircraft и SpaceShuttle

Реализация классов Helicopter, Satellite, Drone (Рисунок 11).

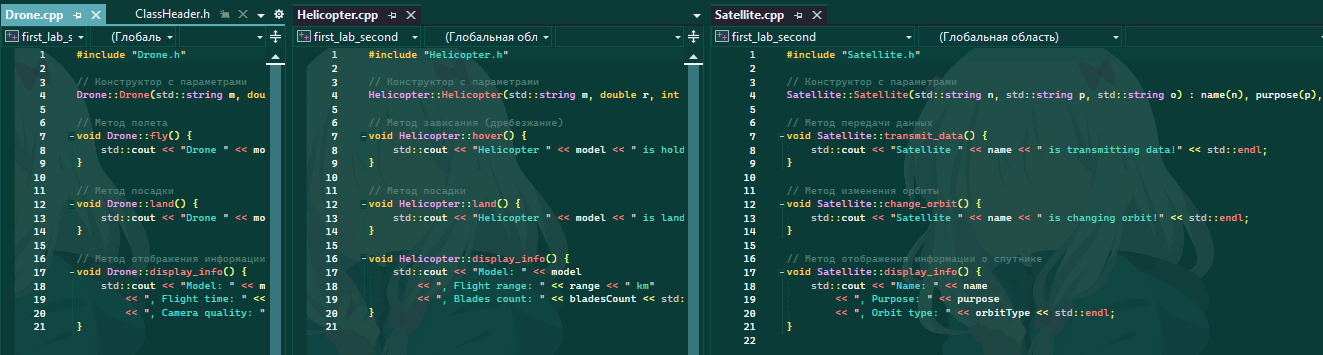


Рисунок 11 - Реализация классов Helicopter, Satellite, Drone

В функции main, использую std::vector, для хранения объектов, заполняю их и вывожу в консоль (Рисунок 12-13).

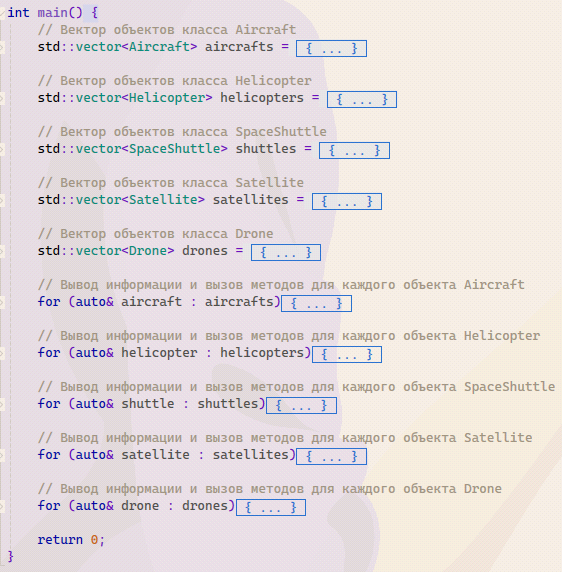


Рисунок 12 - Функция main

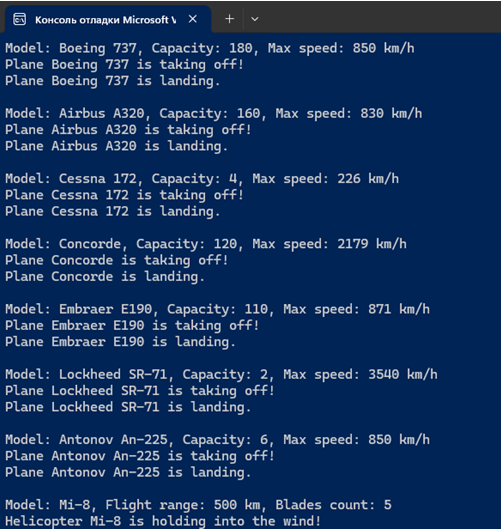


Рисунок 13 - Вывод в консоль

Создал диаграмму классов (Рисунок 14).

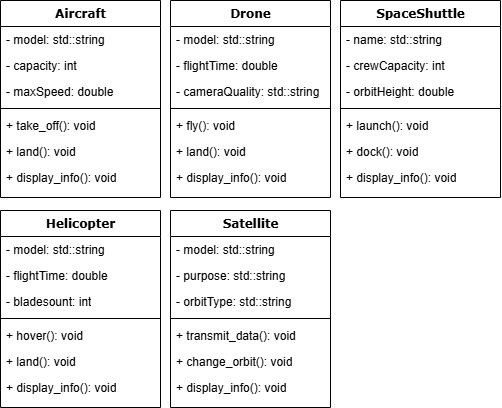


Рисунок 14 - Диаграмма классов

**Контрольные вопросы.**

1. **Основные принципы объектно-ориентированного программирования**

Инкапсуляция. Скрытие деталей реализации объекта и предоставление только интерфейса для взаимодействия с ним. Это позволяет изолировать изменения в одной части программы от других частей, что делает код более надёжным и устойчивым к изменениям.

Наследование. Позволяет создавать новые классы на основе существующих. Это способствует повторному использованию кода и созданию иерархий классов. Наследование позволяет наследникам использовать свойства и методы предков и переопределять или расширять их, если это необходимо.

Полиморфизм. Позволяет объектам разных классов обладать общим интерфейсом. Это позволяет обрабатывать объекты разных типов с помощью общих методов и функций. Полиморфизм делает код более гибким и расширяемым.

1. **Что такое класс?**

Класс — это шаблон или чертеж для создания объектов. Он определяет атрибуты и методы, которые будут у всех объектов того же класса.

1. **Что такое диаграмма классов?**

Диаграмма классов — это графическое представление структуры системы, показывающее классы, их атрибуты и методы, а также отношения между ними (Рисунок 14).

1. **Что такое атрибут?**

Атрибут (или свойство) — это характеристика или состояние объекта. Он представляет данные, которые принадлежат классу и его объектам [5](https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP).