Пермский национальный исследовательский политехнический университет.

Лабораторная работа № 5 по ООП.

«Наследование. Виртуальные функции. Полиморфизм».

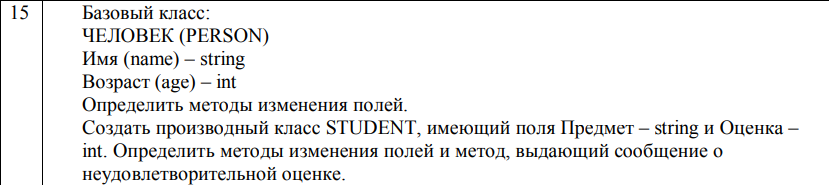
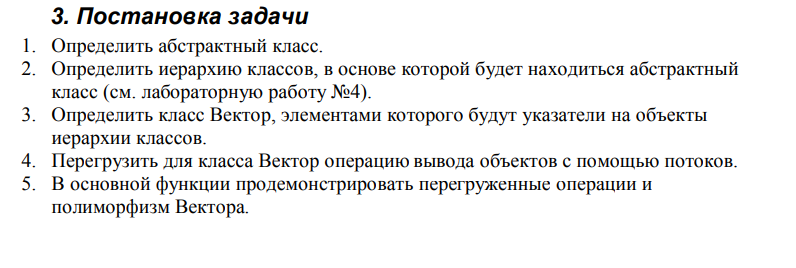
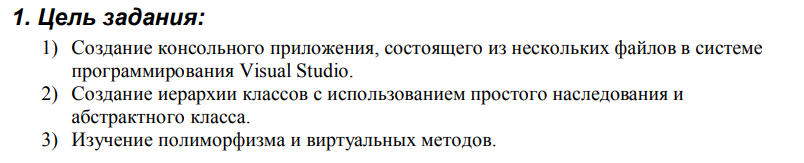
Выполнил: студент группы РИС-23-2б

Вековшинин Иван Николаевич

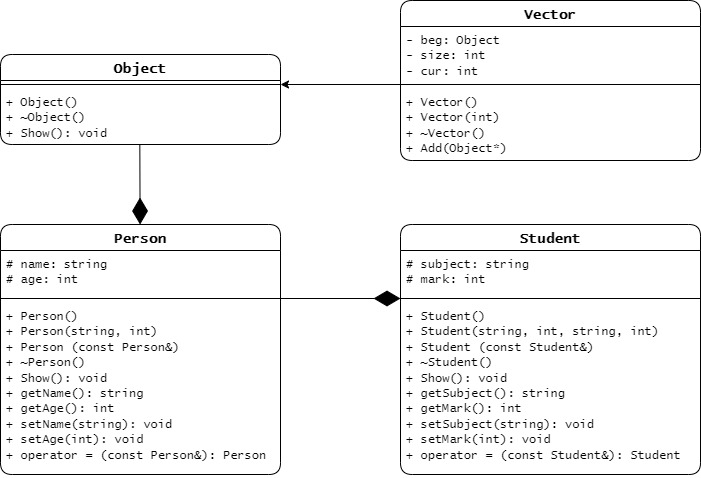
Проверила: доцент кафедры ИТАС

О.А. Полякова.

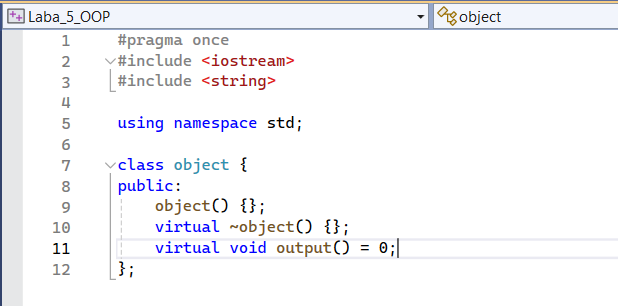
2024 г.



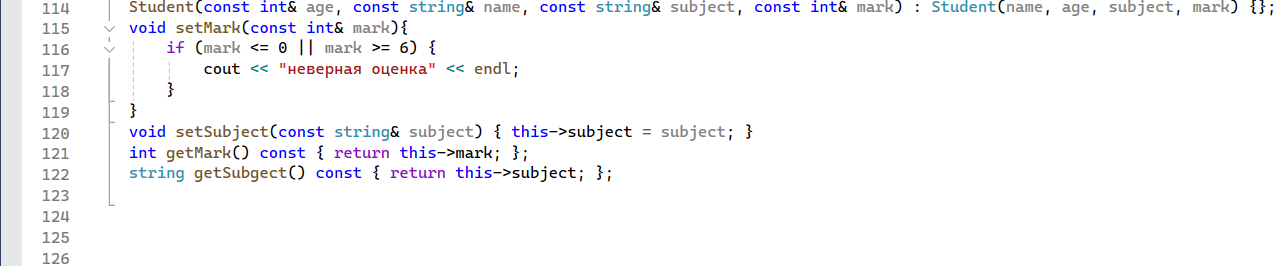
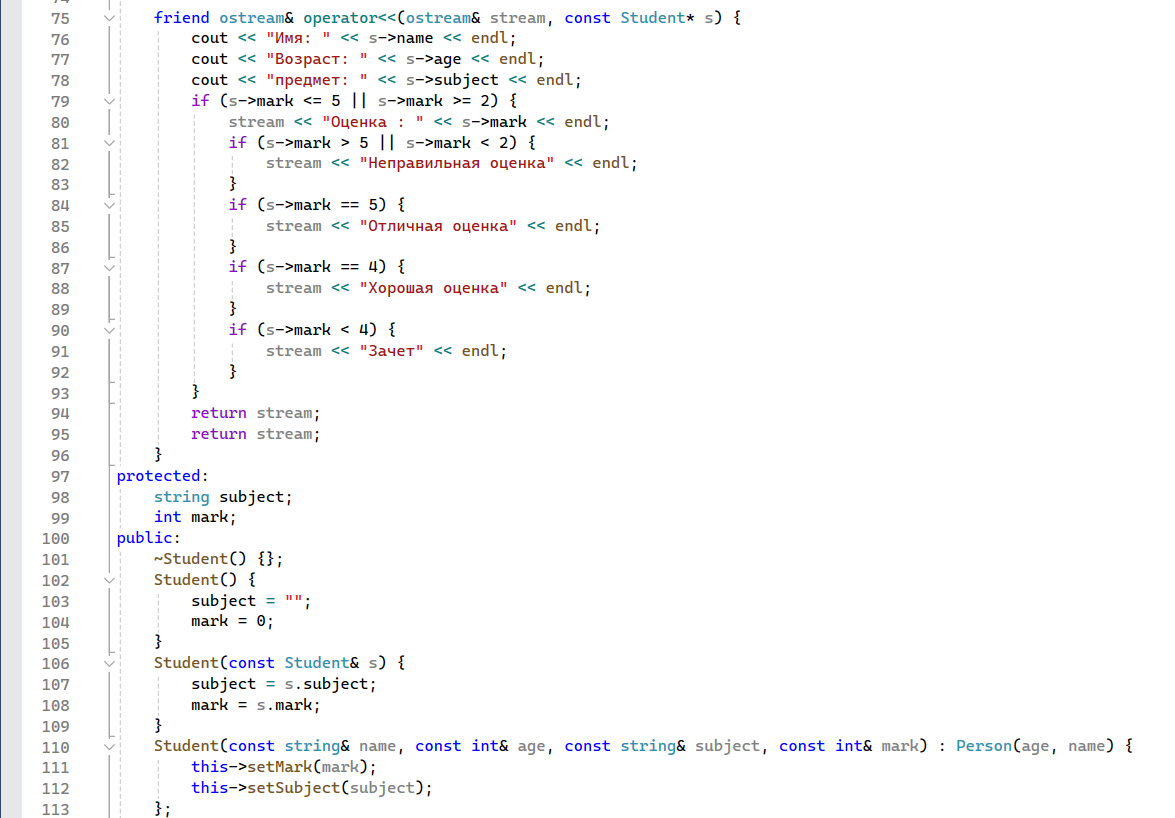
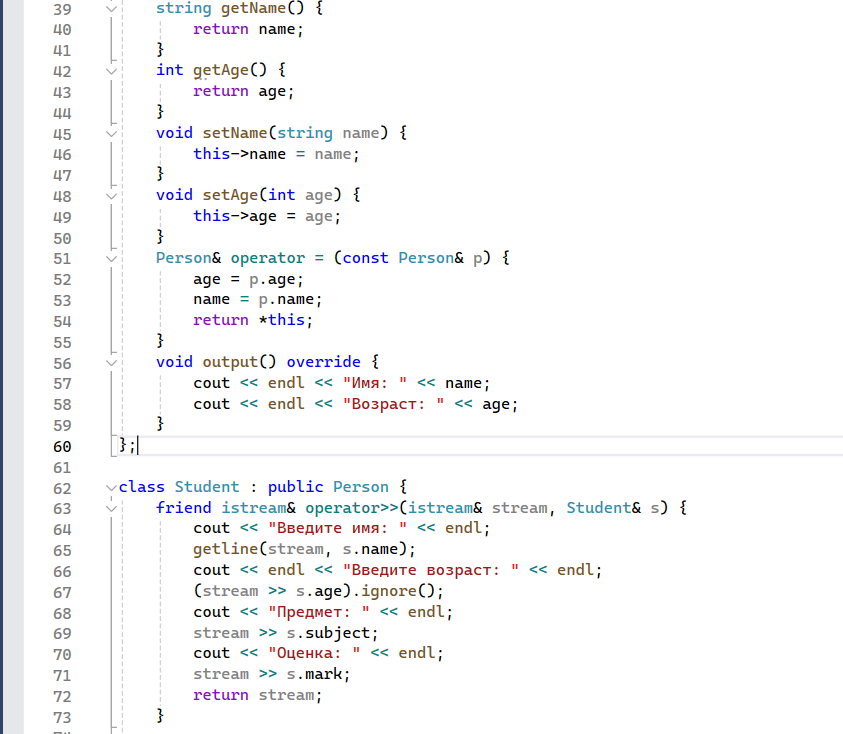
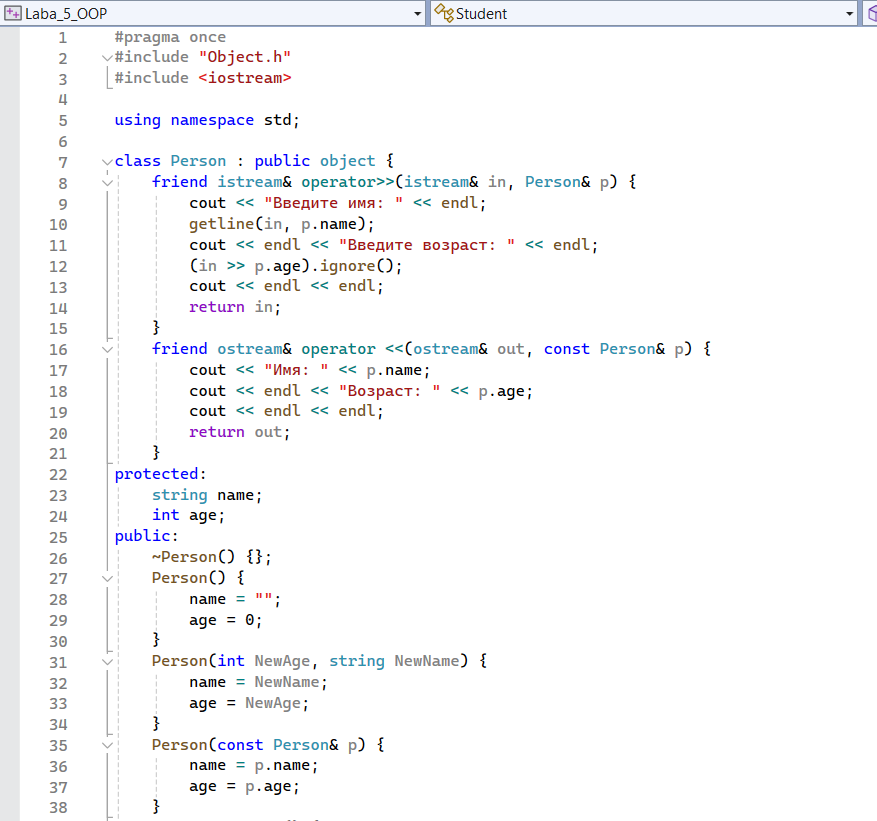
UML Diagram



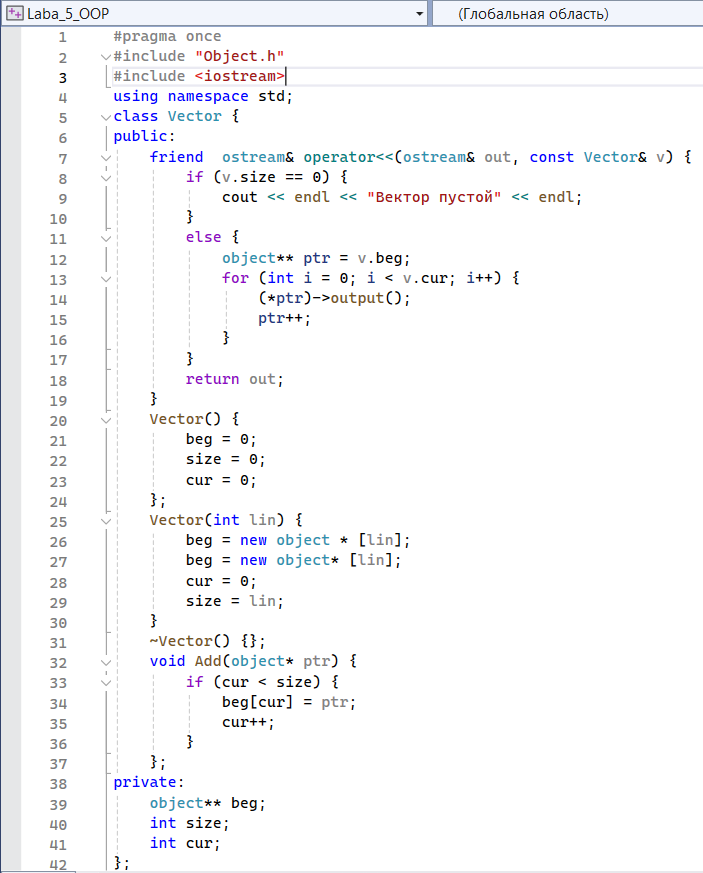
Object.h

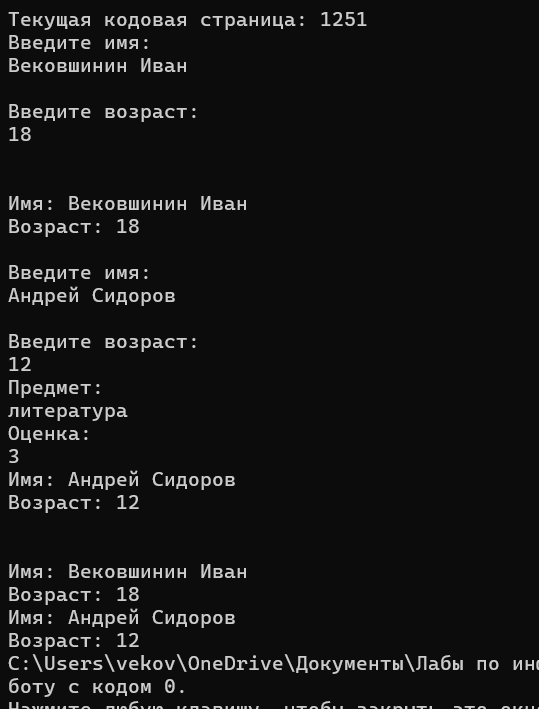


Person.h

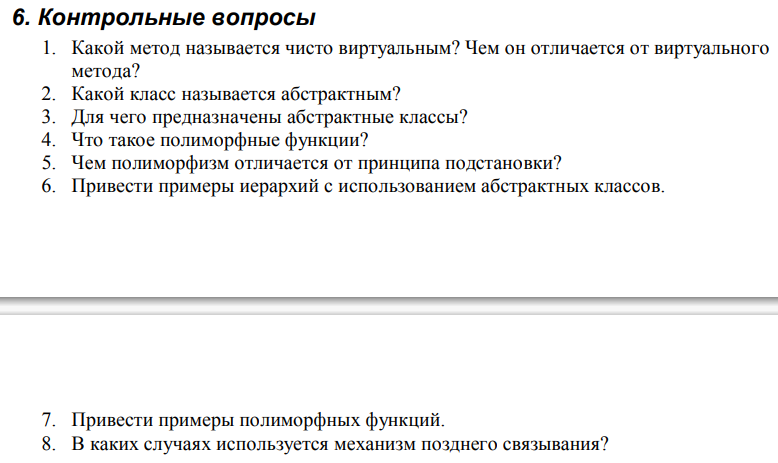


Vector.h





Ответы на вопросы:



1. \*\*Чисто виртуальный метод\*\* (pure virtual method) - это метод в базовом классе, который не имеет реализации и должен быть переопределен в производных классах. Он объявляется с помощью ключевого слова virtual и знака = 0;. Отличие от обычного виртуального метода в том, что чисто виртуальный метод обязывает производные классы предоставить свою реализацию этого метода.

2. \*\*Абстрактный класс\*\* - это класс, который содержит хотя бы один чисто виртуальный метод. Такой класс нельзя создать как объект, только использовать для наследования.

3. \*\*Абстрактные классы\*\* предназначены для создания общего интерфейса для производных классов. Они позволяют определить общие методы и поля, которые должны быть реализованы в конкретных производных классах.

4. \*\*Полиморфные функции\*\* - это функции, которые могут принимать объекты различных типов и вызывать соответствующие методы в зависимости от типа объекта. Полиморфизм позволяет использовать одинаковый интерфейс для объектов разных классов.

5. \*\*Полиморфизм\*\* - это способность объектов различных классов отвечать на одинаковые сообщения. Принцип подстановки (Liskov Substitution Principle) гласит, что объекты производных классов могут быть подставлены вместо объектов базового класса без изменения корректности программы.

6. Примеры иерархий с использованием абстрактных классов:

- Shape (абстрактный класс) -> Circle, Rectangle, Triangle (производные классы)

- Animal (абстрактный класс) -> Cat, Dog, Bird (производные классы)

7.

#include <iostream>

class Shape {

public:

virtual void draw() const = 0;

};

class Circle : public Shape {

public:

void draw() const override {

std::cout << "Drawing a circle." << std::endl;

}

};

class Rectangle : public Shape {

public:

void draw() const override {

std::cout << "Drawing a rectangle." << std::endl;

}

};

void drawShape(const Shape& shape) {

shape.draw();

}

int main() {

Circle circle;

Rectangle rectangle;

drawShape(circle); // Вызовется метод draw() для Circle

drawShape(rectangle); // Вызовется метод draw() для Rectangle

return 0;

}

В данном примере функция drawShape принимает объекты типа Shape (абстрактный класс), что позволяет передавать как объекты класса Circle, так и объекты класса Rectangle, вызывая соответствующие методы draw() для каждого конкретного класса.

8. Механизм позднего связывания (dynamic binding) используется в случаях, когда необходимо определить, какой метод вызывать во время выполнения программы, основываясь на типе объекта. Позднее связывание обеспечивает полиморфизм и позволяет использовать виртуальные функции для обработки объектов различных классов через указатели или ссылки на базовый класс.

Механизм позднего связывания используется в случаях, когда необходимо обеспечить гибкость и расширяемость программы за счет возможности переопределения методов в производных классах и вызова этих методов через указатели или ссылки на базовый класс.