ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №5

Наследование. Виртуальные функции. Полиморфизм

Вариант №14

Выполнил студент группы РИС-23-3Б

Блинов А. Е.

Проверила доцент кафедры ИТАС

О. А. Полякова

2024 г.

**Постановка задачи**

1. Определить абстрактный класс.

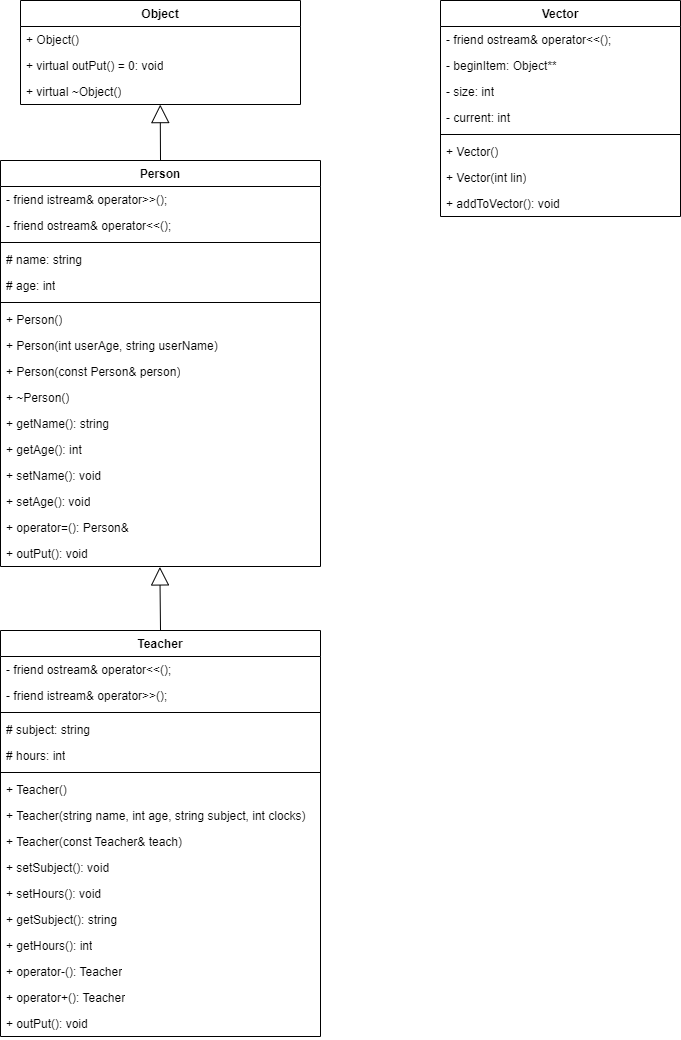
2. Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).

3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.

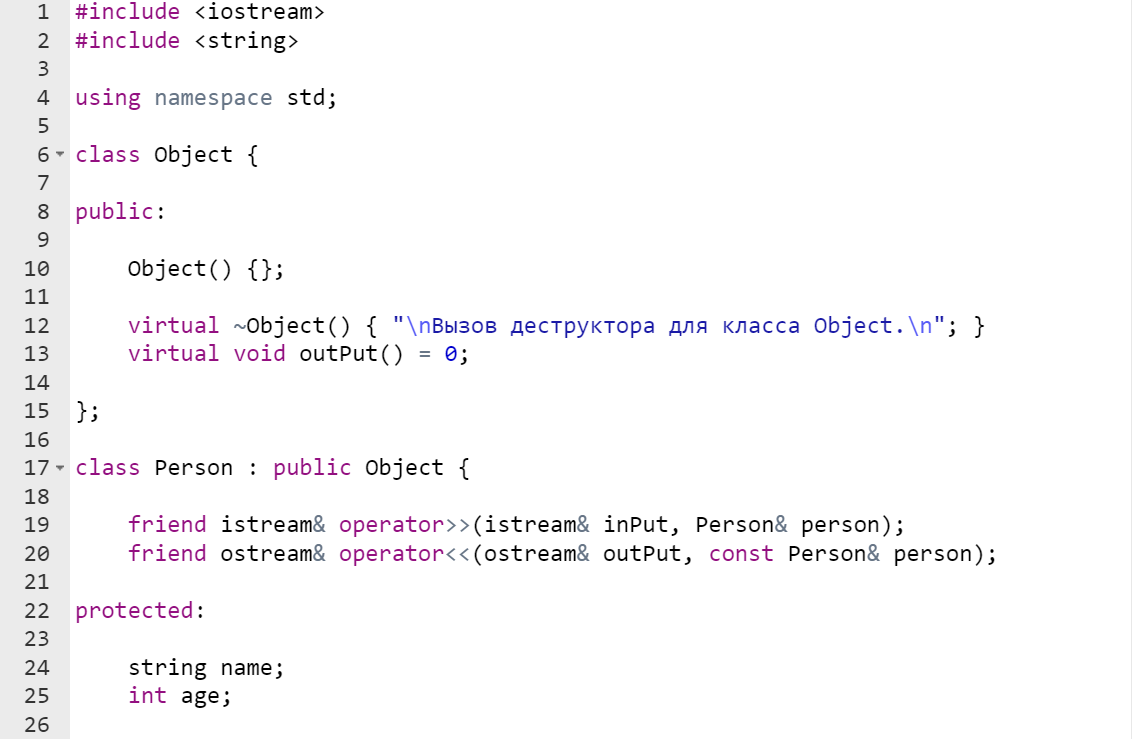
4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков. 5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

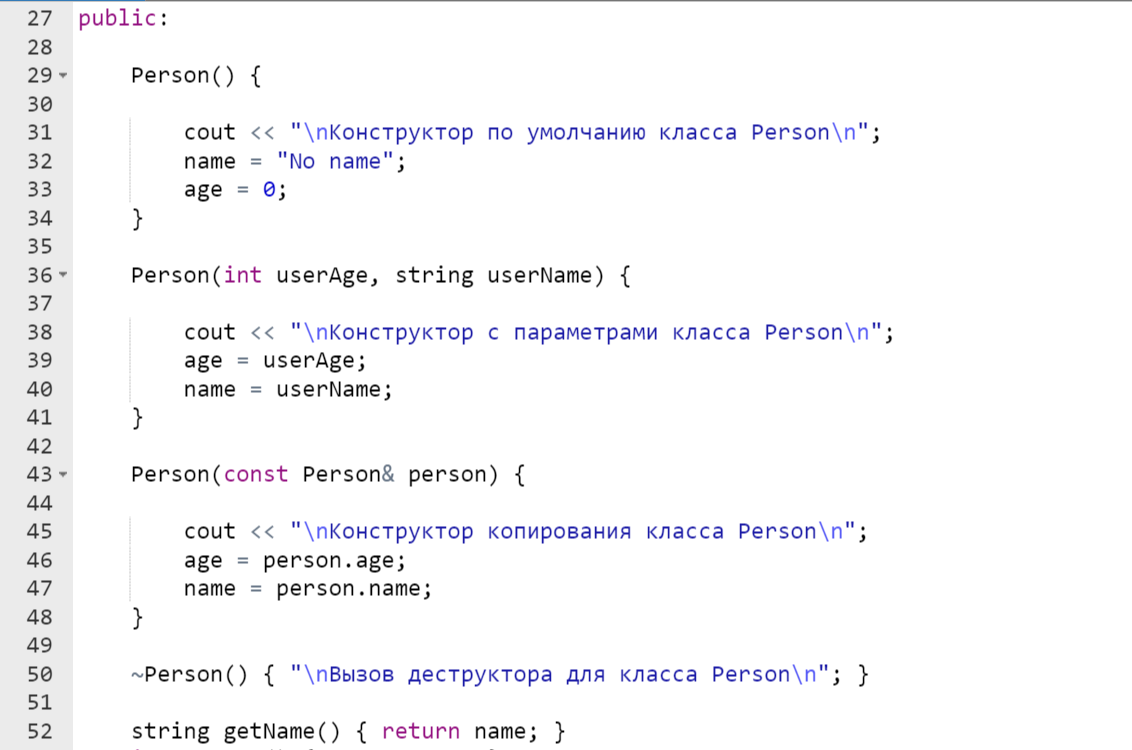
Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int; Определить методы изменения полей. Создать производный класс TEACHER, имеющий поля Предмет – string и Количество часов – int. Определить методы изменения полей, а также увеличения и уменьшения часов.

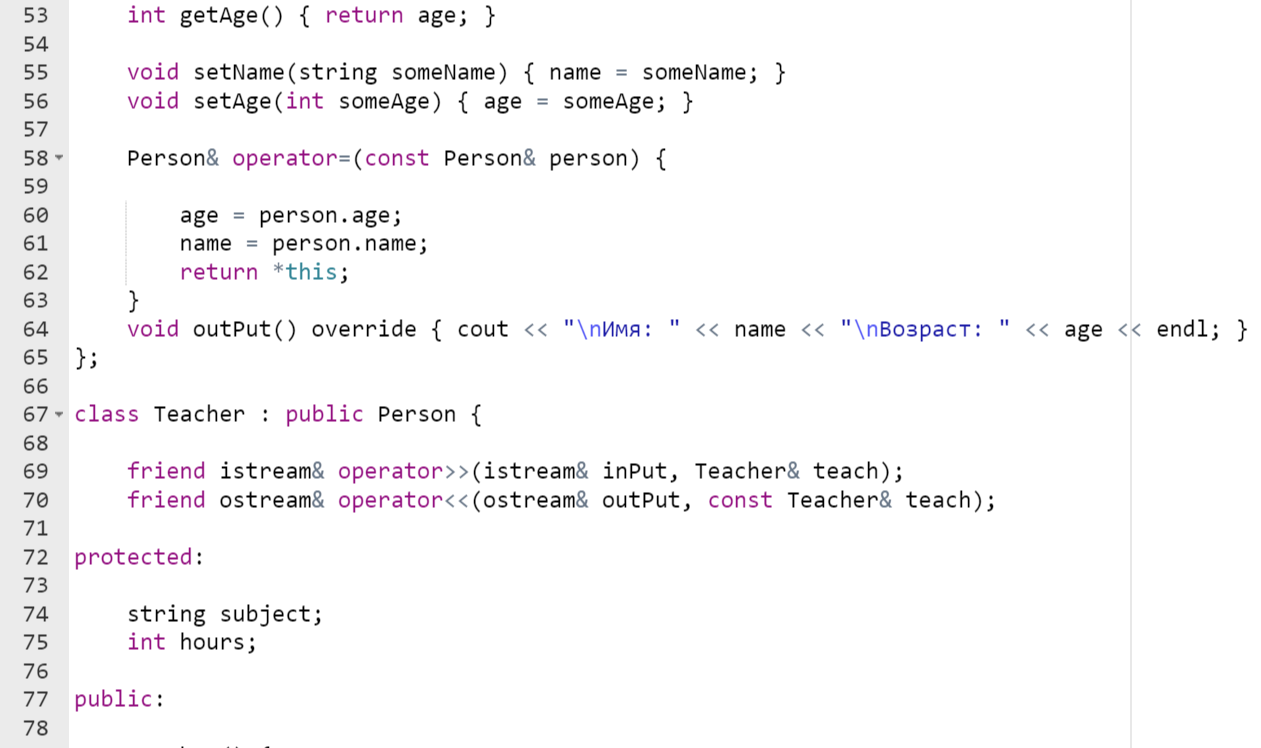
**UML диаграмма**

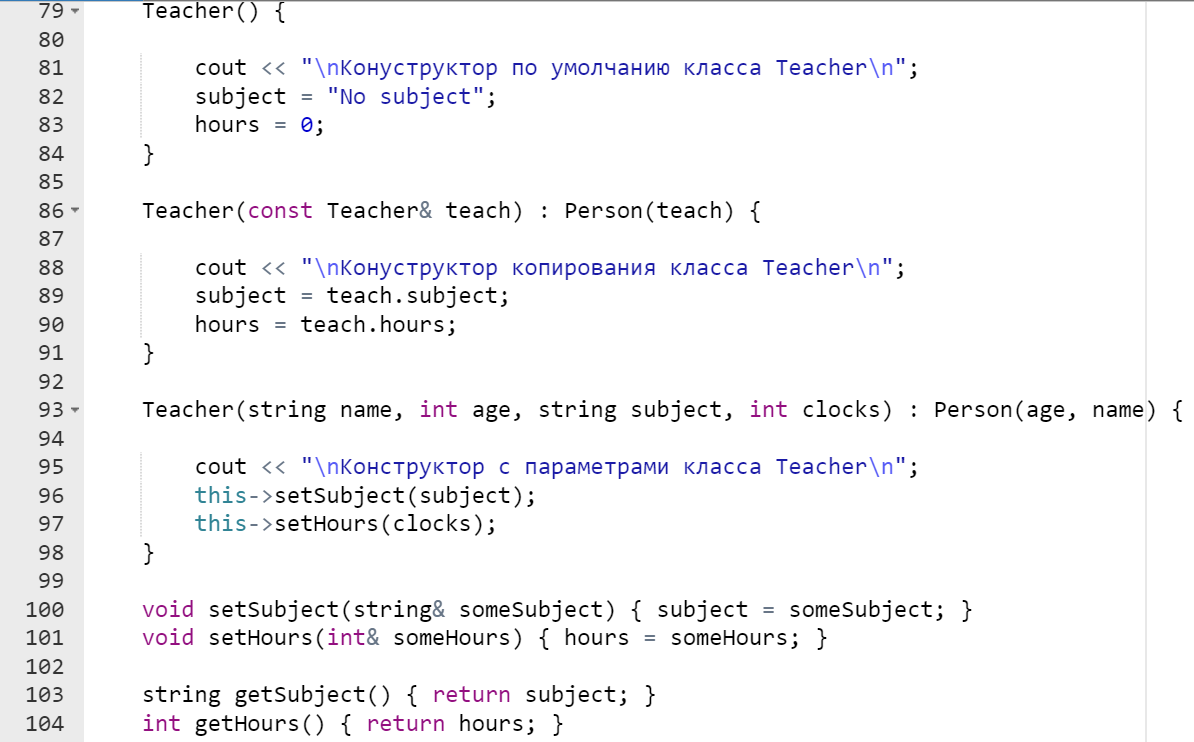


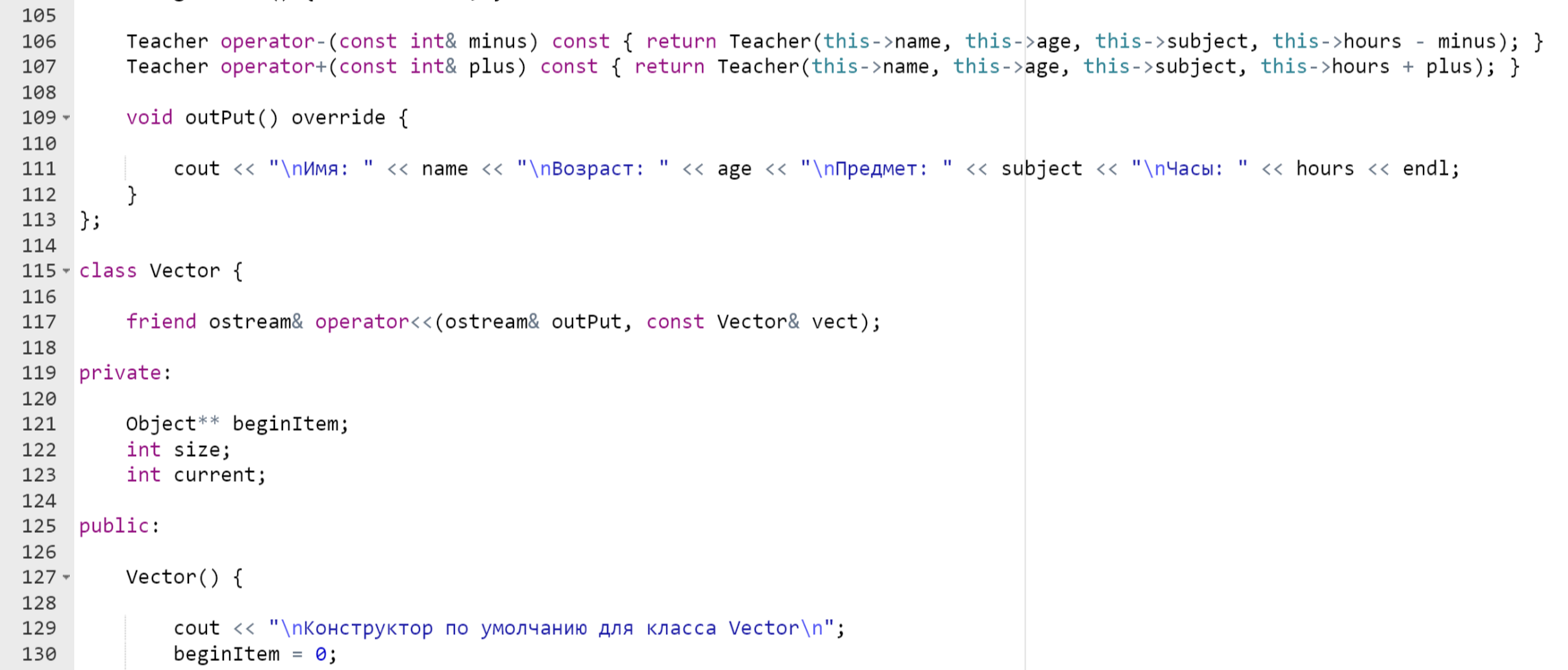
**Код на языке C++**

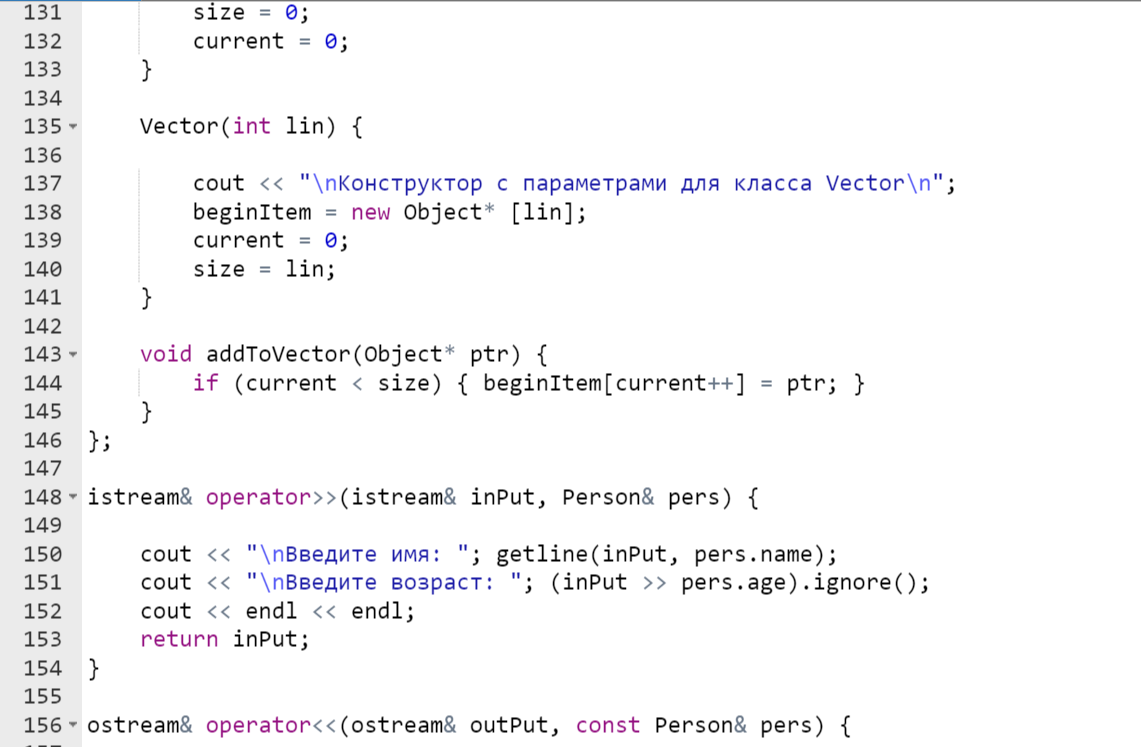
****

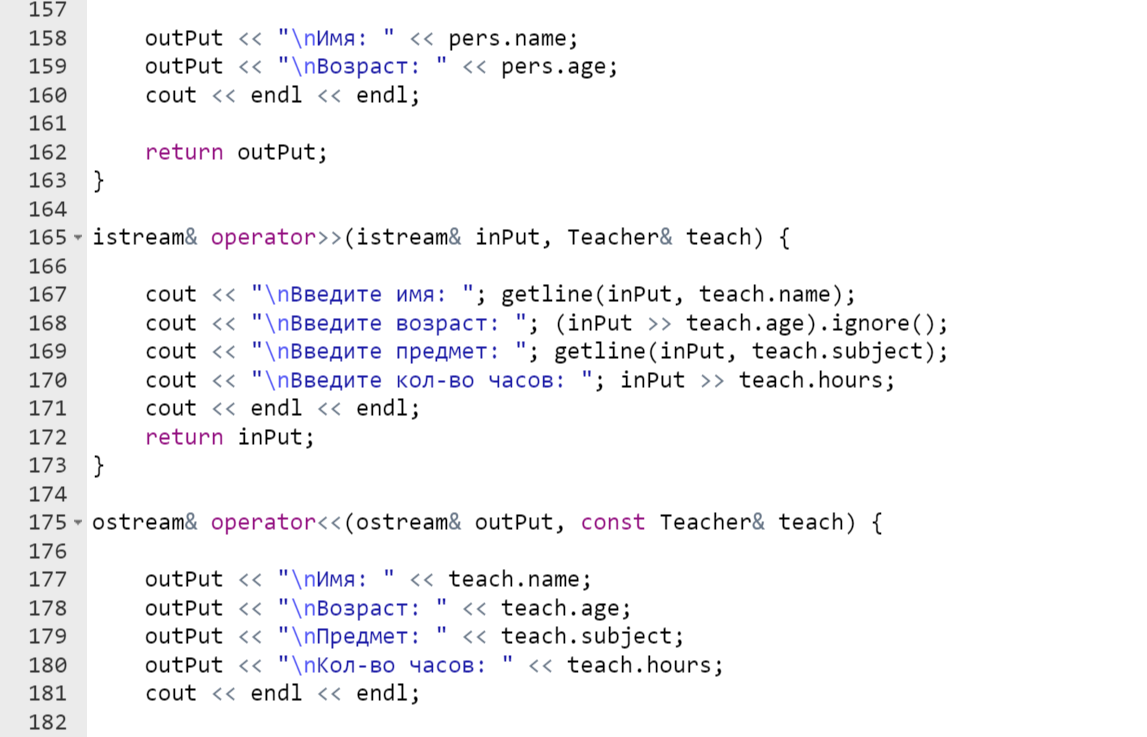
****

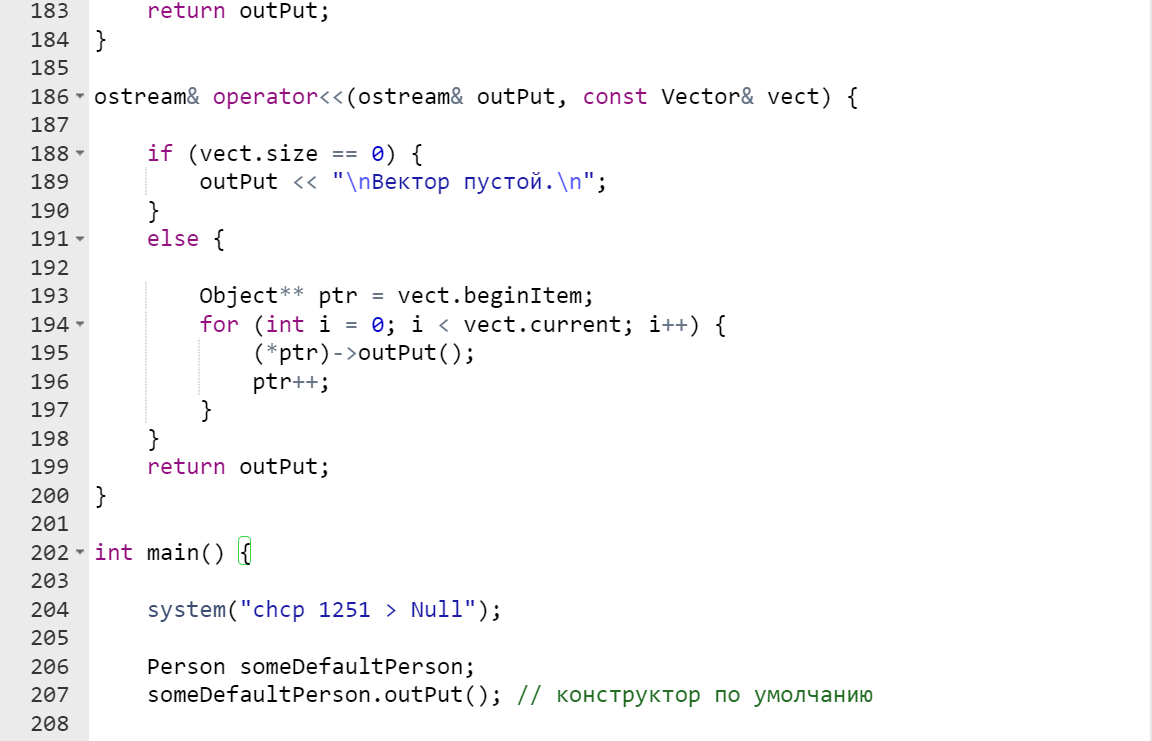
****

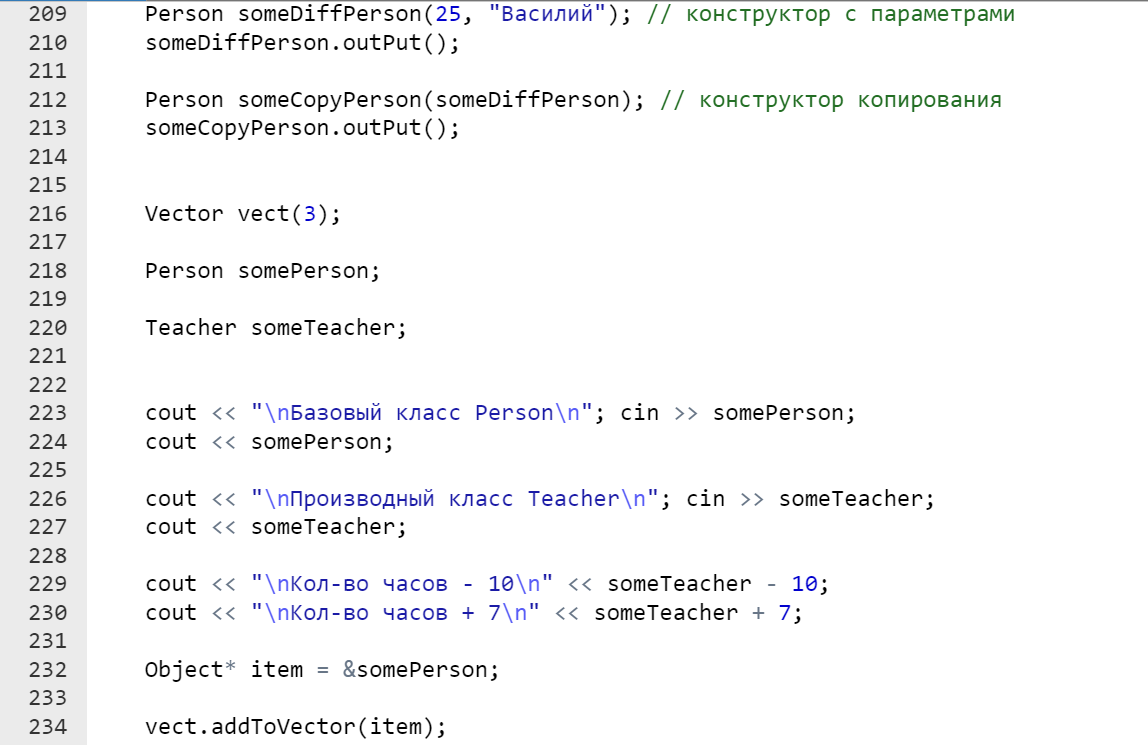


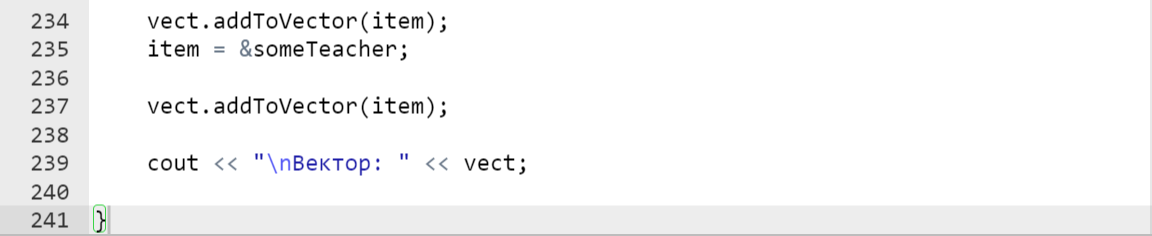




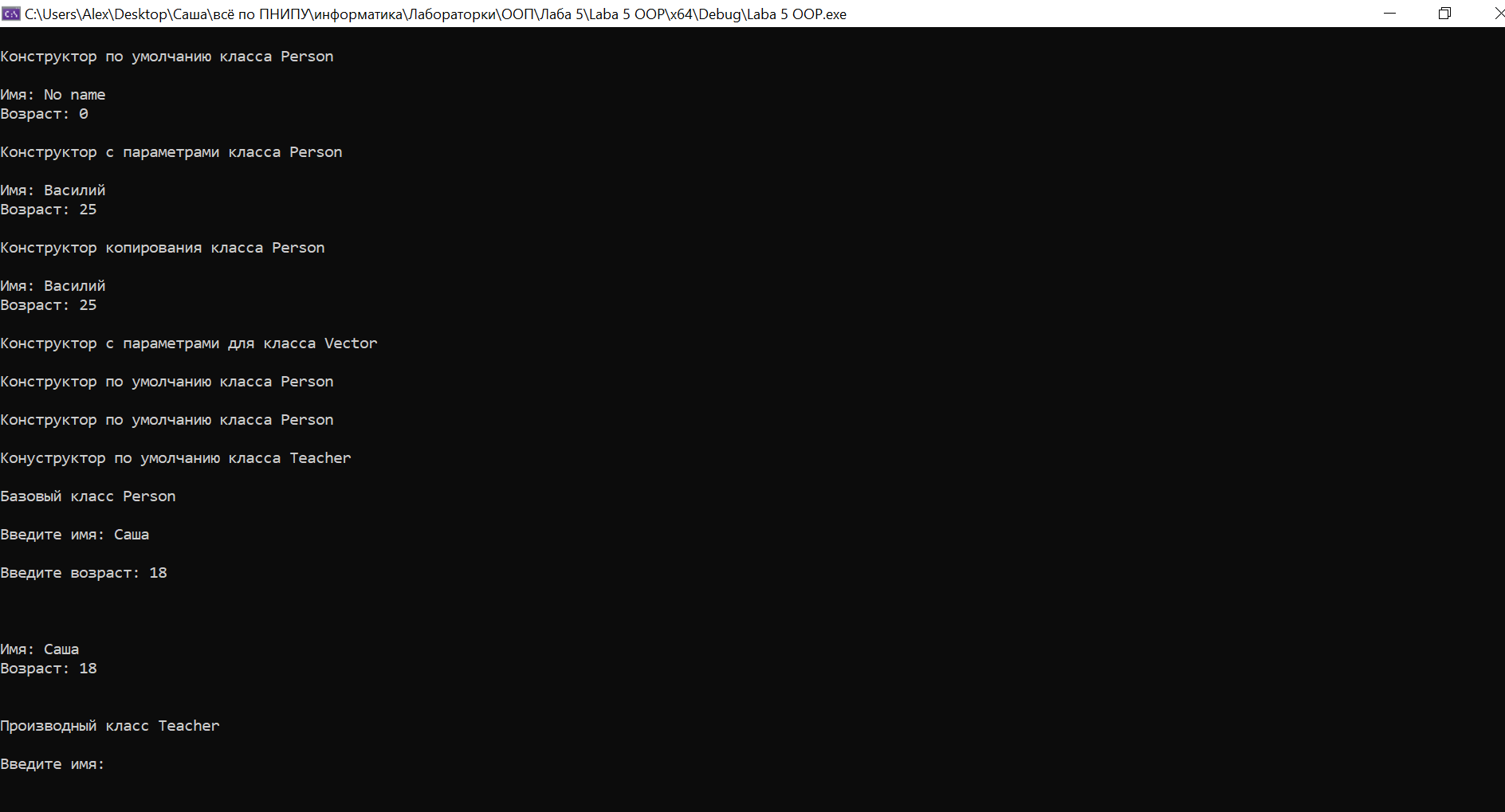








**Результаты работы**





**Выводы**

Программа успешно работает.

GitHub: <https://github.com/alexblin123/Laba-5-OOP.git>

**Контрольные вопросы**

**1. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?**

Чистые виртуальные функции - функции, которые не имеют определения.

Обычная виртуальная функция превращается в чистую, если в конец объявления виртуальной функции добавить "= 0"

Основное различие между виртуальной функцией и чисто виртуальной функцией является то, что виртуальная функция является функцией в базовом классе, который объявлен с использованием виртуального ключевого слова в то время как чистая виртуальная функция является виртуальной функцией в базовом классе без определения функции.

**2. Какой класс называется абстрактным?**

АБСТРАКТНЫЕ КЛАСС - КЛАСС, КОТОРЫЙ СОДЕРЖИТ ИЛИ НАСЛЕДУЕТ ХОТЯ БЫ ОДНУ ЧИСТУЮ ВИРТУАЛЬНУЮ ФУНКЦИЮ БЕЗ ПЕРЕОПРЕДЕЛЕНИЯ.

**3. Для чего предназначены абстрактные классы?**

Абстрактные классы используются в качестве обобщенных концепций, на основе которых можно создавать более конкретные производные классы.

Изображение выглядит как текст

Автоматически созданное описание

**4. Что такое полиморфные функции?**

Полиморфные функции - функции, способные обрабатывать данные разных типов.

**5. Чем полиморфизм отличается от принципа подстановки?**

-

**6. Привести примеры иерархий с использованием абстрактных классов.**

#include <iostream>

using namespace std;

// Абстрактный класс Figure

class Figure {

public:

virtual void draw() const = 0; // Чисто виртуальная функция

virtual double area() const = 0;

};

// Класс Circle, производный от Figure

class Circle : public Figure {

private:

double radius;

public:

Circle(double r) : radius(r) {}

void draw() const override {

cout << "Рисуем круг\n";

}

double area() const override {

return 3.14159 \* radius \* radius;

}

};

// Класс Rectangle, производный от Figure

class Rectangle : public Figure {

private:

double width;

double height;

public:

Rectangle(double w, double h) : width(w), height(h) {}

void draw() const override {

cout << "Рисуем прямоугольник\n";

}

double area() const override {

return width \* height;

}

};

int main() {

Circle circle(5.0);

Rectangle rectangle(4.0, 6.0);

Figure\* figure1 = &circle;

Figure\* figure2 = &rectangle;

figure1->draw();

cout << "Площадь круга: " << figure1->area() << endl;

figure2->draw();

cout << "Площадь прямоугольника: " << figure2->area() << endl;

return 0;

}

**7. Привести примеры полиморфных функций.**

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

**8. В каких случаях используется механизм позднего связывания?**

Позднее связывание означает, что объект связывается с вызовом функции только во время ис­полнения программы, а не раньше. Позднее связывание достигается в С++ с помощью использо­вания виртуальных функций и производных классов. Его достоинством является высокая гиб­кость. Оно может использоваться для поддержки общего интерфейса, позволяя при этом различным объектам иметь свою собственную реализацию этого интерфейса. Более того, оно помогает со­здавать библиотеки классов, допускающие повторное использование и расширение. Использование позднего связывания оправдано только тогда, когда оно улучша­ет структурированность и управляемость программы. Надо иметь в виду, что проигрыш в произ­водительности невелик, поэтому, когда ситуация требует позднего связывания, можно использовать его без всякого сомнения.