Пермский национальный исследовательский политехнический университет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа 4

**Простое наследование. Принцип подстановки.**

Выполнил

Студент гр. РИС-24-3б Рунтов Иван Андреевич

Проверил

Доцент кафедры ИТАС Ольга Андреевна Полякова

г. Пермь, 2025

**Постановка задачи**

1. Определить пользовательский класс.

2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3. Определить в классе деструктор.

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

5. Перегрузить операцию присваивания.

6. Перегрузить операции ввода и вывода объектов с помощью потоков.

7. Определить производный класс.

8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

**Вариант 15**

Базовый класс: ЧЕЛОВЕК (PERSON)

Имя (name) – string

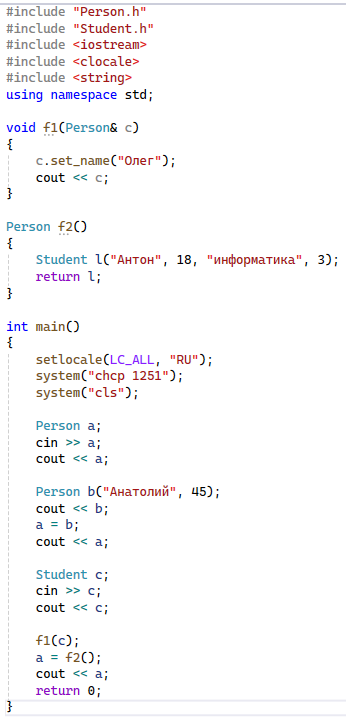
Возраст (age) – int

Определить методы изменения полей. Создать производный класс STUDENT, имеющий поля

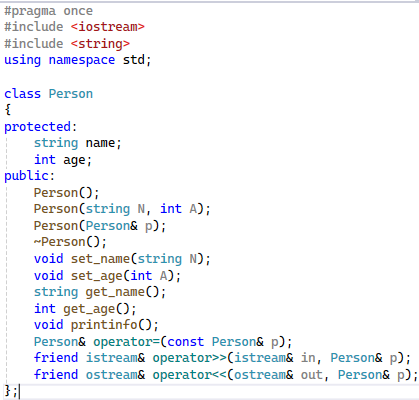
Предмет – string и Оценка – int.

Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

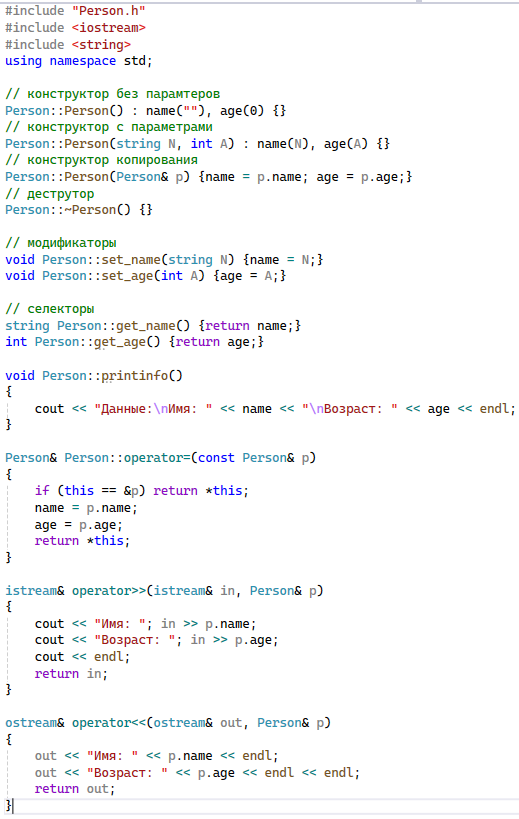
**Исполняемый файл main**



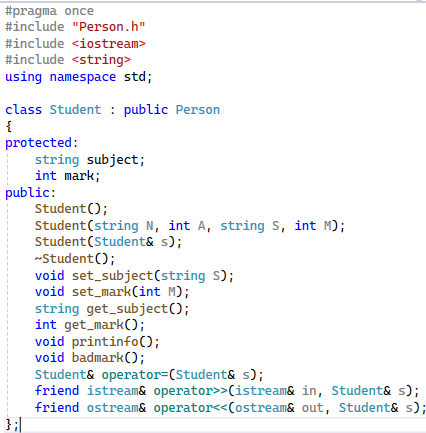
**Определение класса Person**



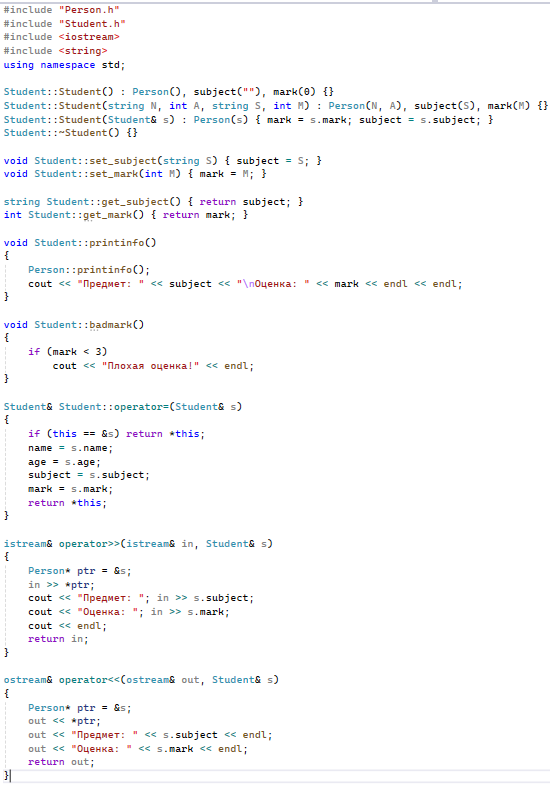
**Описание методов класса Person**



**Определение класса Student**



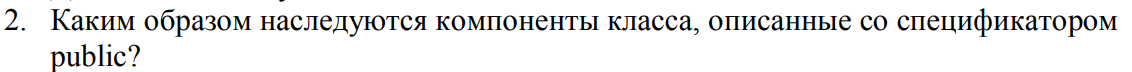
**Описание методов класса Student**



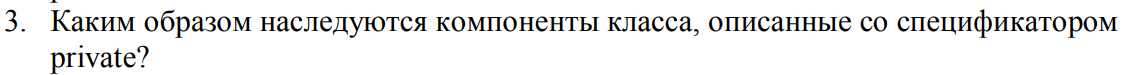
**Ответы на контрольные вопросы**



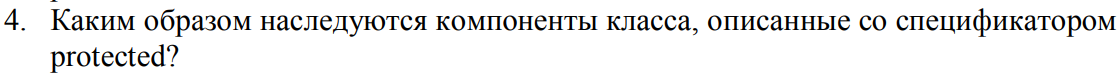
Механизм наследования используется для получения классов, построенных на основе уже существующих классов, при этом они могут быть дополнены или изменены.



При наследовании со спецификатором public все компоненты базового класса сохраняют свой уровень доступа. Private – недоступны, protected – доступны с спецификатором protected, public - доступны с спецификатором public.



Компоненты спецификатора private недоступны в производном классе при публичном наследовании.



Компоненты спецификатора protected при публичном наследовании сохраняют сохраняют protected.



class имя\_производного\_класса : спецификатор\_наследования имя\_базового



Конструкторы не наследуются



Деструкторы не наследуются



Сначала базовый, потом компоненты-объекты, а потом сам производный класс.



Сначала производный, потом компоненты-объекты, а потом базовый класс.



Механизм виртуальных функций и позднего динамического класса применяются в тех случаях, когда в базовом и в каждом производном классе необходимы разные реализации какой-то функции.



Конструкторы не могут быть виртуальными, а деструкторы могут быть виртуальными. Класс, имеющий виртуальные функции должен иметь виртуальный деструктор.



Спецификатор virtual также наследуется в производных классах неявным образом.



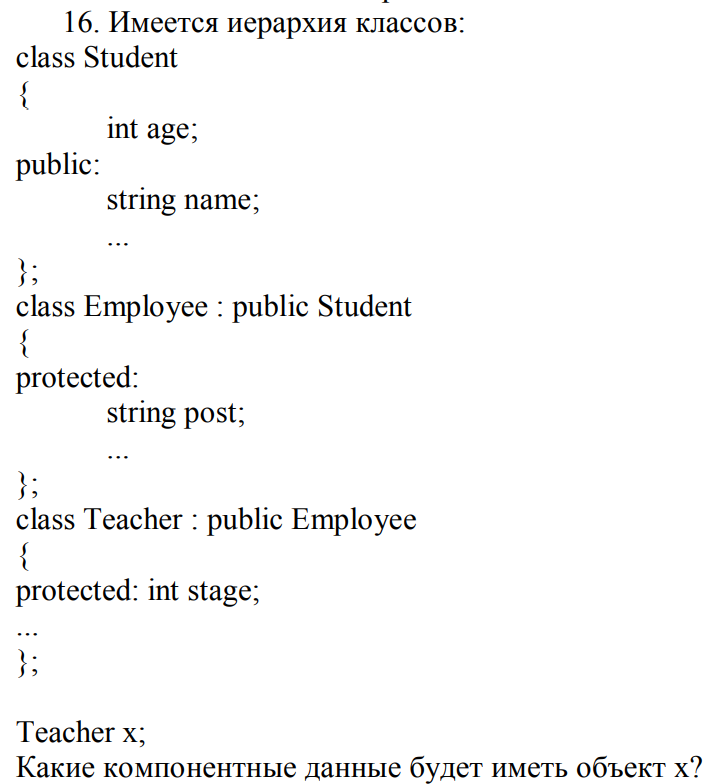
Открытое наследование устанавливает отношение «является» : производный является частью базового.



Закрытое наследование устанавливает отношение «реализация». Принцип подстановки не соблюдается.



Принцип подстановки заключается в том, что вместо объектов базового класса можно подставить объекты производных классов.



Объект x будет иметь данные name, post, stage.



Student()

{

age = 0;

name = “”;

}

Employee()

{  
 name = “”;

post = “”;

}

Teacher()

{

name = “”;

post = “”;

stage = 0;

}



Student(int A, string N)

{

age = A;

name = N;

}

Employee(string P, string N)

{  
 name = N;

post = P;

}

Teacher(string P, string N, int S)

{

name = N;

post = P;

stage = S;

}



Student(Student& s)

{

age = s.age;

name = s.name;

}

Employee(Employee& e)

{  
 name = e.name;

post = e.post;

}

Teacher(Teacher& t)

{

name = t.name;

post = t.post;

stage = t.stage;

}



Student& operator=(Student& s)

{

if (this == &s) return \*this;

age = s.age;

name = s.name;

return \*this;

}

Employee& operator=(Employee& e)

{

if (this == &e) return \*this;

name = e.name;

post = e.post;

return \*this;

}

Teacher& operator=(Teacher& t)

{

if (this == &t) return \*this;

name = t.name;

post = t.post;

stage = t.stage;

return \*this;

}

Ссылка на github: <https://github.com/RuntovIvan/Informatika>