Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

Лабораторная работа №4.

" Простое наследование. Принцип подстановки. "

15 вариант

Выполнил студент гр. РИС-24-2б

Иванова Елена Олеговна

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

Цель:

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Создание иерархии классов с использованием простого наследования.
3. Изучение принципа подстановки.

Задача:

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.

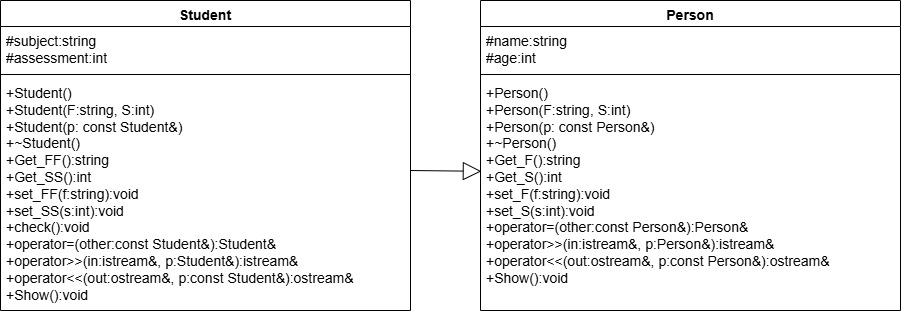
Постановка задачи

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.

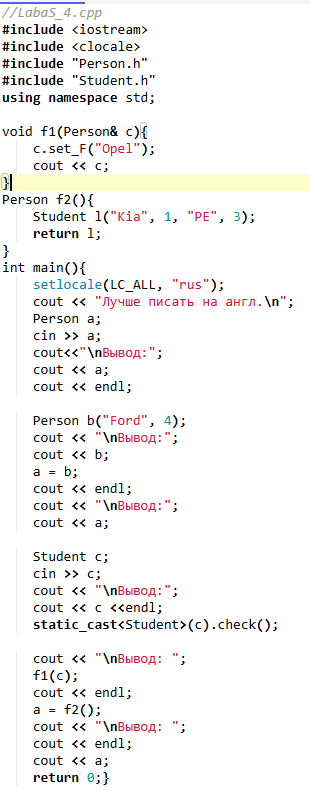
Анализ задачи.

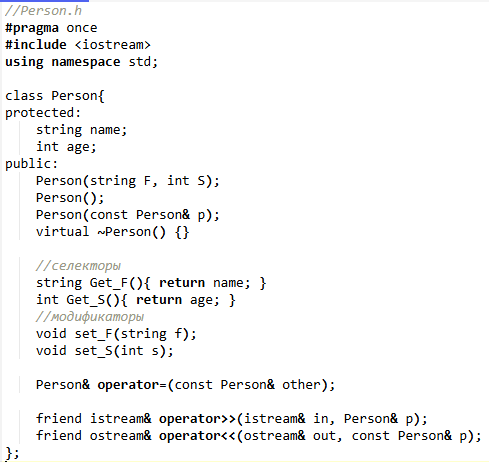
1. Определить пользовательский класс Person c 3 конструкторами, деструктором, селекторами и модификаторами. Перегрузить операции присваивания и ввода-вывода.
2. Определить производный класс Student на основе Person.
3. Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

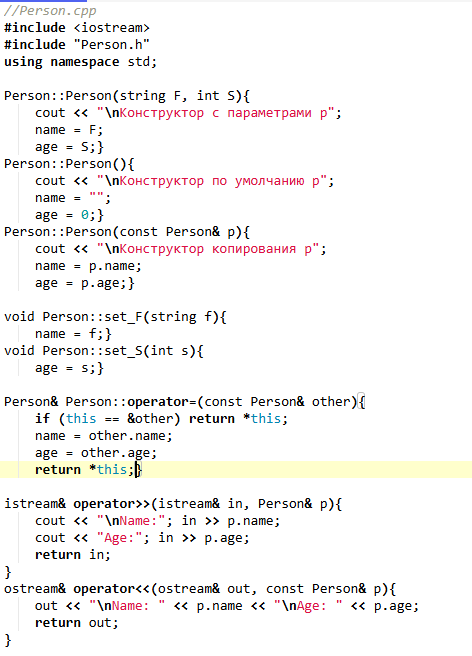
Диаграмма класса.

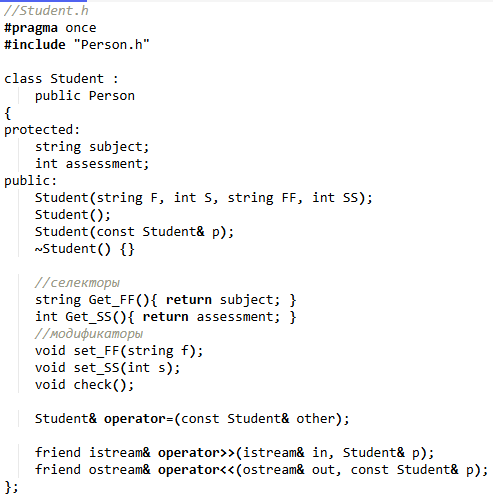


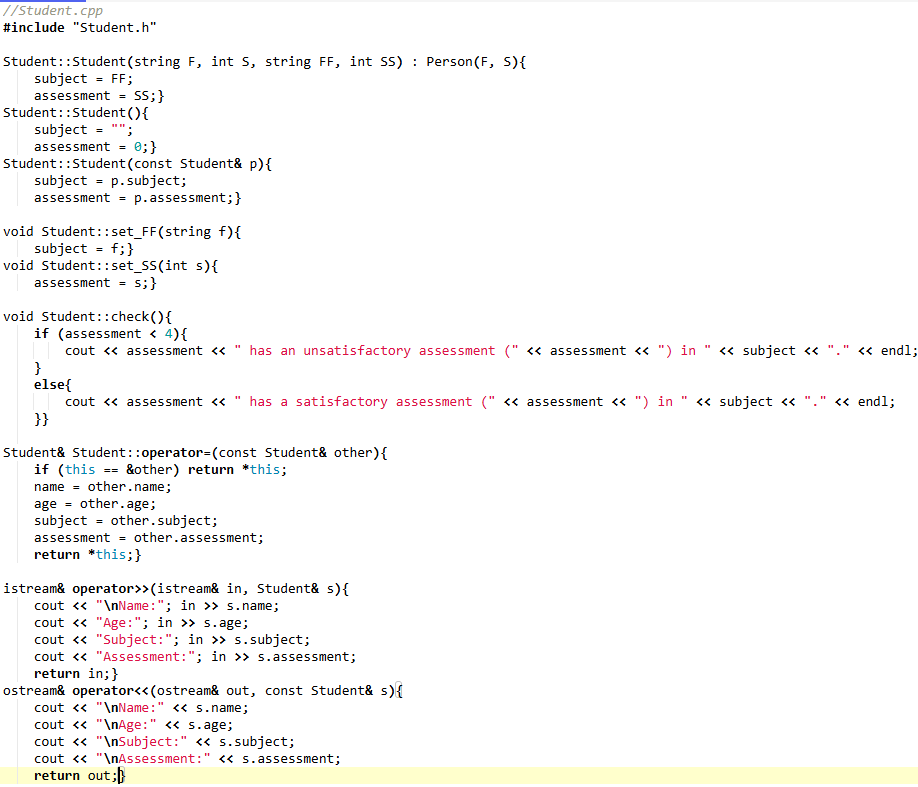
Код.



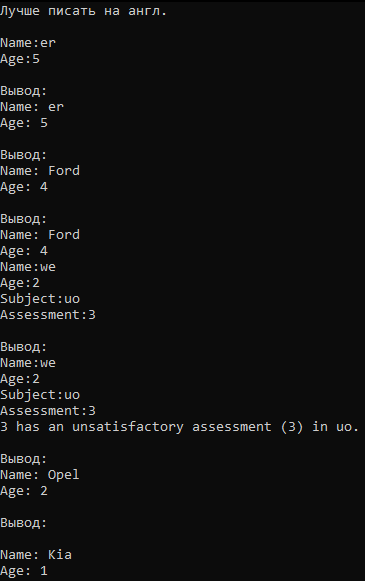








Вывод.



Контрольные вопросы.

1. Для чего используется механизм наследования?

Механизм наследования позволяет создавать новые классы на основе существующих, переиспользуя их поля и методы, а также расширяя или изменяя их поведение.

1. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Члены public базового класса остаются public в производном классе. Члены protected базового класса остаются protected в производном классе. Члены private базового класса недоступны в производном классе.

1. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Члены public и protected базового класса становятся private в производном классе. Члены private базового класса недоступны в производном классе.

1. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

Члены public и protected базового класса становятся protected в производном классе. Члены private базового класса недоступны в производном классе.

1. Каким образом описывается производный класс?

Class Pr\_class : mod\_dostupa bas\_class; {};

1. Наследуются ли конструкторы?

Конструкторы не наследуются по умолчанию.

1. Наследуются ли деструкторы?

Деструкторы не наследуются.

1. В каком порядке конструируются объекты производных классов?
2. Выделение памяти под объект.
3. Конструирование базовых классов
4. Инициализация полей класса
5. Выполнение тела конструктора производного класса
6. В каком порядке уничтожаются объекты производных классов?
7. Выполняется тело деструктора производного класса.
8. Уничтожаются члены класса (поля) в порядке, обратном их объявлению.
9. Вызываются деструкторы базовых классов в порядке, обратном порядку наследования.
10. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции (virtual) - функция-член класса, объявленная с ключевым словом virtual. Позволяет переопределять её в производных классах. Вызов виртуальной функции определяется типом объекта.

Механизм позднего связывания (Dynamic Binding) — это позднее связывание (динамическое). Определяется вызываемой функции во время выполнения, а не компиляции. Реализуется через таблицу виртуальных функций

1. Могут ли быть виртуальными конструкторы? Деструкторы?

Нельзя объявлять конструкторы виртуальными, а деструкторы можно.

1. Наследуется ли спецификатор virtual?

Если функция объявлена виртуальной в базовом классе, то она остаётся виртуальной во всех производных классах, даже если не указано ключевое слово virtual.

1. Какое отношение устанавливает между классами открытое наследование?

Производный класс — это специализированная версия базового класса, сохраняющая все его публичные свойства и поведение. (is-a).

1. Какое отношение устанавливает между классами закрытое наследование?

Производный класс использует функциональность базового класса, но не предоставляет его интерфейс внешнему миру. (implemented-in-terms-of).

1. В чем заключается принцип подстановки?

Объекты производного класса должны быть заменяемы на объекты базового класса без изменения корректности программы.

1. Имеется иерархия классов:

class Student { int age; public: string name; ... };

class Employee :

public Student { protected: string post; ... };

class Teacher :

public Employee { protected: int stage; ... };

Teacher x;

Какие компонентные данные будет иметь объект х?

1. string name: унаследован от Student как public.
2. string post: унаследован от Employee как protected.
3. int stage: объявлен как protected.
4. int age: унаследован от Student, он недоступен напрямую.
5. Для классов Student, Employee и Teacher написать конструкторы без параметров.

Student() : age(0), name(" ") {}

Employee() : post(" ") {}

Teacher() : stage(0) {}

1. Для классов Student, Employee и Teacher написать конструкторы с параметрами.

Student(int a, const string& n) : age(a), name(n) {}

Employee(int a, const string& n, const string& p) : Student(a, n), post(p) {}

Teacher(int a, const string& n, const string& p, int s): Employee(a, n, p), stage(s) {}

1. Для классов Student, Employee и Teacher написать конструкторы копирования.

Student(const Student& other) : age(other.age), name(other.name) {}

Employee(const Employee& other) : Student(other), post(other.post) {}

Teacher(const Teacher& other) : Employee(other), stage(other.stage) {}

1. Для классов Student, Employee и Teacher определить операцию присваивания.

Student& operator=(const Student& other) {

if (this != &other) {

age = other.age;

name = other.name}

return \*this;}

Employee& operator=(const Employee& other) {

if (this != &other) {

Student::operator=(other);

post = other.post;}

return \*this;}

Teacher& operator=(const Teacher& other) {

if (this != &other) {

Employee::operator=(other);

stage = other.stage;}

return \*this;}