Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

**ОТЧЁТ**

Тема: «Простое наследование. Принцип подстановки»

Выполнил

Студент РИС-22-2б

Зубов Р.А.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

# **Содержание**

Постановка задачи**4**

Коды программ**5**

Результаты программ**6**

**Постановка задачи**

Определить пользовательский класс. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования. Определить в классе деструктор. Определить в классе компоненты – функции для просмотра и установки полей данных(селекторы и модификаторы). Перегрузить операцию присваивания. Перегрузить операции ввода и вывода объектов с помощью потоков. Определить производный класс. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

**Задача**

Базовый класс

ЧЕЛОВЕК (PERSON)

Имя (name) - string

Возраст (age) - int

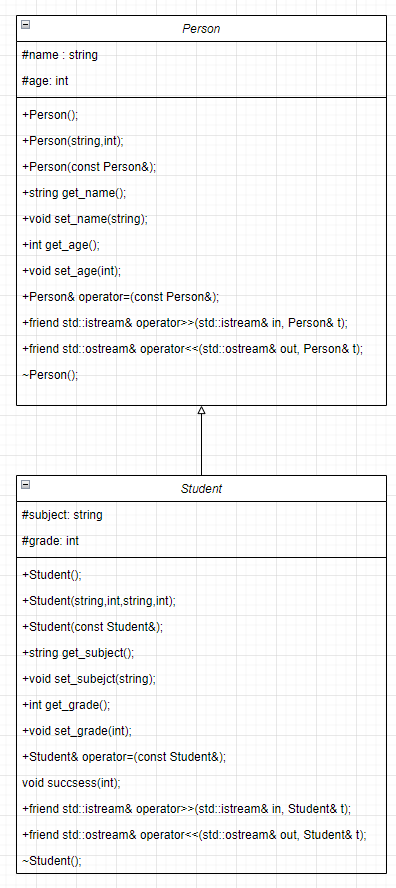
Определить методы изменения полой

Создать производный класо STUDENT, имеющий поля Предмет - string и Оценка - int.

Определить методы изменския полей и метод, выдающий сообщение о

неудовлетворительной оценке

**UML диаграмма**

****

**Описание классов**

Этот код определяет класс Person, который содержит два защищенных поля: name типа std::string и age типа int. В классе также определены конструкторы по умолчанию, конструктор с параметрами и конструктор копирования. Деструктор пустой. Есть два метода доступа: get\_name() и get\_age(), которые возвращают значения соответствующих полей. Методы set\_name() и set\_age() устанавливают значения соответствующих полей. Определены оператор присваивания и операторы ввода и вывода для работы с потоками.

Класс Student является производным от класса Person. Он содержит два защищенных поля: subject типа std::string и grade типа int. В классе также определены конструкторы по умолчанию, конструктор с параметрами и конструктор копирования. Деструктор пустой. Есть два метода доступа: get\_subject() и get\_grade(), которые возвращают значения соответствующих полей. Методы set\_subject() и set\_grade() устанавливают значения соответствующих полей. Определены оператор присваивания и операторы ввода и вывода для работы с потоками. Также есть метод succsess(int), который определяет положительность оценки.

**Определение компонентных функций**

Конструктор по умолчанию Person::Person(void) инициализирует поля name и age пустой строкой и нулем соответственно. Конструктор с параметрами Person::Person(std::string N, int A) инициализирует поля name и age значениями параметров N и A. Конструктор копирования Person::Person(const Person&p) копирует значения полей name и age из объекта p. Методы set\_name(std::string N) и set\_age(int A) устанавливают значения соответствующих полей в значение параметров N и A. Оператор присваивания Person& Person::operator=(const Person&p) копирует значения полей из объекта p, если он не равен текущему объекту.

Конструктор по умолчанию Student::Student(void) вызывает конструктор по умолчанию базового класса Person и инициализирует поля subject и grade пустой строкой и нулем соответственно. Конструктор с параметрами Student::Student(std::string N, int A, std::string S, int G) вызывает конструктор с параметрами базового класса Person(N,A) и инициализирует поля subject и grade значениями параметров S и G. Конструктор копирования Student::Student(const Student&c) копирует значения полей из объекта c. Методы set\_subject(std::string S) и set\_grade(int G) устанавливают значения соответствующих полей в значение параметров S и G. Оператор присваивания Student& Student::operator=(const Student&c) копирует значения полей из объекта c, если он не равен текущему объекту.

**Определение глобальных функций**

Глобальные функции std::istream& operator>>(std::istream&in, Pair&t) и std::ostream& operator<<(std::ostream & out, Pair &t) перегружают операторы >> и << для объектов класса Pair соответственно.

Оператор ввода (>>) перегружен следующим образом:

* Вводится первое поле объекта.
* Вводится второе поле объекта.
* Возвращается ссылка на поток ввода.

Оператор вывода (<<) перегружен следующим образом:

* Выводится первое поле объекта.
* Выводится разделитель " : ".
* Выводится второе поле объекта.
* Выводится символ новой строки.
* Возвращается ссылка на поток вывода.

**Определение функции main()**

Функция f1 принимает ссылку на объект типа Person и устанавливает его поля name и age значениями "John" и 14 соответственно. Затем выводит объект в стандартный поток вывода. Функция f2 создает локальный объект типа Student с параметрами "Travis", 13, "Physics", 2, и возвращает его.

Функция main создает объект типа Person с именем a и считывает его значения из стандартного потока ввода. Затем выводит его в стандартный поток вывода. Создает объект типа Person с именем b с параметрами "Ivan" и 10, и выводит его в стандартный поток вывода. Присваивает объекту a значение объекта b и выводит его в стандартный поток вывода.

Создает объект типа Student с именем c и считывает его значения из стандартного потока ввода. Затем выводит его в стандартный поток вывода. Вызывает функцию f1 с параметром c. Присваивает объекту c значение, возвращаемое функцией f2, и выводит его в стандартный поток вывода.

**Контрольные вопросы**

1. Для чего используется механизм наследования?

- Для создания новых классов, на основе уже существующих.

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

Компоненты класса, описанные со спецификатором public, становятся доступными в базовом классе. Если тип наследования public, то становятся доступными для обращения из экземпляра производного класса. Если тип наследования protected, то доступны только внутри базового класса.

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

Компоненты, описанные со спецификатором private, не доступны в производном классе.

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

-Компоненты, описанные со спецификатором protected, при типе наследования public/protected остаются protected – доступны в производном классе и в классах-наследниках производного. При типе наследования private, становятся private – Доступны только внутри производного класса.

5. Каким образом описывается производный класс?

Class <имя> : <тип наследования> <имя базового>,…

6. Наследуются ли конструкторы?

-Конструкторы не наследуются, однако могут быть вызваны для передачи параметров.

7. Наследуются ли деструкторы?

- Деструкторы не наследуются.

8. В каком порядке конструируются объекты производных классов?

-Сначала Компоненты базового класса, потом базовый класс, потом компоненты производного класса, затем производный. компоненты конструируются в том порядке, в котором они записаны.

9. В каком порядке уничтожаются объекты производных классов?

В обратном порядке создания. Сначала производный класс, затем компоненты производного (в обратном порядке их записи), затем базовый класс, затем компоненты базового класса(в обратном порядке их записи).

10. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции – это функции которые могут быть переопределены в производном. Используя виртуальную функцию, мы можем вызывать функции производного класса, используя указатель базового класса.

Когда в классе объявляется виртуальный метод, то создается специальная таблица для этого класса. Эта таблица представляет собой массив указателей на виртуальные функции. Во время компиляции нет информации о том, какой метод будет вызван. Вызов метода выполняется во время выполнения программы.

11. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы виртуальными быть не могут, в отличие от деструкторов.

12. Наследуется ли спецификатор virtual?

Виртуальность наследуется.

13. Какое отношение устанавливает между классами открытое наследование?

- Открытое наследование является наследованием типа. Т. е. производный класс является частным случаем базового класса (если не происходит переопределение методов класса – для реализации принципа подстановки). Или производный класс может замещать методы базового класса (в том случае, если компонентные функции базового класса описаны со спецификатором virtual – для реализации полиморфизма).

14. Какое отношение устанавливает между классами закрытое наследование?

Производный имеет доступ ко всему функционалу, за исключением private компонентов, класса-предка, но ни наследники, ни экземпляры класса не имеют доступ к этому функционалу. Это нужно для того, чтобы средствами базового класса реализовать новый класс.

15. В чем заключается принцип подстановки?

- Если класс D является частным случаем класса B, то везде, где был использован класс B, можно использовать класс D. Принцип подстановки возможен, если не были переопределены методы базового класса.

16. 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?

- Через объект x можно обратиться только к полю name. Но компонентные функции и друзья класса Teacher имеют доступ к protected полям string post и int stage;