Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

ПРОСТОЕ НАСЛЕДОВАНИЕ. ПРИНЦИП ПОДСТАНОВКИ.

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2024

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе селекторы и модификаторы.
5. Перегрузить операцию присваивания.
6. Перегрузить операцию ввода и вывода объектов с помощью потоков.
7. Определить производный класс.
8. Реализовать функции, получающие и возвращающие объект базового класса.

Для варианта 15:

Базовый класс:

ЧЕЛОВЕК (PERSON)

Имя – string

Возраст – int

Определить методы изменения полей

Производный класс:

STUDENT

Предмет – string

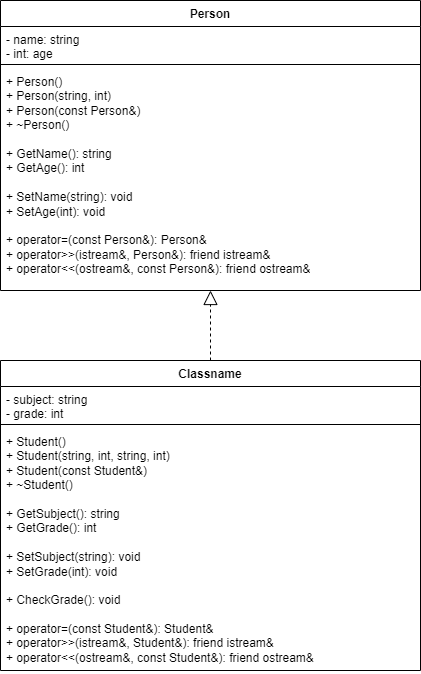
Оценка – int

Определить методы изменения полей и метод сообщающий о неудовлетворительной оценке.

**Анализ задачи**

1. Создается новый файл с расширением .h, где определяется новый класс Person.
2. Создается парный .cpp файл, в котором определяются методы нового класса.
3. При определении производного класса используется модификатор доступа public для полей базового класса.

**UML диаграмма**



**Код**

Person.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Person {

protected:

string name;

int age;

public:

Person();

Person(string, int);

Person(const Person&);

~Person();

string GetName();

int GetAge();

void SetName(string);

void SetAge(int);

Person& operator=(const Person&);

friend istream& operator>>(istream&, Person&);

friend ostream& operator<<(ostream&, const Person&);

};

class Student: public Person {

protected:

string subject;

int grade;

public:

Student();

Student(string, int, string, int);

Student(const Student&);

~Student();

string GetSubject();

int GetGrade();

void SetSubject(string);

void SetGrade(int);

void CheckGrade();

Student& operator=(const Student&);

friend istream& operator>>(istream&, Student&);

friend ostream& operator<<(ostream&, const Student&);

};

Person.cpp

#include "Person.h"

#include <string>

#include <iostream>

using namespace std;

int main() {

system("chcp 1251>null");

Student test\_student;

cout << "Created empty Student object" << endl;

cout << test\_student;

cout << "Enter Student object data" << endl;

cin >> test\_student;

cout << test\_student;

cout << "New Student object copied data from first" << endl;

Student new\_student;

new\_student = test\_student;

cout << new\_student;

new\_student.CheckGrade();

cout << "Student object enterpreted as Person object" << endl;

Person\* test\_ptr = &test\_student;

cout << \*test\_ptr;

cin >> \*test\_ptr;

cout << test\_student;

return 0;

}

Main.cpp

#include "Person.h"

#include <string>

#include <iostream>

using namespace std;

int main() {

system("chcp 1251>null");

Student test\_student;

cout << "Created empty Student object" << endl;

cout << test\_student;

cout << "Enter Student object data" << endl;

cin >> test\_student;

cout << test\_student;

cout << "New Student object copied data from first" << endl;

Student new\_student;

new\_student = test\_student;

cout << new\_student;

new\_student.CheckGrade();

cout << "Student object enterpreted as Person object" << endl;

Person\* test\_ptr = &test\_student;

cout << \*test\_ptr;

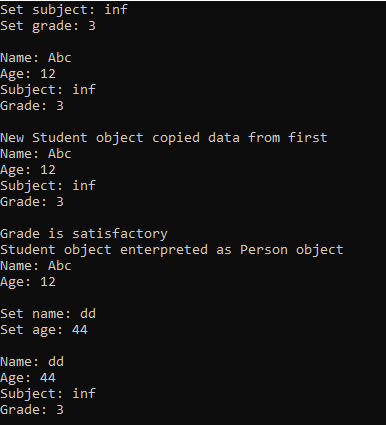
cin >> \*test\_ptr;

cout << test\_student;

return 0;

}

**Решение**



**Выводы**

В ходе выполнения работы были изучены принципы наследования и подстановки, а также методы работы с производными классами.

**Ответы на контрольные вопросы:**

1. Для создания классов имеющих одинаковые атрибуты, без определения каждого атрибута отдельно в каждом классе.
2. Private сегмент не наследуется, спецификаторы остальных сегментов сохраняются.
3. Private сегмент не наследуется, спецификаторы остальных сегментов становятся private.
4. Private сегмент не наследуется, спецификаторы остальных сегментов становятся protected.
5. Class /название производного класса/ : /спецификатор наследования/ /название базового класса/.
6. Нет.
7. Нет.
8. От базового к производному.
9. От производного к базовому.
10. Виртуальные функции – функции со спецификатором virtual. При использовании виртуальных функций они интерпретируются по ходу выполнения программы на основании значения указателя, а не на этапе компиляции.
11. Да.
12. Да.
13. Отношение «является»: базовый класс является частью производного.
14. Отношение наследования реализации, без сохранения принципа подстановки.
15. В любом месте, где используется базовый класс, можно использовать производный.
16. Name, post, stage.
17. Без параметров
    1. Student(){age = 0; name = “”};
    2. Employee():Student(){post = “”};
    3. Teacher():Employee(){stage = 0};
18. С параметрами
    1. Student(int a, string b){age = a; name = b};
    2. Employee(int a, string b, string c):Student(int a, string b){post = c};
    3. Teacher(int a, string b, string c, int d):Employee(int a, string b, string c){stage = d};
19. Копирования
    1. Student(Student& a){age = a.age; name = a.name};
    2. Employee(Employee& a):Student(a){post = a.post};
    3. Teacher(Teacher& a):Employee(a){stage = a.stage};
20. Операция присваивания
    1. Student& operator=(Student& a){this->age = a.age; this->name = a.name; return \*this };
    2. Employee& operator= (Employee& a): operator=(Student)a {this-> post = a.post; return \*this};
    3. Teacher& operator= (Teacher& a): operator=(Employee)a {this-> stage = a.stage; return \*this };

**Github**

<https://github.com/Hitikov/Lab_OOP_4>