Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №5

Тема: «Наследование. Виртуальные функции. Полиморфизм»

Выполнил

Студент группы РИС-22-2б

Малых В.О.

Проверил доц. Кафедры ИТАС

Полякова О.А.

Пермь 2023

# Постановка задачи

# Определить абстрактный класс.

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

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

# Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.

# В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

ВАРИАНТ 15:

Базовый класс:

ЧЕЛОВЕК (PERSON)

Имя (name) - string

Возраст (age) - int

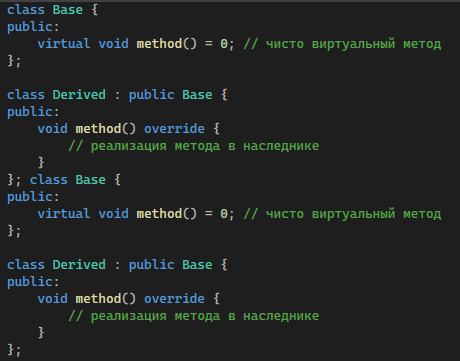
Определить методы изменения полей.

Создать производный класс STUDENT, имеющий поля Предмет - string и Оценка -int. Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке

**Контрольные вопросы**

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

Чисто виртуальный метод не имеет реализации в базовом классе, а только определение (объявление) с помощью ключевого слова "= 0". Он используется для создания абстрактных классов, чтобы все наследники обязательно переопределяли этот метод.



2. Какой класс называется абстрактным?

Абстрактным классом называется класс, содержащий хотя бы один чисто виртуальный метод.

3. Для чего предназначены абстрактные классы?

Абстрактные классы используются для создания общей структуры классов, которые будут иметь общие методы, но специфические реализации для каждого наследника. Также они могут служить для описания интерфейсов.

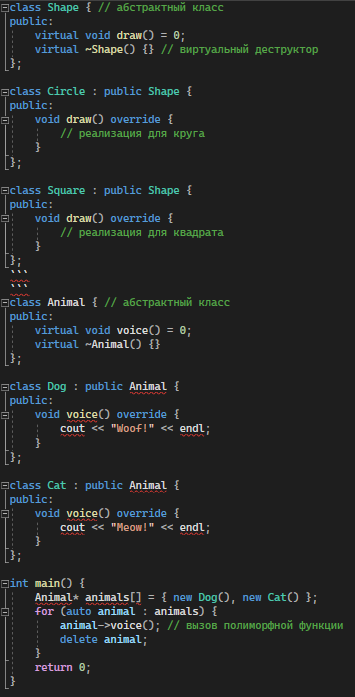
4. Что такое полиморфные функции?

Полиморфная функция - это функция, которая может работать с объектами разных классов, имеющих общего предка, и при этом использовать различные реализации одного и того же метода в зависимости от конкретного класса объекта.

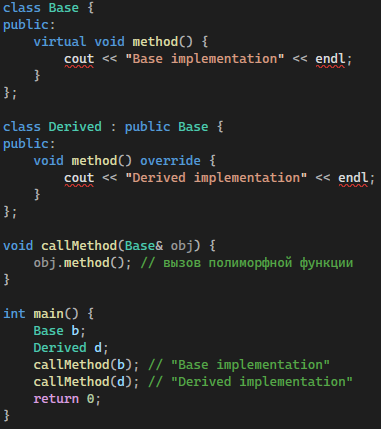
5. Чем полиморфизм отличается от принципа подстановки?

Полиморфизм - это свойство языка, позволяющее использовать один и тот же код для объектов разных классов, имеющих общего предка. Принцип подстановки - это принцип SOLID, который требует, чтобы объекты одного и того же класса взаимозаменяли друг друга в любом месте программы без изменения корректности выполнения программы.

6. Привести примеры иерархий с использованием абстрактных классов.



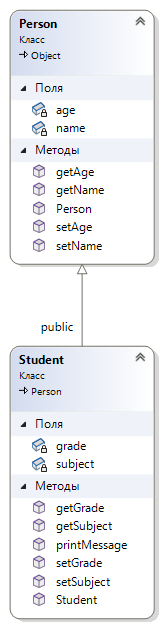
7. Привести примеры полиморфных функций.



8. В каких случаях используется механизм позднего связывания?

Механизм позднего связывания используется в случае, когда нужно вызывать методы наследника из указателя на предка (например, в полиморфных функциях). Позднее связывание гарантирует, что будет вызвана реализация метода из конкретного класса объекта, а не из его базового класса. Позднее связывание достигается с помощью виртуальных функций в базовом классе и ключевого слова "override" в наследнике.

UML – диаграмма



Код программы

**Object.h**

#pragma once

#ifndef OBJECT\_H

#define OBJECT\_H

class Object {

public:

virtual ~Object() = default;

};

#endif

**Student.cpp**

#include "Student.h"

#include <iostream>

Student::Student(const std::string& name, int age, const std::string& subject, int grade)

: Person(name, age), subject(subject), grade(grade)

{}

void Student::setSubject(const std::string& subject) {

this->subject = subject;

}

void Student::setGrade(int grade) {

this->grade = grade;

}

std::string Student::getSubject() const {

return subject;

}

int Student::getGrade() const {

return grade;

}

void Student::printMessage() const {

if (grade < 4) {

std::cout << "Неудовлетворительная оценка по предмету " << subject << "!\n";

}

}

**Student.h**

#pragma once

#ifndef STUDENT\_H

#define STUDENT\_H

#include "Person.h"

#include <string>

class Student : public Person {

public:

Student(const std::string& name, int age, const std::string& subject, int grade);

void setSubject(const std::string& subject);

void setGrade(int grade);

std::string getSubject() const;

int getGrade() const;

void printMessage() const;

private:

std::string subject;

int grade;

};

#endif

**Person.cpp**

#include "Person.h"

Person::Person(const std::string& name, int age)

: name(name), age(age)

{}

void Person::setName(const std::string& name) {

this->name = name;

}

void Person::setAge(int age) {

this->age = age;

}

std::string Person::getName() const {

return name;

}

int Person::getAge() const {

return age;

}

**Person.h**

#pragma once

#ifndef PERSON\_H

#define PERSON\_H

#include "Object.h"

#include <string>

class Person : public Object {

public:

Person(const std::string& name, int age);

void setName(const std::string& name);

void setAge(int age);

std::string getName() const;

int getAge() const;

private:

std::string name;

int age;

};

#endif

**main.cpp**

#include <iostream>

#include "Person.h"

#include "Student.h"

int main() {

setlocale(LC\_ALL, "Rus");

Person p("Петя", 25);

std::cout << "Имя: " << p.getName() << ", Возраст: " << p.getAge() << std::endl;

p.setName("Катя");

p.setAge(20);

std::cout << "Имя: " << p.getName() << ", Возраст: " << p.getAge() << std::endl;

Student s("Иван", 19, "Физкультура", 3);

std::cout << "Имя: " << s.getName() << ", Возраст: " << s.getAge() << ", Предмет: " << s.getSubject() << ", Оценка: " << s.getGrade() << std::endl;

s.setSubject("Физика");

s.setGrade(4);

std::cout << "Имя: " << s.getName() << ", Возраст: " << s.getAge() << ", Предмет: " << s.getSubject() << ", Оценка: " << s.getGrade() << std::endl;

s.setGrade(2);

s.printMessage();

return 0;

}