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 №4

Тема: «Простое наследование. Принцип подстановки.»

Выполнил

Студент группы РИС-22-2б

Скорюпин Д.А.

Проверил доц. Кафедры ИТАС

Полякова О.А.

Пермь 2023

# 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.

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

**Контрольные вопросы**

1. Для чего используется механизм наследования?

Механизм наследования используется для создания новых классов на основе уже существующих, чтобы избежать дублирования кода и повторного описания свойств и методов базового класса в производных классах.

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

Компоненты класса, описанные со спецификатором public, наследуются открыто и доступны для использования в производном классе.

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

Компоненты класса, описанные со спецификатором private, не наследуются и не доступны для использования в производном классе.

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

Компоненты класса, описанные со спецификатором protected, наследуются защищенно и доступны для использования в производном классе и его потомках.

5. Каким образом описывается производный класс?

Производный класс описывается с помощью ключевого слова class, за которым следует имя нового класса и двоеточие, а затем указывается имя базового класса, от которого производится наследование (например, class DerivedClass : public BaseClass).

6. Наследуются ли конструкторы?

Конструкторы наследуются, но не переопределяются.

7. Наследуются ли деструкторы?

Деструкторы наследуются, но не переопределяются.

8. В каком порядке конструируются объекты производных классов?

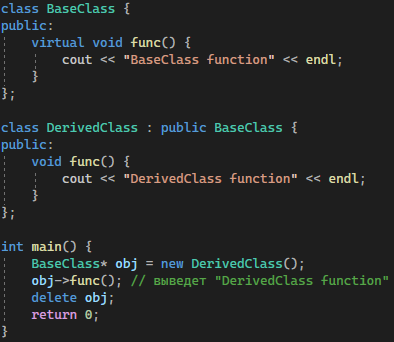
Сначала конструируется базовый класс, затем производный класс.

9. В каком порядке уничтожаются объекты производных классов?

Сначала уничтожается производный класс, затем базовый класс.

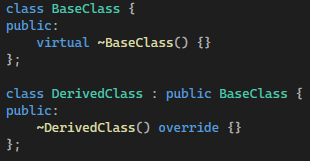
10. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции позволяют вызывать методы производного класса через указатель или ссылку на базовый класс. Механизм позднего связывания обеспечивает выбор правильного метода в зависимости от типа объекта, на который указывает указатель или ссылка. Пример:



11. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы не могут быть виртуальными, деструкторы могут быть виртуальными. Пример:

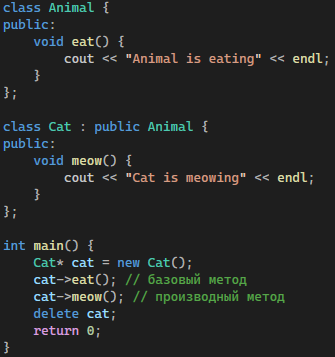


12. Наследуется ли спецификатор virtual?

Спецификатор virtual не наследуется.

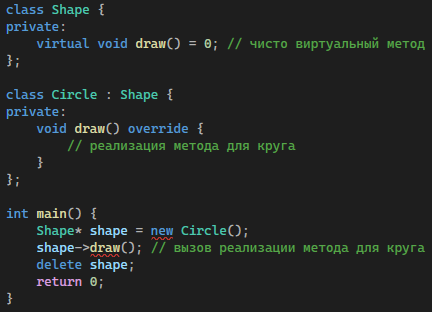
13. Какое отношение устанавливает между классами открытое наследование?

Открытое наследование устанавливает отношение "является" между базовым и производным классами, т.е. производный класс является расширением базового класса. Например:



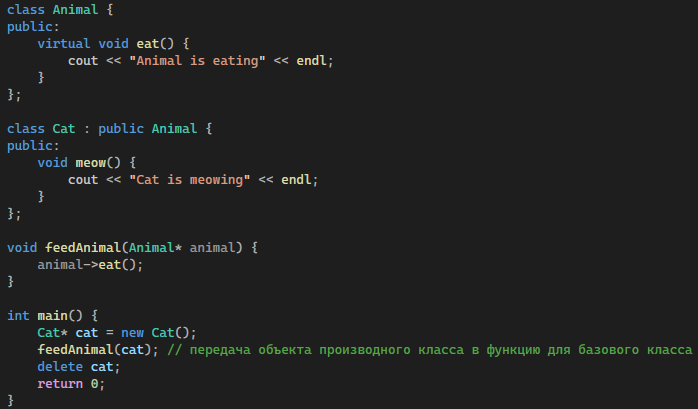
14. Какое отношение устанавливает между классами закрытое наследование?

Закрытое наследование устанавливает отношение "реализует" между базовым и производным классами, т.е. производный класс реализует интерфейс базового класса, но не является его расширением. Например:



15. В чем заключается принцип подстановки?

Принцип подстановки гласит, что объекты производного класса могут использоваться везде, где ожидается объект базового класса, не нарушая при этом корректности программы. Например:



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?

Объект x будет иметь компоненты данных age, name, post и stage.

17. Для классов Student, Employee и Teacher написать конструкторы без параметров.

Конструкторы без параметров для классов Student, Employee и Teacher могут выглядеть так:



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

Конструкторы с параметрами для классов Student, Employee и Teacher могут выглядеть так:



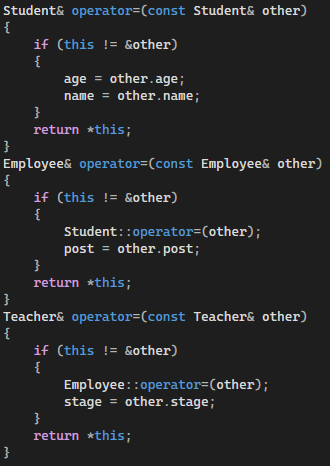
19. Для классов Student, Employee и Teacher написать конструкторы копирования.

Конструкторы копирования для классов Student, Employee и Teacher могут выглядеть так:

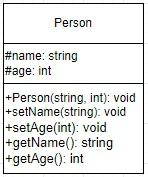


20. Для классов Student, Employee и Teacher определить операцию присваивания.

Операция присваивания для классов Student, Employee и Teacher может быть определена так:



UML – таблица



Код программы

**Person.h**

#pragma once

#include <string>

class Person {

protected:

std::string name;

int age;

public:

Person(std::string n, int a);

void setName(std::string n);

void setAge(int a);

std::string getName();

int getAge();

};

**Person.cpp**

#include "Person.h"

Person::Person(std::string n, int a) {

name = n;

age = a;

}

void Person::setName(std::string n) {

name = n;

}

void Person::setAge(int a) {

age = a;

}

std::string Person::getName() {

return name;

}

int Person::getAge() {

return age;

}

**Main.cpp**#include <iostream>

#include "Person.h"

class Student : public Person {

private:

std::string subject;

int grade;

public:

Student(std::string n = "", int a = 0, std::string s = "", int g = 0);

void setSubject(std::string s);

void setGrade(int g);

std::string getSubject();

int getGrade();

void checkGrade();

friend std::istream& operator>>(std::istream& is, Student& s);

};

Student::Student(std::string n, int a, std::string s, int g) : Person(n, a) {

subject = s;

grade = g;

}

void Student::setSubject(std::string s) {

subject = s;

}

void Student::setGrade(int g) {

grade = g;

}

std::string Student::getSubject() {

return subject;

}

int Student::getGrade() {

return grade;

}

void Student::checkGrade() {

if (grade < 4) {

std::cout << "Оценка студента неудовлетворительная!" << std::endl;

}

else {

std::cout << "Оценка студента удовлетворительная!" << std::endl;

}

}

std::istream& operator>>(std::istream& is, Student& s) {

std::string n, sub;

int a, g;

std::cout << "Введите имя студента: ";

is >> n;

s.setName(n);

std::cout << "Введите возраст студента: ";

is >> a;

s.setAge(a);

std::cout << "Введите название предмета: ";

is >> sub;

s.setSubject(sub);

std::cout << "Введите оценку: ";

is >> g;

s.setGrade(g);

return is;

}

int main() {

Student student;

std::cin >> student;

std::cout << "Имя: " << student.getName() << std::endl;

std::cout << "Возраст: " << student.getAge() << std::endl;

std::cout << "Предмет: " << student.getSubject() << std::endl;

std::cout << "Оценка: " << student.getGrade() << std::endl;

student.checkGrade();

return 0;

}