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

Юсупов Т. 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 2:

Базовый класс:

ПАРА ЧИСЕЛ (PAIR)

Первое\_число (first) - int

Второе\_число (second) - int

Определить методы изменения полей и вычисления произведения чисел.

Создать производный класс ПРЯМОУГОЛЬНИК (RECTANGLE), с полями - сторонами. Определить методы для вычисления площади и периметра прямоугольника.

**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 защищено 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, classDerivedClass : publicBaseClass).

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?

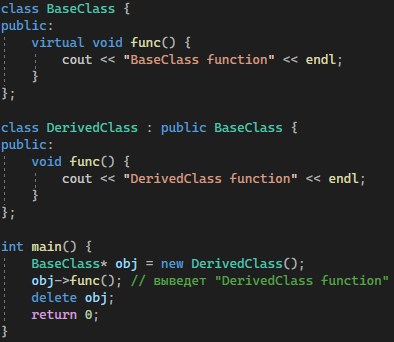
Сначала конструируется базовый класс, затем производный класс.

9. В каком порядке уничтожаются объекты производных классов?

Сначала уничтожается производный класс, затем базовый класс.

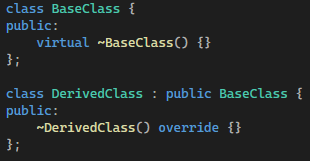
10. Что представляют собой виртуальные функции и механизм позднего связывания?

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:

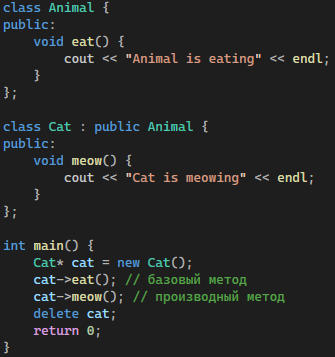


12. Наследуется ли спецификатор virtual?

Спецификатор virtual не наследуется.

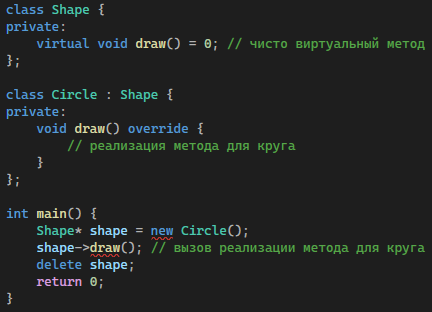
13. Какое отношение устанавливает между классами открытое наследование?

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



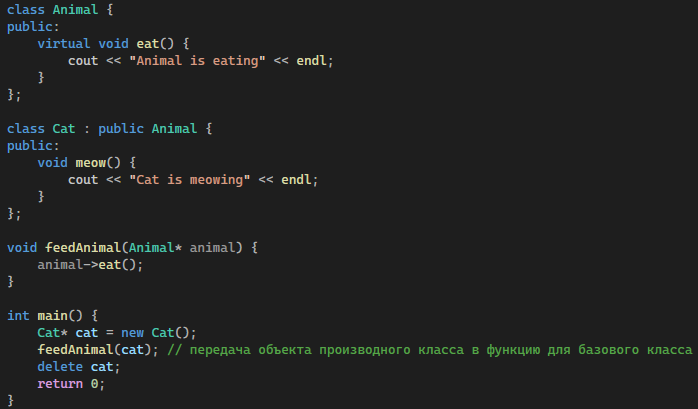
14. Какое отношение устанавливает между классами закрытое наследование?

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:

classStudent

{

int age;

public:

string name;

};

class Employee : public Student

{

protected:

string post;

};

class Teacher : public Employee

{

protected: int stage;

};

Teacherx;

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:



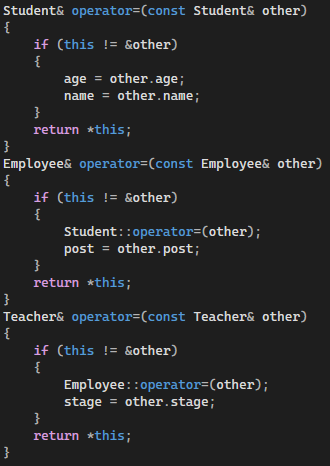
19. Для классов Student, Employee и Teacher написать конструкторы копирования.

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

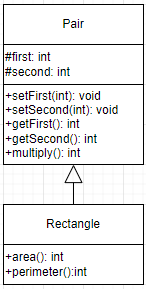


20. Для классов Student, Employee и Teacher определить операцию присваивания.

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



UML– таблица



Кодпрограммы

Pair.h:

#pragmaonce

classPair {

protected:

int first;

int second;

public:

Pair(inta, intb) : first(a), second(b) {}

voidsetFirst(int);

voidsetSecond(int);

intgetFirst();

intgetSecond();

int multiply();

};

Rectangle.h:

#pragmaonce

#include"Pair.h"

classRectangle : publicPair {

public:

Rectangle(inta, intb) : Pair(a, b) {}

intarea();

intperimeter();

};

Pair.cpp:

#include"Pair.h"

voidPair::setFirst(inta)

{

this->first = a;

}

voidPair::setSecond(intb)

{

this->second = b;

}

intPair::getFirst()

{

returnthis->first;

}

intPair::getSecond()

{

returnthis->second;

}

intPair::multiply()

{

returnthis->first \* this->second;

}

Rectangle.cpp:

#include"Rectangle.h"

intRectangle::area()

{

returnthis->first \* this->second;

}

intRectangle::perimeter()

{

return 2 \* (this->first + this->second);

}

Labaratorka4.cpp:

#include"Rectangle.h"

#include<iostream>

usingnamespace std;

int main() {

Rectangle rect(5, 7);

cout <<"Area: "<< rect.area() << endl;

cout <<"Perimeter: "<< rect.perimeter() << endl;

rect.setFirst(10);

rect.setSecond(15);

cout <<"New area: "<< rect.area() << endl;

cout <<"Multiplication: "<< rect.multiply() << endl;

return 0;

}