Министерство цифрового развития, связи и массовых коммуникаций РФ

Федеральное государственное бюджетное общеобразовательное учреждение

высшего образования

“Сибирский Государственный Университет Телекоммуникаций и Информатики”

(СибГУТИ)

Кафедра телекоммуникационных систем и вычислительных средств

Отчет по лабораторной работе 5 на тему: Наследование и полиморфизм

Выполнил: студент 1 курса группы ИА-231

Зырянов Иван Александрович

Проверила: Моренкова Ольга Ильинична

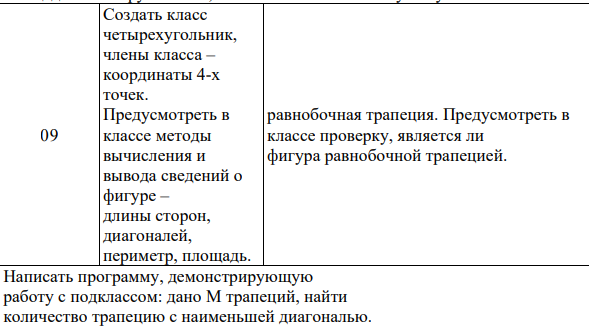
Новосибирск, 2023

# Цель работы: Освоить основные принципы объектно-ориентированного программирования, изучить понятия базового класса и класса-наследника, научиться применять динамический вызов методов, ознакомиться с принципами позднего связывания и полиморфизма.

# Задание:

Определить суперкласс в соответствии с вариантом задания. Описать поля и методы класса. Создать класс Main с методом main() для проверки работы методов класса. Определить дочерний класса в соответствии с вариантом задания. Определить дополнительные поля и добавить механизм их инициализации в конструкторе. Переопределить метод вывода информации об объекте в дочернем классе. Скомпилировать проект и продемонстрировать работу методов.

Вариант 9:



Код:

#include <iostream>

#include <cmath>

using namespace std;

class Quadrilateral {

protected:

    double x1, y1, x2, y2, x3, y3, x4, y4;

public:

    Quadrilateral(double x1\_, double y1\_, double x2\_, double y2\_, double x3\_, double y3\_, double x4\_, double y4\_) {

        x1 = x1\_; y1 = y1\_;

        x2 = x2\_; y2 = y2\_;

        x3 = x3\_; y3 = y3\_;

        x4 = x4\_; y4 = y4\_;

    }

    double distance(double x1\_, double y1\_, double x2\_, double y2\_) {

        return sqrt(pow(x2\_ - x1\_, 2) + pow(y2\_ - y1\_, 2));

    }

    void displayInfo() {

        cout << "Coordinates: (" << x1 << "," << y1 << "), (" << x2 << "," << y2 << "), (" << x3 << "," << y3 << "), (" << x4 << "," << y4 << ")" << endl;

        cout << "Side lengths: " << distance(x1, y1, x2, y2) << ", " << distance(x2, y2, x3, y3) << ", " << distance(x3, y3, x4, y4) << ", " << distance(x4, y4, x1, y1) << endl;

        cout << "Diagonal lengths: " << distance(x1, y1, x3, y3) << ", " << distance(x2, y2, x4, y4) << endl;

        cout << "Perimeter: " << distance(x1, y1, x2, y2) + distance(x2, y2, x3, y3) + distance(x3, y3, x4, y4) + distance(x4, y4, x1, y1) << endl;

        cout << "Area: " << abs((x1\*y2 + x2\*y3 + x3\*y4 + x4\*y1) - (y1\*x2 + y2\*x3 + y3\*x4 + y4\*x1)) / 2 << endl;

    }

};

class IsoscelesTrapezoid : public Quadrilateral {

public:

    IsoscelesTrapezoid(double x1\_, double y1\_, double x2\_, double y2\_, double x3\_, double y3\_, double x4\_, double y4\_) : Quadrilateral(x1\_, y1\_, x2\_, y2\_, x3\_, y3\_, x4\_, y4\_) {

    }

    bool isIsosceles() {

        double topLength = distance(x1, y1, x2, y2);

        double bottomLength = distance(x3, y3, x4, y4);

        double sideLength1 = distance(x1, y1, x4, y4);

        double sideLength2 = distance(x2, y2, x3, y3);

        return (topLength == bottomLength) || (sideLength1 == sideLength2);

}

double getHeight() {

    double topLength = distance(x1, y1, x2, y2);

    double bottomLength = distance(x3, y3, x4, y4);

    double diagonalLength = distance(x1, y1, x3, y3);

    double height = sqrt(pow(diagonalLength, 2) - pow((topLength - bottomLength), 2)) / 2;

    return height;

}

double getArea() {

    double topLength = distance(x1, y1, x2, y2);

    double bottomLength = distance(x3, y3, x4, y4);

    double height = getHeight();

    return ((topLength + bottomLength) / 2) \* height;

}

};

int main() {

IsoscelesTrapezoid it(0, 0, 0, 4, 3, 4, 6, 0);

it.displayInfo();

cout << "Isosceles: " << it.isIsosceles() << endl;

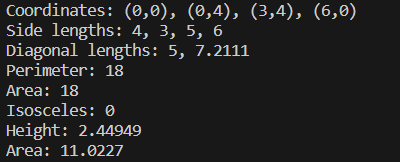
cout << "Height: " << it.getHeight() << endl;

cout << "Area: " << it.getArea() << endl;

return 0;

}

Вывод программы:



Описание полученных результатов:

Программа работает хорошо и выдает то что следует по задание.

Выводы:

Мы освоили основные принципы объектно-ориентированного программирования, изучить понятия базового класса и класса-наследника, научились применять динамический вызов методов, ознакомиться с принципами позднего связывания и полиморфизма.