МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

**Институт математики, естественных и компьютерных наук**

**Информатика и вычислительная техника**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

Наследование в С++. Виртуальные функции.

Дисциплина: «Объектно-ориентированное программирование»

Направление подготовки: 09.03.01. Информатика и вычислительная техника

|  |  |
| --- | --- |
| Руководитель | Сорокин А.Н. |
| Выполнили студенты | Пчелкина О.С. |
| Группа, курс | ВМ-41 |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Вологда

2022 г.

**Цель работы**: научиться применять виртуальные функции при разработке классов на языке С++.

# Вариант 1

1. Создать базовый класс «фигура» с виртуальными функциями: площадь, периметр, вывод информации о фигуре.

Создать производные классы: прямоугольник, круг, прямоугольный треугольник, трапеция.

В производных классах определить: скрытые поля, конструктор по умолчанию, конструктор с параметрами, методы для установки и получения значение полей, методы для расчета площади, периметра и вывода информации.

2. Разработать программу, которая демонстрировала бы работоспособность всех методов классов. Для проверки определить массив ссылок на базовый класс, которым присваиваются адреса различных объектов.

Код программы:

#include<iostream>

#include<math.h>

#include <vector>

using namespace std;

class Figures

{

public:

virtual float Area()

{

return 0;

}

virtual float Perimeter()

{

return 0;

}

virtual void InformatiomOutput()

{

}

};

class Rectangle : Figures

{

private:

float \_width;

float \_height;

public:

Rectangle()

{

\_width = 2;

\_height = 4;

}

Rectangle(float width, float height)

{

\_width = width;

\_height = height;

}

float Area() override

{

return \_width \* \_height;

}

float Perimeter() override

{

return \_width \* 2 + \_height \* 2;

}

void InformatiomOutput() override

{

cout << "Площадь прямоугольника: " << Area() << endl;

cout << "Периметр прямоугольника: " << Perimeter() << endl;

}

void SetWidth(float width)

{

\_width = width;

}

void SetHeight(float height)

{

\_height = height;

}

float GetWidth()

{

return \_width;

}

float GetHeight()

{

return \_height;

}

};

class Circle : Figures

{

private:

float \_radius;

public:

Circle()

{

\_radius = 5;

}

Circle(float radius)

{

\_radius = radius;

}

float Area() override

{

return 3.14 \* \_radius \* \_radius;

}

float Perimeter() override

{

return 2 \* \_radius \* 3.14;

}

void InformatiomOutput() override

{

cout << "Площадь круга: " << Area() << endl;

cout << "Периметр круга: " << Perimeter() << endl;

}

void SetRadius(float radius)

{

\_radius = radius;

}

float GetRadius()

{

return \_radius;

}

};

class RightTriangle : Figures

{

private:

float \_a;

float \_b;

float \_c;

bool Existens()

{

if (\_a + \_c > \_b)

{

if (\_a + \_b > \_c)

{

if (\_c + \_b > \_a)

return true;

}

else return false;

}

else return false;

}

bool TriangleView()

{

if (\_c \* \_c == (\_a \* \_a + \_b \* \_b))

return true;

else return false;

}

public:

RightTriangle()

{

\_a = 3;

\_b = 4;

\_c = 5;

}

RightTriangle(float a, float b, float c)

{

\_a = a;

\_b = b;

\_c = c;

if (!Existens())

{

cout << "Треугольника не существует!";

exit(0);

}

if (!TriangleView())

{

cout << "Треугольник не прямоугольный!";

exit(0);

}

}

float Area() override

{

return \_a \* \_b \* 0.5;

}

float Perimeter() override

{

return sqrt(\_a \* \_a + \_b \* \_b) + \_a + \_b;

}

void InformatiomOutput() override

{

if (!Existens())

{

cout << "Треугольника не существует!";

exit(0);

}

if (!TriangleView())

{

cout << "Треугольник не прямоугольный!";

exit(0);

}

cout << "Площадь прямоугольного треугольника: " << Area() << endl;

cout << "Периметр прямоугольного треугольника: " << Perimeter() << endl;

}

void SetA(float a)

{

\_a = a;

}

void SetB(float b)

{

\_b = b;

}

void SetC(float c)

{

\_c = c;

}

float GetA()

{

return \_a;

}

float GetB()

{

return \_b;

}

float GetC()

{

return \_c;

}

};

class Trapezoid : Figures

{

private:

float \_a;

float \_b;

float \_c;

float \_d;

float HeightCalculation()

{

auto numerator = pow((\_a - \_b), 2) + \_c \* \_c - \_d \* \_d;

auto denominator = 2 \* (\_a - \_b);

auto result = numerator/denominator;

return sqrt(\_c \* \_c - pow(result, 2));

}

bool Existens()

{

vector<float> vec = { \_a, \_b, \_c, \_d };

float max = 0;

float sum = \_a + \_b + \_c + \_d;

for (int i = 0; i < 4; ++i) {

if (vec[i] > max) {

max = vec[i];

}

}

if (max < sum - max)

return true;

else return false;

}

public:

Trapezoid()

{

\_a = 12;

\_b = 27;

\_c = 15;

\_d = 15;

}

Trapezoid(float a, float b, float c, float d)

{

\_a = a;

\_b = b;

\_c = c;

\_d = d;

if (!Existens())

{

cout << "Трапеции не существует!";

exit(0);

}

}

float Area() override

{

return (\_a + \_b) \* HeightCalculation() \* 0.5;

}

float Perimeter() override

{

return \_a + \_b + \_c + \_d;

}

void InformatiomOutput() override

{

if (!Existens())

{

cout << "Трапеции не существует!";

exit(0);

}

cout << "Площадь трапеции: " << Area() << endl;

cout << "Периметр трапеции: " << Perimeter() << endl;

}

void SetA(float a)

{

\_a = a;

}

void SetB(float b)

{

\_b = b;

}

void SetC(float c)

{

\_c = c;

}

void SetD(float d)

{

\_d = d;

}

float GetA()

{

return \_a;

}

float GetB()

{

return \_b;

}

float GetC()

{

return \_c;

}

float GetD()

{

return \_d;

}

};

Программа, демонстрирующая работу класса:

#include <iostream>

#include "Figures.cpp"

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

cout << "Прямоугольник!" << endl;

Rectangle r;

cout << "Высота: " << r.GetHeight() << endl;

cout << "Ширина: " << r.GetWidth() << endl;

r.InformatiomOutput();

float width = 0;

float height = 0;

cout << "Введите новую ширину: ";

cin >> width;

cout << "Введите новую высоту: ";

cin >> height;

r.SetHeight(height);

r.SetWidth(width);

r.InformatiomOutput();

cout << endl;

cout << "Круг!" << endl;

Circle cc;

cout << "Радиус: " << cc.GetRadius() << endl;

cc.InformatiomOutput();

float radius = 0;

cout << "Введите новый радиус: ";

cin >> radius;

cc.SetRadius(radius);

cc.InformatiomOutput();

cout << endl;

cout << "Прямоугольный треугольник!" << endl;

RightTriangle rt;

cout << "А: " << rt.GetA() << endl;

cout << "B " << rt.GetB() << endl;

cout << "C: " << rt.GetC() << endl;

rt.InformatiomOutput();

float a = 0;

float b = 0;

float c = 0;

cout << "Введите A: ";

cin >> a;

cout << "Введите B: ";

cin >> b;

cout << "Введите C: ";

cin >> c;

rt.SetA(a);

rt.SetB(b);

rt.SetC(c);

rt.InformatiomOutput();

cout << endl;

cout << "Трапеция!" << endl;

Trapezoid t;

cout << "A: " << t.GetA() << endl;

cout << "B: " << t.GetB() << endl;

cout << "C: " << t.GetC() << endl;

cout << "D: " << t.GetD() << endl;

t.InformatiomOutput();

float A = 0;

float B = 0;

float C = 0;

float D = 0;

cout << "Введите A: ";

cin >> A;

cout << "Введите B: ";

cin >> B;

cout << "Введите C: ";

cin >> C;

cout << "Введите D: ";

cin >> D;

t.SetA(A);

t.SetB(B);

t.SetC(C);

t.SetD(D);

t.InformatiomOutput();

cout << endl;

Rectangle r2(width, height);

Circle cc2(radius);

RightTriangle rt2(a, b, c);

Trapezoid t2(A, B, C, D);

Figures\*\* figure = new Figures\*[1000];

figure[0] = &r2;

figure[1] = &cc2;

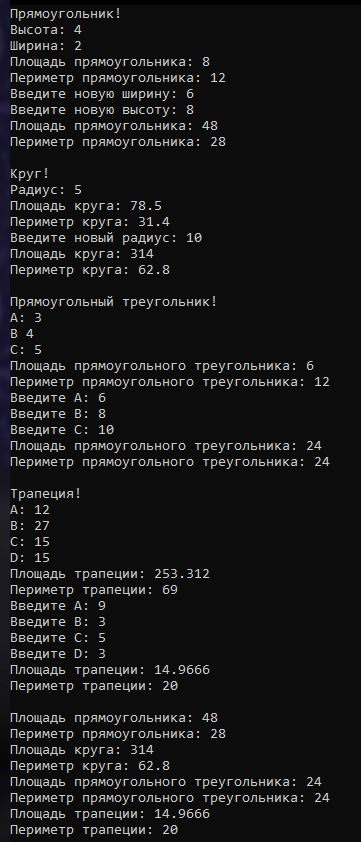
figure[2] = &rt2;

figure[3] = &t2;

for (int i = 0; i < 4; i++)

figure[i]->InformatiomOutput();

}



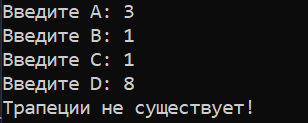
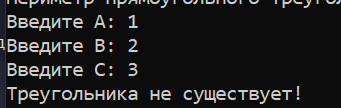


Рис. 1 – Работа программы

# Контрольные вопросы

1. Что понимается под термином «наследование»? Перечислите преимущества и недостатки наследования.

*На основе одних классов можно создавать новые классы, при этом новый класс называется производным или наследником, а класс основа — базовым или предком.*

*Преимущества:*

*- эффективное построение сложных иерархий классов с возможностью их удобной модификации.*

*- удобство в сопровождении (дополнении) программного кода путем введения новых классов с новыми возможностями;*

*- уменьшение количества логических ошибок при разработке сложных программных систем.*

*- возможность реализовывать известные шаблоны проектирования для построения гибкого кода, который не изменяет предыдущих разработок;*

*- использование преимуществ полиморфизма невозможно без наследования.*

*Недостатки:*

*- невозможно изменить унаследованную реализацию во время выполнения;*

*- низкая скорость выполнения.*

*- большая размерность программ благодаря использованию библиотек общего назначения.*

*- увеличение сложности программы в случае неправильного или неумелого использования наследования*

2. Что понимается под термином «виртуальный метод»? Для каких целей применяются виртуальные методы?

*Виртуальный метод — метод класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения.*

*Они позволяют создавать общий код, который может работать как с объектами базового класса, так и с объектами любого его класса-наследника. При этом базовый класс определяет способ работы с объектами, и любые его наследники могут предоставлять конкретную реализацию этого способа.*

**Вывод:** в ходе лабораторной работы ознакомились наследование и виртуальными методами в ООП. Разработали базовый класс и классы наследники на языке С++ и программу для демонстрации работы класса.