**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Механизмы наследования в С++

Студент: Хренникова Ангелина Сергеевна

Группа: M8O-208Б-19

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

**Содержание**

1. Постановка задачи
2. Описание программы
3. Набор тестов
4. Результаты выполнения тестов
5. Листинг программы
6. Вывод
7. Список литературы

**Постановка задачи**

Цель:

* Изучить механизмы работы с наследованием в С++.

Задача: Разработать классы фигур, представляющих трапецию, пятиугольник, ромб, классы которых должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры;

2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;

3. Вычисление площади фигуры.

**Описание программы**

Реализован базовый абстрактный класс, имеющие чисто виртуальные функции для вычисления площади, центра и ввода/вывода потоков. В конструкторах классов-наследников Trapeze и Rhombus реализованы проверки на корректность заданных точек, стоит отметить, что в конструктор точки подаются в любом порядке. Для пятиугольника задается центр фигуры и точка, остальные точки вычисляются программой. Площадь трапеции подсчитывается стандартной формулой (полусумма оснований \* высоту). Площадь ромба подсчитывается как полупроизведение длин диагоналей. Площадь пятиугольника рассчитывается как сумма площадей составляющих его треугольников. Центры всех фигур вычисляются как сумма координат всех составляющих точек по координатам х и у, деленная на количество всех точек.

Для удобства использования программы, создано меню с несколькими командами:

1. Создает новую фигуру заданного типа по введен точкам. Добавляет фигуру в конец вектора фигур.
2. Выводит фигуру, находящуюся в векторе по заданному индексу, а также ее площадь.
3. Выводит фигуру, находящуюся в векторе по заданному индексу, а также ее центр.
4. Выводит все точки фигуры, находящейся в векторе по заданному индексу.
5. Удаляет из вектора фигуру с заданным индексом.
6. Выводит количество фигур в векторе.
7. Вычисляет общую площадь всех фигур.
8. Завершает программу.

**Набор тестов**

Входными данными являются пункты меню и координаты вершин фигур.

Входные данные теста 1: 1 a 1 1 4 5 9 5 12 1 1 b 3 5 3 -1 1 2 5 2 1 c 0 0 0 6 1 a 0 0 2 2 4 2 6 0 2 3 3 1 4 4 6 5 4 6 7 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Индекс | Фигура | Вершины | Геометрический центр | Площадь |
| 1 | Трапеция | а1(1, 1), а2(4, 5),  а3(9, 5), а4(12, 1). | с(6.5, 3) | 32 |
| 2 | Ромб | а1(3, 5), а2(3, -1),  а3(1, 2), а4(5, 2). | c(3, 2) | 12 |
| 3 | Пятиугольник | a0(0, 0), а1(0, 6),  а2(-5.71, 1.85),  а3(5.71, 1.85),  а4(-3.53, -4.85),  а5(3.53, -4.85). | c(0, 0) | 100.289 |
| 4 | Трапеция | а1(0, 0), а2(2, 2), а3(4, 2), а4(6, 0). | c(3, 1) | 8 |

Входные данные теста 2: 1 a -1 -1 -4 -5 -9 -5 -12 -1 1 b 0 2 9 5 4 5 5 2 6 1 c 0 12 0 6 4 3 2 1 3 2 7 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Индекс | Фигура | Вершины | Геом.центр | Площадь |
| 1 | Трапеция | а1(-1, -1), а2(-4, -5),  а3(-9, -5), а4(-12, -1). | с(-6.5, -3) | 32 |
| 2 | Ромб | а1(0, 2), а2(9, 5),  а3(4, 5), а4(5, 2). | c(4.5, 3.5) | 15 |
| 3 | Пятиугольник | a0(0, 12), а1(0, 6),  а2(5.71, 10.15),  а3(-5.71, 10.15),  а4(-3.53, 16.85),  а5(3.53, 16.85). | c(0, 12) | 100.289 |

**Результаты выполнения тестов**

Тест1:

Menu:

1. Add figure.

a. Trapeze.

b. Rhombus.

c. Pentagon.

2. Print figure.

3. Figure area.

4. Center of the figure.

5. Delete figure.

6. Elements count.

7. Area all figures.

8. Exit.

Select menu item: 1

Figure: a

Enter the coordinates of the first point: 1 1

Enter the coordinates of the second point: 4 5

Enter the coordinates of the third point: 9 5

Enter the coordinates of the fourth point: 12 1

Created figure

Trapeze p1:1 1, p2:12 1, p3:4 5, p4:9 5

Select menu item: 1

Figure: b

Enter the coordinates of the first point: 3 5

Enter the coordinates of the second point: 3 -1

Enter the coordinates of the third point: 1 2

Enter the coordinates of the fourth point: 5 2

Created figure

Rhombus, p1: 3 5, p2: 1 2, p3: 3 -1, p4: 5 2

Select menu item: 1

Figure: c

Enter the coordinates of the center point: 0 0

Enter the coordinates of the first point: 0 6

Created figure

Pentagon, p1: 0 6, p2: -5.71 1.85, p3: 5.71 1.85, p4: -3.53 -4.85, p5: 3.53 -4.85

Select menu item: 1

Figure: a

Enter the coordinates of the first point: 0 0

Enter the coordinates of the second point: 2 2

Enter the coordinates of the third point: 4 2

Enter the coordinates of the fourth point: 6 0

Created figure

Trapeze p1:0 0, p2:2 2, p3:4 2, p4:6 0

Select menu item: 2

Enter index: 3

Figure at index 3 - Pentagon, p1: 0 6, p2: -5.71 1.85, p3: 5.71 1.85, p4: -3.53 -4.85, p5: 3.53 -4.85

Select menu item: 3

Enter index: 1

Trapeze p1:1 1, p2:12 1, p3:4 5, p4:9 5

Area: 32

Select menu item: 4

Enter index: 4

Trapeze p1:0 0, p2:2 2, p3:4 2, p4:6 0

Center: 3 1

Select menu item: 6

Elements count: 4

Select menu item: 5

Enter index: 4

Select menu item: 6

Elements count: 3

Select menu item: 7

Area all figures: 144.298

Select menu item: 8

Тест2:

Menu:

1. Add figure.

a. Trapeze.

b. Rhombus.

c. Pentagon.

2. Print figure.

3. Figure area.

4. Center of the figure.

5. Delete figure.

6. Elements count.

7. Area all figures.

8. Exit.

Select menu item: 1

Figure: a

Enter the coordinates of the first point: -1 -1

Enter the coordinates of the second point: -4 -5

Enter the coordinates of the third point: -9 -5

Enter the coordinates of the fourth point: -12 -1

Created figure

Trapeze p1:-1 -1, p2:-12 -1, p3:-4 -5, p4:-9 -5

Select menu item: 1

Figure: b

Enter the coordinates of the first point: 0 2

Enter the coordinates of the second point: 9 5

Enter the coordinates of the third point: 4 5

Enter the coordinates of the fourth point: 5 2

Created figure

Rhombus, p1: 0 2, p2: 4 5, p3: 9 5, p4: 5 2

Select menu item: 6

Error!Select menu item: Elements count: 2

Select menu item: 1

Figure: c

Enter the coordinates of the first point: 0 12

Enter the coordinates of the first point: 0 6

Created figure

Pentagon, p1: 0 6, p2: 5.71 10.15, p3: -5.71 10.15, p4: -3.53 16.85, p5: 3.53 16.85

Select menu item: 4

Enter index: 3

Pentagon, p1: 0 6, p2: 5.71 10.15, p3: -5.71 10.15, p4: -3.53 16.85, p5: 3.53 16.85

Center: 0 12

Select menu item: 2

Enter index: 1

Figure at index 1 - Trapeze p1:-1 -1, p2:-12 -1, p3:-4 -5, p4:-9 -5

Select menu item: 3

Enter index: 2

Rhombus, p1: 0 2, p2: 4 5, p3: 9 5, p4: 5 2

Area: 15

Select menu item: 7

Area all figures: 132.649

Select menu item: 8

**Листинг программы**

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ ls

CMakeCache.txt Figure.cpp Pentagon.cpp Rhombus.h Trapeze.h oop03

CMakeFiles Figure.h Pentagon.h Trapeze.cpp cmake\_install.cmake

CMakeLists.txt Makefile Rhombus.cpp Trapeze.cpp.save main.cpp

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat main.cpp

#include <iostream>

#include <vector>

#include <algorithm>

#include <exception>

#include "Figure.h"

#include "Trapeze.h"

#include "Rhombus.h"

#include "Pentagon.h"

void menu() {

std::cout << "Menu:\n";

std::cout << "1. Add figure.\n";

std::cout << "\ta. Trapeze.\n";

std::cout << "\tb. Rhombus.\n";

std::cout << "\tc. Pentagon.\n";

std::cout << "2. Print figure.\n";

std::cout << "3. Figure area.\n";

std::cout << "4. Center of the figure.\n";

std::cout << "5. Delete figure.\n";

std::cout << "6. Elements count.\n";

std::cout << "7. Area all figures.\n";

std::cout << "8. Exit.\n";

}

int main() {

menu();

std::vector<Figure\*> figures;

int command;

std::cout << "Select menu item: ";

while (std::cin >> command) {

if (command == 1) { //Ввод фигур

char fig\_type;

std::cout << "Figure: ";

std::cin >> fig\_type;

Figure\* new\_fig;

if (fig\_type == 'a') { //Трапеция

new\_fig = new Trapeze;

} else if (fig\_type == 'b') { //Ромб

new\_fig = new Rhombus;

} else if (fig\_type == 'c') { //Пятиугольник

new\_fig = new Pentagon;

} else {

std::cout << "Invalid figure type\n"; //Неверный тип

std::cin.ignore(30000, '\n');

continue;

}

try {

std::cin >> (\*new\_fig);

} catch (std::exception& e) {

std::cout << e.what() << "\n";

delete new\_fig;

continue;

}

figures.push\_back(new\_fig);

std::cout << "Created figure\n";

std::cout << \*new\_fig << "\n";

} else if (command == 2) { //Печать фигуры

int index;

std::cout << "Enter index: ";

std::cin >> index;

index--;

if (index < 0 || index >= figures.size()) {

std::cout << "No object at that index\n";

continue;

}

std::cout << "Figure at index " << index + 1 << " - " << \*figures[index] << "\n";

} else if (command == 3) { //Вычисление площади фигуры

int index;

std::cout << "Enter index: ";

std::cin >> index;

index--;

if (index < 0 || index >= figures.size()) {

std::cout << "No object at that index\n";

continue;

}

std::cout << \*figures[index] << "\n";

std::cout << "Area: " << figures[index]->Area() << "\n";

} else if (command == 4) { //Вычисление центра фигуры

int index;

std::cout << "Enter index: ";

std::cin >> index;

index--;

if (index < 0 || index >= figures.size()) {

std::cout << "No object at that index\n";

continue;

}

std::cout << \*figures[index] << "\n";

std::cout << "Center: " << figures[index]->Center() << "\n";

} else if (command == 5) { //Удаление фигуры

int index;

std::cout << "Enter index: ";

std::cin >> index;

index--;

if (index < 0 || index >= figures.size()) {

std::cout << "No object at that index\n";

continue;

}

delete figures[index];

figures.erase(figures.begin() + index);

} else if (command == 6) { //Размер списка(количество фигур)

std::cout << "Elements count: " << figures.size() << "\n";

} else if (command == 7) {

int index;

double s = 0;

for (index = 0; index < figures.size(); ++index) {

s = s + figures[index]->Area();

}

std::cout << "Area all figures: " << s << "\n";

} else if (command == 8)

break;

else std::cout << "Error!";

std::cout << "Select menu item: ";

}

for (Figure\* ptr : figures) {

delete ptr;

}

return 0;

}

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Figure.cpp

#include "Figure.h"

//Перегрузки операторов и функции для структуры точки

Point operator + (Point lhs, Point rhs) {

return {lhs.x + rhs.x, lhs.y + rhs.y};

}

Point operator - (Point lhs, Point rhs) {

return {lhs.x - rhs.x, lhs.y - rhs.y};

}

Point operator / (Point lhs, double a) {

return { lhs.x / a, lhs.y / a};

}

Point operator \* (Point lhs, double a) {

return {lhs.x \* a, lhs.y \* a};

}

bool operator < (Point lhs, Point rhs) {

return (lhs.x \* lhs.x + lhs.y \* lhs.y) < (lhs.x \* lhs.x + lhs.y \* lhs.y);

}

//операторы и функции для класса векторов

double operator \* (Vector lhs, Vector rhs) {

return lhs.x \* rhs.x + lhs.y \* rhs.y;

}

bool is\_parallel(const Vector& lhs, const Vector& rhs) { //Паралельность

return (lhs.x \* rhs.y - lhs.y \* rhs.y) == 0;

}

bool Vector::operator == (Vector rhs) {

return

std::abs(x - rhs.x) < std::numeric\_limits<double>::epsilon() \* 100

&& std::abs(y - rhs.y) < std::numeric\_limits<double>::epsilon() \* 100;

}

double Vector::length() const { //Длина

return sqrt(x\*x + y\*y);

}

Vector::Vector(double a, double b) //Координаты

: x(a), y(b) {

}

Vector::Vector(Point a, Point b) //Стороны(отрезки)

: x(b.x - a.x), y(b.y - a.y){

}

Vector Vector::operator - () {

return Vector(-x, -y);

}

bool is\_perpendecular(const Vector& lhs, const Vector& rhs) { //Перпендикуляр

return (lhs \* rhs) == 0;

}

double point\_and\_line\_distance(Point p1, Point p2, Point p3) { //Расстояние от точки до прямой

double A = p2.y - p3.y;

double B = p3.x - p2.x;

double C = p2.x\*p3.y - p3.x\*p2.y;

return (std::abs(A\*p1.x + B\*p1.y + C) / std::sqrt(A\*A + B\*B));

}

std::ostream& operator << (std::ostream& os, const Point& p) { //Печать точки

return os << p.x << " " << p.y;

}

std::istream& operator >> (std::istream& is, Point& p) { //Считывание точки

return is >> p.x >> p.y;

}

std::ostream& operator << (std::ostream& os, const Figure& fig) { //Печать фигуры

fig.Print(os);

return os;

}

std::istream& operator >> (std::istream& is, Figure& fig) { //Считывание фигуры

fig.Scan(is);

return is;

}

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Figure.h

#pragma once

#include <numeric>

#include <iostream>

#include <vector>

#include <cmath>

#include <limits>

struct Point { //Структура точки

double x = 0; //Координата х

double y = 0; //Координата у

};

//Перегрузки операторов

std::ostream& operator << (std::ostream& os, const Point& p);

std::istream& operator >> (std::istream& is, Point& p);

Point operator + (Point lhs, Point rhs);

Point operator - (Point lhs, Point rhs);

Point operator / (Point lhs, double a);

Point operator \* (Point lhs, double a);

class Vector { //Класс вектор

public:

explicit Vector(double a, double b); //Точки

explicit Vector(Point a, Point b); //Отрезки

bool operator == (Vector rhs);

Vector operator - ();

friend double operator \* (Vector lhs, Vector rhs);

double length() const; //Длина

double x; //Координата х

double y; //Координата у

};

bool is\_parallel(const Vector& lhs, const Vector& rhs); //Параллельные отрезки

bool is\_perpendecular(const Vector& lhs, const Vector& rhs); //Перпендикулярные отрезки

double point\_and\_line\_distance(Point p1, Point p2, Point p3); //Расстояние от точки до прямой

class Figure { //Класс фигур

public:

virtual Point Center() const = 0; //Геометрический центр

virtual double Area() const = 0; //Площадь

virtual void Print(std::ostream& os) const = 0; //Вывод точек

virtual void Scan(std::istream& is) = 0; //Считывание точек

virtual ~Figure() = default;

};

std::ostream& operator << (std::ostream& os, const Figure& fig);

std::istream& operator >> (std::istream& is, Figure& fig);

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Trapeze.cpp

#include "Trapeze.h"

Trapeze::Trapeze(Point p1, Point p2, Point p3, Point p4) //Конструктор класса трапеции. Проверка на корректность ввода точек

: p1\_(p1), p2\_(p2), p3\_(p3), p4\_(p4){

Vector v1(p1\_, p2\_), v2(p3\_, p4\_);

if (v1 = Vector(p1\_, p2\_), v2 = Vector(p3\_, p4\_), is\_parallel(v1, v2)) {

if (v1 \* v2 < 0) {

std::swap(p3\_, p4\_);

}

} else if (v1 = Vector(p1\_, p3\_), v2 = Vector(p2\_, p4\_), is\_parallel(v1, v2)) {

if (v1 \* v2 < 0) {

std::swap(p2\_, p4\_);

}

std::swap(p2\_, p3\_);

} else if (v1 = Vector(p1\_, p4\_), v2 = Vector(p2\_, p3\_), is\_parallel(v1, v2)) {

if (v1 \* v2 < 0) {

std::swap(p2\_, p3\_);

}

std::swap(p2\_, p4\_);

std::swap(p3\_, p4\_);

} else {

throw std::logic\_error("At least 2 sides of trapeze must be parallel");

}

}

Point Trapeze::Center() const { //Геометрический центр

return (p1\_ + p2\_ + p3\_ + p4\_) / 4;

}

double Trapeze::Area() const { //Площадь

double height = point\_and\_line\_distance(p1\_, p3\_, p4\_);

return (Vector(p1\_, p2\_).length() + Vector(p3\_, p4\_).length()) \* height / 2;

}

void Trapeze::Print(std::ostream& os) const { //Вывод точек

os << "Trapeze p1:" << p1\_ << ", p2:" << p2\_ << ", p3:" << p3\_ << ", p4:" << p4\_;

}

void Trapeze::Scan(std::istream &is) { //Считывание точек

Point p1,p2,p3,p4;

std::cout << "Enter the coordinates of the first point: ";

is >> p1;

std::cout << "Enter the coordinates of the second point: ";

is >> p2;

std::cout << "Enter the coordinates of the third point: ";

is >> p3;

std::cout << "Enter the coordinates of the fourth point: ";

is >> p4;

\*this = Trapeze(p1,p2,p3,p4);

}

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Trapeze.h

#pragma once

#include "Figure.h"

#include <exception>

class Trapeze : public Figure { //Класс трапеции

public:

Trapeze() = default;

Trapeze(Point p1, Point p2, Point p3, Point p4); //Точки

Point Center() const override; //Центр трапеции

double Area() const override; //Вычисляет площадь

void Print(std::ostream& os) const override; //Печать трапеции

void Scan(std::istream& is) override; //Считывание трапеции

private:

Point p1\_, p2\_, p3\_, p4\_; //Четыре точки

};

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Rhombus.cpp

#include "Rhombus.h"

Rhombus::Rhombus(Point p1, Point p2, Point p3, Point p4) //Конструктор класса ромб. Проверка на корректность введеных точек

: p1\_(p1), p2\_(p2), p3\_(p3), p4\_(p4) {

if (Vector(p1\_, p2\_).length() == Vector(p1\_, p4\_).length()

&& Vector(p3\_, p4\_).length() == Vector(p2\_, p3\_).length()

&& Vector(p1\_, p2\_).length() == Vector(p2\_, p3\_).length()) {

} else if (Vector(p1\_, p4\_).length() == Vector(p1\_, p3\_).length()

&& Vector(p2\_, p3\_).length() == Vector(p2\_, p4\_).length()

&& Vector(p1\_, p4\_).length() == Vector(p2\_, p4\_).length()) {

std::swap(p2\_, p3\_);

} else if (Vector(p1\_, p3\_).length() == Vector(p1\_, p2\_).length()

&& Vector(p2\_, p4\_).length() == Vector(p3\_, p4\_).length()

&& Vector(p1\_, p2\_).length() == Vector(p2\_, p4\_).length()) {

std::swap(p3\_, p4\_);

} else {

throw std::logic\_error("This is not rhombus, sides arent equal");

}

}

double Rhombus::Area() const { //Площадь ромба

return Vector(p1\_, p3\_).length() \* Vector(p2\_, p4\_).length() / 2;

}

Point Rhombus::Center() const { //Геометрический центр ромба

return (p1\_ + p3\_) / 2;

}

void Rhombus::Print(std::ostream& os) const { //Печать точек

os << "Rhombus, p1: " << p1\_ << ", p2: " << p2\_ << ", p3: " << p3\_ << ", p4: " << p4\_;

}

void Rhombus::Scan(std::istream &is) { //Считывание точек

Point p1,p2,p3,p4;

std::cout << "Enter the coordinates of the first point: ";

is >> p1;

std::cout << "Enter the coordinates of the second point: ";

is >> p2;

std::cout << "Enter the coordinates of the third point: ";

is >> p3;

std::cout << "Enter the coordinates of the fourth point: ";

is >> p4;

\*this = Rhombus(p1,p2,p3,p4);

}

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Rhombus.h

#pragma once

#include "Figure.h"

class Rhombus : public Figure {

public:

Rhombus() = default;

Rhombus(Point p1\_, Point p2\_, Point p3\_, Point p4\_); //Точки

Point Center() const override; //Геометрический центр ромба

double Area() const override; //Площадь трапеции

void Print(std::ostream& os) const override; //Печать трапеции

void Scan(std::istream& is) override; //Ввод трапеции

private:

Point p1\_, p2\_, p3\_, p4\_; //Четыре точки

};

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Pentagon.cpp

#include "Pentagon.h"

Pentagon::Pentagon(const Point& p0, const Point& p1, const Point& p2, const Point& p3, const Point& p4, const Point& p5) //Конструктор класса пятиугольника

: p0\_(p0), p1\_(p1), p2\_(p2), p3\_(p3), p4\_(p4), p5\_(p5) {}

double Pentagon::Area() const { //Площадь

return

point\_and\_line\_distance(p1\_, p2\_, p3\_) \* Vector(p2\_, p3\_).length() / 2

+ point\_and\_line\_distance(p1\_, p3\_, p4\_) \* Vector(p3\_, p4\_).length() / 2

+ point\_and\_line\_distance(p1\_, p4\_, p5\_) \* Vector(p4\_, p5\_).length() / 2;

}

Point Pentagon::Center() const { //Геометрический центр

return (p1\_ + p2\_ + p3\_ + p4\_ + p5\_) / 5;

}

void Pentagon::Print(std::ostream& os) const { //Печать пятиугольника

os << "Pentagon, p1: " << p1\_ << ", p2: " << p2\_ << ", p3: " << p3\_ << ", p4: " << p4\_ << ", p5: " << p5\_;

}

void Pentagon::Scan(std::istream &is) { //Считывание пятиугольника

Point p0, p1, p2, p3, p4, p5;

std::cout << "Enter the coordinates of the center point: ";

is >> p0;

std::cout << "Enter the coordinates of the first point: ";

is >> p1;

double r = Vector(p0, p1).length();

double a = (sqrt(5) - 1) / 4;

double b = (5 + sqrt(5)) / 8;

p2.x = p0.x + p1.x \* a - p1.y \* b;

p2.y = p0.y + p1.x \*b + p1.y \* a;

p3.x = p0.x + p2.x \* a - p2.y \* b;

p3.y = p0.y + p2.x \*b + p2.y \* a;

p4.x = p0.x + p3.x \* a - p3.y \* b;

p4.y = p0.y + p3.x \*b + p3.y \* a;

p5.x = p0.x + p4.x \* a - p4.y \* b;

p5.y = p0.y + p4.x \*b + p4.y \* a;

\*this = Pentagon(p1,p2,p3,p4,p5);

}

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat Pentagon.h

#pragma once

#include "Figure.h"

class Pentagon : public Figure { //Класс пятиугольника

public:

Pentagon() = default;

explicit Pentagon(const Point& p0, const Point& p1, const Point& p2, const Point& p3, const Point& p4, const Point& p5); //Точки

Point Center() const override; //Геометрический центр

double Area() const override; //Площадь

void Print(std::ostream& os) const override; //Печать пятиугольника

void Scan(std::istream& is) override; //Считывание пятиугольника

private:

Point p0\_, p1\_, p2\_, p3\_, p4\_, p5\_; //Пять точек

};

lina\_tucha@LAPTOP-44CRFC1U:~/labs/oop/lab3$ cat CMakeLists.txt

cmake\_minimum\_required(VERSION 2.8)

project(oop03)

add\_executable(oop03 main.cpp Figure.cpp Trapeze.cpp Rhombus.cpp Pentagon.cpp)

**Вывод**

В результате выполнения данной лабораторной работы я научилась создавать базовые абстрактные классы и их классы наследники на основе класса фигур и наследников:трапеции, ромба и пятиугольника.

**Список источников**

1. Ravesli - Чистые виртуальные функции, Интерфейсы и Абстрактные классы/Электронный диплом/Режим доступа: <https://ravesli.com/urok-168-chistye-virtualnye-funktsii-interfejsy-i-abstraktnye-klassy/> (дата обращения: 24.10.2020).
2. Р. Лафоре, Объектно - ориентированное программирование в С++, 4-е издание. -М.: Питер, 2004 - 924 с. (дата обращения: 24.10.2020).
3. CppStudio - Наследование классов/Электронный ресурс/Режим доступа: <http://cppstudio.com/post/10103/> (дата обращения: 25.10.2020).