Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программрование»

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №3

Тема: Наследование, полиморфизм

Студент:	Колесова М. И.
Группа:	М8О-206Б-18
Руководитель:	Журавлев А.А.
Оценка:	
Дата:	

Москва, 2019

1 – Задание

Разработать классы triangle, octagon, square которые должны наследоваться от базового класса figure. 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.

2 - Репозиторий GitHub

https://github.com/KolesovaMariya/oop_exercise_03

3 - Описание программы

Создан базовый абстрактный класс figure с виртуальными функциями, вычисляющие площадь, геометрический центр, а также вывод и ввод. Наследуемые классы triangle, square, octagon. В каждом классе реализованы функции center, area, print. Для удобства создан вспомогательный класс point, с помощью которого задаются точки. В классах triangle и square реализованы функции проверки того, являются ли фигуры треугольником или квадратом, так же возможен ввод точек в любом порядке, в то время вершины восьмиугольника необходимо вводить строго по порядку. Так же написано простое пользовательское меню со следующими командами:

exit - закончить работу программы;
add triangle/ square/ octagon - добавить фигуру
delete i - удаление по индексу i
areas - выводит площади всех введённых фигур
centers - выводит центры всех введенных фигур
figures - выводит все введенные фигуры
print - выводит все фигуры с их площадями и центрами
destroy – удаляет все фигуры

Все добавленные фигуры помещаются в вектор с элементами типа figure.
Как производились вычисления:

- площадь квадрата вычисляется возведением в квадрат одной из сторон
- площадь треугольника вычисляется по формуле Герона
- площадь восьмиугольника вычисляется с помощью разбиения фигуры на треугольники, и вычисление уже их площади
- геометрический центр вычисляется по формуле $x_c = x_1 + x_{2 +\dots} + x_n$,
- $y_c = y_1 + y_2 + \dots + y_n$, где n – кол-во вершин в фигуре.

4 – Тесты

test_00.txt

```
add triangle 2 1 7 4 10 2
add triangle 4 3 7 4 6 5
add triangle 0 0 0 0 0 0
add triangle 8 8 8 8 8 8
add triangle 0.5 -0.5 0.75 4.25 2 28
add triangle 3 6 4 4.5 5 3
add triangle 0 0 1 2 2 4
print
delete 1
figures
destroy
figure
exit
```

result 00

```
The figure Triangle [(2,1),(7,4),(10,2)] was successfully added;
The figure Triangle [(4,3),(7,4),(6,5)] was successfully added;
The figure Triangle [(0,0),(0,0),(0,0)] was successfully added;
The figure Triangle [(8,8),(8,8),(8,8)] was successfully added;
The figure Triangle [(0.5,-0.5),(0.75,4.25),(2,28)] isn't a triangle, square or octagon;
The figure Triangle [(3,6),(4,4.5),(5,3)] isn't a triangle, square or octagon;
The figure Triangle [(0,0),(1,2),(2,4)] isn't a triangle, square or octagon;
Triangle [(2,1),(7,4),(10,2)]; Center: (6.33333,2.33333); Area: 9.5;
Triangle [(4,3),(7,4),(6,5)]; Center: (5.66667,4); Area: 2;
Triangle [(0,0),(0,0),(0,0)]; Center: (0,0); Area: 0;
Triangle [(8,8),(8,8),(8,8)]; Center: (8,8); Area: 0;
Deleted by index 1;
Figures:
Triangle [(2,1),(7,4),(10,2)]
Triangle [(0,0),(0,0),(0,0)]
Triangle [(8,8),(8,8),(8,8)]
All figures are destroyed;

Process finished with exit code 0
```

test_01.txt

```
add square 0 0 0 4 4 0 4 4
add square 4 1 2 3 4 5 6 3
add square 2 4 3 2 5 3 4 5
add square 2 2 -2 2 2 -2 -2 -2
add square 0 0 0 0 0 0 0 0
add square 2 3 2 4 2 5 4 4
print
```

delete 1
figures
destroy
figure
exit

result 01

The figure Square: [(0,0)(0,4)(4,4)(4,0)] was successfully added;
The figure Square: [(2,3)(4,1)(6,3)(4,5)] was successfully added;
The figure Square: [(2,4)(3,2)(5,3)(4,5)] was successfully added;
The figure Square: [(-2,-2)(-2,2)(2,2)(2,-2)] was successfully added;
The figure Square: [(0,0)(0,0)(0,0)(0,0)] was successfully added;
The figure Square: [(2,3)(2,4)(4,4)(2,5)] isn't a triangle, square or octagon;
Square: [(0,0)(0,4)(4,4)(4,0)]; Center: (2,2); Area: 16;
Square: [(2,3)(4,1)(6,3)(4,5)]; Center: (4,3); Area: 8;
Square: [(2,4)(3,2)(5,3)(4,5)]; Center: (3.5,3.5); Area: 5;
Square: [(-2,-2)(-2,2)(2,2)(2,-2)]; Center: (0,0); Area: 16;
Square: [(0,0)(0,0)(0,0)(0,0)]; Center: (0,0); Area: 0;
Deleted by index 1;
Figures:
Square: [(0,0)(0,4)(4,4)(4,0)]
Square: [(2,4)(3,2)(5,3)(4,5)]
Square: [(-2,-2)(-2,2)(2,2)(2,-2)]
Square: [(0,0)(0,0)(0,0)(0,0)]
All figures are destroyed;

Process finished with exit code 0

tesr_02.txt

add octagon 2 4 4 4 6 3 6 1 4 0 2 0 0 1 0 3
add octagon 5 6 4 5 3 5 2 6 2 7 3 8 4 8 5 7
add octagon 1 4 0 6 1 10 4 8 6 5 7 3 6 0 3 1
add octagon -4 8 -7 4 -5 0 -1 -3 3 -2 7 0 9 7 6 10
add octagon 4 6 3 7 1 4 3 4 3 2 5 3 6 5 5 7
add octagon 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
add octagon 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5
print
delete 1
figures
destroy
figure
exit

result 02

The figure Octagon [(2,4), (4,4), (6,3), (6,1), (4,0), (2,0), (0,1), (0,3)] was successfully added;
The figure Octagon [(5,6), (4,5), (3,5), (2,6), (2,7), (3,8), (4,8), (5,7)] was successfully added;
The figure Octagon [(1,4), (0,6), (1,10), (4,8), (6,5), (7,3), (6,0), (3,1)] was successfully added;
The figure Octagon [(-4,8), (-7,4), (-5,0), (-1,-3), (3,-2), (7,0), (9,7), (6,10)] was successfully added;

```
The figure Octagon [(4,6), (3,7), (1,4), (3,4), (3,2), (5,3), (6,5), (5,7)] was successfully added;
The figure Octagon [(0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0)] was successfully added;
The figure Octagon [(2,2), (2,2), (3,3), (3,3), (4,4), (4,4), (5,5), (5,5)] isn't a triangle, square or
octagon;
Octagon [(2,4), (4,4), (6,3), (6,1), (4,0), (2,0), (0,1), (0,3)]; Center: (3,2); Area: 20;
Octagon [(5,6), (4,5), (3,5), (2,6), (2,7), (3,8), (4,8), (5,7)]; Center: (3.5,6.5); Area: 7;
Octagon [(1,4), (0,6), (1,10), (4,8), (6,5), (7,3), (6,0), (3,1)]; Center: (3.5,4.625); Area: 39;
Octagon [(-4,8), (-7,4), (-5,0), (-1,-3), (3,-2), (7,0), (9,7), (6,10)]; Center: (1,3); Area: 142.5;
Octagon [(4,6), (3,7), (1,4), (3,4), (3,2), (5,3), (6,5), (5,7)]; Center: (3.75,4.75); Area: 13;
Octagon [(0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0)]; Center: (0,0); Area: 0;
Deleted by index 1;
Figures:
Octagon [(2,4), (4,4), (6,3), (6,1), (4,0), (2,0), (0,1), (0,3)]
Octagon [(1,4), (0,6), (1,10), (4,8), (6,5), (7,3), (6,0), (3,1)]
Octagon [(-4,8), (-7,4), (-5,0), (-1,-3), (3,-2), (7,0), (9,7), (6,10)]
Octagon [(4,6), (3,7), (1,4), (3,4), (3,2), (5,3), (6,5), (5,7)]
Octagon [(0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0)]
All figures are destroyed;

Process finished with exit code 0
```

6 - Код программы

point.h

```
#ifndef LAB3_POINT_H
#define LAB3_POINT_H

#include <iostream>
#include <cmath>


struct point {
   double x;
   double y;
};



std::ostream &operator<<(std::ostream &out, point const &a);

std::istream &operator>>(std::istream &in, point &a);
bool operator==(point a, point b);
bool operator>(point a, point b);
void swap(point &a, point &b);
double dist(point a, point b);


struct vec {
   double x;
   double y;
```

```
};
vec vector(point a, point b);
bool same_line(point a, point b, point c);


#endif //LAB3_POINT_H
```

point.cpp

```
#include "point.h"

std::ostream  &operator<< (std::ostream &out, point const &a) {
    out << "(" << a.x << "," << a.y << ")";
    return out;
}

std::istream &operator>>(std::istream &in, point &a) {
    in >> a.x >> a.y;
    return in;
}
bool operator>(point a, point b) {
    if (a.x > b.x) {
        return true;
    } else if ( a.x == b.x) {
        return a.y > b.y;
    } else {
        return false;
    }
}

void swap(point &a, point &b) {
    point tmp = a;
    a = b;
    b = tmp;
}

bool operator==(point a, point b) {
    return((a.x == b.x) && (a.y == b.y));
}

double dist(point a, point b) {
    return sqrt(pow((a.x - b.x),2) + pow((a.y - b.y),2));
}
vec vector(point a, point b) {
    vec v{};
    v.x = b.x - a.x;
    v.y = b.y - a.y;
    return v;
}
```

```
bool same_line(point a, point b, point c) {
   vec ab = vector(a, b);
   vec bc = vector(b, c);
   return ((ab.x) / (bc.x) != (ab.y) / (bc.y));
}
```

figure.h

```
#ifndef LAB3_FIGURE_H
#define LAB3_FIGURE_H

#include "point.h"
#include<iostream>

class figure {
public:
   virtual point center()  const = 0;
   virtual double area() const = 0;
   virtual void print(std::ostream&) const = 0;

   virtual bool is_figure() const = 0;
   virtual ~figure() = default;
};



#endif //LAB3_FIGURE_H
```

figure.cpp

```
#include <iostream>
#include "figure.h"

std::ostream& operator<< (std::ostream& os, const figure& f) {
   f.print(os);
   return os;
}
```

triangle.h

```
#ifndef LAB3_TRIANGLE_H
#define LAB3_TRIANGLE_H

#include"figure.h"

class triangle : public figure {
public:
   triangle() = default;
```

```
    triangle(const point &a, const point &b, const point &c);

    triangle(std::istream &is);

    point center() const override;

    double area() const override;

    void print(std::ostream &os) const override;
    bool is_figure() const;

private:
    point p1{};
    point p2{};
    point p3{};

};
#endif //LAB{}3_TRIANGLE_H
```

triangle.cpp

```
#include "triangle.h"
#include <iostream>
#include <cassert>
triangle::triangle(const point& p1_, const point& p2_, const point& p3_): p1(p1_), p2(p2_),
p3(p3_) {}

triangle::triangle(std::istream &is) {
    is >> p1 >> p2 >> p3;
}

point triangle::center() const {
    point cen{};
    cen.x = (p1.x + p2.x + p3.x)/3;
    cen.y = (p1.y + p2.y + p3.y)/3;
    return cen;
}

double triangle::area() const {
    double p = (dist(p1, p2) + dist(p1, p3) + dist(p2, p3))/2;
    return sqrt(p * (p - dist(p1, p2) ) * (p - dist(p1, p3) ) * (p - dist(p2, p3) ));
}

void triangle::print(std::ostream &os) const {
    os << "Triangle [" << p1 << ","<< p2 << ","<< p3 << "]";
}

bool triangle::is_figure() const {
    vec p1_p2 = vector(p1, p2);
    vec p1_p3 = vector(p1, p3);
    return ((p1_p2.x) / (p1_p3.x) != (p1_p2.y) / (p1_p3.y)) || (p1 == p3 ) || (p1 == p2) || (p2 ==
```

```
p3);
}
```

square.h

```
#ifndef LAB3_SQUARE_H
#define LAB3_SQUARE_H

#include "figure.h"

class square : public figure {
public:
   square() = default;
   square(point p1_, point p2_, point p3_, point p4_);
   square(std::istream &in);
   double area() const override;
   point center() const override;
   void print(std::ostream&) const override ;
   bool is_figure() const override;
   void sort();
private:
    point p1{}, p2{}, p3{}, p4{};
};


#endif //LAB3_SQUARE_H
```

square.cpp

```
#include "square.h"
#include<iostream>
#include<cmath>
square::square(point p1_, point p2_, point p3_, point p4_): p1(p1_), p2(p2_), p3(p3_), p4(p4_) {
   this->sort();
}

square::square(std::istream &is) {
   is >> p1 >> p2 >> p3 >> p4;
   this->sort();
}
double square::area() const {
   return pow(dist(p1, p2), 2);
}
point square::center() const {
   vec p1_p3 = vector(p1, p3);
   point cen = {p1.x + (p1_p3.x)/2, p1.y + (p1_p3.y)/2};
   return cen;
}

void square::print(std::ostream &out) const {
```

```
    out << "Square: [" << p1 << p2 << p3 << p4 << "]";
}

void square::sort() {
    point a[4] = {p1, p2, p3, p4};
    for(int i = 0 ;i < 3; i++) {
        for(int j = 0; j < 3; j++)
            if(a[j] > a[j+1]) {
                swap(a[j], a[j+1]);
            }
    }
    p1 = a[0];
    p2 = a[1];
    p3 = a[3];
    p4 = a[2];
}
bool square::is_figure() const {
    vec p1_p2 = vector(p1, p2);
    vec p2_p3 = vector(p2, p3);
    vec p3_p4 = vector(p3, p4);
    vec p4_p1 = vector(p4, p1);
    double v_mlt_1 = p1_p2.x * p2_p3.x + p1_p2.y * p2_p3.y;
    double v_mlt_2 = p2_p3.x * p3_p4.x + p2_p3.y * p3_p4.y;
    double v_mlt_3 = p3_p4.x * p4_p1.x + p3_p4.y * p4_p1.y;
    double v_mlt_4 = p4_p1.x * p1_p2.x + p4_p1.y * p1_p2.y;
    return (v_mlt_1 == 0) && (v_mlt_2 == 0) && (v_mlt_3 == 0) && (v_mlt_4 == 0);
}
```

octagon.h

```
#ifndef LAB3_OCTAGON_H
#define LAB3_OCTAGON_H
#include"figure.h"

class octagon : public figure {
public:
    octagon() = default;

    octagon(const point &p1_, const point &p2_,
            const point &p3_, const point &p4_,
            const point &p5_, const point &p6_,
            const point &p7_, const point &p8_);

    octagon(std::istream &is);

    point center() const override;

    double area() const override;

    void print(std::ostream &os) const override;
```

```
    bool is_figure() const override;

private:
    point p1{};
    point p2{};
    point p3{};
    point p4{};
    point p5{};
    point p6{};
    point p7{};
    point p8{};

};
#endif //LAB3_OCTAGON_H
```

octagon.cpp

```
#include "octagon.h"
#include "triangle.h"

octagon::octagon (const point &p1_, const point &p2_,
                  const point &p3_, const point &p4_,
                  const point &p5_, const point &p6_,
                  const point &p7_, const point &p8_):
                  p1(p1_), p2(p2_), p3(p3_), p4(p4_),
                  p5(p5_), p6(p6_), p7(p7_), p8(p8_) {}

octagon::octagon(std::istream &is) {
    is >> p1 >> p2 >> p3 >> p4 >> p5 >> p6 >> p7 >> p8;
}

point octagon::center() const {
    point cent{};
    cent.x = (p1.x + p2.x + p3.x + p4.x + p5.x + p6.x + p7.x + p8.x)/8;
    cent.y = (p1.y + p2.y + p3.y + p4.y + p5.y + p6.y + p7.y + p8.y)/8;
    return cent;
}

double octagon::area() const {
    point cent = this->center();
    point m[8] = {p1, p2 ,p3, p4, p5, p6, p7, p8};
    double ar = 0;
    for(int i = 0; i < 7; i++) {
        triangle a{m[i], m[i+1], cent};
        ar += a.area();
    }
    triangle a{p8, p1, cent};
    ar += a.area();
    return ar;
}
```

```
void octagon::print(std::ostream &os) const {
    os << "Octagon [" << p1 << ", "<< p2 << ", "<< p3 << ", "<< p4 << ", "<<
    p5 << ", "<< p6 << ", "<< p7 << ", "<< p8 << "]";
}
bool octagon::is_figure() const {
    bool flag = true;
    point a[8] = {p1, p2, p3, p4, p5, p6, p7, p8};
    for(int i = 0; i < 7; i++) {
        if(a[i] == a[i+1]) {
            flag = false;
            break;
        }
    }
    if (flag) return true;
    for(int i = 0; i < 7; i++) {
        if(!(a[i] == a[i+1])) {
            return false;
        }
    }
    return true;
}
```

main.cpp

```
#include <iostream>
#include <vector>
#include <fstream>

#include "triangle.h"
#include "square.h"
#include "octagon.h"

int main() {
    std::ifstream in(R"(C:\Users\LENOVO\CLionProjects\OOP\lab3\test_02.txt)");
    std::cin.rdbuf(in.rdbuf());
    std::string command;
    std::vector<figure*> figures;
    while(std::cin >> command) {
        bool flag = true;
        if (command == "exit") {
            break;
        } else if (command == "add") {
            std::string figure_type;
            std::cin >> figure_type;
            figure *ptr;
            if (figure_type == "triangle") {
                ptr = new triangle(std::cin);
            } else if (figure_type == "square") {
                ptr = new square(std::cin);
            } else if(figure_type == "octagon") {
```

```
            ptr = new octagon(std::cin);
        } else {
            std::cout << "Invalid input" << std::endl;
            flag = false;
        }
        if(flag && ptr -> is_figure()) {
            figures.push_back(ptr);
            std::cout << "The figure ";
            ptr->print(std::cout);
            std::cout << " was successfully added;" << std::endl;
        }
        if(!ptr->is_figure()) {
            std::cout <<"The figure ";
            ptr->print(std::cout);
            std::cout << " isn't a triangle, square or octagon;" << std::endl;
        }

    } else if (command == "delete") {
        int i;
        std::cin >> i;
        delete figures[i];
        figures.erase(figures.begin() + i);
        std::cout << "Deleted by index " << i << ";" <<  std::endl;
    } else if (command == "areas") {
        std::cout << "Areas " << std::endl;
        for(figure *elem : figures) {
            elem -> print(std::cout);
            std::cout << " Area: " <<  elem->area() << std::endl;
        }
    } else if (command == "centers") {
        std::cout << "Centers " << std::endl;
        for (figure *elem : figures) {
            elem -> print(std::cout);
            std::cout << " Center: " << elem->center() << std::endl;
        }
    } else if (command == "figures") {
        std::cout << "Figures: " << std::endl;
        for (figure *elem : figures) {
            elem->print(std::cout);
            std::cout << std::endl;
        }
    } else if(command == "print") {

        for (figure *elem : figures) {
            elem->print(std::cout);
            std::cout << "; Center: " << elem->center() << "; Area: " << elem->area() << ";" <<
std::endl;
        }
    } else if (command == "destroy") {
        while(!figures.empty()) {
```

```
            delete figures[0];
            figures.erase(figures.begin());
        }
        std::cout << "All figures are destroyed;" << std::endl;
    } else if (command == "help") {
        std::cout << "The program works with figures: triangle, square, octagon.\n "
                  "Available command: \n"
                  "exit - finish the program;\n"
                  "add triangle/ square/ octagon - add figure;\n"
                  "delete i - delete by index i;\n"
                  "areas - output the areas of all figures;\n"
                  "centers - output the centers of all figures;\n"
                  "figures - output all the figures;\n"
                  "print - output all the figures, their areas and centers;\n"
                  "destroy - destroy all the figures;\n";
    }
  }
}
```

7 – Вывод

Я научилась работать с базовыми абстрактными классами, делать наследуемые классы, а так же делать простейшее пользовательское меню.