

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа
по курсу «ООП»**

**Тема:
Наследование, полиморфизм.**

Студент:	Касимов М.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	6
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Figure.h:

```
#ifndef OOP_EXERCISE_03_FIGURE_H
#define OOP_EXERCISE_03_FIGURE_H
#include <iostream>
#include "point.h"

struct figure {
    virtual point center() const = 0;
    virtual void print(std::ostream&) const = 0 ;
    virtual double square() const = 0;
    virtual ~figure() = default;
};
#endif //OOP_EXERCISE_03_FIGURE_H
```

Hgon.cpp:

```
#include "hgon.h"

point hgon::center() const {
    double x,y;
    x = (a1.x + a2.x + a3.x + a4.x + a5.x + a6.x + a7.x + a8.x) / 8;
    y = (a1.y + a2.y + a3.y + a4.y + a5.y + a6.y + a7.y + a8.y) / 8;
    point p(x,y);
    return p;
}

void hgon::print(std::ostream& os) const {
    os << "coordinate:\n" << "{" << "\n" << a1 << "\n' << a2 << "\n' << a3 << "\n' << a4 << "\n'
    << a5 << "\n' << a6 << "\n" << a7 << "}" << "\n";
}

double hgon::square() const {
    return (-0.5) * ((a1.x*a2.y + a2.x*a3.y + a3.x*a4.y + a4.x*a5.y + a5.x*a6.y +
a6.x*a7.y + a7.x*a8.y + a8.x*a1.y) - ( a1.y*a2.x + a2.y*a3.x + a3.y*a4.x + a4.y*a5.x
+ a5.y*a6.x + a6.y*a7.x + a7.y*a8.x + a8.y*a1.x));
}

hgon::hgon(point p1, point p2, point p3, point p4, point p5, point p6, point p7,point
p8) {
    a1 = p1;
    a2 = p2;
    a3 = p3;
    a4 = p4;
    a5 = p5;
    a6 = p6;
    a7 = p7;
    a8 = p8;
```

```
}
```

Hgon.h:

```
//
```

```
// Created by magom on 20.10.2019.
```

```
//
```

```
#ifndef OOP_EXERCISE_03_8GON_H
```

```
#define OOP_EXERCISE_03_8GON_H
```

```
#include "figure.h"
```

```
struct hgon : figure{
```

```
private:
```

```
    point a1,a2,a3,a4,a5,a6,a7,a8;
```

```
public:
```

```
    point center() const override ;
```

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

```
    double square() const override ;
```

```
    hgon() = default;
```

```
    hgon(point p1,point p2,point p3,point p4,point p5,point p6,point p7,point p8);
```

```
};
```

```
#endif //OOP_EXERCISE_03_8GON_H
```

Hexagon.cpp:

```
#include "hexagon.h"
```

```
point hexagon::center() const {
```

```
    double x,y;
```

```
    x = (a1.x + a2.x + a3.x + a4.x + a5.x + a6.x) / 6;
```

```
    y = (a1.y + a2.y + a3.y + a4.y + a5.y + a6.y) / 6;
```

```
    point p(x,y);
```

```
    return p;
```

```
}
```

```
void hexagon::print(std::ostream& os) const {
```

```
    os << "coordinate:\n" << "{\n" << a1 << '\n' << a2 << '\n' << a3 << '\n' << a4 << '\n' << a5 << '\n' << a6 << "}\n";
```

```
}
```

```
double hexagon::square() const {
```

```
    return (-0.5) * ((a1.x*a2.y + a2.x*a3.y + a3.x*a4.y + a4.x*a5.y + a5.x*a6.y + a6.x*a1.y) - (a1.y*a2.x + a2.y*a3.x + a3.y*a4.x + a4.y*a5.x + a5.y*a6.x + a6.y*a1.x));
```

```
}
```

```

hexagon::hexagon(point p1, point p2, point p3, point p4, point p5, point p6) {
    a1 = p1;
    a2 = p2;
    a3 = p3;
    a4 = p4;
    a5 = p5;
    a6 = p6;
}

```

Hexagon.h:

```

#ifndef OOP_EXERCISE_03_HEPTAGON_H
#define OOP_EXERCISE_03_HEPTAGON_H

```

```

#include "figure.h"

```

```

struct heptagon : figure{
private:
    point a1,a2,a3,a4,a5,a6,a7;
public:
    point center() const override ;
    void print(std::ostream&) const override ;
    double square() const override ;
    heptagon() = default;
    heptagon(point p1,point p2,point p3,point p4,point p5,point p6,point p7);
};

```

```

#endif //OOP_EXERCISE_03_HEPTAGON_H

```

Pentagon.cpp:

```

#include "pentagon.h"
#include <cmath>
#include "point.h"

```

```

point pentagon::center() const {
    double x,y;
    x = (a1.x + a2.x + a3.x + a4.x + a5.x) / 5;
    y = (a1.y + a2.y + a3.y + a4.y + a5.y) / 5;
    point p(x,y);
    return p;
}

```

```

void pentagon::print(std::ostream& os) const {
    os << "coordinate:\n" << "{\n" << a1 << '\n' << a2 << '\n' << a3 << '\n' << a4 << '\n'
    << a5 << '\n' << "}\n";
}

```

```
double pentagon::square() const{
    //метод Гаусса(алгоритм шнурования)
    return (-0.5) * ((a1.x*a2.y + a2.x*a3.y + a3.x*a4.y + a4.x*a5.y + a5.x*a1.y) - (
a1.y*a2.x + a2.y*a3.x + a3.y*a4.x + a4.y*a5.x + a5.y*a1.x ));
}
```

```
pentagon::pentagon(point p1, point p2, point p3, point p4, point p5) {
    a1 = p1;
    a2 = p2;
    a3 = p3;
    a4 = p4;
    a5 = p5;
}
```

Pentagon.h:

```
#ifndef OOP_EXERCISE_03_PENTAGON_H
#define OOP_EXERCISE_03_PENTAGON_H
```

```
#include "figure.h"
```

```
struct pentagon : figure{
private:
    point a1,a2,a3,a4,a5;
public:
    point center() const override ;
    void print(std::ostream&) const override ;
    double square() const override ;
    //pentagon(std::istream&);
    pentagon() = default;
    pentagon(point p1,point p2,point p3,point p4,point p5);
};
```

```
#endif //OOP_EXERCISE_03_PENTAGON_H
```

Point.cpp:

```
#include "point.h"
```

```
std::istream& operator >> (std::istream& is,point& p ) {
    return is >> p.x >> p.y;
}
```

```
std::ostream& operator << (std::ostream& os,const point& p) {
    return os << p.x << ' ' << p.y;
}
```

Point.h:

```
#ifndef OOP_EXERCISE_03_POINT_H
#define OOP_EXERCISE_03_POINT_H
```

```

#include <iostream>

struct point {
    double x, y;
    point (double a, double b) { x = a, y = b; };
    point() = default;
};

std::istream& operator >> (std::istream& is, point& p );
std::ostream& operator << (std::ostream& os, const point& p);

#endif //OOP_EXERCISE_03_POINT_H
Main.cpp:
#include <iostream>
#include <vector>
#include "point.h"
#include "figure.h"
#include "pentagon.h"
#include "hexagon.h"
#include "hgon.h"

void menu() {
    std::cout << "1 - add\n"
                "2 - delete\n"
                "3 - call common functions for the entire array\n"
                "4 - total area\n"
                "5 - exit\n";
}

int choice() {
    int i;
    std::cin >> i;
    return i;
}

int main() {
    std::vector<figure*> v;
    int i, j;
    double S;
    menu();
    while ((i = choice()) != 5) {

```

```

switch (i)
{
    case 1:
        figure* f;
        std::cout << " 5 - pentagon\n 6 - hexagon\n 8 - 8gon\n";
        std::cin >> j;
        if ( j == 5 ) {
            point p1,p2,p3,p4,p5;
            std::cin >> p1 >> p2 >> p3 >> p4 >> p5;
            f = new pentagon(p1,p2,p3,p4,p5);
            v.push_back(f);
        }
        else if ( j == 6 ) {
            point p1,p2,p3,p4,p5,p6;
            std::cin >> p1 >> p2 >> p3 >> p4 >> p5 >> p6;
            f = new hexagon(p1,p2,p3,p4,p5,p6);
            v.push_back(f);
        }
        else if ( j == 8 ) {
            point p1,p2,p3,p4,p5,p6,p7,p8;
            std::cin >> p1 >> p2 >> p3 >> p4 >> p5 >> p6 >> p7 >> p8;
            f = new hgon(p1,p2,p3,p4,p5,p6,p7,p8);
            v.push_back(f);
        }
        break;
    case 2:
        std::cout << " enter index\n";
        std::cin >> j;
        if ( j >= v.size() )
            break;
        delete v[j];
        for (; j < v.size() - 1; ++j) {
            v[j] = v[j + 1];
        }
        v.pop_back();
        break;
    case 3:
        for (auto elem: v) {
            elem->print(std::cout);
            std::cout << elem->center() << std::endl;
            std::cout << elem->square() << std::endl;
        }
        break;
    case 4:
        S = 0;

```

```

        for (auto elem: v) {
            S+=elem->square();
        }
        std::cout << S << std::endl;
        break;
    default:
        std::cout << " error\n";
        break;
    }
}
return 0;
}

```

2. Ссылка на репозиторий на GitHub.

https://github.com/magomed2000kasimov/oop_exercise_03

3. Набор тестов.

test_01.test:

```

1
5
1 1
2 5
4 7
5 4
4 2
1
6
-1 2
-2 4
-1 6
1 6
2 4
1 2
1
8
1 3
0.7 4.7
2.3 6
4 6
5 4
4 2
3 1
2 2
3
4

```


5
test_02.test:

1
5
1 1
2 5
4 7
5 4
4 2
1
6
-1 2
-2 4
-1 6
1 6
2 4
1 2
1
8
1 3
0.7 4.7
2.3 6
4 6
5 4
4 2
3 1
2 2
4
2
0
3
4
5

4. Результаты выполнения тестов.

test_01.result:

1 - add
2 - delete
3 - call common functions for the entire array
4 - total area
5 - exit
5 - pentagon
6 - hexagon
8 - 8gon
5 - pentagon

6 - hexagon
8 - 8gon
5 - pentagon
6 - hexagon
8 - 8gon

coordinate:

{
1 1
2 5
4 7
5 4
4 2
}
3.2 3.8
13

coordinate:

{
-1 2
-2 4
-1 6
1 6
2 4
1 2}
0 4
12

coordinate:

{
1 3
0.7 4.7
2.3 6
4 6
5 4
4 2
3 1}
2.75 3.5875
14.105
39.105

test_02.result:

1 - add
2 - delete
3 - call common functions for the entire array
4 - total area
5 - exit
5 - pentagon
6 - hexagon

```
8 - 8gon
5 - pentagon
6 - hexagon
8 - 8gon
5 - pentagon
6 - hexagon
8 - 8gon
39.105
enter index
coordinate:
{
-1 2
-2 4
-1 6
1 6
2 4
1 2}
0 4
12
coordinate:
{
1 3
0.7 4.7
2.3 6
4 6
5 4
4 2
3 1}
2.75 3.5875
14.105
26.105
```

5. Объяснение результатов работы программы.

- 1) Метод `center() const` возвращает точку с x – деление суммы x координат всех точек данной фигуры на их количество, y – аналогично x .
- 2) Метод `print(std::ostream&) const` печатает координаты всех точек данной фигуры.
- 3) Метод `square() const` вычисляет площадь данной фигуры по методу Гаусса (формула землемера, метод шунтирования) и возвращает это значение.
- 4) Удаление в `main.cpp` фигуры из вектора по индексу происходит:
 - удаляется фигура с помощью `delete`.
 - элементы вектора сдвигаются влево циклом `for`, чтобы закрыть индекс удаленного элемента.
 - используется метод вектора `pop_back()`.

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с наследованием и динамическим полиморфизмом в C++, с системой сборки Cmake, с системой контроля версий git. Узнал о виртуальных функциях, абстрактных классах. Также я узнал для себя новый способ вычисления площади по координатам — метод Гаусса.

7. Литература.

- 1) лекции по ООП МАИ.
- 2) Г.Шилдт «C++».