

Московский Авиационный Институт  
(Национальный Исследовательский Университет)

**Лабораторная работа №3 по курсу**  
**«Объектно-ориентированное программирование»**

Студент: Шаларь Игорь Павлович

Группа: М8О-208Б-20

Вариант: 23

Преподаватель: Дорохов Евгений Павлович

Оценка: \_\_\_\_\_

Дата: \_\_\_\_\_

## Цель:

- Программирование классов на языке C++
- Управление памятью в языке C++
- Изучение базовых понятий ООП.
- Знакомство с классами в C++.
- Знакомство с перегрузкой операторов.
- Знакомство с дружественными функциями.
- Знакомство с операциями ввода-вывода из стандартных библиотек.

## Задание:

### Задание

Необходимо спроектировать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания. Классы должны удовлетворять следующим правилам:

- Должны быть названы также, как в вариантах задания и расположены в отдельных файлах: отдельно заголовки (имя\_класса\_с\_маленькой\_буквы.h), отдельно описания методов (имя\_класса\_с\_маленькой\_буквы.cpp);
- Иметь общий родительский класс Figure;
- Содержать конструктор по умолчанию;
- Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока std::cin, расположенных через пробел. *Пример:*  
`0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0`
- Содержать набор общих методов:
  - `size_t` VertexesNumber() - метод, возвращающий количество вершин фигуры;
  - `double` Area() - метод расчета площади фигуры;
  - `void` Print(std::ostream& os) - метод печати типа фигуры и ее координат вершин в поток вывода os в формате:  
`Rectangle: (0.0, 0.0) (1.0, 0.0) (1.0, 1.0) (0.0, 1.0)\n`

Программа должна позволять:

- Вводить произвольные фигуры и добавлять их в общий контейнер. Разрешается использовать стандартный контейнеры std;
- Распечатывать содержимое контейнера;

Вариант 23: triangle hexagon octagon

## Описание программы:

figure.h - описание родительского класса для всех классов-фигур.

triangle.h, octagon.h, hexagon.h - заголовочные файлы описывающие классы-фигуры.

triangle.cpp, octagon.cpp, hexagon.cpp - реализация.

main.cpp - основной файл, взаимодействие с пользователем, добавление фигур в стандартный std контейнер vector и их вывод.

### Примеры работы:

#### input:

```
t 0 0 2 0 0 2
p
h 1 1 2 2 3 3 4 4 5 5 6 6
o 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8
p
t 0 0 2 0 0 2
p
q
```

#### output:

```
Triangle: (0.0, 0.0) (2.0, 0.0) (0.0, 2.0)

Triangle: (0.0, 0.0) (2.0, 0.0) (0.0, 2.0)
Hexagon: (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) (6.0, 6.0)
Octagon: (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) (6.0, 6.0) (7.0, 7.0) (8.0, 8.0)

Triangle: (0.0, 0.0) (2.0, 0.0) (0.0, 2.0)
Hexagon: (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) (6.0, 6.0)
Octagon: (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) (6.0, 6.0) (7.0, 7.0) (8.0, 8.0)
Triangle: (0.0, 0.0) (2.0, 0.0) (0.0, 2.0)
```

### Дневник отладки:

Была неправильно реализована перегрузка операторов ввода и вывода.  
Ошибки были исправлены.

### Выводы:

Узнал, как перегружать операторы и как использовать дружественные функции, что является полезным навыком при работе с классами. Познакомился с таким базовым понятием ООП, как наследование.

Улучшил свои общие навыки программирования классов и работы с памятью. Научился хранить разные объекты в одном контейнере.

## Исходный код

### CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.20)

set(CMAKE_CXX_STANDARD 14)

add_executable(lab1 main.cpp hexagon.cpp octagon.cpp triangle.cpp)
```

### figure.h:

```
#ifndef FIGURE_H
#define FIGURE_H

class Figure {
public:
    virtual void Print(std::ostream &os) = 0;
    virtual size_t VerticesNumber() = 0;
    virtual double Area() = 0;
    virtual ~Figure() {}
};

#endif
```

### triangle.h:

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <iostream>

#include "figure.h"
```

```

class Triangle : public Figure{

public:

    Triangle(std::istream &is);

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

    void Print (std::ostream &os);

    size_t VerticesNumber();

    double Area();

    // ~Triangle();

private:

    std::pair <double, double> p[3];

};

#endif

```

#### **octagon.h:**

```

#ifndef OCTAGON_H
#define OCTAGON_H

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

class Octagon : public Figure{

public:

    Octagon(std::istream &is);

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

    void Print (std::ostream &os);

```

```

        size_t VerticesNumber();

        double Area();

        // ~Octagon();

private:

        std::pair <double, double> p[8];

};

#endif

```

#### **hexagon.h:**

```

#ifndef HEXAGON_H
#define HEXAGON_H

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

class Hexagon : public Figure{

public:

        Hexagon(std::istream &is);

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

        void Print (std::ostream &os);

        size_t VerticesNumber();

        double Area();

        // ~Hexagon();

private:

        std::pair <double, double> p[6];

```

```
};
```

```
#endif
```

**triangle.cpp:**

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

```
#include <iostream>
```

```
#include <math.h>
```

```
using namespace std;
```

```
double tr_distance(pair <double, double> a, pair<double, double> b){  
    return sqrt(pow(a.first - b.first, 2) + pow(a.second - b.second, 2));  
}
```

```
double tr_S_triangle (double a, double b, double c){  
    double p = (a + b + c) / 2;  
    return sqrt(p * (p - a) * (p - b) * (p - c));  
}
```

```
Triangle::Triangle(istream &is){  
    for (int i = 0; i < 3; i++) is >> this->p[i].first >> this->p[i].second;  
}
```

```
void Triangle::Print (ostream &os){  
    os << fixed;  
    os.precision(1);  
    os << "Triangle:";  
    for (int i = 0; i < 3; i++) os << " (" << this->p[i].first << ", " << this->p[i].second << ")";  
}
```



```

        os << endl;
    }

    size_t Triangle::VerticesNumber(){
        return 3;
    }

    double Triangle::Area(){
        double res = 0, a, b, c;

        a = tr_distance(this->p[0], this->p[1]);
        b = tr_distance(this->p[1], this->p[2]);
        c = tr_distance(this->p[0], this->p[2]);

        res += tr_S_triangle(a, b, c);

        return res;
    }

    //Triangle::~~Triangle(){
    //    cout << "Deleted" << endl;
    //}

```

#### **octagon.cpp:**

```

#include "octagon.h"

#include <iostream>

#include <math.h>

using namespace std;

double oct_distance(pair <double, double> a, pair<double, double> b){
    return sqrt(pow(a.first - b.first, 2) + pow(a.second - b.second, 2));
}

```

```
double oct_S_triangle (double a, double b, double c){
    double p = (a + b + c) / 2;
    return sqrt(p * (p - a) * (p - b) * (p - c));
}
```

```
Octagon::Octagon(istream &is){
    for (int i = 0; i < 8; i++) is >> this->p[i].first >> this->p[i].second;
}
```

```
void Octagon::Print (ostream &os){
    os << fixed;
    os.precision(1);
    os << "Octagon:";
    for (int i = 0; i < 8; i++) os << " (" << this->p[i].first << ", " << this->p[i].second << ")";
    os << endl;
}
```

```
size_t Octagon::VerticesNumber(){
    return 8;
}
```

```
double Octagon::Area(){
    double res = 0, a, b, c;
    for (int i = 1; i < 7; i++){
        a = oct_distance(this->p[0], this->p[i]);
        b = oct_distance(this->p[i], this->p[i + 1]);
        c = oct_distance(this->p[0], this->p[i + 1]);
        res += oct_S_triangle(a, b, c);
    }
}
```

```

        return res;
    }

//Octagon::~~Octagon(){
//    cout << "Deleted" << endl;
//}

```

#### **hexagon.cpp:**

```

#include "hexagon.h"

#include <iostream>

#include <math.h>

using namespace std;

double hex_distance(pair <double, double> a, pair<double, double> b){
    return sqrt(pow(a.first - b.first, 2) + pow(a.second - b.second, 2));
}

double hex_S_triangle (double a, double b, double c){
    double p = (a + b + c) / 2;
    return sqrt(p * (p - a) * (p - b) * (p - c));
}

Hexagon::Hexagon(istream &is){
    for (int i = 0; i < 6; i++) is >> this->p[i].first >> this->p[i].second;
}

void Hexagon::Print (ostream &os){
    os << fixed;
    os.precision(1);

```

```

    os << "Hexagon:";

    for (int i = 0; i < 6; i++) os << " (" << this->p[i].first << ", " << this->p[i].second << ")";

    os << endl;
}

size_t Hexagon::VerticesNumber(){

    return 6;

}

double Hexagon::Area(){

    double res = 0, a, b, c;

    for (int i = 1; i < 5; i++){

        a = hex_distance(this->p[0], this->p[i]);

        b = hex_distance(this->p[i], this->p[i + 1]);

        c = hex_distance(this->p[0], this->p[i + 1]);

        res += hex_S_triangle(a, b, c);

    }

    return res;

}

//Hexagon::~Hexagon(){

//    cout << "Deleted" << endl;

//}

```

**main.cpp:**

```

#include "hexagon.h"

#include "octagon.h"

#include "triangle.h"

#include "figure.h"

#include <iostream>

```

```

#include<vector>

using namespace std;

int main(){

    cout << "Add triangle: t [x1] [y1] [x2] [y2] [x3] [y3]" << endl;

    cout << "Example: t 1 2 3 4 5 6" << endl;

    cout << "Add hexagon: h [x1] [y1] [x2] [y2] [x3] [y3] [x4] [y4] [x5] [y5] [x6] [y6]" << endl;

    cout << "Add octagon: o [x1] [y1] [x2] [y2] [x3] [y3] [x4] [y4] [x5] [y5] [x6] [y6] [x7] [y7] [x8] [y8]" <<
        endl;

    cout << "Print: p" << endl;

    cout << "Stop: q" << endl;

    cout << "
    _____" << endl << endl;

    char ch;

    vector <Figure *> v;

    cin >> ch;

    while (ch != 'q'){

        if (ch == 't'){

            Figure * f = new Triangle(cin);

            v.push_back(f);

        }

        if (ch == 'h'){

            Figure * f = new Hexagon(cin);

            v.push_back(f);

        }

        if (ch == 'o'){

            Figure * f = new Octagon(cin);

            v.push_back(f);

        }
    }
}

```

```
    if (ch == 'p'){  
        for (int i = 0; i < v.size(); i++) v[i]->Print(cout);  
    }  
    cin >> ch;  
}  
return 0;  
}
```