Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Объектно-ориентированное программирование»
III Семестр

Задание 3
Вариант 24
Наследование, полиморфизм

Студент:	Синявский А.В
Группа:	М80-208Б-18
Преподаватель:	Журавлёв А.А
Оценка:	
Дата:	

Москва 2019

1. Код программы на языке C++
1.1 figure.h

```
#ifndef OOP_EXERCISE_03_FIGURE_H
#define OOP_EXERCISE_03_FIGURE_H


class Dot {
public:
    double x;
    double y;
    Dot();
    Dot(double X, double Y);
    Dot& operator=(const Dot &A);
    Dot operator+(const Dot &A);
    Dot operator-(const Dot &A);
    Dot operator/(const double &A);
    friend std::ostream &operator<<(std::ostream &os, const Dot& A);
    friend std::istream &operator>>(std::istream &is, Dot& A);
    double Length(const Dot &A);
};

Dot operator""_dot(const char* str, size_t size);

class Figure {
public:
    virtual Dot Center() = 0;
    virtual void PrintOut(std::ostream& os) = 0;
    virtual double Area() = 0;
    virtual ~Figure() = default;
};

class Octagon : public Figure {
private:
    Dot* coordinates;
public:
    Octagon();
    explicit Octagon(std::istream& is);
    Dot Center() override;
    void PrintOut(std::ostream& os) override;
    double Area() override;
    ~Octagon() override;
};


class Triangle : public Figure {
private:
    Dot *coordinates;
public:
    Triangle();
    explicit Triangle(std::istream& is);
    Dot Center() override;
```

```
    void PrintOut(std::ostream& os) override;
    double Area() override;
    ~Triangle() override;
};

class Square : public Figure {
private:
    Dot *coordinates;
public:
    Square();
    explicit Square(std::istream& is);
    Dot Center() override;
    double Area() override;
    void PrintOut(std::ostream& os) override;
    ~Square() override;
};


#endif //OOP_EXERCISE_03_FIGURE_H
```

1.2 figure.cpp

```
#include <iostream>
#include <sstream>
#include <cmath>
#include "Figure.h"

//Методы класса Dot

Dot::Dot() {
    x = 0;
    y = 0;
}

Dot::Dot(double X, double Y) {
    x = X;
    y = Y;
}



Dot operator""_dot(const char* str, size_t size) {
    std::istringstream is(str);
    char tmp;
    double x, y;
    is >> x >> tmp >> y;
    return {x, y};
}

Dot& Dot::operator=(const Dot &A) {
    this->x = A.x;
    this->y = A.y;
```

```
    return *this;
}

Dot Dot::operator+(const Dot &A) {
    Dot res;
    res.x = this->x + A.x;
    res.y = this->y + A.y;
    return res;
}

Dot Dot::operator-(const Dot &A) {
    Dot res;
    res.x = this->x - A.x;
    res.y = this->y - A.y;
    return res;
}

Dot Dot::operator/(const double &A) {
    Dot res;
    res.x = this->x / A;
    res.y = this->y / A;
    return res;
}

std::ostream &operator<<(std::ostream &os, const Dot& A) {
    os << "(" << A.x << "; " << A.y << ")";
    return os;
}

std::istream &operator>>(std::istream &is, Dot& A) {
    is >> A.x >> A.y;
    return is;
}

double Dot::Length(const Dot &A) {
    double res;
    res = sqrt(pow(this->x - A.x, 2) + pow(this->y - A.y, 2));
    return res;
}

//конец Dot

//Методы класса Octagon

double Octagon::Area() {
    double res = 4*sin(0.785398)*pow(this->Center().Length(this->coordinates[0]), 2);
    return res;
}

Dot Octagon::Center() {
    Dot res(0, 0);
    for (int i = 0; i < 8; ++i) {
```

```
        res = res + coordinates[i];
    }
    res = res / 8.0;
    return res;
}

void Octagon::PrintOut(std::ostream& os) {
    for (int i = 0; i < 8; ++i) {
        os << this->coordinates[i];
        if (i != 7) {
            os << ", ";
        }
    }
    os << '\n';
}

Octagon::Octagon() {
    coordinates = new Dot[8];
    for (int i = 0; i < 8; ++i) {
        coordinates[i] = "0.0 0.0"_dot;
    }
}

Octagon::Octagon(std::istream &is) {
    coordinates = new Dot[8];
    for (size_t i = 0; i < 8; ++i) {
        is >> coordinates[i];
    }
}

Octagon::~Octagon() {
    delete[] this->coordinates;
}

//конец Octagon

//методы класса Triangle

Triangle::Triangle() {
    coordinates = new Dot[3];
    for (int i = 0; i < 3; ++i) {
        coordinates[i] = "0.0 0.0"_dot;
    }
}

Triangle::Triangle(std::istream &is) {
    coordinates = new Dot[3];
    for (size_t i = 0; i < 3; ++i) {
        is >> coordinates[i];
    }
}
```

```
Dot Triangle::Center() {
    Dot res(0, 0);
    for (int i = 0; i < 3; ++i) {
        res = res + coordinates[i];
    }
    res = res / 3.0;
    return res;
}

double Triangle::Area() {
    double x1 = this->coordinates[0].x;
    double x2 = this->coordinates[1].x;
    double x3 = this->coordinates[2].x;
    double y1 = this->coordinates[0].y;
    double y2 = this->coordinates[1].y;
    double y3 = this->coordinates[2].y;

    double res = std::abs((x2 - x1)*(y3 - y1) - (x3 - x1)*(y2 - y1)) / 2;
    return res;
}

void Triangle::PrintOut(std::ostream& os) {
    for (int i = 0; i < 3; ++i) {
        os << this->coordinates[i];
        if (i != 2) {
            os << ", ";
        }
    }
    os << '\n';
}

Triangle::~Triangle() {
    delete[] coordinates;
}

//конец Triangle

//методы класса Square

Square::Square() {
    coordinates = new Dot[2];
    for (int i = 0; i < 2; ++i) {
        coordinates[i] = "0.0 0.0"_dot;
    }
}

Square::Square(std::istream &is) {
    coordinates = new Dot[2];
    for (size_t i = 0; i < 2; ++i) {
        is >> coordinates[i];
    }
}
```

```
Dot Square::Center() {
    Dot res = (this->coordinates[0] + this->coordinates[1]) / 2;
    return res;
}

double Square::Area() {
    double res = this->coordinates[0].Length(this->coordinates[1]);
    res = pow(res , 2) / 2;
    return  res;
}

void Square::PrintOut(std::ostream& os) {
    Dot C = this->Center();
    double tmp;
    Dot res[2];
    for (int i = 0; i < 2; ++i) {
        res[i] = this->coordinates[i];
        res[i] = res[i] - C;
        tmp = res[i].y;
        res[i].y = res[i].x;
        res[i].x = -tmp;
        res[i] = res[i] + C;
    }
    os << res[0] << ", " << this->coordinates[0] << ", "
    << res[1] << ", " << this->coordinates[1] << '\n';
}

Square::~Square() {
    delete[] coordinates;
}
```

1.3 main.cpp

```
#include <iostream>
#include <vector>
#include "Figure.h"



int main() {
    std::vector<Figure*> v;
    int opt = 0, index = 0;
    Dot tmp_center;
    double tmp_area;

    while (opt != -1) {
        std::cout << "Choose option (-2 for man, -1 to close)" << '\n';
        std::cin >> opt;
        switch (opt) {
            case -1:
                for (auto & figure: v) {
```

```
            delete figure;
        }
        break;
    case -2:
        std::cout << "Option 1: add octagon. Defined by 8 points (x then y coordinate for each)\n"
        << "Option 2: add Triangle (3 points)\n"
        << "Option 3: add square (2 points from ends of a diagonal line)\n"
        << "Option 4: print all figures\n"
        << "Option 5: count and print centers of all figures\n"
        << "Option 6: count and print areas of all figures\n"
        << "Option 7: print all available info about all figures\n"
        << "Option 8: count and print summary area of all figures\n"
        << "Option 9: erase figure from vector by it`s index (starts from 1)\n";
        break;
    case 1:
        v.push_back(new Octagon(std::cin));
        break;
    case 2:
        v.push_back(new Triangle(std::cin));
        break;
    case 3:
        v.push_back(new Square(std::cin));
        break;
    case 4:
        for (size_t i = 0; i < v.size(); ++i) {
            std::cout << '\t' << i+1 << ':' << std::endl;
            v[i]->PrintOut(std::cout);
        }
        break;
    case 5:
        for (size_t i = 0; i < v.size(); ++i) {
            std::cout << '\t' << i+1 << ':' << std::endl;
            tmp_center = v[i]->Center();
            std::cout << tmp_center << std::endl;
        }
        break;
    case 6:
        for (size_t i = 0; i < v.size(); ++i) {
            std::cout << '\t' << i+1 << ':' << std::endl;
            tmp_area = v[i]->Area();
            std::cout << tmp_area << std::endl;
        }
        break;
    case 7:
        for (size_t i = 0; i < v.size(); ++i) {
            std::cout << '\t' << i+1 << ':' << std::endl;
            tmp_area = v[i]->Area();
            tmp_center = v[i]->Center();
            std::cout << "Angle coordinates:\t";
            v[i]->PrintOut(std::cout);
            std::cout << "Center:\t\t\t" << tmp_center << '\n' << "Area:\t\t\t" << tmp_area << '\n';
        }
```

```
            break;
        case 8:
            tmp_area = 0;
            for (auto & figure : v) {
                tmp_area += figure->Area();
            }
            std::cout << "Summary area: " << tmp_area << std::endl;
            break;
        case 9:
            std::cout << "choose index of a figure you want to delete (from 1 to " << v.size() << "): ";
            std::cin >> index;
            delete v[index-1];
            v.erase(v.begin()+index-1);
            break;
        default:
            std::cout << "Sorry, there is no such option" << '\n';
            break;
      }
   }
   return 0;
}
```

2. Ссылка на репозиторий на GitHub

https://github.com/Siegmeyer1/oop_exercise_03

3. Набор тестов

1)
```
-2
2
0 2 2 0 0 0
3
0 2 2 2
1
2 1 1 2 -1 2 -2 1 -2 -1 -1 -2 1 -2 2 -1
7
8
9
2
4
5
6
-1
```

2)
```
2
1 -1 -1 -1 -1 3
7
```

3
1 0 -3 -2
3
-2 1 0 -3
7
-1

4. Результаты тестов

1)
Choose option (-2 for man, -1 to close)
-2
Option 1: add octagon. Defined by 8 points (x then y coordinate for each)
Option 2: add Triangle (3 points)
Option 3: add square (2 points from ends of a diagonal line)
Option 4: print all figures
Option 5: count and print centers of all figures
Option 6: count and print areas of all figures
Option 7: print all available info about all figures
Option 8: count and print summary area of all figures
Option 9: erase figure from vector by it`s index (starts from 1)
Choose option (-2 for man, -1 to close)
2
0 2 2 0 0 0
Choose option (-2 for man, -1 to close)
3
0 2 2 2
Choose option (-2 for man, -1 to close)
1
2 1 1 2 -1 2 -2 1 -2 -1 -1 -2 1 -2 2 -1
Choose option (-2 for man, -1 to close)
7
1:
Angle coordinates: (0; 2), (2; 0), (0; 0)
Center: (0.666667; 0.666667)
Area: 2
2:
Angle coordinates: (1; 1), (0; 2), (1; 3), (2; 2)
Center: (1; 2)
Area: 2
3:
Angle coordinates: (2; 1), (1; 2), (-1; 2), (-2; 1), (-2; -1), (-1; -2), (1; -2), (2; -1)
Center: (0; 0)
Area: 14.1421
Choose option (-2 for man, -1 to close)
8
Summary area: 18.1421

```
Choose option (-2 for man, -1 to close)
9
choose index of a figure you want to delete (from 1 to 3): 2
Destructed square
Choose option (-2 for man, -1 to close)
4
        1:
(0; 2), (2; 0), (0; 0)
        2:
(2; 1), (1; 2), (-1; 2), (-2; 1), (-2; -1), (-1; -2), (1; -2), (2; -1)
Choose option (-2 for man, -1 to close)
5
        1:
(0.666667; 0.666667)
        2:
(0; 0)
Choose option (-2 for man, -1 to close)
6
        1:
2
        2:
14.1421
Choose option (-2 for man, -1 to close)
-1
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab3/build$
```

```
2)
Choose option (-2 for man, -1 to close)
2
1 -1 -1 -1 -1 3
Choose option (-2 for man, -1 to close)
7
      1:
Angle coordinates:     (1; -1), (-1; -1), (-1; 3)
Center:                (-0.333333; 0.333333)
Area:                  4
Choose option (-2 for man, -1 to close)
3
1 0 -3 -2
Choose option (-2 for man, -1 to close)
3
-2 1 0 -3
Choose option (-2 for man, -1 to close)
7
      1:
Angle coordinates:     (1; -1), (-1; -1), (-1; 3)
Center:                (-0.333333; 0.333333)
Area:                  4
      2:
Angle coordinates:     (-2; 1), (1; 0), (0; -3), (-3; -2)
```

```
Center:             (-1; -1)
Area:               10
      3:
Angle coordinates:    (-3; -2), (-2; 1), (1; 0), (0; -3)
Center:             (-1; -1)
Area:               10
Choose option (-2 for man, -1 to close)
-1
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab3/build$
```

5. Объяснение работы программы

Абстрактный класс Figure описывается в заголовочном файле. Его наследниками являются классы Octagon, Triangle и Square. В них переопределяются чистые виртуальные методы из Figure. Для возможности “красиво” удалять фигуры из вектора, в абстрактном классе определён виртуальный деструктор, перезаписываемый во всех потомках. Для удобного хранения и проведения операций с координатами вершин был введён класс точек координатной плоскости Dot с перегруженными операторами элементарных математических действий. Функции вывода у всех трёх фигур отличаются лишь количесвом точек, во всех них считаются средние координаты по X и У. Функции вывода работают схоже у всех классов кроме квадрата. В случае квадрата, чтобы избежать проверок на “квадратность”, я решил задавать его двумя точками, являющимися концами произвольной диагонали квадрата. Площадь при таком задании вычисляется просто – квадрат произведения заданной диагонали на синус 45 градусов, центр также не вызывает вопросов, а для вывода квадрата высчитываются 2 оставшиеся незаданные точки.

Функция main по сути представляет собой интерфейс вызова заданных функций. Согласно заданию все фигуры помещаются в стандартный вектор, из которого удаляются по индексу при помощи стандартного метода erase. При удалении отдельно вызывается деструктор удаляемой фигуры.

Вывод

Проделав работу, я изучил основы наследования классов в C++ и на наглядном примере увидел, в чём смысл наследования и как оно может упростить написание кода.