Лабораторная работа №1

Цель работы

Целью лабораторной работы является:

- 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

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 расположенны в раздельных 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;

Листинг

Figure.h

```
//
```

```
// Created by Илья Рожков on 12.09.2021.
//

#ifndef LAB1_FIGURE_H
#define LAB1_FIGURE_H
#include "iostream"
#include <utility>
#include <math.h>
#include <cmath>




class Figure {

public:
virtual void Print() const = 0;
virtual size_t VertexesNumber() const = 0;
virtual double Area() const = 0;


};


#endif //LAB1_FIGURE_H
```

Geron

Formula.cpp

```
//
// Created by Илья Рожков on 16.09.2021.
//
#include "GeronFormula.h"
#include<cmath>


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

double getDistance(const std::pair<double, double> &x, const std::pair<double,
return sqrt(pow((x.first - y.first), 2) + pow((x.second - y.second), 2));
}

double GeronFormulaFromCordinates(const Cordinate &a, const Cordinate &b,
double x = getDistance(a, b);
double y = getDistance(b, c);
double z = getDistance(c, a);
return GeronFormula(x, y, z);
```

```
}

double AreaOfMultigone(const std::vector<Cordinate> &cordinates) {
double s = 0;
for (int i = 0; i < cordinates.size(); i += 3)
s += GeronFormulaFromCordinates(cordinates[i], cordinates[(i + 1) %
return s;
}
```

GeronFormula.h

```
//
// Created by Илья Рожков on 16.09.2021.
//

#ifndef LAB1_GERONFORMULA_H
#define LAB1_GERONFORMULA_H

#include <utility>
#include <vector>

typedef std::pair<double, double> Cordinate;


double GeronFormula(double a, double b, double c);
```

```
    double getDistance(const std::pair<double, double>& x , const std::pair<double,
    double GeronFormulaFromCordinates(const Cordinate& a, const Cordinate&
    double AreaOfMultigone(const std::vector<Cordinate>& cordinates);


    #endif //LAB1_GERONFORMULA_H
//
    // Created by Илья Рожков on 16.09.2021.
    //

    #include "Hexagon.h"

    Hexagon::Hexagon() {
    for (int i = 0; i < 6; i++) {
    Cordinate elemt = std::make_pair(0, 0);
    _cordinates.push_back(elemt);
    }

    }

    Hexagon::Hexagon(const std::vector<Cordinate> &cordinates) :
    if (_cordinates.size() != 6) {
    throw "wrong size";
    }

    }

    size_t Hexagon::VertexesNumber() const {
    return 6;
    }
```

```
double Hexagon::Area() const {
return AreaOfMultigone(_cordinates);
}

void Hexagon::Print() const {
for (int i = 0; i < _cordinates.size(); i++)
std::cout << _cordinates[i].first << ' ' << _cordinates[i].second << std::endl;

}

std::ostream &operator<<(std::ostream &out, const Hexagon &r) {
for (int i = 0; i < r._cordinates.size(); i++)
out << r._cordinates[i].first << ' ' << r._cordinates[i].second << std::endl;
return out;
}

std::istream &operator>>(std::istream &in, Hexagon &r) {
for (int i = 0; i < 6; i++)
in >> r._cordinates[i].first >> r._cordinates[i].second;
return in;
}

Hexagon::~Hexagon() {

}
```

Hexago

n.
h

```
//
// Created by Илья Рожков on 16.09.2021.
//

#ifndef LAB1_HEXAGON_H
#define LAB1_HEXAGON_H
#include "Figure.h"
#include "GeronFormula.h"



class Hexagon : public Figure {
public:
Hexagon();
~Hexagon();
Hexagon(const std::vector<Cordinate>& cordinates);
size_t VertexesNumber() const override;
double Area() const override;
void Print() const override;
friend std::ostream& operator<<(std::ostream &out, const Hexagon& r);
friend std::istream& operator>> (std::istream &in, Hexagon& r);



protected:
std::vector<Cordinate> _cordinates;

};
```

	#endif //LAB1_HEXAGON_H