

1. Постановка задачи

Важно учитывать при реализации:

- Необходимо выполнить разделение на h и cpp файлы для каждого класса. h файлы содержат определение, cpp файлы содержат реализацию. функция main обязана располагаться в отдельном cpp файле.
- Необходимо написать для каждого класса все конструкторы: с параметром, без, конструктор копирования, виртуальный деструктор.
- Результат работы программы должен сохраняться в выходной файл. Также должна быть возможность загрузки из файла данных.
- Реализовать пользовательское меню согласно заданию. Все параметры вводятся пользователем или из файла, не должно существовать в программе параметров, заданных по умолчанию. Должно быть представлено максимально возможное меню пользователя, со всеми действиями, которые может выполнить пользователь при работе с программой.
- Обратите внимание на формулировку задания, если не указано создание абстрактного класса, то НЕ нужно создавать абстрактный класс.
- Номер варианта соответствует номеру студента в списке

Задание. 17го варианта в списке нет, есть всего 16. Остаток от деления 17 на 16 равен 1, поэтому в качестве варианта будет использован вариант №1:

Создать абстрактный базовый класс с виртуальной функцией «Площадь». Создать производные классы «Прямоугольник», «Круг», «Прямоугольный треугольник», «Трапеция» со своими функциями площади и переменными. Для проверки определить массив ссылок на абстрактный класс, которым присваиваются адреса различных объектов.

2. Формализация

В абстрактном базовом классе Figure помимо виртуальной функции Square будет также виртуально перегружен оператор вывода в поток, для предоставления дополнительного удобства пользователю. Будет создан файл с объявлением абстрактного класса Figure.h, однако файл Figure.cpp существовать не будет, потому что абстрактный класс не подразумевает реализации. Стоит отметить, что массив «ссылок» не мог бы существовать в рамках стандарта языка C++ в принципе, поэтому вместо этого будет определен std::vector указателей на класс Figure.

3. Исходный код

lab3.h

```
//  
// Created by user on 4/15/2024.  
//  
  
#ifndef CMAKEPROJECT_LAB3_H  
#define CMAKEPROJECT_LAB3_H  
  
#include "Circle.h"  
#include "Rectangle.h"  
#include "RectTriangle.h"  
#include "Trapezoid.h"  
  
#include "../libs/libs.h"  
using namespace std;  
  
int lab3();
```

```

        #endif //CMAKEPROJECT_LAB3_H
lab3.cpp
    //
    // Created by user on 4/15/2024.
    //

#include "lab3.h"
int lab3(){

    ifstream is;
    ofstream os;
    double a,b,c,d;

    vector<Figure*> array;

    is.open(R"(C:\Users\user\Desktop\cmakeproject\source\lab3\Input.txt)");
    is>>a;
    Circle* circle = new Circle(a);

    is>>a>>b;
    RectTriangle* rectTriangle = new RectTriangle(a,b);

    is>>a>>b;
    Rectangle* rectangle = new Rectangle(a,b);

    is>>a>>b>>c>>d;
    Trapezoid* trapezoid = new Trapezoid(a,b,c,d);

    is.close();
    array.push_back(circle);
    array.push_back(rectTriangle);
    array.push_back(rectangle);
    array.push_back(trapezoid);

    system("cls");
    int inp;
    do{
        cout<<"1. Print squares to Output.txt"<<endl;
        cout<<"2. Print squares here"<<endl;
        cout<<"3.    Print    parameters    of    figures    from\nInput.txt"<<endl;
        cout<<"4.    Refresh    parameters    from    Input.txt\nagain"<<endl;

```

```

        cout<<"5. End"<<endl;

        cin>>inp;
        system("cls");

        switch(inp){
            case 1:

os.open(R"(C:\Users\user\Desktop\cmakeproject\source\lab3\Outp
ut.txt)");
                os<<"Circle:                                "<<array[0]-
>square()<<endl;
                os<<"RectTriangle:                            "<<array[1]-
>square()<<endl;
                os<<"Rectangle:                                "<<array[2]-
>square()<<endl;
                os<<"Trapezoid:                                "<<array[3]-
>square()<<endl;
                os.close();
                break;
            case 2:
                cout<<"Circle:                                "<<array[0]-
>square()<<endl;
                cout<<"RectTriangle:                            "<<array[1]-
>square()<<endl;
                cout<<"Rectangle:                                "<<array[2]-
>square()<<endl;
                cout<<"Trapezoid:                                "<<array[3]-
>square()<<endl;
                break;
            case 3:
                for(int i = 0;i!=4;++i) array[i]-
>operator<<(cout)<<endl;
                break;
            case 4:
                array.clear();
                delete circle;
                delete rectTriangle;
                delete rectangle;
                delete trapezoid;

is.open(R"(C:\Users\user\Desktop\cmakeproject\source\lab3\Inpu
t.txt)");
                is>>a;
                circle = new Circle(a);

```

```

        is>>a>>b;
        rectTriangle = new RectTriangle(a,b);

        is>>a>>b;
        rectangle = new Rectangle(a,b);

        is>>a>>b>>c>>d;
        trapezoid = new Trapezoid(a,b,c,d);
        array.push_back(circle);
        array.push_back(rectTriangle);
        array.push_back(rectangle);
        array.push_back(trapezoid);
        is.close();
        for(int i = 0;i!=4;++i) array[i]-
>operator<<(cout)<<endl;
        break;
    case 5:
        break;
    default:
        if (inp == EOF) inp = 6;
        cout << "There is no enough command. If you
want to exit enter 5\n";
        break;
    }
}while(inp != 5);

    delete circle;
    delete rectTriangle;
    delete rectangle;
    delete trapezoid;

    return 0;
}

```

Figure.h

```

//
// Created by user on 4/15/2024.
//

#ifndef CMAKEPROJECT_FIGURE_H
#define CMAKEPROJECT_FIGURE_H

#include <iostream>
class Figure {
public:

```

```
        virtual double square() = 0;
        virtual std::ostream& operator<<(std::ostream &os) = 0;
};
```

```
        #endif //CMAKEPROJECT_FIGURE_H
Circle.h
```

```
//
// Created by user on 4/15/2024.
//
```

```
#ifndef CMAKEPROJECT_CIRCLE_H
#define CMAKEPROJECT_CIRCLE_H
```

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

```
class Circle : public Figure{
private:
    double radius;
public:
    explicit Circle(double radius);
    explicit Circle();
    Circle(const Circle& another);
    virtual ~Circle();

    double square() override;

    std::ostream &operator<<(std::ostream &os) override;
};
```

```
        #endif //CMAKEPROJECT_CIRCLE_H
Circle.cpp
```

```
//
// Created by user on 4/15/2024.
//
```

```
#include "Circle.h"
```

```
double Circle::square() {
    return radius*radius*M_PI;
}
```

```

Circle::Circle(double radius) : radius(radius) {}

Circle::Circle() : radius(1) {}

Circle::~~Circle() = default;

Circle::Circle(const Circle &another) : radius(another.radius)
{}

std::ostream &Circle::operator<<(std::ostream &os) {
    os<<radius;
    return os;
}

```

Rectangle.h

```

//
// Created by user on 4/15/2024.
//

#ifndef CMAKEPROJECT_RECTANGLE_H
#define CMAKEPROJECT_RECTANGLE_H

#include "Figure.h"

class Rectangle : public Figure{
private:
    double _a,_b;

public:
    explicit Rectangle(double a, double b);
    explicit Rectangle();
    Rectangle(const Rectangle& another);
    virtual ~Rectangle();

    double square() override;

    std::ostream &operator<<(std::ostream &os) override;
};

#endif //CMAKEPROJECT_RECTANGLE_H

```

Rectangle.cpp

```

//
// Created by user on 4/15/2024.

```

```

//

#include "Rectangle.h"

double Rectangle::square() {
    return _a*_b;
}

std::ostream &Rectangle::operator<<(std::ostream &os) {
    return os<<_a<<" "<<_b;
}

Rectangle::Rectangle(double a, double b) : _a(a), _b(b){}

Rectangle::Rectangle() : _a(1), _b(1){}

Rectangle::Rectangle(const Rectangle &another) :
    _a(another._a),
    _b(another._b) {}

Rectangle::~~Rectangle() =default;
RectTrinagle.h
//
// Created by user on 4/15/2024.
//

#ifndef CMAKEPROJECT_RECTTRIANGLE_H
#define CMAKEPROJECT_RECTTRIANGLE_H

#include "Figure.h"
#include <cmath>

class RectTriangle : public Figure{
private:
    double _a,_b;
public:
    explicit RectTriangle(double a, double b);
    explicit RectTriangle();
    RectTriangle(const RectTriangle& another);
    virtual ~RectTriangle();

    double square() override;

    std::ostream &operator<<(std::ostream &os) override;
};

```

```

        #endif //CMAKEPROJECT_RECTTRIANGLE_H
RectTriangle.cpp
    //
    // Created by user on 4/15/2024.
    //

    #include "RectTriangle.h"

    double RectTriangle::square() {
        return _a*_b/2.;
    }

    RectTriangle::RectTriangle(double a, double b) : _a(a),
    _b(b){}

    RectTriangle::RectTriangle() : _a(1), _b(1){}

    RectTriangle::RectTriangle(const RectTriangle &another):
        _a(another._a),
        _b(another._b) {}

    std::ostream &RectTriangle::operator<<(std::ostream &os) {
        return os<<_a<<" "<<_b;
    }

    RectTriangle::~RectTriangle() = default;
Trapezoid.h
    //
    // Created by user on 4/15/2024.
    //

    #ifndef CMAKEPROJECT_TRAPEZOID_H
    #define CMAKEPROJECT_TRAPEZOID_H

    #include "Figure.h"
    #include <cmath>

    class Trapezoid : public Figure{
    private:
        double _a,_b,_c,_d;
    public:
        explicit Trapezoid(double a, double b,double c,double d);
        explicit Trapezoid();

```



```

        Trapezoid(const Trapezoid& another);
        virtual ~Trapezoid();

        double square() override;

        std::ostream &operator<<(std::ostream &os) override;
};

#endif //CMAKEPROJECT_TRAPEZOID_H
Trapezoid.cpp
//
// Created by user on 4/15/2024.
//

#include "Trapezoid.h"

double Trapezoid::square() {
    return (_a+_b)/2
        *sqrt(_c*_c-((_a-_b)*(_a-_b)+_c*_c-_d*_d)
            /(2*_a-2*_b));
}

Trapezoid::Trapezoid(double a, double b, double c, double d):
    _a(a), _b(b), _c(c), _d(d){}

Trapezoid::Trapezoid() : _a(0), _b(0), _c(0), _d(0){}

Trapezoid::Trapezoid(const Trapezoid &another) :
    _a(another._a), _b(another._b),
    _c(another._c), _d(another._d){}

std::ostream &Trapezoid::operator<<(std::ostream &os) {
    return os<<_a<<" "<<_b<<" "<<_c<<" "<<_d;
}

Trapezoid::~~Trapezoid() = default;

```

4. Результаты работы программы

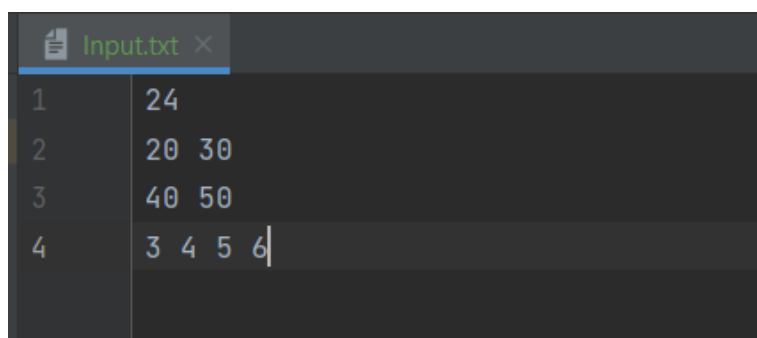
Программа представляет собой меню(рис. 1) из пяти пунктов.

```
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab3.exe

1. Print squares to Output.txt
2. Print squares here
3. Print parameters of figures from Input.txt
4. Refresh parameters from Input.txt again
5. End
```

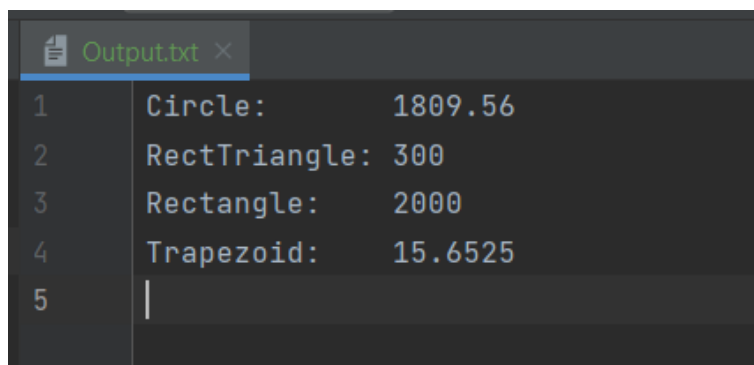
Рис. 1 – Главное меню

Первый пункт выводит результат расчета площадей предоставленных в файле Input.txt в файл Output.txt.



Line	Input
1	24
2	20 30
3	40 50
4	3 4 5 6

Рис. 2 – Input.txt



Line	Output
1	Circle: 1809.56
2	RectTriangle: 300
3	Rectangle: 2000
4	Trapezoid: 15.6525
5	

Рис. 3 – Output.txt

Второй пункт выводит те же данные, но уже в консоль.

```
Circle:      1809.56
RectTriangle: 300
Rectangle:   2000
Trapezoid:   15.6525
1. Print squares to Output.txt
2. Print squares here
3. Print parameters of figures from Input.txt
4. Refresh parameters from Input.txt again
5. End
```

Рис. 4 – Использование второго пункта

Третий пункт выводит в консоль исходные данные из Input.txt. Под капотом не копирование Input.txt, а восстановление данных из объектов и вывод их через виртуально перегруженный, а потом переопределенный и реализованный у наследников оператор вывода в поток.

```
24
20 30
40 50
3 4 5 6
1. Print squares to Output.txt
2. Print squares here
3. Print parameters of figures from Input.txt
4. Refresh parameters from Input.txt again
5. End
```

Рис. 5 – Использование третьего пункта

Четвертый пункт заново открывает Input.txt и считывает данные оттуда. Под капотом происходит удаление объектов и создание новых, с новыми данными, а после вызывается третий пункт.

```
25
19 31
38 52
10 11 12 13
1. Print squares to Output.txt
2. Print squares here
3. Print parameters of figures from Input.txt
4. Refresh parameters from Input.txt again
5. End
```

Рис. 6 – Использование четвертого пункта

После чего можно использовать новые данные во всех остальных пунктах. Ограничений на количество обновлений нет. Воспользуемся, например, вторым пунктом.

```
Circle:      1963.5
RectTriangle: 294.5
Rectangle:   1976
Trapezoid:   120.636
1. Print squares to Output.txt
2. Print squares here
3. Print parameters of figures from Input.txt
4. Refresh parameters from Input.txt again
5. End
```

Рис. 7 – Обновленные данные

И мы видим новые данные площадей.

5. Выводы

В ходе лабораторной работы я научился использовать абстрактные классы в языке программирования C++. Был реализован абстрактный класс Figure и 4 класса наследника Circle, Rectangle, RectTriangle, Trapezoid. В массиве ссылок на Figure, мы не обращались с разными фигурами как с абстракциями и могли лишь вызывать объявленные виртуальные методы, такие как перегруженный оператор вывода в поток и вычисление площади. Также была освоена работа с файлами с помощью fstream.