Лабораторная работа №1

Задача: Необходимо спроектировать и запрограммировать на языке C++ классы фигур, согласно вариантов задания.

Классы должны удовлетворять следующим правилам:

- Должны иметь общий родительский класс Figure.
- Должны иметь общий виртуальный метод Print, печатающий параметры фигуры и ее тип в стандартный поток вывода cout.
- Должный иметь общий виртуальный метод расчета площади фигуры - Square.
- Должны иметь конструктор, считывающий значения основных параметров фигуры из стандартного потока cin.
- Должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Программа должна позволять вводить фигуру каждого типа с клавиатуры, выводить параметры фигур на экран и их площадь.

Фигуры: треугольник, квадрат, прямоугольник

1 Введение

ООП (или объектно-ориентированное программирование) представляет собой способ организации кода программы, когда основными строительными блоками программы являются объекты и классы, а логика работы программы построена на их взаимодействии.
Класс - это такая структура данных, которую может формировать сам программист. В терминах ООП, класс состоит из полей (по-простому - переменных) и методов (по-простому - функций). И, как выяснилось, сочетание данных и функций работы над ними в одной структуре дает невообразимую мощь. Объект - это конкретный экземпляр класса.
Также все языки ООП, включая C++, основаны на трех основополагающих концепциях, называемых инкапсуляцией, полиморфизмом и наследованием. Инкапсуляция - это механизм, который объединяет данные и методы, манипулирующие этими данными, и защищает и то и другое от внешнего вмешательства или неправильного использования.
Наследование – это процесс, посредством которого, один объект может приобретать свойства другого. Точнее, объект может наследовать свойства другого объекта и добавлять к ним черты, характерные только для него.Полиморфизм – это свойство, которое позволяет одно и тоже имя использовать для решения нескольких технически разных задач. Применительно к ООП, целью полиморфизма, является использование одного имени для задания общих для класса действий.

2 Код программы

main.cpp

```
#include <cstdlib>
#include <iostream>
#include "Triangle.h"
#include "FSquare.h"
#include "Rectangle.h"


void func(Figure* ptr) {
ptr->Print();
std::cout <<ptr->Square() <<std::endl;
delete ptr;
}

int main() {
setlocale(LC_ALL,"Russian");
int a;
std::cout <<"----------------------------------------"<<std::endl;
std::cout <<"------------------МЕНЮ------------------"<<std::endl;
std::cout <<"|Выбирите действие:                    |" <<std::endl;
std::cout <<"|1-Вычислить площадь треугольника      |" <<std::endl;
std::cout <<"|2-Вычислить площадь квадрата          |" <<std::endl;
std::cout <<"|3-Вычислить площадь прямоугольника    |" <<std::endl;
std::cout <<"|4-Выход                               |" <<std::endl;
do {
if (!(std::cin >>a)) {
```

```
std::cin.clear();
while (std::cin.get() != '\n');
}
switch (a) {
case 1:func(new Triangle(std::cin));break;
case 2:func(new FSquare(std::cin));break;
case 3:func(new Rectangle(std::cin));break;
case 4:break;
default: std::cout <<"Неверный ввод. Попробуйте снова" <<std::endl;
break;
}
} while (a != 4);
return 0;
}
```

Triangle.h

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <iostream>
#include <cstdlib>
#include "Figure.h"

class Triangle : public Figure {
public:
Triangle();
Triangle(std::istream &is);

double Square() override;
void Print() override;

virtual ~Triangle();
private:
double side_a;
double side_b;
double side_c;
};

#endif
```

Triangle.cpp

```
#include "Triangle.h"
#include <cmath>
#include <iostream>

Triangle::Triangle() {
side_a = 0.0;
side_b = 0.0;
```

```
side_c = 0.0;
}

Triangle::Triangle(std::istream &is) {
std::cout <<"Введите значение a:";
while (!(is >>side_a)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение a:";
}
std::cout <<"Введите значение b:";
while (!(is >>side_b)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение b:";
}
std::cout <<"Введите значение c:";
while (!(is >>side_c)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение c";
}
}

double Triangle::Square() {
std::cout <<"Площадь = ";
double p= double(side_a + side_b + side_c) / 2.0;
return sqrt(p * (p -double(side_a))*(p -double(side_b))*(p -double(side_c)));
}

void Triangle::Print() {
std::cout <<"Сторона a=" <<side_a <<",Сторона b=" <<side_b <<",Сторона c=" <<side_c <<std::endl;
}

Triangle::~Triangle() {
std::cout <<"Triangle deleted" <<std::endl;
}
```

Recatngle.cpp

```
#include "Rectangle.h"
#include <cmath>
#include <iostream>

Rectangle::Rectangle() {
```

```
side_a = 0.0;
side_b = 0.0;
}

Rectangle::Rectangle(std::istream& is) {
std::cout <<"Введите значение a:";
while (!(is >>side_a)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение a:";
}
std::cout <<"Введите значение b:";
while (!(is >>side_b)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение b:";
}
}

double Rectangle::Square() {
std::cout <<"Площадь = ";

return side_a*side_b;
}

void Rectangle::Print() {
std::cout <<"Сторона a=" <<side_a <<",Сторона b=" <<side_b <<std::endl;
}

Rectangle::~Rectangle() {
std::cout <<"Rectangle deleted" <<std::endl;
}
```

Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include <iostream>
#include <cstdlib>
#include "Figure.h"

class Rectangle : public Figure {
public:
Rectangle();
Rectangle(std::istream& is);
```

```
double Square() override;
void Print() override;

virtual ~Rectangle();
private:
double side_a;
double side_b;
};

#endif
```

Figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

class Figure {
public:
virtual double Square() = 0;
virtual void Print() = 0;
virtual ~Figure() {};
};

#endif
```

FSquare.cpp

```
#include "FSquare.h"
#include <cmath>
#include <iostream>

FSquare::FSquare() {
side_a = 0.0;
}

FSquare::FSquare(std::istream &is) {
std::cout <<"Введите значение a:";
while (!(is >>side_a)) {
std::cout <<"Неверный ввод" <<std::endl;
is.clear();
while (std::cin.get() != '\n');
std::cout <<"Введите значение a:";
}
}

double FSquare::Square() {
std::cout <<"Площадь = ";
return pow(side_a,2);
}
```

```
void FSquare::Print() {
std::cout <<"Сторона a=" <<side_a <<std::endl;
}

FSquare::~FSquare() {
std::cout <<"FSquare deleted" <<std::endl;
}
```

FSquare.cpp

```
#ifndef FSQUARE_H
#define FSQUARE_H

#include <iostream>
#include <cstdlib>
#include "Figure.h"

class FSquare : public Figure {
public:
FSquare();
FSquare(std::istream &is);

double Square() override;
void Print() override;

virtual ~FSquare();
private:
double side_a;
};

#endif
```

3 Вывод программы:

```
D:\лабы\lab1\Debug>lab1.exe
----------------------------------------
------------------МЕНЮ------------------
|Выбирите действие:                    |
|1-Вычислить площадь треугольника      |
|2-Вычислить площадь квадрата          |
|3-Вычислить площадь прямоугольника    |
|4-Выход                               |
1
Введите значение a:3
Введите значение b:4
Введите значение c:5
Сторона a=3,Сторона b=4,Сторона c=5
Площадь = 6
```

```
Triangle deleted
3
Введите значение a:2
Введите значение b:2.2
Сторона a=2,Сторона b=2.2
Площадь = 4.4
Rectangle deleted
3
Введите значение a:3.3
Введите значение b:1.1
Сторона a=3.3,Сторона b=1.1
Площадь = 3.63
Rectangle deleted
2
Введите значение a:4.4
Сторона a=4.4
Площадь = 19.36
FSquare deleted
4

D:\лабы\lab1\Debug>
```

4 Вывод

В результате выполнения данной работы я получила начальные знания об языке C++, а именно узнала о наследовании, полиморфизме, виртульных методах, классах, разлчных перегрузках операций.