Отчёт по лабораторной работе №_____03____ по курсу ___2

студента группы _____М80-208Б-18_____, № по списку _3

Адреса www, e-mail, jabber, skype _____

_____anisimov.valera2000@yandex.ru

Работа выполнена: “13“ _____Октября_____ 2019г.

1. **Тема**: ____Операторы, литералы_____
2. **Цель работы**: ____Наследование, полиморфизм_____
3. **Задание** (*вариант № 3*):

Разработать классы согласно варианту задания. Классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать общий набор методов:

- Вычисление геометрического центра фигуры
- Вывод в стандартный поток std::cout координат вершин фигуры
- Вычисление площади фигуры

Создать программу, которая позволяет:

- Вводить из стандартного вода std::cin фигуры, согласно варианту задания
- Сохранять созданные фигуры в динамический массив std::vector<Figure*>
- Вызывыть для всего массива общие функции (1 – 3)
- Необходимо уметь вычислять общую площадь фигур в массиве
- Удалять из массива фигуру по индексу

Фигуры (Вариант 3):

Прямоугольник,трапеция, ромб.

4. **Адрес репозитория на GitHub** https://github.com/wAlienUFOx/oop_exercise_03
5. **Код программы на C++**

main.cpp

```
#include <iostream>
#include "figure.h"
#include "rhombus.h"
#include "trapeze.h"
#include "rectangle.h"
#include <vector>
#include <string>

void read_figt(std::vector<Figure *>& fig)
{
 int figt;
 Trapeze *t = nullptr;
 Rhombus *rh = nullptr;
 Rectangle *re = nullptr;

 std::cout << "Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle\n";
 std::cin >> figt;
 switch (figt) {
 case 1:
  try{
   t = new Trapeze(std::cin);
  } catch(std::logic_error& err){
   std::cout << err.what() << std::endl;
   break;
  }
  fig.push_back(dynamic_cast<Figure*>(t));
  break;
```

```
  case 2:
    try{
      rh = new Rhombus(std::cin);
    }catch(std::logic_error& err){
      std::cout << err.what() << std::endl;
      break;
    }
    fig.push_back(dynamic_cast<Figure*>(rh));
    break;
  case 3:
    try {
      re = new Rectangle(std::cin);
    } catch(std::logic_error& err){
      std::cout << err.what() << std::endl;
      break;
    }
    fig.push_back(dynamic_cast<Figure*>(re));
    break;
  default:
    std::cout << "Wrong. Try 1 - trapeze, 2 - rhombus or 3 - rectangle\n";
  }
}

void Tut(){
  std::cout << "         2         3         2        2    3\n";
  std::cout << "        *******        *       ****\n";
  std::cout << "       *********      ***      ****\n";
  std::cout << "      ***********  1*****3  ****\n";
  std::cout << "    *************    ***    ****\n";
  std::cout << "  ***************    *      ****\n";
  std::cout << "  1                   4   4     1     4\n";
}

int main(){
  unsigned int index;
  double Tarea = 0;
  std::string operation;
  std::vector<Figure*> fig;
  Tut();
  std::cout << "Operations: add / delete / out / quit\n";

  while (std::cin >> operation) {
    if (operation == "add") {
      read_figt(fig);
    }
    else if (operation == "delete") {
      std::cin >> index;
      delete fig[index];
      for (; index < fig.size() - 1; ++index) {
       fig[index] = fig[index + 1];
      }
      fig.pop_back();
    }
    else if (operation == "out") {
      Tarea = 0;
      for (unsigned int i = 0; i < fig.size(); i++) {
       std::cout << i << ":\n";
       std::cout << "Area: " << fig[i]->area() << std::endl;
       std::cout << "Center: " << fig[i]->center() << std::endl;
       std::cout << "Coordinates: ";
       fig[i]->print(std::cout);
       std::cout << std::endl;
```

```
      Tarea += fig[i]->area();
     }
     std::cout << "Total area: " << Tarea << std::endl;
    }
    else if (operation == "quit"){
     for (unsigned int i = 0; i < fig.size(); ++i) {
      delete fig[i];
     }
     return 0;
    }
    else {
     std::cout << "Wrong. Operations: add / delete / out / quit\n";
    }
  }
}
```

point.h

```
#ifndef POINT_H
#define POINT_H

#include <iostream>

class Point {
public:
      Point();
      Point(double x, double y);
      double X() const;
      double Y() const;
      friend std::ostream& operator<< (std::ostream& out, const Point& p);
      friend std::istream& operator>> (std::istream& in, Point& p);
private:
      double x;
      double y;
};



#endif
```

point.cpp

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

Point::Point() : x{0}, y{0}
{}

Point::Point(double x, double y) : x{x}, y{y}
{}

double Point::X() const
{
      return x;
}

double Point::Y() const
{
      return y;
}
```

```
std::ostream& operator<< (std::ostream& out, const Point& p)
{
	out << "(" << p.X() << ";" << p.Y() << ")";
	return out;
}


std::istream& operator>> (std::istream& in, Point& p)
{
	 in >> p.x >> p.y;
}
```

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

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

class Figure {
public:
	virtual double area() const = 0;
	virtual Point center() const = 0;
	virtual std::ostream& print(std::ostream& out) const = 0;
	virtual ~Figure() = default;
};

#endif
```

figure.cpp

```
#include "figure.h"
```

rhombus.h

```
#ifndef RHOMBUS_H
#define RHOMBUS_H

#include "figure.h"
#include "point.h"

class Rhombus : public Figure {
public:
	Rhombus();
	Rhombus(std::istream& in);
	double area() const override;
	Point center() const override;
	std::ostream& print(std::ostream& out) const override;
private:
	Point A;
	Point B;
	Point C;
	Point D;
};

#endif

#endif
```

rhombus.cpp

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

Rhombus::Rhombus() : A{0, 0}, B{0, 0}, C{0, 0}, D{0,0} {}
Rhombus::Rhombus(std::istream& in) {
  in >> A >> B >> C >> D;
  double a, b, c, d;
  a = sqrt((B.X()- A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  b = sqrt((C.X()- B.X()) * (C.X() - B.X()) + (C.Y() - B.Y()) * (C.Y() - B.Y()));
  c = sqrt((C.X()- D.X()) * (C.X() - D.X()) + (C.Y() - D.Y()) * (C.Y() - D.Y()));
  d = sqrt((D.X()- A.X()) * (D.X() - A.X()) + (D.Y() - A.Y()) * (D.Y() - A.Y()));
  if(a != b || a != c || a != d)
    throw std::logic_error("It`s not a rhombus");
}



double Rhombus::area() const{
  double d1 = sqrt((C.X() - A.X()) * (C.X() - A.X()) + (C.Y() - A.Y()) * (C.Y() - A.Y()));;
  double d2 = sqrt((B.X() - D.X()) * (B.X() - D.X()) + (B.Y() - D.Y()) * (B.Y() - D.Y()));;
  return d1 * d2 / 2;
}

Point Rhombus::center() const
{
      return Point{(A.X() + B.X() + C.X() + D.X()) / 4, (A.Y() + B.Y() + C.Y() + D.Y()) / 4};
}

std::ostream& Rhombus::print(std::ostream& out) const
{
      out << A << " " << B << " " << C << " " << D;
      return out;
}
```

rectangle.h

```
#ifndef RECTNAGLE_H
#define RECTNAGLE_H

#include "figure.h"
#include "point.h"

class Rectangle : public Figure {
public:
      Rectangle();
      Rectangle(std::istream& in);
      double area() const override;
      Point center() const override;
      std::ostream& print(std::ostream& out) const override;
private:
      Point A;
      Point B;
      Point C;
      Point D;
};

#endif
```

rectangle.cpp

```
#include "rectangle.h"
```

```
#include <cmath>

Rectangle::Rectangle() : A{0, 0}, B{0, 0}, C{0,0}, D{0,0} {}

Rectangle::Rectangle(std::istream& in) {
  in >> A >> B >> C >> D;
  double a, b, c, d, d1, d2, ABC, BCD, CDA, DAB;
  a = sqrt((B.X()- A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  b = sqrt((C.X()- B.X()) * (C.X() - B.X()) + (C.Y() - B.Y()) * (C.Y() - B.Y()));
  c = sqrt((C.X()- D.X()) * (C.X() - D.X()) + (C.Y() - D.Y()) * (C.Y() - D.Y()));
  d = sqrt((D.X()- A.X()) * (D.X() - A.X()) + (D.Y() - A.Y()) * (D.Y() - A.Y()));
  d1 = sqrt((B.X()- D.X()) * (B.X() - D.X()) + (B.Y() - D.Y()) * (B.Y() - D.Y()));
  d2 = sqrt((C.X()- A.X()) * (C.X() - A.X()) + (C.Y() - A.Y()) * (C.Y() - A.Y()));
  ABC = (a * a + b * b - d2 * d2) / 2 * a * b;
  BCD = (b * b + c * c - d1 * d1) / 2 * b * c;
  CDA = (d * d + c * c - d2 * d2) / 2 * d * c;
  DAB = (a * a + d * d - d1 * d1) / 2 * a * d;
  if(ABC != BCD || ABC != CDA || ABC != DAB)
    throw std::logic_error("It`s not a rectangle");
}

double Rectangle::area() const{
  double a =  sqrt((B.X() - A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  double b =  sqrt((C.X() - B.X()) * (C.X() - B.X()) + (C.Y() - B.Y()) * (C.Y() - B.Y()));;
  return a * b;
}

Point Rectangle::center() const{
  return Point{(A.X() + B.X() + C.X() + D.X()) / 4, (A.Y() + B.Y() + C.Y() + D.Y()) / 4};
}

std::ostream& Rectangle::print(std::ostream& out) const{
  out << A << " " << B << " " << C << " " << D;
  return out;
}
```

trapeze.h

```
#ifndef TRAPEZE_H
#define TRAPEZE_H

#include "figure.h"
#include "point.h"

class Trapeze : public Figure {
public:
       Trapeze();
       Trapeze(std::istream& in);
       double area() const override;
       Point center() const override;
       std::ostream& print(std::ostream& out) const override;
private:
       Point A;
       Point B;
       Point C;
       Point D;
};
#endif
```

trapeze.cpp

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

Trapeze::Trapeze() : A{0, 0}, B{0, 0}, C{0, 0}, D{0,0} {}

Trapeze::Trapeze(std::istream& in) {
  in >> A >> B >> C >> D;
  double a, b, c, d;
  a = sqrt((B.X()- A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  b = sqrt((C.X()- B.X()) * (C.X() - B.X()) + (C.Y() - B.Y()) * (C.Y() - B.Y()));
  c = sqrt((C.X()- D.X()) * (C.X() - D.X()) + (C.Y() - D.Y()) * (C.Y() - D.Y()));
  d = sqrt((D.X()- A.X()) * (D.X() - A.X()) + (D.Y() - A.Y()) * (D.Y() - A.Y()));
  if(a != c || (B.X() - C.X()) / b != (A.X() - D.X()) / d || (B.Y() - C.Y()) / b != (A.Y() - D.Y()) / d)
    throw std::logic_error("It`s not a isosceles trapeze");
}



double Trapeze::area() const{
  double a = sqrt((C.X() - B.X()) * (C.X() - B.X()) + (B.Y() - C.Y()) * (B.Y() - C.Y()));
  double b = sqrt((B.X() - A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  double l = sqrt((D.X() - A.X()) * (D.X() - A.X()) + (A.Y() - D.Y()) * (A.Y() - D.Y()));
  double c = (l - a) / 2;
  double h = sqrt((b * b) - (c * c));
  return 0.5 * h * (a + l);

}

Point Trapeze::center() const
{
  double a = sqrt((C.X() - B.X()) * (C.X() - B.X()) + (B.Y() - C.Y()) * (B.Y() - C.Y()));
  double b = sqrt((B.X() - A.X()) * (B.X() - A.X()) + (B.Y() - A.Y()) * (B.Y() - A.Y()));
  double l = sqrt((D.X() - A.X()) * (D.X() - A.X()) + (A.Y() - D.Y()) * (A.Y() - D.Y()));
  double c = (l - a) / 2;
  double h = sqrt((b * b) - (c * c));
  double y_ = (2 * l + a) * h / (a + l) / 3;
  if (B.X() == C.X() && D.X() < C.X())
    return Point{D.X() + h - y_, (A.Y() + B.Y() + C.Y() + D.Y()) / 4};
  if (B.X() == C.X() && C.X() < D.X())
    return Point{C.X() + h - y_, (A.Y() + B.Y() + C.Y() + D.Y()) / 4};
  return Point{(A.X() + B.X() + C.X() + D.X()) / 4, (B.Y() + C.Y()) / 2 - ((B.Y() + C.Y()) / 2 - (D.Y() + A.Y()) / 2) *
y_ / h};
}

std::ostream& Trapeze::print(std::ostream& out) const
{
  out << A << " " << B << " " << C << " " << D;
  return out;
}



CMakeLists.txt

cmake_minimum_required (VERSION 3.5)

project(lab3)

add_executable(oop_exercise_03
  main.cpp
  figure.cpp
  rhombus.cpp
```

```
 point.cpp
 rectangle.cpp
 trapeze.cpp)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra")

set_target_properties(oop_exercise_03 PROPERTIES CXX_STANDART 14 CXX_STANDART_REQUIRED ON)
```

6. Набор testcases

test_01.txt	Координаты вершин	Ожидаемый результат	
Трапеция	(0;0) (1;1) (2;1) (3;0)	Площадь	2
0		Центр	(1.5; 0.41)
0			
1			
1			
2			
1			
3			
0			
Ромб	(0:2)(1;4)(2;2)(1;0)	Площадь	4
0		Центр	(1;2)
2			
1			
4			
2			
2			
1			
0			
Прямоугольник	(-4;0)(-4;5)(0;5)(0;0)	Площадь	20
-4		Центр	(-2;2.5)
0			
-4			
5			
0			
5			
0			
0			
out			
delete 1			
out			

test_02.txt	Координаты вершин	Ожидаемый результат	
Трапеция	(0;-2) (1;0) (4;0) (5;-2)	Площадь	8
0		Центр	(2.5; -1.08)
-2			
1			
0			
4			
0			
5			
-2			
Ромб	(0:-2)(-1;0)(0;2)(1;0)	Площадь	4
0		Центр	(0;0)
-2			

-1
0
0
2
1
0

Прямоугольник	(-2;-2)(-2;-1)(2;-1)(2;-2)	Площадь	4
-2		Центр	(0;-1.5)
-2			
-2			
-1			
2			
-1			
2			
-2			
out			
delete 1			
delete 1			
out			
delete 0			
out			

test_03.txt	Координаты вершин	Ожидаемый результат	
Трапеция	(0;4) (2;3) (2;1) (0;0)	Площадь	6
0		Центр	(0.89;2)
4			
2			
3			
2			
1			
0			
0			
Ромб	(0:1)(3;2)(6;1)(3;0)	Площадь	6
0		Центр	(3;1)
1			
3			
2			
6			
1			
3			
0			
Прямоугольник	(-4;0)(-4;5)(0;5)(0;0)	Площадь	8
0		Центр	(2;1)
2			
4			
2			
4			
0			
0			
0			
out			
delete 0			
out			

7. Результаты выполнения тестов

```
walien@PC-name:~/2kurs/CPP/lab3/tmp$ ./oop_exercise_03 < ~/2kurs/CPP/lab3/test_01.txt
    2    3       2     2  3
    *******       *     ****
```

```
   *********     ***    ****
  ********** 1 *****3   ****
 **********    ***    ****
**********    *     ****
 1          4   4     1  4
Operations: add / delete / out / quit
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
0:
Area: 2
Center: (1.5;0.416667)
Coordinates: (0;0) (1;1) (2;1) (3;0)
1:
Area: 4
Center: (1;2)
Coordinates: (0;2) (1;4) (2;2) (1;0)
2:
Area: 20
Center: (-2;2.5)
Coordinates: (-4;0) (-4;5) (0;5) (0;0)
Total area: 26
0:
Area: 2
Center: (1.5;0.416667)
Coordinates: (0;0) (1;1) (2;1) (3;0)
1:
Area: 20
Center: (-2;2.5)
Coordinates: (-4;0) (-4;5) (0;5) (0;0)
Total area: 22
walien@PC-name:~/2kurs/CPP/lab3/tmp$ ./oop_exercise_03 < ~/2kurs/CPP/lab3/test_02.txt
    2    3       2     2  3
    *******       *     ****
   *********     ***    ****
  ********** 1 *****3   ****
 **********    ***    ****
**********    *     ****
 1          4   4     1  4
Operations: add / delete / out / quit
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
0:
Area: 8
Center: (2.5;-1.08333)
Coordinates: (0;-2) (1;0) (4;0) (5;-2)
1:
Area: 4
Center: (0;0)
Coordinates: (0;-2) (-1;0) (0;2) (1;0)
2:
Area: 4
Center: (0;-1.5)
Coordinates: (-2;-2) (-2;-1) (2;-1) (2;-2)
Total area: 16
0:
Area: 8
Center: (2.5;-1.08333)
Coordinates: (0;-2) (1;0) (4;0) (5;-2)
Total area: 8
Total area: 0
walien@PC-name:~/2kurs/CPP/lab3/tmp$ ./oop_exercise_03 < ~/2kurs/CPP/lab3/test_03.txt
```

```
   2    3      2     2  3
   *******      *    ****
  *********     ***    ****
 **********  1 *****3   ****
 *********    ***    ****
*********     *     ****
 1          4   4     1  4
```

Operations: add / delete / out / quit
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
Fig types: 1 - trapeze; 2 - rhombus; 3 - rectangle
0:
Area: 6
Center: (0.888889;2)
Coordinates: (0;4) (2;3) (2;1) (0;0)
1:
Area: 6
Center: (3;1)
Coordinates: (0;1) (3;2) (6;1) (3;0)
2:
Area: 8
Center: (2;1)
Coordinates: (0;2) (4;2) (4;0) (0;0)
Total area: 20
0:
Area: 6
Center: (3;1)
Coordinates: (0;1) (3;2) (6;1) (3;0)
1:
Area: 8
Center: (2;1)
Coordinates: (0;2) (4;2) (4;0) (0;0)
Total area: 14

8. Объяснение результатов работы программы - вывод

в figure.h задаётся базовый класс Figure задающий общий принцип струтуры для классов — наследников — Rectangle, Trapeze и Rhombus.
Наследование позволяет избежать дублирования лишнего кода при написании классов, т. к. класс может использовать переменные и методы другого класса как свои собственные. В данном случае класс Figure является абстрактным — он определяет интерфес для переопределения методов классами Rectangle, Trapeze и Rhombus.
Полиморфизм позволяет использовать одно и то же имя для различных действий, похожих, но технически отличающихся. В данной лаборторной работе он осуществляется посредством виртуальных функций.