**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Наследование,полиморфизм

Студент: Минибаев Айдар

Группа: 80-201Б-18

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. **Постановка задачи**

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

1. Вычисление геометрического центра фигуры;

2. Вывод в стандартный поток вывода std::cout координат вершин фигуры; 3. Вычисление площади фигуры;

Создать программу, которая позволяет:

• Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.

• Сохранять созданные фигуры в динамический массив std::vector<Figure\*>

• Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.

• Необходимо уметь вычислять общую площадь фигур в массиве.

• Удалять из массива фигуру по индексу;

Вариант 9:

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Треугольник | Квадрат | Прямоугольник |

1. **Описание программы**

Программа содержит вектор геометрических фигур. Пользователь может добавлять фигуры в конец вектора, удалять фигуры по индексу, менять фигуры местами, очищать вектор, выводить геометрический центр, площадь, список вершин для конкретной фигуры и для всех фигур в векторе.

Реализовал абстрактный класс Figure() с виртуальными функциями Read(), Print(), Area(), Centre(), которые затем определял для каждого класса фигур. Реализовал три производных класса от класса Figure(): Triangle(), Square(), Rectangle().

Фигуры вводятся по координатам в плоскости xOy. Для этого реализовал класс Point() и функцию ввода координат readPoint().

Для работы с вектором фигур реализовал функцию вывода всех фигур с данными PrintVec() и функцию суммирования площадей фигур TotalArea(), лежащих в векторе.

В основной функции создается указатель на объект класса фигуры, который в процессе выполнения программы может указывать адреса объектов производных классов(квадраты, треугольники). Реализовал интерактивное меню и различные исключения.

1. **Набор testcases**

Тестовые файлы: test\_\_01.txt, test\_\_02.txt. Тесты представляют из себя набор команд , которые демонстрируют работу программы. Первое число задает операцию из предложенных в интерактивном меню. Если вызван первый пункт меню и выбрана фигура, то затем вбиваются координаты указанной точки фигуры, так для треугольника необходима вбить 3 координаты, а для квадрата и прямоугольника достаточно 2 координат. При вызове операции удаления из вектора нужно ввести n-ую фигуру, которую необходимо удалить.

test\_01.txt

1

1

1

2

5

2

3

5

1

2

0

0

2

2

1

3

0

0

4

3

3

5

1

3

2

0

Проверка всех операций над фигурами.

test\_02.txt

1

1

fdg

02f

0

0

1

1

2

2

4

8

0

7

11

4

1

2

0

0

3

4

1

1

3

3

2

1

3

3

134fdg

0

0

1

1

0

0

4

2

6

5

11

Проверка попыток неккоректного ввода.

1. **Результаты выполнения тестов.**

Все тесты успешно пройдены, программа выдает верные результаты.

**test\_01.txt**

./oop\_exercise\_03

Enter a number for action:

1) Add Figure

2) Display data of all figures

3) Print total area

4) Remove all figure

5) Remove figure by index

5) Print Menu

0) Exit

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

1

Enter the three coordinates of the triangle

Enter the coordinates of point a by x, y :

x = 1

y = 2

For point b:

x = 5

y = 2

For point c:

x = 3

y = 5

Triangle added

Triangle:

Peaks: A(1;2), B(5;2), C(3;5)

Geometric center: O(3;3)

Area: 6

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

2

Enter the two coordinates of the Square

Enter the coordinates lower left point a by x, y :

x = 0

y = 0

For right upper point СЃ:

x = 2

y = 2

Square added

Square:

Peaks: A(0;0), B(0;2), C(2;2), D(2;0)

Geometric center: O(1;1)

Area: 4

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

3

Enter the two coordinates of the Rectangle

Enter the coordinates lower left point a by x, y :

x = 0

y = 0

For right upper point СЃ:

x = 4

y = 3

Rectangle added

Rectangle:

Peaks: A(0;0), B(0;3), C(4;3), D(4;0)

Geometric center: O(2;1.5)

Area: 12

Enter a number for action:

3

The total area of all shapes is 22

Enter a number for action:

5

Number of the shape: 1

Enter a number for action:

3

The total area of all shapes is 16

Enter a number for action:

2

1.

Square:

Peaks: A(0;0), B(0;2), C(2;2), D(2;0)

Geometric center: O(1;1)

Area: 4

2.

Rectangle:

Peaks: A(0;0), B(0;3), C(4;3), D(4;0)

Geometric center: O(2;1.5)

Area: 12

Enter a number for action:

0

Process finished with exit code 0

**test\_02.txt**

./oop\_exercise\_03

Enter a number for action:

1) Add Figure

2) Display data of all figures

3) Print total area

4) Remove all figure

5) Remove figure by index

5) Print Menu

0) Exit

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

1

Enter the three coordinates of the triangle

Enter the coordinates of point a by x, y :

x = fdg

y = dgf

Incorrect input of coordinates

Re-enter the coordinates of this point

x = 0

y = 0

For point b:

x = 1

y = 1

For point c:

x = 2

y = 2

the triangle is incorrect

Re-enter the triangle parameters

Enter the coordinates of point a by x, y :

x = 4

y = 8

For point b:

x = 0

y = 7

For point c:

x = 11

y = 4

Triangle added

Triangle:

Peaks: A(4;8), B(0;7), C(11;4)

Geometric center: O(5;6.33333)

Area: 11.5

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

2

Enter the two coordinates of the Square

Enter the coordinates lower left point a by x, y :

x = 0

y = 0

For right upper point С:

x = 3

y = 4

the square is incorrect

Re-enter the square parameters

Enter the coordinates lower left point a by x, y :

x = 1

y = 1

For right upper point С:

x = 3

y = 3

Square added

Square:

Peaks: A(1;1), B(1;3), C(3;3), D(3;1)

Geometric center: O(2;2)

Area: 4

Enter a number for action:

2

1.

Triangle:

Peaks: A(4;8), B(0;7), C(11;4)

Geometric center: O(5;6.33333)

Area: 11.5

2.

Square:

Peaks: A(1;1), B(1;3), C(3;3), D(3;1)

Geometric center: O(2;2)

Area: 4

Enter a number for action:

1

1) Triangle

2) Square

3) Rectangle

3

Enter the two coordinates of the Rectangle

Enter the coordinates lower left point a by x, y :

x = 3

y = 134fdg

Incorrect input of coordinates

Re-enter the coordinates of this point

x = 0

y = 0

For right upper point С:

x = 1

y = 1

the rectangle is incorrect

Re-enter the rectangle parameters

Enter the coordinates lower left point a by x, y :

x = 0

y = 0

For right upper point С:

x = 4

y = 2

Rectangle added

Rectangle:

Peaks: A(0;0), B(0;2), C(4;2), D(4;0)

Geometric center: O(2;1)

Area: 8

Enter a number for action:

6

Enter a number for action:

1) Add Figure

2) Display data of all figures

3) Print total area

4) Remove all figure

5) Remove figure by index

5) Print Menu

0) Exit

Enter a number for action:

5

Number of the shape: 11

libc++abi.dylib: terminating with uncaught exception of type std::runtime\_error: Incorrect index

Process finished with exit code 6

1. **Листинг программы**

#include <vector>

#include <iostream>

#include <exception>

#include <cmath>

/\*

\* Минибаев Айдар М8О-201Б

\* ========================

\* Вариант 9:

\* Треугольник

\* Квадрат

\* Прямоугольник

\*/

class Point {

public:

double x, y;

Point(double a = 0, double b = 0) : x(a), y(b) {}

Point readPoint();

friend double dist(Point p1, Point p2){

return sqrt(((p1.x - p2.x) \* (p1.x - p2.x)) + ((p1.y - p2.y) \* (p1.y - p2.y)));

}

};

bool isNumber(const std::string& s){

return !s.empty() && s.find\_first\_not\_of("-.0123456789") == std::string::npos; }

Point Point::readPoint(){

double a,b;

std::string x, y;

bool check = true;

while(check) {

std::cout << "x = ";

std::cin >> x;

std::cout << "y = ";

std::cin >> y;

if (isNumber(x) && isNumber(y)) {

a = std::stod(x);

b = std::stod(y);

Point C(a, b);

check = false;

return C;

} else {

std::cout << "Incorrect input of coordinates" << std::endl;

std::cout << "Re-enter the coordinates of this point" << std::endl;

}

}

}

class Figure {

protected:

Point centre;

double area;

std::string name = "";

public:

virtual double Area() = 0;

virtual Point Centre() = 0;

virtual std::ostream &Print(std::ostream &out) = 0;

virtual std::istream &Read(std::istream &in) = 0;

virtual ~Figure() = default;

};

class Triangle: public Figure{

Point A, B, C;

public:

Triangle() {}

Triangle(Point a, Point b, Point c);

Point Centre() override;

double Area() override;

std::ostream &Print(std::ostream &out) override;

std::istream &Read(std::istream &in) override;

};

class Square: public Figure{

Point A, B, C, D;

public:

Square() {}

Square(Point a, Point c, Point b, Point d);

Point Centre() override;

double Area() override;

std::ostream &Print(std::ostream &out) override;

std::istream &Read(std::istream &in) override;

};

class Rectangle: public Figure{

Point A, B, C, D;

public:

Rectangle() {}

Rectangle(Point a, Point c, Point b, Point d);

Point Centre() override;

double Area() override;

std::ostream &Print(std::ostream &out) override;

std::istream &Read(std::istream &in) override;

};

Triangle::Triangle(Point a, Point b, Point c) {

name = "Triangle";

A = a; B = b; C = c;

area = Area();

centre = Centre();

}

Point Triangle::Centre() {

Point O;

O.x = (A.x + B.x + C.x) / 3;

O.y = (A.y + B.y + C.y) / 3;

return O;

}

double Triangle::Area() {

double S, p, AB, AC, BC;

AB = dist(A,B);

AC = dist(A,C);

BC = dist(B,C);

p = (AB + BC + AC) / 2;

S = sqrt(p\* (p-AB) \* (p-AC) \* (p-BC));

return S;

}

std::ostream &Triangle::Print(std::ostream &out) {

out << std::endl << name << ":" << std::endl;

out << "Peaks: " << "A(" << A.x << ";" << A.y << "), B("

<< B.x << ";" << B.y << "), C(" << C.x << ";" << C.y << ")" << std::endl;

out << "Geometric center: O(" << centre.x << ";" << centre.y << ")" << std::endl;

out << "Area: " << area << std::endl;

out << std::endl;

return out;

}

std::istream &Triangle::Read(std::istream &in) {

Point a, b, c;

bool k = true;

double AB, BC, AC;

std::cout << "Enter the three coordinates of the triangle" << std::endl;

while(k){

std::cout << "Enter the coordinates of point a by x, y :" << std::endl;

a = a.readPoint();

std::cout << "For point b:" << std::endl;

b = b.readPoint();

std::cout << "For point c:" << std::endl;

c = c.readPoint();

AB = dist(a, b);

AC = dist(a, c);

BC = dist(b, c);

if (((AB + AC) > BC) && ((AB + BC) > AC) && ((BC + AC) > AB)) {

(\*this) = Triangle(a, b, c);

return in;

k = false;

} else {

std::cout << "the triangle is incorrect" << std::endl;

std::cout << "Re-enter the triangle parameters" << std::endl;

}

}

}

Square::Square(Point a, Point c, Point b, Point d){

name = "Square";

A = a; C = c; B = b; D = d;

area = Area();

centre = Centre();

}

Point Square::Centre() {

Point O;

O.x = (A.x + C.x) / 2;

O.y = (A.y + C.y) / 2;

return O;

}

double Square::Area() {

double S, AB;

AB = dist(A,B);

S = AB \* AB;

return S;

}

std::ostream &Square::Print(std::ostream &out) {

out << std::endl << name << ":" << std::endl;

out << "Peaks: " << "A(" << A.x << ";" << A.y << "), B("

<< B.x << ";" << B.y << "), C(" << C.x << ";" << C.y << "), D("

<< D.x << ";" << D.y << ")" << std::endl;

out << "Geometric center: O(" << centre.x << ";" << centre.y << ")" << std::endl;

out << "Area: " << area << std::endl;

out << std::endl;

return out;

}

std::istream &Square::Read(std::istream &in) {

Point a, c;

bool k = true;

double AB, AD;

std::cout << "Enter the two coordinates of the Square" << std::endl;

while(k){

std::cout << "Enter the coordinates lower left point a by x, y :" << std::endl;

a = a.readPoint();

std::cout << "For right upper point СЃ:" << std::endl;

c = c.readPoint();

Point b(a.x, c.y), d(c.x, a.y);

AB = dist(a,b);

AD = dist(a,d);

if ((a.x < c.x) && (a.y < c.y) && (AB == AD)) {

(\*this) = Square(a, c, b, d);

return in;

k = false;

} else {

std::cout << "the square is incorrect" << std::endl;

std::cout << "Re-enter the square parameters" << std::endl;

}

}

}

Rectangle::Rectangle(Point a, Point c, Point b, Point d) {

name = "Rectangle";

A = a; C = c; B = b; D = d;

area = Area();

centre = Centre();

}

Point Rectangle::Centre() {

Point O;

O.x = (A.x + C.x) / 2;

O.y = (A.y + C.y) / 2;

return O;

}

double Rectangle::Area() {

double S, AB, AD;

AB = dist(A,B);

AD = dist(A,D);

S = AB \* AD;

return S;

}

std::ostream &Rectangle::Print(std::ostream &out) {

out << std::endl << name << ":" << std::endl;

out << "Peaks: " << "A(" << A.x << ";" << A.y << "), B("

<< B.x << ";" << B.y << "), C(" << C.x << ";" << C.y << "), D("

<< D.x << ";" << D.y << ")" << std::endl;

out << "Geometric center: O(" << centre.x << ";" << centre.y << ")" << std::endl;

out << "Area: " << area << std::endl;

out << std::endl;

return out;

}

std::istream &Rectangle::Read(std::istream &in) {

Point a, c;

bool k = true;

double AB, AD;

std::cout << "Enter the two coordinates of the Rectangle" << std::endl;

while(k){

std::cout << "Enter the coordinates lower left point a by x, y :" << std::endl;

a = a.readPoint();

std::cout << "For right upper point СЃ:" << std::endl;

c = c.readPoint();

Point b(a.x,c.y), d(c.x, a.y);

AB = dist(a,b);

AD = dist(a,d);

if ((a.x < c.x) && (a.y < c.y) && (AB != AD)) {

(\*this) = Rectangle(a, c, b, d);

return in;

k = false;

} else {

std::cout << "the rectangle is incorrect" << std::endl;

std::cout << "Re-enter the rectangle parameters" << std::endl;

}

}

}

struct list\_empty : public std::exception {

const char\* what() const throw ()

{

return "The list is currently empty";

}

};

void PrintVec(const std::vector<Figure\*> &l) {

int num = 1;

for (auto e : l) {

std::cout << num << ". ";

e->Print(std::cout);

std::cout << std::endl;

++num;

}

}

double TotalArea(const std::vector<Figure\*>& l) {

double res = 0;

for (auto &e : l) {

res += e->Area();

}

return res;

}

int main() {

std::vector<Figure\*> FigureVec;

bool check = true;

int command = 0, option = 6;

Figure \*fig;

while(check) {

switch(option) {

case 0:

check = false;

break;

case 1:

std::cout << "1) Triangle" << std::endl;

std::cout << "2) Square" << std::endl;

std::cout << "3) Rectangle" << std::endl;

std::cin >> command;

switch (command) {

case 0: break;

case 1:

fig = new Triangle;

fig->Read(std::cin);

FigureVec.push\_back(fig);

std::cout << "Triangle added" << std::endl;

fig->Print(std::cout);

break;

case 2:

fig = new Square();

fig->Read(std::cin);

FigureVec.push\_back(fig);

std::cout << "Square added" << std::endl;

fig->Print(std::cout);

break;

case 3:

fig = new Rectangle();

fig->Read(std::cin);

FigureVec.push\_back(fig);

std::cout << "Rectangle added" << std::endl;

fig->Print(std::cout);

break;

}

break;

case 2:

if(FigureVec.empty()) throw list\_empty();

PrintVec(FigureVec);

break;

case 3 :

if (FigureVec.empty()) throw list\_empty();

std::cout << "The total area of all shapes is " << TotalArea(FigureVec) << std::endl;

break;

case 4:

if (FigureVec.empty()) throw list\_empty();

for (std::vector<Figure \*>::iterator i = FigureVec.begin(); i != FigureVec.end(); i++) {

delete \*i;

}

FigureVec.erase(FigureVec.begin(), FigureVec.end());

std::cout << "Vector is now empty" << std::endl;

break;

case 5 :

int num;

if (FigureVec.empty()) throw list\_empty();

std::cout << "Number of the shape: ";

std::cin >> num;

if (num < 1 || num > FigureVec.size()) throw std::runtime\_error("Incorrect index");

delete FigureVec[--num];

FigureVec.erase(FigureVec.begin() + num);

break;

case 6:

std::cout << "Enter a number for action:" << std::endl;

std::cout << "1) Add Figure" <<

std::endl << "2) Display data of all figures" <<

std::endl << "3) Print total area" <<

std::endl << "4) Remove all figure" <<

std::endl << "5) Remove figure by index" <<

std::endl << "5) Print Menu" <<

std::endl << "0) Exit" << std::endl;

break;

}

if(check){

std::cout << "Enter a number for action:" << std::endl;

std::cin >> option;

}

}

return 0;

}

1. **Вывод**

Научился использовать наследование в C++ для реализации иерархии классов.

**СПИСОК ЛИТЕРАТУРЫ**

1. Наследование в C++ [электронный ресурс]. URL: <https://habr.com/ru/post/445948/>
2. std::vector [электронный ресурс]. URL: <https://ru.cppreference.com/w/cpp/container/vector>