**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: Черемисинов Максим (староста)

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. **Постановка задачи**

Написать программу с базовым классом Figure и производными классами квадрата, прямоугольника с трапеции, которые наследуются от класса Figure. Должна быть возможность вычисления центра фигуры и ее площади при помощи переопределения виртуальных функций родительского класса.

1. **Репозиторий github**

[https://github.com/devepodete/oop\_exercise\_03](https://github.com/devepodete/oop_exercise_02)

1. **Описание программы**

В родительском классе некоторые функции объявлены чисто виртуальными, которые переопределяются в классах-наследниках. К фигуре могут быть применены методы: вычислить квадрат стороны, вычислить центр фигуры, вычислить площадь, распечатать координаты вершин, координаты центра, площадь.

Все объекты добавляются в std::vector<Figure\*>. Далее в зависимости от действия над ними всеми выполняется необходимое действие. Также реализована возможность удаления элемента по заданному индексу.

Точки необходимо задавать по-порядку.

1. **Набор testcases**

**test\_\_01.txt**

1 1

0 0 4 0 4 4 0 4

1 2

1 0 2 0 2 5 1 5

1 3

0 0 5 0 4 3 1 3

2 1 2 2

2 3 2 4 2 5

Создание трёх различных фигур. Проверка вычисления координат центра и их площадей.

**test\_\_02.txt**

1 1

0 0 4 0 4 4 0 4

1 2

1 0 2 0 2 5 1 5

1 3

0 0 5 0 4 3 1 3

3 4 3 1

2 5

Проверка удаления элемента по заданному индексу.

1. **Результаты выполнения тестов.**

Все тесты успешно пройдены, программа выдает правильные ответы, корректно обрабатывает время.

1. **Листинг программы**

/\*

Написать программу с базовым классом Figure и производными

классами квадрата, прямоугольника и трапеции, которые наследуются от класса

Figure. Должна быть возможность вычисления центра фигуры и ее площади при помощи

переопределения виртуальных функций родительского класса.

Черемисинов Максим

М8О-201Б-18

\*/

#include <iostream>

#include <vector>

#include <cmath>

class Figure{

protected:

double area\_;

double xCenter\_, yCenter\_;

std::vector<double> X;

std::vector<double> Y;

public:

virtual void calculateCenter() = 0;

virtual void calculateArea() = 0;

virtual void printCords() = 0;

void printCenter(){

std::cout << xCenter\_ << " " << yCenter\_ << std::endl;

}

void printArea(){

std::cout << area\_ << std::endl;

}

double sideSquare(double x1, double y1, double x2, double y2){

return (x1-x2)\*(x1-x2) + (y1-y2)\*(y1-y2);

}

Figure(){}

virtual ~Figure(){}

};

class Square: public Figure{

public:

Square(){}

Square(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){

X.push\_back(X1);

X.push\_back(X2);

X.push\_back(X3);

X.push\_back(X4);

Y.push\_back(Y1);

Y.push\_back(Y2);

Y.push\_back(Y3);

Y.push\_back(Y4);

}

~Square(){}

void calculateArea()override{

if(X.size() < 4){

area\_ = 0;

}

area\_ = sideSquare(X[0], Y[0], X[1], Y[1]);

}

//через среднее арифметическое

void calculateCenter(){

xCenter\_ = yCenter\_ = 0;

for(int i = 0; i < X.size(); i++){

xCenter\_ += X[i];

yCenter\_ += Y[i];

}

xCenter\_ = xCenter\_/X.size();

yCenter\_ = yCenter\_/Y.size();

}

void printCords(){

std::cout << "Square: ";

for(int i = 0; i < X.size(); i++){

std::cout << "<" << X[i] << ", " << Y[i] << "> ";

}

std::cout << std::endl;

}

};

class Rectangle: public Figure{

public:

Rectangle(){}

Rectangle(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){

X.push\_back(X1);

X.push\_back(X2);

X.push\_back(X3);

X.push\_back(X4);

Y.push\_back(Y1);

Y.push\_back(Y2);

Y.push\_back(Y3);

Y.push\_back(Y4);

}

~Rectangle(){}

void calculateArea()override{

if(X.size() < 4){

return;

}

double area1 = abs((X[0]-X[2])\*(Y[1]-Y[2])-(Y[0]-Y[2])\*(X[1]-X[2]));

double area2 = abs((X[0]-X[2])\*(Y[3]-Y[2])-(Y[0]-Y[2])\*(X[3]-X[2]));

area\_ = 0.5\*(area1 + area2);

}

//через окружность

void calculateCenter()override{

if(X.size() < 4){

return;

}

if(X[0] == X[1]){

std::swap(X[1], X[2]);

std::swap(Y[1], Y[2]);

}

yCenter\_ = ((X[0]-X[1])\*(X[0]\*X[0]+Y[0]\*Y[0]-X[2]\*X[2]-Y[2]\*Y[2])-(X[0]-X[2])\*(X[0]\*X[0]+Y[0]\*Y[0]-X[1]\*X[1]-Y[1]\*Y[1]))/(2\*((Y[0]-Y[2])\*(X[0]-X[1])-(X[0]-X[2])\*(Y[0]-Y[1])));

xCenter\_ = (X[0]\*X[0]+Y[0]\*Y[0]-X[1]\*X[1]-Y[1]\*Y[1]-2\*yCenter\_\*(Y[0]-Y[1]))/(2\*(X[0]-X[1]));

}

void printCords(){

std::cout << "Rectangle: ";

for(int i = 0; i < X.size(); i++){

std::cout << "<" << X[i] << ", " << Y[i] << "> ";

}

std::cout << std::endl;

}

};

class Trapezoid: public Figure{

public:

Trapezoid(){}

Trapezoid(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){

X.push\_back(X1);

X.push\_back(X2);

X.push\_back(X3);

X.push\_back(X4);

Y.push\_back(Y1);

Y.push\_back(Y2);

Y.push\_back(Y3);

Y.push\_back(Y4);

}

~Trapezoid(){}

void calculateArea()override{

area\_ = 0;

for(int i = 0; i < X.size()-1; i++){

area\_ += X[i]\*Y[i+1]-X[i+1]\*Y[i];

}

area\_ \*= 0.5;

}

//общая формула для n-угольника

void calculateCenter()override{

if(area\_ == 0){

return;

}

xCenter\_ = yCenter\_ = 0;

for(int i = 0; i < X.size(); i++){

if(i == X.size()-1){

xCenter\_ += (X[i]+X[0])\*(X[i]\*Y[0]-X[0]\*Y[i]);

yCenter\_ += (Y[i]+Y[0])\*(X[i]\*Y[0]-X[0]\*Y[i]);

}else{

xCenter\_ += (X[i]+X[i+1])\*(X[i]\*Y[i+1]-X[i+1]\*Y[i]);

yCenter\_ += (Y[i]+Y[i+1])\*(X[i]\*Y[i+1]-X[i+1]\*Y[i]);

}

}

xCenter\_ = xCenter\_/(6\*area\_);

yCenter\_ = yCenter\_/(6\*area\_);

}

void printCords(){

std::cout << "Trapezoid: ";

for(int i = 0; i < X.size(); i++){

std::cout << "<" << X[i] << ", " << Y[i] << "> ";

}

std::cout << std::endl;

}

};

void printMenu(){

std::cout << "Доступные команды:" << std::endl;

std::cout << "0. Выход" << std::endl;

std::cout << "1. Добавить фигуру" << std::endl;

std::cout << "2. Вызвать функцию для всех фигур" << std::endl;

std::cout << "3. Удалить фигуру по индексу" << std::endl;

std::cout << "4. Вывести это меню" << std::endl;

}

int main(){

std::vector<Figure\*> v;

printMenu();

while(true){

std::cout << "Номер: ";

int k;

std::cin >> k;

switch(k){

case 0:{

for(size\_t i = 0; i < v.size(); i++){

delete v[i];

}

return 0;

}

case 1:{

std::cout << "1. Квадрат" << std::endl;

std::cout << "2. Прямоугольник" << std::endl;

std::cout << "3. Трапеция" << std::endl;

std::cout << "Номер: ";

int a;

std::cin >> a;

if(a < 1 || a > 3){

std::cout << "Неверный номер" << std::endl;

break;

}

std::cout << "Введите координаты: ";

double x1, y1, x2, y2, x3, y3, x4, y4;

std::cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;

Figure \*s;

switch(a){

case 1:

s = new Square(x1, y1, x2, y2, x3, y3, x4, y4);

break;

case 2:

s = new Rectangle(x1, y1, x2, y2, x3, y3, x4, y4);

break;

case 3:

s = new Trapezoid(x1, y1, x2, y2, x3, y3, x4, y4);

break;

}

v.push\_back(s);

break;

}

case 2:{

std::cout << "1. Посчитать площадь" << std::endl;

std::cout << "2. Посчитать центр" << std::endl;

std::cout << "3. Распечатать площадь" << std::endl;

std::cout << "4. Распечатать центр" << std::endl;

std::cout << "5. Распечатать координаты" << std::endl;

std::cout << "Номер: ";

int a;

std::cin >> a;

if(a < 1 || a > 5){

std::cout << "Неверный номер" << std::endl;

break;

}

switch(a){

case 1:{

for(int i = 0; i < v.size(); i++){

(\*v[i]).calculateArea();

}

break;

}

case 2:{

for(int i = 0; i < v.size(); i++){

(\*v[i]).calculateCenter();

}

break;

}

case 3:{

for(int i = 0; i < v.size(); i++){

(\*v[i]).printArea();

}

break;

}

case 4:{

for(int i = 0; i < v.size(); i++){

(\*v[i]).printCenter();

}

break;

}

case 5:{

for(int i = 0; i < v.size(); i++){

(\*v[i]).printCords();

}

break;

}

}

break;

}

case 3:{

std::cout << "Индекс: ";

size\_t id;

std::cin >> id;

if(id < 0 || id > v.size()-1){

std::cout << "Индекс выходит за границы массива" << std::endl;

break;

}

std::vector<Figure\*> next;

for(int i = 0; i < v.size(); i++){

if(i == id){

continue;

}

next.push\_back(v[i]);

}

std::swap(v, next);

break;

}

case 4:{

printMenu();

break;

}

}

}

}

1. **Вывод**

Реализовал программу с абстрактным классом-родителем и классами-потомками для создания квадратов, прямоугольников и трапеций. Благодаря динамическому полиморфизму на этапе работы программы определяется, какая функция должна быть вызвана для каждого объекта, на который указывает указатель типа класса-родителя.

**Список литературы**

1. Шилдт, Герберт. С++: базовый курс, 3-е изд. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2018. - 624 с. : ил. - Парал. тит. англ.
2. Справочник по языку С++ [Электронный ресурс]. URL: [http://www.cplusplus.com/reference](http://www.cplusplus.com/reference/deque/) (дата обращения: 01.10.2019).