**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Механизмы наследования в С++

Студент: Почечура Артемий Андреевич

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure.

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.

Вариант 12: трапеция, ромб, пятиугольник.

1. **Описание программы**

В данной программе реализован принцип наследования классов на примере фигур. От класса Figure в моей работе наследуются классы Trapezoid, Rhombus и Pentagon. Путём интерактивного ввода определяется, какое действие будет совершено: ввод точек трапеции, ромба или пятиугольника, удаление из динамического вектора фигуры на указанной позиции и вывод координат всех точек, площадь и координаты центра каждой фигуры, лежащей в векторе. При вводе происходит врпроверка, является ли данная фигура трапецией (проверка на параллельность сторон), ромбом (параллельность сторон и их равенство) или пятиугольником (сумма углов). Все координаты должны вводиться по часовой стрелке.

1. **Набор тестов и результаты их исполнения**

Тест 1:

0 - exit

1 - select a trapezoid

2 - select a rhombus

3 - select a pentagon

4 - delete an element of vector

5 - print centre, square and points of each figure of vector

1

0 0

1 1

2 1

3 0

2

3 0

0 -2

-3 0

0 2

5

centre: 1.5 0.5

area: 2

points of figure:

0 0

1 1

2 1

3 0

centre: 0 0

area: 12

points of figure:

3 0

0 -2

-3 0

0 2

4

select a position

1

5

centre: 0 0

area: 12

points of figure:

3 0

0 -2

-3 0

0 2

Тест 2:

0 - exit

1 - select a trapezoid

2 - select a rhombus

3 - select a pentagon

4 - delete an element of vector

5 - print centre, square and points of each figure of vector

1

0 0

1 1

2 0

4 1

Wrong selection

Тест 3:

0 - exit

1 - select a trapezoid

2 - select a rhombus

3 - select a pentagon

4 - delete an element of vector

5 - print centre, square and points of each figure of vector

4

vector is empty

5

2

0 3

2 0

-2 0

0 -3

Wrong selection

1. **Листинг программы**

// Почечура Артемий Андреевич

// М8О-206Б-20

// Лабораторная работа №3

// Вариант 12: Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры: трапеция, ромб, пятиугольник.

#include <iostream>

#include <math.h>

#include <string>

#include <vector>

#include <algorithm>

#include <map>

#include <queue>

#include <stack>

#include <set>

using namespace std;

double koef\_nakl(pair<double,double> a, pair<double,double> b) {

double k = (a.second - b.second) / (a.first - b.first);

return k;

}

bool trapezoid\_check(pair<double,double> a, pair<double,double> b, pair<double,double> c, pair<double,double> d) {

double Area= (a.first\*b.second+b.first\*c.second+c.first\*d.second+d.first\*a.second- a.second\*b.first-b.second\*c.first-c.second\*d.first-d.second\*a.first)/2;

if (Area == 0) {

return false;

}

if (koef\_nakl(a, b) == koef\_nakl(c, d)) { return true; }

else if (koef\_nakl(b, c) == koef\_nakl(a, d)) { return true; }

return false;

}

bool rhombus\_check(pair<double,double> a, pair<double,double> b, pair<double,double> c, pair<double,double> d) {

double Area= (a.first\*b.second+b.first\*c.second+c.first\*d.second+d.first\*a.second-

a.second\*b.first-b.second\*c.first-c.second\*d.first-d.second\*a.first)/2;

if (Area == 0) {

return false;

}

if (koef\_nakl(a, b) == koef\_nakl(c, d) && ((a.first\*-b.first)\*(a.first\*-b.first)+(a.second\*-b.second))==((c.first\*-d.first)\*(c.first\*-d.first)+(c.second\*-d.second))) { return true; } else if (koef\_nakl(b, c) == koef\_nakl(a, d) && ((b.first\*-c.first)\*(b.first\*-c.first)+(b.second\*-c.second))==((a.first\*-d.first)\*(a.first\*-d.first)+(a.second\*-d.second))) { return true; }

return false;

}

long long cross(pair<double,double> a, pair<double,double> b) {

return a.first \* b.second - a.second \* b.first;

}

bool operator < (pair<double,double> a, pair<double,double> b) {

return cross(a, b) < 0;

}

class Figure{

public:

virtual double area()=0;

virtual pair<double, double> centre()=0;

virtual bool select()=0;

virtual void print()=0;

virtual ~Figure(){}

};

class Trapezoid: public Figure{

pair<double,double> v[4];

public:

bool select(){

for(int i=0;i<4;i++){

cin >> v[i].first >> v[i].second;

}

if(!trapezoid\_check(v[0],v[1],v[2],v[3])){

return false;

}

return true;

}

void print(){

for(int i=0;i<4;i++){

cout << v[i].first << " " << v[i].second << "\n";

}

}

pair<double, double> centre(){

pair<double, double> a;

a.first=0;

a.second=0;

for(int i=0;i<4;i++){

a.first=a.first+v[i].first;

a.second=a.second+v[i].second;

}

a.first=a.first/4;

a.second=a.second/4;

return a;

}

double area(){

return abs(v[0].first\*v[1].second+v[1].first\*v[2].second+v[2].first\*v[3].second+v[3].first\*v[0].second-v[0].second\*v[1].first-v[1].second\*v[2].first-v[2].second\*v[3].first-v[3].second\*v[0].first)/2;

}

~Trapezoid(){};

};

class Rhombus: public Figure{

pair<double, double> v[4];

public:

bool select(){

for(int i=0;i<4;i++){

cin >> v[i].first >> v[i].second;

}

if(!rhombus\_check(v[0],v[1],v[2],v[3])){

return false;

}

return true;

}

void print(){

for(int i=0;i<4;i++){

cout << v[i].first << " " << v[i].second << "\n";

}

}

pair<double, double> centre(){

pair<double, double> a;

a.first=0;

a.second=0;

for(int i=0;i<4;i++){

a.first=a.first+v[i].first;

a.second=a.second+v[i].second;

}

a.first=a.first/4;

a.second=a.second/4;

return a;

}

double area(){

return abs(v[0].first\*v[1].second+v[1].first\*v[2].second+v[2].first\*v[3].second+v[3].first\*v[0].second-v[0].second\*v[1].first-v[1].second\*v[2].first-v[2].second\*v[3].first-v[3].second\*v[0].first)/2;

}

~Rhombus(){};

};

class Pentagon: public Figure{

pair<double, double> v[5];

public:

bool select(){

for(int i=0;i<5;i++){

cin >> v[i].first >> v[i].second;

}

for(int i = 0; i < 5; i++){

if(v[i] < v[(i+1)%5]){

return false;

}

}

for(int i = 0; i < 5; i++){

double h=acos((v[(i+1)%5].first-v[i].first)\*(v[(i+2)%5].first-v[(i+1)%5].first)+(v[(i+1)%5].second-v[i].second)\*(v[(i+2)%5].second-v[(i+1)%5].second));

if(h!=0,942){

return false;

}

}

return true;

}

void print(){

for(int i=0;i<5;i++){

cout << v[i].first << " " << v[i].second << "\n";

}

}

pair<double, double> centre(){

pair<double, double> a;

a.first=0;

a.second=0;

for(int i=0;i<4;i++){

a.first=a.first+v[i].first;

a.second=a.second+v[i].second;

}

a.first=a.first/4;

a.second=a.second/4;

return a;

}

double area(){

return abs(v[0].first\*v[1].second+v[1].first\*v[2].second+v[2].first\*v[3].second+v[3].first\*v[4].second+v[4].first\*v[0].second-

v[0].second\*v[1].first-v[1].second\*v[2].first-v[2].second\*v[3].first-v[3].second\*v[4].first-v[4].second\*v[0].first)/2;

}

~Pentagon(){};

};

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr); cout.tie(nullptr);

cout << "0 - exit\n";

cout << "1 - select a trapezoid\n";

cout << "2 - select a rhombus\n";

cout << "3 - select a pentagon\n";

cout << "4 - delete an element of vector\n";

cout << "5 - print centre, square and points of each figure of vector\n";

vector<Figure\*> v;

int n=50;

while(n!=0){

if(!(cin >> n)){

cout << "Wrong values\n";

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

delete v[i];

}

return 0;

} else if(n==1){

Figure \*A=new Trapezoid;

if(!(\*A).select()){

cout << "Wrong selection\n";

return 0;

}

v.push\_back(A);

} else if(n==2) {

Figure \*A=new Rhombus;

if(!(\*A).select()){

cout << "Wrong selection\n";

return 0;

}

v.push\_back(A);

} else if(n==3){

Figure \*A=new Pentagon;

if(!(\*A).select()){

cout << "Wrong selection\n";

return 0;

}

v.push\_back(A);

} else if(n==4){

if(v.size()!=0){

int i;

cout << "select a position\n";

cin >> i;

v.erase(v.begin()+i-1);

} else {

cout << "vector is empty\n";

}

} else if(n==5){

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

pair<double, double> a=(\*v[i]).centre();

cout << "centre: " << a.first << " " << a.second << "\n" << "area: " << (\*v[i]).area() << "\n" << "points of figure:\n";

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

}

}

}

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

delete v[i];

}

return 0;

}

1. **Вывод**

В ходе данной работы я реализовал родительский класс Figure и дочерние классы Trapezoid, Rhombus и Pentagon для работы с абстрактными фигурами, трапециями, ромбами и пятиугольниками соответственно. Реализовал набор функций и операторов для использования данных классов.

**ЛИТЕРАТУРА**

1. Уроки программирования на языке С++ [Электронный ресурс]. URL: <https://ravesli.com/uroki-cpp> (дата обращения 24.09.2021)