#include <iostream>

#include <vector>

#include <string>

#include <cmath>

class Point{

public:

Point(){

x = 0.0;

y = 0.0;

}

Point(double x1, double y1) : x(x1), y(y1) {}

double length(Point &p){

return sqrt(pow(p.x - x,2) + pow(p.y - y,2));

}

friend std::istream &operator>>(std::istream &in, Point &p);

friend std::ostream &operator<<(std::ostream &out, Point p);

double x;

double y;

};

std::istream &operator>>(std::istream &in, Point &p){

in >> p.x;

in >> p.y;

return in;

}

std::ostream &operator<<(std::ostream &out, Point p){

out << "(" << p.x << ";" << p.y << ");";

return out;

}

class Figure{

public:

Figure(){

std::cout << "Creating figure..." << std::endl;

name = "Unknown..";

}

~Figure(){

std::cout << "Deleting figure..." << std::endl;

}

std::vector<Point> verticles;

std::string name;

Point center(){

double x\_mid = 0, y\_mid = 0;

for (Point &p : verticles){

x\_mid += p.x;

y\_mid += p.y;

}

return Point(x\_mid / verticles.size(),y\_mid / verticles.size());

}

bool check\_verticles(){

double cur\_len = verticles[0].length(verticles[verticles.size() - 1]);

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

double fig\_len = verticles[i].length(verticles[i + 1]);

if (abs(fig\_len - cur\_len) >= 2e+1){

std::cout << "Figure must have equal sides. Try again!" << std::endl;

return false;

}

if (fig\_len == 0){

std::cout << "Points should be different. Try again!" << std::endl;

return false;

}

}

return true;

}

virtual double square() = 0;

friend std::istream& operator>>(std::istream &in, Figure &figure);

friend std::ostream& operator<<(std::istream &out, Figure &figure);

};

std::istream& operator>>(std::istream &in, Figure &figure){

do{

for (auto &vertex : figure.verticles){

in >> vertex;

}

}while (!figure.check\_verticles());

return in;

}

std::ostream& operator<<(std::ostream &out, Figure &figure){

out << figure.name << "{";

for (Point &p : figure.verticles){

out << p << " ";

}

out << "}";

return out;

}

class Rectangle : public Figure{

public:

Rectangle(){

verticles.resize(4);

name = "Rectangle";

std::cout << "Rectangle is created!" << std::endl;

}

~Rectangle(){

std::cout << "Deleting Rectangle..." << std::endl << "Rectangle is successfully deleted!" << std::endl;

}

double square(){

return pow(verticles[0].length(verticles[1]),2);

}

};

class Trapezoid : public Figure{

public:

Trapezoid(){

verticles.resize(4);

name = "Trapezoid";

std::cout << "Trapezoid is created!" << std::endl;

}

~Trapezoid(){

std::cout << "Deleting Trapezoid..." << std::endl << "Trapezoid is successfully deleted!" << std::endl;

}

double square(){

double a = verticles[1].length(verticles[2]);

double b = verticles[0].length(verticles[3]);

double c = (verticles[0].length(verticles[1]));

double p = (a + b + 2 \* c) / 2;

return sqrt((p - a) \* (p - b) \* pow((p - c),2));

}

};

class Rhombus : public Figure{

public:

Rhombus(){

verticles.resize(4);

name = "Rhombus";

std::cout << "Rhombus is created!" << std::endl;

}

~Rhombus(){

std::cout << "Deleting Rhombus..." << std::endl << "Rhombus is successfully deleted!" << std::endl;

}

double square(){

return (verticles[0].length(verticles[2]) \* verticles[1].length(verticles[3])) / 2;

}

};

int main(){

unsigned int amount;

std::cout << "Enter the amount of figures you want to enter: " << std::endl;

std::cin >> amount;

std::vector<Figure \*> figures;

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

char type;

do {

std::cout << "Enter the type of figure (r - rectangle, t - trapezoid, h - rhombus)" << std::endl;

std::cin >> type;

if (type != 'r' && type != 'R' && type != 't' && type != 'T' && type != 'h' && type != 'H') {

std::cout << "Incorrect type. Try again." << std::endl;

}

} while (type != 'r' && type != 'R' && type != 't' && type != 'T' && type != 'h' && type != 'H');

if (type == 'R' || type == 'r') {

auto \*R = new Rectangle;

std::cout << "Enter vertices of this figure" << std::endl;

std::cin >> \*R;

figures.push\_back(R);

} else if (type == 'T' || type == 't') {

auto \*T = new Trapezoid;

std::cout << "Enter vertices of this figure" << std::endl;

std::cin >> \*T;

figures.push\_back(T);

} else if (type == 'H' || type == 'h') {

auto \*RH = new Rhombus;

std::cout << "Enter vertices of this figure" << std::endl;

std::cin >> \*RH;

figures.push\_back(RH);

}

}

std::cout << std::endl;

double total\_square = 0;

std::cout << "List of figures:" << std::endl;

for (auto &figure : figures) {

std::cout << \*figure << std::endl;

double cur\_square = figure->square();

total\_square += cur\_square;

std::cout << "Square: " << cur\_square << std::endl;

std::cout << "Center: " << figure->center() << std::endl;

std::cout << std::endl;

}

std::cout << "Total square of figures is " << total\_square << std::endl;

std::cout << std::endl;

unsigned int amount\_delete;

do {

std::cout << "Enter the amount of figures you want to delete: " << std::endl;

std::cin >> amount\_delete;

if (amount\_delete > figures.size()) {

std::cout << "Size of vector is less than your number. Try again" << std::endl;

}

} while (amount\_delete > figures.size());

std::cout << std::endl;

for (int i = 0; i < amount\_delete; ++i) {

int id;

do {

std::cout << "Enter the id of figure you want to delete "

"(from " << 0 << " to " << figures.size() - 1 << "): " << std::endl;

std::cin >> id;

if (id < 0 || id >= figures.size()) {

std::cout << "Wrong id. Try again" << std::endl;

}

} while (id < 0 || id >= figures.size());

delete figures[id];

figures.erase(figures.begin() + id);

}

std::cout << std::endl;

std::cout << "List of remaining figures:" << std::endl;

for (auto &figure : figures) {

std::cout << \*figure << std::endl;

double cur\_square = figure->square();

total\_square += cur\_square;

std::cout << "Square: " << cur\_square << std::endl;

std::cout << "Center: " << figure->center() << std::endl;

std::cout << std::endl;

}

std::cout << std::endl;

if (!figures.empty()) {

std::cout << "Deleting remained figures:" << std::endl;

for (auto &figure : figures) {

delete figure;

}

}

return 0;

}