C++ Programming Practice Report

|  |  |
| --- | --- |
| Student ID | 18511510019 |
| Student Name | MOHAMED NUR |
| Practice No. |  |
| Practice Title |  |
| Date |  |
| Place |  |
| Mark |  |
| Checked by | Wingo WU |

# Design a Circle class

## Objects

1. to learn well the overloaded operator
2. to get a deep understanding of OOP concepts
3. to learn well the overloaded functions

## Problems

### Design a circle class for 2D circle

Circle has attributes: center position(x,y), radius.

Circle has behavoirs: Get area, which can zoom, move, and two circles can be compared by its radius. greater than, equal, less than. You are required to overload comparison opeators(< > == ) for circle class based on its radius.

Test your Circle class with following program.

int test1()

{

double x1,y1,r1;

cin>>x1>>y1>>r1;

Circle c1(x1,y1,r1); // 3 and 5 is center , 2.5 is radius;

cin>>x1>>y1>>r1;

Circle c2(x1,y1,r1); // 3 and 5 is center , 2.5 is radius;

c1.Show();

c2.Show();

if( c1> c2)

cout<<" c1 is bigger than c2"<<endl;

else if(c1==c2)

cout<<" c1 is equal to c2"<<endl;

else

cout<<" c1 is smaller than c2"<<endl;

r1=rand() %50;

r2=rand()%50;

c1.SetRadius(r1);

c2.SetRadius(r2);

c1.Show();

c2.Show();

if( c1> c2)

cout<<" c1 is bigger than c2"<<endl;

else if(c1==c2)

cout<<" c1 is equal to c2"<<endl;

else

cout<<" c1 is smaller than c2"<<endl;

}

### sorting on circle class

Review Practice P3, write overloaded functions(input, sort, output) on Circle.

void Test()

{

Circle c[10];

Input(c,10);

sort(c,10);

output(c,10);

}

### Problem 3: use function template

copy the function template in Practice 2 to here, dont need to do any change on it but you maybe have to do some change on circle class. Run the test function in Problem 2.

void Test()

{

Circle c[10];

Input(c,10);

sort(c,10);

output(c,10);

}

## Results

### 

[Program in text format]

// 61.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

using namespace std;

class Circle {

double x, y, ridus;

const double PI = 3.14;

public:

Circle() : x(0.0), y(0.0), ridus(0.0) { cout << "Circle constructer called" << endl; };

Circle(double newX, double newY, double newRidus) : x(newX), y(newY), ridus(newRidus) { };

double GetArea();

void Zoom();

void Move();

double GetX() { return x; };

double GetY() { return y; };

double GetRidus() { return ridus; };

void SetX(double newX) { x = newX; };

void SetY(double newY) { y = newY; };

void SetRidus(double newRidus) { ridus = newRidus; };

bool operator<(Circle &c);

bool operator>(Circle &c);

bool operator==(Circle &c);

void Show();

};

double Circle::GetArea() {

return PI \* (ridus \* ridus);

}

bool Circle :: operator<(Circle &c) {

return this->GetRidus() < c.GetRidus();

}

bool Circle :: operator>(Circle &c) {

return this->GetRidus() > c.GetRidus();

}

bool Circle :: operator==(Circle &c) {

return this->GetRidus() == c.GetRidus();

}void Circle::Show() {

cout <<"x = " << this -> GetX() << " y = " << this->GetY() << " ridus = " << this->GetRidus() << endl;

}

void test1()

{

double x1, y1, r1, r2;

cin >> x1 >> y1 >> r1;

Circle c1(x1, y1, r1); // 3 and 5 is center , 2.5 is radius;

cin >> x1 >> y1 >> r1;

Circle c2(x1, y1, r1); // 3 and 5 is center , 2.5 is radius;

c1.Show();

c2.Show();

if (c1> c2)

cout << " c1 is bigger than c2" << endl;

else if (c1 == c2)

cout << " c1 is equal to c2" << endl;

else

cout << " c1 is smaller than c2" << endl;

r1 = rand() % 50;

r2 = rand() % 50;

c1.SetRidus(r1);

c2.SetRidus(r2);

c1.Show();

c2.Show();

if (c1> c2)

cout << " c1 is bigger than c2" << endl;

else if (c1 == c2)

cout << " c1 is equal to c2" << endl;

else

cout << " c1 is smaller than c2" << endl;

}

int main()

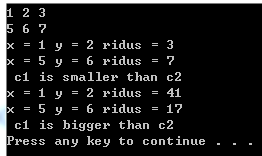
{

test1();

return 0;

}

[Screen shot]



### 

[Program in text format]

// 61.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

using namespace std;

class Circle {

double x, y, ridus;

const double PI = 3.14;

public:

Circle() : x(0.0), y(0.0), ridus(0.0) { cout << "Circle constructer called" << endl; };

Circle(double newX, double newY, double newRidus) : x(newX), y(newY), ridus(newRidus) { };

double GetArea();

void Zoom();

void Move();

double GetX() { return x; };

double GetY() { return y; };

double GetRidus() { return ridus; };

void SetX(const double newX) { x = newX; };

void SetY(const double newY) { y = newY; };

void SetRidus(const double newRidus) { ridus = newRidus; };

bool operator<(Circle &c);

bool operator>(Circle &c);

bool operator==(Circle &c);

void operator=(Circle &c);

void Show();

friend ostream& operator<<(ostream & out, const Circle &c);

friend istream& operator>>(istream & out, Circle &c);

};

ostream& operator<<(ostream & out, const Circle &c) {

out << "x = " << c.x << " y = " << c.y << " ridus = " << c.ridus << endl;

return out;

}

istream& operator>>(istream & out, Circle &c) {

out >> c.x >> c.y >> c.ridus;

return out;

}

double Circle::GetArea() {

return PI \* (ridus \* ridus);

}

bool Circle :: operator<(Circle &c) {

return this->GetRidus() < c.GetRidus();

}

bool Circle :: operator>(Circle &c) {

return this->GetRidus() > c.GetRidus();

}

void Circle :: operator=(Circle &c) {

this->SetRidus(c.GetRidus());

}

bool Circle :: operator==(Circle &c) {

return this->GetRidus() == c.GetRidus();

}void Circle::Show() {

cout <<"x = " << this -> GetX() << " y = " << this->GetY() << " ridus = " << this->GetRidus() << endl;

}

void input(Circle a[], int size) {

for (int i = 0; i < size; i++)

cin >> a[i];

}

void sort(Circle a[], int size) {

int i, j;

for (i = 1; i < size; i++) {

for (j = 0; j < size - 1; j++) {

if (a[j] > a[j + 1]) {

swap(a[j], a[j + 1]);

}

}

}

}

void output(Circle a[], int size) {

for (int i = 0; i < size; i++)

cout << a[i] << " ";

cout << endl;

}

void test2()

{

Circle c[3];

input(c, 3);

sort(c, 3);

output(c, 3);

}

int main()

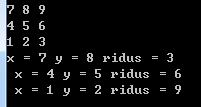
{

test2();

return 0;

}

[Screen shot]



### 

[Program in text format]

Everything the same as the last one except the following code;

template<typename T>

void input(T a[], int size) {

for (int i = 0; i < size; i++)

cin >> a[i];

}

template<typename T>

void sort(T a[], int size) {

int i, j;

for (i = 1; i < size; i++) {

for (j = 0; j < size - 1; j++) {

if (a[j] > a[j + 1]) {

swap(a[j], a[j + 1]);

}

}

}

}

template<typename T>

void output(T a[], int size) {

for (int i = 0; i < size; i++)

cout << a[i] << " ";

cout << endl;

}

[Screen shot]

