

深圳大学实验报告

课程名称： 面向对象程序设计

实验项目名称： 实验六 类的自动化

学院： 医学院

专业： 生物医学工程

指导教师： 李乔亮、邓云

报告人： 陈焕鑫 学号： 2016222042 班级： 生工 2 班

实验时间： 2018.11.21

实验报告提交时间： 2018.11.26

教务部制

实验目的:

熟练掌握 C++ 中类与对象的使用

实验内容:

1. 对圆进行抽象，完成类的定义，实现人机接口。

要求:

- 1) 要求提供修改圆属性、圆平移、缩放、求面积，求与另外一圆的重叠面积、打印等接口。
 - 2) 圆具有特定的名字，且长度不固定。
 - 3) 至少提供 3 种构造方式，包括 1 个无参构造、1 个有参构造，和拷贝构造。
 - 4) 在 **Main** 中采用多种方式构造 4 个圆，通过打印接口输出这些圆的基本信息（基本属性，面积等），平均面积和每两个圆间的重叠面积，对这些圆进行平移，缩放等操作后，再次打印输出，并与数学计算的结果对比，确保正确性。
2. 在题 1 的工程中依次求取 4 个圆的面积，通过追踪 **this** 指针的变化，解释该求面积函数分别求得 4 个圆面积的过程（请调试 **this** 指针与圆对象之间的动态关系）。

实验环境与程序代码:

实验环境：win10 系统下的 Visual Studio 2017

程序代码如下所示:

```
//circle.h

#ifndef _CIRCLE_H_
#define _CIRCLE_H_

#define PI 3.1415926 //定义圆周率的大小

class CCircle
{
private:
    float radius;          //半径
    float point_x, point_y; //圆心
    float area;            //面积
    char *name;            //名字
    static int num;        //圆的个数
    static float allArea;  //圆的总面积
public:
    CCircle();              //无参构造函数
    CCircle(float i_radius, float i_px, float i_py, const char *i_name); //有参构造函数
    CCircle(const CCircle &i_circle); //拷贝构造函数
    ~CCircle();            //析构函数
    void printCircle(void); //打印圆的信息
    float getArea(void);    //获取圆的面积
    void move(float offset_x, float offset_y); //平移
    bool zoom(float ratio); //缩放
    float getOverlapArea(const CCircle &i_circle); //计算两个圆的重叠面积
    char* getName();        //获取圆的名字
    static float getAverageArea(void); //获取平均面积
};

#endif

//circle.cpp

#include <iostream>
#include <math.h>
#include "circle.h"
using namespace std;

#define max(a,b) ((a) > (b)) ? (a) : (b) //求最大值函数
```

```

int CCircle::num = 0;          //初始化静态变量 num
float CCircle::allArea = 0;    //初始化静态变量 allArea

CCircle::CCircle()
{
    radius = 1;                //默认半径为 1
    point_x = 0;               //默认圆心在 origin
    point_y = 0;
    area = radius * radius * PI; //求得面积
    name = new char[strlen("default") + 1]; //申请内存空间
    strcpy(name, "default");      //复制字符串
    num++;                       //圆的数目加 1
    allArea += area;             //更新总面积
}

CCircle::CCircle(float i_radius, float i_px, float i_py, const char *i_name)
{
    radius = i_radius;          //半径
    point_x = i_px;             //圆心
    point_y = i_py;
    name = new char [strlen(i_name) + 1]; //申请内存空间
    strcpy(name, i_name);       //复制字符串
    area = radius * radius * PI; //求得面积
    num++;                     //圆的数目加 1
    allArea += area;           //更新总面积
}

CCircle::CCircle(const CCircle &i_circle)
{
    char cpyName[20] = "COPY_"; //初始化字符串前缀

    radius = i_circle.radius;    //半径
    point_x = i_circle.point_x;  //圆心
    point_y = i_circle.point_y;
    strcat(cpyName, i_circle.name); //字符串拼接
    name = new char [strlen(cpyName) + 1]; //申请内存空间
    strcpy(name, cpyName);        //复制字符串到 name
    area = radius * radius * PI;  //求得面积
    num++;                       //圆的数目加 1
    allArea += area;             //更新总面积
}

CCircle::~~CCircle()

```

```

{
    num--;          //圆的数目减1
    allArea -= area; //更新总面积
    delete[]name;   //释放指针
}

void CCircle::printCircle()
{
    cout << "-----" << endl;
    cout << "名字: " << name << endl;
    cout << "半径: " << radius << endl;
    cout << "圆心: [" << point_x << ", " << point_y << "]" << endl;
    cout << "面积: " << area << endl;
    cout << "-----" << endl;
}

float CCircle::getArea(void)
{
    return area;    //返回面积的值
}

void CCircle::move(float offet_x, float offset_y)
{
    point_x += offet_x; //x 加上偏移量
    point_y += offset_y; //y 加上偏移量
}

bool CCircle::zoom(float ratio)
{
    if(ratio <= 0)      //当比例小于0时
    {
        cout << "缩放比例小于0, 错误!" << endl; //打印错误
        return false;    //返回错误
    }

    allArea -= area;      //先减去原来的面积
    radius = radius * ratio; //对半径进行缩放
    area = radius * radius * PI; //求得面积
    allArea += area;      //加上新的面积

    return true;          //返回正确
}

float CCircle::getOverlapArea(const CCircle &i_circle)

```

```

{
    float S_overlap = 0;
    float centerDistance;

    //求圆心距
    centerDistance = sqrt(pow(point_x - i_circle.point_x, 2) + pow(point_y -
i_circle.point_y, 2));

    if (centerDistance >= radius + i_circle.radius) //如果不相交
    {
        cout << name << " 和 " << i_circle.name << " 不相交" << endl;
        S_overlap = 0;
        return 0;
    }
    else if (centerDistance <= abs(radius - i_circle.radius)) //如果相互包含
    {
        if (radius < i_circle.radius) //半径小的被包含
        {
            cout << name << " 包含于 " << i_circle.name << endl;
            S_overlap = area;
        }
        else if (radius > i_circle.radius)
        {
            cout << i_circle.name << " 包含于 " << name << endl;
            S_overlap = i_circle.area;
        }
        else if (radius == i_circle.radius) //半径相等即重叠
        {
            cout << name << " 和 " << i_circle.name << " 刚好重合" << endl;
            S_overlap = area;
        }
    }
    else //相交
    {
        float height1 = radius * radius + centerDistance * centerDistance -
i_circle.radius*i_circle.radius;
        float height2 = i_circle.radius*i_circle.radius + centerDistance * centerDistance
- radius * radius;
        float thita1 = 2 * acos(height1 / (2 * radius*centerDistance));
        float thita2 = 2 * acos(height2 / (2 * i_circle.radius*centerDistance));

        float S_arch1 = radius * radius * (thita1 / 2 - sin(thita1) / 2);
        float S_arch2 = i_circle.radius * i_circle.radius * (thita2 / 2 - sin(thita2) / 2);
    }
}

```

```

        S_overlap = S_arch1 + S_arch2;

        cout << name << " 和 " << i_circle.name << " 相交" << endl;
    }

    return S_overlap;
}

char* CCircle::getName()
{
    return name;        //返回 name 指针
}

float CCircle::getAverageArea(void)
{
    return allArea / num;    //返回平均面积
}

//main.cpp
#include <iostream>
#include "circle.h"
using namespace std;

int main()
{
    CCircle a, b(2.5,0,0,"circle1"), c(1, 1, 0, "circle2");
    CCircle d(c);    //拷贝 c 构造 d
    a.printCircle();    //打印圆 a 的面积
    b.printCircle();    //打印圆 b 的面积
    c.printCircle();    //打印圆 c 的面积
    d.printCircle();    //打印圆 d 的面积

    cout << endl << "平均面积: " << CCircle::getAverageArea() << endl << endl;    //打印
    平均面积

    cout << a.getName() << "和" << b.getName() << " 的重叠面积:" << a.getOverlapArea(b) <<
    endl << endl;

    cout << a.getName() << "和" << c.getName() << " 的重叠面积:" << a.getOverlapArea(c) <<
    endl << endl;

    cout << a.getName() << "和" << d.getName() << " 的重叠面积:" << a.getOverlapArea(d) <<
    endl << endl;

    cout << b.getName() << "和" << c.getName() << " 的重叠面积:" << b.getOverlapArea(c) <<
    endl << endl;

    cout << b.getName() << "和" << d.getName() << " 的重叠面积:" << b.getOverlapArea(d) <<
    endl << endl;

    cout << c.getName() << "和" << d.getName() << " 的重叠面积:" << c.getOverlapArea(d) <<

```

```
endl << endl;

cout << "-----" << endl;

a.move(1, 1);    //平移圆 a
b.zoom(2);      //缩放圆 b
c.move(-1, -3); //平移圆 c
d.zoom(0.5);    //缩放圆 d

a.printCircle(); //打印圆 a 的面积
b.printCircle(); //打印圆 b 的面积
c.printCircle(); //打印圆 c 的面积
d.printCircle(); //打印圆 d 的面积

cout << endl << "平均面积: " << CCircle::getAverageArea() << endl << endl; //打印平均面积

cout << a.getName() << "和" << b.getName() << " 的重叠面积:" << a.getOverlapArea(b) << endl << endl;

cout << a.getName() << "和" << c.getName() << " 的重叠面积:" << a.getOverlapArea(c) << endl << endl;

cout << a.getName() << "和" << d.getName() << " 的重叠面积:" << a.getOverlapArea(d) << endl << endl;

cout << b.getName() << "和" << c.getName() << " 的重叠面积:" << b.getOverlapArea(c) << endl << endl;

cout << b.getName() << "和" << d.getName() << " 的重叠面积:" << b.getOverlapArea(d) << endl << endl;

cout << c.getName() << "和" << d.getName() << " 的重叠面积:" << c.getOverlapArea(d) << endl << endl;

return 0;
}
```


实验结果与分析:

1、

在 circle.h 代码中，首先定义了圆周率 PI 的大小为 3.1415926。在类 CCircle 中定义了圆的各种属性为私有变量，包括半径 radius、圆心的 point_x、point_y 两个坐标点，圆的面积，圆的名字 name。还包括类的静态变量：圆的个数 num 和圆的总面积 allArea。其中，圆的名字使用的类型是字符串指针，一开始并未给指针分配内存空间，而是在程序运行的过程中使用 new 关键字动态地为它分配内存空间。类 CCircle 提供的接口有打印圆的信息 printCircle 函数、获取圆的面积 getArea 函数、平移 move 函数、缩放 zoom 函数、获取圆的名字 getName 函数、获取平均面积 getAverageArea 函数还有计算两个圆重叠面积 getOverlapArea 函数。其中，获取平均面积 getAverageArea 函数设置为静态函数，因为它是属于整个类的属性，而不是特指某一个对象，所以设置为内部静态函数。圆的构造函数有三个，分别是一个无参构造函数、一个有参构造函数（形参列表为半径、圆心 x 坐标，圆心 y 坐标和圆的名字）、一个析构函数。在无参或者有参构造函数中为 name 指针分配了内存空间，所以在析构函数中使用 delete 关键字释放内存空间。每调用一次无参或者有参构造函数生成一个对象，num 的值就增加 1，同时 allArea 的值也加上新增加的圆的面积。每释放一个对象，就对 num 的值减去 1，同时减去释放的圆的面积。最后，如果我们要获取圆的平均面积，只需要调用函数，在函数中使用 allArea 除去 num 的值即可。

在这些接口中，获取两个圆的重叠面积这个函数比较复杂，涉及到一些几何的知识。两个圆之间的位置关系具体有以下三种情况，从左到右从上到下分别是相交、相离和包含。当两个圆相离时，两个圆的圆心距大于两个圆的半径之和，此时重叠面积为 0。当一个圆包含另一个圆时，被包含的圆半径较小，圆心距小于两圆半径之差，重叠面积为较小的那个圆的面积。剩下的情况就是两个圆相交，相交面积为两个弧形的面积和。

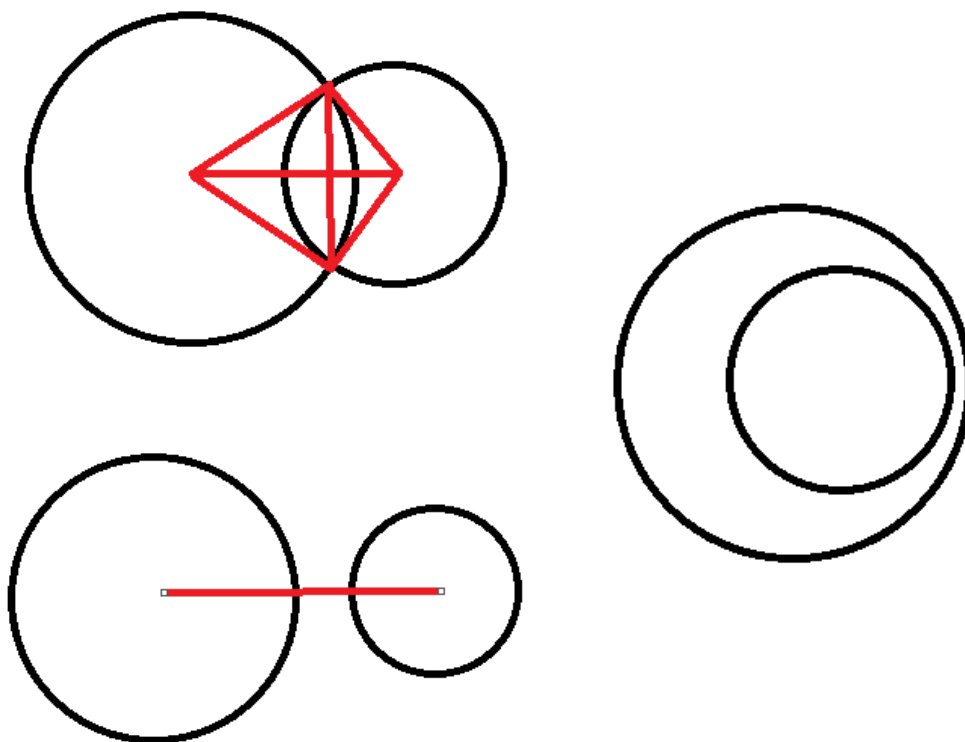


图 0 – 两个圆的位置关系

在 main.cpp 中，首先声明了四个圆对象，之后对四个圆进行相应的操作。

如图 1 为对四个圆进行变换之前的信息情况，圆 1 的名字为 default，半径为 1，圆心为[0, 0]，面积为 3.14159。圆二的名字为 circle1，半径为 2.5，圆心为[0, 0]，面积为 19.635 等等。

```
-----
名字: default
半径: 1
圆心: [0, 0]
面积: 3.14159
-----

名字: circle1
半径: 2.5
圆心: [0, 0]
面积: 19.635
-----

名字: circle2
半径: 1
圆心: [1, 0]
面积: 3.14159
-----

名字: COPY_circle2
半径: 1
圆心: [1, 0]
面积: 3.14159
-----

平均面积: 7.26493

default 包含于 circle1
default和circle1 的重叠面积:3.14159

default 和 circle2 相交
default和circle2 的重叠面积:1.22837

default 和 COPY_circle2 相交
default和COPY_circle2 的重叠面积:1.22837

circle2 包含于 circle1
circle1和circle2 的重叠面积:3.14159

COPY_circle2 包含于 circle1
circle1和COPY_circle2 的重叠面积:3.14159

circle2 和 COPY_circle2 刚好重合
circle2和COPY_circle2 的重叠面积:3.14159
```

图 1 – 对各个圆作变换前的情况

对圆进行变换之后得到的圆的信息如图 2 所示。

```
-----
名字: default
半径: 1
圆心: [1, 1]
面积: 3.14159
-----

名字: circle1
半径: 5
圆心: [0, 0]
面积: 78.5398
-----

名字: circle2
半径: 1
圆心: [0, -3]
面积: 3.14159
-----

名字: COPY_circle2
半径: 0.5
圆心: [1, 0]
面积: 0.785398
-----

平均面积: 21.4021

default 包含于 circle1
default和circle1 的重叠面积:3.14159

default 和 circle2 不相交
default和circle2 的重叠面积:0

default 和 COPY_circle2 相交
default和COPY_circle2 的重叠面积:0.350767

circle2 包含于 circle1
circle1和circle2 的重叠面积:3.14159

COPY_circle2 包含于 circle1
circle1和COPY_circle2 的重叠面积:0.785398

circle2 和 COPY_circle2 不相交
circle2和COPY_circle2 的重叠面积:0
```

图 2- 对各个圆作变换后的情况

2、

为了追踪 this 指针的变换，设计了如图 3 所示的代码。将断点设置在 a.getArea 处，进行调试。调试的过程中，对变量 this，&a，&b，&c，&d 进行监视。

调试开始时，如图 4，this 显示 “this 只能用于非静态成员变量内部”，说明了在对象

的外部使用 this 指针是非法的。而 a, b, c, d 的地址分别为 0x97fedc, 0x97fec0, 0x97fea4, 0x97fe88。当程序执行到进入 a 对象函数 getArea 时，如图 5 观察监视窗口可以看到 this 指针指向的地址是 0x97fedc，正好是对象 a 的地址，这就说明了在调用对象的函数时，程序会自动地将 this 指针指向相应的对象的首地址。图 6，图 7，图 8 也是同样的情况。

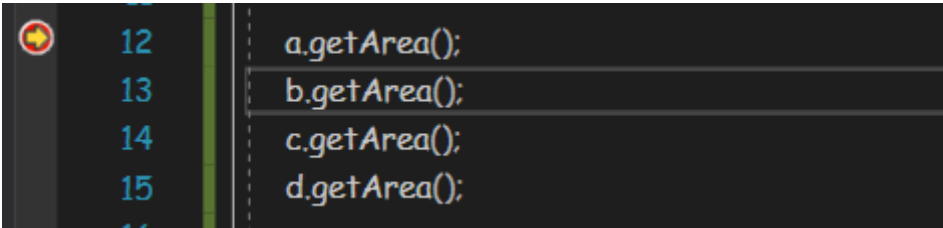


图 3 – 调试代码片段







 this	"this"只能用于非静态成员函数内部	
▶  &a	0x0097fedc {radius=1.00000000 point_x=0.000000000 poin...	CCircle *
▶  &b	0x0097fec0 {radius=2.50000000 point_x=0.000000000 poin...	CCircle *
▶  &c	0x0097fea4 {radius=1.00000000 point_x=1.00000000 point...	CCircle *
▶  &d	0x0097fe88 {radius=1.00000000 point_x=1.00000000 point...	CCircle *

图 4 – 初始情况下各个指针的情况







名称	值	类型
▲  this	0x0097fedc {radius=1.00000000 point_x=0.000000000 poin...	CCircle *
 radius	1.00000000	float
 point_x	0.000000000	float
 point_y	0.000000000	float
 area	3.14159250	float
▶  name	0x02f6d308 "default"	char *

图 5 – 求 a 面积时 this 指针的情况







▲  this	0x0097fec0 {radius=2.50000000 point_x=0.000000000 poin...	CCircle *
 radius	2.50000000	float
 point_x	0.000000000	float
 point_y	0.000000000	float
 area	19.6349545	float
▶  name	0x02f6d570 "circle1"	char *

图 6 – 求 b 面积时 this 指针的情况







▲  this	0x0097fea4 {radius=1.00000000 point_x=1.00000000 point...	CCircle *
 radius	1.00000000	float
 point_x	1.00000000	float
 point_y	0.000000000	float
 area	3.14159250	float
▶  name	0x02f6d490 "circle2"	char *

图 7 – 求 c 面积时 this 指针的情况







▲  this	0x0097fe88 {radius=1.00000000 point_x=1.00000000 point...	CCircle *
 radius	1.00000000	float
 point_x	1.00000000	float
 point_y	0.000000000	float
 area	3.14159250	float
▶  name	0x02f60568 "COPY_circle2"	char *

图 8 – 求 d 面积时 this 指针的情况

指导教师批阅意见：

成绩评定：

指导教师签字：

年 月 日

备注：

- 注： 1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。