实验八上机指导（1）

游泳池类：我们将游泳池的复制构造函数和赋值运算符声明为私有成员，从而防止对象调用它们。如果不这样做，则需要为其设计复制构造函数和赋值运算符，这就需要识别其封装的指针g指向的对象到底是什么对象（是CCircle、CRectangle、CTriangle？还是其他的从CGraph派生出来的类型？）。

本实验重点理解多态的使用：从下面的程序中可以看出，当使用了多态之后，无论游泳池的形状是什么样的，游泳池类的程序都无需改动，这就使程序具有很好的可扩展性，程序也有很好的重用性。

另外，在下面的程序中已经实现了文本模式读写文件的功能。请大家练习二进制文件的读写功能。

```
//file: graph.h ------------------------------------------------------
#ifndef __GRAPH_H__
#define __GRAPH_H__

#include<iostream>
#include<math.h>
#include<fstream>
#include"MyString.h"

using namespace std;

#define PI 3.14159


class CGraph
{
public:
     CGraph() : name("graph") { }              //提供默认构造函数
     CGraph(const char * p) : name(p) { }
     //即使没有使用堆上的内存也应该为基类提供虚析构函数
     virtual ~CGraph() { cout << "~CGraph" << endl; }
     virtual double area() = 0;
     virtual double perimeter() = 0;

     //以文本模式保存和读取文件（二进制模式的请大家自行实现）
     virtual ostream & write_text(ostream & o)
     {
          name.write_text(o);
          return o;
     }

     virtual ifstream & read_text(ifstream & in)
     {
          name.read_text(in);
          return in;
     }
private:
     MyString name;
};

class CRectangle : public CGraph
```

```
{
public:
    CRectangle() : CGraph("rectangle"), height(0), width(0) { }
    CRectangle(const char * p, double h,double w);
    CRectangle(const CRectangle & r);
    //由于基类的析构函数是虚的，所以即使这里不声明为虚的，它也是虚的
    ~CRectangle() { cout << "~CRectangle" << endl;  }
    double area();
    double perimeter();

    //文本模式的文件读写
    virtual ostream & write_text(ostream & o)
    {
        CGraph::write_text(o);
        o << " " << height << " " << width << " " << endl;
        return o;
    }

    virtual ifstream & read_text(ifstream & in)
    {
        CGraph::read_text(in);
        in.get();
        in >> height >> width;
        in.get();
        return in;
    }
private:
    double height;
    double width;
};

class CCircle : public CGraph
{
public:
    CCircle() : CGraph("circle"), radius(0) { }
    CCircle(const char * p, double r);
    CCircle(const CCircle & c);
    double area();
    double perimeter();

    //文本模式的文件读写
    virtual ostream & write_text(ostream & o)
    {
        CGraph::write_text(o);
        o << " " << radius << " " << endl;
        return o;
    }

    virtual ifstream & read_text(ifstream & in)
    {
        CGraph::read_text(in);
        in.get();
        in >> radius;
        in.get();
        return in;
```

```
    }
private:
    double radius;
};

class CTriangle : public CGraph
{
public:
    CTriangle() : CGraph("triangle"), s1(0), s2(0), s3(0) { }
    CTriangle(const char * p, double a, double b, double c);
    CTriangle(const CTriangle & t);
    double area();
    double perimeter();

    //文本模式的文件读写
    virtual ostream & write_text(ostream & o)
    {
        CGraph::write_text(o);
        o << " " << s1 << " " << s2 << " " << s3 << " " << endl;
        return o;
    }

    virtual ifstream & read_text(ifstream & in)
    {
        CGraph::read_text(in);
        in.get();
        in >> s1 >> s2 >> s3;
        in.get();
        return in;
    }
private:
     double s1, s2, s3;
};

#endif


//file: graph.cpp
#include "graph.h"

CRectangle::CRectangle(const char * p, double h, double w) : CGraph(p), height(h), width(w)
{
}

CRectangle::CRectangle(const CRectangle & r) : CGraph(r), height(r.height), width(r.width)
{
}

double CRectangle::area()
{
    return height * width;
}

double CRectangle::perimeter()
{
```

```
    return 2 * (height + width);
}

CCircle::CCircle(const char * p, double r) : CGraph(p), radius(r)
{
}

CCircle::CCircle(const CCircle & c) : CGraph(c), radius(c.radius)
{
}

double CCircle::area()
{
    return PI * radius * radius;
}

double CCircle::perimeter()
{
    return 2 * PI * radius;
}

CTriangle::CTriangle(const char * p, double a, double b, double c) : CGraph(p), s1(a), s2(b), s3(c)
{
}

CTriangle::CTriangle(const CTriangle & t) : CGraph(t), s1(t.s1), s2(t.s2), s3(t.s3)
{
}

double CTriangle::area()
{
    double p = (s1 + s2 + s3) / 2;
    return pow(p * (p - s1) * (p - s2) * (p - s3), 0.5);
}

double CTriangle::perimeter()
{
    return s1 + s2 + s3;
}


//file: swimpool.h
#pragma once
#include "graph.h"

// 简化起见，我们只设计一个游泳池类，并设计计算其面积和周长的函数
class CSwimPool
{
public:
    // 将形参p指向的空间（必须在堆中分配）交给CSwimPool类管理，此时该类的g指针指向上述内存
    CSwimPool(CGraph * p) { g = p; }
    /* 类似于这个的写法是错误的，原因是CGraph应是抽象类，不能定义其对象；memcpy函数也只能实现浅复制，且只能复制CGraph类中的数据
    CSwimPool::CSwimPool(const CGraph * p)
    {
```

```
    g = new CGraph;
    memcpy(g, p, sizeof(CGraph));
    }
    */
    virtual ~CSwimPool() { if (g != NULL) delete g; }
    double area() { return g != NULL ? g->area() : -1; }
    double perimeter() { return g != NULL ? g->perimeter() : -1; }

    //对于文件读写，由于无法预测其图形类到底是什么，所以只能通过调用对应的虚函数完成读写
    //对于指示具体图形类型的标记，应该在类的外部确定
    ostream & write_text(ostream & o)
    {
        g->write_text(o);
        return o;
    }

    ifstream & read_text(ifstream & in)
    {
        g->read_text(in);
        return in;
    }
private:
    CSwimPool(const CSwimPool & pool);              //声明为私有函数，防止对象调用该函数
    CSwimPool & operator=(const CSwimPool & pool); //声明为私有函数，防止对象调用该函数
    CGraph * g;
};


//file: main.cpp
#include "swimpool.h"

int main()
{
    CGraph * pGraph = new CRectangle;
    cout << pGraph->area() << endl;
    delete pGraph;     //演示先调用派生类的析构函数再调用基类的析构函数

    CCircle * p_c = new CCircle("circle", 3);
    CRectangle * p_r = new CRectangle("rect", 4, 7.5);
    CTriangle * p_t = new CTriangle("triangle", 3, 4, 5);

    CSwimPool pool(p_c);
    cout << "面积是：" << pool.area() << "\t周长是：" << pool.perimeter() << endl;

    CSwimPool pool2(p_r);
    cout << "面积是：" << pool2.area() << "\t周长是：" << pool2.perimeter() << endl;

    CSwimPool pool3(p_t);
    cout << "面积是：" << pool3.area() << "\t周长是：" << pool3.perimeter() << endl;

    ofstream out("data.txt");
    //将pool、pool2、pool3写入到文件data.txt中，在写入每个对象之前需要写入一个标记以指示图形的类
    //型。这里，我们用1表示圆形，用2表示矩形，用3表示三角形（为方便读取，我们首先写入游泳池数量）
    out << 3 << " ";   //写入游泳池数量
    out << 1 << " ";
```

```
    pool.write_text(out);
    out << 2 << " ";
    pool2.write_text(out);
    out << 3 << " ";
    pool3.write_text(out);
    out.close();

    //下面从上面的文件中读取数据，恢复游泳池数据并计算其面积和周长
    ifstream in("data.txt");
    int num;
    in >> num;          //读取游泳池数量
    //由于游泳池类没有默认构造函数，我们只能先准备指向游泳池对象的指针
    CSwimPool ** pPool = new CSwimPool*[num];
    int type;
    for (int i = 0; i < num; i++)
    {
        in >> type;
        in.get();
        switch (type)
        {
        case 1:
            pGraph = new CCircle;
            break;
        case 2:
            pGraph = new CRectangle;
            break;
        case 3:
            pGraph = new CTriangle;
            break;
        }
        pPool[i] = new CSwimPool(pGraph);
        pPool[i]->read_text(in);
    }
    for (int i = 0; i < num; i++)
        cout << "面积是：" << pPool[i]->area() << "\t周长是：" << pPool[i]->perimeter() << endl;

    for (int i = 0; i < num; i++)
        delete pPool[i];
    delete[] pPool;

    return 0;
}
```