沈阳航空航天大学

高级程序设计实验报告

实验名称：**面向对象程序设计综合应用**

实验题目：**画图软件的设计与实现**

学　　院：　计算机学院

专　　业：　软件工程

班　　级：　软件1802

学　　号：　183401050225

姓　　名：　黄远鹏

成　　绩：

指导教师：　许　莉

完成时间：　2019　年　7　月　7　日

目　录

1 实验目的

本次《面向对象程序设计综合应用》实验，是基于“高级程序设计”课程学习内容的重要实践环节，实验目的是通过综合性实验，培养和提高学生的独立分析问题、解决实际问题的能力和计算机语言编程能力。

2 实验内容

设计一款模拟画图软件，能够模拟 Windows 画图板上各种图元的绘制功能（可以使用文字显示代替图形绘制），能够保存、读取用户的绘图数据。

3 实验要求

3.1 实验总体要求

（1）阅读设计题目、任务内容，规划设计进度，并进行软件各相关功能模块的设计。

（2）在编译环境下，用面向对象的程序设计思想进行软件设计、开发及调试。

（3）进行实验报告编写与整理。

（4）实验结束时，要求进行成果演示（由老师验收相关程序运行成果并打分）；每人须上交实验报告（纸质、电子）。

3.2 实验功能要求

能够使用封装、继承、多态和抽象机制构造灵活、具有一定弹性和扩展型的面向对象程序。

通过程序调试加深对抽象、封装、继承、多态等面向对象概念的理解，掌握方法覆盖、方法重载及类型转换等机制的实现方法。

学习并掌握使用 UML 类图、交互图描述软件设计的方法及技术。

4 系统分析及类的划分

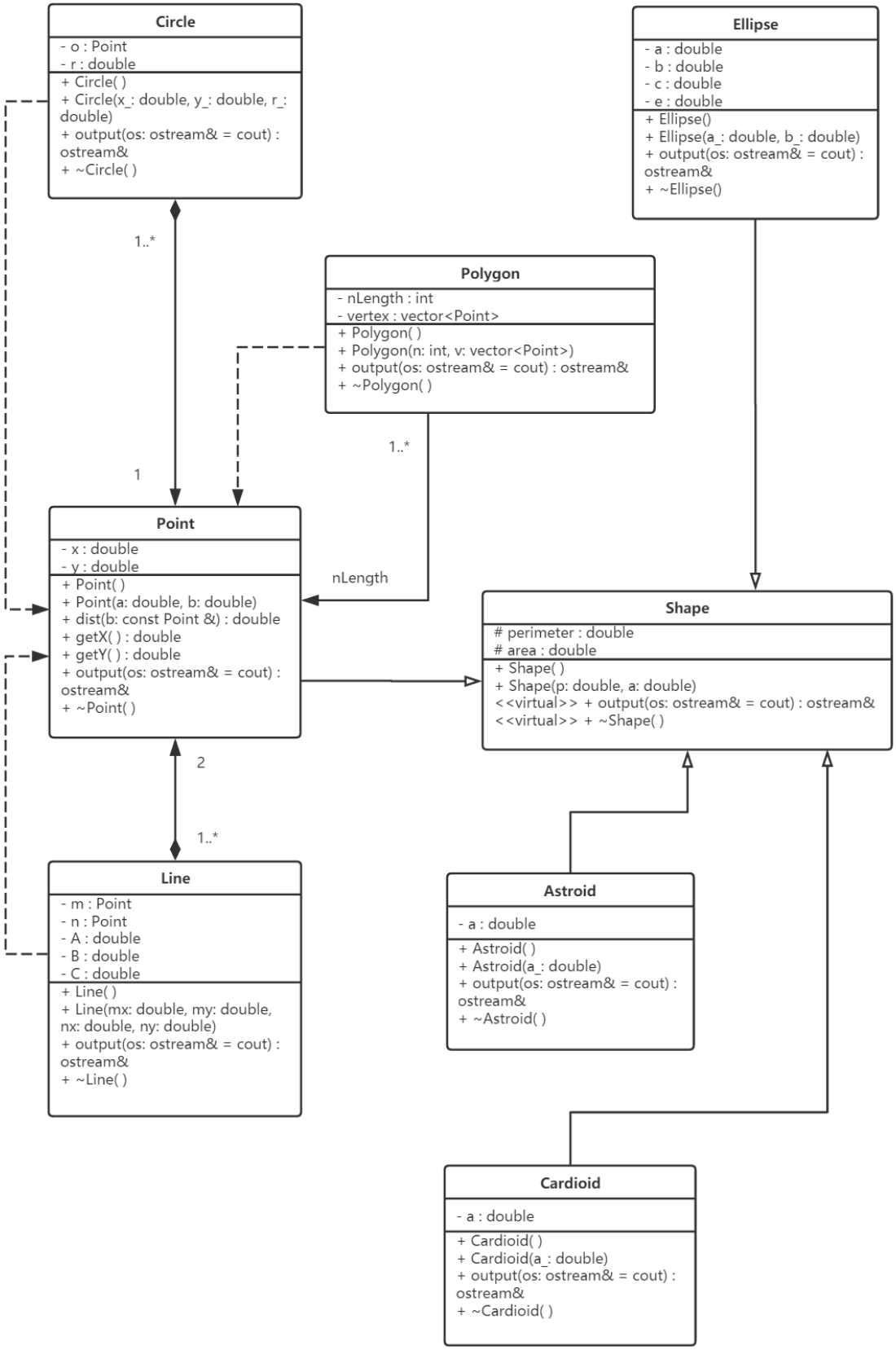

图 4.1　本程序的 UML 类图

本程序共包括 8 个类。

（1）基类：Shape 类，包含两个保护类型数据成员：

- perimeter：周长，double 型。
- area：面积，double 型。

有四个公有类型函数成员：

- Shape( )和 Shape(double p,double a)：两个重载的构造函数。
- virtual ostream &output(ostream &os = cout)：输出函数，带默认参数 std::cout，即默认向标准输出写入数据。在给定文件输出流时，也可向给定文件写入数据。基类的输出函数只输出周长和面积，各派生类在覆盖该函数时可视情况调用此函数。
- virtual ~Shape( )：虚析构函数。

（2）派生类：包括 Point（点）、Line（线段）、Circle（圆）、Ellipse（椭圆）、Polygon（多边形）、Astroid（星形线）、Cardioid（心形线）共 7 个类。派生类根据自身图形的特点，在基类的基础上添加了自己的私有数据成员。其中 Point 类被多个类所组合，添加了多个公有接口供其它类使用。

程序提供了 6 个函数用于绘制图形，各个函数大体流程相同，伪代码如下：

```
1.  function make_className( )
2.      重置窗口标题
3.      输入图形的参数
4.      while 输入非法
5.          输出提示信息
6.          重新输入参数
7.      end while
8.      申请新内存空间，初始化对象
9.      将新生成的对象地址添加到地址序列中
10.     输出“添加成功”
11.     调用 output 方法，输出图形的信息
12. end function
```

5 实验中遇到的问题及解决办法

问题及解决办法见表 5-1。

表 5-1　问题及解决办法

序号	问题描述	解决办法
1	在线段类中，最初采用斜截式（$y=kx+b$）来表示一条直线，但后来发现斜截式无法表示所有直线。	斜截式无法表示与x轴垂直的直线。换用一般式：$Ax+By+C=0$，即可表示所有直线。
2	在保存到文件的函数中，最开始使用范围**for**语句： `for(auto p:addr)` 提示“无法推导"auto"类型(需要初始值设定项)”。	**auto**让编译器通过初始值来推算变量的类型，但在定义addr时未给出初始值。换用传统的**for**语句，即可解决此问题。
3	通过**new**得到的各个派生类对象，在**delete**时要调用析构函数。各个类的析构函数要不要写成虚析构函数？	询问老师得到答案：应写成虚析构函数。
4	对于线段类的输出，如何让程序输出的方程与实际书写时的方程保持一致？	考虑到直线方程参数较多，为了让方程输出的形式与我们平时书写的格式尽量一致，需考虑一些特殊情况。比如直线方程不含某一项，则不输出该项；某一项系数为1或-1时，要考虑是否输出该项系数；通过调整系数正负号，让第一项系数始终为正数等。

6 实验结果

6.1 主菜单

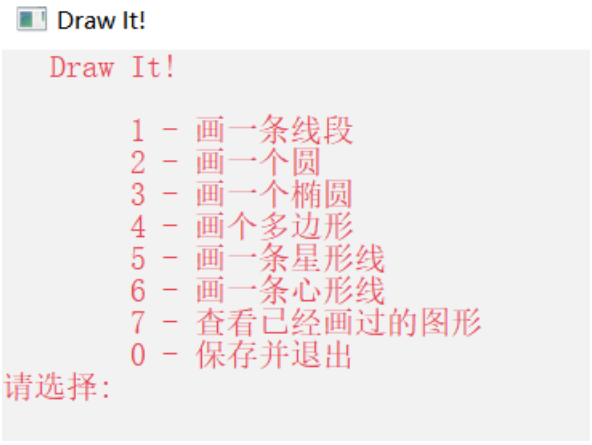

图 6.1.1 主菜单

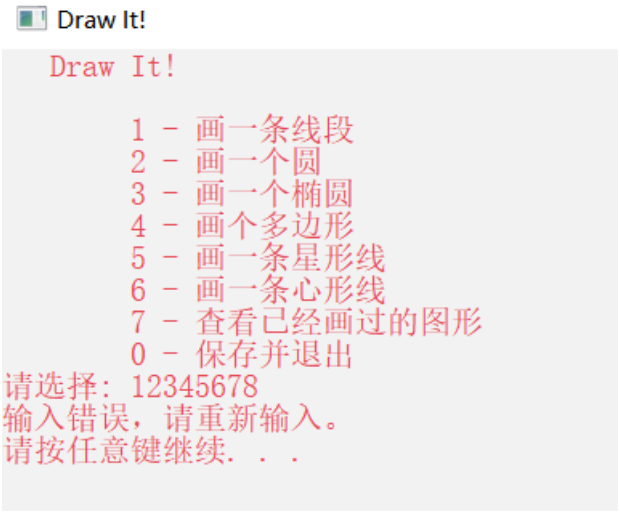

图 6.1.2 输入错误选项时，系统会输出提示信息

6.2 绘画模块

每个图形的测试见图 6.2.1 至图 6.2.7，每个测试的输入均包括无效数据和有效数据，其中椭圆类还包括一种特殊情况（图 6.2.4）。

测试用例见表 6-1。

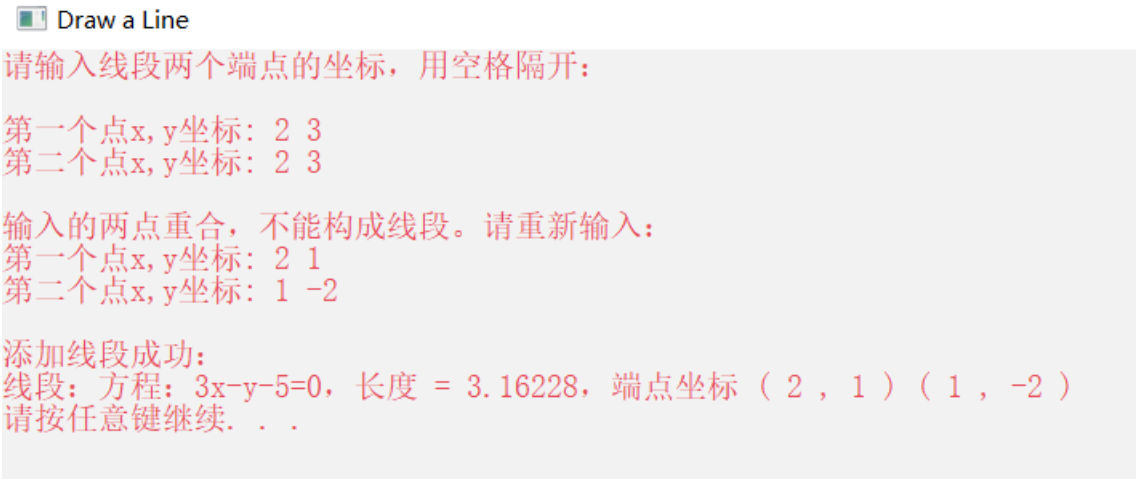

图 6.2.1　线段类测试

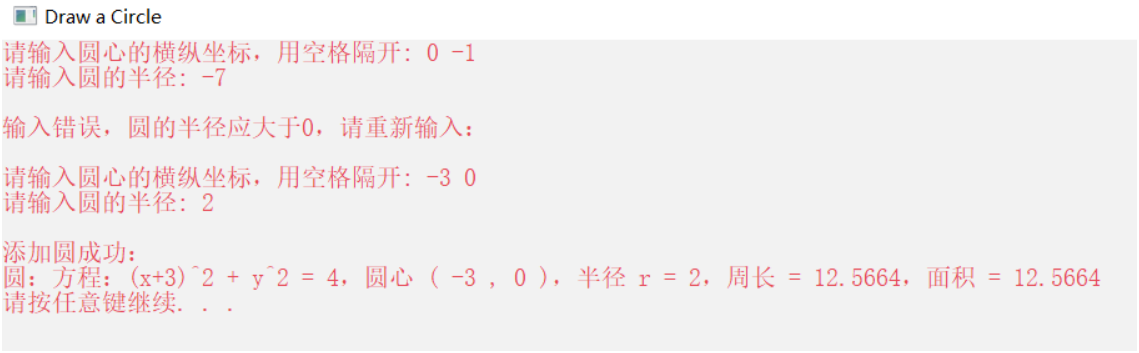

图 6.2.2　圆类测试

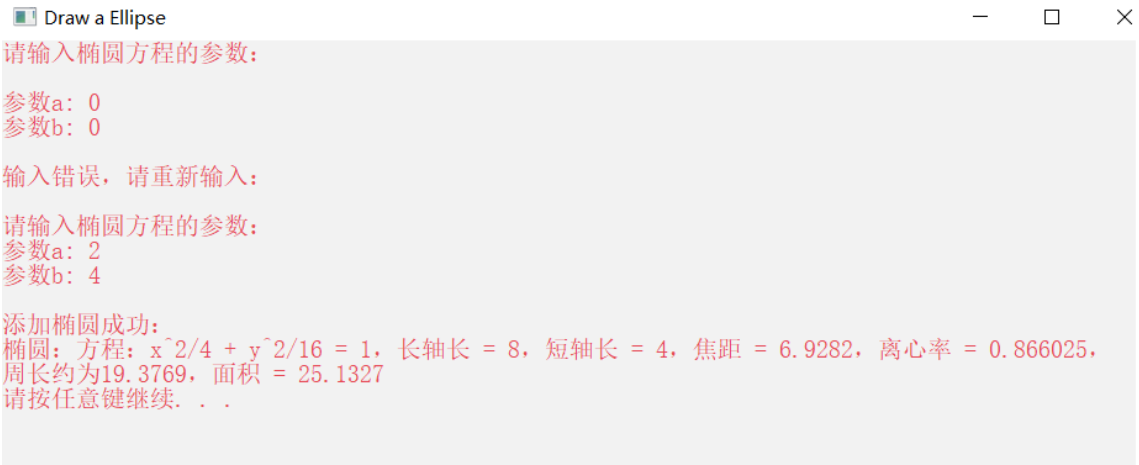

图 6.2.3　椭圆类测试(1)

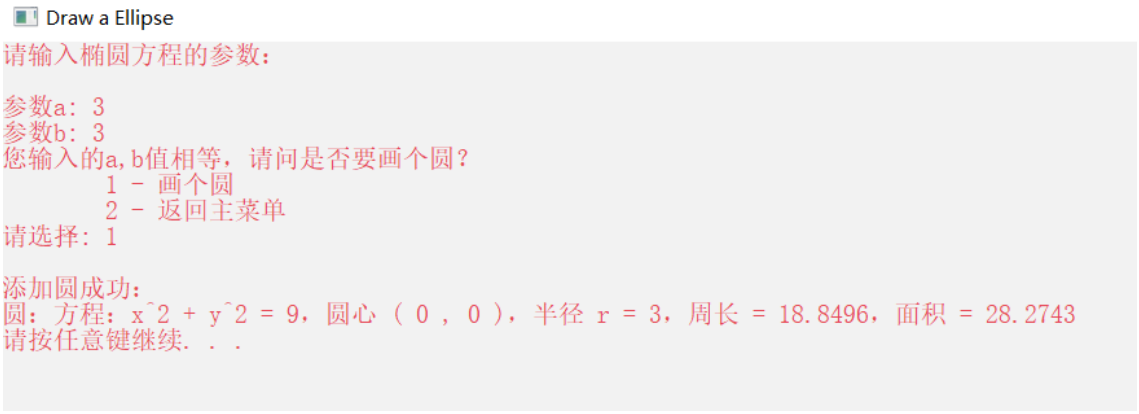

图 6.2.4　椭圆类测试(2)

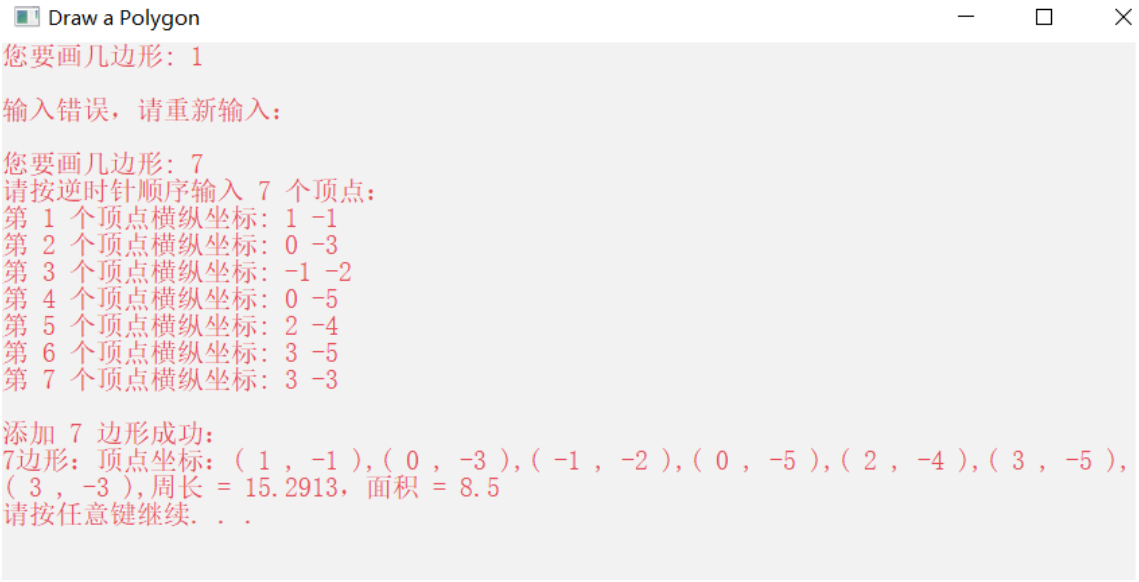

图 6.2.5　多边形类测试

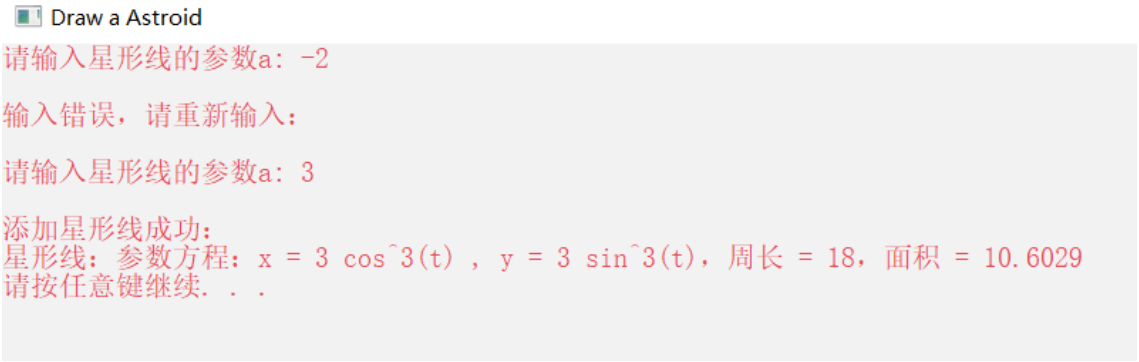

图 6.2.6　星形线类测试

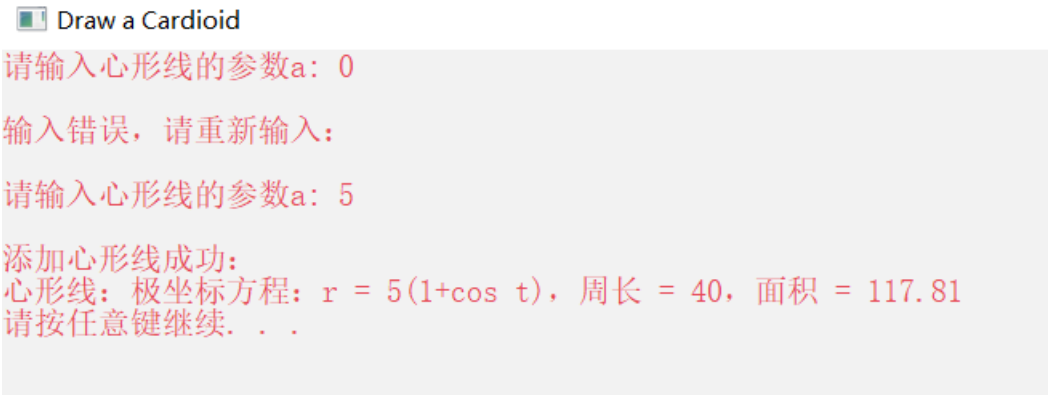

图 6.2.7　心形线类测试

表 6-1　输入样例

图形类型	#	输　入	测试用例说明
线段	1-1	2　1 1　-2	方程：$3x-y-5=0$ A，B，C 均不为 0，B = -1
	1-2	0　0.2 7　0.2	方程：$5y-1=0$ A = 0，与 x 轴平行直线，B = -1
	1-3	-7　0 -7　7	方程：$x+7=0$ B = 0，与 y 轴平行直线，A = 1
	1-4	2　-5 -0.2　0.5	方程：$5x+2y=0$ C = 0，A 无“+”号
	1-5	2　0 -4　0	方程：$y=0$ A = 0，C = 0
	1-6	0　-3 0　1	方程：$x=0$ B = 0，C = 0
	1-7	2　3 2　3	输入数据非法：两点重合，不能构成线段
圆	2-1	2　3 4	方程：$(x-2)^2+(y-3)^2=16$
	2-2	-3　0 2	方程：$(x+3)^2+y^2=4$ 圆心纵坐标为 0
	2-3	0　-1 -7	输入数据非法：圆的半径应大于 0
	2-4	5　3 0	输入数据非法：圆的半径应大于 0

续表

图形类型	#	输　入	测试用例说明
椭圆	3-1	2 4	方程：$\frac{y^2}{16}+\frac{x^2}{4}=1$
	3-2	4 2	方程：$\frac{x^2}{16}+\frac{y^2}{4}=1$
	3-3	-4 0	输入数据非法：a，b 均应大于 0
	3-4	0 2	输入数据非法：a，b 均应大于 0
	3-5	0 0	输入数据非法：a，b 均应大于 0
	3-6	3 3	方程：$x^2+y^2=9$ a = b，其实是个圆
多边形	4-1	3 1　1 -2　1 -3　-2	三角形 周长 = $8+\sqrt{10}$，面积 = $\frac{9}{2}$
	4-2	4 0　0 3　-1 4　1 1　2	平行四边形 周长 = $2\sqrt{5}+2\sqrt{10}$，面积 = 7
	4-3	7 1　-1 0　-3 -1　-2 0　-5 2　-4 3　-5 3　-3	不规则七边形 周长 = $2+4\sqrt{2}+2\sqrt{5}+\sqrt{10}$ 面积 = $\frac{17}{2}$
	4-4	1	输入数据非法
星形线	5-1	3	方程：$\begin{cases}x=3\cos^3 t\\y=3\sin^3 t\end{cases}$
	5-2	-2	输入数据非法
心形线	6-1	5	方程：$r=5(1+\cos t)$
	6-2	0	输入数据非法

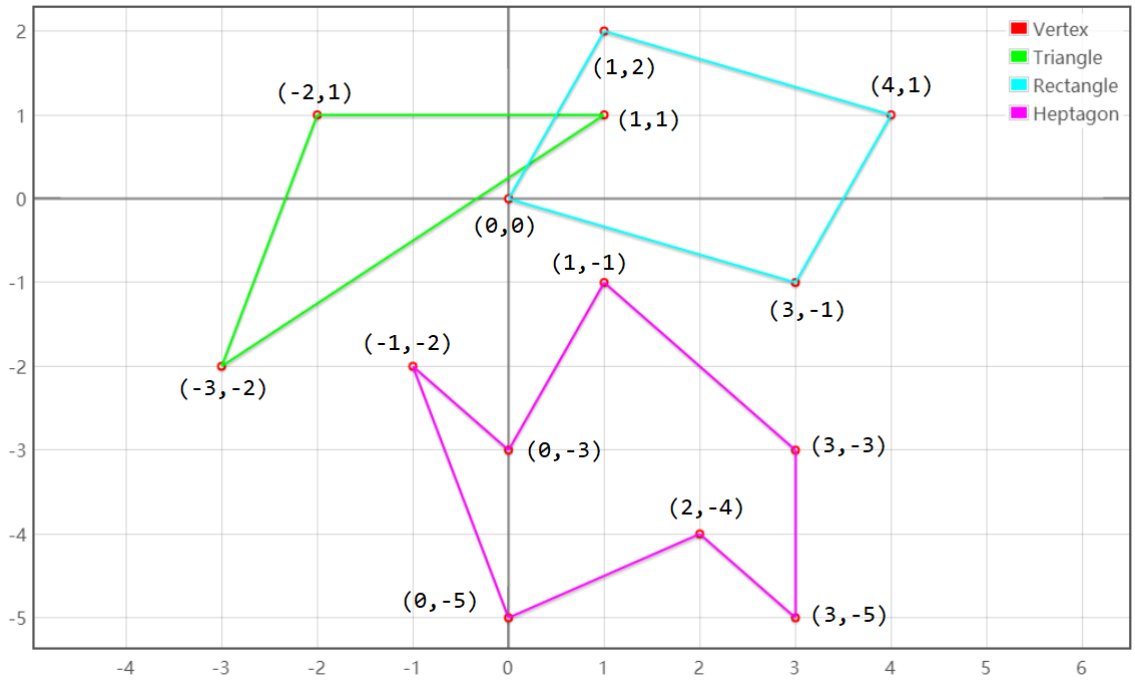

图 6.2.8　样例 4-1、4-2、4-3 示意图

6.3　查看已经画过的图形

图 6.3.1　未添加任何图形，输出提示信息

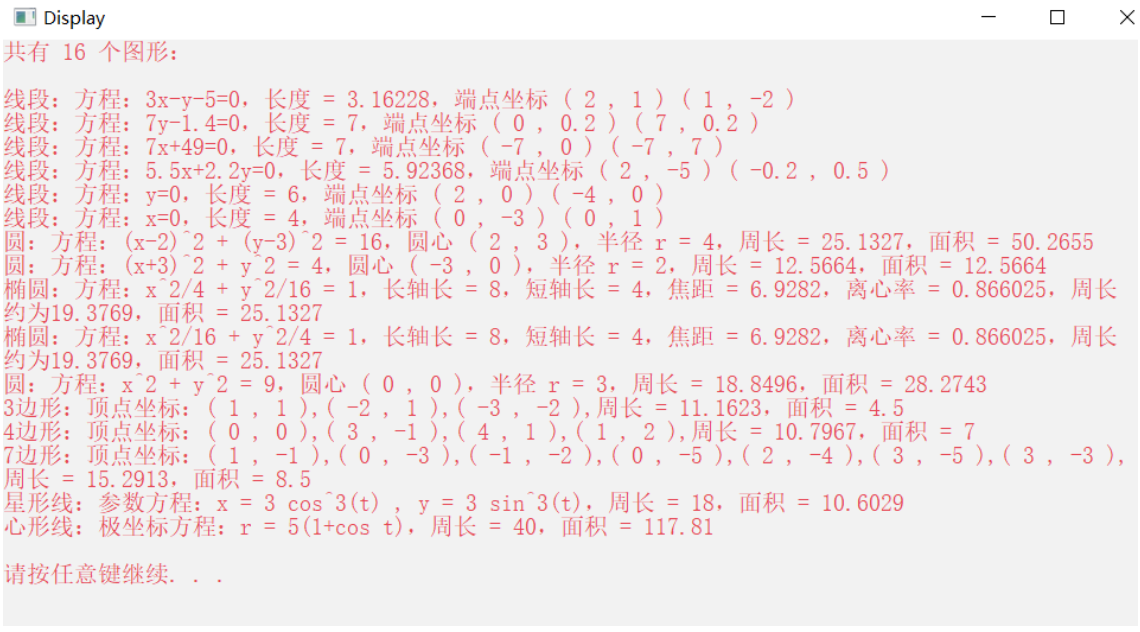

图 6.3.2　查看全部图形

6.4 保存并退出

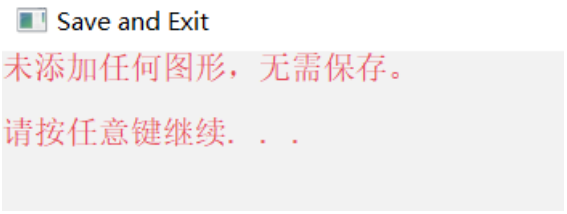

图 6.4.1 无任何图形时，输出提示信息

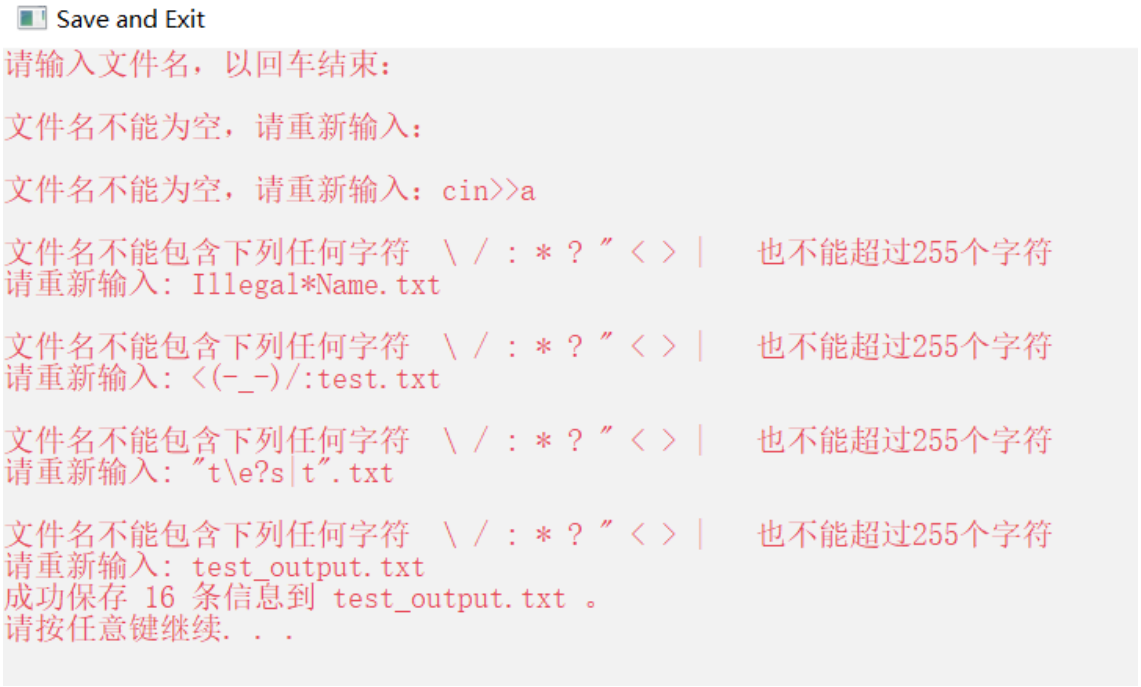

图 6.4.2 保存到文件

图 6.4.3 保存的文件：test_output.txt

7 实验总结及体会

本次《面向对象程序设计综合应用》实验是上一次《面向过程程序设计》实验的延续，是在面向过程的基础上，通过引入面向对象的思想——封装、继承和多态，来构造具有一定弹性和扩展型的面向对象程序。

如果说上一次实验锻炼了我程序设计、代码调试的能力，本次实验主要锻炼了我查阅资料的能力。计算机科学在飞速发展，程序设计语言也在不断地修订与更新。能够阅读有关资料，是一个合格的程序员必备的素养。本次实验过程中我查阅了许多资料。为了使用 IO 库，以及 vector 对象，我阅读了《C++ Primer》相关章节的内容；为了准确的画出 UML 类图，我仔细阅读了《C++程序设计基础教程》中的相关内容。

能够成功的完成本次实验，我要感谢《高级程序设计》课的任课教师许莉老师。虽然这门课中面向对象部分的课时有限，但通过她认真、细致的讲解，我初步建立起了面向对象的思想，能够运用封装、继承与多态机制构造简单的面向对象的程序。

虽然本次实验时间紧、任务重，但凭借着一股不放弃的精神，我顺利的完成了程序设计与实验报告的编写工作。由于时间和能力有限，这个程序中还存在着一些不足之处。但我相信随着今后的深入学习，一定可以弥补这些不足之处。我将在追求完美的路上不断前行，不断鞭策自己努力学习，提高自己的代码设计能力和程序调试能力。

8 源程序（含注释）

```
/* ******************************************************
|              面向对象程序设计综合应用
|     选    题  画图软件的设计与实现 Draw It
|     姓    名  黄远鹏      学    号  183401050225
|     班    级  软件 1802
|     完成时间  2019 年 7 月 7 日
\  ****************************************************** */
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
#include <regex>
using namespace std;

const double PI=acos(-1.0);     // PI 圆周率
// equals0：判断浮点数是否为 0     返回值：true（为 0），false（非 0）
inline bool equals0(double num)
{
    return fabs(num) < 1e-8;
}
class Shape     // Shape 类：作为其他类的基类
{
protected:
    double perimeter,area;     // perimeter 周长,area 面积
public:
    Shape() {}
    Shape(double p,double a):perimeter(p),area(a) {}
    // output：输出函数
    // 传入参数及返回值：输出流的引用 std::ostream &os,默认参数为 std::cout
    virtual ostream &output(ostream &os = cout)
    {
        os << "周长 = " << perimeter << "，面积 = " << area;
        return os;
    }
    virtual ~Shape() {}
};
vector<Shape*> addr;     // 存放画过的图形的地址
class Point : public Shape     // 点类
```

```
{
private:
    double x,y;
public:
    Point() {}
    Point(double a,double b):Shape(0,0),x(a),y(b) {}
    // dist：计算两点间的距离
    // 传入参数：另一个点 b(Point 常量引用)   返回值：a,b 间距离(double)
    double dist(const Point &b)
    {
        double dx=x-b.x,dy=y-b.y;
        return sqrt(dx*dx + dy*dy);
    }
    double getX() { return x; }
    double getY() { return y; }
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "( " << x << " , " << y << " )";
        return os;
    }
    ~Point() {}
};
class Line : public Shape     // 线段类
{
private:
    Point m,n;      // 线段的两个端点
    double A,B,C;   // 直线方程的三个参数
public:
    Line() {}
    Line(double mx,double my,double nx,double ny):Shape(),m(mx,my),n(nx,ny),
        A(ny-my),B(mx-nx),C(nx*my-mx*ny)
    {
        area = 0;
        perimeter = m.dist(n);
        if(A<0)   // 让第一项为正
            A *= -1, B *= -1, C *= -1;
        else if(equals0(A) && B<0)
            B *= -1, C *= -1;
        if(equals0(A) && equals0(C)) B=1;   // A,C 都为 0，B 恒为 1,例如 3y=0 即
y=0
        if(equals0(B) && equals0(C)) A=1;   // B,C 都为 0，A 恒为 1
    }
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "线段：方程：";
```

```
        if(!equals0(A))     // A 为 0 时不输出此项
        {
            if(equals0(A-1)) os << "x";   // A 为正 1 不输出参数
            else if(equals0(A+1)) os << "-x";   // A 为负 1 不输出 1 只输出负号
            else  os << A << "x";
        }
        if(!equals0(B))     // B 为 0 时不输出此项
        {
            if(!equals0(A))   // 前一项非 0 时输出+或-号
            {
                if(equals0(B-1))  // B 为正 1,例如 2x+y
                    os << "+y";
                else if(equals0(B+1))   // B 为负 1,例如 2x-y
                    os << "-y";
                else
                    os << showpos << B << "y" << noshowpos;
            }
            else   // 前项为 0，不输出+或-号
            {
                if(equals0(B-1))  // y
                    os << "y";
                else if(equals0(B+1))   // -y
                    os << "-y";
                else
                    os << B << "y";
            }
        }
        if(!equals0(C))   // C 为 0 时不输出
            os << showpos << C << noshowpos;
        os << "=0，长度 = " << m.dist(n) << "，端点坐标 ";
        m.output(os) << " ";
        n.output(os);
        return os;
    }
    ~Line() {}
};
// make_Line：根据用户输入，生成一条线段并添加其地址到 addr 中
void make_Line()
{
    system("title Draw a Line");
    double x1,y1,x2,y2;
    cout << "请输入线段两个端点的坐标，用空格隔开：\n" << endl;
    cout << "第一个点 x,y 坐标:" << ends;
    cin >> x1 >> y1;
    cout << "第二个点 x,y 坐标:" << ends;
```

```
    cin >> x2 >> y2;
    while(fabs(x1 - x2) < 1e-8 && fabs(y1 - y2) < 1e-8)     // 两点重合，不
能构成线段
    {
        cout << "\n 输入的两点重合，不能构成线段。请重新输入：" << endl;
        cout << "第一个点 x,y 坐标:" << ends;
        cin >> x1 >> y1;
        cout << "第二个点 x,y 坐标:" << ends;
        cin >> x2 >> y2;
    }
    Line *L=new Line(x1,y1,x2,y2);
    addr.push_back(L);     // 新生成的线段添加到地址序列中
    cout << endl << "添加线段成功：" << endl;
    L->output() << endl;
    system("pause");
    return;
}
class Circle : public Shape     // 圆类
{
private:
    Point o;    // 圆心
    double r;   // 半径
public:
    Circle() {}
    Circle(double                    x_,double                    y_,double
r_):Shape(PI*r_*2,PI*r_*r_),o(x_,y_),r(r_) {}
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "圆：方程：" << showpos;
        if(equals0(o.getX())) os << "x^2";   // 圆心在 y 轴上，不输出横坐标
        else os << "(x"  << -o.getX() << ")^2";
        if(equals0(o.getY())) os << " + y^2";   // 圆心在 x 轴上，不输出纵坐标
        else os << " + (y" << -o.getY() << ")^2";
        os << noshowpos << " = " << r*r << "，圆心 ";
        o.output(os) << "，半径 r = " << r << "，";
        Shape::output(os);
        return os;
    }
    ~Circle() {}
};
// make_Circle：根据用户输入，生成一个圆并添加其地址到 addr 中
void make_Circle()
{
    system("title Draw a Circle");
    double x,y,r;
```

```
    cout << "请输入圆心的横纵坐标，用空格隔开:" << ends;
    cin >> x >> y;
    cout << "请输入圆的半径:" << ends;
    cin >> r;
    while(r<=0)  // 半径必须大于 0
    {
        cout << "\n 输入错误，圆的半径应大于 0，请重新输入：\n" << endl;
        cout << "请输入圆心的横纵坐标，用空格隔开:" << ends;
        cin >> x >> y;
        cout << "请输入圆的半径:" << ends;
        cin >> r;
    }
    Circle *c = new Circle(x,y,r);
    addr.push_back(c);
    cout << endl << "添加圆成功：" << endl;
    c->output() << endl;
    system("pause");
    return;
}
class Ellipse : public Shape     // 椭圆类
{
private:
    double a,b,c,e;   // a 长半轴，b 短半轴，c 焦半径，e 离心率
public:
    Ellipse() {}
    Ellipse(double a_,double b_):Shape(),a(a_),b(b_)
    {
        double p = a>b ? a : b,q = a>b ? b : a;
        c=sqrt(p*p - q*q);

    perimeter=PI*(b+a)*(1+3*pow((b-a)/(b+a),2)/(10+sqrt(4-3*pow((b-a)/(b+a
),2))));
        area=PI*a*b;
        e=c/p;
    }
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "椭圆：方程：";
        os << "x^2";
        if(!equals0(a-1)) os << "/" << a*a;   // a=1 时不输出 a^2
        os << " + y^2";
        if(!equals0(b-1)) os << "/" << b*b;   // b=1 时不输出 b^2
        os << " = 1，长轴长 = " << ( (a>b) ? a*2 : b*2 )
            << "，短轴长 = " << ( (a>b) ? b*2 : a*2 )
            << "，焦距 = " << c*2 << "，离心率 = " << e
```

```
            << "，周长约为" << perimeter << "，面积 = " << area;
        return os;
    }
    ~Ellipse() {}
};
// make_Ellipse：根据用户输入，生成一个椭圆并添加其地址到 addr 中
void make_Ellipse()
{
    system("title Draw a Ellipse");
    double a,b;
    cout << "请输入椭圆方程的参数：\n" << endl;
    cout << "参数 a:" << ends;
    cin >> a;
    cout << "参数 b:" << ends;
    cin >> b;
    while(a<=0 || b<=0)   // 输入参数非法
    {
        cout << "\n 输入错误，请重新输入：\n" << endl;
        cout << "请输入椭圆方程的参数：" << endl;
        cout << "参数 a:" << ends;
        cin >> a;
        cout << "参数 b:" << ends;
        cin >> b;
    }
    if(equals0(a-b))   // a==b,其实是个圆
    {
        cout << "您输入的 a,b 值相等，请问是否要画个圆？" << endl;
        cout << "        1 - 画个圆\n        2 - 返回主菜单" << endl;
        cout << "请选择:" << ends;
        int choice1;
        cin >> choice1;
        if(choice1==1)
        {
            Circle *c = new Circle(0,0,a);
            addr.push_back(c);
            cout << endl << "添加圆成功：" << endl;
            c->output() << endl;
            system("pause");
        }
        return;
    }
    Ellipse *e = new Ellipse(a,b);
    addr.push_back(e);
    cout << endl << "添加椭圆成功：" << endl;
    e->output() << endl;
```

```
    system("pause");
    return;
}
class Astroid : public Shape     // 星形线
{
private:
    double a;   // 星形线方程唯一的参数
public:
    Astroid() {}
    Astroid(double a_):Shape(a_*6,(PI*a_*a_*3)/8),a(a_) {}
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "星形线：参数方程：";
        os << "x = " << a << " cos^3(t) , y = " << a << " sin^3(t)，";
        Shape::output(os);
        return os;
    }
    ~Astroid() {}
};
// make_Astroid：根据用户输入，生成一条星形线并添加其地址到 addr 中
void make_Astroid()
{
    system("title Draw a Astroid");
    double a;
    cout << "请输入星形线的参数 a:" << ends;
    cin >> a;
    while(a<=0)   // 输入参数非法
    {
        cout << "\n 输入错误，请重新输入：\n" << endl;
        cout << "请输入星形线的参数 a:" << ends;
        cin >> a;
    }
    Astroid *as = new Astroid(a);
    addr.push_back(as);
    cout << endl << "添加星形线成功：" << endl;
    as->output() << endl;
    system("pause");
    return;
}
class Cardioid : public Shape     // 心形线
{
private:
    double a;
public:
    Cardioid() {}
```

```
    Cardioid(double a_):Shape(a_*8,(PI*a_*a_*3)/2),a(a_) {}
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << "心形线：极坐标方程：";
        os << "r = " << a << "(1+cos t)，";
        Shape::output(os);
        return os;
    }
    ~Cardioid() {}
};
// make_Cardioid：根据用户输入，生成一条心形线并添加其地址到 addr 中
void make_Cardioid()
{
    system("title Draw a Cardioid");
    double a;
    cout << "请输入心形线的参数 a:" << ends;
    cin >> a;
    while(a<=0)   // 输入参数非法
    {
        cout << "\n 输入错误，请重新输入：\n" << endl;
        cout << "请输入心形线的参数 a:" << ends;
        cin >> a;
    }
    Cardioid *c = new Cardioid(a);
    addr.push_back(c);
    cout << endl << "添加心形线成功：" << endl;
    c->output() << endl;
    system("pause");
    return;
}
class Polygon : public Shape     // 多边形
{
private:
    int nLength;     // 边的数目
    vector<Point> vertex;     // 多边形的每个顶点
public:
    Polygon() {}
    Polygon(int n,vector<Point> v):Shape(0,0),nLength(n),vertex(v)
    {
        perimeter += v[n-1].dist(v[0]);
        area += v[n-1].getX()*v[0].getY() - v[0].getX()*v[n-1].getY();
        for(int i=0;i<nLength-1;i++)
        {
            perimeter += v[i].dist(v[i+1]);
            area += v[i].getX()*v[i+1].getY() - v[i+1].getX()*v[i].getY();
```

```
        }
        area /= 2;
    }
    ostream &output(ostream &os = cout)   // 输出函数
    {
        os << nLength << "边形：顶点坐标：";
        for(auto v=vertex.begin();v!=vertex.end();v++)
            v->output(os) << ",";
        Shape::output(os);
        return os;
    }
    ~Polygon() {}
};
// make_Polygon：根据用户输入，生成一个多边形并添加其地址到 addr 中
void make_Polygon()
{
    system("title Draw a Polygon");
    int n;
    double x,y;
    cout << "您要画几边形:" << ends;
    cin >> n;
    while(n<3)   // 输入非法
    {
        cout << "\n 输入错误，请重新输入：\n" << endl;
        cout << "您要画几边形:" << ends;
        cin >> n;
    }
    cout << "请按逆时针顺序输入 " << n << " 个顶点：" << endl;
    vector<Point> v;
    for(int i=1;i<=n;i++)
    {
        cout << "第 " << i << " 个顶点横纵坐标:" << ends;
        cin >> x >> y;
        v.push_back(Point(x,y));
    }
    Polygon *p = new Polygon(n,v);
    addr.push_back(p);
    cout << endl << "添加 " << n << " 边形成功：" << endl;
    p->output() << endl;
    system("pause");
    return;
}
// printDirection：输出提示信息
inline void printDirection()
{
```

```
    cout << "   Draw It!" << endl << endl;
    cout << "       1 - 画一条线段"   << endl;
    cout << "       2 - 画一个圆"     << endl;
    cout << "       3 - 画一个椭圆"   << endl;
    cout << "       4 - 画个多边形"   << endl;
    cout << "       5 - 画一条星形线" << endl;
    cout << "       6 - 画一条心形线" << endl;
    cout << "       7 - 查看已经画过的图形" << endl;
    cout << "       0 - 保存并退出"   << endl;
    cout << "请选择:" << ends;
}
// display：查看已经画过的图形
void display()
{
    system("title Display");
    if(addr.empty())   // addr 为空
    {
        cerr << "未添加任何图形。" << endl;
        system("pause");
        return;
    }
    cout << "共有 " << addr.size() << " 个图形：" << endl << endl;
    for(auto it=addr.begin(),end=addr.end(); it!=end; ++it)
        (*it)->output() << endl;
    cout << endl;
    system("pause");
    return;
}
// saveAndExit：输出所有图形到指定文件,释放申请的内存
void saveAndExit()
{
    system("title Save and Exit");
    if(addr.empty())   // addr 为空
    {
        cerr << "未添加任何图形，无需保存。\n" << endl;
        system("pause");
        return;
    }
    cout << "请输入文件名，以回车结束：";
    string outFile;
    getline(cin,outFile);
    while(outFile.empty())
    {
        cerr << "\n 文件名不能为空，请重新输入：";
        getline(cin,outFile);
```

```
    }
    regex r("^[^/\\\\:\\*\\?\\<\\>\\|\"]{1,255}$");
    while(!regex_match(outFile,r))     // 输入的文件名非法
    {
        cerr << "\n 文件名不能包含下列任何字符  \\ / : * ? \" < > |   也不能超过 255 个字符" << endl;
        cerr << "请重新输入:" << ends;
        getline(cin,outFile);
    }
    ofstream fstrm(outFile);
    int cnt = 0;     // 保存输出信息条数
    for(auto it=addr.begin(),end=addr.end(); it!=end; ++it)
    {
        (*it)->output(fstrm) << endl;
        ++cnt;
    }
    cout << "成功保存 " << cnt << " 条信息到 " << outFile << " 。" << endl;
    fstrm.close();
    for(auto it=addr.begin(),end=addr.end(); it!=end; ++it)
        delete *it;
    system("pause");
    return;
}
int main()
{
    system("color fc");
    string choice;     // choice 接收用户的指令
    while(true)
    {
        system("cls");     // 清屏
        system("title Draw It!");     // 设置窗口标题
        fflush(stdin);
        printDirection();

        getline(cin,choice);
        if(choice.size()!=1)   // 输入字符过多，说明输入有误
        {
            cerr << "输入错误，请重新输入。" << endl;
            system("pause");
            continue;
        }
        switch(choice[0])
        {
        case '1':system("cls"); make_Line();     break;   // 画一条线段
        case '2':system("cls"); make_Circle();   break;   // 画一个圆
```

```
        case '3':system("cls"); make_Ellipse();  break;   // 画一个椭圆
        case '4':system("cls"); make_Polygon();  break;   // 画个多边形
        case '5':system("cls"); make_Astroid();  break;   // 画一条星形线
        case '6':system("cls"); make_Cardioid(); break;   // 画一条心形线
        case '7':system("cls"); display();       break;   // 查看已经画过的图
形
        case '0':system("cls"); saveAndExit();   return 0;   // 保存并退出

        default:
            {
                cerr << "输入错误，请重新输入。" << endl;
                system("pause");
                continue;
            }
        }
    }

    return 0;
}
```