第7章　数据的抽象与封装

1、实体、对象与类的概念	2、类的定义	3、对象声明与引用	4、私有、公有与保护
5、日期类的设计	6、两种程序设计思想	7、汽车类的设计	8、几何图形圆类的设计
9、构造函数的定义	10、重载构造函数	11、析构函数的定义	12、整数翻译函数
13、实际意义的析构函数	14、Person类的设计	15、对象与指针	16、this指针

再述OOP的抽象

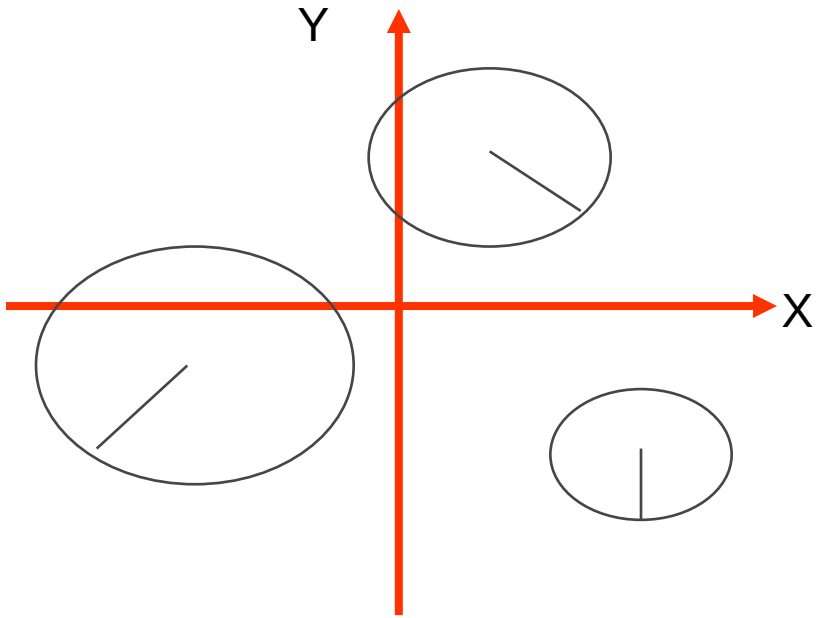
- 将客观世界中的实体抽象描述为类
  - 分析抽象某个实体有哪些特征：数据成员
  - 分析抽象某个实体有哪些功能或操作：函数成员
- 实体与实体的不同就在于特征和功能的不同
- 抽象描述因人而异，即设计产生的类各不相同
  - 有些人抽象得准确、完备
  - 有些人抽象得粗糙、简单

再述OOP的抽象

- 程序设计目的是让计算机自动去求解问题
  - 首先搞清楚问题涉及哪些实体(对象)
  - 例如求自幂数？涉及正整数对象，即整数变量
  - 求圆面积？涉及浮点对象或双精度对象
  - 加密解密？涉及字符串对象，即字符数组变量
  - 在选取或定义某个实体对象时，尽可能是较好的选取或定义，使时空效率最佳
  - 数据成员占用空间少，函数成员运行速度快

设计平面上的几何图形：圆

- 如何抽象几何图形： 圆
- 圆的特征分析1：
  - 圆心坐标值（X，Y）和半径值R
- 圆的特征分析2：
  - 外切正方形，两个点坐标值
- 圆的特征分析3：
  - 内接正方形，两个点坐标值
- 圆的特征分析4：
  - 还有吗？圆周上的三个坐标值
- 圆的功能
  - 设置初值、计算面积、输出圆属性、得到圆属性等



设计平面上的几何图形：圆

- 先设计点类，再设计圆类
- 点的特征和功能
  - 特征：点坐标，X和Y轴上的整数值
  - 功能：设置坐标、打印坐标、得到坐标等
- 圆的特征和功能
  - 特征：点坐标和半径
  - 功能：设置初值、计算面积、输出圆属性、得到圆属性等

```
class Point      //点类定义
{
        int x, y; //点的x和y坐标
public:
        void InitPoint( int, int );  // 设置坐标
        int GetX() { return x; }     // 取x坐标
        int GetY() { return y; }     // 取y坐标
        void Print();                //输出点的坐标
};
```

```
//类外定义两个成员函数
void Point::InitPoint( int a, int b )
{
        x = a;
        y = b;
}
void Point::Print()
{
        cout << '[' << x << ", " << y << ']';
}
```

圆类的定义

```
class Circle   //圆类定义
{
private:
        double radius;    //定义半径对象（变量）
        Point Center;     //定义圆心对象（变量）
public:
        void InitCircle( double, Point );    //设置圆类的数据
        double GetRadius();                  //取半径
        Point GetCenter();                   //取圆心
        double Area();                       //计算面积a
        void Print();                        //输出圆心坐标和半径
};
```

圆类成员函数的定义

```
//类外定义成员函数
void Circle::InitCircle( double r, Point p )
 {
        radius = ( r >= 0 ? r : 0 );
        Center = p;
 }
double Circle::GetRadius() {  return radius;  }
Point Circle::GetCenter() {  return Center;  }
double Circle::Area() {return 3.14159 * radius * radius; }
void Circle::Print()
{
        cout << "Center = ";
        Center.Print();
        cout << "; Radius = " << radius << endl;
}
```

```
// 测试主函数
int main()
{
        Point p,center;
        p.InitPoint(30,50);
        center.InitPoint(120,80);
        Circle c;
        c.InitCircle(10.0 ,center);

        cout << "Point p:";
        p.Print();
        cout << endl;
        cout << "Circle c:";
        c.Print();
        cout << "The center of circle c:";
        c.GetCenter().Print();
        cout << "\nThe area of circle c:" << c.Area() << endl;
        return 0;
}
```