**综述：**使用MFC进行三种直线绘制算法的效率，误差，光滑度比较。

**框架：**使用MFC：  
cgDrawLine.h包括其他项目特定的标头（包括 Resource.h），并声明 Ccg2022XJDrawLineApp 应用程序类。

cgDrawLineDoc.h编辑文件以添加特殊文档数据并实现文件保存和加载。

CgDrawLineView.h用于查看 cgDrawLineDoc 对象，声明主要的绘制函数

cgControlSelect.h继承MainFrm类，用于一些控件的使用。

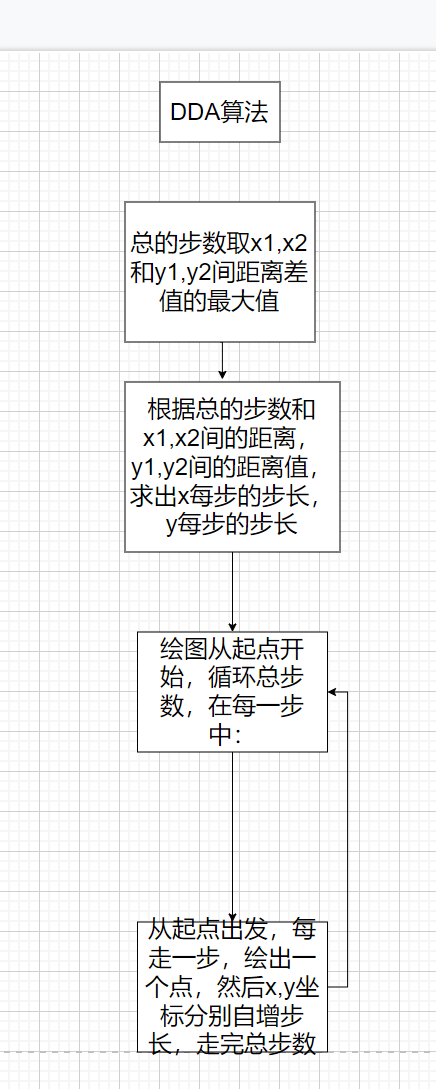
算法原理：

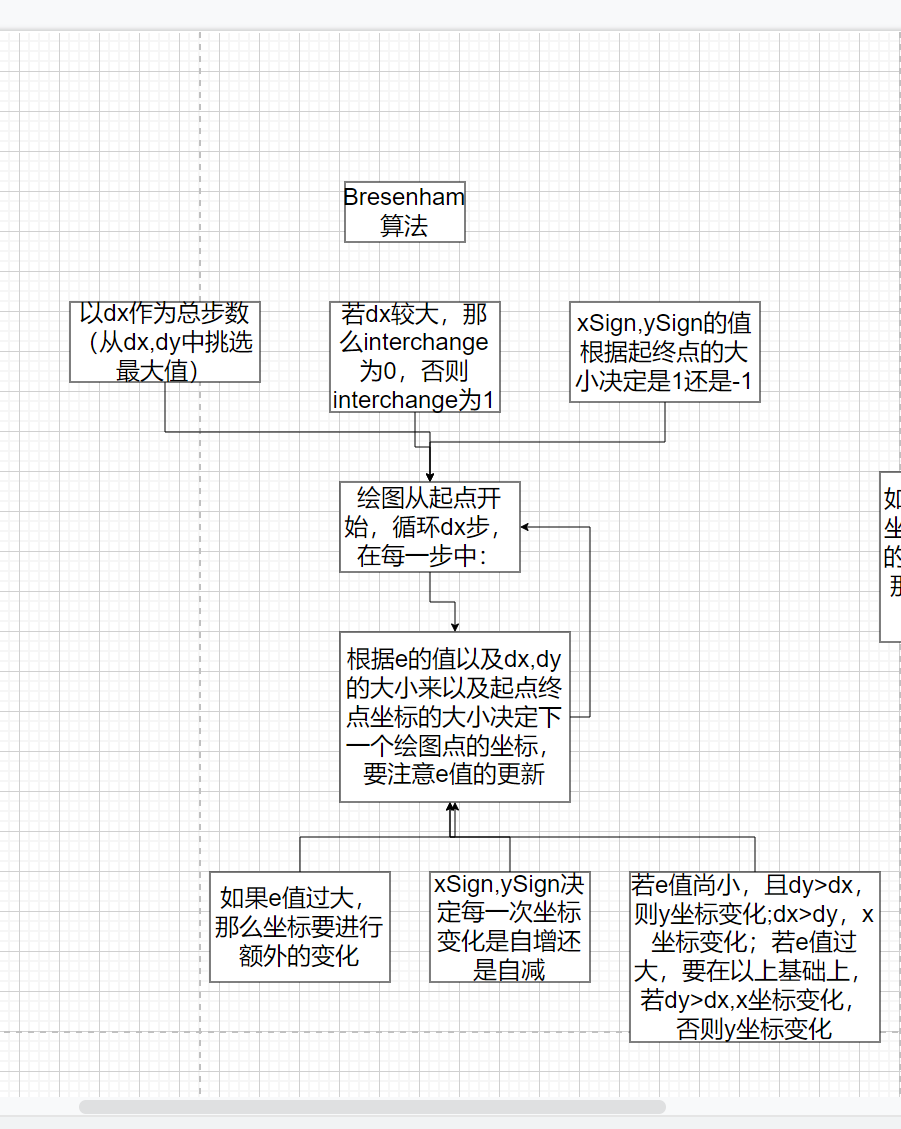
### DDA：直线的一阶导数是连续的，而且Δy与Δx是成一定比例的 因此我们可以通过将当前位置的坐标在每个方向上增加一定的增量来确定下一个点的位置。这就涉及到了增量算法：在一个迭代算法中，如果每一步的x,y值是前一步的值加上一个增量来获得的，则称为增量算法。

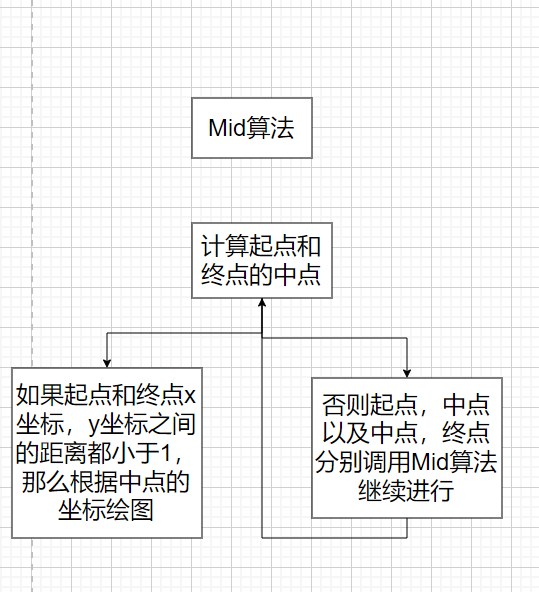
**Bresenham：**采用递推步进的办法，令每次最大变化方向的坐标步进一个象素，同时另一个方向的坐标依据误差判别式的符号来决定是否也要步进一个象素。

**Mid算法：**采用二分递归的方法，直到取中点的两点的距离足够小才进行绘制

**处理流程：**

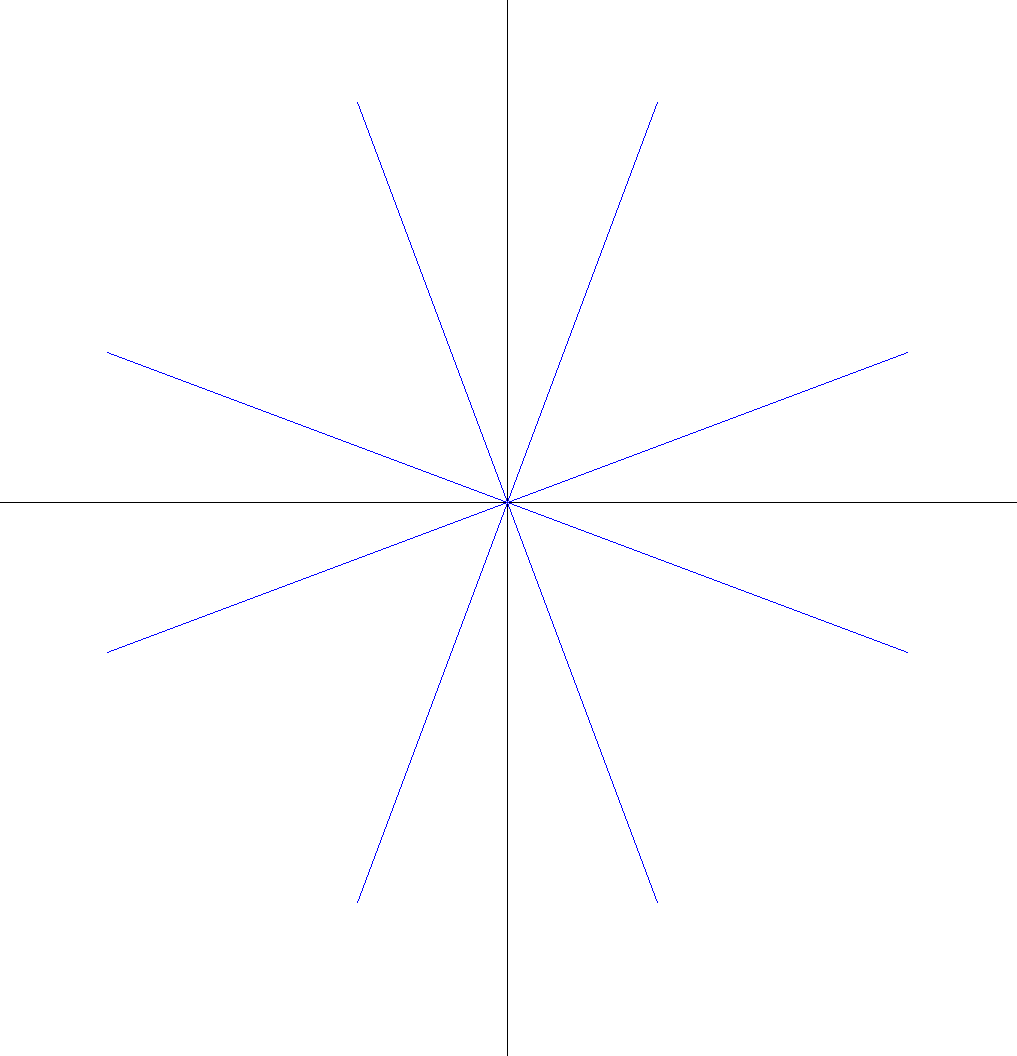
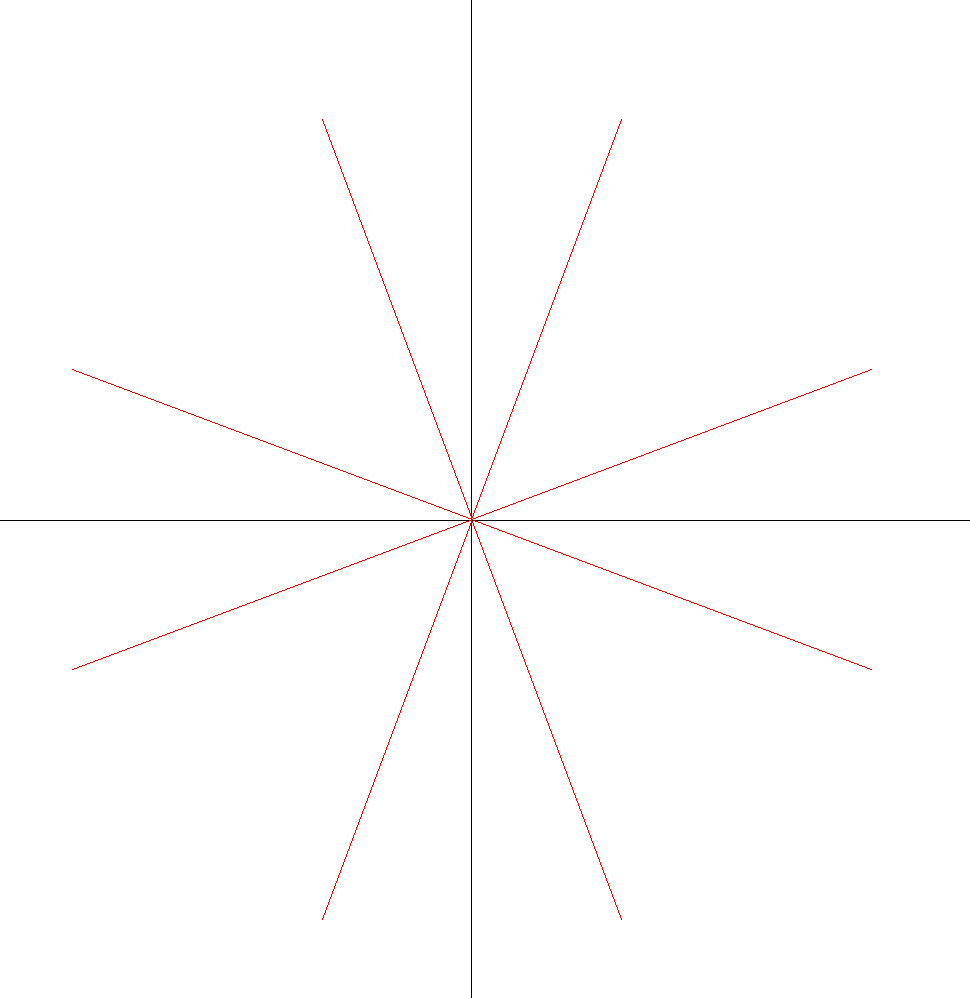
****

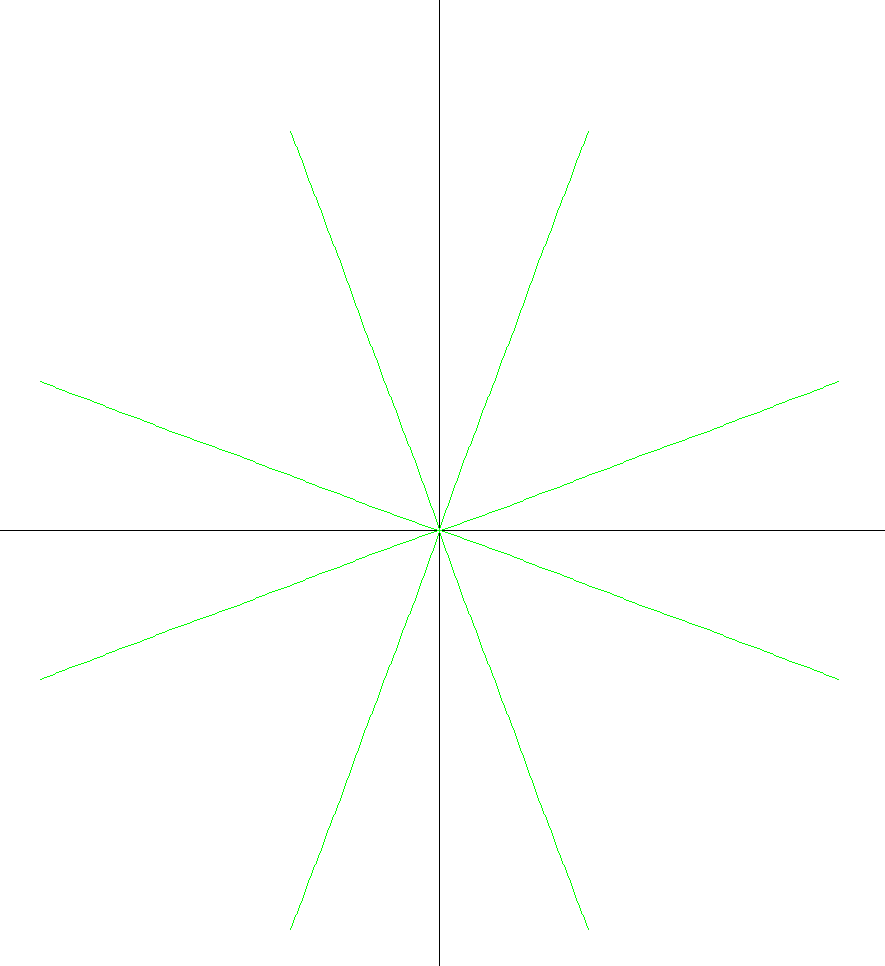
****

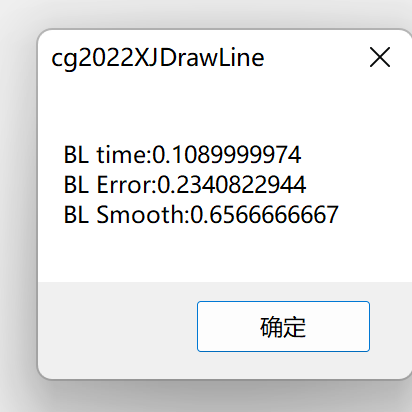
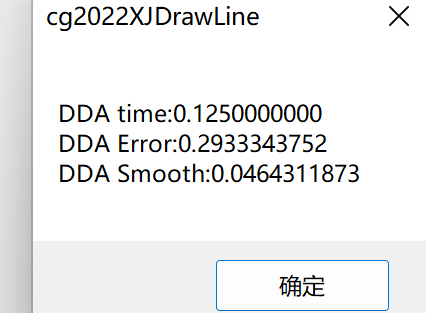
****

**算法分析说明：**

**三种算法所绘制：**

****

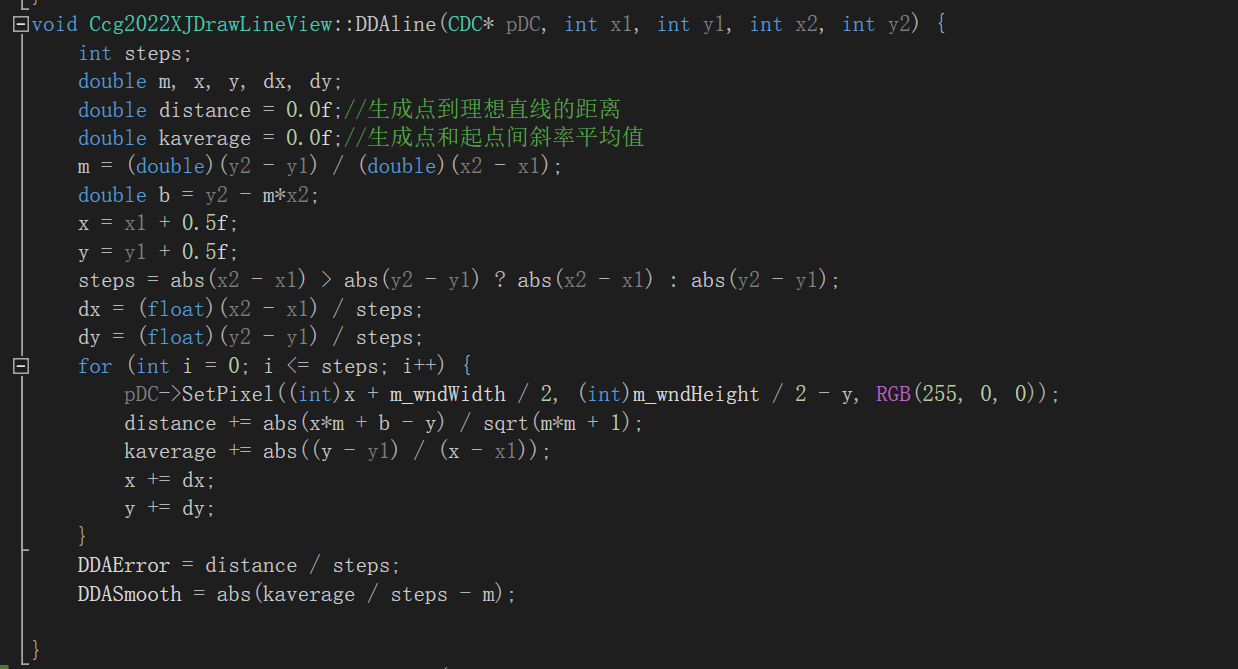
****

****

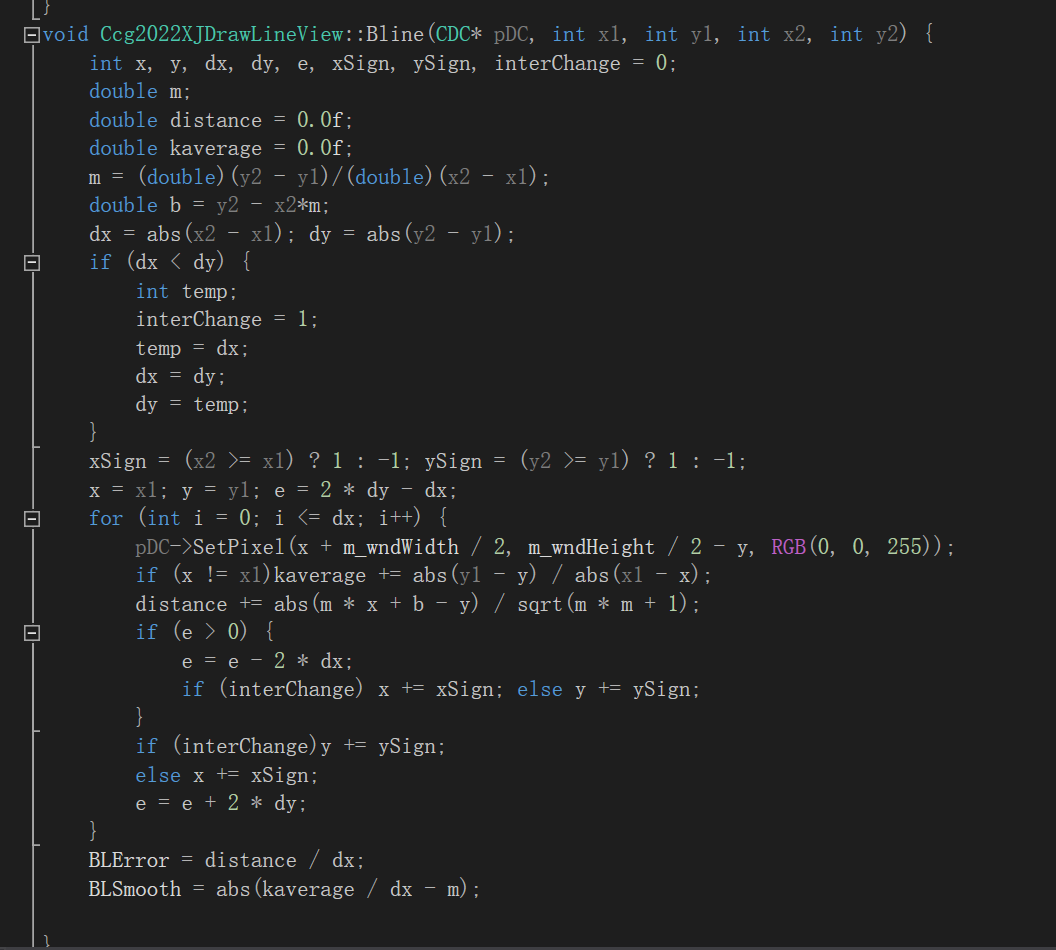
****

**每个算法都经过了10次循环，从效率上来说，Bresenham算法效率较高，mid算法和DDA算法次之；从误差上来说，M算法误差为0，B算法次之，D算法最差；从光滑度上来说，DDA算法较好，B算法次之，Mid算法最差。**

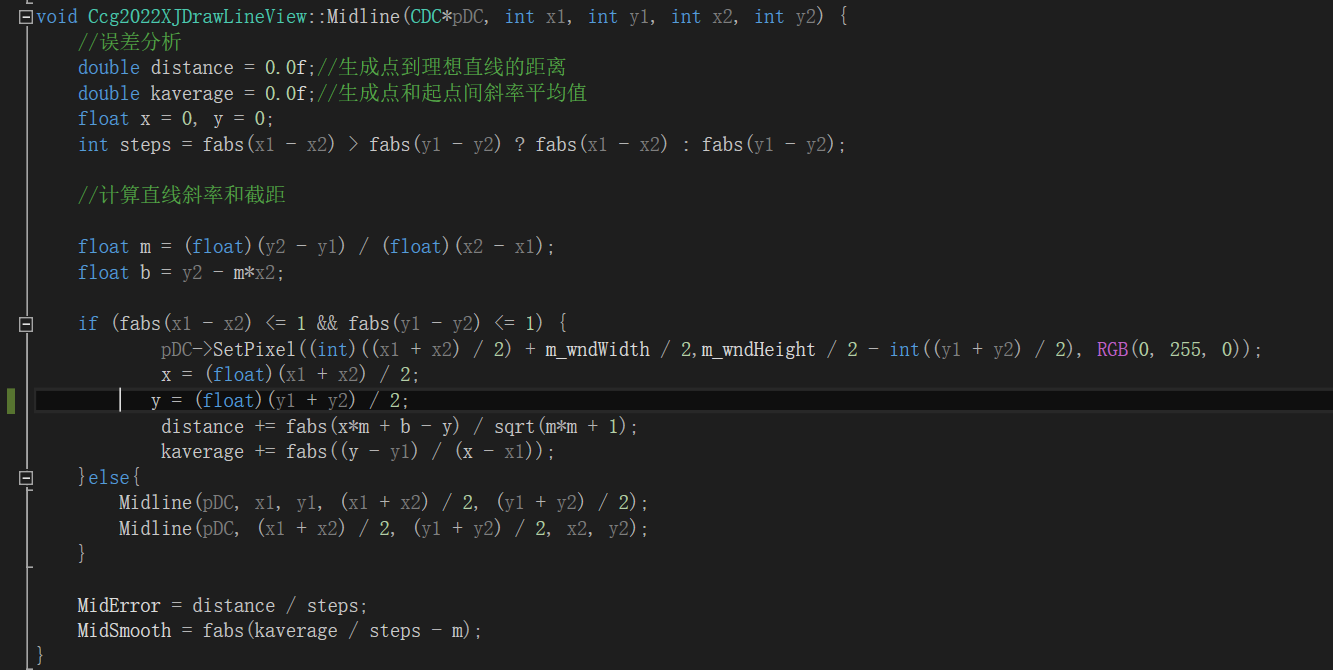
**核心代码：DDA:**

****

**B:**

****

**MID:**

****

**实验1扩展：编程实现Bresenham圆/圆弧生成算法**

**算法介绍：**

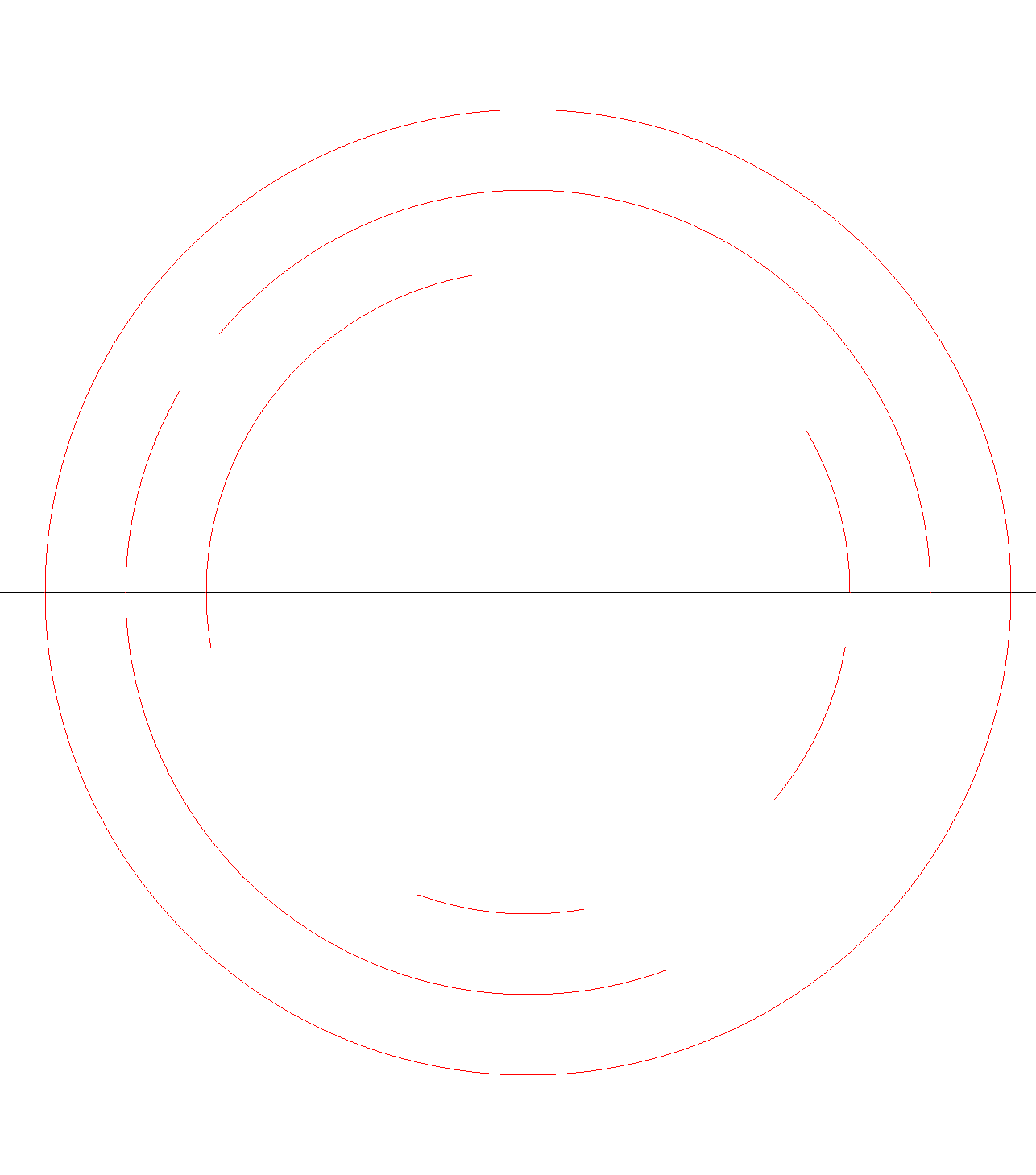
1. **起始点的下一个位置有三个供选点，哪一点的误差最小就选择哪一个点。但这种方法只能在45-90度范围内效果较好，其他范围内效果差，所以其他范围内的圆弧要通过这1/8的范围进行对称得到。**
2. **要注意的是有些范围角的范围和45-90度范围对称后角度是互补的，因此要注意这一部分角的处理。**

**算法效果：**

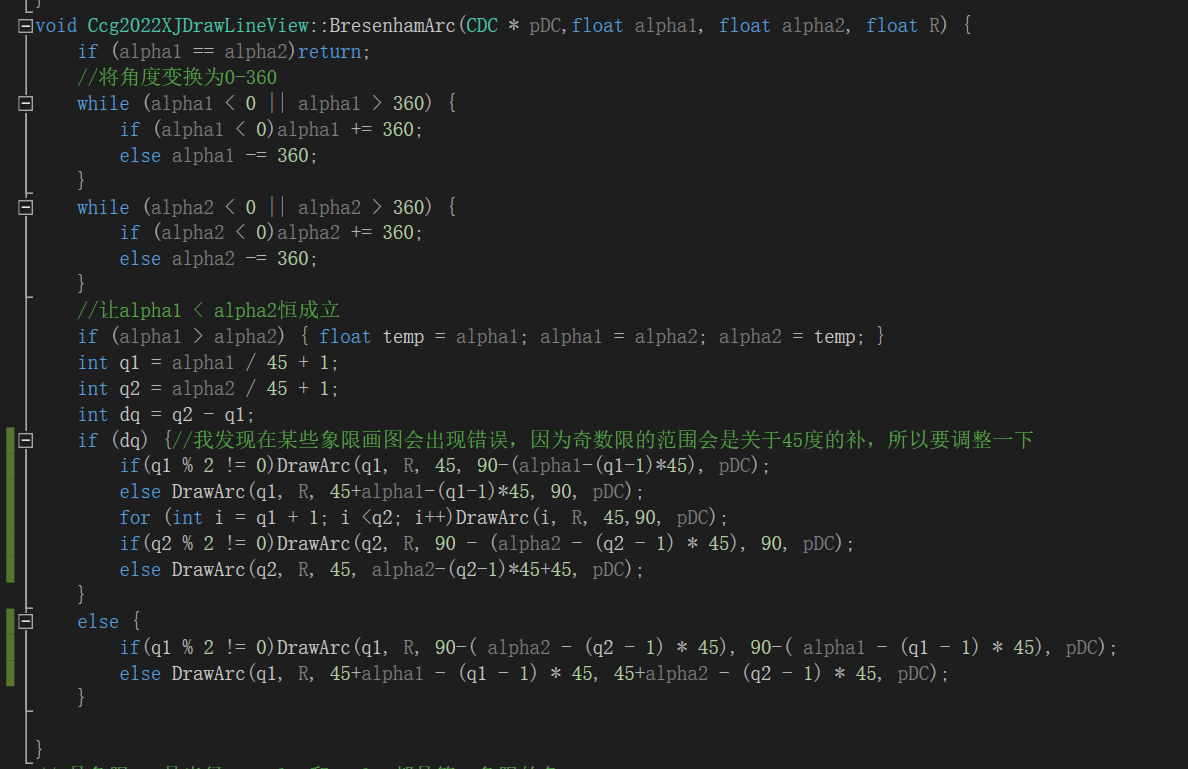
**内圈：0-30度，100-190度，250-280度，320-350度**

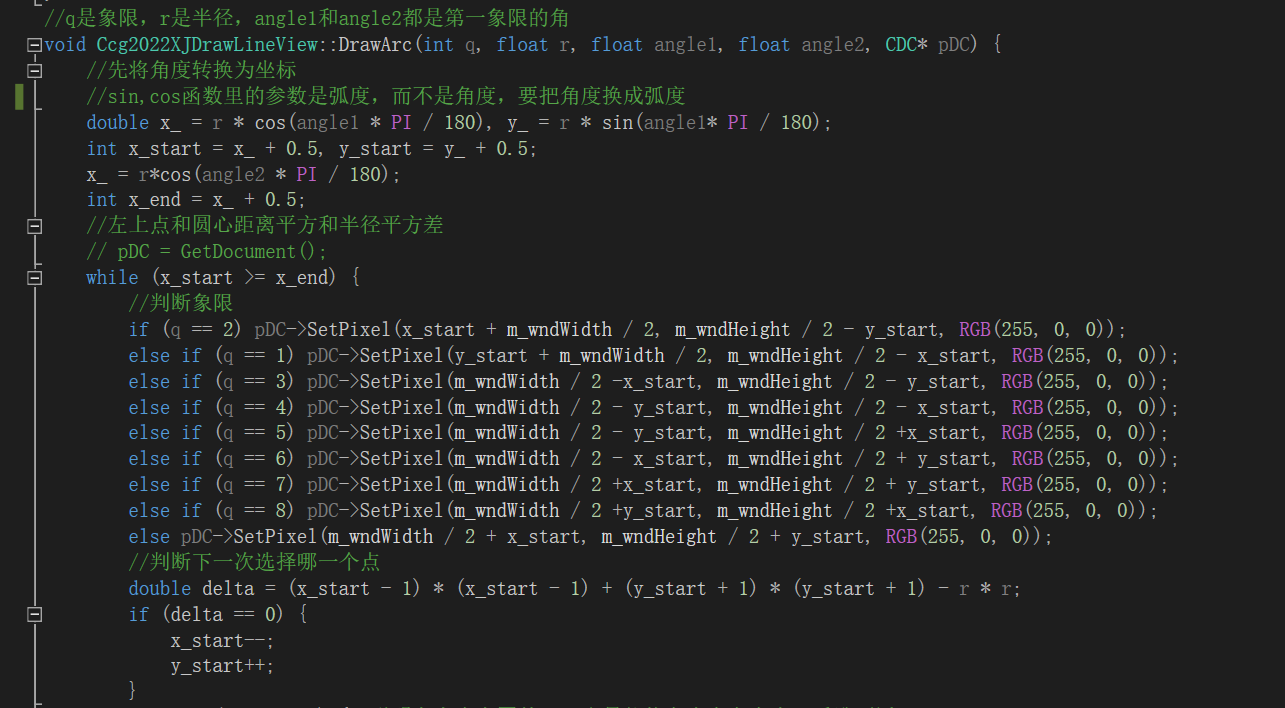
**中圈：0-140度，150-290度**

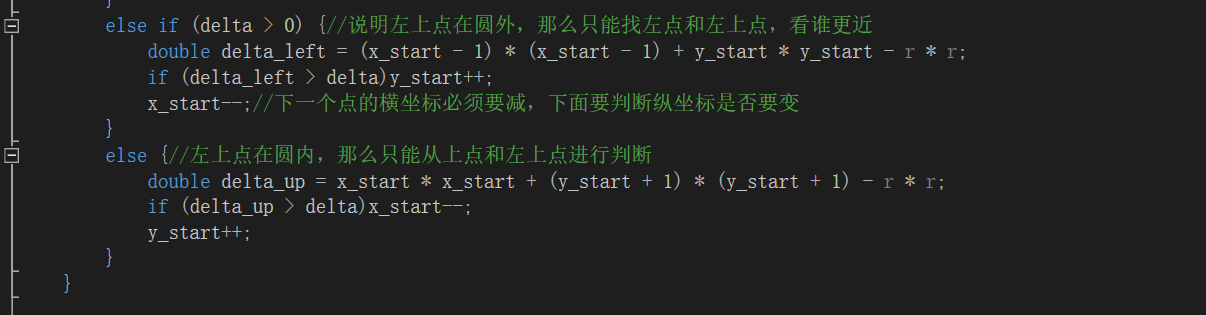
**外圈：0-360度**

****

**算法代码：**

****

****

****