中国矿业大学
CHINA UNIVERSITY OF MINING AND TECHNOLOGY

《计算机地图制图》实验报告

姓名：马骁

学号：07212393

班级：地信 21-1 班

中国矿业大学

2023 年 6 月

实验一：简单地图绘制编程（3 学时）

一、实验目的

了解地图数据库基本原理，掌握地图绘制编程方法

二、实验内容

1、对于给定的地图数据库，将地图数据库中的中国内陆边界线要素(China.txt)及城市点要素(City.txt)读取出来，并绘制相应地图至屏幕区域上，如下图 1 所示。

图 1 实验结果示意图

2、将点的名称以注记形式显示在地图上。

三、实验思路

1、 首先要添加必要的头文件和定义 point 的结构体，在 View 类中添加数据成员 pts 和 count 并初始化。

2、 在 View 中定义读取 China 和 City 文件函数，利用 ifstream 类进行文件

读取

3、 在文件打开菜单添加消息处理函数 OnOpenFile，添加读入 China 和 City。文件的函数名以及图形无效函数 Invalidate()，设置 OnLButtonDown 函数。

4、 地图绘制，在 OnDraw 函数中循环绘制通过 CDC 类进行点，线的绘制。

四、关键代码

1、定义 MyPoint 结构体

```
#include <string> //C++字符串类
using namespace std;//命名空间，string类在std命名空间下
struct MyPoint
{
int x;
int y;
int rank;
string name;
int id;
};
```

2、在头文件中添加新的变量

```
MyPoint *pChinaPts;//存储China数据
MyPoint *pCityPts;//存储City数据
int chinaPtsCount;//China的点数
int cityPtsCount;//City 的点数
```

3、在构造函数中初始化

```
C地信1班072123932View::C地信1班072123932View() noexcept
{
    // TODO: 在此处添加构造代码
    pChinaPts = NULL;
    chinaPtsCount = 0;
    pCityPts = NULL;
    cityPtsCount = 0;
}
```

4、添加头文件#include<fstream>

并在类中自定义两个函数，分别用于 China 和 City 文件的读取，并将结果保存至 pChinaPts 及 pCityPts 数组中

```
int  C地信1班072123932View::readChinaFromFile()
```

```
{
    ifstream ifle("D:\\CUMT\\GIS课程\\计算机制图\\code\\1_map\\China.txt");
    ifle >> chinaPtsCount;
    if (pChinaPts) delete[] pChinaPts;
    pChinaPts = new MyPoint[chinaPtsCount];
    string temp;
    ifle >> temp >> temp >> temp >> temp >> temp;

    for (int i = 0; i < chinaPtsCount; i++)
    {
        ifle >> pChinaPts[i].id >> temp >> pChinaPts[i].x >> temp >> pChinaPts[i].y;
    }
    return 0;
}

int C地信1班072123932View::readCityFromFile()
{
    ifstream ifile("D:\\CUMT\\GIS课程\\计算机制图\\code\\1_map\\City.txt");
    ifile >> cityPtsCount;
    if (pCityPts) delete[] pCityPts;
    pCityPts = new MyPoint[cityPtsCount];
    string temp;
    ifile >> temp >> temp >> temp >> temp >> temp >> temp >> temp >> temp >> temp;
    for (int i = 0; i < cityPtsCount; i++)
    {
        ifile >> pCityPts[i].id >> temp >> pCityPts[i].name >> temp >>
pCityPts[i].rank >> temp >> pCityPts[i].x >> temp >> pCityPts[i].y;
    }
    return 0;
   }
```

5、添加文件打开函数

```
void C地信1班072123932View::OnFileOpen()
{
    // TODO: 在此添加命令处理程序代码
    readChinaFromFile();
    readCityFromFile();
    Invalidate();
}
```

6、最后在 OnDraw 函数下进行图形绘制

```
void C地信1班072123932View::OnDraw(CDC *pDC)
{
    C地信1班072123932Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
```

```
    if (!pDoc)
        return;
    CPen pen(PS_SOLID, 1, RGB(255, 0, 0)); // 创建红色实线画笔
    CBrush brush(RGB(255, 255, 255)); // 创建无填充色画刷
    pDC->SelectObject(&pen); // 选入画笔
    pDC->SelectObject(&brush); // 选入画刷
    for (int i = 0; i < chinaPtsCount - 1; i++)//绘制china
    {
        pDC->MoveTo(pChinaPts[i].x, 600 - pChinaPts[i].y);
        pDC->LineTo(pChinaPts[i + 1].x, 600 - pChinaPts[i + 1].y);
    }
    for (int i = 0; i < cityPtsCount; i++)//绘制city
    {
        pDC->Ellipse(pCityPts[i].x - 2, 600 - pCityPts[i].y - 2, pCityPts[i].x + 2,
600 - pCityPts[i].y + 2);
        ////////////////////////////////////////////////////////////////////
        //////字符串类型转换，将string转为CString
        ////////////////////////////////////////////////////////////////////
#ifdef _UNICODE //如果是unicode工程
        USES_CONVERSION; CString tempStr(pCityPts[i].name.c_str());
#else //如果是多字节工程
        CString ans;
        tempStr.Format("%s", pCityPts[i].name.c_str());
#endif // _UNICODE
        ////////////////////////////////////////////////////////////////////
        pDC->TextOut(pCityPts[i].x + 3, 600 - pCityPts[i].y + 3, tempStr); //输出注记
    }
}
```

五、实验结果

编辑好代码进行运行，选择文件进行打开，结果如图 2 所示。

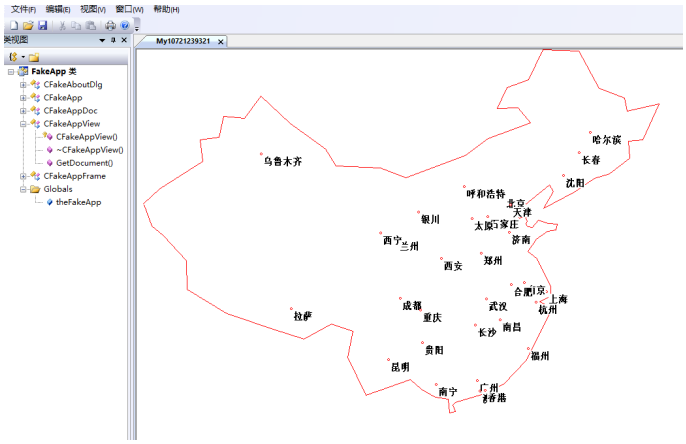


图 2 绘制结果图

六、实验体会

通过第一次实验，让我利用微软提供的 MFC 模板类，用 C++实现计算机地图制图的绘制，利用文件读写，将 City，China 两个文件中的数据读取到数组中，利用循环和 CDC 提供的绘图工具，成功在计算机的屏幕上实现地图的绘制。

中国矿业大学
CHINA UNIVERSITY OF MINING AND TECHNOLOGY

《计算机地图制图》实验报告

姓名：马骁

学号：07212393

班级：地信 21-1 班

中国矿业大学

2023 年 6 月

实验二　中点画线算法直线绘制

一、实验目的

1、 了解中点画线算法的基本原理；

2、 学会使用 VC++实现直线生成算法编程.

二、实验内容

对于任意输入的两点（斜率任意、起始点顺序任意），利用中点画线算法实现的直线生成与绘制，要求能正确绘制以下 6 条直线。

直线	x0	y0	x1	y1
1	200	500	800	500
2	500	200	500	800
3	800	650	200	350
4	650	800	350	200
5	200	650	800	350
6	350	800	650	200

三、实验思路

原理推导：

首先为了简化问题，由于从起点到终点和终点到起点画出的线是一样的，通过交换起始点，我们可以将 k 的 8 个取值简化为 4 个。

然后是 4 个方向的原理推导，四个方向代表了四个不同直线斜率的范围，算法推导如图 1 所示。

任意斜率中点画线算法

1. $0 \le k \le 1$　　$F(x,y) = ax + by + c = 0$

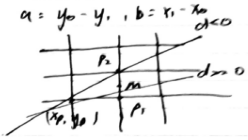
设当前点坐标 (x_p, y_p)

则中点坐标为 $m(x_p+1, y_p+0.5)$

$d = F(m) = F(x_p+1, y_p+0.5)$

$= a(x_p+1) + b(y_p+0.5) + c$

$= F(x_p, y_p) + a + 0.5b$



若 $d<0$　则取右上点 P_2.

$d_{i+1} = F(x_p+2, y_p+1.5)$　　当前点为 P_2，指导 P_2 的下一个像素

$= a(x_p+2) + b(y_p+1.5) + c$

$= d_i + a + b$

若 $d>0$ 则取正右点 P_1

$d_{i+1} = F(x_p+2, y_p+0.5)$　　当前点为 P_1，指导 P_1 的下一个像素

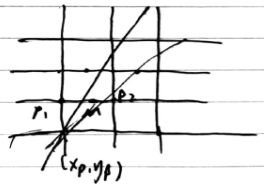
$= a(x_p+2) + b(y_p+0.5) + c$

$= d_i + a$

$D_i < 0$ 取右上　$D_{i+1} = D_i + 2a + 2b$

$D_i > 0$ 取正右　$D_{i+1} = D_i + 2a$

$D_0 = 2d_0 = 2a + b$



2. $k > 1$

设当前点坐标 (x_p, y_p)

中点坐标 $m(x_p+0.5, y_p+1)$

$d = F(m) = F(x_p+0.5, y_p+1)$

$= F(x_p, y_p) + 0.5a + b$

若 $d<0$ 则取垂正上方 P_1

$d_{i+1} = F(x_p+0.5, y_p+2)$

$= a(x_p+0.5) + b(y_p+2) + c$

$= a$ ~~$F(x_p, y_p)$~~ $d_i + b$.

若 $d>0$　则取右上方 P_2

$d_{i+1} = F(x_p+1.5, y_p+2)$

$= a(x_p+1.5) + b(y_p+2) + c$

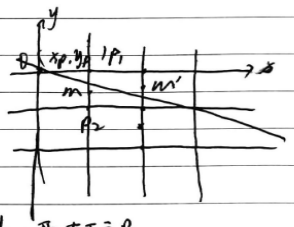
$= d_i + a + b$.

$D_i < 0$ 取正上方

$D_{i+1} = D_i + 2b$.

$D_i > 0$ 取右上.　$D_{i+1} = D_i + 2a + 2b$

$D_0 = 2d_0 = a + 2b$



$-1 \le k < 0$

设当前点 (x_p, y_p)

则中点坐标 $(x_p+1, y_p-0.5)$

$d = F(m) = F(x_p+1, y_p-0.5)$

$= a(x_p+1) + b(y_p-0.5) + c$

$= F(x_p, y_p) + a - 0.5b$.

若 $d<0$ 则取正右方 P_1

$d_{i+1} = F(x_p+2, y_p-0.5)$

$= a(x_p+2) + b(y_p-0.5) + c$

$=$ ~~$F(x_p, y_p) +$~~

$d_i + a$.

若 $d>0$ 取右下方 P_2

$d_{i+1} = F(x_p+2, y_p-1.5)$

$= a(x_p+2) + b(y_p-1.5) + c$

$= d_i + a - b$

$D_i < 0$ 取正右　$D_{i+1} =$ ~~D_i~~ $+ 2a$

$D_i > 0$ 取右下方　$D_{i+1} = D_i + 2a - 2b$

$D_0 = 2d_0 = 2a - b$

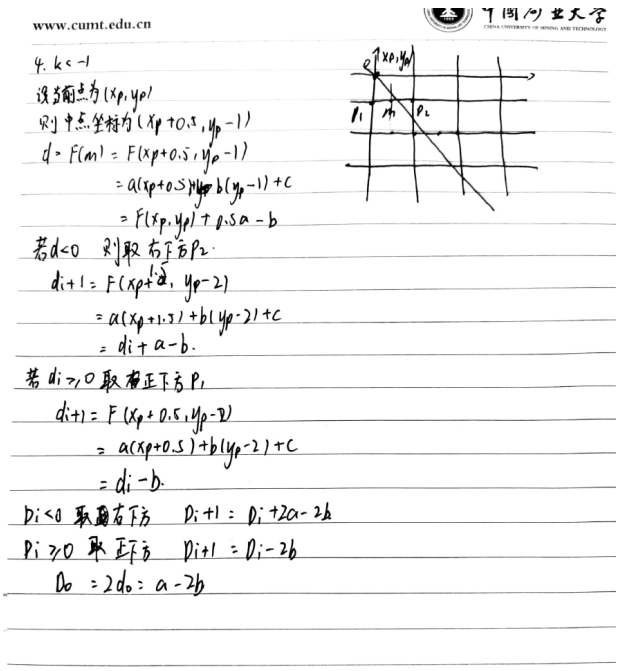


图 1　中点直线绘制算法推导

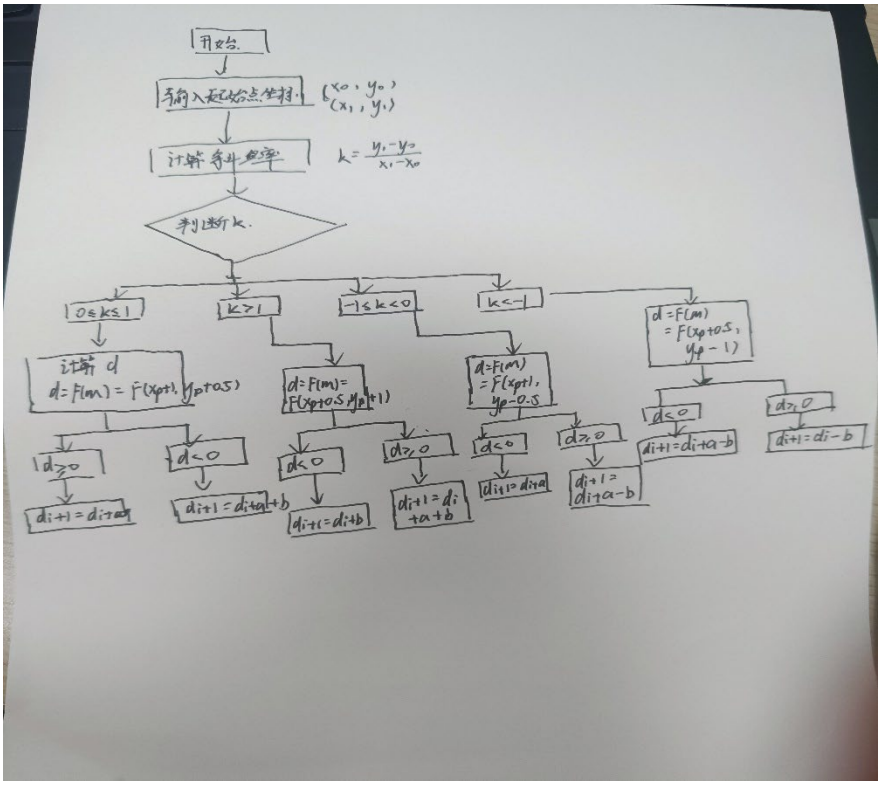


图 2 终点直线绘制算法流程图

四、关键代码

中点直线算法函数 MidPointLine：

```
void MidPointLine(int x0, int y0, int x1, int y1, int color, CDC* pDC)
{
    int a, b, d1, d2, d, x, y;
    if (x0 > x1 && y0 > y1) //交换函数，判断起始点坐标大小
    {
        int tmp;
        tmp = x0;
        x0 = x1;
        x1 = tmp;
        int tmp2;
        tmp2 = y0;
        y0 = y1;
        y1 = tmp2;
    }
    float k = (float)(y1 - y0) / (x1 - x0);
    a = y0 - y1;
    b = x1 - x0;
    if (k>=0&&k<=1)
    {
        d1 = a + a;
        d2 = a + b + a + b;
        d = a + a+b;
        x = x0;
        y = y0;
        while (x <= x1)
        {
            pDC->SetPixel(x, y, color);
            if(d < 0)
            {
                x++;
                y++;
                d += d2;
            }
            else {
                x++;
                d += d1;
            }
        }
    }
    else if (x0 == x1)
    {
        y = y0;
        while (y <= y1)
```

```
        {
            pDC->SetPixel(x0, y, color);
            y++;
        }
    }
    else if (k<0&&k>=-1)
    {
        d1 = a + a;
        d2 = a + a - b - b;
        d = a + a - b;
        x = x0;
        y = y0;
        while (x <= x1)
        {
            pDC->SetPixel(x, y, color);
            if (d < 0)
            {
                x++;
                d += d1;
            }
            else {
                x++;
                y--;
                d += d2;
            }
        }

    }
    else if (k < -1)
    {
        d1 = -b - b;
        d2 = a + a - b - b;
        d = a - b - b;
        x = x0;
        y = y0;
        while (x <= x1)
        {
            pDC->SetPixel(x, y, color);
            if (d < 0)
            {
                x++;
                y--;
                d += d2;
            }
```

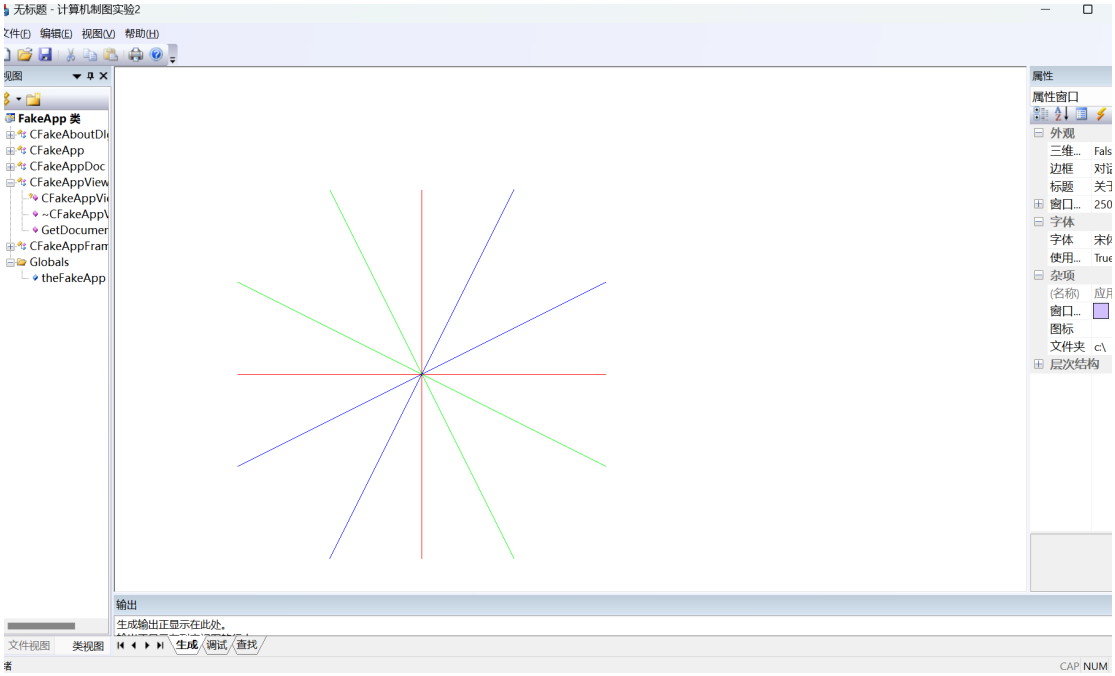
```
            else {
                y--;
                d += d1;
            }
        }
    }

    else if (k > 1)
    {
        d1 = b + b;
        d2 = a + a + b + b;
        d = a + b + b;
        x = x0;
        y = y0;
        while (x <= x1)
        {
            pDC->SetPixel(x, y, color);
            if (d < 0)
            {
                y++;
                d += d1;
            }
            else {
                x++;
                y++;
                d += d2;
            }
        }
    }
}
```

中点画线算法的调用与直线绘制：

```
void C计算机制图实验2View::OnDraw(CDC* pDC)
{
    C计算机制图实验2Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;
    MidPointLine(200, 500, 800, 500,RGB(255,0,0), pDC);
    MidPointLine(500, 200, 500, 800, RGB(255, 0, 0), pDC);
    MidPointLine(800, 650, 200, 350, RGB(0, 255, 0), pDC);
    MidPointLine(650, 800, 350, 200, RGB(0, 255, 0), pDC);
    MidPointLine(200, 650, 800, 350, RGB(0, 0, 255), pDC);
    MidPointLine(350, 800, 650, 200, RGB(0, 0, 255), pDC);
}
```

五、实验结果



六、实验体会

中点直线算法是计算机图形学的一个重要的算法，它能够高效的生成直线，通过编码前的公式推导和流程图绘制，为编码提供了思路和基础，在编写代码时需要注意判断条件等，难度适中，进一步加深了对直线绘制算法的理解。

中国矿业大学
CHINA UNIVERSITY OF MINING AND TECHNOLOGY

《计算机地图制图》实验报告

姓名：马骁

学号：07212393

班级：地信 21-1 班

中国矿业大学

2023 年 6 月

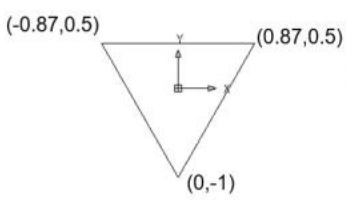
实验三：点状地物的符号化（3 学时）

一、实验目的

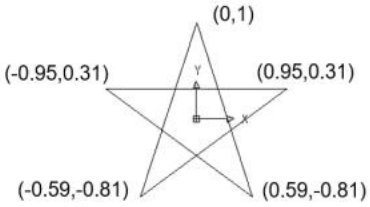
了解地图符号化基本原理，掌握点状地图符号的绘制方法

二、实验内容

1、了解地图符号化基本原理，掌握点状地图符号的绘制方法。



1



2

2、在实验二的基础上，根据地图符号库的地图符号以及 City 表的 Rank 属性，将不同城市以不同的地图符号绘制出来。

OBJECTID	Shape	NAME	Rank	xFlt	yFlt	xInt	yInt
1	长二进制数据	北京	1	116.3809433	39.92361450	563	375
2	长二进制数据	天津	2	117.2034988	39.13111877	573	362
3	长二进制数据	石家庄	2	114.4897766	38.04512787	538	344
4	长二进制数据	太原	2	112.5693512	37.87111282	514	341
5	长二进制数据	呼和浩特	2	111.6632996	40.82094193	502	390
6	长二进制数据	沈阳	2	123.4116821	41.79661560	654	407
7	长二进制数据	长春	2	125.3154297	43.89256287	678	442
8	长二进制数据	哈尔滨	2	126.6433411	45.74149323	695	473

三、实验思路

1、参考信息块法，自定义一个统一的地图符号数据结构。可利用结构体来对地理符号库进行定义，如图 1 所示。

```
};
struct TPointSym {
    int id;
    string name;
    int numOfPts;       //点数量
    float* xCoords;
    float* yCoords;
    int* tai;
};
```

图 1 自定义地图符号结构体

2、将两个地图符号以统一的数据结构进行描述，并存入地图符号库中，地图符号库如图 2 所示。

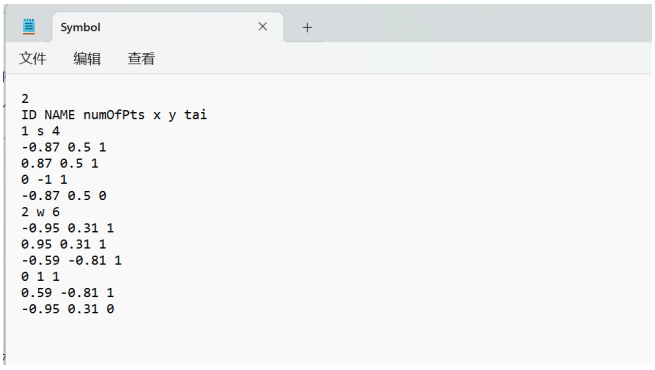


图 2 地图符号库设计

结构说明：

第一行：总符号数

第二行：文字注记，用 temp 存入。

第三行：id 符号名 符号点数（为保证闭环数量要+1）

第四行：x 坐标 y 坐标 抬落笔码

3、设计能够读写地图符号库的函数 readSymbolFromfile，识别地图符号库的数字数据，并存入数组中。

4、设计 DrawSymbol 函数，从数组中拿出数据进行符号绘制，并能够实现坐标系平移。

四、关键代码

接着实验一进行代码编写，实验一中定义的关键代码在这里不重复提及。

1、定义地图符号结构体

```
struct TPointSym {
    int id;
    string name;
    int numOfPts;      //点数量
    float* xCoords;
    float* yCoords;
    int* tai;
   };
```

2、定义成员函数和成员变量

```
// 特性
public:
    CMFCApplication1Doc* GetDocument() const;
    MyPoint* pChinaPts;
    MyPoint* pCityPts;
    int chinaPtsCount;
    int cityPtsCount;
    TPointSym* pSyms;
    int numOfSyms;
// 重写
public:
    virtual void OnDraw(CDC* pDC);  // 重写以绘制该视图
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    int readChinaFromFile();
    int readCityFromFile();
    int readSymbolFromFile();
    void DrawSymbol(double x1,double y1,TPointSym& Syms,double scale, CDC* pDC);
```

3、成员变量初始化

```
CMFCApplication1View::CMFCApplication1View() noexcept
{
    // TODO: 在此处添加构造代码
    pChinaPts = NULL;
    chinaPtsCount = 0;
    pCityPts = NULL;
    cityPtsCount = 0;
    pSyms = NULL;
    numOfSyms = 0;
   }
```

4、读取文件函数

```
int CMFCApplication1View::readSymbolFromFile()
{
    ifstream ifile("D:\\CUMT\\GIS课程\\计算机制图\\计算机地图制图\\计算机地图制图\\Symbol.txt");//将文件绑定在一个文件流上
    ifile >> numOfSyms;
    if (pSyms)
    {
        delete[] pSyms;
    }
    pSyms = new TPointSym[numOfSyms];
    string temp;
    ifile >> temp >> temp >> temp >> temp >> temp >> temp;
    for (int i = 0; i < numOfSyms; i++)
    {
        ifile >> pSyms[i].id >> pSyms[i].name >> pSyms[i].numOfPts;
        pSyms[i].xCoords = new float[pSyms[i].numOfPts];
        pSyms[i].yCoords = new float[pSyms[i].numOfPts];
        pSyms[i].tai = new int[pSyms[i].numOfPts];
        for (int j = 0; j < pSyms[i].numOfPts; j++)
        {
            ifile >> pSyms[i].xCoords[j]>> pSyms[i].yCoords[j] >> pSyms[i].tai[j];
        }
```

```
    }
    return 0;
    }
```

5、DrawSymbol 函数

```
void CMFCApplication1View::DrawSymbol(double x1, double y1, TPointSym& Syms, double scale, CDC* pDC)
{
    int lastTai = Syms.tai[0];
    for (int i = 1; i < Syms.numOfPts; i++)
    {
        if (lastTai == 1)
        {
            pDC->MoveTo(Syms.xCoords[i - 1] * scale + x1, Syms.yCoords[i - 1] * scale + y1);
            pDC->LineTo(Syms.xCoords[i] * scale + x1, Syms.yCoords[i] * scale + y1);
        }
        lastTai = Syms.tai[i];
    }
    }
```

6、在 OnFileOpen 中添加读取文件函数

```
void CMFCApplication1View::OnFileOpen()
{
    // TODO: 在此添加命令处理程序代码
    readChinaFromFile();
    readCityFromFile();
    readSymbolFromFile();
    Invalidate();

    }
```

7、在 OnDraw 中实现绘制

```
void CMFCApplication1View::OnDraw(CDC* pDC)
{
    ASSERT_VALID(pDC);
    if (!pDC)
        return;
    // TODO: 在此处为本机数据添加绘制代码

    CPen pen(PS_SOLID, 1, RGB(255, 0, 0)); // 创建红色实线画笔
    CBrush brush(RGB(255, 255, 255)); // 创建无填充色画刷
    pDC->SelectObject(&pen); // 选入画笔
```

```
pDC->SelectObject(&brush); // 选入画刷
for (int i = 0; i < chinaPtsCount; i++)
{
    if (i == 0)
    {
        pDC->MoveTo(pChinaPts[i].x, 600-pChinaPts[i].y);
    }
    else
    {
        pDC->LineTo(pChinaPts[i].x, 600-pChinaPts[i].y);
    }
}
// 绘制City
for (int i = 0; i < cityPtsCount; i++)
{
    if (pCityPts[i].rank == 1)
    {
        DrawSymbol(pCityPts[i].x - 2, 600 - pCityPts[i].y - 2, pSyms[0], 10, pDC);
    }
    else {
        DrawSymbol(pCityPts[i].x - 2, 600 - pCityPts[i].y - 2, pSyms[1], 10, pDC);
    }

    //CString tempStr(pCityPts[i].name.c_str());
    //pDC->TextOutW(pCityPts[i].x + 3, 600 - pCityPts[i].y + 3, tempStr);
}
}
```

五、实验结果

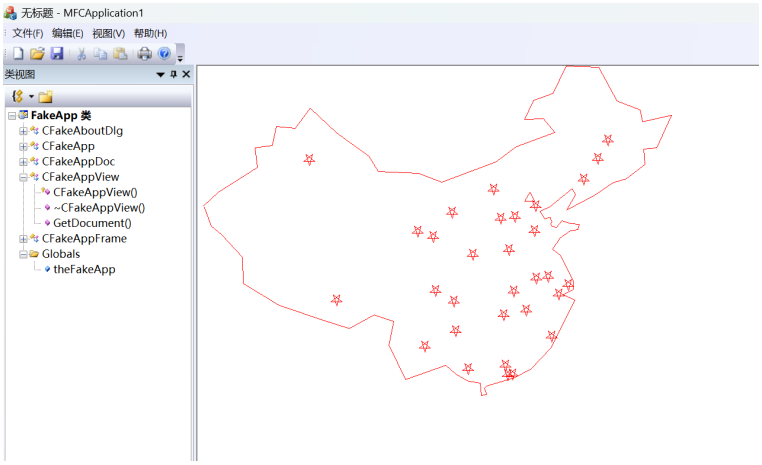


图 3 实验结果图

可以看到首都北京和其他省会的符号不一样，成功绘制！

六、实验体会

本次实验设计地图符号难度较大，我对地图符号库和文件读写 DrawSymbol 中坐标系平移的知识掌握还是不深。对 C++的指针，结构体的知识掌握还需要增强，好在最后老师的讲解下理清了实现步骤，解决了一些问题，最后能成功绘制，完成实验。