## ImagePoint.h

// ImagePoint.h: interface for the CImagePoint class.

//

//////////////////////////////////////////////////////////////////////

#if !defined(AFX\_IMAGEPOINT\_H\_\_03F4A380\_1FEC\_473F\_9862\_D8354A51BF01\_\_INCLUDED\_)

#define AFX\_IMAGEPOINT\_H\_\_03F4A380\_1FEC\_473F\_9862\_D8354A51BF01\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ]

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* << 图像点坐标 >> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ]

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ]

class CImagePoint

{

// [ \*\*\*\*\*\*\*\* ] ......................................................

// [ 成员变量 ] ......................................................

// [ \*\*\*\*\*\*\*\* ] ......................................................

public:

long m\_row; // [ 行坐标 ]

long m\_column; // [ 列坐标 ]

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ................................................

// [ 构造和析构函数 ] ................................................

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ................................................

public:

CImagePoint(); // [ 构造函数 ]

CImagePoint( long row, long column ); // [ 初始化构造函数 ]

virtual ~CImagePoint(); // [ 析构函数 ]

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ..................................................

// [ 功能接口函数 ] ..................................................

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ..................................................

public:

bool operator==( CImagePoint point ) const; // [ 等号重载 ]

bool operator!=( CImagePoint point ) const; // [ 不等号重载 ]

CImagePoint operator+( CImagePoint point ) const; // [ 加号重载 ]

CImagePoint operator-( CImagePoint point ) const; // [ 减号重载 ]

void operator+=( CImagePoint point ); // [ 加等于号重载 ]

void operator-=( CImagePoint point ); // [ 减等于号重载 ]

bool Is\_in\_rectangle( RECT rect ) const; // [ 判断点坐标是否在矩形内( 包括在矩形的四条边上 ) ]

};

#endif // !defined(AFX\_IMAGEPOINT\_H\_\_03F4A380\_1FEC\_473F\_9862\_D8354A51BF01\_\_INCLUDED\_)

## ImagePoint.cpp

// ImagePoint.cpp: implementation of the CImagePoint class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "ImagePoint.h"

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

// [ \*\*\*\*\*\*\*\* ] ..........................................................

// [ 构造函数 ] ..........................................................

// [ \*\*\*\*\*\*\*\* ] ..........................................................

CImagePoint::CImagePoint()

{

m\_row = 0; // [ 行坐标 ]

m\_column = 0; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ....................................................

// [ 初始化构造函数 ] ....................................................

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ....................................................

CImagePoint::CImagePoint( long row, long column )

{

m\_row = row; // [ 行坐标 ]

m\_column = column; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\* ] ..........................................................

// [ 析构函数 ] ..........................................................

// [ \*\*\*\*\*\*\*\* ] ..........................................................

CImagePoint::~CImagePoint()

{

}

// [ \*\*\*\*\*\*\*\* ] ..........................................................

// [ 等号重载 ] ..........................................................

// [ \*\*\*\*\*\*\*\* ] ..........................................................

bool CImagePoint::operator==( CImagePoint point ) const

{

return m\_row == point.m\_row // [ 行坐标 ]

&& m\_column == point.m\_column ; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\*\*\* ] ........................................................

// [ 不等号重载 ] ........................................................

// [ \*\*\*\*\*\*\*\*\*\* ] ........................................................

bool CImagePoint::operator!=( CImagePoint point ) const

{

return m\_row != point.m\_row // [ 行坐标 ]

|| m\_column != point.m\_column; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\* ] ..........................................................

// [ 加号重载 ] ..........................................................

// [ \*\*\*\*\*\*\*\* ] ..........................................................

CImagePoint CImagePoint::operator+( CImagePoint point ) const

{

CImagePoint temp; // [ 临时点坐标 ]

temp.m\_row = m\_row + point.m\_row; // [ 行坐标 ]

temp.m\_column = m\_column + point.m\_column; // [ 列坐标 ]

return temp; // [ 返回临时点坐标 ]

}

// [ \*\*\*\*\*\*\*\* ] ..........................................................

// [ 减号重载 ] ..........................................................

// [ \*\*\*\*\*\*\*\* ] ..........................................................

CImagePoint CImagePoint::operator-( CImagePoint point ) const

{

CImagePoint temp; // [ 临时点坐标 ]

temp.m\_row = m\_row - point.m\_row; // [ 行坐标 ]

temp.m\_column = m\_column - point.m\_column; // [ 列坐标 ]

return temp; // [ 返回临时点坐标 ]

}

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ......................................................

// [ 加等于号重载 ] ......................................................

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ......................................................

void CImagePoint::operator+=( CImagePoint point )

{

m\_row += point.m\_row; // [ 行坐标 ]

m\_column += point.m\_column; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ......................................................

// [ 减等于号重载 ] ......................................................

// [ \*\*\*\*\*\*\*\*\*\*\*\* ] ......................................................

void CImagePoint::operator-=( CImagePoint point )

{

m\_row -= point.m\_row; // [ 行坐标 ]

m\_column -= point.m\_column; // [ 列坐标 ]

}

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ....................

// [ 判断点坐标是否在矩形内( 包括在矩形的四条边上 ) ] ....................

// [ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ] ....................

bool CImagePoint::Is\_in\_rectangle( RECT rect ) const

{

return m\_row >= rect.top && m\_row <= rect.bottom // [ 行坐标 ]

&& m\_column >= rect.left && m\_column <= rect.right ; // [ 列坐标 ]

}