```
/*****************************************************************************************************************************
练习2、3、4 旋转、缩放以及仿射变换 投影变换
*****************************************************************************************************************************/

void warpaffine()
{
//读取图片
cv::Mat srcMat = cv::imread("d:\\lena.jpg");

//判断图片是否读取成功
if (srcMat.empty()) return;

/*********************************************旋转及缩放*************************************/

//初始化旋转角度以及缩放尺度
auto angle{ -10.0 };
float scale = 1;


//设置旋转中心
cv::Point2f center(srcMat.cols*0.5, srcMat.rows*0.5);

//获得变换矩阵
/***********************************************
getRotationMatrix2D () 函数模型:
getRotationMatrix2D( Point2f center, double angle, double scale );
参数介绍:
. Point2f center: Point2f类型的center，表示旋转的中心点
. double angle: 表示旋转的角度
. double scale: 图像缩放因子
***********************************************/
const cv::Mat affine_matrix_zoom = cv::getRotationMatrix2D(center, angle, scale);


//定存放结果的图像容器
cv::Mat dstMat;

//利用仿射变换函数
/***********************************************
warpAffine () 函数模型:
warpAffine( InputArray src, OutputArray dst,
                              InputArray M, Size dsize,
                              int flags = INTER_LINEAR,
                              int borderMode = BORDER_CONSTANT,
                              const Scalar& borderValue = Scalar());
参数介绍:
. InputArray src: 输入图像，Mat类对象即可
. OutputArray dst: 输出图像，需要和原图片有一样的尺寸和类型
. InputArray M: 2*3的变换矩阵
. Size dsize: Size类型的dsize表示输出图像的尺寸
. int flags: int类型的flag，插值方法的标识符
. int borderMode:边界像素模式
. const Scalar& borderValue:恒定边界下取的值
***********************************************/
cv::warpAffine(srcMat, dstMat, affine_matrix_zoom, srcMat.size());



/*********************************************仿射变换部分*************************************/


Mat affine_Mat;

//变换前3点坐标
const cv::Point2f src_pt[] = {
  cv::Point2f(200,200),
  cv::Point2f(250,200),
  cv::Point2f(200,100),
};

//变换后3点坐标
const cv::Point2f warp_pt[] = {
  cv::Point2f(300,100),
  cv::Point2f(300,50),
  cv::Point2f(200,100),
};

//计算仿射变换后的矩阵
//获得变换矩阵
/***********************************************
getAffineTransform () 函数模型:
getAffineTransform( const Point2f src[], const Point2f dst[] );
参数介绍:
. const Point2f src[]: 原始图像的点
. const Point2f dst[]: 目标图像的点
***********************************************/
const cv::Mat affine_matrix = cv::getAffineTransform(src_pt, warp_pt);

cv::warpAffine(srcMat, affine_Mat, affine_matrix, srcMat.size());


/*********************************************投影变换部分*************************************/
```

```
Mat perspective_Mat;


//变换前4点坐标
cv::Point2f pts1[] = {
cv::Point2f(150,150),
cv::Point2f(150,300),
cv::Point2f(350,300),
cv::Point2f(350,150),

};

//变换后4点坐标
cv::Point2f pts2[] = {
cv::Point2f(200,150),
cv::Point2f(200,300),
cv::Point2f(340,270),
cv::Point2f(340,180),
};

//投影变换矩阵生成(3*3)
cv::Mat perspective_matrix = cv::getPerspectiveTransform(pts1, pts2);

//投影变换
/*********************************************
warpPerspective()  函数模型:
warpPerspective( InputArray src, OutputArray dst,
                                    InputArray M, Size dsize,
                                    int flags = INTER_LINEAR,
                                    int borderMode = BORDER_CONSTANT,
                                    const Scalar& borderValue = Scalar());
参数介绍:
. InputArray src: 输入图像，Mat类对象即可
. OutputArray dst: 输出图像，需要和原图片有一样的尺寸和类型
. InputArray M: 透视的变换矩阵
. Size dsize: Size类型的dsize表示输出图像的尺寸
. int flags: int类型的flag，插值方法的标识符
. int borderMode:边界像素模式
. const Scalar& borderValue:恒定边界下取的值
*********************************************/
cv::warpPerspective(srcMat, perspective_Mat, perspective_matrix, srcMat.size());


//显示所有结果
imshow("srcMat", srcMat);
imshow("dstMat", dstMat);
imshow("affine_Mat", affine_Mat);
imshow("perspective_Mat", perspective_Mat);

waitKey(0);
}
```