**实验报告**

专业：计算机科学与技术

姓名：陈哲恺

学号：3210102035

日期：2022/11/2

课程名称： 图像信息处理 指导老师： 宋明黎 成绩：

实验名称： 图像转换

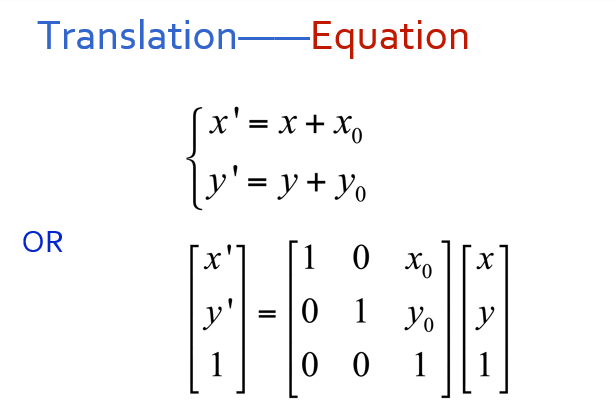
# 一、实验目的和要求

对24bit-bmp图像进行位移，旋转，缩放，剪切和镜像操作的实现。

# 二、实验内容和原理

**位移**

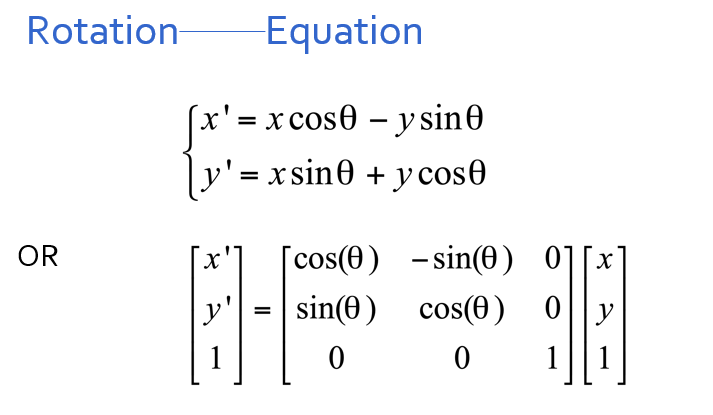
以图片中心点建坐标系，使用如图所示变换，所得即为位移后结果。



即将所有原坐标，分别进行在x轴和y轴上的坐标移动。

**旋转**

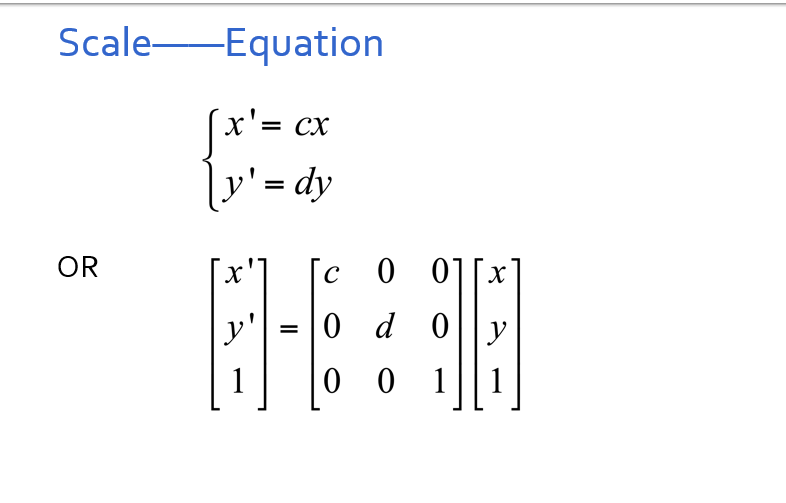
以图片中心建坐标系，进行如图的坐标变换，所得即转换后结果。



即将所有原坐标，做极坐标的角度变换。

**缩放**

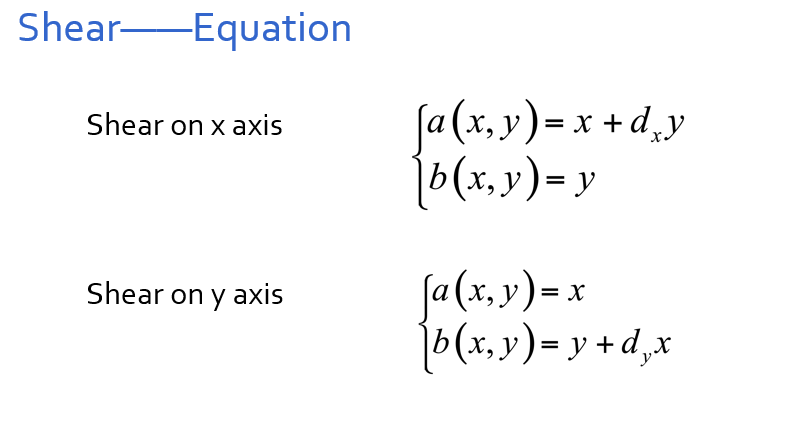
以图片中心建坐标系，进行如图所示的坐标变换，所得即转换后结果。



即将所有坐标按比例移动所在位置，要注意放大缩小时会导致部分坐标空缺或重复，需进行处理。

**剪切**

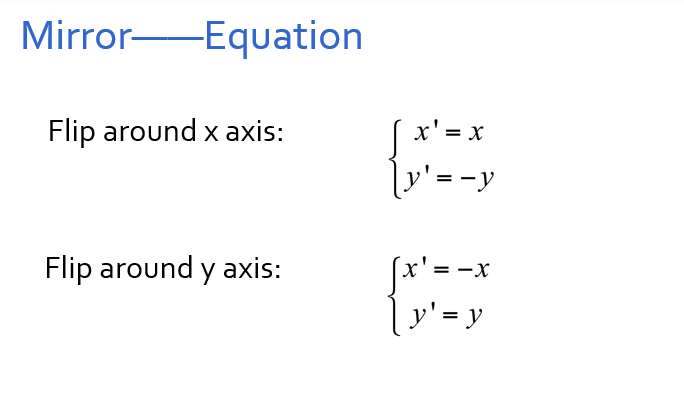
以图片中心建坐标系，进行如图所示的坐标变换，所得即转换后结果。



即将所有坐标按照行数或列数进行偏移，实现倾斜效果。

**镜像**

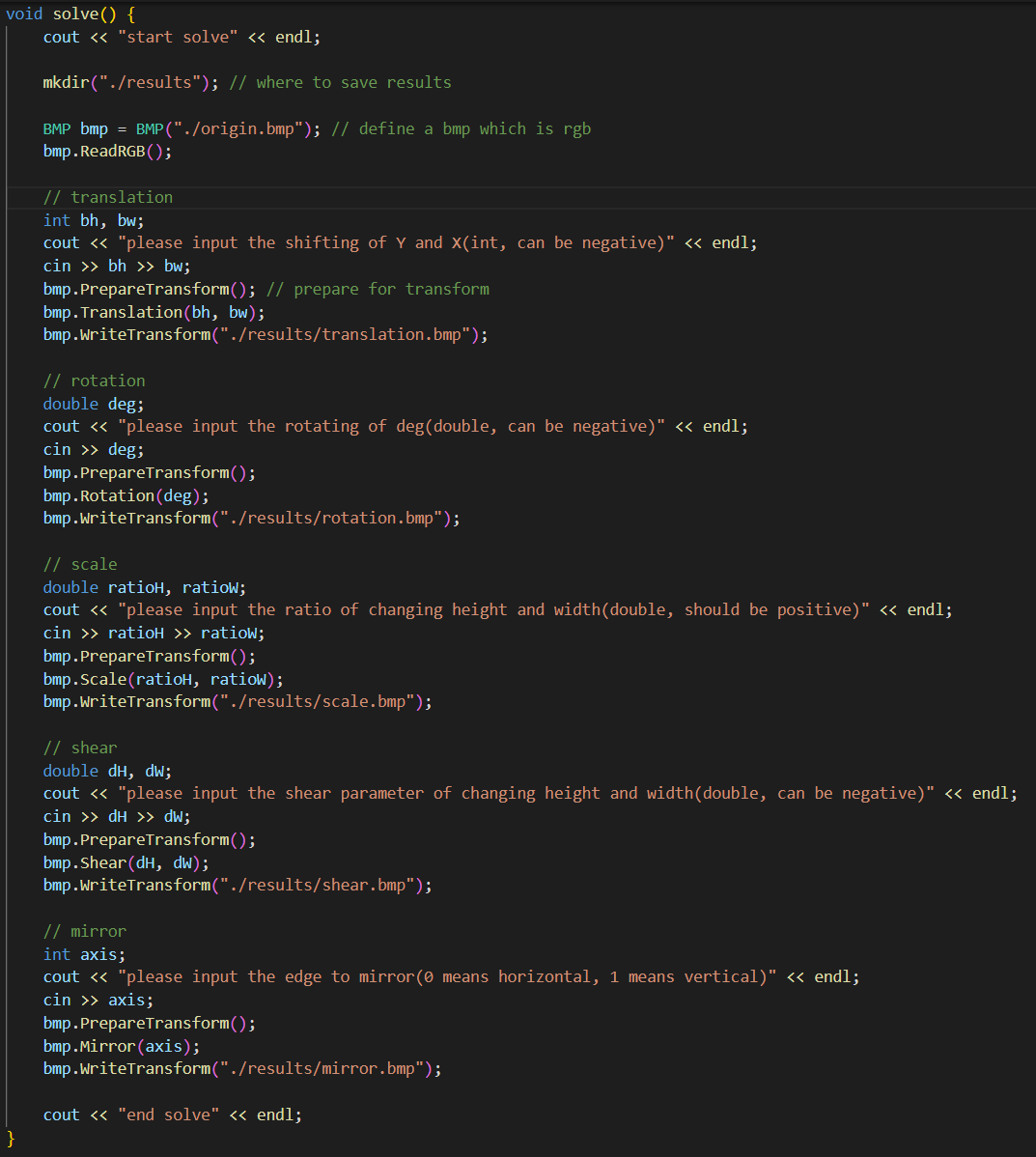
以图片中心建坐标系，进行如图所示的坐标变换，所得即转换后结果。



即将X轴坐标互换或将Y轴坐标互换。

# 三、实验步骤与分析

**全部过程展示**



**详细过程展示**

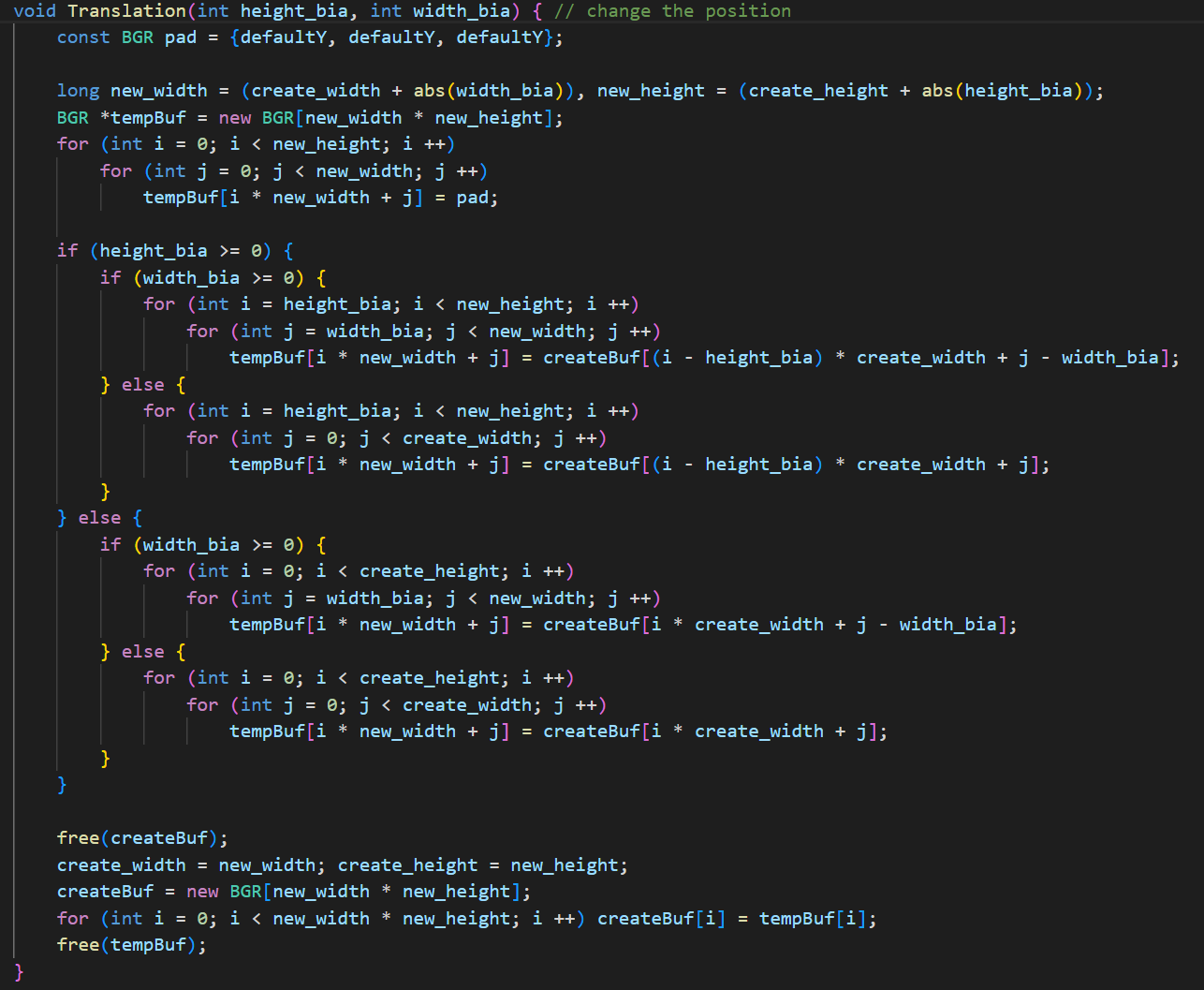
1. 读取bmp

与之前实验一致。

1. 转化产生YUV格式的值

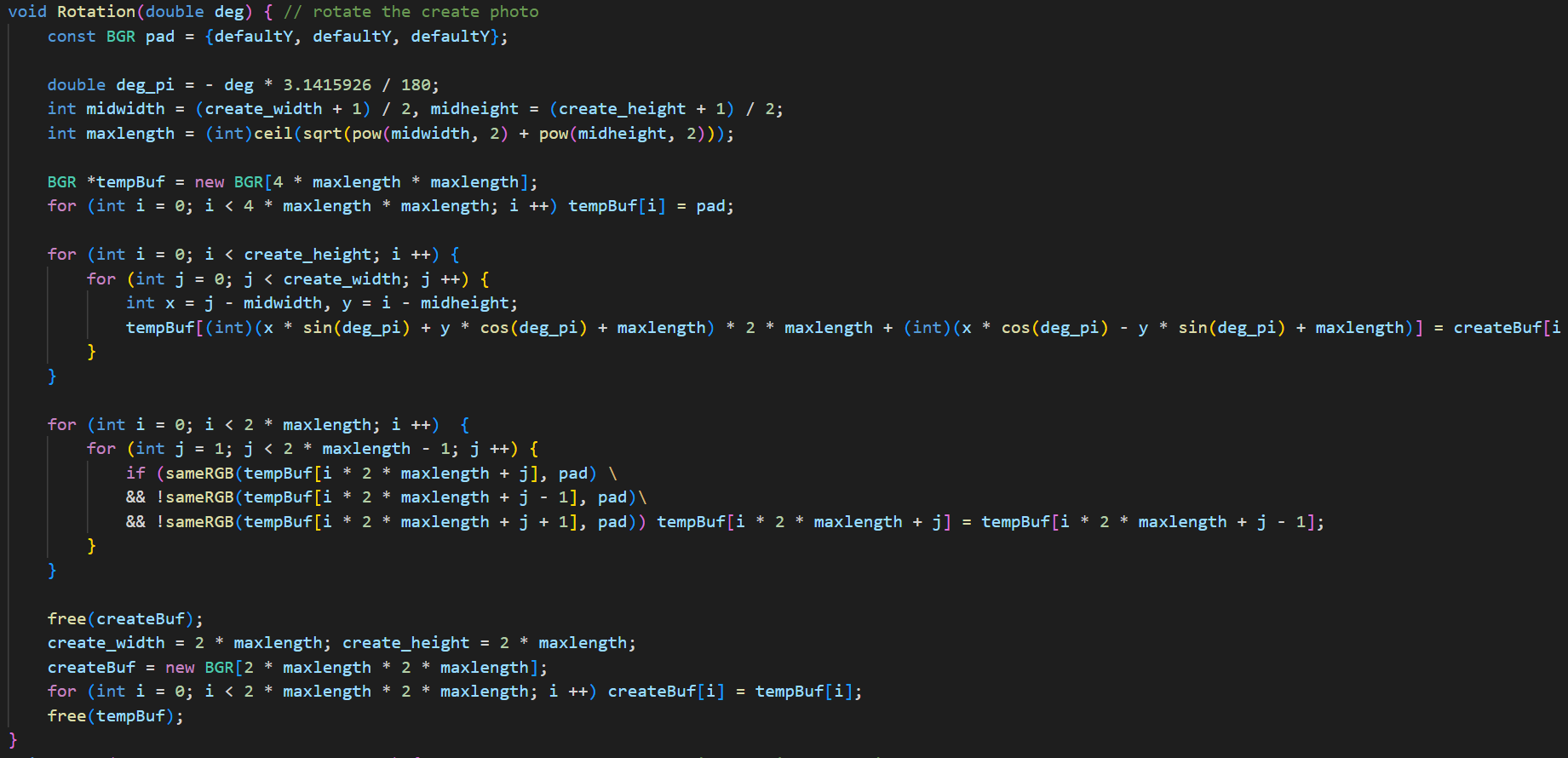
与之前实验一致。

1. 分别对图片进行位移、旋转、缩放、剪切、镜像操作
   1. 位移



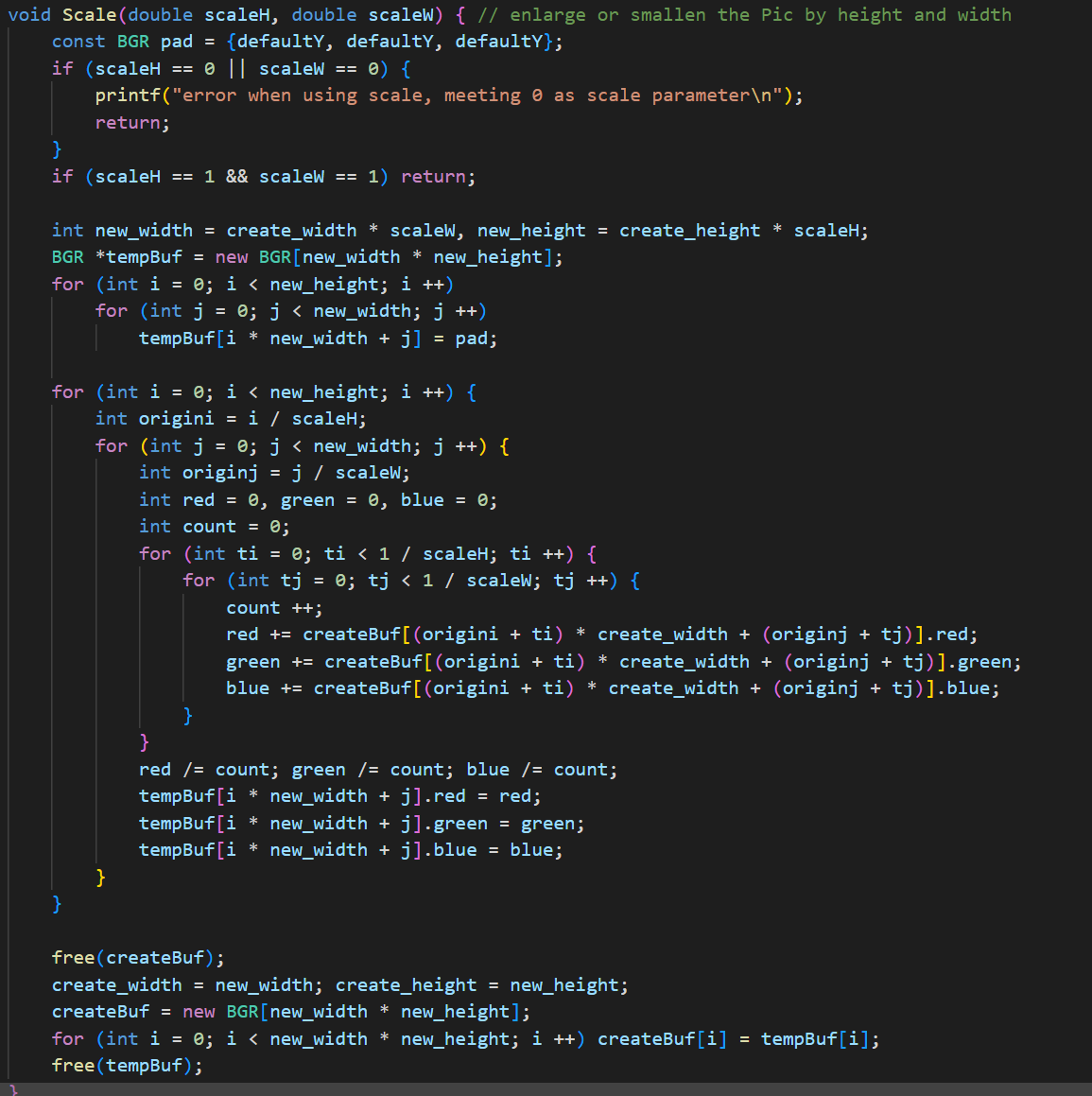
判断输入的值的正负，Y轴上上移为正，X轴上右移为正，首先进行图像大小的扩大，再根据输入值在对应位置填充原图像。

* 1. 旋转



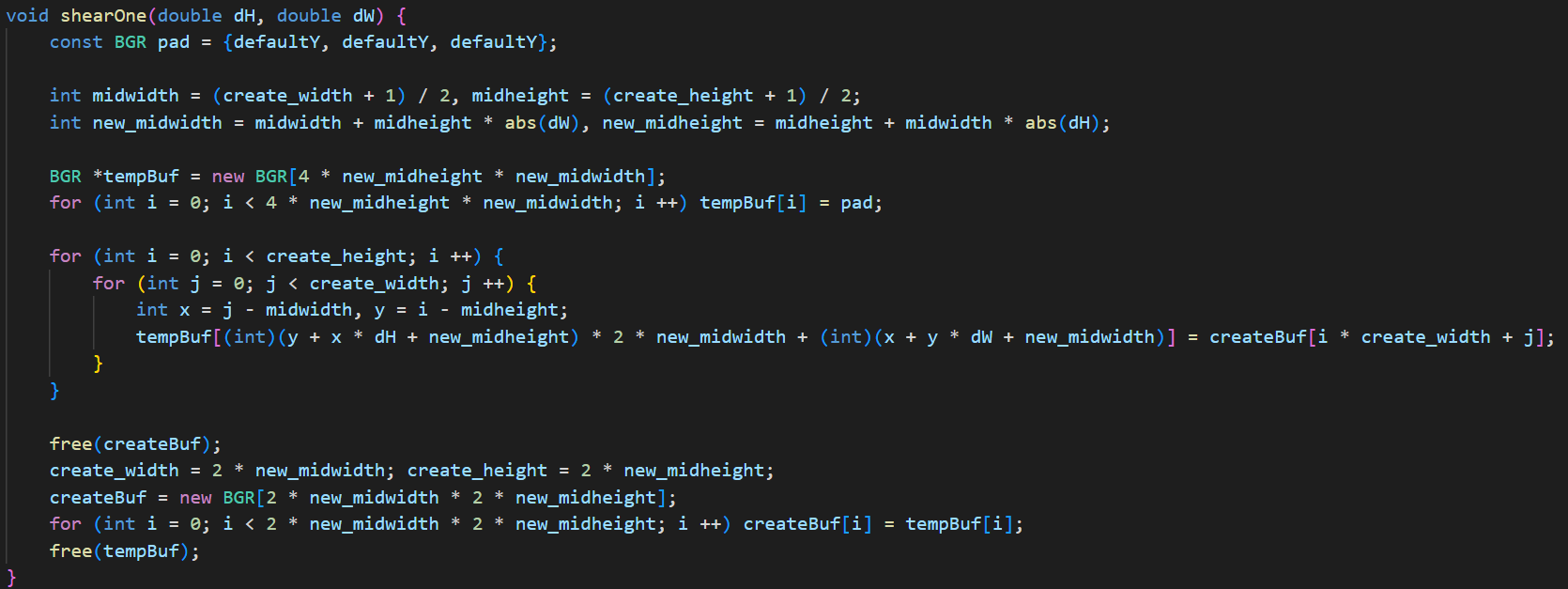
首先进行图像大小的扩大，再将角度值转换为弧度值，然后根据公式将所有原图像坐标映射到新的图像上，最后对处于图像中但是由于旋转丢失的像素点使用前一位的RGB值进行填充。

* 1. 缩放



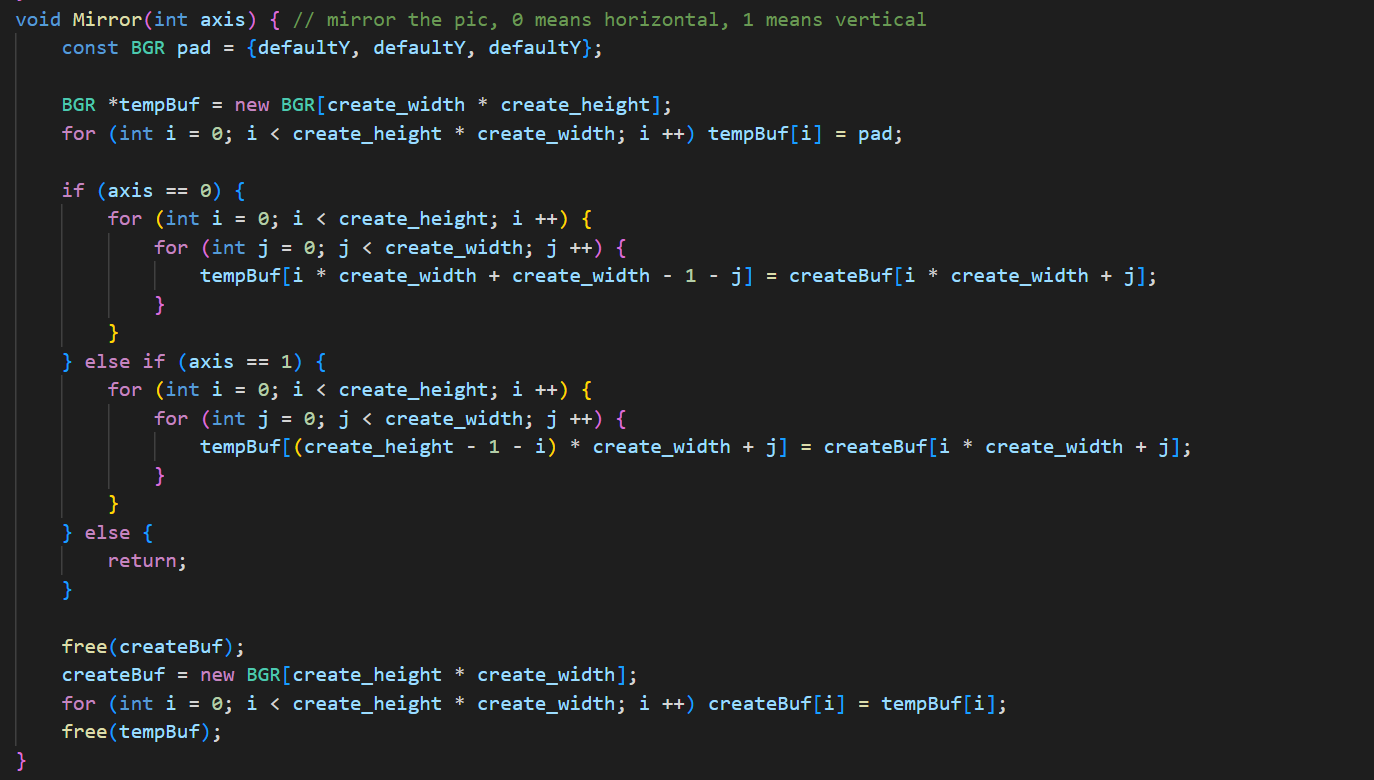
首先根据缩放大小创建新的图像大小，然后将原图像的点映射到新图像，如果新图像在该轴上进行了扩充，则使用该轴前一位置的值进行填充；如果新图像在该轴上进行了缩小，则使用可映射到该位置的所有原值的均值进行填充。

* 1. 剪切



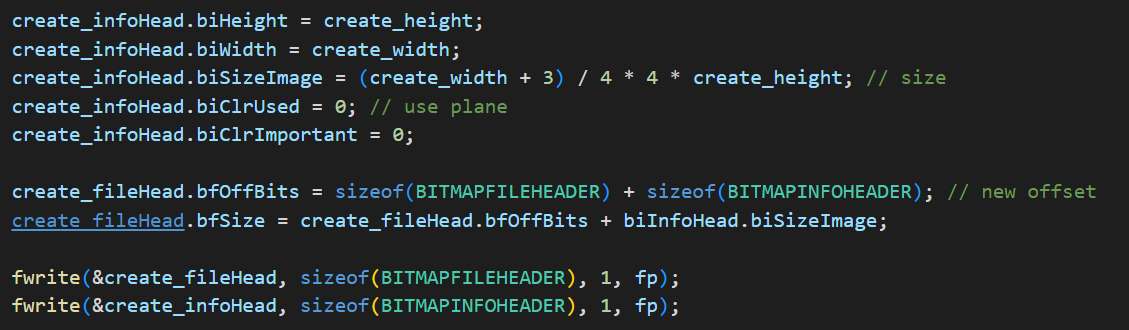
首先根据需要位移的量进行新图像的创建，然后根据X轴和Y轴的依次偏移量计算出原图像在新图像上的对应点并进行赋值。

* 1. 镜像



根据输入的值（0代表Y轴，1代表X轴），进行图片坐标的翻转。

1. 输出图片



需根据转换后图片的长宽等信息进行头文件的更改，其余与之前实验一致。

# 四、实验环境及运行方法

如果希望使用default.bat脚本（windows）进行自动编译运行以及展示成果，需确保环境下有g++编译器，以及python环境。此时，可直接使用在本文件夹下使用./default.bat进行运行。

如果希望自己进行编译等，请确保环境下至少有一个C/C++编译器，在本文件夹下使用命令行输入

{C编译器} [-Ofast] code.cpp -o program

即可编译，然后命令行输入./program进行运行测试，结果会保存在results文件夹中，也可以使用指令

python show.py

直接展示所有结果。

如果需要更换源bmp文件，直接更改本文件夹下origin.bmp为所需文件即可，具体使用说明可参考README.md和default.bat脚本

PS：本次程序无需传入参数，但需要根据相关提示输入信息从而实现所需效果

# 五、实验结果展示

./default

**参数参考1**

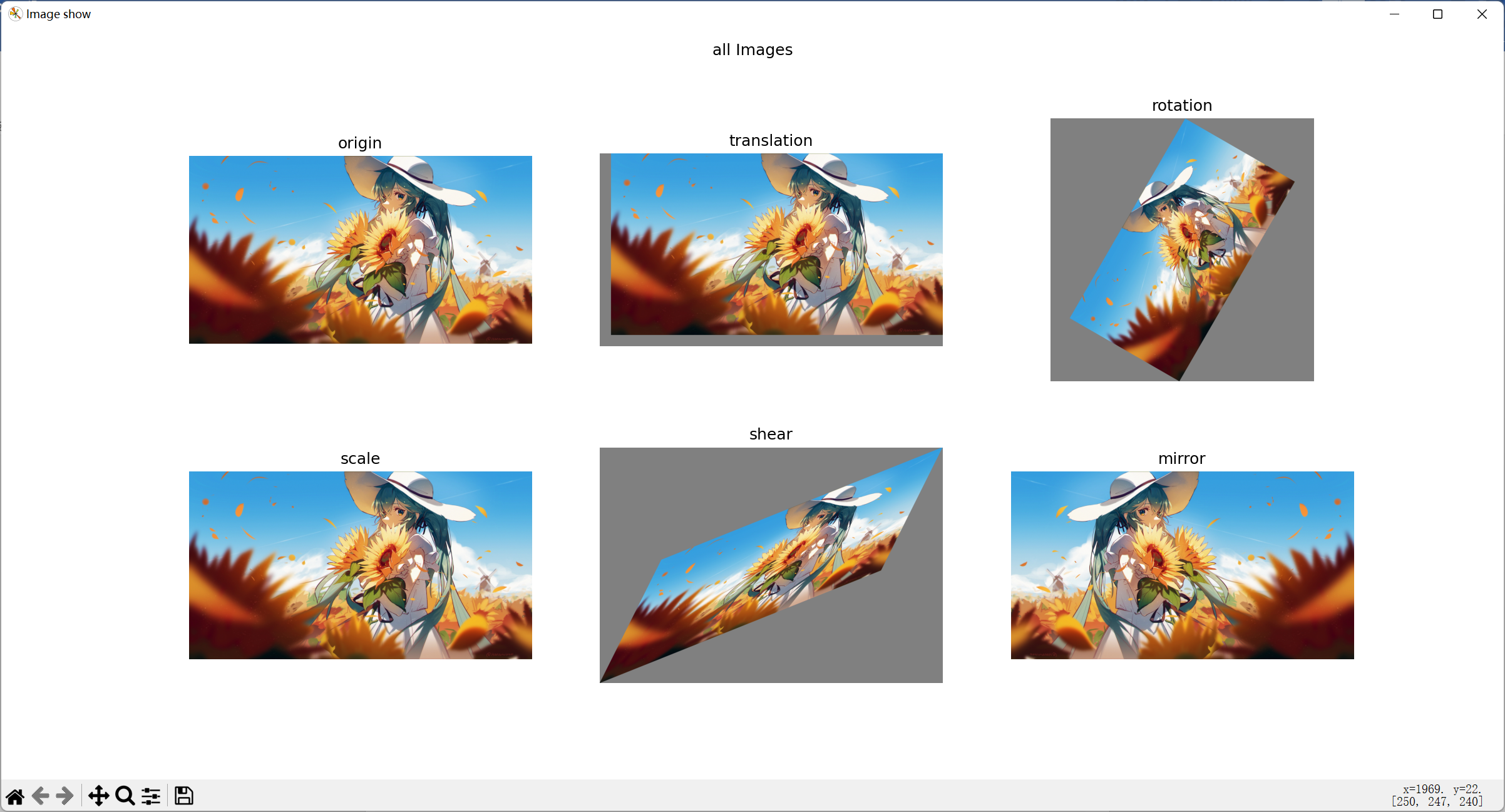
**100 100**

**300**

**2 2**

**0.5 0.5**

**0**



**参数参考2**

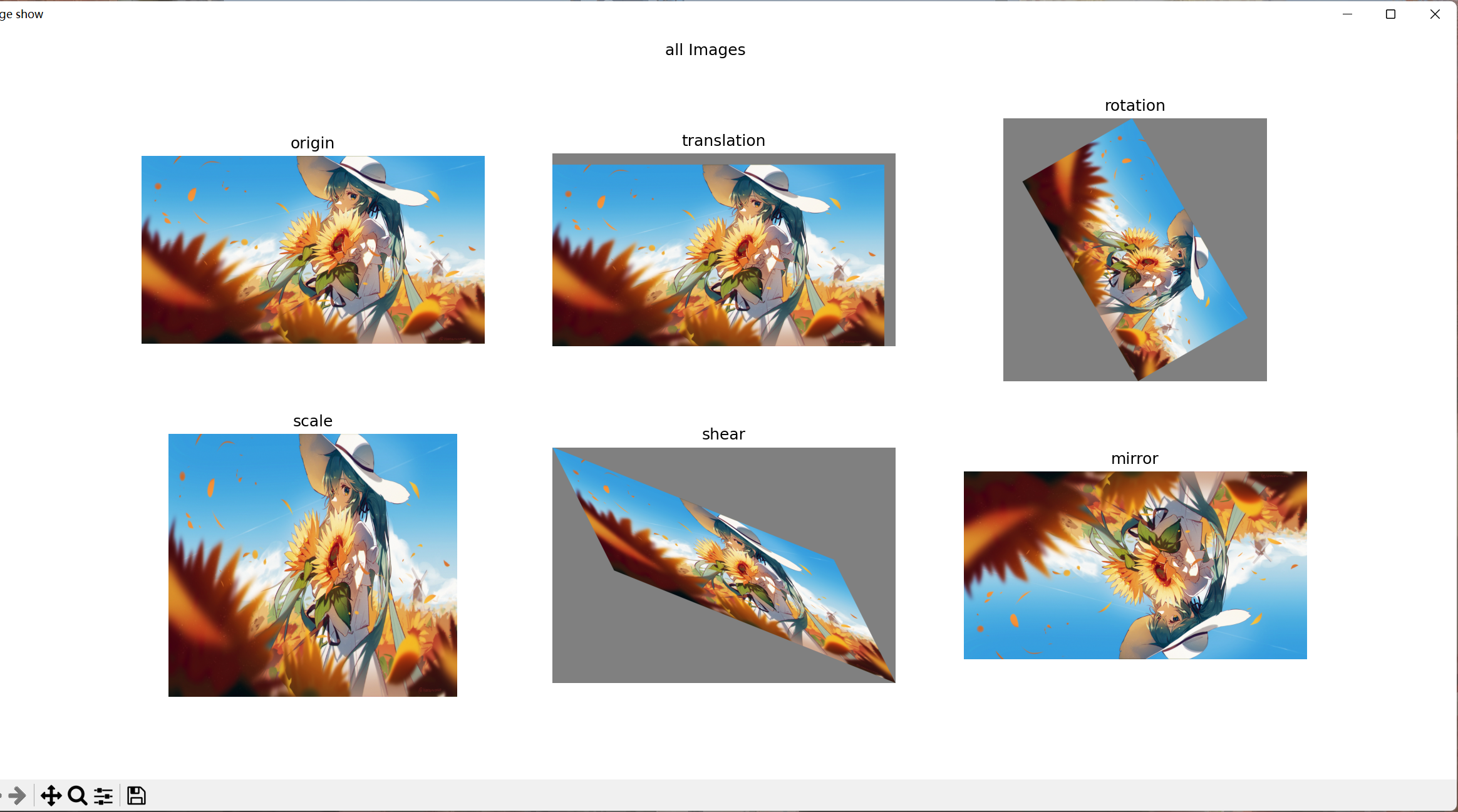
**-100 -100**

**-300**

**0.5 0.3**

**-0.5 -0.5**

**1**



其中各图片分别为，

原始图片、位移图片、旋转图片、

缩放图片、剪切图片、镜像图片

# 六、心得体会

通过这次实验，了解了对图像进行操作的原理并进行了实现，前几次实验出现的BUG在这次实验中被作为效果来实现，还真是有一种别样的感觉呢。在整个实验中学到了不少，本来有些图像操作我一直在用，但一直很好奇它的原理，比如shear，通过这次实验也彻底了解了。在这么多次实验过后，感觉自己的bmp类已经很完善了呢（不是。