



# 图像信息处理实验报告

**Digital Image Processing (Experiments)**

|  |  |
| --- | --- |
| 姓 名 | **王祚滨** |
| 指导老师 | **宋明黎** |
| 学 号 | **3180104933** |
| 专业班级 | **信息安全1801** |

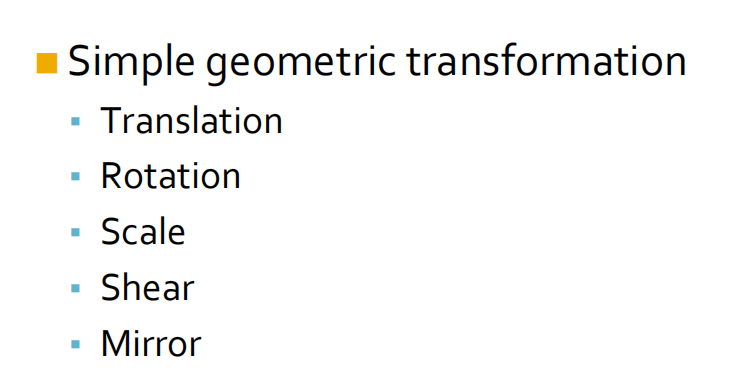
**二〇一九年**

**秋冬学期**

## 实验四

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 实验四 | **指导老师：** | 宋明黎 | **成绩：** |  |

### 一、实验目的和要求（必填）



通过实验四完成几何变换，并在旋转和缩放中体验插值。

要求:禁止作弊。

### 二、实验内容和原理（必填）

实验内容：

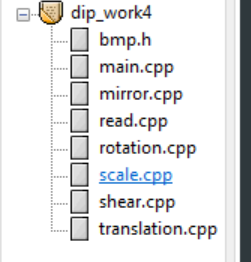
对位图文件进行几何变换，包括平移、旋转、缩放、错切、镜像

原理：

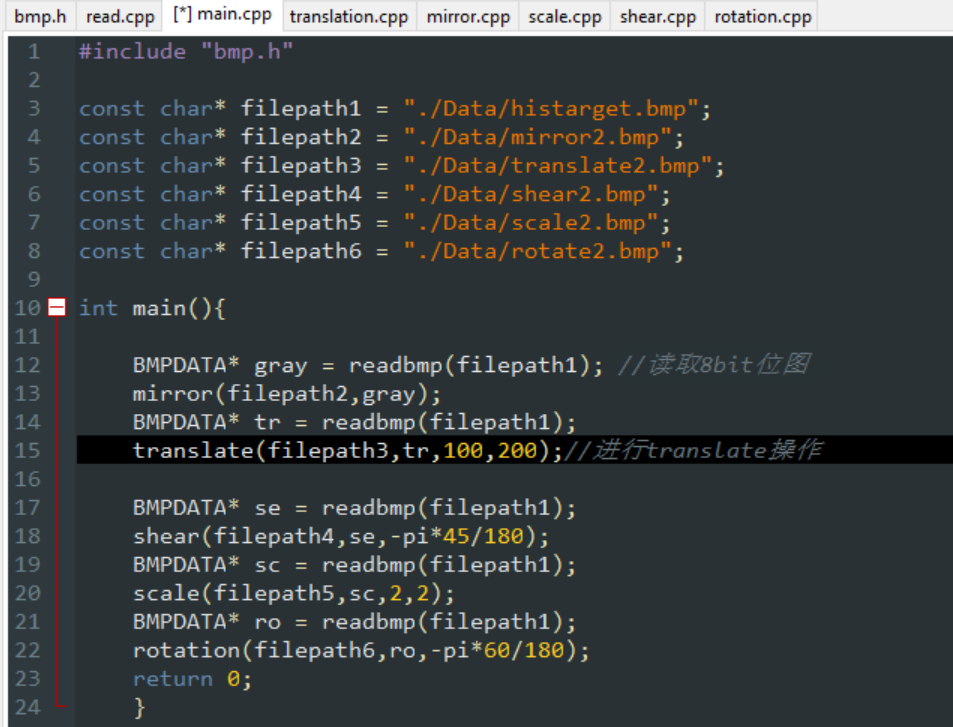
利用C语言中文件读写函数及指针，结合BMP图结构对每个像素点的坐标进行相应形式的转换，从而实现几何变换操作。

### 三、源代码与分析

本次project涉及到的函数较多，共由1个头文件与7个.cpp文件组成，与以往相似，每个.cpp文件负责相应部分的转换。



其中，main.cpp函数如下：



translate函数对应四个参数分别对应写入文件位置，将要处理的bmp图结构，x方向平移长度，y方向平移长度。

shear函数对应三个参数分别对应写入文件位置、将要处理的bmp图结构、错切角度。

mirror函数对应两个参数分别对应写入文件位置、将要处理的bmp图结构。

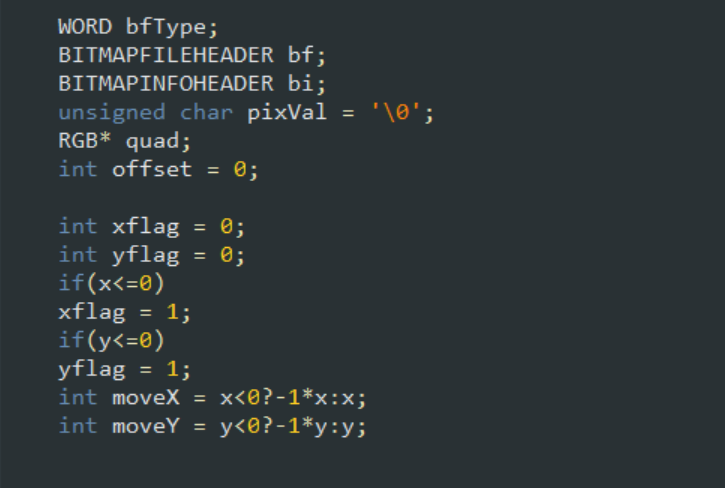
scale函数对应四个函数分别对应写入文件位置、将要处理的bmp图结构、x方向缩放倍数、y方向缩放倍数。

rotation函数对应三个函数分别对应写入文件位置、将要处理的bmp图结构、旋转角度。

总体来看，main函数较为简洁，函数分布逻辑较为清晰。

接下来结合各个函数部分源代码解析对bmp图结构作出的操作。

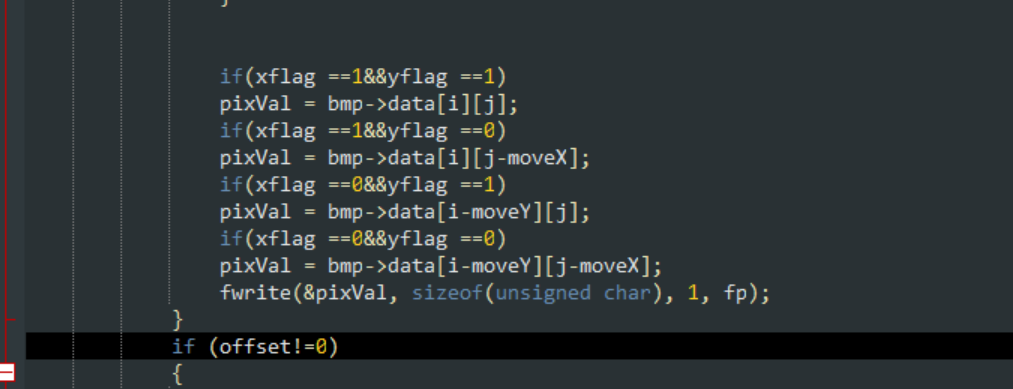
Translation.cpp



首先对传入参数x,y进行正负判定，以对应进行不同转换，



在遍历时进行操作，将不在平移范围内的区域设置为255，即设置成白色



在平移范围内的区域则关于x,y正负进行相关换算，

最后得到相对应的平移后图片进行输出。

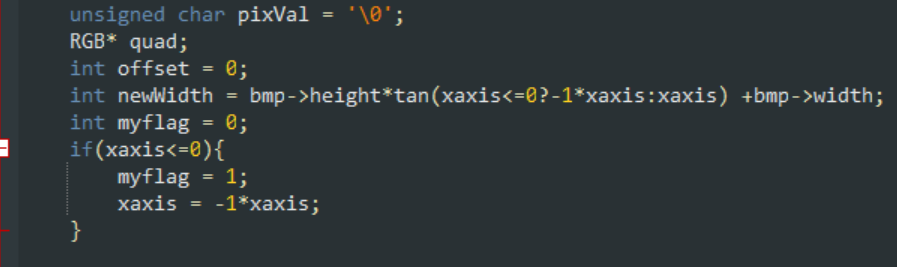
Mirror.cpp



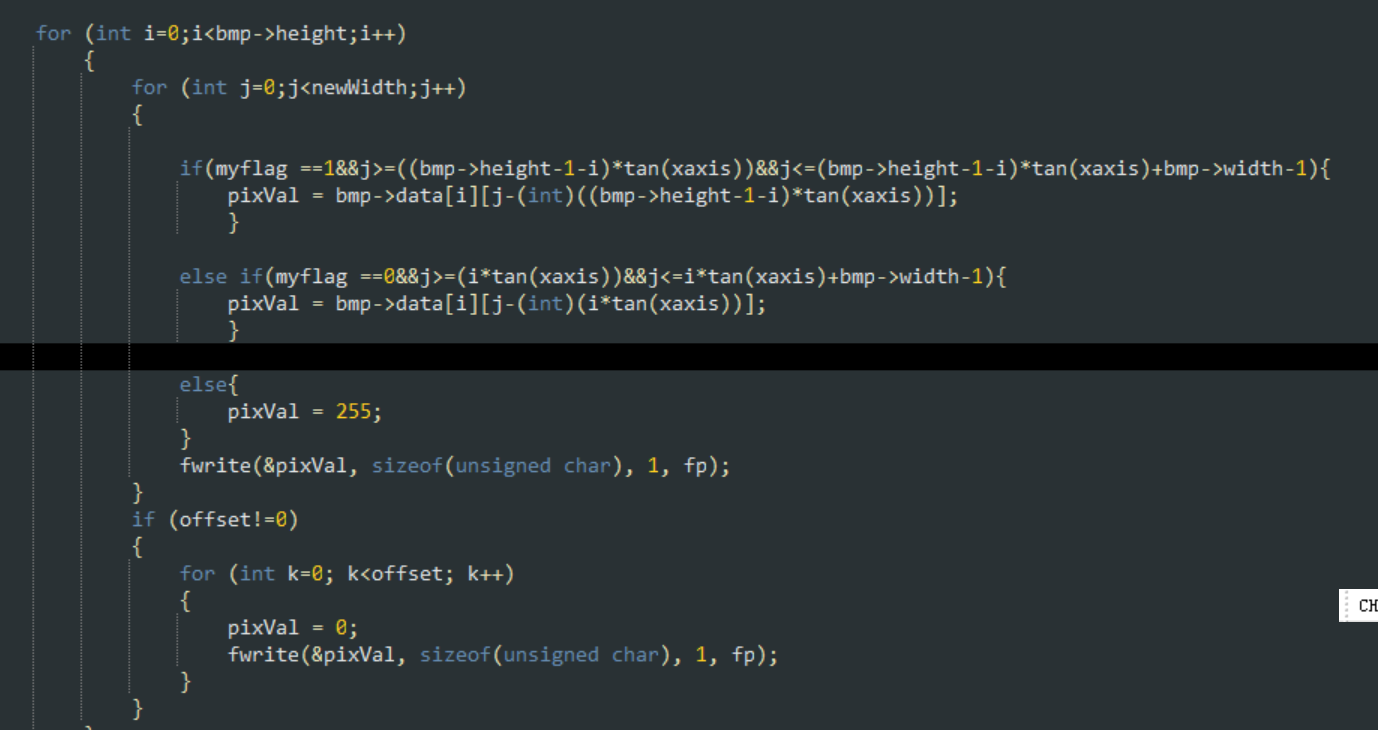
镜像操作较为简单，在输出时倒序进行输出即可，例如对x方向镜像则j从大到小循环，y方向则i从大到小循环，图中代码同时进行x方向与y方向的镜像操作。

Shear.cpp

错切操作与平移操作类似，

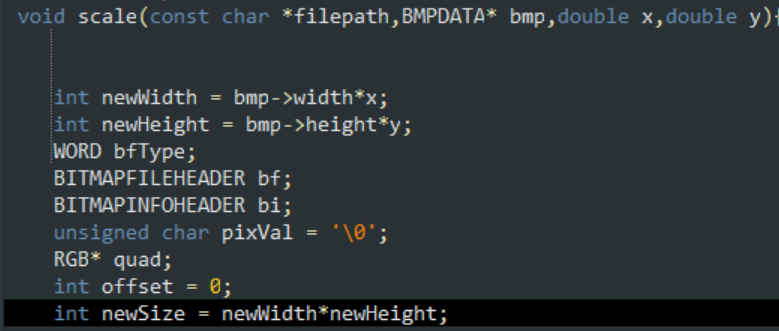


先对传入参数正负进行判断，其次利用math.h中tan()函数对整张画布大小进行重构。



借用myflag函数值进行判断，若myflag =1，则传入参数为负，构图时留白区域也进行相应转换，translation函数想法相同，在遍历时对显示图片区域进行坐标转换，得到错切后图片进行输出。

Scale.cpp



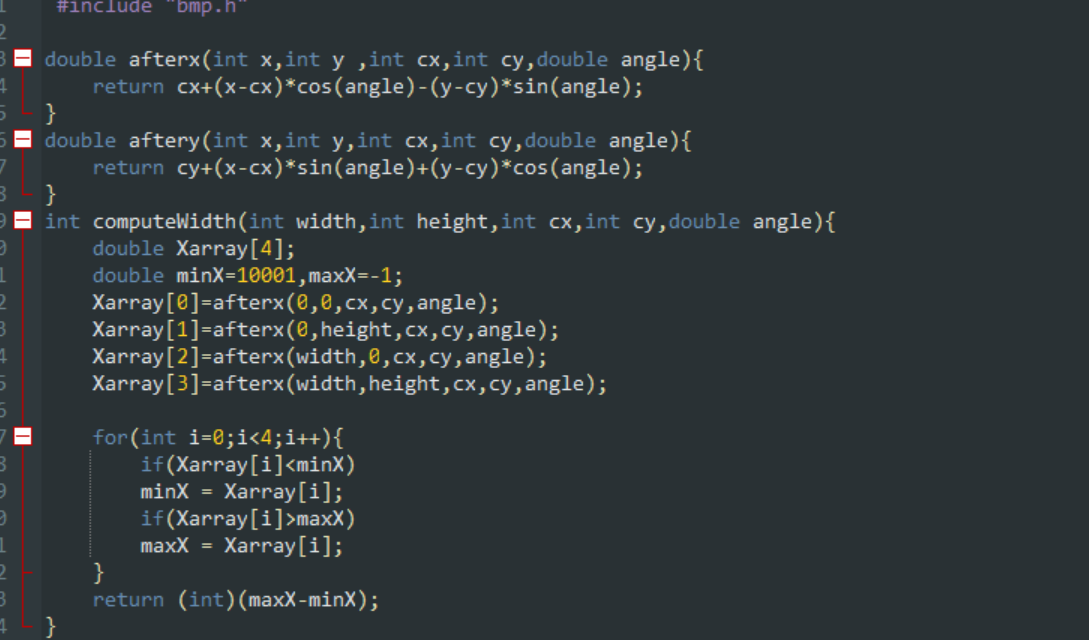
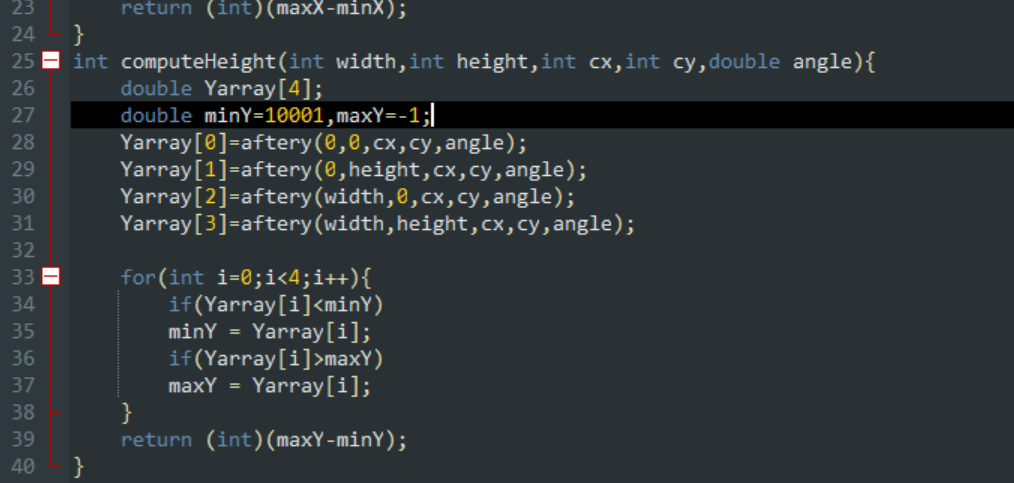
利用放大倍数x,y进行整个画布大小的重构，并改写bmp结构中的头部分，使其大小适应newSize;



新建newdata指针对应画布，先对原data中元素进行变换，此次选取最临近插值，对越界部分进行填充，其他部分进行对应坐标填充，最后得到缩放后图片进行输出。

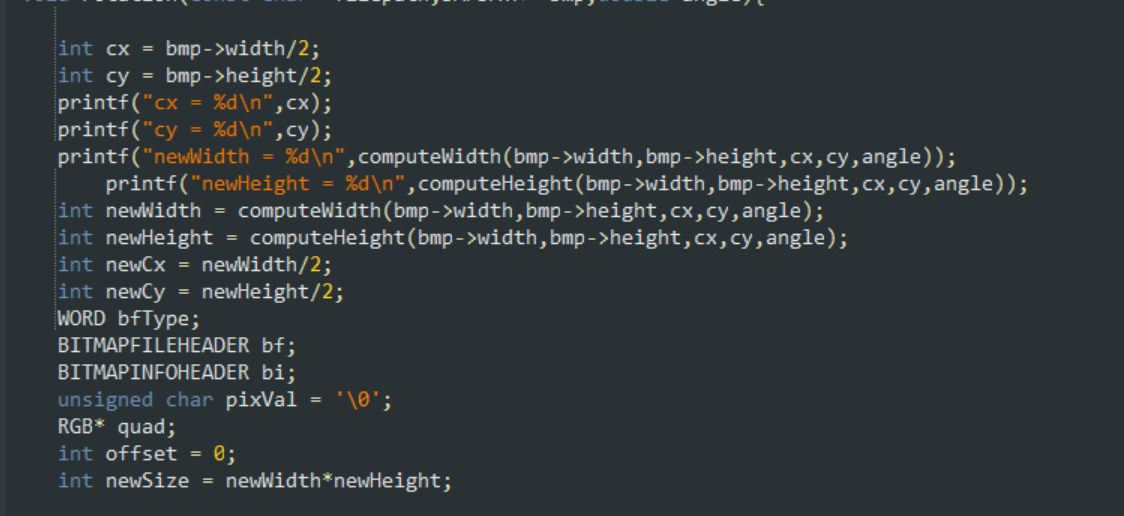
Rotation.cpp

旋转部分操作比较复杂，采取几个函数复合实现

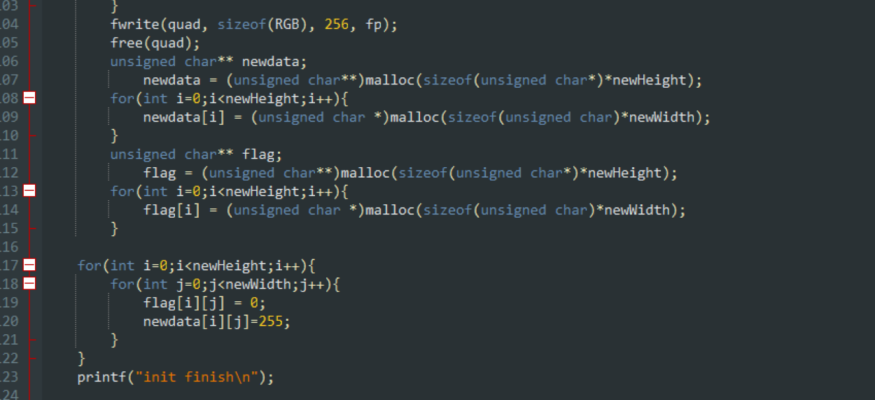
其中afterx,aftery函数进行对点旋转后坐标的计算，

computeWidth,computeHeight函数进行对整个画布重构大小的计算。

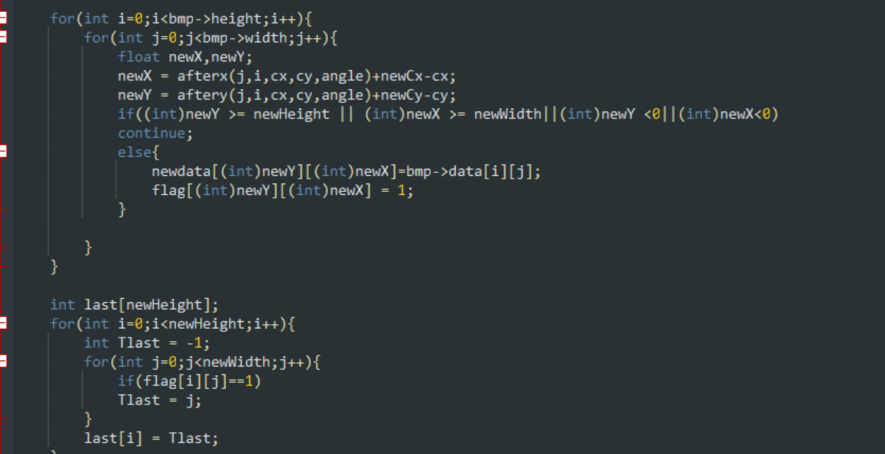


其中cx,cy表示原画布的中心位置

newCx,newCy表示旋转后重构画布的中心位置



其中newdata用来表示新画布每个像素的值，flag代表该位置是否由原Data经过坐标变换而得到。

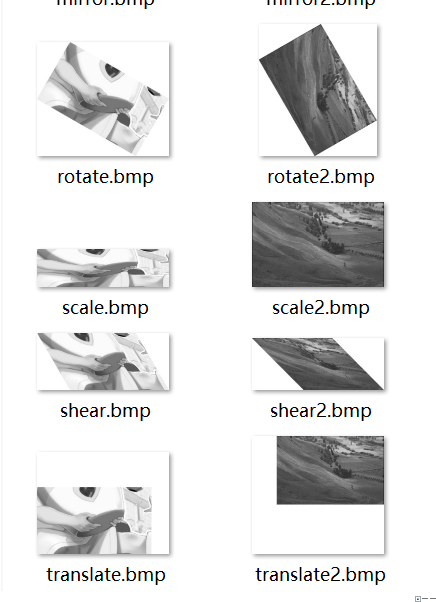


如果是经过坐标变换得到，flag对应位置设为1，并由此计算每一行最后一个有值的位置。

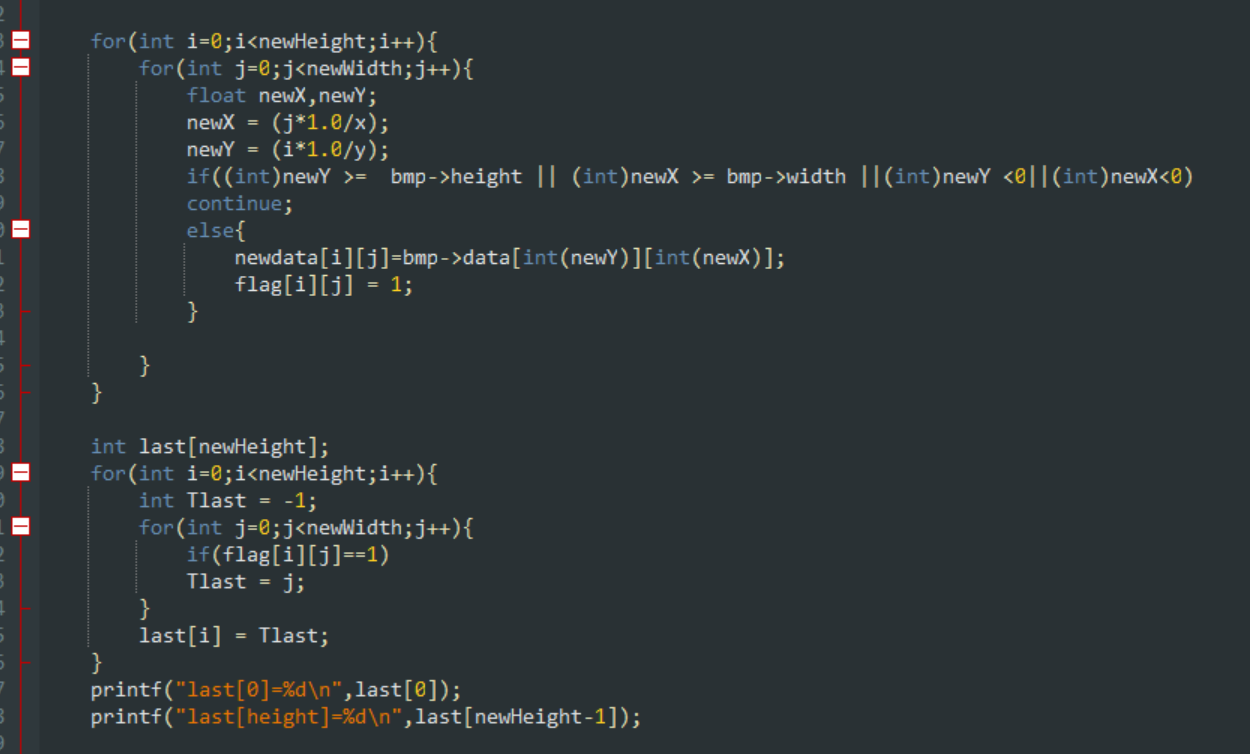


最后，在遍历时判断是否为最后一个有值位置，如果不是则对其中未赋值的点进行插值赋值，最后得到旋转后图像进行输出。

最后结果如图所示：

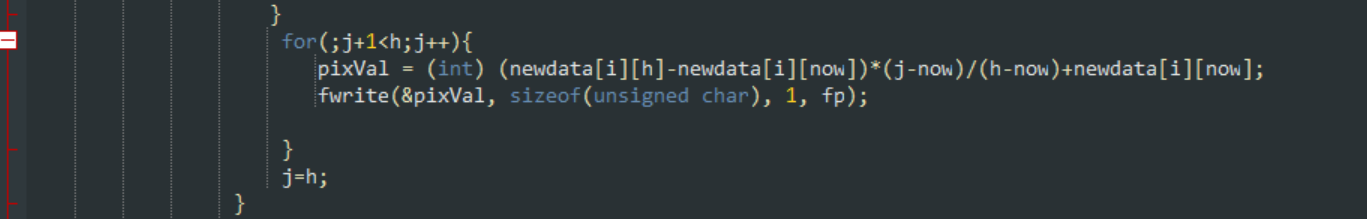


附：后期改用线性插值，对每一行数据用Flag数组进行整理，转换后的图像有像素值则flag为1，对有像素值的点进行线性变换。





更改后scale.cpp



更改后的rotation.cpp

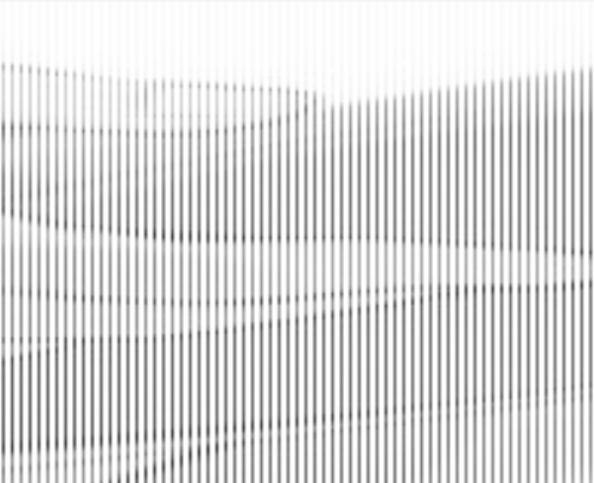
### 遇到的问题及心得

本次实验虽然感觉难度不大，但实际操作时还是遇到了一些问题，下一一列出。

1. 进行平移、错切操作时对传入参数为负时的考虑。

在最初稿成型时，并没有考虑到负数情况，第一次尝试后发现并不能得到预期效果，于是在这两个函数中针对负数情况进行了重写，最终得到普适图。

1. 在scale函数中遇到的问题



最初并没有考虑进行插值，得到的图片只是对data变换进行了操作，产生白条，后期对其进行改写，得到适应效果。

其次，在构建newdata时，最初采用

unsigned char newdata[newHeight][newWidth]={0};

方式进行设置，发现此种形式并不可行，在扩大10倍时进行到这一步便会产生Bug。

随后针对此形式进行改写，得到了预期效果。

unsigned char\*\* newdata;

newdata = (unsigned char\*\*)malloc(sizeof(unsigned char\*)\*newHeight);

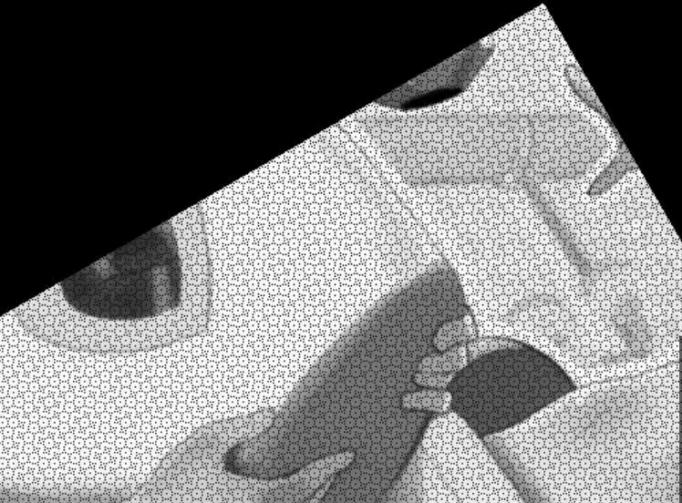
for(int i=0;i<newHeight;i++){

newdata[i] = (unsigned char \*)malloc(sizeof(unsigned char)\*newWidth);

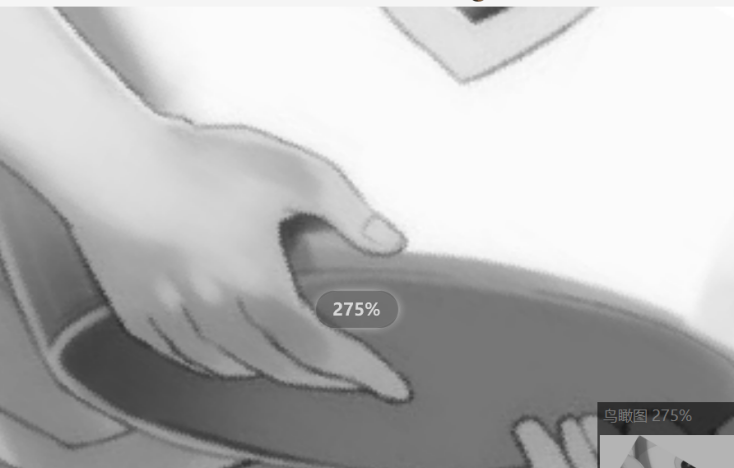
}

1. 在旋转函数中遇到的问题。

插值函数没有构建好，导致产生花纹



对插值部分进行重新改写，消除了花纹。



其次，在最初时没有考虑到整体画布改变后原中心与现中心也产生了偏差，坐标变换最初也出现了问题。

心得：

经过此次实验，收获蛮大，但些许地方还有改进可能，比如在插值部分进行二插值算法，但在课下找了部分资料进行学习，还是没有写出二插值算法的思路，因此改用了最近插值算法。

另外，经过旋转操作得到的图片不够细腻，像素点选取不到位，还需进一步加强。

总体来看，对几何变换有了更深一步的认知，也看到了实力上的差距。

后期改用了线性插值算法，得到转换后的图像更加细腻。