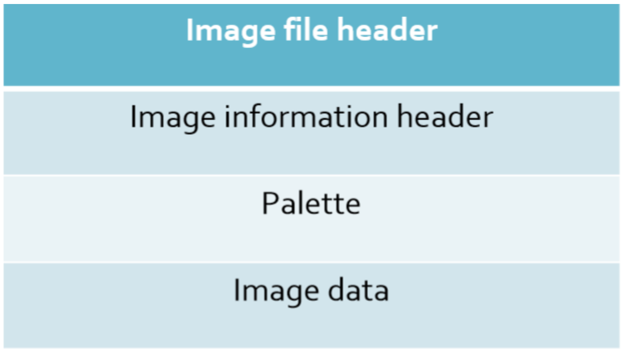
Assignment 1

/*叶俊利 3130100672*/

1. Reading a BMP file

The format of bmp file is shown below:



According the standard of bmp file, I read in a bmp file through the following code:

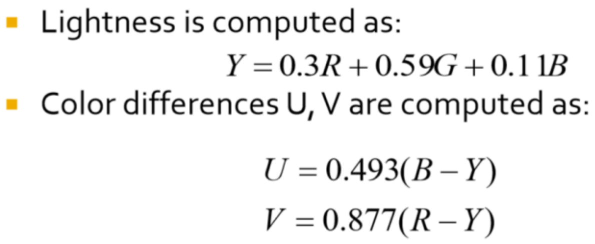
```
        BITMAPFILEHEADER bf; //BMP文件头结构体
        BITMAPINFOHEADER bi; //BMP信息头结构体
        FILE* fp;           //指向文件的指针
        RGBQUAD *ipRGB; //
        DWORD LineByte, ImgSize;
        unsigned char * * Imgdata;
        int i, j;
        char fileName[256];

        //打开文件

        fp = fopen("original.bmp", "rb");
        if (fp == NULL){
                printf("Open file error!");
                exit(0);
        }

        //读取信息头、文件头
        fread(&bf, sizeof(BITMAPFILEHEADER), 1, fp); //把指针fp所指向的文件的头信息写入bf
  （地址）
        fread(&bi, sizeof(BITMAPINFOHEADER), 1, fp);
        LineByte = bi.biSizeImage / bi.biHeight; //计算位图的实际宽度
        ImgSize = (DWORD)LineByte*bi.biHeight;
        Imgdata = new unsigned char*[bi.biHeight]; //声明一个指针数组
        RGB** img_rgb = (RGB**)malloc(bi.biHeight*sizeof(RGB*));
        for (int i = 0; i < bi.biHeight; i++)
                img_rgb[i] = (RGB*)malloc(sizeof(RGB)*bi.biWidth);
        YUV** img_yuv = (YUV**)malloc(bi.biHeight*sizeof(YUV*));
        for (int i = 0; i < bi.biHeight; i++)
                img_yuv[i] = (YUV*)malloc(sizeof(YUV)*bi.biWidth);
        if (bi.biBitCount == 24){
                for (i = 0; i < bi.biHeight; i++)
```

```
                        Imgdata[i] = new unsigned char[(bi.biWidth * 3 + 3) / 4 * 4]; //
每个数组元素也是一个指针数组
                for (i = 0; i < bi.biHeight; i++)
                for (j = 0; j < (bi.biWidth * 3 + 3) / 4 * 4; j++)
                        fread(&Imgdata[i][j], 1, 1, fp);//每次只读取一个字节，存入数组
        }
        for (int i = 0; i < bi.biHeight; i++)
        for (int j = 0; j < bi.biWidth; j++){
                img_rgb[i][j].b = Imgdata[i][3 * j];
                img_rgb[i][j].g = Imgdata[i][3 * j + 1];
                img_rgb[i][j].r = Imgdata[i][3 * j + 2];
        }
        fclose(fp);
```

In the code shown, img_rgb[][] denotes the rgb data of the bit map, and thus we can convert it into YUV format.

2. Converting to YUV format

The equation of converting rgb to yuv is shown below:



- Lightness is computed as:

$$Y = 0.3R + 0.59G + 0.11B$$

- Color differences U, V are computed as:

$$U = 0.493(B - Y)$$
$$V = 0.877(R - Y)$$

Using the following code, we can convert the file from rgb format to yuv format.

```
        for (int i = 0; i < bi.biHeight; i++)
        for (int j = 0; j < bi.biWidth; j++){
                img_yuv[i][j].y = 0.3*img_rgb[i][j].r + 0.59*img_rgb[i][j].g +
0.11*img_rgb[i][j].b;
                img_yuv[i][j].u = 0.493*(img_rgb[i][j].b - img_yuv[i][j].y);
                img_yuv[i][j].v = 0.877*(img_rgb[i][j].r - img_yuv[i][j].y);}
```

3. Converting to grayscale BMP

To convert it to into grayscale bmp, first we need to change information about biBitCount, biSizeImage, bfOffBits and bfSize, and then we need to build a palette. At last, we write the grayscale data in to the file. The code is shown:

```
        fp = fopen("gray.bmp", "wb");
        BITMAPFILEHEADER bf2;
        BITMAPINFOHEADER bi2;
        int nBytesPerLine2 = ((bi.biWidth + 3) / 4) * 4;
        int nImageSize2 = nBytesPerLine2 * bi.biHeight;
        memcpy(&bi2, &bi, sizeof(BITMAPINFOHEADER));
        bi2.biBitCount = 8;
        bi2.biSizeImage = nImageSize2;
```

```
bf2.bfType = 0x4d42;
bf2.bfReserved1 = bf2.bfReserved2 = 0;
bf2.bfOffBits = sizeof(bf2)+sizeof(BITMAPINFOHEADER)+256 * sizeof(RGBQUAD);
bf2.bfSize = bf2.bfOffBits + nImageSize2;
fwrite(&bf2, sizeof(BITMAPFILEHEADER), 1, fp);
fwrite(&bi2, sizeof(BITMAPINFOHEADER), 1, fp);
RGBQUAD *ipRGB2 = (RGBQUAD *)malloc(256 * sizeof(RGBQUAD));
for (i = 0; i < 256; i++)
{
        ipRGB2[i].rgbRed = ipRGB2[i].rgbGreen = ipRGB2[i].rgbBlue = i;
        ipRGB2[i].rgbReserved = 0;
}
fwrite(ipRGB2, sizeof(RGBQUAD), 256, fp);
unsigned char *ImgData2 = new unsigned char[nImageSize2];
for (i = 0; i < bi.biHeight; i++)
{
        for (int j = 0; j < bi.biWidth; j++)
                ImgData2[i*nBytesPerLine2 + j] = img_yuv[i][j].y;
}
fwrite(ImgData2, nImageSize2, 1, fp);
fclose(fp);
```

4. Changing Y value and Converting back to rgb BMP

First, we adjust the value of Y, as the range of Y is [0, 255], we need to pay attention to the value when we adjust it.

```
for (int i = 0; i < bi.biHeight; i++)
        for (int j = 0; j < bi.biWidth; j++)
                img_yuv[i][j].y = (img_yuv[i][j].y + 30 > 255) ? 255 : (img_yuv[i][j].y + 30);
```

Then, the equation to convert yuv into rgb is shown below:

$$R = Y + 1.14V$$
$$G = Y - 0.39U - 0.58V$$
$$B = Y + 2.03U$$

Thus we can convert it back to rgb:

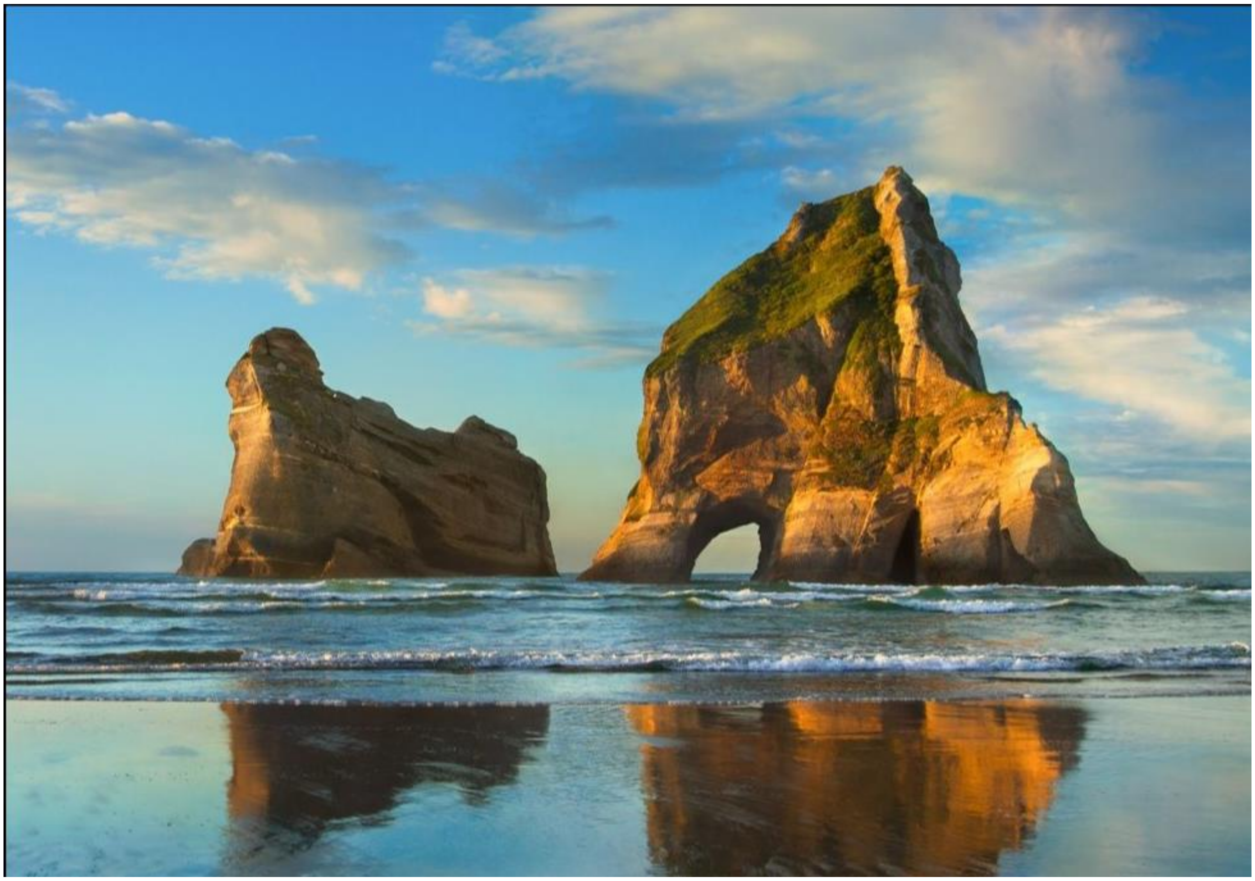
```
        for (int i = 0; i < bi.biHeight; i++)
        for (int j = 0; j < bi.biWidth; j++){
                img_rgb[i][j].r = img_yuv[i][j].y + 1.14 * img_yuv[i][j].v;
                img_rgb[i][j].b = img_yuv[i][j].y + 2.03 * img_yuv[i][j].u;
                img_rgb[i][j].g = img_yuv[i][j].y - 0.39*img_yuv[i][j].u - 0.58*1.14 *
img_yuv[i][j].v;

}
```

At last, we write all the data back to a bmp file.

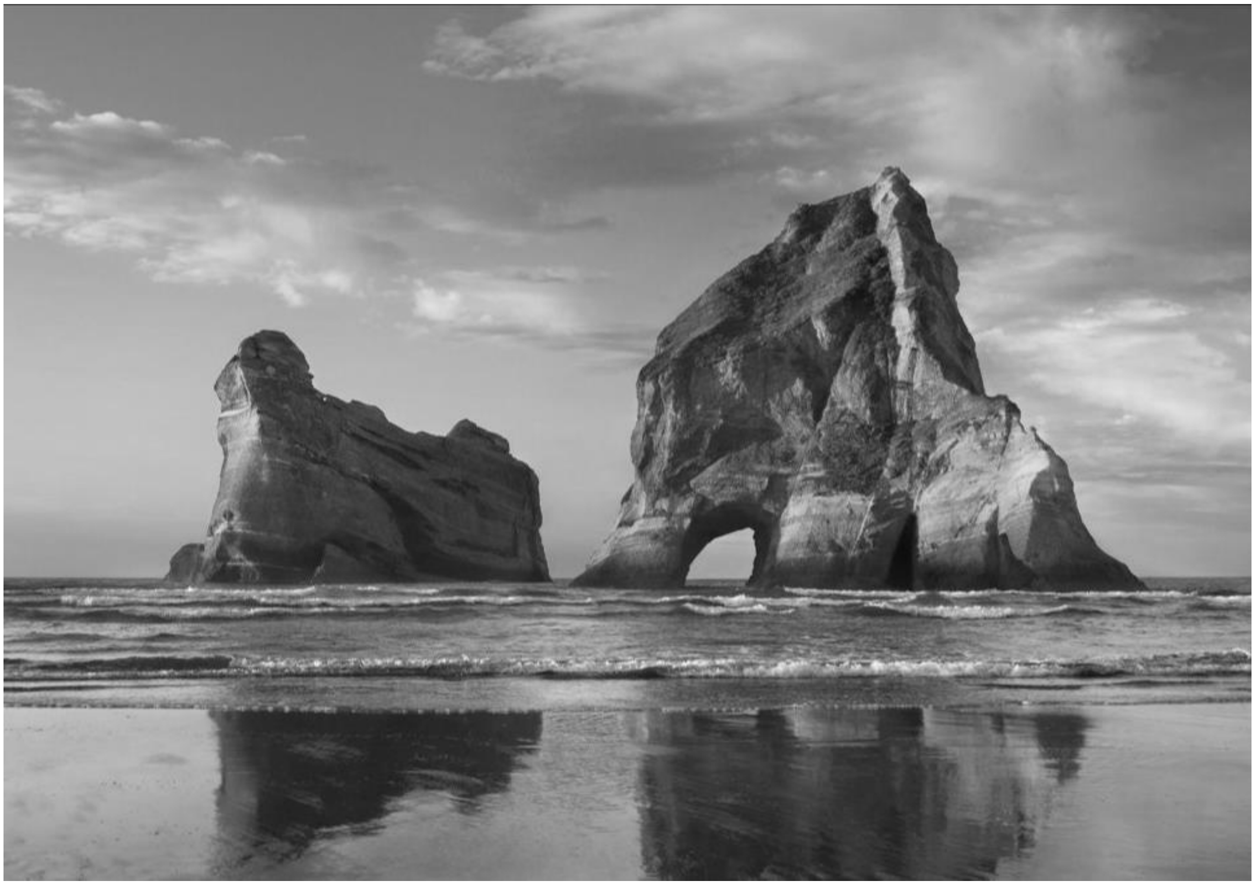
completed.

5. Results

Original picture:



Grayscale picture:



Increase Y by 30, and then convert it back to rgb: