第三周实验报告

**《程序设计基础课程设计》实验报告**

班级: 2班

姓名: 周钰坤,申康

学号: 21009200438

所选题目: 3\_2

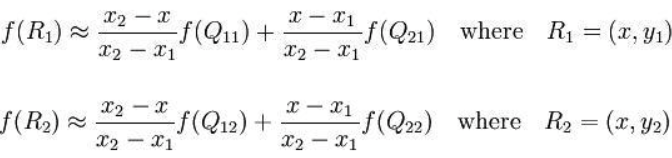
第3\_2题

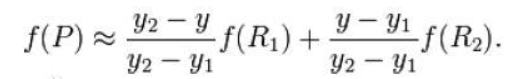
算法描述:

* 整个bmp文件的内容可以分为3到4块，第一块是bmp的文件头用于描述整个bmp文件的情况。结构如下：

​ 第一个bfType用于表示文件类型，如果它是bmp文件，那么它这个位置的值一定是”BM” 也就是0x4D42。第二个bfSize表示整个文件的字节数。第三第四个 则保留，目前无意义，最后一个相当重要，表示，位图的数据信息离文件头的偏移量，以字节为单位。

* 第二块是位图信息头，即BITMAPINFOHEADER，用于描述整个位图文件的情况，结构如下：
* 第三块就是调色板信息或者掩码部分，如果是8位位图 则存放调色板 ；16 与32位位图则存放RGB颜色的掩码，这些掩码以DWORD大小来存放。
* 为了进行放缩操作，我们根据放缩的比例把后来对应的图片的位置的像素，映射到原先的像素位置，用原先的像素进行填充。
* 位图快速缩放公式：





源代码:

Zoom.cpp

#include<stdio.h>

#include<malloc.h>

#include<string.h>

#include<stdlib.h>

#include<math.h>

//文件头

typedef struct tagBITMAPFILEHEADER {

    //unsigned short bfType;      // 保存图片类型，读取时需要注释掉，文本标识符只能单独进行读写

    unsigned int   bfSize;        // 文件大小

    unsigned short bfReserved1;   // 保留，设置为0

    unsigned short bfReserved2;   // 保留，设置为0

    unsigned int   bfOffBits;     // 从文件头到实际的图像数据之间的字节的偏移量(没调色板的话是54)

}BITMAPFILEHEADER;

//信息头

typedef struct tagBITMAPINFOHEADER {

    unsigned int    biSize;          // 此结构体的大小

    unsigned int    biWidth;         // 图像的宽

    unsigned int    biHeight;        // 图像的高

    unsigned short  biPlanes;        // 颜色平面数 恒为1

    unsigned short  biBitCount;      // 一像素所占的位数 Windows系统有8,16,24

    unsigned int    biCompression;   // 说明图象数据压缩的类型，0为不压缩

    unsigned int    biSizeImage;     // 图像大小, 值等于上面文件头结构中bfSize-bfOffBits

    int             biXPelsPerMeter; // 说明水平分辨率，用像素/米表示 一般为0

    int             biYPelsPerMeter; // 说明垂直分辨率，用像素/米表示 一般为0

    unsigned int    biClrUsed;       // 说明位图实际使用的彩色表中的颜色索引数（设为0的话，则说明使用所有调色板项）

    unsigned int    biClrImportant;  // 说明对图象显示有重要影响的颜色索引的数目 如果是0表示都重要

}BITMAPINFOHEADER;

//调色板

typedef struct tagRGBQUAND{

    unsigned char r;

    unsigned char g;

    unsigned char b;

    unsigned char rgbReserved;

}RGBQUAND;

int main(int argc, char\* argv[]){

    //打开图片

    char \*oldPhoto = argv[1], \*newPhoto = argv[3];

    FILE \*fp1 = fopen(oldPhoto, "rb"), \*fp2 = fopen(newPhoto, "wb");

    double pzoom = atof(argv[2]) / 100.0;

    if (fp1 == NULL || fp2 == NULL) {

        printf("Opening photos failed!\n");

        if(fp1 == NULL) fclose(fp1);

        if(fp2 == NULL) fclose(fp2);

        return -1;

    }

    //单独读取bmp图片文本标识符0x4d42

    unsigned short fileType;

    fread(&fileType, sizeof(unsigned short), 1, fp1);

    if(fileType != 0x4d42){//如果不是的话证明不是bmp图片

        printf("The photo is not of bmp type!\n");

        return -1;

    }

    //读取原图信息

    BITMAPFILEHEADER fileHeader;    // 原图文件头

    BITMAPINFOHEADER infoHeader;    // 原图消息头

    fread(&fileHeader, sizeof(BITMAPFILEHEADER), 1, fp1);

    fread(&infoHeader, sizeof(BITMAPINFOHEADER), 1, fp1);

    int byte = infoHeader.biBitCount / 8;//每个像素的字节数

    RGBQUAND \*palette = (RGBQUAND\*)malloc((int)pow(2, infoHeader.biBitCount) \* 4);//分配调色板空间

    if(infoHeader.biBitCount != 24)//如果是24位图的没有调色板

        fread(palette, sizeof(RGBQUAND), (int)pow(2, infoHeader.biBitCount), fp1);

    //得到原图宽高和修改后的宽高

    unsigned int oldWidth, oldHeight, newWidth, newHeight;

    oldWidth = infoHeader.biWidth;

    oldHeight = infoHeader.biHeight;

    printf("Oldphoto's height:%d\n", oldHeight);

    printf("Oldphoto's width:%d\n", oldWidth);

    //图像显示不出来原因在于图像长或宽不是4的倍数 下面这一步可以保证得到的宽高是4的倍数

    newHeight = ((int)(oldHeight \* pzoom) + 3) / 4 \* 4;

    newWidth = ((int)(oldWidth \* pzoom) + 3) / 4 \* 4;

    printf("Newphoto's height:%d\n", newHeight);

    printf("Newphoto's width:%d\n", newWidth);

    unsigned int oldsize = oldWidth \* oldHeight \* byte, newsize = newWidth \* newHeight \* byte;

    //获取原图位图数据

    unsigned char \*sourceData = (unsigned char\*)malloc(oldsize);

    if(infoHeader.biBitCount == 24){//无调色板时

        fseek(fp1, 54, SEEK\_SET);//文件指针指向文件的第54个字节

        fread(sourceData, oldsize, 1, fp1);

    }else if(infoHeader.biBitCount == 8){//有调色板是要加上分配调色板所需要的空间

        fseek(fp1, 1078, SEEK\_SET);//文件指针指向文件的第54+2^8\*4=1078个字节

        fread(sourceData, oldsize, 1, fp1);

    }

    //修改两个header的数据并把修改后的header(及调色板信息)写入新图片中

    infoHeader.biWidth = newWidth;

    infoHeader.biHeight = newHeight;

    if(infoHeader.biBitCount == 24){

        fileHeader.bfSize = 54 + newsize;

        infoHeader.biSizeImage = newsize;

    }else if(infoHeader.biBitCount == 8){

        fileHeader.bfSize = 1078 + newsize;

        infoHeader.biSizeImage = newsize;

    }

    fwrite(&fileType, sizeof(unsigned short), 1, fp2);

    fwrite(&fileHeader, sizeof(BITMAPFILEHEADER), 1, fp2);

    fwrite(&infoHeader, sizeof(BITMAPINFOHEADER), 1, fp2);

    if(infoHeader.biBitCount!=24)   fwrite(palette,sizeof(RGBQUAND),pow(2,infoHeader.biBitCount),fp2);

    //使用双线性差值法进行图片缩放

    double p,q;

    unsigned int x1, y1, x2, y2;                //原图所在像素点的宽高

    unsigned int X, Y;

    unsigned char \*pDestination;                //修改像素的位置（即字节偏移量）

    unsigned char \*pSource1,\*pSource2;          //获取像素的位置（即字节偏移量）

    unsigned char \*destinationData = (unsigned char\*)malloc(newsize);//开好新图片的位图数据所需空间

    for(Y = 0; Y < newHeight; Y++){

        y1 = Y / pzoom;

        y2 = Y / pzoom + 1;

        q = Y / pzoom - y1;

        pDestination = destinationData + Y \* newWidth \* byte;

        pSource1 = sourceData + y1 \* oldWidth \* byte;

        pSource2 = sourceData + y2 \* oldWidth \* byte;

        for(X = 0; X < newWidth; X++){

            x1 = X / pzoom;

            x2 = X / pzoom + 1;

            p = X / pzoom - x1;

            if(byte == 3){

            \*(pDestination + X\*byte) = \*(pSource1 + x1\*byte) \* (1-p) \* (1-q) +

                                       \*(pSource1 + x2\*byte) \* p \* (1-q) +

                                       \*(pSource2 + x1\*byte) \* (1-p) \* q +

                                       \*(pSource2 + x2\*byte) \* p \* q;

            \*(pDestination + X\*byte+1) = \*(pSource1 + x1\*byte+1) \* (1-p) \* (1-q) +

                                         \*(pSource1 + x2\*byte+1) \* p \* (1-q) +

                                         \*(pSource2 + x1\*byte+1) \* (1-p) \* q +

                                         \*(pSource2 + x2\*byte+1) \* p \* q;

            \*(pDestination + X\*byte+2) =\*(pSource1 + x1\*byte+2) \* (1-p) \* (1-q) +

                                        \*(pSource1 + x2\*byte+2) \* p \* (1-q) +

                                        \*(pSource2 + x1\*byte+2) \* (1-p) \* q +

                                        \*(pSource2 + x2\*byte+2) \* p \* q;

            }

            else if(byte == 1){

                \*(pDestination + X\*byte) = \*(pSource1 + x1\*byte) \* (1-p) \* (1-q) +

                                           \*(pSource1 + x2\*byte) \* p \* (1-q) +

                                           \*(pSource2 + x1\*byte) \* (1-p) \* q +

                                           \*(pSource2 + x2\*byte) \* p \* q;

            }

        }

    }

    //将位图数据写入新的图片并进行后续处理

    fwrite(destinationData, newsize, 1, fp2);

    printf("success!\n");

    free(destinationData);

    free(sourceData);

    free(palette);

    fclose(fp1);

    fclose(fp2);

    return 0;

}

测试结果:

编译: g++ zoom.cpp -o zoom.exe

运行: zoom.exe origin.bmp 200 target1.bmp

zoom.exe origin.bmp 100 target2.bmp

