**《程序设计基础课程设计》第3周实验报告**

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：一、文件处理—— 二选一

**第5题**

**1.原始题目及要求；**

**（1）位图图像文件缩放**

要求：编写一个程序，可以在命令行输入参数，完成指定文件的缩放，并存储到新文件，命令行参数如下

zoom file1.bmp 200 file2.bmp

第一个参数为可执行程序名称，第二个参数为原始图像文件名，第三个参数为缩放比例（百分比），第四个参数为新文件名

**2.题目的分析**

（1）涉及知识点

文件读写、结构体定义、内存管理、基本图像处理算法、命令行参数等

（2）题目功能理解

由于文图文件特有的图片阻止形式，所以需要定义特定的结构体来存储文图文件的头部信息。而位图文件的放缩则可以采用简单的线性插值算法实现。

**3. 题目的总体设计：设计思路、算法描述**

（1）程序模块：

（A）图片放缩模块

实现位图文件的放缩。

（2）模块调用关系

模块间无相互调用关系。

（3）输入输出数据说明

输入数据：形如zoom file1.bmp 200 file2.bmp的命令行命令。

输出数据：放缩后的位图。

（4）总体流程

1）命令行输入形如的命令zoom file1.bmp 200 file2.bm，若成功则显示成功，并生成指定的文件，若失败则提示失败。

**4. 各功能模块/函数的设计说明：**

（1）bool bmp\_zoom( const char \*src, const char \*dst , int percent )

函数名称: bmp\_zoom

函数功能: 图片放大缩小

参数: const char \*src( 要被放缩的图片名称 ), const char \*dst( 放缩结果的名称 ) ,

int percent( 放缩的百分比 )

返回值: bool( 表示操作是否存在成功 )

执行流程：

1）打开源图片并创建目标图片，成功则转2，若失败则返回false。

2）获取源图片文件头部信息转3.

3）获取原图片的位图数据转4。

4）修改原照片的宽高，并将修改过的头部信息写入新图片中，转5.

5）创建存储图片内容的缓冲区，若失败返回false，否则转6.

6）采用线性插值法进行图片放缩，结果写入文件，释放堆空间，关闭文件，返回

true。

可能结果：

1）放缩失败，返回false。

2）放缩成功，返回true。

**5. 源程序：**

（1）bmp\_zoom \_test.c

#include"bmp\_zoom.h"

#include"stdio.h"

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

if( argc != 4 ){

printf("参数错误\n");

return 0;

}

char \*src = argv[1];

int percent = atoi( argv[2] );

char \*dst = argv[3];

printf("%s", bmp\_zoom( src, dst, percent )? "放缩成功\n":"放缩失败\n");

return 0;

}

（2）bmp\_zoom.h

#ifndef \_\_BMP\_ZOOM\_H

#define \_\_BMP\_ZOOM\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#define COLOR\_WIDTH 3

#pragma pack(1) /\* 必须在结构体定义之前使用,这是为了让结构体中各成员按1字节对齐\*/

typedef struct tagBITMAPFILEHEADER {

unsigned short bfType; //保存图片类型。 'BM'

unsigned int bfSize; //位图文件的大小，以字节为单位（3-6字节，低位在前）

unsigned short bfReserved1; //位图文件保留字，必须为0(7-8字节）

unsigned short bfReserved2; //位图文件保留字，必须为0(9-10字节）

unsigned int bfOffBits; // RGB数据偏移地址,位图数据的起始位置，以相对于位图（11-14字节，低位在前）

} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER {

unsigned int biSize; //本结构所占用字节数（15-18字节）

unsigned int biWidth; //位图的宽度，以像素为单位（19-22字节）

unsigned int biHeight; //位图的高度，以像素为单位（23-26字节）

unsigned short biPlanes; //目标设备的级别，必须为1(27-28字节）

unsigned short biBitCount; //每个像素所需的位数，必须是1（双色）（29-30字节）,4(16色），8(256色）16(高彩色)或24（真彩色）之一

unsigned int biCompression; //位图压缩类型，必须是0（不压缩），（31-34字节）

// 1(BI\_RLE8压缩类型）或2(BI\_RLE4压缩类型）之一

unsigned int biSizeImage; //位图的大小(其中包含了为了补齐行数是4的倍数而添加的空字节)，以字节为单位（35-38字节）

unsigned int biXPelsPerMeter; //位图水平分辨率，每米像素数（39-42字节）

unsigned int biYPelsPerMeter; //位图垂直分辨率，每米像素数（43-46字节)

unsigned int biClrUsed; //位图实际使用的颜色表中的颜色数（47-50字节）

unsigned int biClrImportant; //位图显示过程中重要的颜色数（51-54字节）

} BITMAPINFOHEADER;

/\*

\* 函数名称：bmp\_zoom

\* 函数功能：图片放大缩小

\* 参数: const char \*src( 要被放缩的图片名称 ), const char \*dst( 放缩结果的名称 ) ,

\* int percent( 放缩的百分比 )

\* 返回值：bool( 表示操作是否存在成功 )

\*/

bool bmp\_zoom( const char \*src, const char \*dst , int percent ){

FILE \*fpsrc = fopen(src, "rb");// 打开源图片

FILE \*fpdst = fopen(dst, "wb");// 创建目标图片

if( !fpsrc || !fpdst ){// 打开失败

return false;

}

BITMAPFILEHEADER head;

BITMAPINFOHEADER info;

memset(&head,0,sizeof(BITMAPFILEHEADER));

memset(&info,0,sizeof(BITMAPINFOHEADER));

fread(&head, sizeof(BITMAPFILEHEADER), 1, fpsrc );

fread(&info, sizeof(BITMAPINFOHEADER), 1, fpsrc );

// 获取源图片文件头部信息

unsigned int src\_width = info.biWidth; //获取原图片的宽

unsigned int src\_height = info.biHeight; //获取原图片的高

//获取原图片的位图数据

unsigned char \*src\_data = (unsigned char \*)malloc(src\_width \* src\_height \* COLOR\_WIDTH);

fseek(fpsrc, 54, SEEK\_SET);

fread(src\_data, src\_width \* src\_height \* COLOR\_WIDTH, 1, fpsrc);

//修改原照片的宽高

double scale = (double)percent/100;

unsigned int dst\_width = src\_width \* scale ;

unsigned int dst\_height = src\_height \* scale;

double rate\_y = (double)src\_height/dst\_height;

double rate\_x = (double)src\_width/dst\_width;

head.bfSize = dst\_width \* dst\_height \* COLOR\_WIDTH + 54;

info.biWidth = dst\_width;

info.biHeight = dst\_height;

//将修改过的头信息写进新照片

fwrite(&head, sizeof(BITMAPFILEHEADER), 1, fpdst);

fwrite(&info, sizeof(BITMAPINFOHEADER), 1, fpdst);

unsigned char \*dst\_data = (unsigned char \*)malloc(dst\_width \* dst\_height \* COLOR\_WIDTH);

//创建存储图片内容的缓冲区

if( !dst\_data ){

return false;

}

//采用线性插值法进行图片放缩

for(unsigned int dst\_y = 0; dst\_y<dst\_height; ++dst\_y){

unsigned int src\_y = (unsigned int)(rate\_y \* dst\_y);

for(unsigned int dst\_x=0; dst\_x<dst\_width; ++dst\_x){

unsigned int src\_x = (unsigned int)(rate\_x \* dst\_x);

memcpy( dst\_data+(dst\_y\*dst\_width+dst\_x)\*COLOR\_WIDTH, src\_data+(src\_y\*src\_width+src\_x)\*COLOR\_WIDTH, COLOR\_WIDTH );

}

}

//结果写入文件

fseek(fpdst, 54, SEEK\_SET);

fwrite(dst\_data, dst\_width \* dst\_height \* 3, 1, fpdst);

//释放堆空间

free(dst\_data);

free(src\_data);

//关闭文件

fclose(fpsrc);

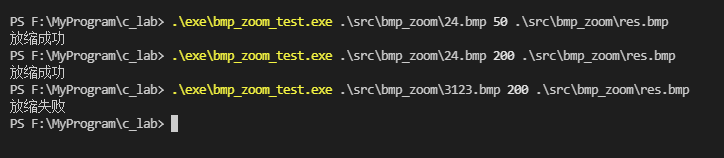
fclose(fpdst);

return true;

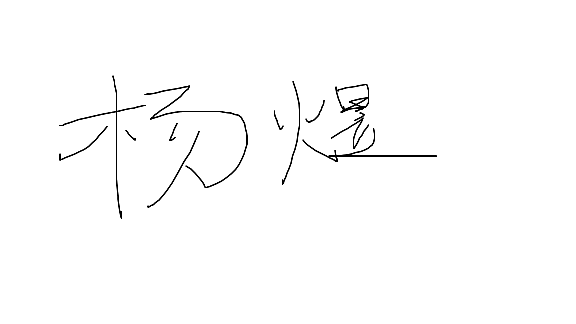
}

#endif

**6. 测试数据（输入、输出）：**



下面两图分别是原图和缩小后的图片，放大图片由于过大无法在文档中显示。

放大的图片有无

**7. 小结：**

本次实验圆满完成，实验结果符合实验要求。

实验过程中，我复习到了c语言中文件读写、结构体定义、内存管理、基本图像处理算法、命令行参数等重要知识，收货颇丰。

但此次实验还并不完美，由于浮点数与整型之间的舍入问题，导致个小数倍的放缩在某些特定情况下会出现不完美的结果。