**《C语言课程设计》实验报告**

专业：­ 计算机科学与技术 班级： 1803011

姓名： 张帅豪 学号： 18030100101 分工：50%

姓名： 赵宇轩 学号： 18030100103 分工：50%

1. **问题描述**

**题目3：**位图图像文件缩放

**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、bmp图像组成。

**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

**说明：**通过该练习，学生需要掌握

（1）二进制文件操作

（2）位图文件的构成以及相关结构体（可以查阅资料，如https://www.cnblogs.com/l2rf/p/5643352.html）

（3）图像缩放算法（如：最临近点插值算法、双线性内插值算法、双立方插值算法等。可以选择一种算法。相关算法可以查阅资料）

1. **技术关键点分析**
2. 定义位图文件头结构

/位图文件头 长度固定：14字节   
typedef struct {   
 unsigned short int type; // 位图文件类型，必须是0x424D，即字符串“BM”   
 unsigned int size; // 位图文件大小，包括这14个字节   
 unsigned short int reserved1;   
 unsigned short int reserved2;   
 unsigned int offset; // 文件头到实际的位图数据的偏移字节数，bitmap文件前3个部分(文件头、信息头、颜色表)的长度之和   
} FILEHEADER;   
   
定义位图信息头解构

// 位图信息头 长度固定：40字节   
typedef struct {   
 unsigned int size; //本结构的长度，为40个字节   
 int width,height; //位图的宽高（单位：像素）   
   
 unsigned short int planes; // 目标设备的级别，必须是1。   
 unsigned short int bits; // 每个像素所占的位数（bit），其值必须为1（黑白图像）、4（16色图）、8（256色）、24（真彩色图），新的BMP格式支持32位色。   
 unsigned int compression; // 位图压缩类型，有效的值为BI\_RGB（未经压缩）、BI\_RLE8、BI\_RLE4......   
 unsigned int imagesize; // 实际的位图数据占用的字节数   
 int xresolution,yresolution;// 指定目标设备的水平分辨率、垂直分辨率 （单位：像素/米）   
 unsigned int ncolours; // 位图实际用到的颜色数   
 unsigned int importantcolours; // 位图显示过程中重要的颜色数   
} INFOHEADER;

3.定义文件缩放函数

（1）读取文件信息

FILE \*beg;   
 FILE \*tar;   
 //将给定文件和目标文件打开   
 beg = fopen(begin , "rb"); //读入文件   
 tar = fopen(target , "wb"); //写入文件   
 //判断读入文件是否为空   
 if(beg == NULL){   
 printf("错误：读入文件为空\n");   
 return ;   
 }

FILEHEADER filehead;   
 INFOHEADER infohead;   
 //读取给定文件的信息并保存   
 fread(&filehead , sizeof(filehead) , 1 , beg);//位图文件头   
 fread(&infohead , sizeof(infohead) , 1 , beg);//位图信息头   
 //获取宽度和高度   
 int width = infohead.width;   
 int height = infohead.height;   
 //获取真正的数据信息   
 unsigned char \*data = (unsigned char\*)malloc(sizeof(unsigned char)\*width\*3\*height);   
 fseek(beg,54,SEEK\_SET);   
 fread(data,width\*3\*height,1,beg);   
（2）计算缩放后文件大小

//新的图片的高度和宽度   
 int new\_width = (int)1.0\*a\*width;   
 int new\_height = (int)1.0\*a\*height;

（3）

//更改文件的信息头和BMP信息头   
 filehead.size = new\_width\*new\_height\*3+54;   
 infohead.height = new\_height;   
 infohead.width = new\_width;   
 fwrite(&filehead,sizeof(filehead),1,tar);   
 fwrite(&infohead,sizeof(infohead),1,tar);

（4）位图数据置于目标文件存储的数组上

//分配新的数据空间   
 unsigned char \*new\_data = (unsigned char\*)malloc(sizeof(unsigned char)\*new\_width\*3\*new\_height);   
 long src\_x,src\_y;//分别代表原先的x,原先的y;   
 //下面运行核心代码，思路是把后来的位置映射到原先的位置，把原来像素的数据写入后来像素的数据   
 long i,j;   
 for(i=0;i<new\_height;i++)   
 {   
 src\_y = (long)1.0\*i/a;   
 unsigned char \*src\_row\_begin = data + src\_y\*3\*width;//换算成原先行的起始处;   
 for(j=0;j<new\_width;j++)   
 {   
 src\_x = (long)1.0\*j/a;   
 memcpy(new\_data + i\*new\_width\*3+j\*3,src\_row\_begin + src\_x \* 3,3);   
 }   
 }

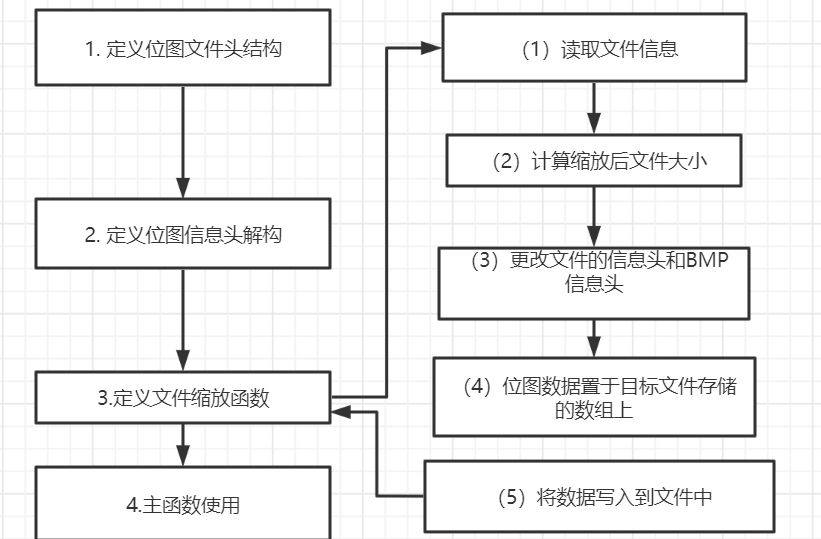
（5）将数据写入到文件中

fseek(tar,54,SEEK\_SET);   
 fwrite(new\_data,new\_width\*3\*new\_height,1,tar);   
 printf("写入文件成功！");   
 free(data);   
 free(new\_data);   
 fclose(beg);   
 fclose(tar);

4.主函数使用

double a = atoi(argv[2])/100.0;   
 change\_bmp(argv[1],argv[3],a);   
 printf("转换文件完成 1.0--->%f",a);

1. **流程图**



1. **测试与分析**

