**《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4：**RLE压缩解压算法

**涉及知识点：**文件读写、位操作、内存管理、结构体定义、RLE算法、命令行参数

**要求：**编写一个程序，可以在命令行输入参数，完成指定文件的压缩解压

**命令行参数如下**

rle file1 –c(-d) file2

第一个参数为可执行程序名称，第二个参数为原始文件名，第三个参数为压缩或解压缩选项，第四个参数为新文件名

**说明：**

通过该练习，学生需要掌握：

（1）文件操作

（2）位操作

（3）RLE算法（相关算法可以查阅资料，我们这里要求数据都是1个字节作为单位长度的）

1. **技术关键点分析**
2. 读取原始文件

FILE\* f1;   
 FILE\* f2;   
 f1 = fopen(argv[1], "rb");   
 if(f1 == NULL){   
 printf("找不到这文件!\n");   
 return 0;   
 }   
 int t = 0;   
 int a = 0;   
 //读取文件   
 while ((a = fgetc(f1)) != EOF) {   
 file\_src[t++] = a;   
 }   
 f2 = fopen(argv[3], "wb");   
 int size = t;   
 printf("输入字符数%d",size);

1. 选择解码还是编码

if( strcmp(argv[2], "-d") == 0){   
 size = RLEdecode(file\_src, size, file\_dst, MAX\_CHAR);   
 printf("输出字符数%d",size);   
 }else if(strcmp(argv[2], "-e") == 0){   
 size = RLEencode(file\_src, size, file\_dst, MAX\_CHAR);   
 printf("输出字符数%d",size);   
 }

3（1）解码

//RLE 解码算法，正好是编码算法的反过程   
int RLEdecode(unsigned char\* src,int src\_len,unsigned char\* dst,int dst\_len)   
{   
 unsigned char \*data = src;   
 int left = src\_len;   
 int point = 0;//输出缓冲区指针   
 while(data < src + src\_len)   
 {   
 unsigned char zifu = \*data++;   
 int count = zifu & 0x7f;   
 //请注意，一定要注意，这里是巨大bug易发生地，因为==的优先级其实比&要高，一定要加括号，一定要注意   
 if((zifu&0x80)==0x80)//是重复字符   
 {   
 for(int i=0;i<count;i++)   
 {   
 dst[point++] = \*data;   
 }   
 data++;   
 }else   
 {   
 for(int i=0;i<count;i++)   
 {   
 dst[point++] = \*data++;   
 }   
 }   
 }   
 return point;   
}

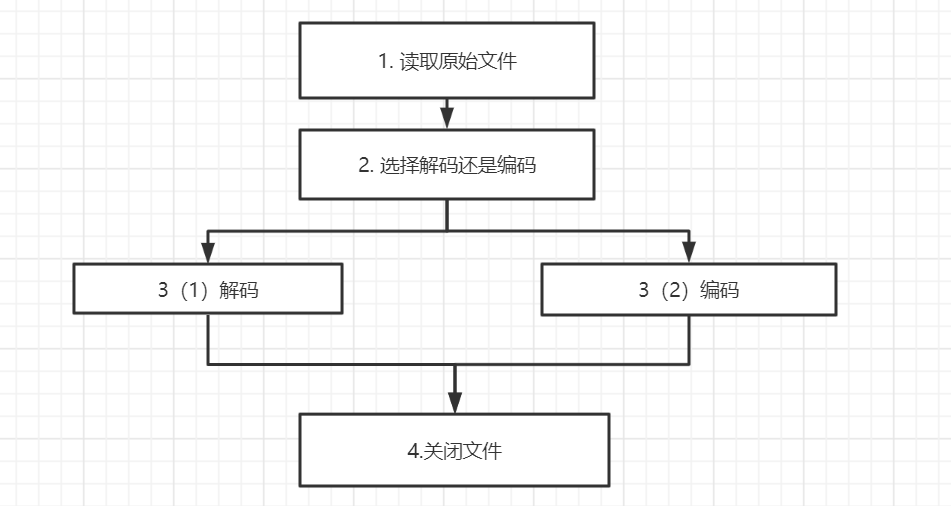
3（2）编码

//RLE算法，编码，输入参数为源数组，源数组长度，目标数组，目标数组长度   
int RLEencode(unsigned char\* src,int src\_len,unsigned char\* dst,int dst\_len)   
{   
 unsigned char \*data = src;   
 int left = src\_len;   
 int point = 0;//输出缓冲区指针   
 while(left > 0)   
 {   
 int count = 0;   
 if(Isrepeat(data,left))   
 {   
 count =getrepeatlength(data,left);   
 dst[point++] = count|0x80;//写重复数据的数据量   
 dst[point++] = \*data;//写字符   
 //指针移动   
 data+=count;   
 left-=count;   
 }   
 else{   
 count =getnorepeatlength(data,left);   
 dst[point++] = count;   
 for(int i=0;i<count;i++)   
 {   
 dst[point++] = \*data;   
 data++;   
 }   
 left-=count;   
 }   
 }   
 return point;   
}   
int Isrepeat(unsigned char\* src,int left)   
{   
 if(left<3)////判断是否是重复数据，注意，只有重复数据大于3的时候才被判定为i重复数据   
 {   
 return 0;   
 }   
 if(src[0] == src[1]&&src[1] == src[2])   
 {   
 return 1;   
 }   
 return 0;   
}   
//获得重复数据的长度   
int getrepeatlength(unsigned char\* src,int left)   
{   
 unsigned char data = src[0];   
 int len = 1;   
 while(len<left&&len<127&&src[len] == data)   
 {   
 len++;   
 }   
 return len;   
}   
//获得不重复数据的长度   
int getnorepeatlength(unsigned char\* src,int left)   
{   
 if(left < 3)   
 return left;   
 //不重复数据的长度至少为2   
 int len = 2;   
 //first和second代表前后两个字符，不断向后移动   
 unsigned char first = src[0];   
 unsigned char second = src[1];   
 while(len<left&&len<127)   
 {   
 if(!(first==second&&src[len]==second))   
 {   
 first = second;   
 second = src[len];   
 len++;   
 }   
 else   
 {   
 break;   
 }   
 }   
 return len;   
}

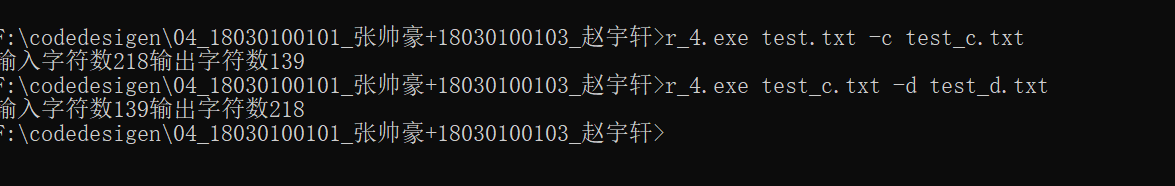
4.关闭文件

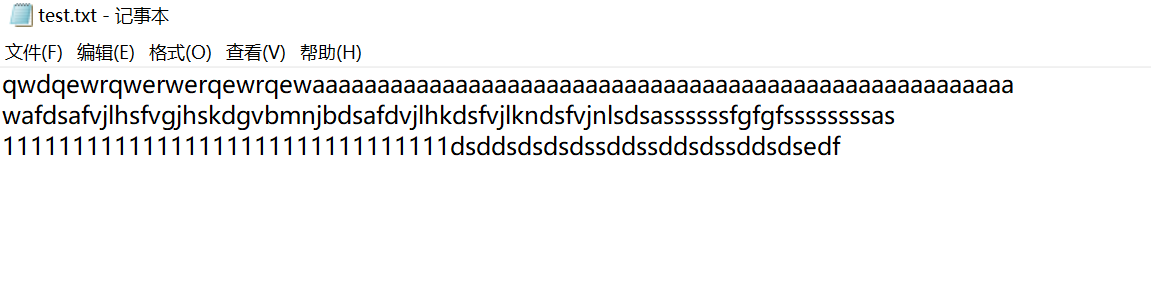
fwrite(file\_dst, size, sizeof(unsigned char), f2);   
 fclose(f1);   
 fclose(f2);

1. **流程图**



1. **测试与分析**





解压缩

