**实验报告**

题目：文件压缩

姓名：伊昕宇 PB16001749

1. **需求分析**
2. 实现任意二进制文件的压缩解压。
3. 将词频表保存到文件，压缩后解压所需全部信息从压缩的文件中得到。
4. 对于一般txt文档实现效果明显的压缩结果并正确解压，大文件非文本文件正确压缩和解压。
5. 利用huffman树实现，限定语言C/C++。
6. **概要设计**
7. 设定串的抽象数据类型定义：

**ADT** String {

**数据对象**：D = { ai | ai ∈ CharacterSet , i = 1 , 2 , ... , n >= 0 }

**数据关系**：R1 = { < ai-1 , ai > | ai-1 , ai ∈ D , i = 2 , ... , n }

**基本操作**：

strcpy ( String a , String b )

初始条件：字符串a、b存在，a的空间足够大。

操作结果：将b复制到a中，以a为起始位置。

strcat ( String a , String b )

初始条件：字符串a、b存在，a的空间足够大。

操作结果：将b连接到a后，自动添加 \0 结束。

strcmp ( String a , String b )

初始条件：字符串a、b存在。

操作结果：比较字符串的大小，若a大返回正值，相等返回0，b大返回 负值。

strlen ( String a )

初始条件：字符串a存在。

操作结果：返回字符串a的串长。

strrev ( String a )

初始条件：字符串a存在。

操作结果：原地逆置字符串a。

strncpy ( String a , String b , int n )

初始条件：字符串a、b存在，a的空间足够大。

操作结果：将字符串b中的n位复制到a中，以a为起始位置。

strtol ( String a , NULL , int base )

初始条件：字符串a可以转换成数字。Base合理

操作结果：将字符串a以base进制，转换为10进制数字并返回。

} **ADT** String ;

以上函数全部在string.h头文件中。

1. 设定Huffman树的抽象数据类型为：

**ADT** HTree {

**数据对象**：D = { ai | ai ∈ NumberSet , i = 1 , 2 , ... , n >= 0 }

**数据关系**：二叉树的关系

**基本操作**：

createHTree ( )

初始条件：charNum数组（字符出现次数表）已存在。

操作结果：根据现有的charNum次数生成HuffmanTree，0次不写

createHuffmanCode ( )

初始条件：Huffman树T已存在。

操作结果：生成每个字符的huffman编码，并保存在字符串数组thr中

} **ADT** HTree ;

遍历等操作在生成文件的函数中。

1. 设定文件的抽象数据类型为：

**ADT** FILE \* {

**数据对象**：二进制文件

**数据关系**：在同一文件中的关系

**基本操作**：

openFile ( String fileName , String mood = "rb+" )

初始条件：mood合理。

操作结果：尝试用mood的方式打开名为fileName的文件，失败则退出 程序，成功则返回这个文件。

getCharNum ( FILE \* &fp )

初始条件：fp是可读文件。

操作结果：统计文件中每个字符出现的次数，并保存数组charNum中。

writeCompFile ( FILE \* &fp1 )

初始条件：fp1是可读文件。

操作结果：创建一个fp1文件的压缩文件，返回压缩文件的大小（字节数）。

writeDiscompFile ( FILE \*&fp1 , char c , unsigned int end )

初始条件：fp1是可读压缩文件，c为补了几个0，end为文件本身内容的 结束位置。

操作结果：创建一个fp1文件的解压文件，返回解压文件的大小（字节数）。

comp ( FILE \* &fp1 )

初始条件：fp1是可读文件。

操作结果：压缩fp1，打印出压缩信息。

discomp ( FILE \* &fp1 )

初始条件：fp1是可读压缩文件。

操作结果：解压缩fp1，打印出解压缩信息。

} **ADT** FILE \* ;

还用到了fread , fseek , ftell , fopen , fprintf , fscanf , feof , fwrite , fclose 等函数，不在此说明了。

1. **详细设计**
2. **头文件、宏定义、全局变量以及数据结构定义**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#define MAXFILENAMELENGTH 1000 //文件名最大长度

#define CLEARSCREEN system("cls"); //清空屏幕

#define PAUSE system("pause");exit(0); //暂停并退出

typedef char \*String;

typedef struct{

unsigned char ch; //存放的8bit数

unsigned long weight,parent,lchild,rchild;

}HTNode;

typedef struct{

HTNode data[512]; //0不用

int root; //根

}HTree;

HTree T;

bool oneChar=false,emptyFile=false; //文件只有一种字符，huffman树无法生成

static String thr[256]; //每个字符的huffman编码

char fileName[MAXFILENAMELENGTH]; //文件名

unsigned long charNum[256]; //每个字符出现的次数

1. **打开文件和菜单打印**

将文件直接拖入运行框即可读入文件名，打开这个文件，打印菜单，选择压缩还是解压，调用相应函数最后关闭文件流，代码如下

char n;

printf("请拖放入文件\n");

gets(fileName); //读入文件名 不能超长

FILE \*fp1=openFile(fileName); //打开文件

CLEARSCREEN //清空屏幕

printf("1. 压缩\n2. 解压\n请输入序号");

loop:scanf("%d",&n);

switch(n){

case 1:CLEARSCREEN comp(fp1);break; //压缩

case 2:CLEARSCREEN discomp(fp1);break; //解压

default:printf("输入有误，请重新输入\n");goto loop;

}

fclose(fp1);

1. **统计文件字符数**

统计文件中每个字符（从 0000 0000 到 1111 1111 ，当成unsigned char，共0 - 255，256个字符）的出现次数，保存在charNum数组中。

此时读字符应用fread(&ch,1,1,fp);来实现。且在每个读操作之后应立即进行feof判断，如果结束（此时读失败）结束循环。

void getCharNum(FILE \*&fp){ //统计文件中每个字符出现的次数

memset(charNum,0,sizeof(charNum)); //清空

fseek(fp,0L,SEEK\_SET); //指针移到文件开头

fseek(fp,0L,SEEK\_CUR); //切换读写指针

unsigned char ch; //ch 0-255

while(1){

fread(&ch,1,1,fp); //读一字节内容

if(feof(fp))break; //文件结束

charNum[ch]++; //计数加1

}

}

1. **建立Huffman树**

通过charNum数组建立Huffman树，出现次数为0的字符不进行编码。此时有两个特殊情况：

1. 空文件。无法建立Huffman树，需要单独考虑
2. 单字符文件。文件中只有一种字符，Huffman树可以建立，但是编码时变成空码，需单独考虑。

int getSmallestWeight(int &a,int &b){ //获得有两个最小权重(>0)的字符在tree中的位置 返回0表示已经结束并给root赋值 否则返回结束位置

int i;

a=b=0;

for(i=1;i<512&&T.data[i].weight;i++) if(T.data[i].parent==0){ a=i; break;} //找到1个a

for(i++;i<512&&T.data[i].weight;i++) if(T.data[i].parent==0){ b=i; break;} //继续找到1个b

if(b==0){ //第一次true时找到a（root）但是没有b

T.root=a;

return 0; //只能找到1个（根）

}

for(i++;i<512&&T.data[i].weight;i++){

if(T.data[i].parent==0&&T.data[i].weight<T.data[a].weight)a=i; //parent!=0

else if(T.data[i].parent==0&&T.data[i].weight<T.data[b].weight)b=i;

} //获取weight最小的两个i

return i;

}

void createHTree(){ //根据现有的charNum次数生成HuffmanTree 0次不写

memset(T.data,0,sizeof(T.data)); //清空

int j=1,a,b,pos,nnum=0;

for(int i=0;i<=255;i++){

if(charNum[i]==0)continue; //此字符不出现

nnum++;

T.data[j].rchild=T.data[j].lchild=T.data[j].parent=0;

T.data[j].weight=charNum[i];

T.data[j++].ch=i; //j+1，下一次输入在下一个位置

}

if(nnum==0) { emptyFile=true; return;} //空文件

else if(nnum==1){ oneChar=true; return;} //只有一个字符

while(pos=getSmallestWeight(a,b)){ //能找到两个最小

T.data[pos].lchild=a;

T.data[pos].rchild=b;

T.data[pos].parent=0;

T.data[pos].weight=T.data[a].weight+T.data[b].weight;

T.data[a].parent=T.data[b].parent=pos;

}

}

1. **建立每个字符的Huffman码（解压操作不需要）**

为了压缩时不用每次都访问Huffman树，浪费时间，建立256个字符串的数组thr，只在这时访问一次Huffman树得到这个数组，以后的访问通过数组进行。

void createHuffmanCode(){ //生成每个字符的huffman编码

if(emptyFile==true||oneChar==true)return;

for(int i=1;!T.data[i].lchild;i++){ //遍历所有叶子

int j=i,e,p=0;

char temp[300]={0};

while(T.data[j].parent){ //没到根的时候

temp[p++]=(T.data[T.data[j].parent].rchild==j)+'0'; //右子树时为1 左子树为0

j=T.data[j].parent;

}

thr[T.data[i].ch]=(String) malloc(sizeof(char)\*(p+1));

strcpy(thr[T.data[i].ch],strrev(temp)); //原地逆置

}

}

1. **压缩操作**

unsigned int writeCompFile(FILE \*&fp1){ //生成压缩文件 返回文件大小

int l=strlen(fileName),oo;

while(--l&&fileName[l]!='.'); //找到最后一个点

char tt[30]={0},temp[300]={0},temp2[9]={0};

strcpy(tt+1,fileName+l+1); //保存后缀名

tt[0]=strlen(tt+1); //tt[0]保存字符串长

strcpy(fileName+l+1,"zip");

FILE \*fp2=openFile(fileName,"wb+"); //打开一个压缩文件

if(emptyFile==true){fclose(fp2); return 0;} //空文件

else if(oneChar==true){

unsigned char ch;

fread(&ch,1,1,fp1); //是哪个字符

fseek(fp1,0L,SEEK\_END); //移到开头

fprintf(fp2,"%c%d%c",0,ftell(fp1),ch); //存入 0 字符数 字符

fclose(fp2);

return 6; //大小为6

}

fwrite(tt,tt[0]+1,1,fp2); //存后缀名

fwrite(charNum,sizeof(charNum),1,fp2); //写入频数

fseek(fp1,0L,SEEK\_SET); //移到开头

fseek(fp1,0L,SEEK\_CUR); //切换读写指针

unsigned char ch; //存放二进制

while(!feof(fp1)){

fread(&ch,1,1,fp1); //读二进制

if(feof(fp1)){

oo=strlen(temp);

strcat(temp,"00000000"); //不足8位补0

}

else strcat(temp,thr[ch]); //连接字符串

l=strlen(temp);

while((l-=8)>0){ //多于8位

strncpy(temp2,temp,8); //复制8位出来

unsigned char tm=strtol(temp2,NULL,2);//给tm

fwrite(&tm,1,1,fp2); //写到压缩文件中

strcpy(temp,temp+8); //删除前8位

}

}

fprintf(fp2,"%c",8-oo); //存入 补了多少个0

unsigned int ll=ftell(fp2);

fclose(fp2);

return ll;

}

void comp(FILE \*&fp1){ //压缩文件fp1

clock\_t begin=clock(),end;

printf("请耐心等待...");

getCharNum(fp1); //统计文件中每个字符出现的次数

createHTree(); //生成HuffmanTree

createHuffmanCode(); //生成每个字符的huffman编码

unsigned int l=writeCompFile(fp1); //生成压缩文件 获得文件大小

CLEARSCREEN

end=clock();

printf("压缩成功！\n压缩前 文件大小%.1fkb\n压缩后 文件大小%.1fkb\n压缩率%3.1f%%\n",ftell(fp1)/1024.0,l/1024.0,emptyFile==true?100.0:(float)l/ftell(fp1)\*100.0);

printf("用时%.1fs\n压缩速率%.1fkb/s\n",(float)(end-begin)/CLOCKS\_PER\_SEC,ftell(fp1)/(float)(end-begin));

PAUSE

}

1. **解压操作**

unsigned int writeDiscompFile(FILE \*&fp1,char c,unsigned int end){ //end为结尾位置，补了c个0

FILE \*fp2=openFile(fileName,"wb+"); //打开一个压缩文件

unsigned char ch;

int k=8,p=T.root;

fread(&ch,1,1,fp1);

if(ftell(fp1)!=end){ //因为可能读完一个字节文件就已经结束 如果结束跳过if

while(1){

while(k&&T.data[p].rchild)p=((ch>>--k)&1?T.data[p].rchild:T.data[p].lchild);//找到叶子节点

if(!k){

fread(&ch,1,1,fp1);

k=8;

if(ftell(fp1)==end)break; //删掉字符串中结尾c个0

}

else {

fwrite(&(T.data[p].ch),1,1,fp2);

p=T.root;

}

}

} //处理完8位8位的字符 剩余一个不满8位的

do{

while(k>c&&T.data[p].rchild)p=((ch>>--k)&1?T.data[p].rchild:T.data[p].lchild);//找到叶子节点

fwrite(&(T.data[p].ch),1,1,fp2);

p=T.root;

}while(k>c); //写进最后几位

unsigned int ll=ftell(fp2);

fclose(fp2);

return ll;

}

void discomp(FILE \*&fp1){

char tt[30],cc,c;

unsigned int end;

if((c=fgetc(fp1))==EOF){ printf("这是一个空文件!\n"); fclose(fp1); PAUSE} //空文件

else if(c==0){ //单字符文件

FILE \*fp2=openFile(fileName,"wb+"); //打开一个压缩文件

fseek(fp1,0L,SEEK\_SET);

fseek(fp1,0L,SEEK\_CUR); //切换指针

fscanf(fp1,"%d%c",&end,&c); //读数量和字符

while(end--)fwrite(&c,1,1,fp2); //写字符

fclose(fp2);

fclose(fp1);

printf("解压成功！\n");

PAUSE

}

printf("请耐心等待...");

clock\_t begin=clock(),endd;

fseek(fp1,-1L,SEEK\_END); //读补了几个0

fseek(fp1,0L,SEEK\_CUR); //切换指针

end=ftell(fp1);

fscanf(fp1,"%c",&c); //几个0

fseek(fp1,0L,SEEK\_SET);

fseek(fp1,0L,SEEK\_CUR); //切换指针

fscanf(fp1,"%c",&cc); //后缀名位数

fread(tt,cc,1,fp1); //读后缀名

fread(charNum,sizeof(charNum),1,fp1); //读频数表

int l=strlen(fileName);

while(--l&&fileName[l]!='.'); //找到最后一个点

strcpy(fileName+l+1,tt);

createHTree(); //生成HuffmanTree

unsigned int ll=writeDiscompFile(fp1,c,end);

endd=clock();

CLEARSCREEN

printf("解压成功！\n用时%.1fs\n解压速率%.1fkb/s\n",(float)(endd-begin)/CLOCKS\_PER\_SEC,ll/(float)(endd-begin));

PAUSE

}

1. **调试分析**
2. 正如上面说的，空文件压缩会失败，Huffman树无法生成，需要单独考虑，而单字符文件生成的Huffman树只有一个根节点，生成编码时生成空串，可以压缩（全是空串）但是无法解压，也进行了单独考虑。
3. 由于进行了补0操作，而只有在写到文件末尾时才知道补几个0，于是在压缩文件的末尾写了一个字符表示补了几个0（0~7），解压时先移动到文件末尾-1的位置，设置标志，这里是结尾，再读了这个字符即可。文件会长一个字节。
4. 由于Huffman树太大，选择将频数表存在文件中。256个字符，每个字符的频数大小sizeof ( unsigned long ) == 4，即占用空间4 x 256 =1 kb，所以压缩多字符小文件时压缩率会变得很大。
5. 为了解压时能自动生成确定文件名的文件，考虑要把后缀名存到文件中。则需要现存一个后缀名的位数，在保存整个后缀名。所以：

多字符压缩文件的形式是：

**后缀名位数（多字符文件标志） + 后缀名 + 频数表( 1 kb ) + 压缩内容 + 补0的个数**

**总长（1kb + 压缩后大小）**

单字符压缩文件的形式是：

**0（单字符文件标志，1b）+ 字符出现次数（4b） + 字符内容 （1b）**

**总长 6 byte**

空文件压缩形式是：

**空文件（所以不知道后缀名是什么，不可以解压） 总长0 B**

所以压缩的文件必须是**有后缀名的文件。**

1. fclose必须写。feof返回真的位置不是结束位置，而是读失败的之后的位置。需要用wb+，rb+的方式打开。如果**文件名有中文路径名有时会出错**，如果**文件名中没有后缀名一定会出错**。如果不能拖动可以直接手动输入文件名。
2. **用户手册**
3. 请运行exe文件后，将文件直接拖入黑框，会自动输入文件名，最好不要有中文等特殊字符，确保有后缀（有 . xxxxx 即可）。之后输入1压缩2解压。请耐心等待。
4. **测试结果**
5. 小txt，空文件txt和单字符txt。

正确压缩解压。压缩大小1k（单字符6b，空文件0）。能正确处理文字。内容不变。

1. 对2Mb以上的 doc 文件

英文压缩率在38%左右，有明显压缩效果，中文也可以正确压缩解压，内容不变。

1. 对4Mb的bmp 图片

压缩前 文件大小4218.8kb

压缩后 文件大小3821.9kb

压缩率90.6%

压缩用时2.0s

压缩速率2146.1kb/s

解压用时2.7s

解压速率1603.6kb/s

解压后的图片能正确打开

1. 对100Mb的mp4视频

压缩前 文件大小87852.0kb

压缩后 文件大小85499.1kb

压缩率97.3%

压缩用时22.6s

压缩速率3981.1kb/s

解压用时60.8s

解压速率1479.5kb/s

能正确压缩解压并播放。