

杭州电子科技大学

HANGZHOU DIANZI UNIVERSITY

数据结构课程设计报告

(2021-2022-2 学期)



题目	赫夫曼编码译码器			
学院	理学院			
专业	信息与计算科学			
组号	第十六组	学号	姓名	分工
		20071226	童繁	流程图
		20071227	王瀚功	数据
		20071228	王赛豪	文案
		20071229	吴政豪	调试
		20071230	武琦	代码
时间	2022 年 5 月 2 日			

目录

1. 需求分析.....	1
2. 项目亮点.....	1
3. 概要设计.....	1
4. 详细设计.....	3
4.1 定义.....	3
4.2 重要函数.....	5
4.2.1 字符串转比特流.....	5
4.2.2 比特流转字符串.....	6
5. 用户手册.....	7
5.1 界面.....	7
5.2 初始化.....	8
5.3 编码.....	10
5.4 译码.....	11
5.5 打印代码文件.....	11
5.6 打印赫夫曼树.....	12
6. 心得体会.....	14
7. 附录	15
7.1 definition.h	15
7.2 main.c	16
7.3 huffman.c	17
7.4 function.c.....	20

1. 需求分析

- (1) 初始化: 从终端读入字符集大小 n , 以及 n 个字符和 n 个权值, 建立哈夫曼树, 并将它存于文件 `hfmTree` 中。
- (2) 编码: 利用已建好的哈夫曼树(如不在内容, 则从文件 `hfmTree` 中读入), 对文件 `ToBeTran` 中的正文进行编码, 然后将结果存入文件 `CodeFile` 中。
- (3) 译码: 利用已建好的哈夫曼树将文件 `CodeFile` 中的代码进行译码, 结果存入文件 `TextFile` 中。
- (4) 打印代码文件: 将文件 `CodeFile` 以紧凑格式显示在终端上, 每行 50 个代码。同时将此字符形式的编码文件写入文件 `CodePrint` 中。
- (5) 打印哈夫曼树: 将已在内存中的哈夫曼树以树的方式显示在终端上, 同时将此字符形式的哈夫曼树写入文件 `TreePrint` 中。
- (6) 编码文件 `CodeFile` 中的每个 0 和 1 实际上占了一个字节的空间, 为了最大限度利用码点存储能力, 将编码结果以二进制形式存放在文件 `CodeFile` 中。
- (7) 实现各个转换操作的源/目的文件, 均可由用户自己选择指定。

2. 项目亮点

- (1) 使用位操作将代码字符串转化为比特流保存, 前 4 个字节存储位数, 后面的字节存储内容, 极大节省了存储空间;
- (2) 用户可以自己指定文件进行所有操作, 包括不同赫夫曼树的读取和保存;
- (3) 编码时小写字母转换为大写;
- (4) 译码后可以选择在终端打印译码结果;
- (5) 建立了功能库和结构库实现相关操作。

3. 概要设计

首先建立结构库和功能库存储相关函数, 然后建立需要的数据结构类型: 赫夫曼树节点结构 `HuffmanTree`、赫夫曼编码表结构 `HuffmanCode`、代码字符串 `huffmanCode`、译文字符串 `txtCode`、文件名称字符串 `name` 以及字符集 `character` 和权重数组 `weight`。

程序运行时首先让用户读取需要的赫夫曼树, 然后建立赫夫曼树和编码, 用户选择相应功能进行操作, 在每次操作结束后页面回到主界面。

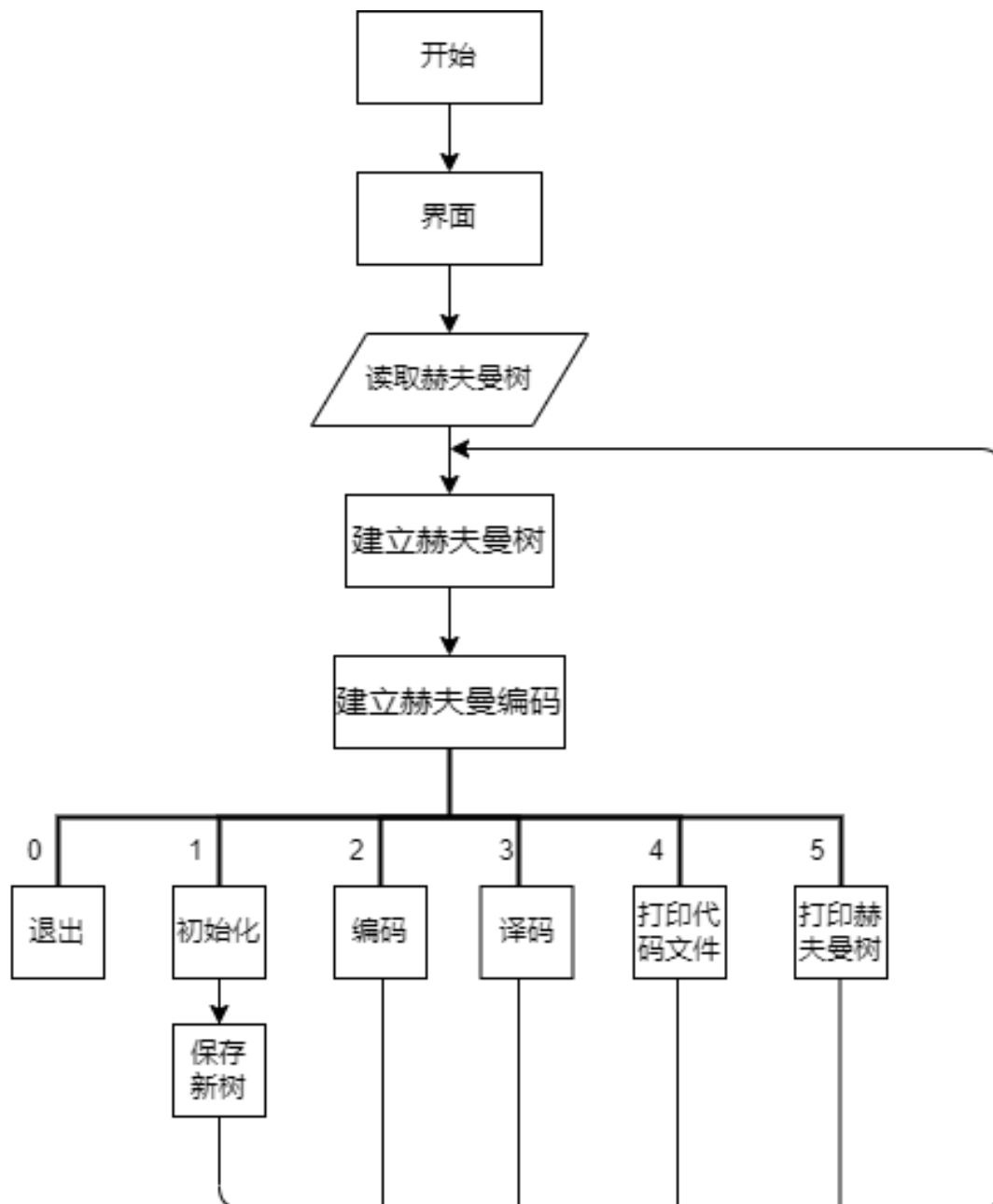


图 1: 主函数流程图

4. 详细设计

4.1 定义

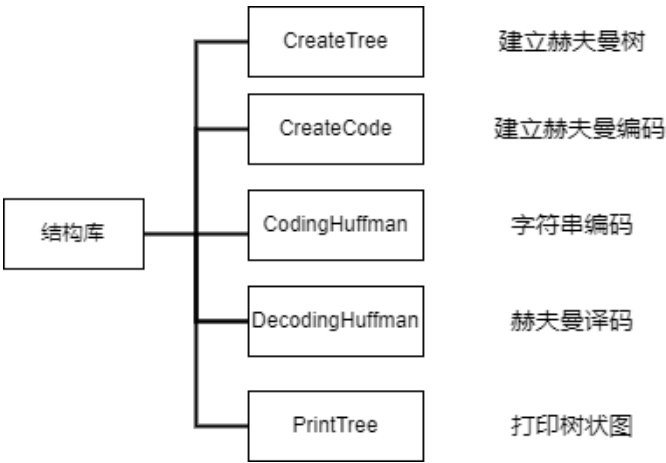


图 2: 结构库

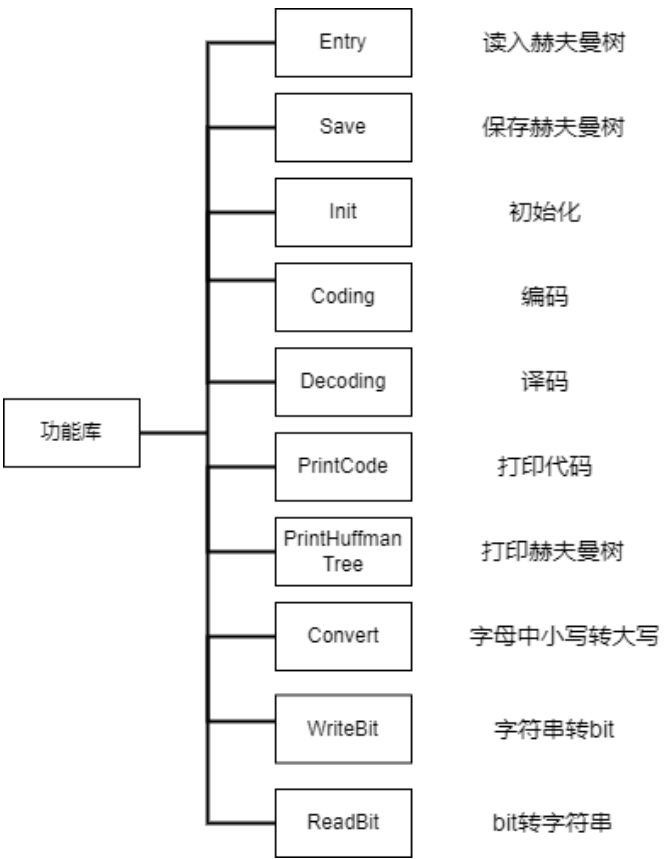


图 3: 功能库

```
1 #include<stdio.h>
```

```
2  #include<string.h>
3  #include<stdlib.h>
4
5  //枚举类型返回值
6  typedef enum status
7  {
8      OK=1,
9      ERROR=0,
10 }Status;
11 //赫夫曼树节点存储结构
12 typedef struct tnode
13 {
14     char character;//字符
15     int weight;//权重
16     int parent,lchild,rchild;
17 }HuffmanTree;
18 //赫夫曼编码表结构
19 typedef struct cnode
20 {
21     int start;
22     char bits[1000];
23     char character;
24 }HuffmanCode;
25 //定义相关变量
26 char huffmanCode[1000];//代码字符串
27 char txtCode[1000];//译文字符串
28 char name[100];//文件名字符串
29 FILE *fp;//文件指针
30 char character[]={'\0',' ','A','B','C','D','E','F','G','H','I','J','K',
31                 'L','M','N','O','P','Q','R','S','T','U','V','W','X',
32                 'Y','Z','.',',','?',':',';','!'};//字符集
33 int weight[]={0};//权重
34 //结构库函数
35 Status CreateTree(HuffmanTree T[],int leafNum);//建立赫夫曼树
36 Status CreateCode(HuffmanCode code[],HuffmanTree T[],int leafNum);//建立赫夫曼编码
37 Status CodingHuffman(HuffmanCode code[],HuffmanTree T[],char enter[],int
    leafNum);//字符串编码
38 Status DecodingHuffman(HuffmanCode code[],HuffmanTree T[],char s[],int
    leafNum);//赫夫曼译码
39 Status PrintTree(HuffmanTree T[],int root,int type,int level,char
    filename[]);//打印树状图
40 //功能库函数
41 Status Entry();//读入赫夫曼树
42 Status Save();//保存赫夫曼树
43 Status Init();//初始化
44 Status Coding(HuffmanCode code[],HuffmanTree T[],int leafNum);//编码
45 Status Decoding(HuffmanCode code[],HuffmanTree T[],int leafNum);//译码
```

```
46 Status PrintCode();//打印代码
47 Status PrintHuffmanTree(HuffmanTree T[],int leafNum);//打印赫夫曼树
48 Status Convert(char s[]);//字母中小写转大写
49 Status WriteBit(char code[]);//字符串转bit
50 Status ReadBit();//bit转字符串
```

4.2 重要函数

4.2.1 字符串转比特流

Listing 1: WriteBit

```
1 Status WriteBit(char
    code[])//将字符串转化为bit:开头4个字节保存了该文件存储的位数,后面的字节为存储内容
2 {
3     char *p;
4     int i,j=-1,count,num,left;
5     printf("请输入代码存储文件名:");
6     fflush(stdin);gets(name);strcat(name,".txt");
7     fp=fopen(name,"wb");
8     count=strlen(code);//字符串个数
9     num=count/8;//存储字符串需要的字节数
10    left=count%8;//字符串剩余不足8位的个数
11    if(left==0)
12    {
13        p=(char*)malloc(sizeof(char)*num);
14        memset(p,0,num);
15    }
16    else
17    {
18        p=(char*)malloc(sizeof(char)*(num+1));
19        memset(p,0,num+1);
20    }
21    for(i=0;i<count;i++)//位运算,每8个字符以2进制的形式储存在一个字符中
22    {
23        if(i%8==0) j++;
24        p[j]<<=1;
25        code[i]-='0';
26        p[j]|=code[i];
27    }
28    if(left!=0)//如果left不为0,需要把剩余的几位向左边靠拢
29    {
30        p[j]<<=8-left;
31        fwrite(&count,sizeof(count),1,fp);
32        fwrite(p,1,num+1,fp);
```

```
33     }
34     else
35     {
36         fwrite(&count,sizeof(count),1,fp);
37         fwrite(p,1,num,fp);
38     }
39     fclose(fp);
40 }
```

4.2.2 比特流转字符串

Listing 2: ReadBit

```
1 Status ReadBit()//bit转字符串
2 {
3     strcpy(huffmanCode,"");
4     char *p;
5     int i,j=-1,count,num,left,t=0;
6     unsigned char flag=128; //即0b1000000,用于做位运算,注意要用无符号的字符型
7     printf("请输入代码存储文件名:");
8     fflush(stdin);gets(name);strcat(name,".txt");
9     if((fp=fopen(name,"rb"))==NULL)
10    {
11        printf("该文件夹下无%s!\n",name);
12        return ERROR;
13    }
14    fread(&count,sizeof(count),1,fp);
15    num=count/8;
16    left=count%8;
17    if(left==0)
18    {
19        p=(char*)malloc(sizeof(char)*num);
20        fread(p,1,num,fp);
21    }
22    else
23    {
24        p=(char*)malloc(sizeof(char)*(num+1));
25        fread(p,1,num+1,fp);
26    }
27    fclose(fp);
28    for(i=0;i<count;i++)
29    {
30        if(i%8==0)
31        {
32            j++;
```



```
33     flag=128;
34 }
35 if((p[j]&flag)) huffmanCode[t]='1';
36 else huffmanCode[t]='0';
37 t++;
38 flag/=2;
39 }
40 return OK;
41 }
```

5. 用户手册

5.1 界面



图 4: 用户界面

Listing 3: hfmTree.txt

```
1 size:27
2 字符 权重
3 186
```

```
4 A 64
5 B 13
6 C 22
7 D 32
8 E 103
9 F 21
10 G 15
11 H 47
12 I 57
13 J 1
14 K 5
15 L 32
16 M 20
17 N 57
18 O 63
19 P 15
20 Q 1
21 R 48
22 S 51
23 T 80
24 U 23
25 V 8
26 W 18
27 X 1
28 Y 16
29 Z 1
```

5.2 初始化

在原有树的基础上进行修改和添加字符集操作。

```
请输入序号:1
请输入字符集数:5
请输入第1个字符:.\
请输入'.'的权重:80
请输入第2个字符:?
请输入'?'的权重:20
请输入第3个字符:!\
请输入'!'的权重:15
请输入第4个字符:;\
请输入';'的权重:18
请输入第5个字符:.,
请输入','的权重:30
初始化成功!
```

图 5: 初始化字符集

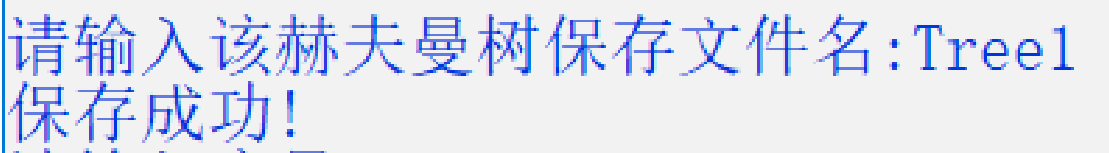


图 6: 保存新的赫夫曼树

Listing 4: Tree1.txt

```
1 size:34
2
3 186
4 A 64
5 B 13
6 C 22
7 D 32
8 E 103
9 F 21
10 G 15
11 H 47
12 I 57
13 J 1
14 K 5
15 L 32
16 M 20
17 N 57
18 O 63
19 P 15
20 Q 1
21 R 48
22 S 51
23 T 80
24 U 23
25 V 8
26 W 18
27 X 1
28 Y 16
29 Z 1
30 , 30
31 . 80
32 ? 20
33 : 0
34 ; 18
35 ! 0
```

5.3 编码

```
请输入序号:2  
请输入编码文件名:ToBeTran  
请输入代码存储文件名:CodeFile  
编码成功!
```

图 7: 编码

Listing 5: ToBeTran.txt

```
1 THIS PROGRAM IS MY FAVORITE
```

CodeFile 仅占 19 字节。

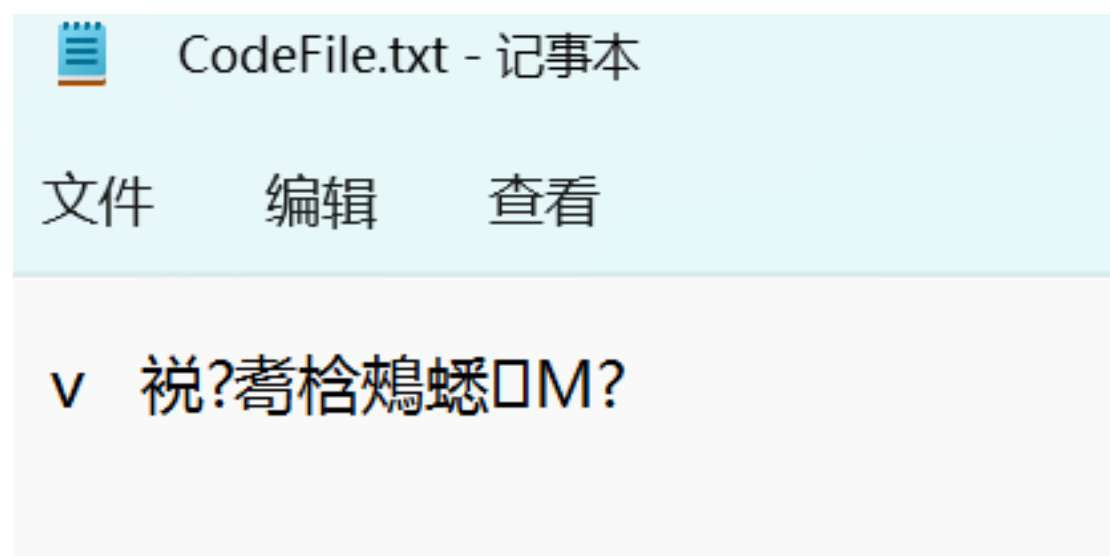


图 8: 代码

5.4 译码

```
请输入序号:3
请输入代码存储文件名:CodeFile
请输入译码结果存储文件名:TextFile
译码成功, 是否打印结果(y|n):y
THIS PROGRAM IS MY FAVORITE
```

图 9: 译码

Listing 6: TextFile.txt

```
1 THIS PROGRAM IS MY FAVORITE
```

5.5 打印代码文件

```
请输入序号:4
请输入代码存储文件名:CodeFile
请输入打印代码文件名:CodePrint
11010001011000111111000100010100110000100101010110
01011101100011111110010100011111110011101011000001
001001001101101010
结果存入CodePrint.txt文件成功!
```

图 10: 打印代码文件

未进行压缩的代码占 122 字节（包含换行符）。

Listing 7: CodePrint.txt

```
1 11010001011000111111000100010100110000100101010110
2 01011101100011111110010100011111110011101011000001
3 001001001101101010
```

5.6 打印赫夫曼树

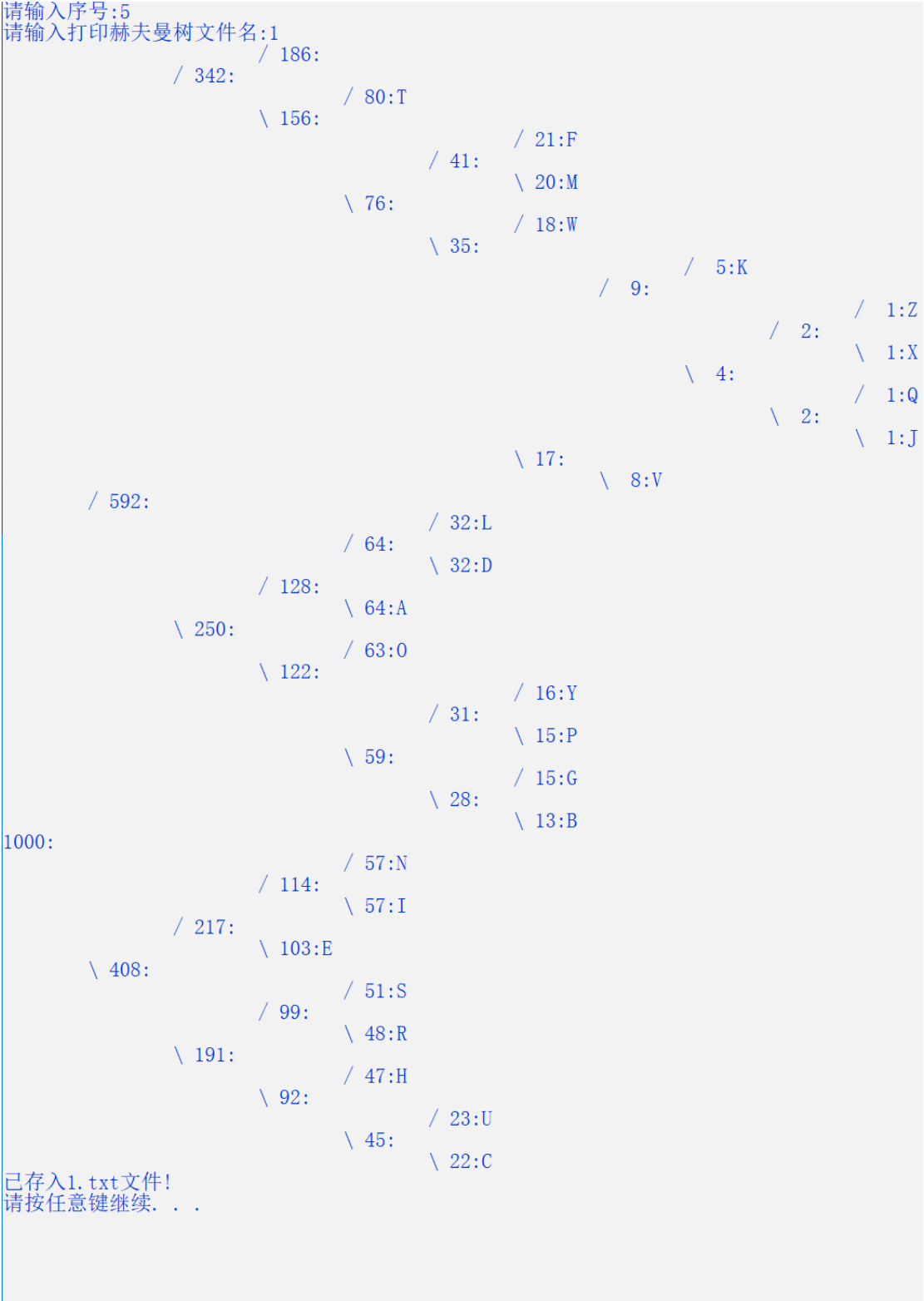


图 11: 打印赫夫曼树

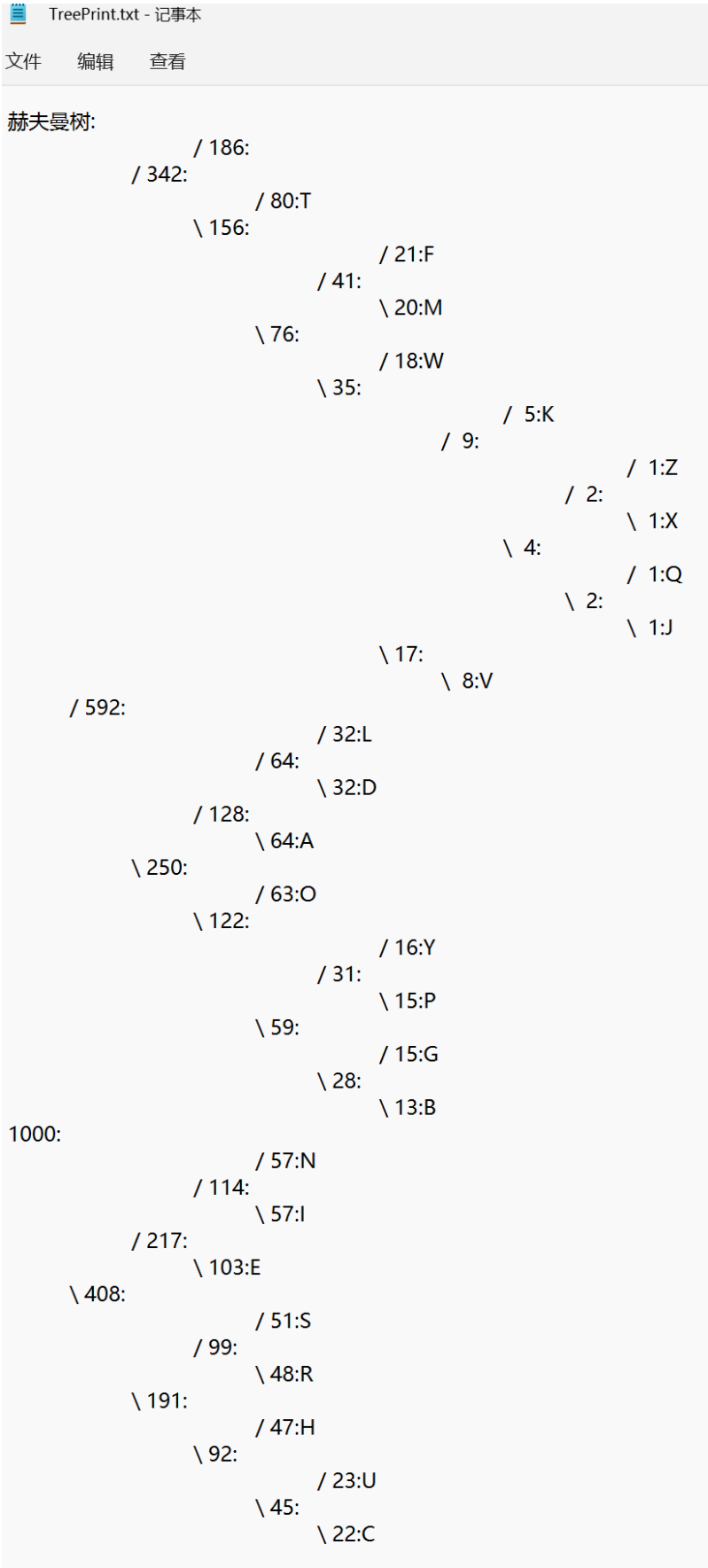


图 12: TreePrint.txt

6. 心得体会

这次课程设计的心得体会通过实践我们的收获如下：

1. 在这次的赫夫曼编码译码的过程中，我们更深刻地了解了赫夫曼树的特点与用法。
2. 在做一个较大的程序过程中，应该学会边编写程序边运行，即完成了一个功能，也要对其调试，这样有利于我们高效地完成项目，并在调试 BUG 的过程可以大大减小难度。
3. 必须要有良好的编程习惯。首先编码风格要统一规范，这样不仅有利于代码的阅读，更有利于代码的维护。其次在一些代码方面要细心谨慎，减少 BUG 出现的机率。
4. 更加系统地学习了 C 语言的二进制位操作，对文件函数的运用更加熟练。

7. 附录

7.1 definition.h

```

1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4
5  //枚举类型返回值
6  typedef enum status
7  {
8      OK=1,
9      ERROR=0,
10 }Status;
11 //赫夫曼树节点存储结构
12 typedef struct tnode
13 {
14     char character;//字符
15     int weight;//权重
16     int parent,lchild,rchild;
17 }HuffmanTree;
18 //赫夫曼编码表结构
19 typedef struct cnode
20 {
21     int start;
22     char bits[1000];
23     char character;
24 }HuffmanCode;
25 //定义相关变量
26 char huffmanCode[1000];//代码字符串
27 char txtCode[1000];//译文字符串
28 char name[100];//文件名字符串
29 FILE *fp;//文件指针
30 char character[]={'\0',' ','A','B','C','D','E','F','G','H','I','J','K',
31                 'L','M','N','O','P','Q','R','S','T','U','V','W','X',
32                 'Y','Z','.',',','?',':',';','!'};//字符集
33 int weight[]={0};//权重
34 //结构库函数
35 Status CreateTree(HuffmanTree T[],int leafNum);//建立赫夫曼树
36 Status CreateCode(HuffmanCode code[],HuffmanTree T[],int leafNum);//建立赫夫曼编码
37 Status CodingHuffman(HuffmanCode code[],HuffmanTree T[],char enter[],int
    leafNum);//字符串编码
38 Status DecodingHuffman(HuffmanCode code[],HuffmanTree T[],char s[],int
    leafNum);//赫夫曼译码
39 Status PrintTree(HuffmanTree T[],int root,int type,int level,char
    filename[]);//打印树状图

```

```
40 //功能库函数
41 Status Entry();//读入赫夫曼树
42 Status Save();//保存赫夫曼树
43 Status Init();//初始化
44 Status Coding(HuffmanCode code[],HuffmanTree T[],int leafNum);//编码
45 Status Decoding(HuffmanCode code[],HuffmanTree T[],int leafNum);//译码
46 Status PrintCode();//打印代码
47 Status PrintHuffmanTree(HuffmanTree T[],int leafNum);//打印赫夫曼树
48 Status Convert(char s[]);//字母中小写转大写
49 Status WriteBit(char code[]);//字符串转bit
50 Status ReadBit();//bit转字符串
```

7.2 main.c

```

1  #include "definition.h"
2  #include "huffman.c"
3  #include "function.c"
4  Status main()
5  {
6      int leafNum,choice,flag=1;
7      A:system("cls");
8      system("color F1");
9      printf(" 赫夫曼编码译码器\n");
10     printf("-----\n");
11     printf(" 1---初始化\n 2---编码\n 3---译码\n 4---打印代码文件\n 5---打印赫夫曼树\n\n\n 0--退出系统\n");
12     printf("-----\n");
13     if(flag==1) Entry();
14     leafNum=sizeof(character)-1;
15     HuffmanCode Code[leafNum+1];
16     HuffmanTree Tree[2*leafNum+1];
17     CreateTree(Tree,leafNum);CreateCode(Code,Tree,leafNum);
18     if(flag==2) Save(leafNum);
19     flag=0;
20     printf("请输入序号:");
21     scanf("%d",&choice);
22     switch(choice)
23     {
24         case 1:Init();flag=2;system("pause");goto A;
25         case 2:Coding(Code,Tree,leafNum);system("pause");goto A;
26         case 3:Decoding(Code,Tree,leafNum);system("pause");goto A;
27         case 4:PrintCode();system("pause");goto A;
28         case 5:PrintHuffmanTree(Tree,leafNum);system("pause");goto A;
29         case 0:break;
30         default:goto A;

```

```
31     }
32     system("pause");
33     return OK;
34 }
```

7.3 huffman.c

```
1  Status CreateTree(HuffmanTree T[],int leafNum)//建立赫夫曼树
2  {
3      int huffmanNum=2*leafNum;
4      int i,j,p1,p2,least1,least2;
5      for(i=1;i<=huffmanNum;i++)//初始化赫夫曼树
6      {
7          T[i].character='\0';
8          T[i].parent=0;
9          T[i].lchild=0;
10         T[i].rchild=0;
11         T[i].weight=0;
12     }
13     for(i=1;i<=leafNum;i++)//录入字符和频度
14     {
15         T[i].character=character[i];
16         T[i].weight=weight[i];
17     }
18     for(i=leafNum+1;i<=huffmanNum;i++)
19     {
20         p1=0;p2=0;least1=least2=10000;
21         for(j=1;j<i;j++)
22         {
23             if(T[j].parent==0)
24                 if(T[j].weight<least1)
25                 {
26                     least2=least1;
27                     least1=T[j].weight;
28                     p2=p1;
29                     p1=j;
30                 }
31             else
32             {
33                 if(T[j].weight<least2)
34                 {
35                     least2=T[j].weight;
36                     p2=j;
37                 }
38             }

```

```

39     }
40     T[p1].parent=i;
41     T[p2].parent=i;
42     T[i].lchild=p1;
43     T[i].rchild=p2;
44     T[i].weight=T[p1].weight+T[p2].weight;
45 }
46 T[huffmanNum-1].parent=0;
47 }
48
49 Status CreateCode(HuffmanCode code[],HuffmanTree T[],int leafNum)//建立赫夫曼编码
50 {
51     int i,child,parent;
52     HuffmanCode buffer;
53     for(i=1;i<=leafNum;i++)
54     {
55         buffer.character=character[i];
56         buffer.start=leafNum;
57         child=i;
58         parent=T[i].parent;
59         while(parent!=0)
60         {
61             buffer.start--;
62             if(T[parent].lchild==child)
63                 buffer.bits[buffer.start]='0';
64             else buffer.bits[buffer.start]='1';
65             child=parent;
66             parent=T[parent].parent;
67         }
68         code[i]=buffer;
69     }
70 }
71
72 Status CodingHuffman(HuffmanCode code[],HuffmanTree T[],char enter[],int
    leafNum)//字符串编码
73 {
74     strcpy(huffmanCode,"");
75     int i,j,k,n=0;
76     for(i=0;i<strlen(enter);i++)
77     {
78         for(j=1;j<=leafNum;j++)
79         {
80             if(enter[i]==T[j].character)
81             {
82                 for(k=code[j].start;k<leafNum;k++)
83                 {
84                     huffmanCode[n]=code[j].bits[k];

```

```

85         n++;
86     }
87 }
88 }
89 }
90 }
91
92 Status DecodingHuffman(HuffmanCode code[], HuffmanTree T[], char s[], int
    leafNum) // 赫夫曼译码
93 {
94     int huffmanNum = 2 * leafNum;
95     strcpy(txtCode, "");
96     int i = huffmanNum - 1, t = 0;
97     char *q = NULL;
98     q = s;
99     while(*q != '\0')
100     {
101         if(*q == '0') i = T[i].lchild;
102         if(*q == '1') i = T[i].rchild;
103         if((T[i].lchild == 0) && (T[i].rchild == 0))
104         {
105             txtCode[t] = code[i].character;
106             t++;
107             i = huffmanNum - 1;
108         }
109         q++;
110     }
111 }
112
113 Status PrintTree(HuffmanTree T[], int root, int type, int level, char
    filename[]) // 打印树状图
114 {
115     int i;
116     FILE *print = fopen(filename, "a+");
117     if(root == 0)
118     {
119         fclose(print);
120         return OK;
121     }
122     PrintTree(T, T[root].rchild, 2, level + 1, filename);
123     switch(type)
124     {
125     case 0:
126         printf("%2d:%c\n", T[root].weight, T[root].character);
127         fprintf(print, "%2d:%c\n", T[root].weight, T[root].character);
128         break;
129     case 1:

```

```

130     for(i=0;i<level;i++)
131     {
132         printf("\t");
133         fputc('\t',print);
134     }
135     printf("\ \ %2d:%c\n",T[root].weight,T[root].character);
136     fprintf(print,"\ \ %2d:%c\n",T[root].weight,T[root].character);
137     break;
138 case 2:
139     for(i=0;i<level;i++)
140     {
141         printf("\t");
142         fputc('\t',print);
143     }
144     printf("/ \ %2d:%c\n",T[root].weight,T[root].character);
145     fprintf(print,"/ \ %2d:%c\n",T[root].weight,T[root].character);
146     break;
147 default:break;
148 }
149 fclose(print);
150 PrintTree(T,T[root].lchild,1,level+1,filename);
151 }

```

7.4 function.c

```

1 Status Entry()//读入赫夫曼树
2 {
3     int i=1,j=1,k,m;
4     char buffer[1024],*p;
5     A:printf("请输入赫夫曼树读取文件名:");
6     fflush(stdin);gets(name);strcat(name,".txt");
7     if((fp=fopen(name,"r"))==NULL)
8     {
9         printf("该文件夹下不存在%s文件!\n",name);
10        goto A;
11    }
12    while((fgets(buffer,1024,fp))!=NULL)
13    {
14        if(i>2)
15        {
16            character[j]=buffer[0];
17            m=0;
18            for(k=0;buffer[k]!='\0';k++)
19            {
20                if(buffer[k]>='0'&&buffer[k]<='9')

```

```

21         {
22             p[m]=buffer[k];
23             m++;
24         }
25     }
26     weight[j]=atoi(p);
27     memset(p,0,sizeof(p));
28     j++;
29 }
30 i++;
31 }
32 fclose(fp);
33 printf("赫夫曼树建立成功!\n");
34 }
35
36 Status Save()//保存赫夫曼树
37 {
38     printf("请输入该赫夫曼树保存文件名:");
39     fflush(stdin);gets(name);strcat(name,".txt");
40     fp=fopen(name,"w");
41     fprintf(fp,"size:%d\n",sizeof(character));
42     fprintf(fp,"字符\t权重\n");
43     for(int i=1;i<sizeof(character);i++)
44     {
45         fprintf(fp,"%c\t%d\n",character[i],weight[i]);
46     }
47     printf("保存成功!\n");
48     fclose(fp);
49 }
50
51 Status Init()//初始化
52 {
53     int i,j,n,ret,w,flag=0;
54     char c;
55     A:printf("请输入字符集数:");
56     ret=scanf("%d",&n);
57     if(n<0||ret==0) goto A;
58     for(i=0;i<n;i++)
59     {
60         printf("请输入第%d个字符:",i+1);
61         fflush(stdin);scanf("%c",&c);
62         B:printf("请输入\'%c\'的权重:",c);
63         ret=scanf("%d",&w);
64         if(w<0||ret==0) goto B;
65         if(w>0)
66         {
67             for(j=0;j<sizeof(character)-1;j++)

```

```

68     {
69         if(c==character[j])
70         {
71             weight[j]=w;
72             flag=1;
73             break;
74         }
75     }
76     if(!flag)
77     {
78         character[j]=c;
79         weight[j]=w;
80     }
81 }
82 }
83 printf("初始化成功!\n");
84 }
85
86 Status Coding(HuffmanCode code[],HuffmanTree T[],int leafNum)//编码
87 {
88     int i;
89     char enter[1000];
90     printf("请输入编码文件名:");
91     fflush(stdin);gets(name);strcat(name, ".txt");
92     if((fp=fopen(name, "r"))==NULL)
93     {
94         printf("该文件夹下无%s!\n",name);
95         return ERROR;
96     }
97     fgets(enter,1000,fp);
98     fclose(fp);
99     Convert(enter);
100    CodingHuffman(code,T,enter,leafNum);
101    WriteBit(huffmanCode);
102    printf("编码成功!\n");
103 }
104
105 Status Decoding(HuffmanCode code[],HuffmanTree T[],int leafNum)//译码
106 {
107     char choice;
108     if(ReadBit())
109     {
110         DecodingHuffman(code,T,huffmanCode,leafNum);
111         printf("请输入译码结果存储文件名:");
112         fflush(stdin);gets(name);strcat(name, ".txt");
113         fp=fopen(name, "w");
114         fputs(txtCode,fp);

```



```

115     fclose(fp);
116     printf("译码成功,");
117     A:printf("是否打印结果(y|n):");
118     fflush(stdin);scanf("%c",&choice);
119     if(choice=='y') printf("%s\n",txtCode);
120     else if(choice=='n') return OK;
121     else goto A;
122 }
123 return OK;
124 }
125
126 Status PrintCode()//打印代码
127 {
128     if(ReadBit())
129     {
130         printf("请输入打印代码文件名:");
131         fflush(stdin);gets(name);strcat(name, ".txt");
132         fp=fopen(name, "w");
133         for(int i=0;i<strlen(huffmanCode);i++)
134         {
135             printf("%c",huffmanCode[i]);
136             fprintf(fp, "%c",huffmanCode[i]);
137             if((i+1)%50==0)
138             {
139                 printf("\n");
140                 fprintf(fp, "\n");
141             }
142         }
143         fclose(fp);
144         printf("\n结果存入%s文件成功!\n",name);
145     }
146 }
147
148 Status PrintHuffmanTree(HuffmanTree T[],int leafNum)//打印赫夫曼树
149 {
150     printf("请输入打印赫夫曼树文件名:");
151     fflush(stdin);gets(name);strcat(name, ".txt");
152     fp=fopen(name, "w");
153     fprintf(fp, "赫夫曼树:\n");
154     fclose(fp);
155     PrintTree(T,2*leafNum-1,0,0,name);
156     printf("已存入%s文件!\n",name);
157 }
158
159 Status Convert(char s[])//字母中小写转大写
160 {
161     char *p;

```

```

162     p=s;
163     while(*p!='\0')
164     {
165         if(*p>='a'&&*p<='z') *p-=32;
166         p++;
167     }
168 }
169
170 Status WriteBit(char
    code[])//将字符串转化为bit:开头4个字节保存了该文件存储的位数,后面的字节为存储内容
171 {
172     char *p;
173     int i,j=-1,count,num,left;
174     printf("请输入代码存储文件名:");
175     fflush(stdin);gets(name);strcat(name,".txt");
176     fp=fopen(name,"wb");
177     count=strlen(code);//字符串个数
178     num=count/8;//存储字符需要的字节数
179     left=count%8;//字符串剩余不足8位的个数
180     if(left==0)
181     {
182         p=(char*)malloc(sizeof(char)*num);
183         memset(p,0,num);
184     }
185     else
186     {
187         p=(char*)malloc(sizeof(char)*(num+1));
188         memset(p, 0, num + 1);
189     }
190     for(i=0;i<count;i++)//位运算,每8个字符以2进制的形式储存在一个字符中
191     {
192         if(i%8==0) j++;
193         p[j]<=1;
194         code[i]!='0';
195         p[j]|=code[i];
196     }
197     if(left!=0)//如果left不为0,需要把剩余的几位向左边靠拢
198     {
199         p[j]<=8-left;
200         fwrite(&count,sizeof(count),1,fp);
201         fwrite(p,1,num+1,fp);
202     }
203     else
204     {
205         fwrite(&count,sizeof(count),1,fp);
206         fwrite(p,1,num,fp);
207     }

```

```
208     fclose(fp);
209 }
210
211 Status ReadBit()//bit转字符串
212 {
213     strcpy(huffmanCode,"");
214     char *p;
215     int i,j=-1,count,num,left,t=0;
216     unsigned char flag=128; //即0b1000000,用于做位运算,注意要用无符号的字符型
217     printf("请输入代码存储文件名:");
218     fflush(stdin);gets(name);strcat(name,".txt");
219     if((fp=fopen(name,"rb"))==NULL)
220     {
221         printf("该文件夹下无%s!\n",name);
222         return ERROR;
223     }
224     fread(&count,sizeof(count),1,fp);
225     num=count/8;
226     left=count%8;
227     if(left==0)
228     {
229         p=(char*)malloc(sizeof(char)*num);
230         fread(p,1,num,fp);
231     }
232     else
233     {
234         p=(char*)malloc(sizeof(char)*(num+1));
235         fread(p,1,num+1,fp);
236     }
237     fclose(fp);
238     for(i=0;i<count;i++)
239     {
240         if(i%8==0)
241         {
242             j++;
243             flag=128;
244         }
245         if((p[j]&flag)) huffmanCode[t]='1';
246         else huffmanCode[t]='0';
247         t++;
248         flag/=2;
249     }
250     return OK;
251 }
```