

project1程序验收时间： 11周

project1提交资料时间： 12周

注意： 延迟会扣分哦

PROJECT2 内容和要求

---A SIMPLE HUFFMAN CODER

实验目的

1. 利用二叉树设计一款简单的huffman编码器。
2. 熟练掌握并灵活运用Huffman树的逻辑结构、存储结构、基本操作

实验内容及要求（功能）

程序验收标准，完成40分

1. 友好的用户界面，给出简单用户帮助
2. 从磁盘读入一个仅包括英文字母及标点符号的文本文件(如f1.txt)，统计各字符的频度，据此构建huffman树，对各字符进行编码，并将字符集，频度集及相应编码码字集输出在显示器或保存在另一个文本文件（如 f1_result.txt）中。基本要求
3. 对整个文件进行编码并将结果保存在一个二进制文件（如 f1_result.huf）中，同时计算压缩率。中级要求 5分
4. 根据f1_result.txt和f1_result.huf的内容，解码出原始的文本文件并另存到磁盘（如 f1_reconstruct.txt）高级要求 +5分

关于程序验收，实验报告及源程序代码提交

- 程序验收时间： 16周实验课（可提前验收）
 - 若完成项目（**基本要求**），可示意老师去验收，用**给定的文本文件(f1.txt, f2.txt, f3.txt)**之一现场验收
 - 当场删掉几行程序，要求在5分钟内补齐再编译运行。
 - 能运行，结果正确
 - **注意**，只在**实验课上**验收！
- 小组实验报告、源程序、个人总结报告提交时间： 17周**周五前**
 - 上传至QQ群

关于成绩说明

- 按期完成程序验收，完成基本要求得分40，完成基本要求+中级要求得分45，完成基本要求+中级要求+高级要求得分50。未完成（包括验收中运行结果不正确）0分
- 小组实验报告按期提交，内容完整充分，得满分20，内容不够充分完整酌情扣分。未提交0分
- 个人成绩，满分30（按个人计分，包括5分考勤分），按照个人总结中内容确定。
 - 注：若与小组报告中的分工不一致，会导致严重扣分。
 - 未提交 0分

运行演示

huffmanCode.exe

MAIN 函数 建议框架

.....

```
void main(){
```

.....

```
cout<<"操作命令说明: "<<endl;
```

```
cout<<"统计输入文件字符频度并对字符集编括码并输出至文件（基本要求）： 1"<<endl;
```

```
cout<<" 对整个文件编括码并保存编码结果到一个二进制文件（中级要求）： 2"<<endl;
```

```
cout<<"          文件解码并将解码结果保存为一文本文件（高级要求）： 3"<<endl;
```

```
cout<<"                                退出： 4"<<endl;
```

```
do {
```

```
    printf("\n$$"); scanf("%c", &option );    //用户选择操作命令键入
```

```
    switch(option ) {    //根据用户选择调用相关函数完成指定操作
```

```
        case '1':    //基本要求，输入文件，统计，编码字符集并输出
```

```
            char_code( ); break;
```

```
        case '2':    //中级要求，编括码整个文件并保存至一个二进制文件
```

```
            File_Code(); break;
```

```
        case '3':    //高级要求， 解码并保存解码结果到一个文本文件，以便与原始的文件作对比
```

```
            File_Decode(); break;    }
```

```
    } while(option != '4' );
```

```
}
```

程序验收标准

CHAR_CODE 函数框架（建议）

```
void char_code() {  
    ... ..  
    HuffTree<char> *HuffmanTree;    //用来存放构建好的Huffman树  
    int n; char *Code;  
    Char s[NUMBER]; int w[NUMBER]; //用来存放统计好的字符集和频度集  
    Stat (s, w, n); // 统计 字符频度对 集  
    HuffmanTree = HuffmanBuild(s, w, n);    // 建huffman树  
    HuffmanTree->print( );  
    cout<<endl<<" the coding result is:"<<endl;  
    Code = new char[n];    //分配 编码结果的工作空间  
    Encoder(HuffmanTree->root( ), Code, -1); //对各字符编码  
}
```

函数原型（建议）

- `void Stat(char * s , int * w, int& num);` //读入文本文件并统计频度不为零的字符集s，相应频度集 w，及个数num
- `HuffTree<T> * HuffmanBuild(char* s, int * w, int num)` //利用字符集s，相应频度集 w，及个数num构建 huffman 树
- `Void Encoder(HuffNode<char> * ht1, char * Code, int length)` //从构建好的 huffman树的根开始对各个叶子结点编码并输出编码结果。

频度统计一点建议

```
void Stat( char * s , int * w, int& num)
```

```
{
```

```
.....
```

初始化 **w**中每个元素为**0**

从输入文件逐个读入一个字符 **ch**,

```
w[ch]++
```

```
s[ch] = ch;
```

删除**w**和**s**中 频度为**0**的元素 , 同时统计频度不为**0**的字符个数放入 **num**

```
.....
```

```
}
```

HUFFMAN TREE构建思路(建议)

```
HuffTree<T> * HuffmanBuild( char* s, int* w, int num)
```

```
{ .....
```

```
    HuffTree<T> * ttree[NUMBER], *temp;
```

用字符集 **s** 和权值集 **w** 新申请num棵树，令ttree的元素指向他们

重复下列步骤直到ttree中只剩一个元素 {

- a) 从ttree中选择权值最小的的2个元素（即树），假定其下标为s1,s2;
- b) 以权值小者作为左孩子，权值大者作为右孩子新申请一棵huffman树，令temp指向该树；
- c) 让 ttree中 下标为 min(s1,s2) 的元素指向 temp指向的树;
- d) 从ttree中删除下标为max (s1,s2)的元素

```
}
```

```
    返回ttree[0];
```

```
}
```

huffman树构建过程
的数集合 F

HUFFMAN 编码思路(建议)

```
Void HuffmanCode(HuffNode<char> * ht1, char * Code, int length=-1)
```

```
{ .....
```

若 ht1 为空 return

若 ht1 为叶子结点 打印（输出） ht1 的 val, weight, Code;

否则 如果 ht1 的左孩子不为空

copy Code 到一个临时字符串 temp_c,

令 len=length; len++; temp_c[len] = '0' ;

递归调用 HuffmanCode (ht1->left(), temp_c, len)

否则 如果 ht1 的右孩子不为空

copy Code 到一个临时字符串 temp_c,

令 len=length; len++; temp_c[len] = '1' ;

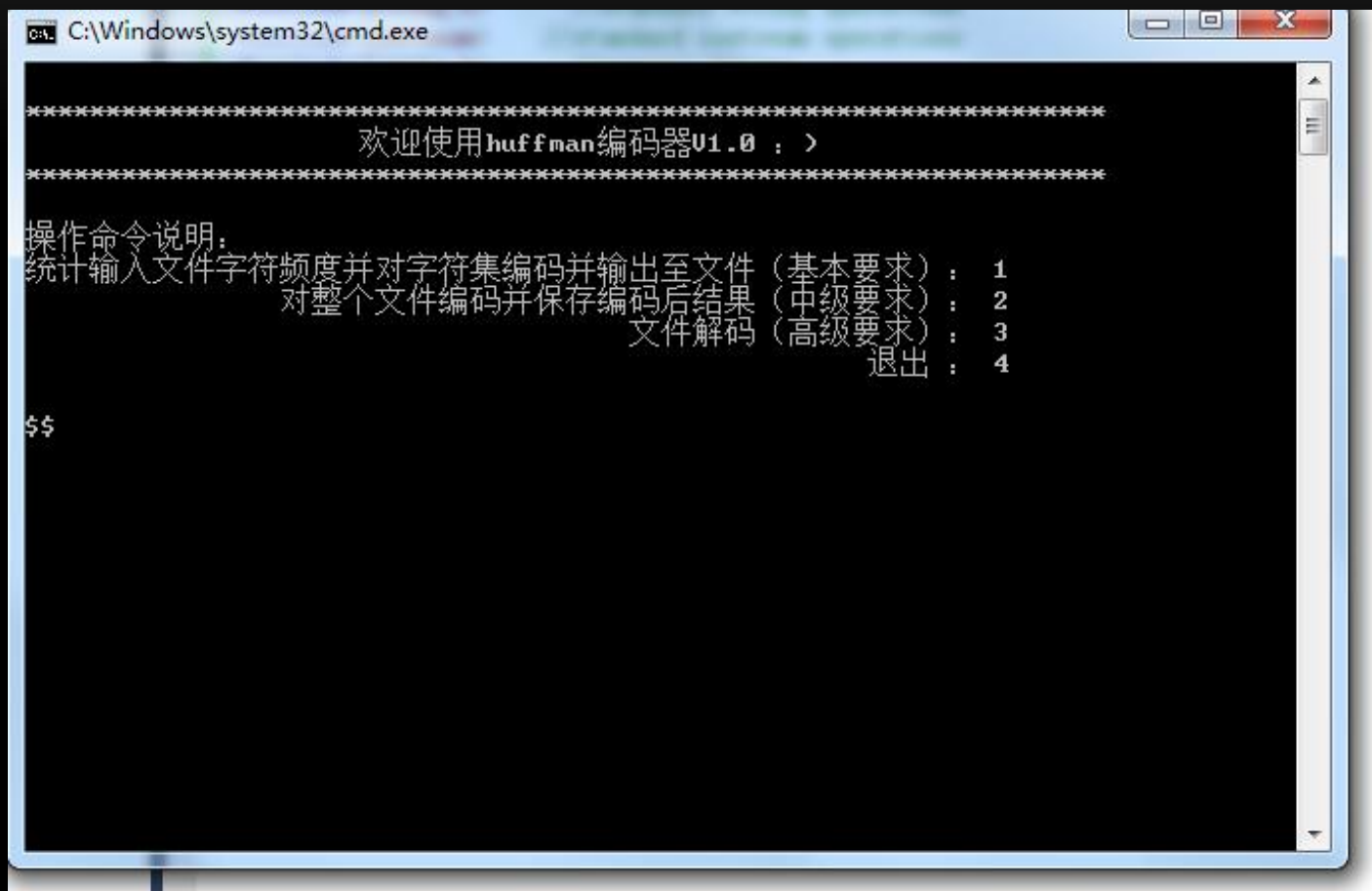
递归调用 HuffmanCode (ht1->right(), temp_c, len)

```
.....
```

```
}
```

也可定义一全局数组来存放

用户界面 (建议)



```
C:\Windows\system32\cmd.exe

*****
                欢迎使用huffman编码器U1.0 : >
*****

操作命令说明:
统计输入文件字符频度并对字符集编码并输出至文件 (基本要求) : 1
对整个文件编码并保存编码后结果 (中级要求) : 2
文件解码 (高级要求) : 3
退出 : 4

$$
```

关于中级及高级要求的补充建议

- 文件编码结果输出为按bit输出，解码时输入也是按比特输入

```
struct Buffer{           //字节缓冲
    char ch;
    unsigned int bits; }; //实际比特数

void Write(unsigned int bit) { //向outfp中写入一个比特
    buf.bits++;    buf.ch=(buf.ch<<1)+bit;
    if(buf.bits==8) { //缓冲区已满,写入outfp
        fputc( buf.ch, buf);  buf.bits=0;  buf.ch=0;  }
    }

void Read(unsigned int &bit) { //从infp中读出一个比特
    if (buf.bits==0) {  buf.ch=fgetc(infp);  buf.bits=8;  }
    bit=(buf.ch & 128)>>7;  buf.ch=buf.ch<<1;  buf.bits--;
}
```

buf, infp, outfp
可定义为全局变量

关于高级要求的补充建议

解码思路，

输入：待解码二进制文件 和 huffmantree，或者包含字符权值对集合及编码二进制流的待解码二进制文件），

输出：解码后的原始文本文件

Do while 未到达输入二进制文件末尾

 设huffmantree根节点为当前结点

 Do while 当前结点不为叶子结点

 1) 从输入文件读入一个比特位 c ,

 2) 若 $c=0$, 更新当前结点为其左孩子, 否则, 更新当前结点为其右孩子

 写当前结点的val(字符)至输出文件