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---能对一篇英文文本文件进行定制化编码。
2. **熟练掌握并灵活运用Huffman编码树的逻辑结构、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_ccode.txt）中. 基本要求
3. 根据2）获得的码字集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, 同时计算压缩比。 中级要求 5分
4. 根据f1_ccode.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 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： 15，16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(f3.txt) 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：16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
     HuffmanCode(HuffmanTree->root( ), Code, -1);    //对各字符编码
}
```

CHAR_CODE 涉及的函数原型（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 HuffmanCode(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];
    HuffTree<T> *temp, *t1, *t2;
```

huffman树构建过程的树集合 F，用minHeap组织

1） 用字符集 s 和权值集w 新申请num棵huffman树，并将其放入数组ttree中

2） 以ttree为基础初始化并构建小堆Forest

3） 重复下列步骤直到Forest中只剩一个结点 {

a) Call removefirst 2次，获得Forest中最小和次小的两颗树 t1, t2；

b) 以t1作为左孩子，t2作为右孩子新申请一棵huffman树，令temp指向该树；

c) Call insert 将temp插入Forest;

……

}

4） 返回temp;

}

HUFFMAN 编码思路(建议)

```
Void  HuffmanCode(HuffNode<char> * ht1, char * Code, int length)
{   ……
    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

关于中级及高级要求的一点建议

- 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, outfp);   buf.bits=0;   buf.ch=0;   }
}
void   WriteToOutfp() {        //强行写入outfp
      unsigned int l=buf.bits;
      if (l>0)        for(unsigned int i=0;i<8-l;i++)  Write(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

用户界面（建议）


C:\Windows\system32\cmd.exe
欢迎使用huffman编码器V1.0 :)
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
$$