**西 安 邮 电 大 学**

**（计算机学院）**

数据结构课程设计报告

# 题 目： 哈夫曼编译码器

**专业名称： 软件工程**

**班 级： 1901**

**学生姓名： 刘宇阳**

**学号（8位）： 04193016**

**指导教师： 王燕**

**设计起止时间：** 2020年12月28日—2021年1月1日

## 一. 设计目的

数据结构课程设计的主要目的是使学生通过系统分析、系统设计、编程调试、写实验报告等环节，进一步掌握应用系统设计的方法和步骤，灵活运用并深刻理解典型数据结构在软件开发中的应用，进一步提高分析问题和解决问题的能力，提高程序设计水平。

## 二. 设计内容

2. 哈夫曼编译码器

利用哈夫曼编码进行信息通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本，编写代码实现一个哈夫曼的编/译码器，要求在发送端通过一个编码系统对待传数据预先编码，在接收端将传来的数据进行译码（复原）。

具体功能包括：

1. 建立哈夫曼树：读入文件(\*.souce)，统计文件中字符出现的频度，并以这些字符的频度作为权值，建立哈夫曼树。
2. 编码：利用已建立好的哈夫曼树，获得各个字符的哈夫曼编码，并对正文进行编码，然后输出编码结果，并存入文件(\*.code)中。
3. 译码：利用已建立好的哈夫曼树将文件(\*.code)中的代码进行译码，并输出译码结果，并存入文件(\*.decode)中。
4. 利用位操作，实现文件的压缩与解压。（选作）

## 三．概要设计

### 1．功能模块图；

### 2．各个模块详细的功能描述。

1.建树：首先定义适合哈夫曼树的节点类型﹐需要定义的有当前节点的字符﹐当前节点的左子﹑右子和父亲指针·在建立哈夫曼树之前还需要对山现的字符和权重进行统计和记录﹐并且定义一个可以筛选出最小权重的函数。初始化树节点之后开始建立哈夫曼树·先在所有可能出现的字符中筛选出当前权重最小的两个字符﹐将这两个字符分别作为新节点的左子和右子建立一个小的二叉树﹐并将两个字符的权重之和赋值给新节点﹐将新二叉树放入筛选字符中﹐再将筛选过的两个字符从筛选列表中淘汰掉·依次对列表中剩下的字符进行权重最小的筛选﹐直到根节点（如果编码表共有N个字符﹐则2\*N-1就为最终根节点）为止﹐也就是当筛选列表为空的时候﹐哈夫曼树即建立完成。

2.编码（Encode）：此部分的主要功能为建立哈夫曼树，并对文件进行二进制编码。首先，用户输入一个自建的文件（.souce），将其中的内容存储到字符串中以便程序编码调用。开始对需要编码的字符进行编码，捋字符逐一读取与刚刚建立的编码表中的每个叶子节点代表的字符进行比较﹐找出相同的对象，构建哈夫曼树，初始化叶子节点，给各节点的左右孩子分支赋值“0”和“1”，并将其输出在屏幕上，写入自动新建的文件（.code）

3.译码（decode）：此部分的主要功能为利用第一步编码建立好的哈夫曼树对文件进行译码。首先要求用户输入需要译码的文件（\*.code），程序读取内容，利用哈夫曼树的左右孩子值来辨别结点，当密文中的当前字符是0'的时候﹐则索引走向左子节点﹔当是‘1'的时候﹐则走向右子节点。以此类推﹐一直走到叶子节点为止﹐则当前叶子节点所代表的字符即为前一段密文的解码结果﹐再对下一个字符依次从根节点开始解码﹐如此循环对每一段密文进行解码直到解码结束，并写入一个新文件（\*.decode）

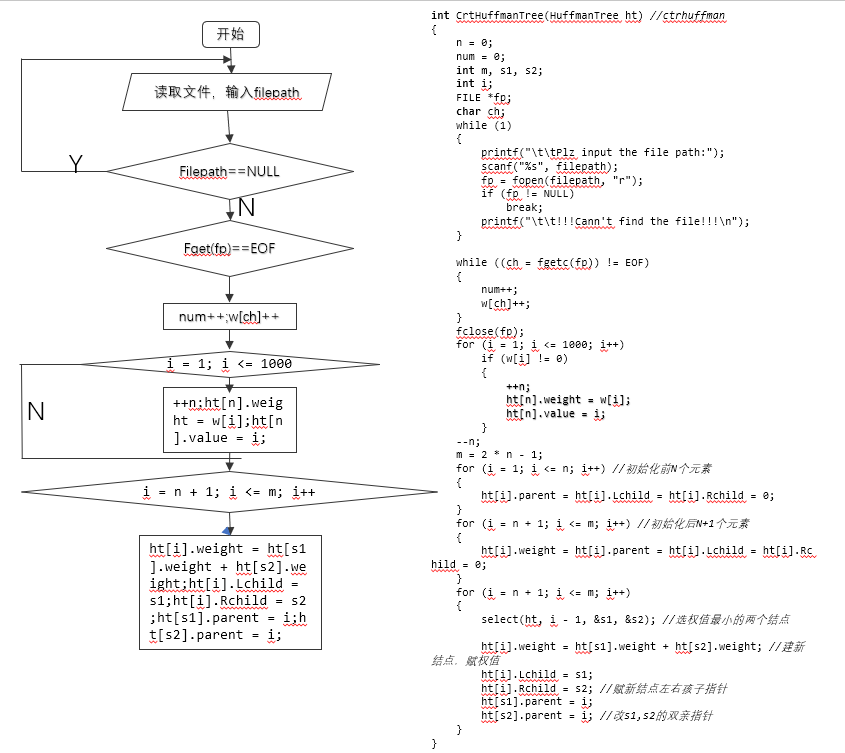
4.压缩（compress）：统计字符种类及频度：ASCII码值当下标，字符频度当对应下标的值。构建哈夫曼树：在没有访问过的节点中，找最小字符频度下标来构建哈夫曼树。构建哈夫曼编码：递归思想构建。生成压缩文件（.ycyHuf后缀名）：把字符的哈夫曼编码以二进制形式写入目标文件中。给压缩文件头部写入元数据，解压缩时需使用这些数据。

5.解压（uncompress）：统计字符种类及频度：ASCII码值当下标，字符频度当对应下标的值。构建哈夫曼树：在没有访问过的节点中，找最小字符频度下标来构建哈夫曼树。构建哈夫曼编码：递归思想构建。生成压缩文件（.ycyHuf后缀名）：把字符的哈夫曼编码以二进制形式写入目标文件中。给压缩文件头部写入元数据，解压缩时需使用这些数据。

## 四．详细设计

### 1．功能函数的调用关系图

### 2．各功能函数的数据流程图



### 3．重点设计及编码

|  |
| --- |
| 1.select函数的功能为选双亲为0且权值最小的两结点  void select(HuffmanTree ht, int end, int \*s1, int \*s2) //  {  int i;  int m1, m2;  m1 = m2 = Maxvalue;  for (i = 1; i <= end; i++)  {  if (ht[i].parent == 0)  {  if (ht[i].weight < m1)  {  m2 = m1;  m1 = ht[i].weight;  \*s2 = \*s1;  \*s1 = i;  }  else if (ht[i].weight < m2)  {  m2 = ht[i].weight;  \*s2 = i;  }  }  }  } |
| 2.建立哈夫曼树  for (i = 1; i <= 256; i++)  if (w[i] != 0)  {  ++n;  ht[n].weight = w[i];  ht[n].value = i;  }  --n;  m = 2 \* n - 1;  for (i = 1; i <= n; i++) //初始化前N个元素成为根结点  ht[i].parent = ht[i].Lchild = ht[i].Rchild = 0;  for (i = n + 1; i <= m; i++) //初始化后N-1个空元素  ht[i].weight = ht[i].parent = ht[i].Lchild = ht[i].Rchild = 0;  for (i = n + 1; i <= m; i++)  {  select(ht, i - 1, &s1, &s2); //选权值最小的两个结点  ht[i].weight = ht[s1].weight + ht[s2].weight; //建新结点，赋权值  ht[i].Lchild = s1;  ht[i].Rchild = s2; //赋新结点左右孩子指针  ht[s1].parent = i;  ht[s2].parent = i; //改s1,s2的双亲指针  }  } |
| 3.哈夫曼编码  char \*cd;  int start;  cd = (char \*)malloc(n \* sizeof(char)); //临时编码数组  cd[n - 1] = '\0'; // 从后向前逐位求编码，首先放编码结束符  printf("\t\t\tvaule times huffmancode\n");  for (i = 1; i <= n; i++) //从每个叶子开始，求相应的哈夫曼编码  {  start = n - 1;  int c = i;  int p = ht[i].parent;  while (p != 0)  {  --start;  if (ht[p].Lchild == c)  cd[start] = '0'; //左分支得“0”  else  cd[start] = '1';  c = p;  p = ht[p].parent;  }  hc[i] = (char \*)malloc((n - start) \* sizeof(char));  strcpy(hc[i], &cd[start]);  } |
| 4.文件操作：打开并读取文件，新建并写入文件  char filepath[100]; //定义全局变量  FILE \*fp, \*fp1; //定义文件指针  while(1)  {  printf("\t\tPlz input the file path:");  scanf("%s", filepath); //输入文件路径  fp = fopen(filepath, "r"); 从文件中读数据  if (fp != NULL)  break; 如果读取成功则跳出循环  printf("\t\t!!!Cann't find the file!!!\n"); 如果读取失败则继续输入文件路径读取  } //循环，如果输入文件路径错误可重新输入    char filename[100]; //  fp = fopen(filepath, "r");  fp1 = fopen("HuffmanByLYY.code", "w+");//打开可写入得新建文件  while ((ch = fgetc(fp)) != EOF) //将文件中的字符逐个赋值于数组中  {  num++;  w[ch]++;  } |
| 5.使用\*.souce中形成的哈夫曼树将\*.code二进制文件译码回\*.decode原文件  char str[100];  printf("\n\t\tThe decoding result is:\n\t\t");  while ((ch = fgetc(fp)) != EOF)  {  if (ht[j].Lchild && ht[j].Rchild)  {  if (ch == '0')  j = ht[j].Lchild; //若遍历到0，则为左孩子  else  j = ht[j].Rchild; //若遍历到1，则为右孩子  }  if (ht[j].Lchild == 0 && ht[j].Rchild == 0)//当循环到叶子时开始写入字符  {  fprintf(fp2, "%c", ht[j].value); //数组元素写入文件中  printf("%c", ht[j].value); //打印字符  j = 2 \* n - 1;  str[i++] = ht[j].value;  }  i++;  } |

## 五．测试数据及运行结果

### 1．正常测试数据和运行结果

要求提供3组正常测试数据和运行结果

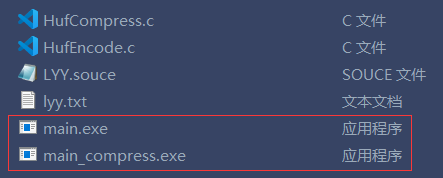
一、编译码：

1. 编码

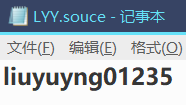
首先在WindowsPowerShell中使用命令g++ \*.c -o \*编译源文件生成.exe可执行程序



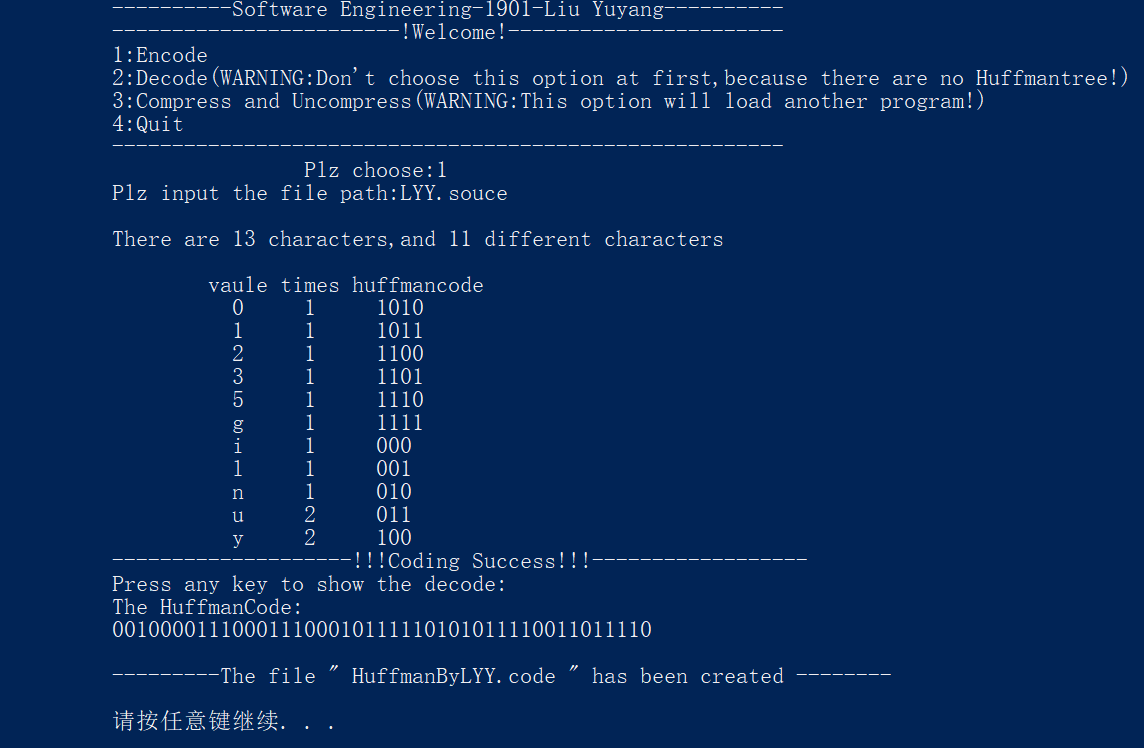
如图：



且LYY.Souce中文件内容为



运行程序并输入文件名：



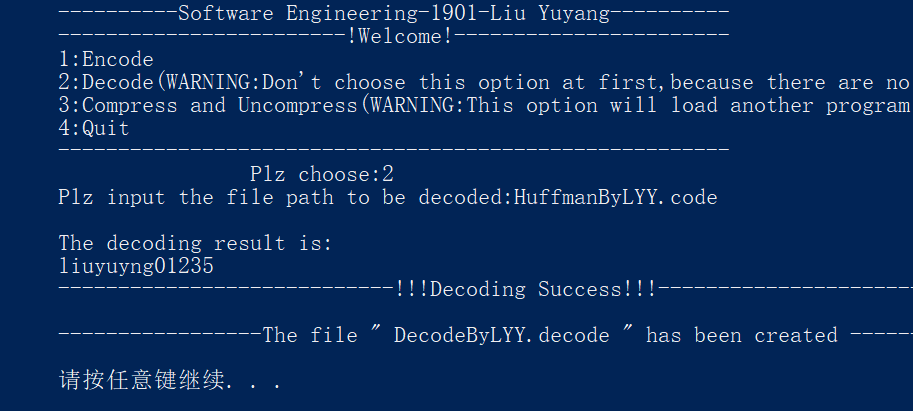
然后回到文件夹会发现HuffmanByLYY.code文件已被创建，打开文件：



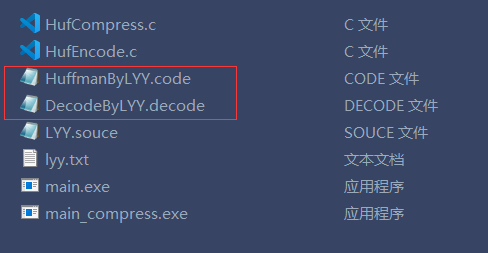
编码成功！！！

2.译码：

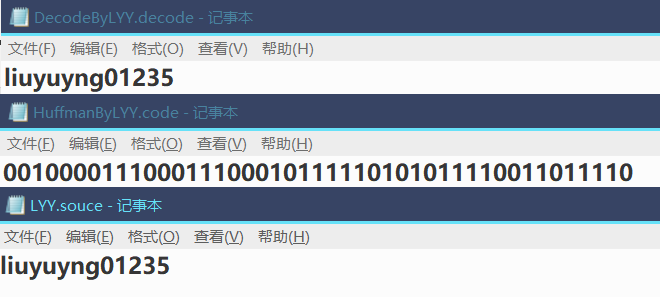
在编码成功的基础上，对HuffmanByLYY.code进行译码



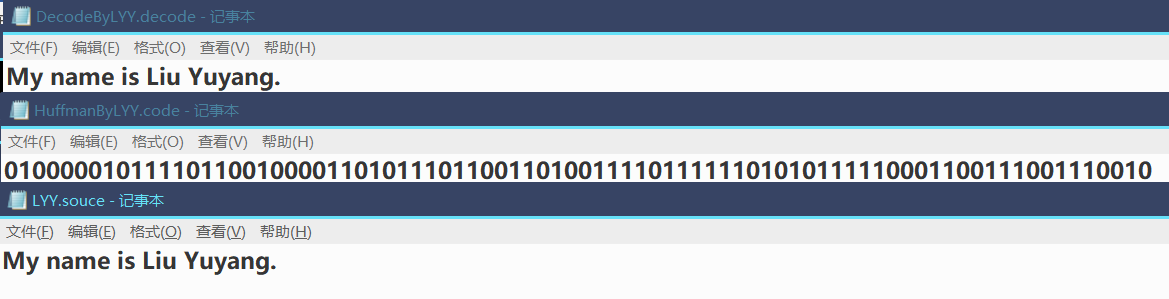
回到文件夹中发现DecodeByLYY.decode文件已被创建



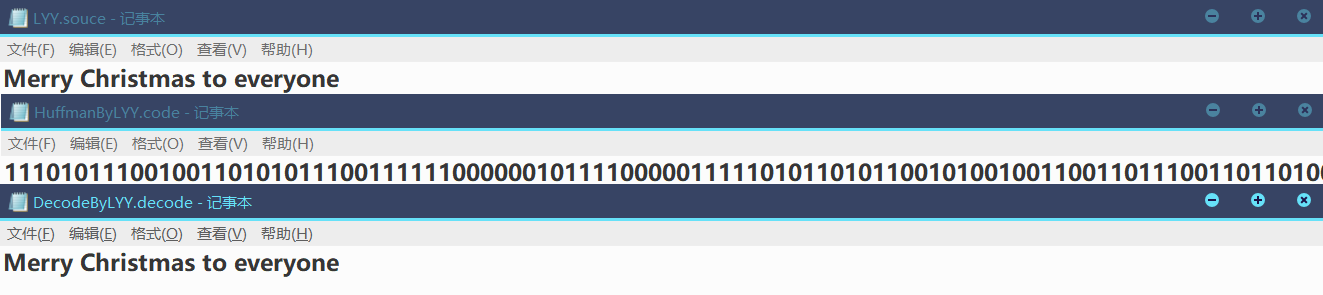
打开文件DecodeByLYY.decode与LYY.souce，会发现两文件相同，证明哈夫曼编译码成功！



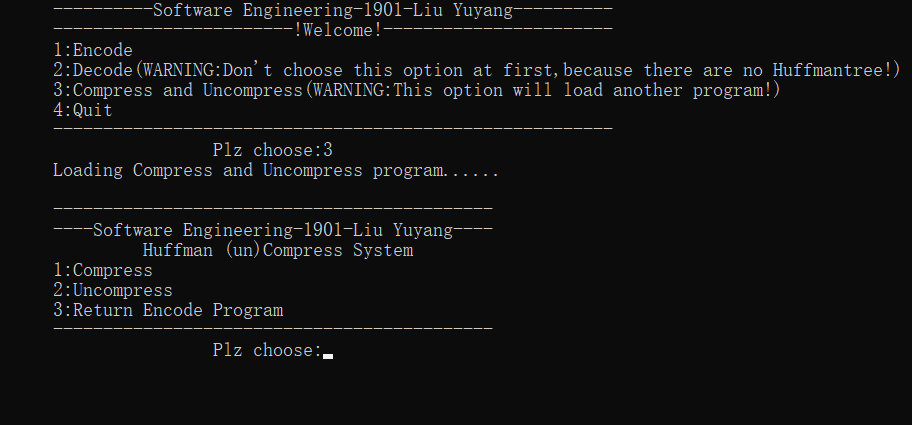
用同种方式进行另外两组测试哈夫曼编译码得结果为：



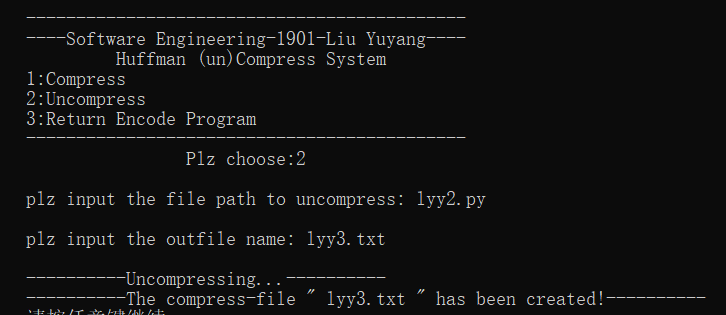
②



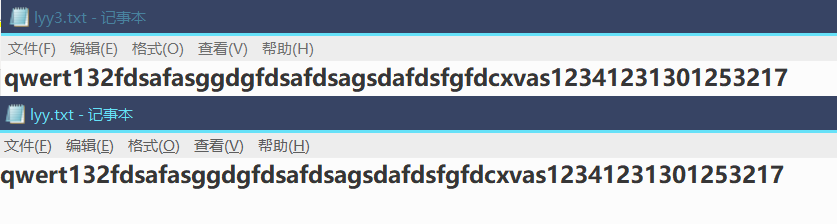
压缩与解压结果：



↑↑↑输入3来进入压缩程序，输入1，再输入压缩文件路径与输出路径



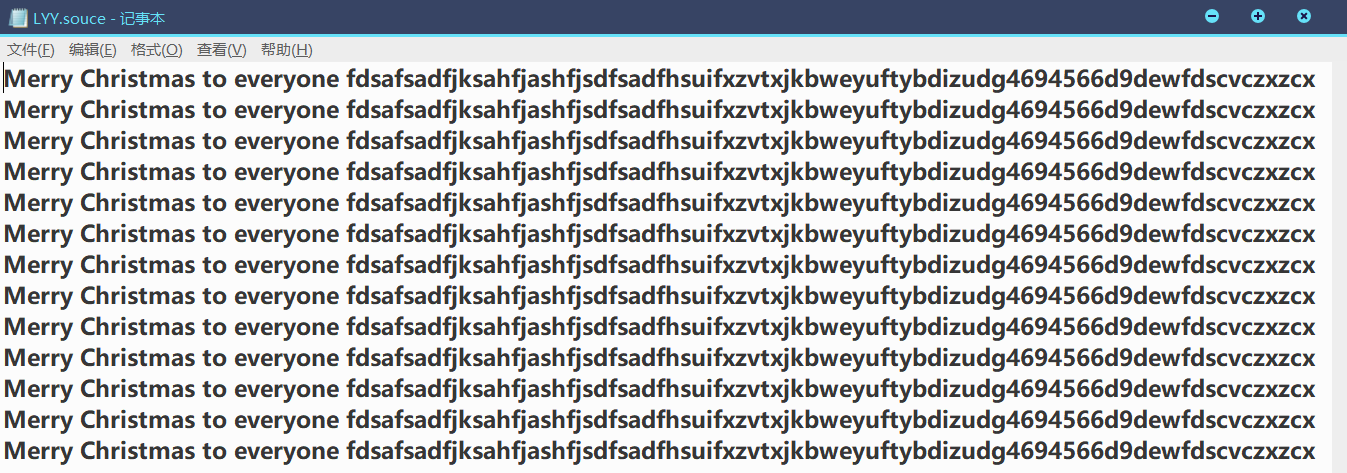
解压同理，会发现压缩与解压前后文件内容一致

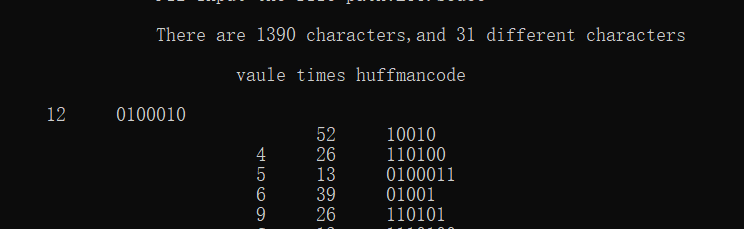


### 2．异常测试数据及运行结果

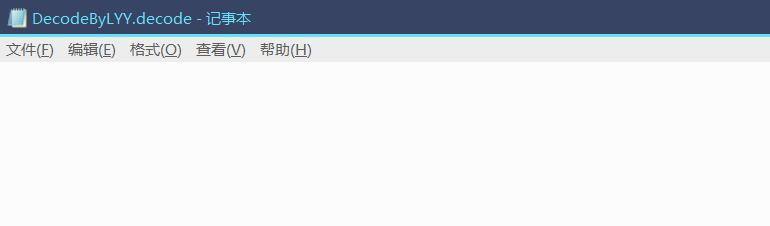
要求提供2组异常测试数据和运行结果

当文件内容过多时，会出现译码无法正常进行的情况





译码时会出现无法译回和源文件一致的内容，甚至出现没有内容的情况。



## 六．调试情况，设计技巧及体会

### 1．改进方案

经过三天的时间，终于完成了哈夫曼编译码器的课程设计，期间遇到过很多问题，也出现过很多bug，但经过自己的不断努力与改进，将一个个问题都解决了。

开始时，经常出现无法构建哈夫曼树，编码出来的文件出现乱码的情况，，一直查不出来是哪里出现了错误，后来更正了全局变量和文件编码格式，最终成功的完成编码。

现在程序还是存在不全之处的，如果源文件长度过长的话，编码后的二进制文件再进行译码时会出现程序卡死的情况，译码文件也没有存入任何内容。

### 2．体会

对设计及调试过程的心得体会。

三个课题里我选择了哈夫曼编译码器，利用哈夫曼编码进行信息通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本，一开始我觉得它能稍微简单，但是写了一个上午之后，我发现有很多改不完的bug，不但要考虑如何实现函数的基本功能，还要兼顾代码的简洁性，UI方面我也考虑了很长时间来布局，最终经过了三天的代码编写与调试，我完成了课程设计的基本要求。

在这期间，我发现自己在编写代码这方面还有很多需要学习的地方，需要通过更多的代码来进行练习，这次课程设计锻炼了我的思维逻辑能力与编码能力，但我还需要不断的努力来提升我自己。

## 七．参考文献

[1] 王曙燕.数据结构与算法. 北京：高等教育出版社. 2019

[2] 王曙燕.数据结构与算法. 北京：人民邮电出版社. 2013

[3] 耿国华.数据结构C语言描述. 北京：高等教育出版社. 2011

[4] 严蔚敏.数据结构. 北京：清华大学出版社. 2012

[5] 王曙燕.C语言程序设计教程. 北京：人民邮电出版社. 2014