## 

项目说明文档

数据结构课程组队项目设计

——霍夫曼树编码及其图形化演示

队伍成员：

2051877 邓凯伦

2051498 储岱泽

学 院：软件学院

专 业：软件工程

指导老师：张颖

同济大学

Tongji University

目录

[1.需求分析 4](#_Toc14625)

[2.程序设计 4](#_Toc6098)

[2.1数据结构设计 4](#_Toc32015)

[2.2 类的设计 4](#_Toc30489)

[2.2.1 Node类 4](#_Toc17065)

[2.2.2 HuffmanTree类 5](#_Toc20001)

[2.2.3 最小堆 5](#_Toc20399)

[2.2.4 System类 6](#_Toc21460)

[2.2.5 EasyButton类 6](#_Toc3813)

[2.2.6 EasyTextBox类 6](#_Toc8545)

[3.项目实现 7](#_Toc4180)

[3.1建立霍夫曼树 7](#_Toc15351)

[3.2字符进行霍夫曼编码 8](#_Toc19598)

[3.3 霍夫曼译码 8](#_Toc10613)

[3.4 文件流读写操作实现 8](#_Toc19237)

[4.图形化实现 9](#_Toc20376)

[4.1图形化工具 9](#_Toc3500)

[4.2界面风格设计 9](#_Toc6489)

[4.3总体绘图设计 9](#_Toc27650)

[4.4 easyx实现文本框输入 12](#_Toc31914)

[4.5 easyx实现按钮部件 12](#_Toc13239)

[4.6霍夫曼树绘图 13](#_Toc2738)

[5.项目测试 14](#_Toc11206)

[5.1功能测试 14](#_Toc18222)

[5.1.1 文件输入及字符统计功能测试 14](#_Toc26365)

[5.1.2 霍夫曼树的构建及图形化测试 14](#_Toc10154)

[5.1.3 霍夫曼编码以及图形化测试 15](#_Toc25289)

[5.1.4获得输入文件对应的编码文件 15](#_Toc25000)

[5.1.5译码功能测试 16](#_Toc7491)

[5.2 错误和边界测试 17](#_Toc9978)

[6.项目说明 18](#_Toc10704)

# 1.需求分析

基于数据结构课程内容的学习，在组队项目的设计当中考虑运用课程所学知识——霍夫曼树为主体进行项目的建立。与定长编码以及一般的可变编码相比，霍夫曼编码有其强大的优势：不定长编码，可以保证解码的无二义性；应用场景很广，常见用于文件压缩（无损）；最重要的是它的构建原理——基于字符的出现频率进行权重分配来构建，因此它最强大的优点就是节省空间。

基于上述分析，我们综合采用类进行分装来构建霍夫曼树。考虑到项目的设计是基于用户的，因此项目实现中采用EasyX图形库对其进行可视化，美观地显示字符编码以及霍夫曼树结构等。同时考虑功能的多样性，设计中采用菜单界面进行功能分化，实现多功能统一。

# 2.程序设计

## 2.1数据结构设计

基于上述分析，可以将其分为数据结构化和可视化两个部分。数据结构上面利用最小堆，构建HuffmanTree类进行霍夫曼树的封装实现；可视化上设计EasyButton 、EasyTextBox类分别实现按钮控件和文本框控件。考虑整个项目为一个整体，因此在实现中将其视为一个系统，用System类进行封装，实现功能的中心化和统一化。

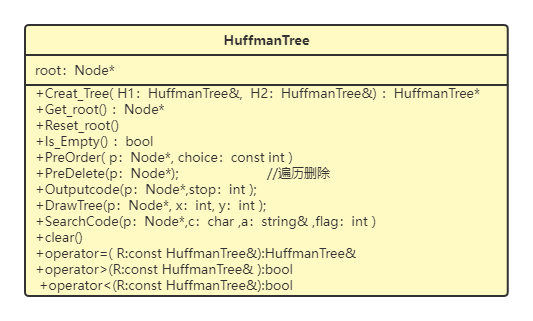
## 2.2 类的设计

在霍夫曼树编码及其图形化演示项目中，一共设置了6个类，进行总体的封装实现，类具体设计说明如下：

### 2.2.1 Node类

Node类用于表示树的节点，其内部包含了一个int类型成员Data表示编码权重，char类型成员Name表示字符名称，string类型成员Code记录该节点对应字符的编码；同时包含两个Node\*指针分别指向左右子女。类设计如图1

### 2.2.2 HuffmanTree类

HuffmanTree类包含了一个Node\*成员root表示树的根节点，内部设计Creat\_Tree()操作，传入两个HufffmanTree引用,生成一个新的HufffmanTree作为返回，以便放入堆中；设计DrawTree()操作实现霍夫曼树的绘制，PreOrder ()操作实现递归遍历进行霍夫曼编码，SearchCode()操作实现霍夫曼编码的查找，利用不同的成员函数实现对应的功能，类设计如图2.

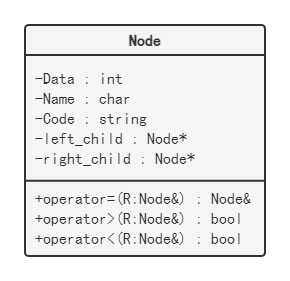


图1：Node类设计图 图2：HuffmanTree类设计图

### 2.2.3 最小堆

在霍夫曼树的构建当中，利用最小堆存储所有字符及其权重，每一次从最小堆中取两个节点，生成一颗二叉树，根节点权值为两个字符权重之和，然后又将该树重新放入最小堆当中，从而实现霍夫曼树的构建。最小堆MinHeap类设计当中，其内部包含一个存储HuffmanTree的数组，以及两个int型变量Maxsize和Currentsize表示堆的最大节点数目和当前节点数目。通过提供操作creat\_mheap()实现树的创建，Insert()和Delete()操作实现节点的插入和删除，DownSift()和UpSift()操作实现堆的向下和向上调整，类的设计如图3。

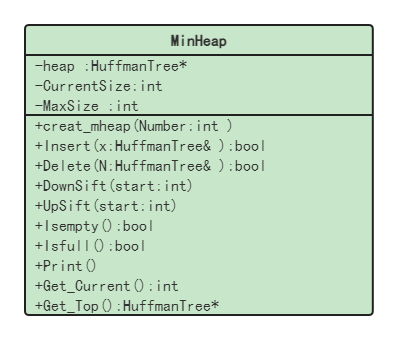
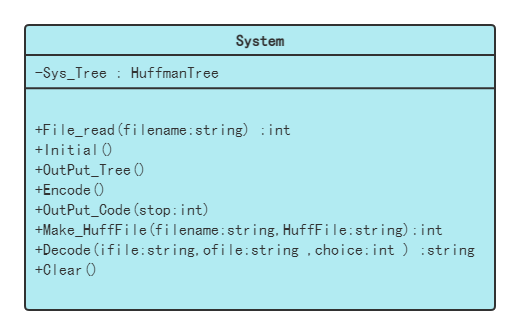


图3：最小堆设计 图4：System类设计

### 2.2.4 System类

System类将霍夫曼树进行封装，内部包含HuffmanTree成员Sys\_Tree,提供文件读写操作File\_read(),Encode()和OutPut\_Code()操作分别进行编码以及编码输出，提供Decode()操作进行解码。利用System实现了项目功能的封装统一，类设计如图4。

### 2.2.5 EasyButton类

EasyButton用于进行图形化中按钮的实现，内部设置4个int型成员left,right,top,bottom表示控件坐标，wchar\_t\* text表示输出内容，userfunc记录控件消息，表示用户按下这个按钮后接下来程序会执行的操作。设计Creat()操作和Show()操作实现界面的绘制，类设计如图5.

### 2.2.6 EasyTextBox类

EasyTextBox类用于进行图形化中文本框的实现，内部设置4个int型成员left,right,top,bottom表示控件坐标，wchar\_t\* text表示输出内容，size\_t maxlen表示文本框的最大内容长度，设计Creat()操作和Show()操作实现界面的绘制，OnMessage()操作实现文本输入，类设计如图6.

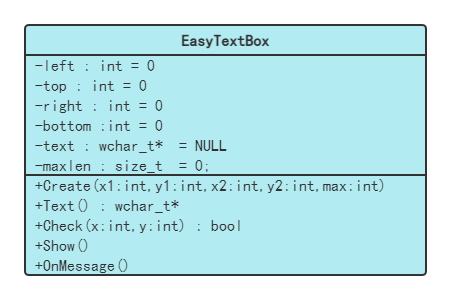
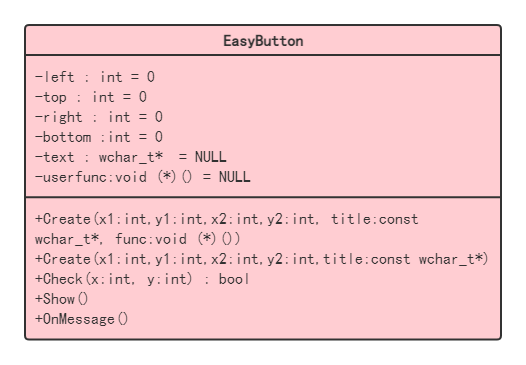


图5：EasyButton类设计 图6：EasyTextBox类设计

# 3.项目实现

## 3.1建立霍夫曼树

初始时输入文件名，利用File\_read()从文件中读取字符数据并进行统计，将结果输出到文件当中。然后读取输出的结果文件，将每一个字符视为只有一个根节点的霍夫曼树，利用最小堆存储所有的字符节点，通过DownSift()操作实现堆的向下调整。

当堆不为空时，每次从堆顶中取两个元素，用调用Creat\_Tree()函数构建一颗新的霍夫曼树，并重新放入堆中。新的霍夫曼树构建流程如图7。

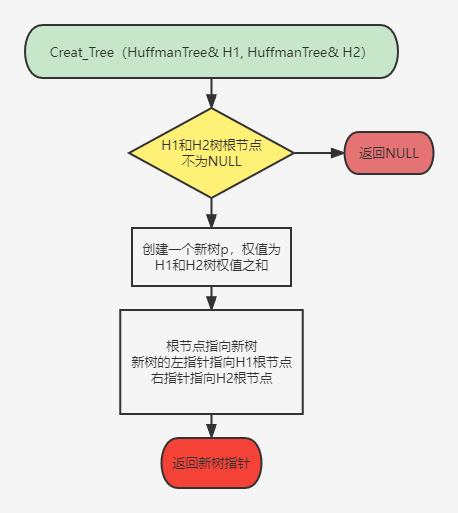


图7：霍夫曼树构建流程图

## 3.2字符进行霍夫曼编码

获取霍夫曼树字符的编码通过递归函数PreOrder()进行实现；利用一个string变量进行霍夫曼编码的记录，通过递归遍历的调用形式，得到字符的编码，流程如图8.

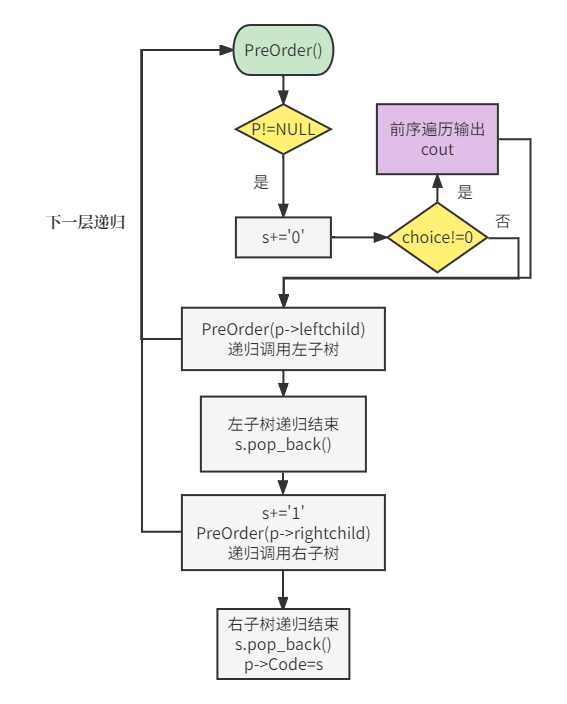
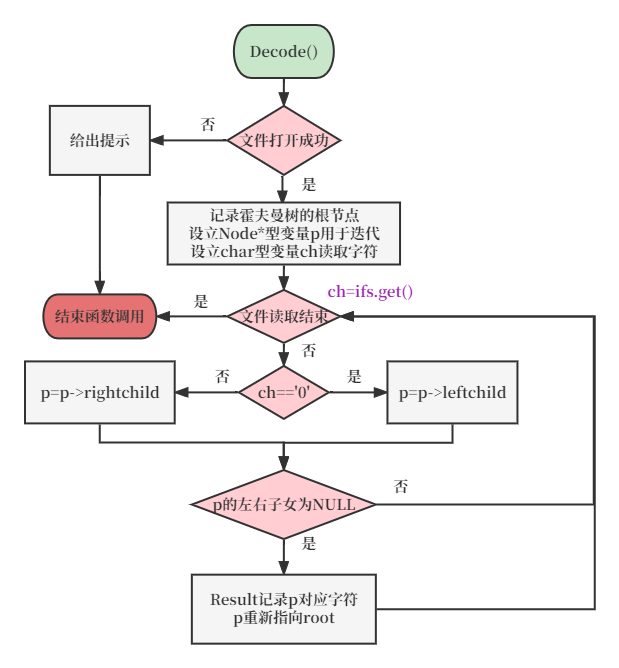


图8：霍夫曼编码流程图 图9：霍夫曼译码流程图

## 3.3 霍夫曼译码

霍夫曼译码操作是从文件中读取需要译码的内容，利用一个string类型变量Result来记录最终的结果。每次迭代从根节点出发，若读到的是0，则向左迭代，否则向右迭代，若迭代至节点p,若其为叶节点，则说明该节点是需要译码的字符，Result记录节点p对应的字符，具体流程如图9.

## 3.4 文件流读写操作实现

在本项目中，主要的数据输入和输出是通过文件读写的方式进行的，每次读写前判断文件打开是否成功，提高稳健性。文件读取利用了ifstream类， 采取一个个字符读取的方式，因为在编码和译码都需要逐个分析。而在写入时利用ofstream类，将要写入数据用string变量存储好，直接利用流操作符进行写入。

## 4.图形化实现

### 4.1图形化工具

由于我们采用的编程语言是C++，所以我们第一个就想到了C++的一个功能强大的图形库——Easyx来实现我们编程的可视化。在Easyx图形库中会提供一些基本的图形，我们可以对这些图形进行组装，来实现我们霍夫曼编码译码器的图形化。

### 4.2界面风格设计

项目整体的色彩风格是比较偏向于简约风，主色为蓝色，黑色与白色。没有过多累赘与花里胡哨的东西，给人以大气清爽的视觉效果。同时，提示语简洁明了，具有鲜明的导向作用，同时操作简单，可以让用户在拿到程序之后很快的上手使用，无需观看过多教程。

### 4.3总体绘图设计

运行程序之后首先看见的是“开始”界面，当我们用鼠标点击Start之后，程序就进入下一个界面，要求用户输入需要译码文件名称，在输入过程中，我们可以看见光标的跳动以及输入内容的显示，同时也可以删除自己输入的内容，就像是一般的编辑器一样的操作。这样的文本框部件在easyx中本是没有原形的，在下面也会详细的解释这样一个逼真的文本框部件是怎么实现的。输入完要编码的文件名称之后，就直接按“OK”键，就可以进入“菜单”，选择接下来要进行的操作。

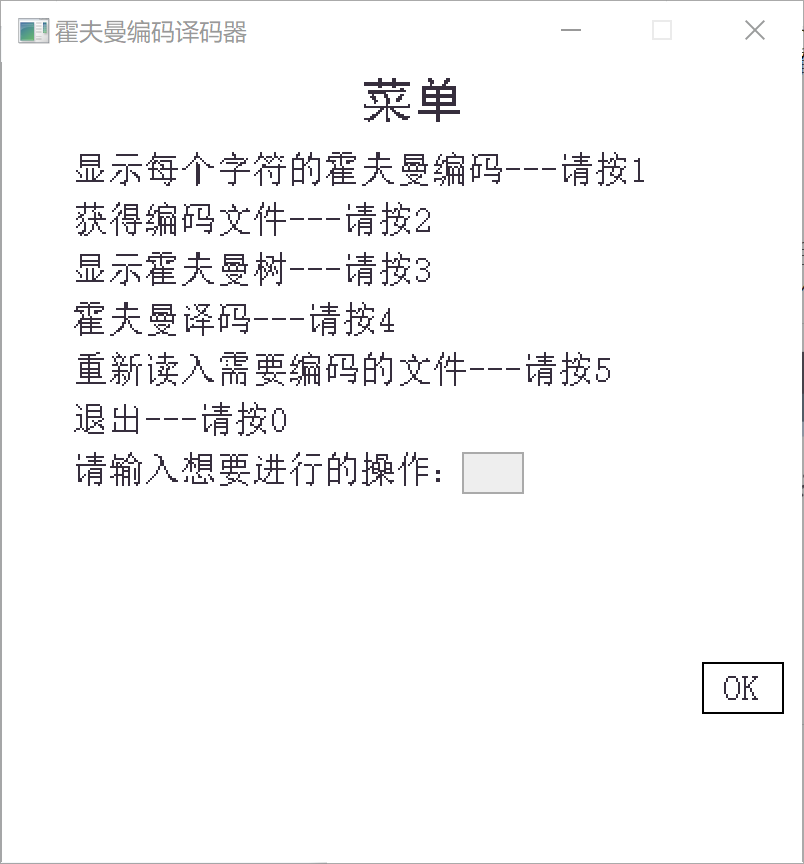
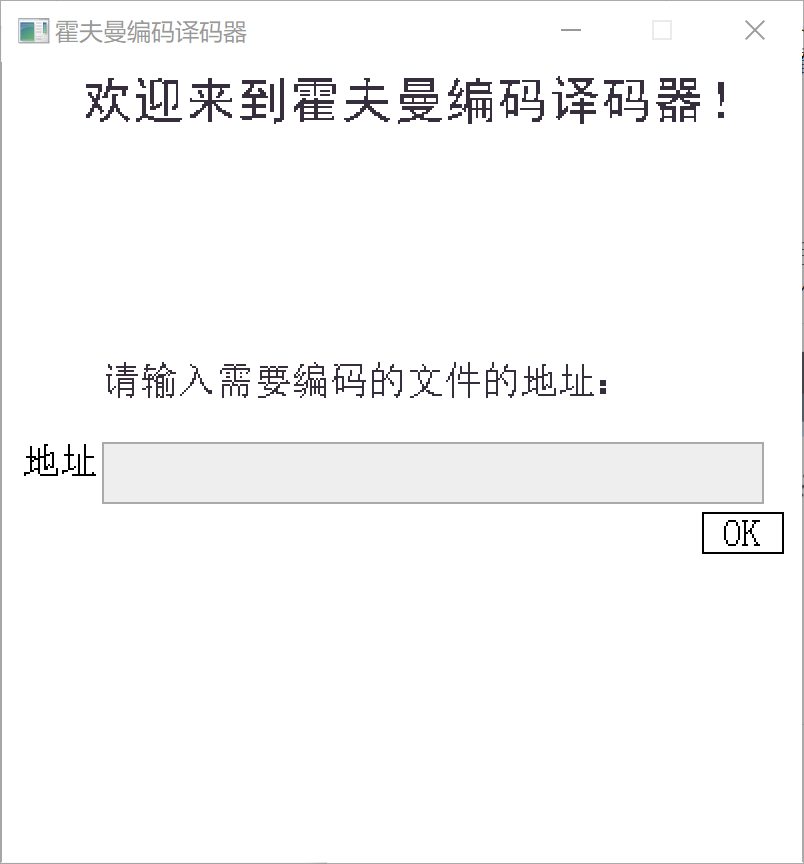


图10：初始操作界面流程

在菜单中显示了程序所支持的所有功能，将需要进行的操作对应的编码输入到文本框中，按下OK便可以执行对应的操作：（操作界面如图10）

首先，输入“1”查看每一个字符对应的霍夫曼编码，界面转移到霍夫曼编码的显示界面，首先改变窗口大小从400\*400到800\*800，为的是能够显示更多的字符，然后开始打印字符及其对应的编码，每三个为一行，没打完一个字Sleep(100)，让用户更清楚的看见每一个字符对应的编码。界面效果如图11：

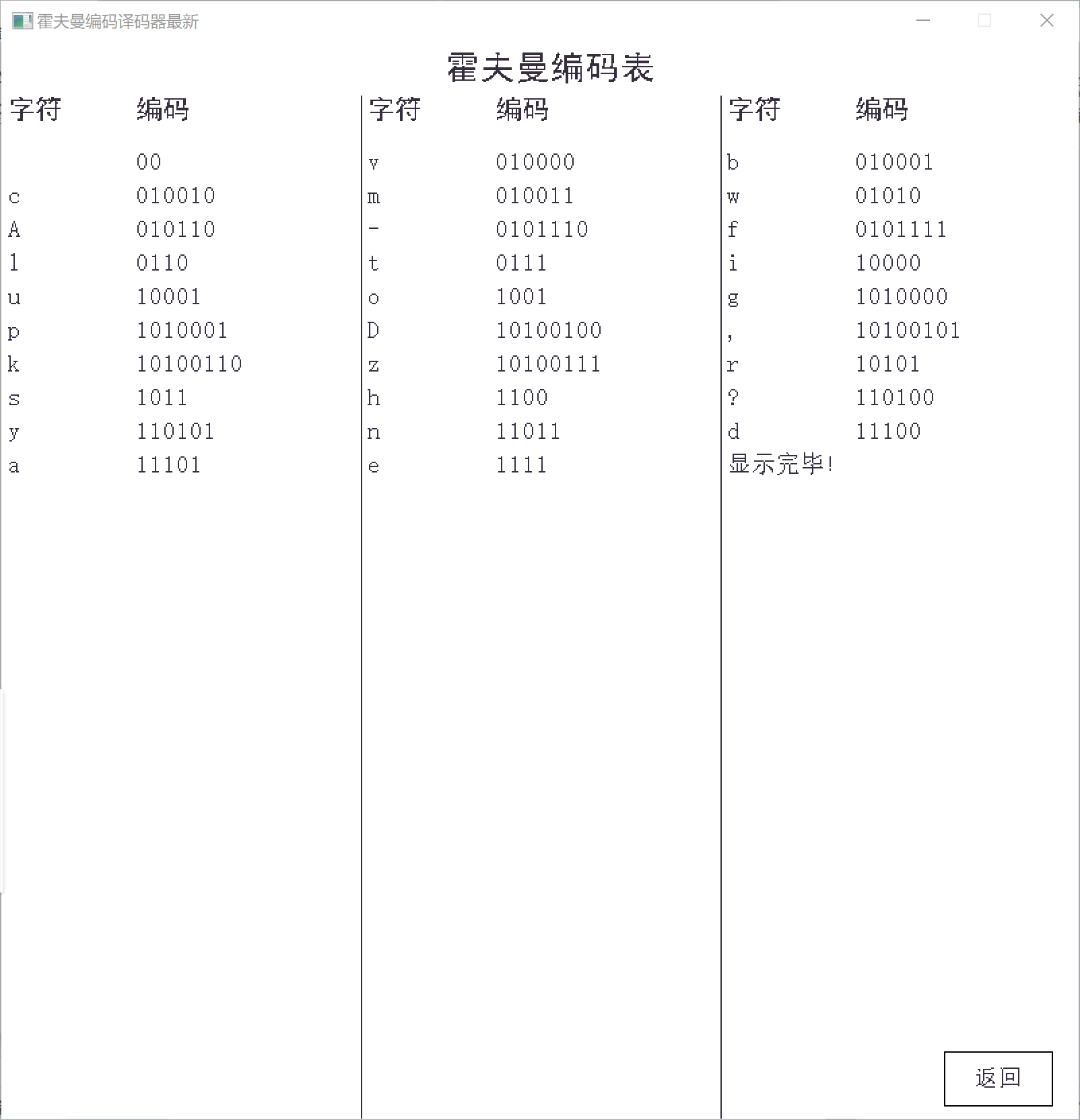


图11：字符霍夫曼编码图形化显示

如果在菜单中选择了“2”“获得编码文件”，则会显示如图12所示的界面。在中间的灰色文本框内输入希望输出后的文件名称，按下OK键，即可将按照上面的霍夫曼编码编码的源文件输出到exe文件夹下。

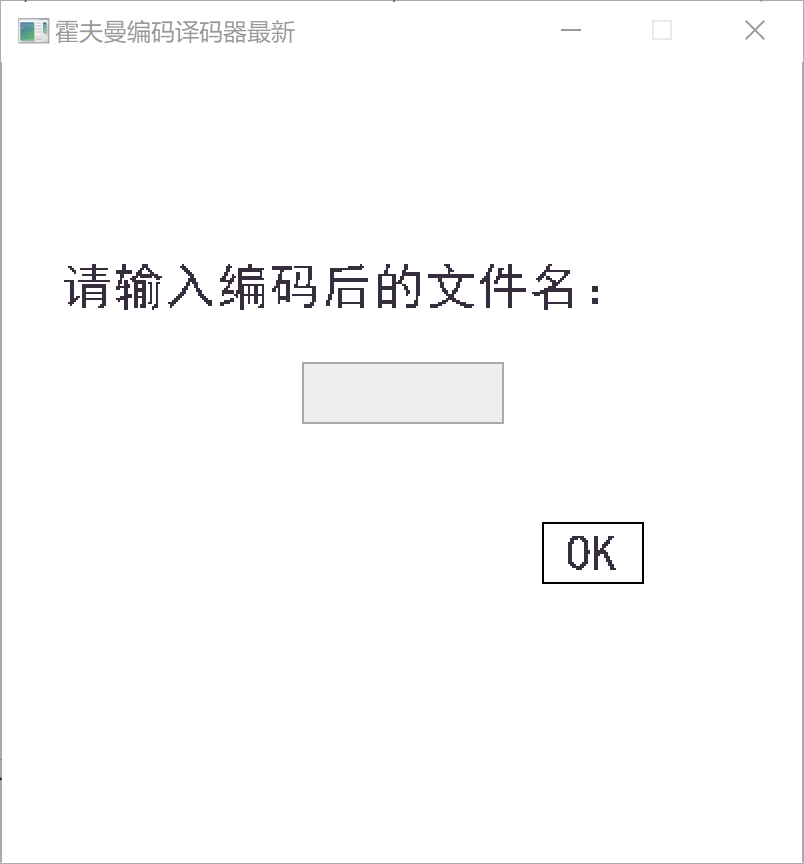


图12：获得编码文件界面图

如果在菜单中选择了“3”则首先初始化界面，使得界面变大，然后开始绘制哈夫曼树，如图13，按“返回”可以返回菜单。

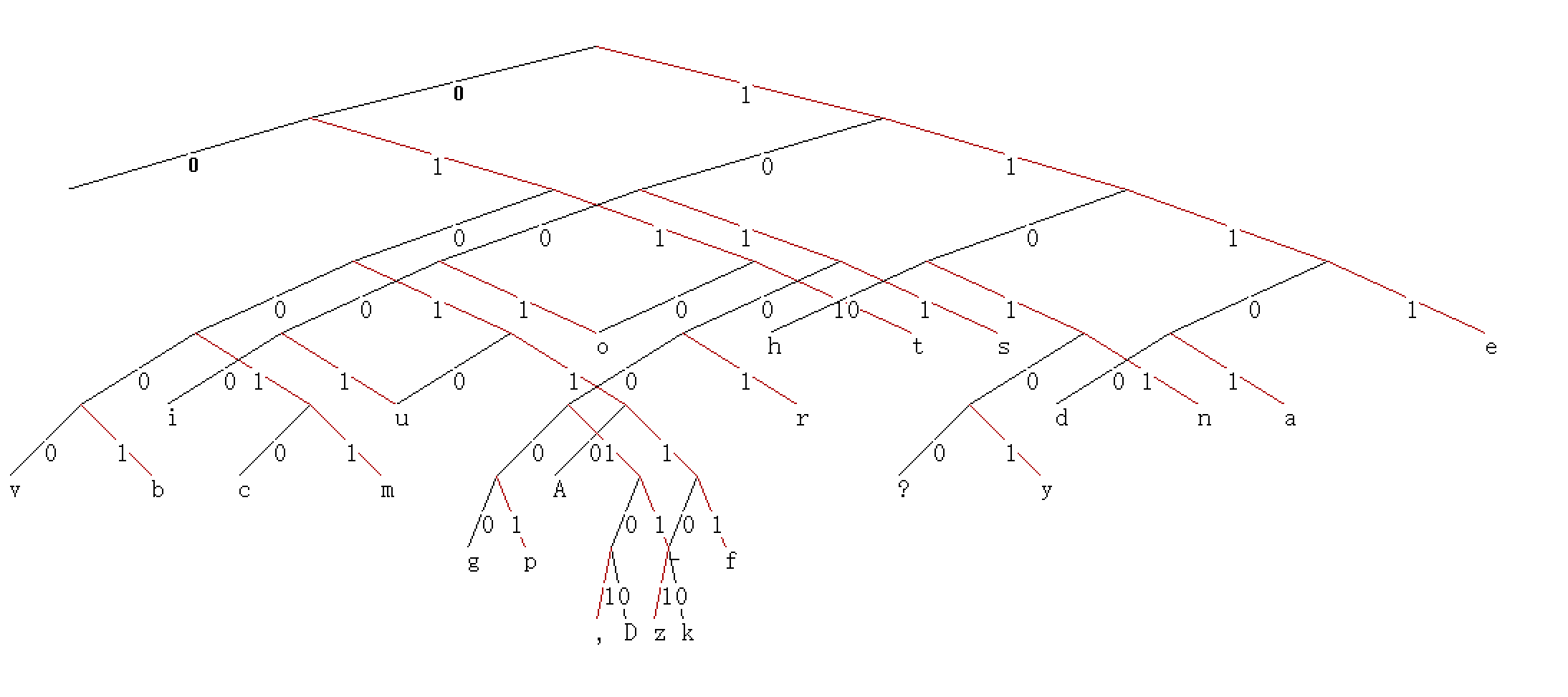


图13：霍夫曼树图形化绘制

在菜单中选择“4”可以对通过这套系统编码后的仅含有二进制的文件进行译码，他一共有两种选择，手动输入文件或者是读入外部文件，然后进行译码。用户可以自己选择输入方式。同时也可以将译码后的文件输出到外部保存。

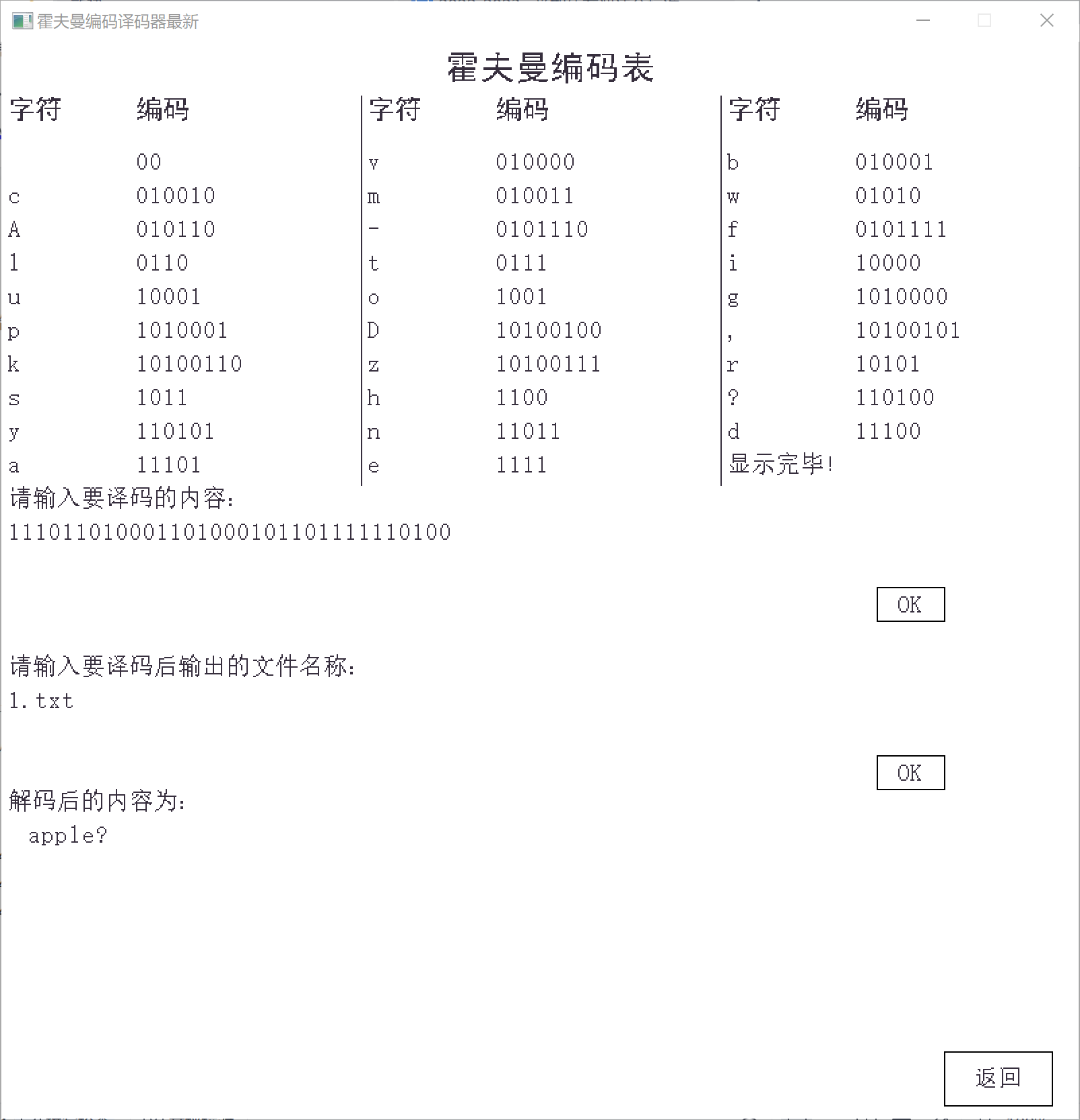
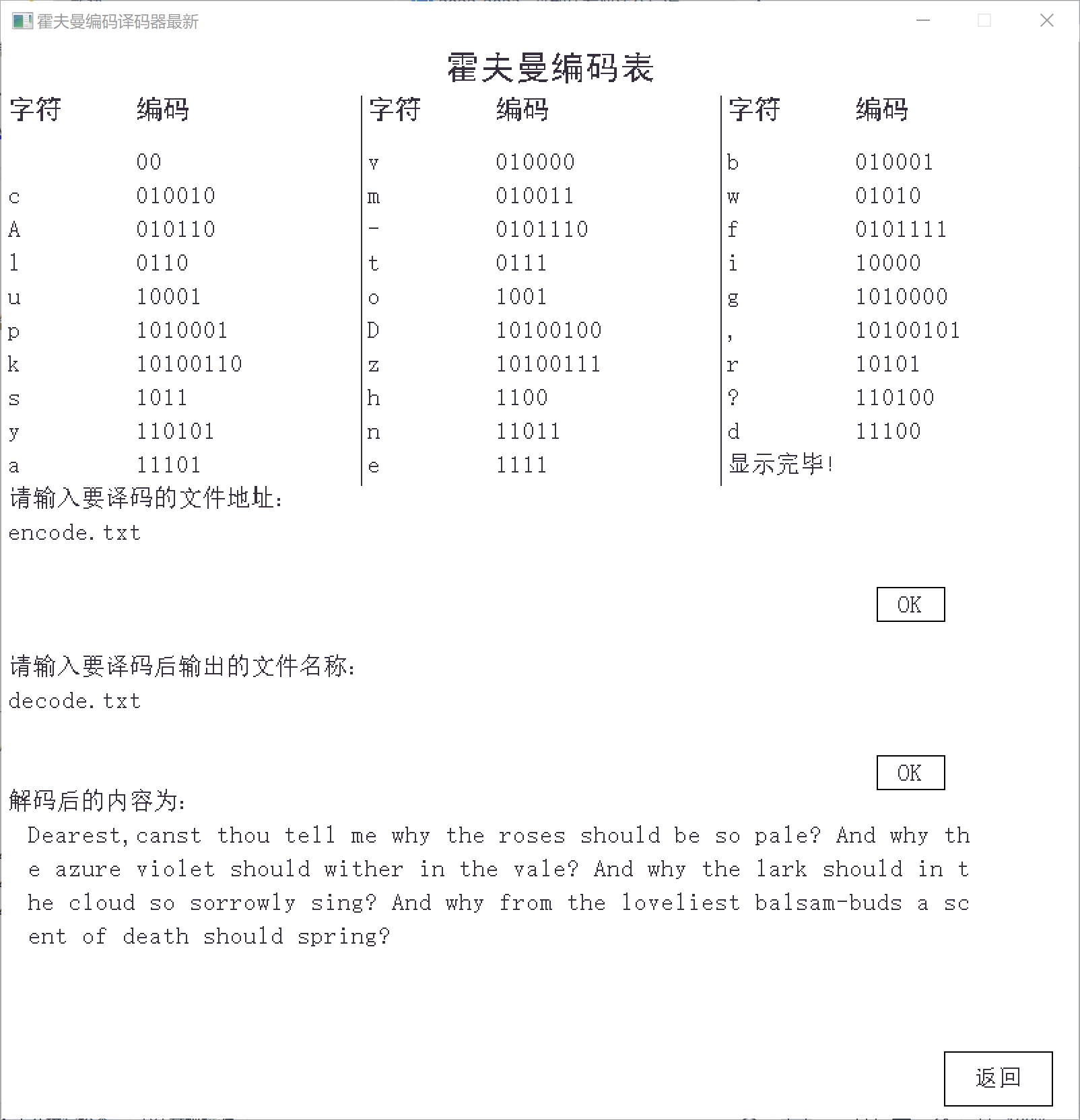
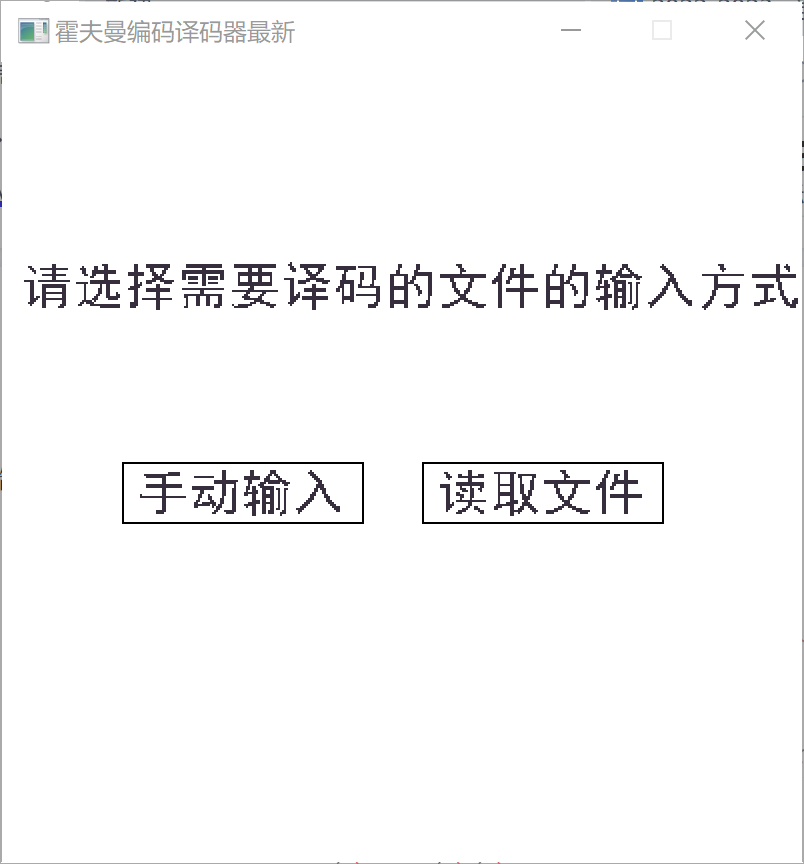


图14：霍夫曼译码图形化界面

在菜单中如果选择“5”则会让用户重新输入一个文件，针对这个文件建立一个哈夫曼树，进行新一轮的操作。

### 4.4 easyx实现文本框输入

在easyx图形库中本是没有文本框输入的控件的，为了实现和用户较好的交互效果，我们使用了一个叫EasyTextBox的类来实现文本框效果的实现，将用户输入的内容实时的展现在文本框中，有光标闪动，并且能保存用户输入的内容。该类的结构如图6 EasyTextBox类设计图所示。

在使用该类进行文本框构建的时候，只需要先创立一个EasyTextBox对象，然后使用Create设置其所在位置，以及最多容纳的字符数量即可在画面对应位置生成一个文本框。Check()函数用于感应该文本框范围内是否接收到鼠标的点击，如果接受到了鼠标的点击，则开始输入模式，调动OnMessage()进行文本输入。最后可以通过调用Text()来查看文本框内输入的内容。

### 4.5 easyx实现按钮部件

跟文本框控件一样，在easyx中也同样没有给出按钮部件，这也是需要我们自己添加的。所以在我们的程序中又加入了EasyButton这个类，来实现按钮的功能，从而增强用户交互体验。EasyButton类设计如图5.

同样，用户创建了对象之后利用Create将按钮呈现在屏幕上，然后通过Check来判断鼠标点击的位置，如果点击了按钮，那就进入OnMessage()函数，执行userfunc。

下面是在“输入需要编码的文件”这一功能中使用该文本框和按钮的示例代码：

由于Text()函数返回的是wchar\_t\*类型的，而filename要求是string类型，所以最后还需要将其转换成string类型再返回。

### 4.6霍夫曼树绘图

对于霍夫曼树的绘图，我采用的是对树进行前序遍历，一边遍历一边画图。在遍历到每个节点时先判断其是否是叶子节点，如果左边还有节点，那就计算出左孩子节点的坐标，并画出线连接左边的节点，然后继续向左遍历，直到遍历到叶子节点，则不画线，打印给出叶子节点对应的字符信息。接下来向右边遍历，如果右边还有节点，那就先计算右孩子节点的坐标，并画线连接右孩子节点，然后继续向右边遍历，直到遍历到叶子节点，输出字符信息并返回。

就这样一边遍历一边画图就形成了一棵哈夫曼树。

然而，为了使得每一个字符节点能够尽量不重叠，我引入了一个变量dis来调整每一次画树枝的时候线的倾斜程度。每次遍历到更深一层的节点，dis就增加1，从而使得计算下一个节点的时候x增加的更少，就可以使得每一个叶子节点尽量不出现重叠的情况。具体流程如图15

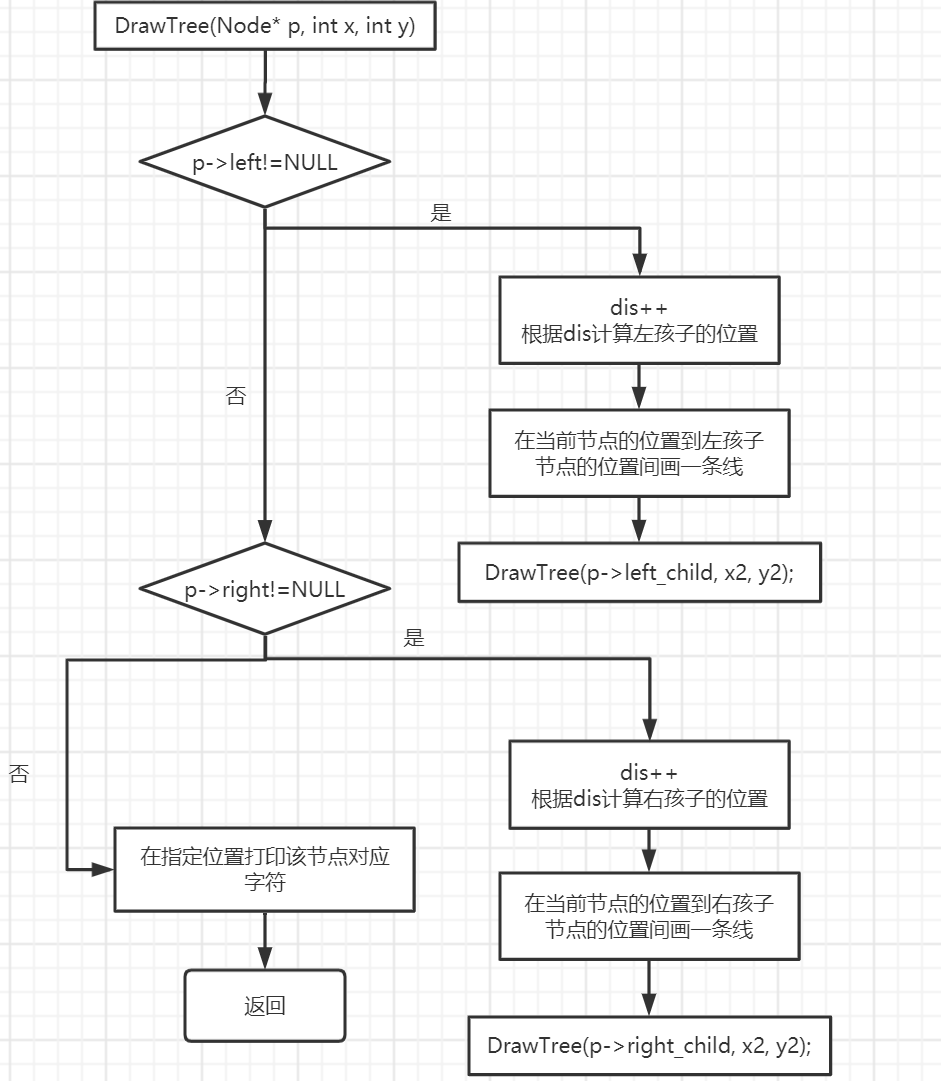


图15:霍夫曼树绘制流程图

# 5.项目测试

## 5.1功能测试

### 5.1.1 文件输入及字符统计功能测试

测试用例：输入文件try2.txt，在data.txt中查看字符统计如表1所示

经人工比对以及程序设定比对，统计结果符合预期

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | **b** | **c** | **d** | **e** | **f** | **g** | **h** |
| **25** | 6 | 3 | 12 | 44 | 9 | 6 | 17 |
| **i** | k | l | m | n | o | p | q |
| **22** | 1 | 14 | 12 | 20 | 18 | 7 | 1 |
| **r** | s | t | u | v | w | y | z |
| **22** | 17 | 26 | 20 | 6 | 5 | 18 | 1 |
| **A** | B | I | M | , | . | (空格) | q |
| **2** | 2 | 2 | 1 | 5 | 4 | 74 | 1 |

表1：try2.txt字符统计

### 5.1.2 霍夫曼树的构建及图形化测试

测试用例：输入文件try2.txt，得到霍夫曼树并图形化如图16。

测试结果符合预期

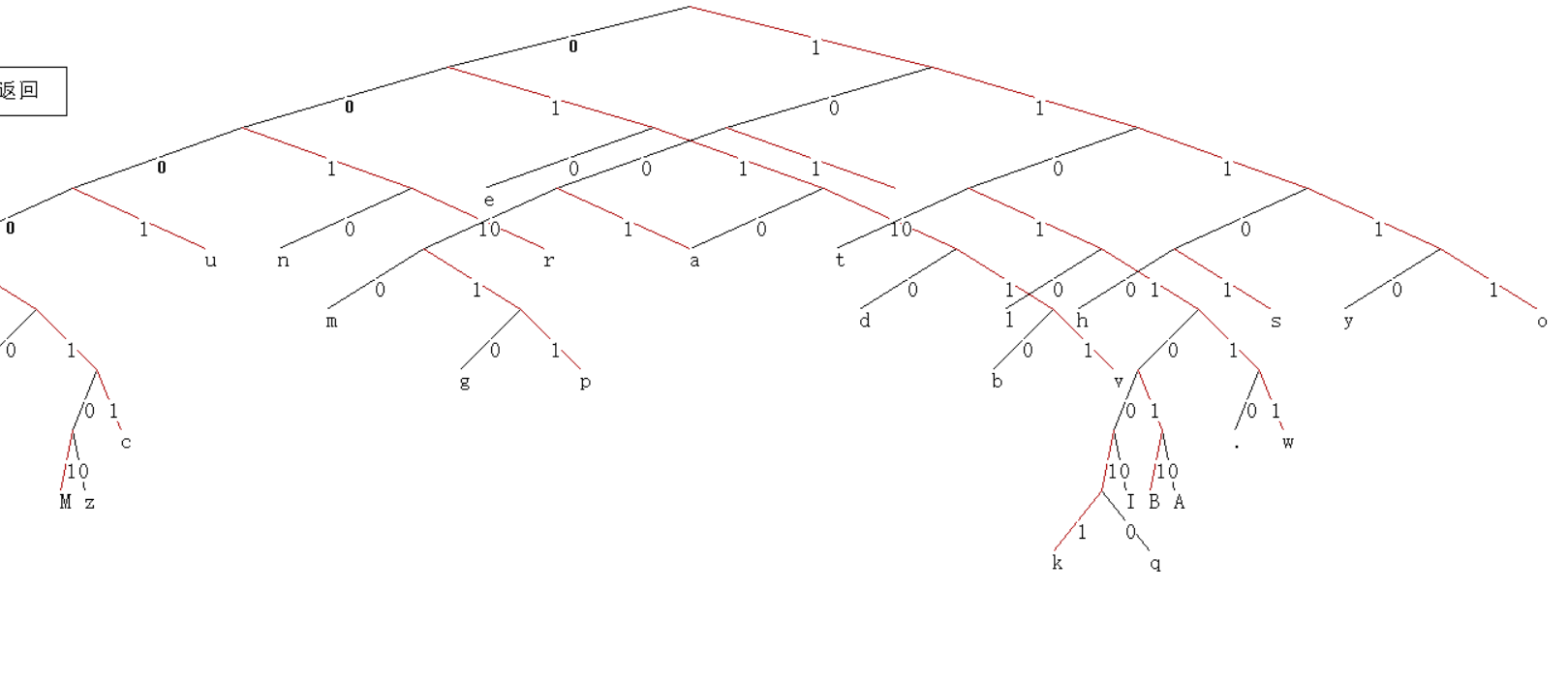


图16：霍夫曼树构建及测试

### 5.1.3 霍夫曼编码以及图形化测试

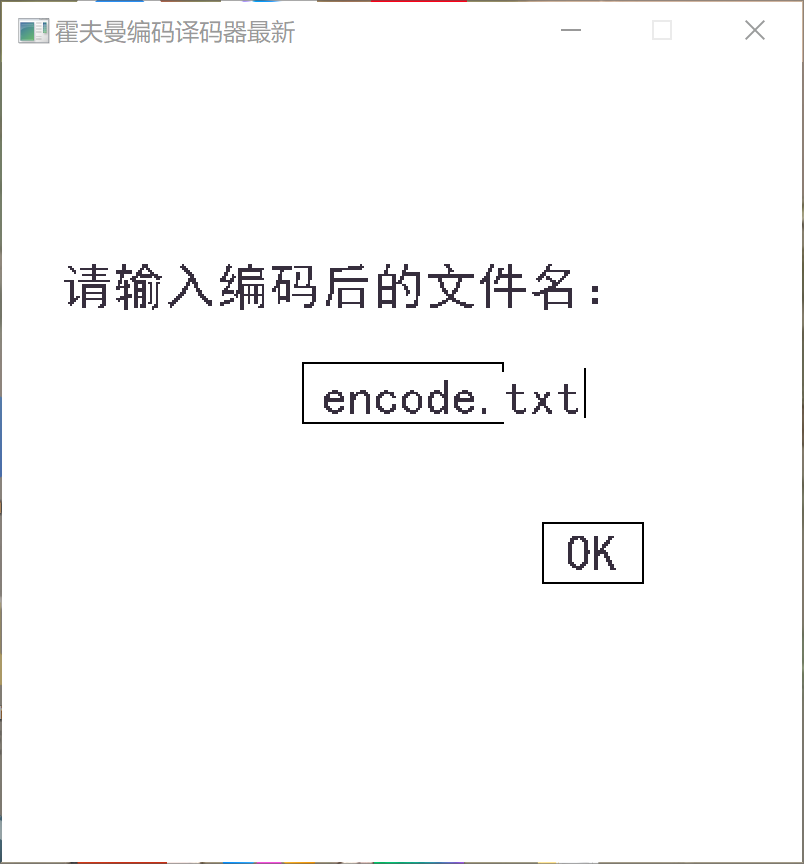
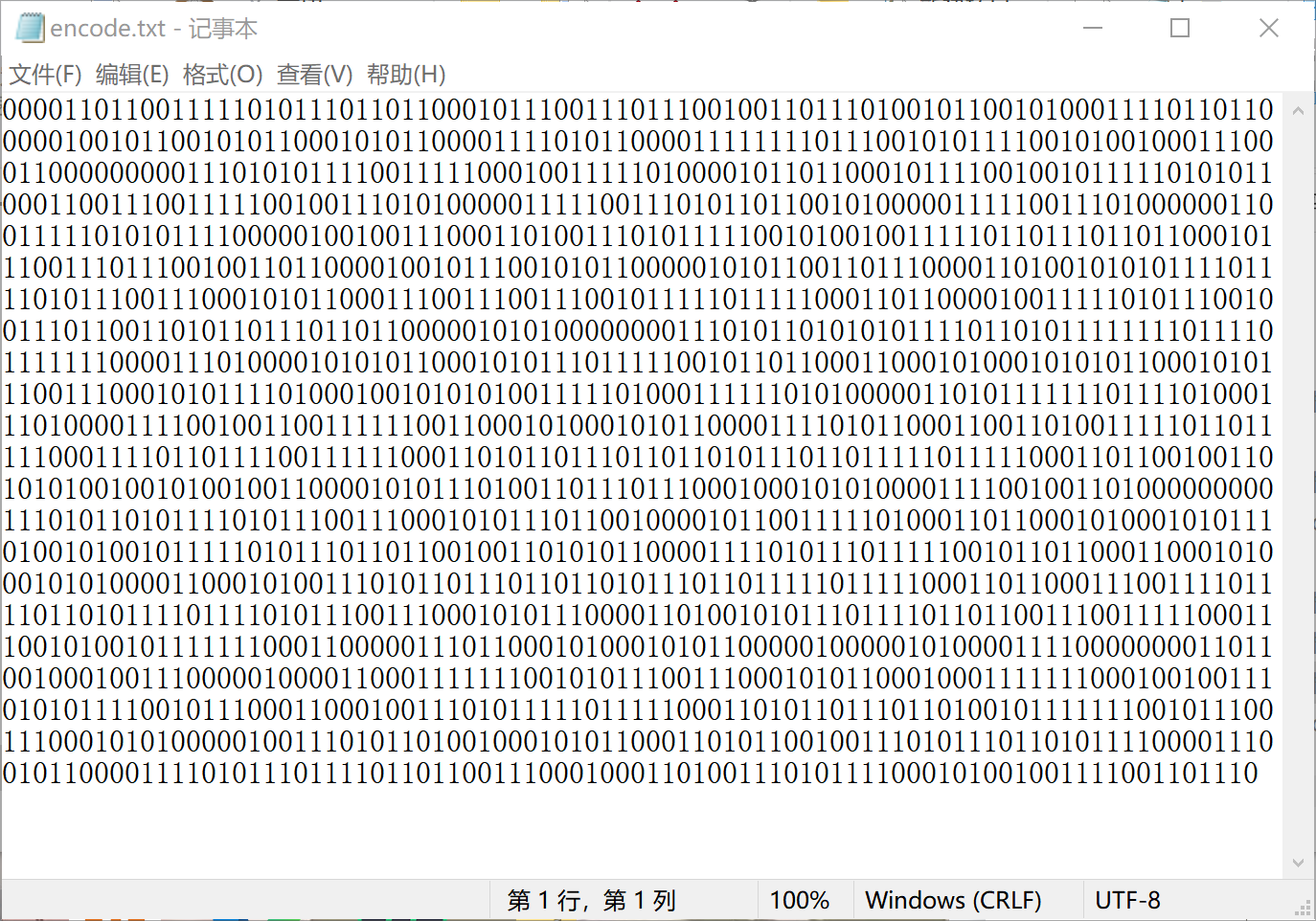
测试用例：任意输入文件try2.txt，菜单项选择“1”,得到文件字符的霍夫曼编码并图形化如图17



图17：霍夫曼编码及图形化测试

### 5.1.4获得输入文件对应的编码文件

在上面5.1.3输入文件try2.txt并且进行每个字符的编码显示的基础上，按下“返回”按钮，回到“菜单界面”。选择2，获得try2.txt利用其生成的霍夫曼编码全文编码后的文件encode.txt。操作以及获得的编码文件如图18。

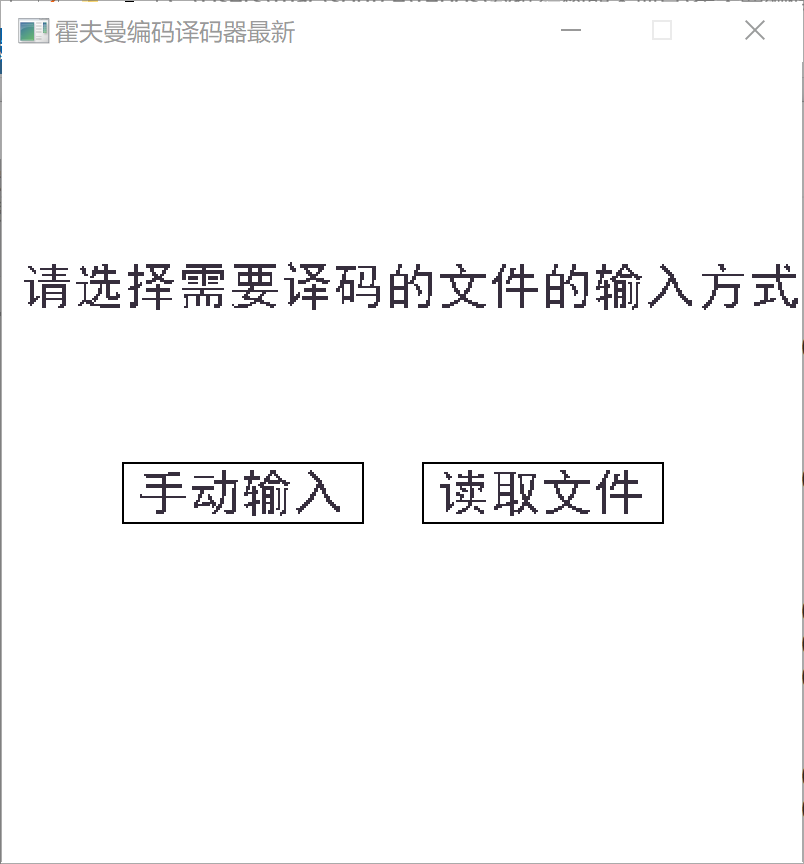
 

**图18：获得输入文件的编码后的文件**

### 5.1.5译码功能测试

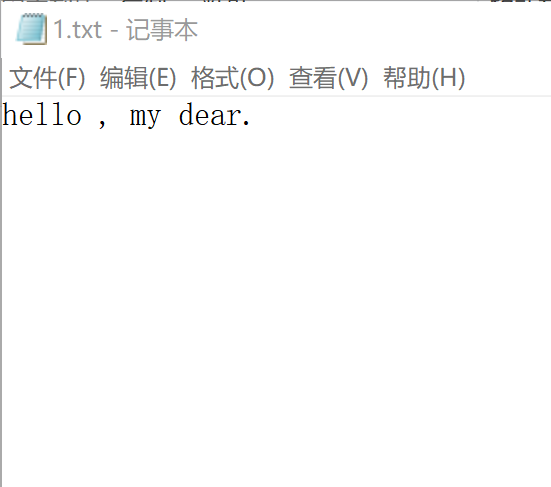
在上面5.1.4的基础上，按下OK即可生成编码后的文件。文件生成成功之后会弹出界面提示文件已经生成成功。然后返回菜单界面，选择4,即可进行译码功能。

本译码器支持两种形式输入需要译码的文件，一个是“手动输入”，还有一个是“读取文件”。我们首先进行手动输入测试。



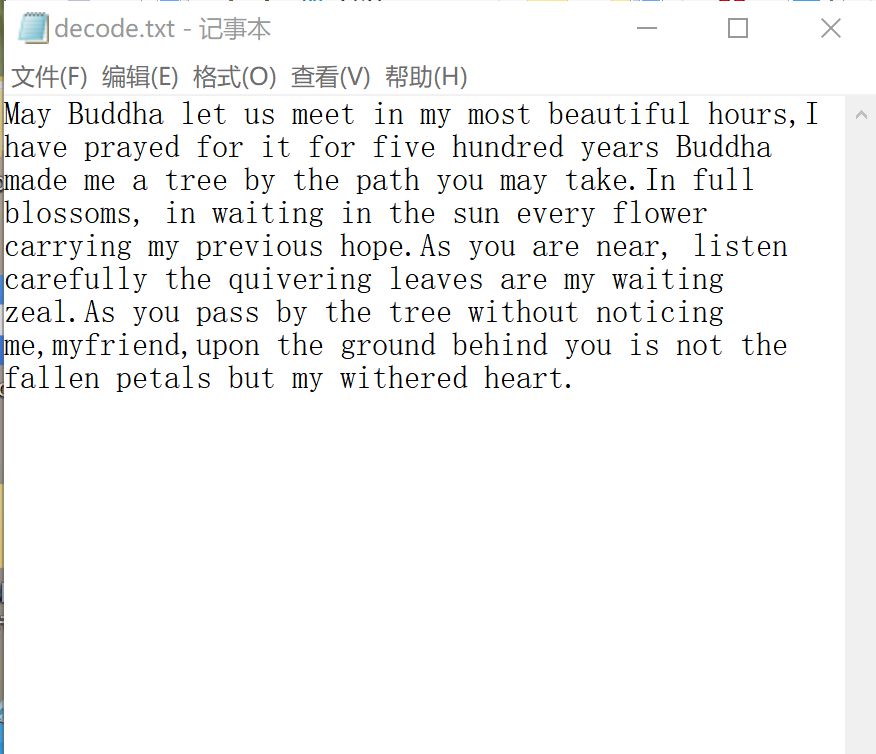
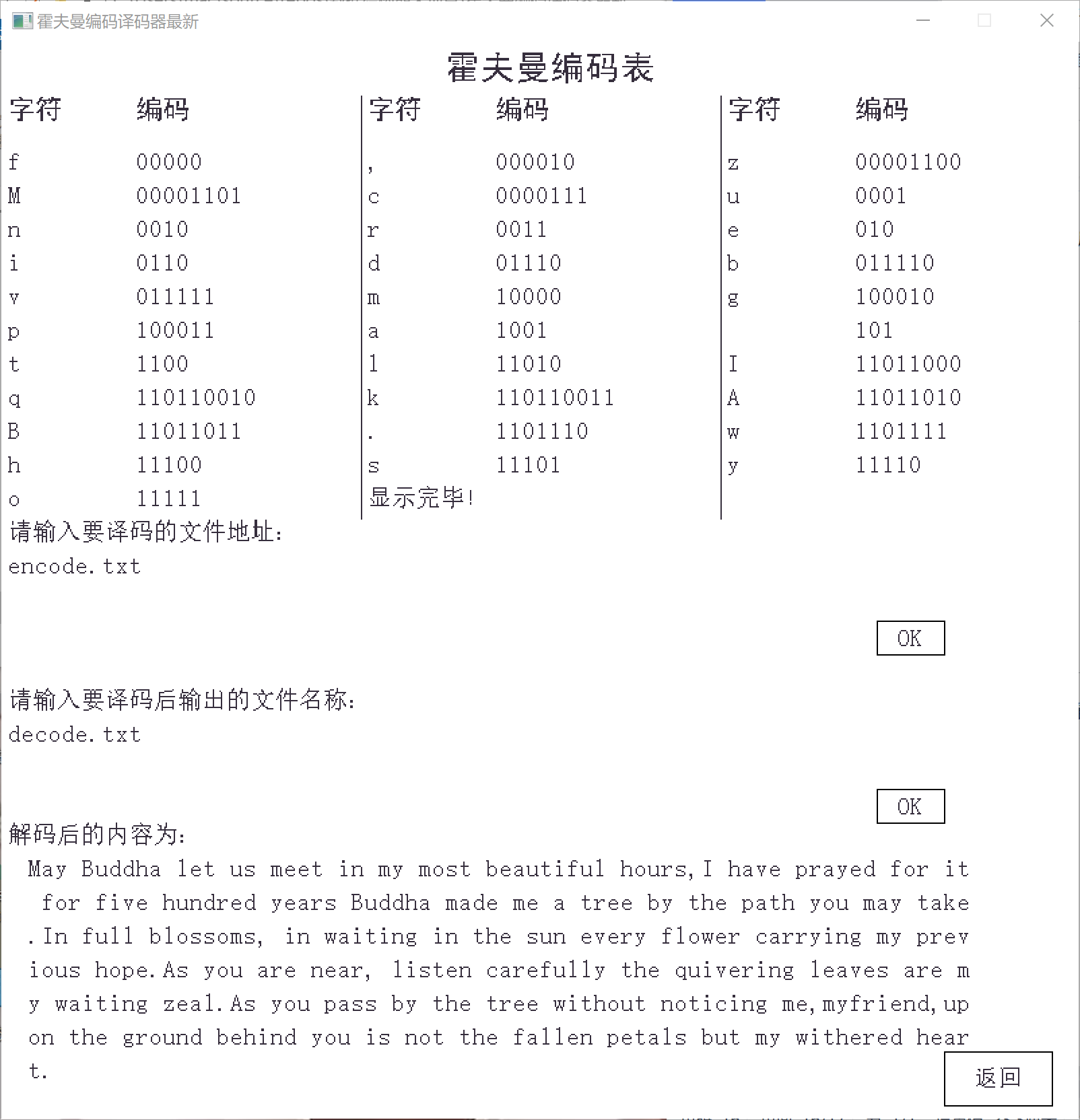
**图19：选择需要的文件输入方式**

在该选择界面鼠标点击“手动输入”，然后在文本框中输入需要译码的内容（如图20），按“OK”便可以将译码结果输出在界面上，同时也能输出到我们指定的文件中。



**图20：手动输入编码并将译码结果输出到指定文件**

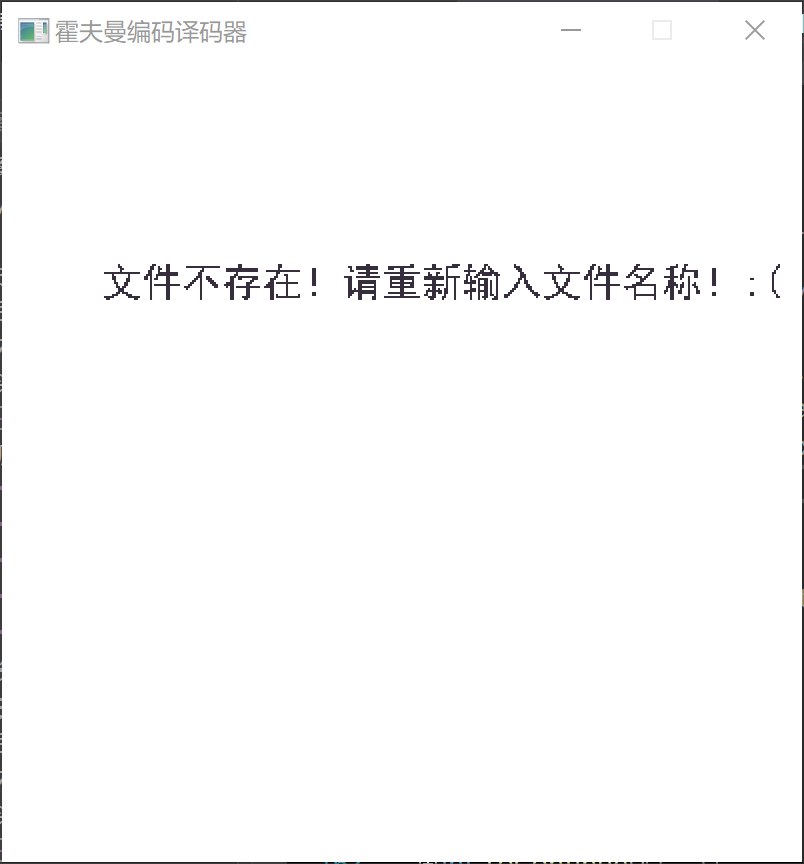
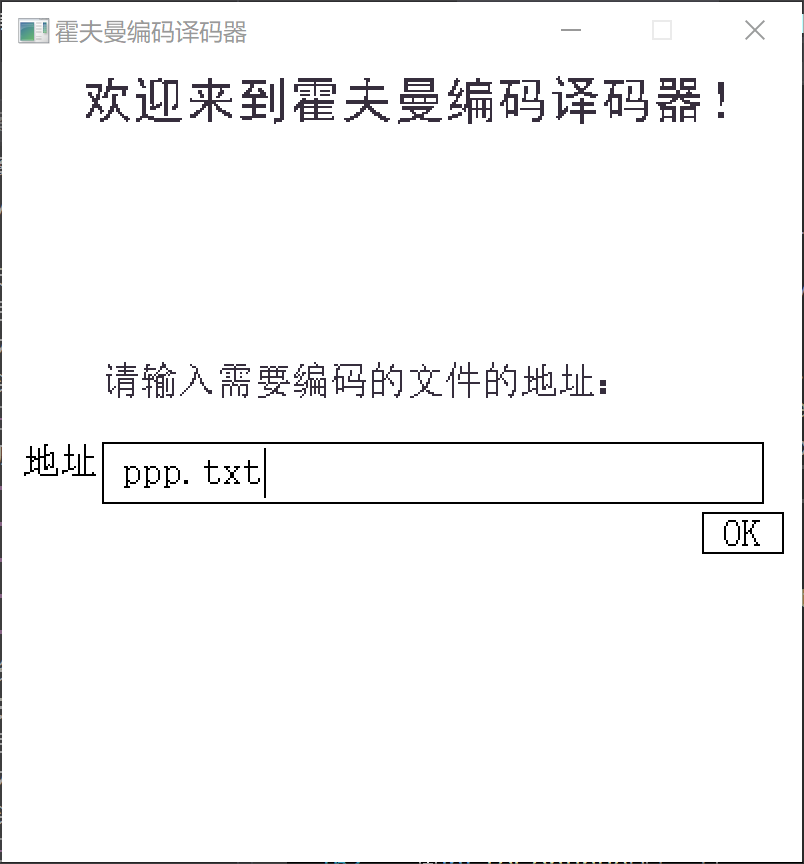
我们也可以在选择界面选择“读取文件”，从外部读取用该套霍夫曼编码得到的编码文件，然后将文件内的编码翻译出来，在屏幕上显示，并且将译码内容输出到指定文件保存。我们这边选择输入encode.txt，即之前的try2.txt用霍夫曼编码编码后的文件，译码之后将内容输出到decode.txt，输出完成后打开decode.txt与try2.txt进行比较，发现内容完全相同。（如图20）



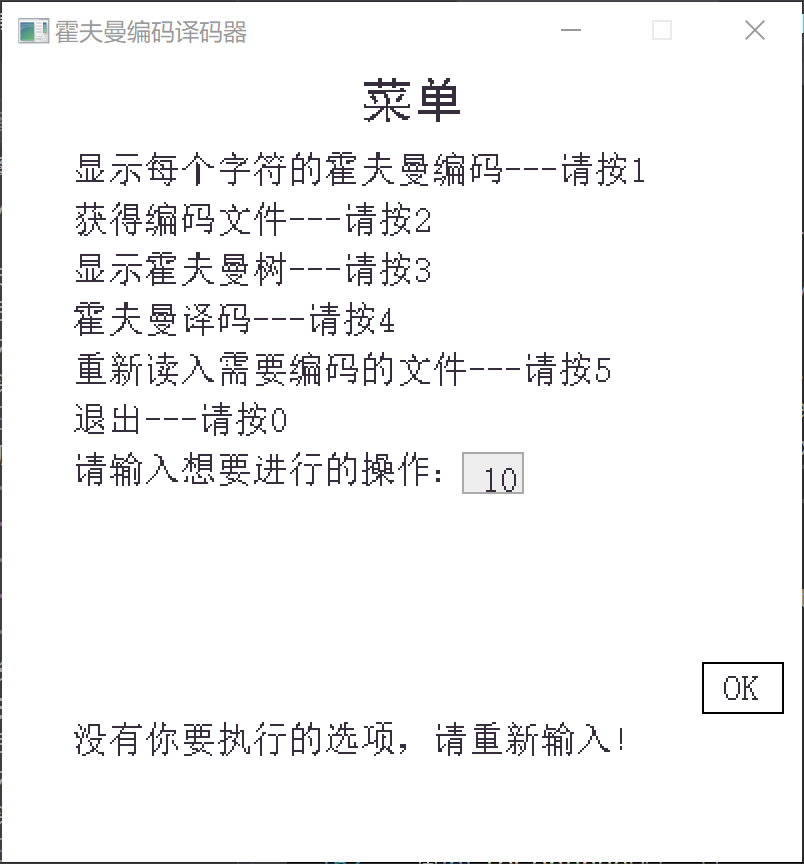
**图20：手动输入编码并将译码结果输出到指定文件**

## 5.2 错误和边界测试

# 考虑到读入文件的时候，用户需要手动输入文件名称，可能会有输入错误，从而导致输入的文件并不存在，那也就无法打开建立霍夫曼树。所以我们的程序在这里做了一些特判，如果输入的文件不存在，就给出提示，让用户重新输入。下图为输入不存在的文件然后给出提示让用户重新输入。



除此之外，用户也可能在“菜单”中错误的输入了不存在的选项，我们也在这里做了特判。如果用户输入不存在的选项会被要求重新输入。



# 6.项目说明

本项目是基于数据结构课程内容学习自行进行完成设计的，鉴于网上有许多类似的项目，我们并未采用照搬修改的方式进行改进，而是选择自己设计并于网络上的类似项目进行比对，得到**优点与改进**如下：1.采用了类进行封装，保护数据；2.霍夫曼树的实现当中采用了堆进行构建；3.图形化界面有利于用户操作；4.文件流的方式进行操作有利于存储和传输。

同时，因为Easyx这个图形库有许多功能是不具备或者说不如人意的，因此在设计当中我们还自己设计了类进行文本框的输入和按钮设置，提高用户使用的交互体验感。

本项目对于较多字符的霍夫曼树都是可以实现绘制的，但是当字符数量大到一定程度时，绘制会超过屏幕界面。同时经过队伍讨论，霍夫曼树的完美绘制位置和路径应该需要经过数学推导，调整其高度和宽度。

对于多字节字符输入，以及更多字符兼容性问题，在一开始做项目时有欠缺考虑的地方，感谢张颖老师在项目展示时给我们提出的建议。后来经过课后小组讨论与反思，一致认同可以另外增加哈希表的数据结构来存储从文件中读取的各类字符，从而支持更多类型字符的编码。目前项目仅支持输入英文文件进行编码与译码，这是我们课后需要改进的地方。