基于哈夫曼编码原理实现的一款简单文本文件压缩软件

**Ⅰ 作业要求**

根据哈夫曼编码原理，实现一款文本文件的压缩软件，要求具

有压缩和解压功能。简述：

(1)实现思路；

(2)采用的存储结构；

(3)编写代码实现压缩和解压（包括代码和运行结果的截图）；

(4)分析代码的时间复杂度和空间复杂度

**Ⅱ 作业简述**

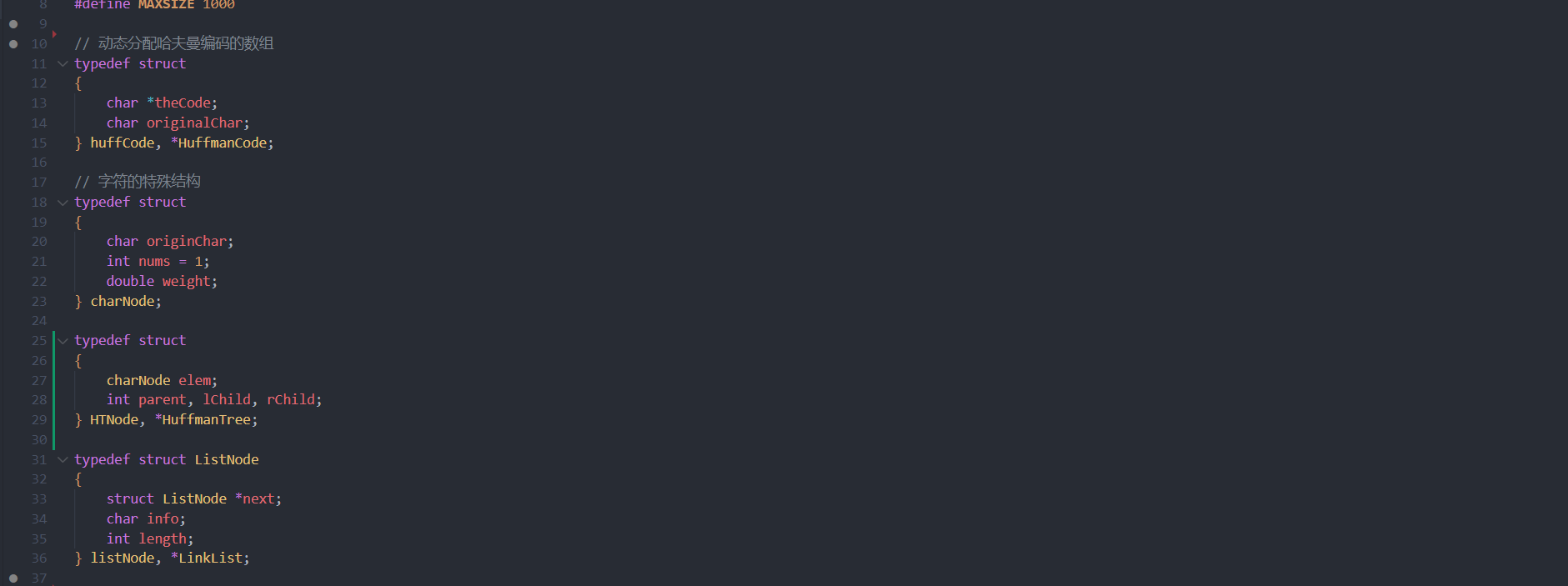
1. **实现思路：**
2. **前期的准备工作：**将根据哈夫曼编码原理，读取一个本地文件夹下的文本文件， 首先对文件里的所有文本内容进行分析，得到数据。数据内容包括，所有出现的字符类型，每个字符出现的相对应的概率，并以概率为标准，为每个字符进行合理的赋予权值，以求的文件压缩之后对应的二进制文件的长度达到最短，同时保存源文件中的内容为之后对文件进行压缩编码做好准备。
3. **构造哈夫曼树**：以上述得到的数据，开始构造哈夫曼树，将权值，字符值依次存进哈夫曼树中得到正确的哈夫曼树结构。
4. **构造哈夫曼编码表**：根据哈夫曼编码原则遍历得到的哈夫曼树，对每个字符进行编码，得到一个由该文件内所有字符组成的一个特定的哈夫曼编码表。保存编码表，为压缩文件编码做好准备。

4) **编码译码：**逐个遍历源文件中的各个字符，在构造好的哈夫曼编码表中找到相对应的哈夫曼编码，填入文件。遍历完毕，压缩文件操作完成。类似的，逐个遍历已经压缩好的文件中的内容，当在其中找到有效的哈夫曼编码之后，将其编码替换为对应的字符，实现译码，解压缩文件操作完成。

**（2）采用的存储结构：**顺序存储结构（哈夫曼编码表） 链式存储结构（哈夫曼树）。

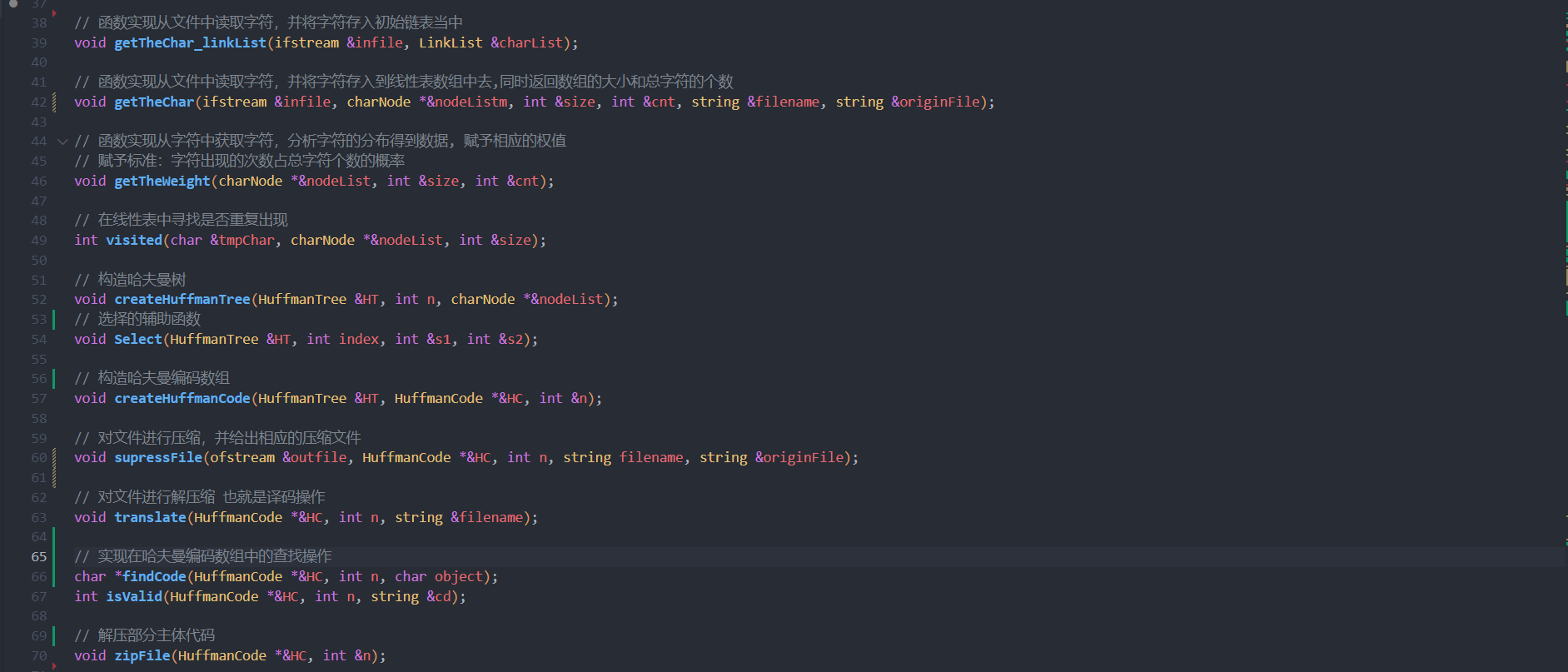
（3）**编写代码实现压缩解压缩**

* **代码基本结构体部分**



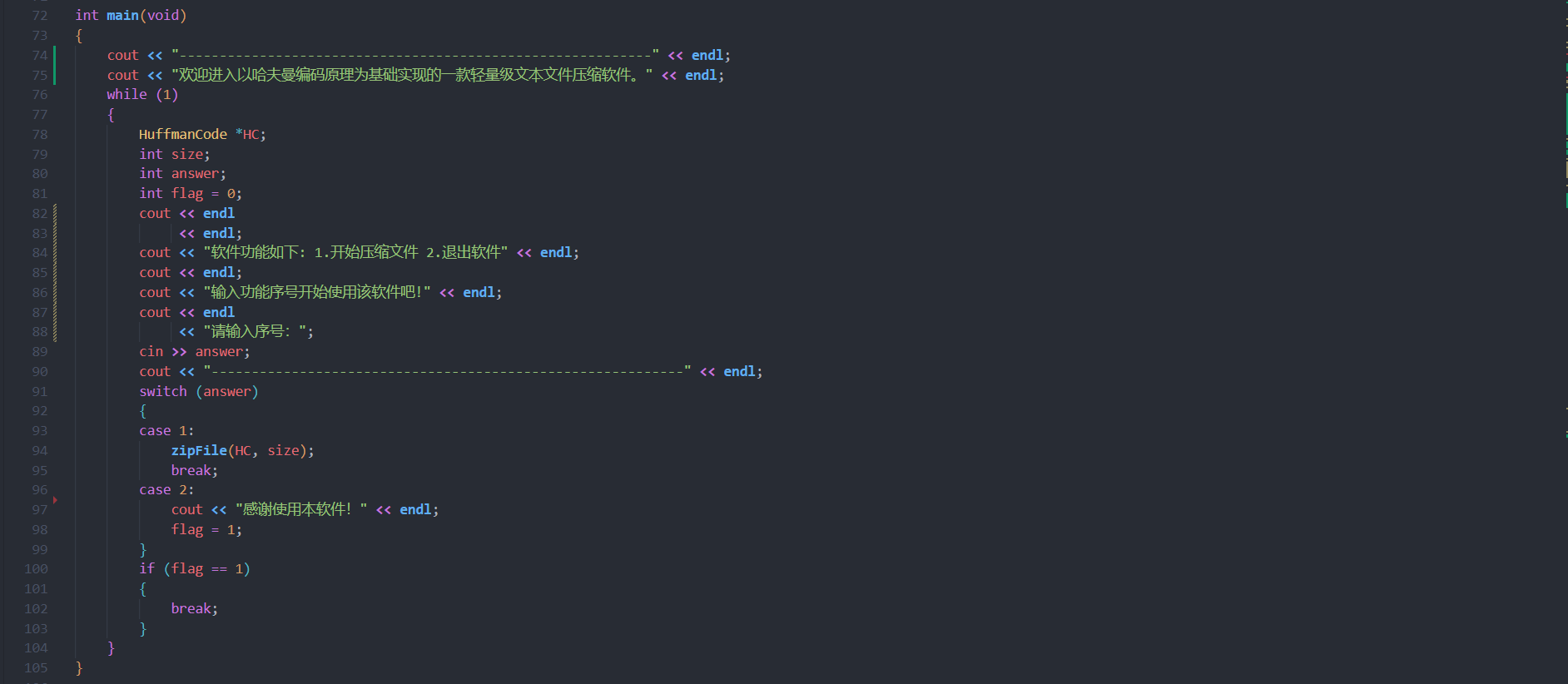
具体实现规定了，哈夫曼编码的数组，不仅包括哈夫曼编码，还包括了对应的字符，使得后续操作更加方便。以及哈夫曼树和树结点的创建。

* **代码主体函数部分**

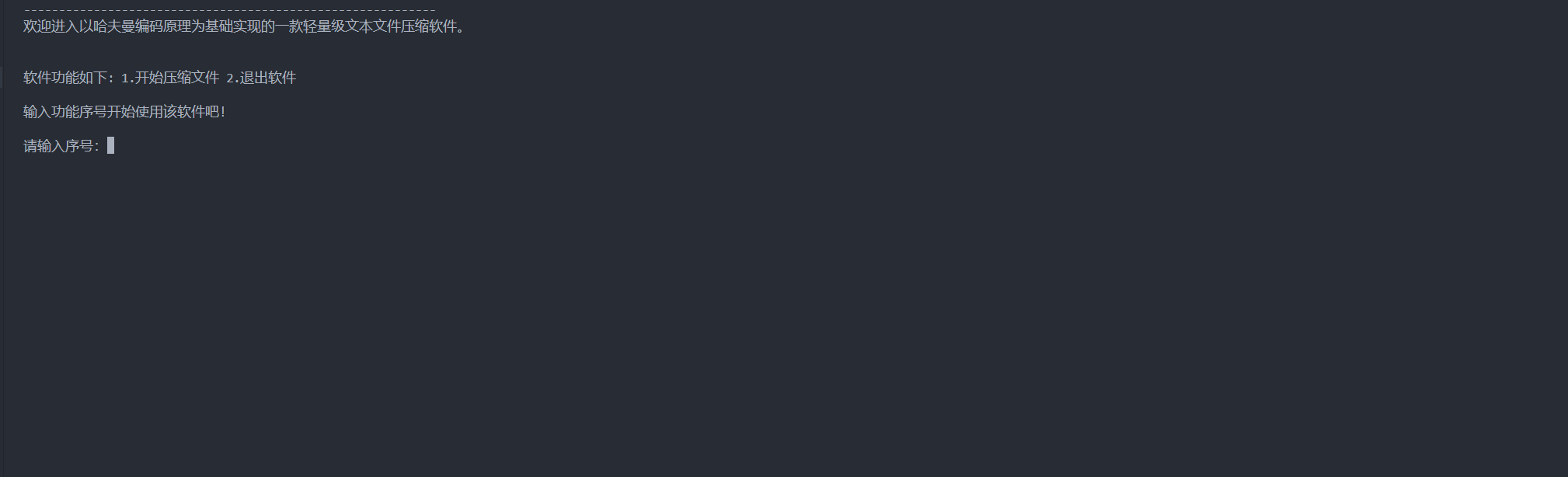
****

程序主要的函数声明，其中各个函数的功能以在注释中给出。具体实现将在下文给出。

* 主函数部分

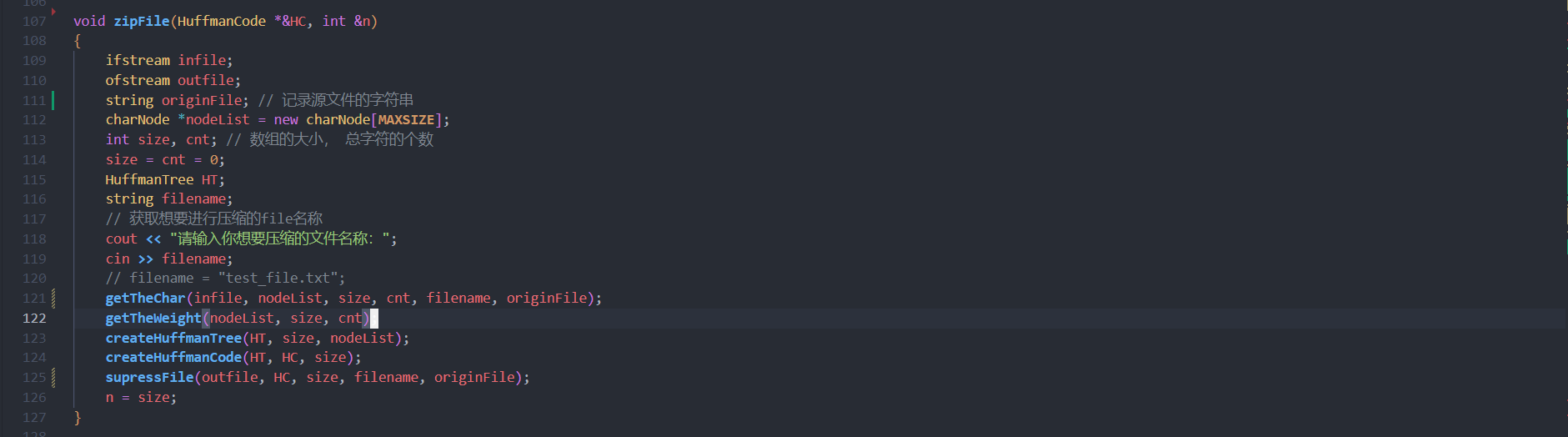


主函数主要实现了软件的用户欢迎界面。运行效果如下：

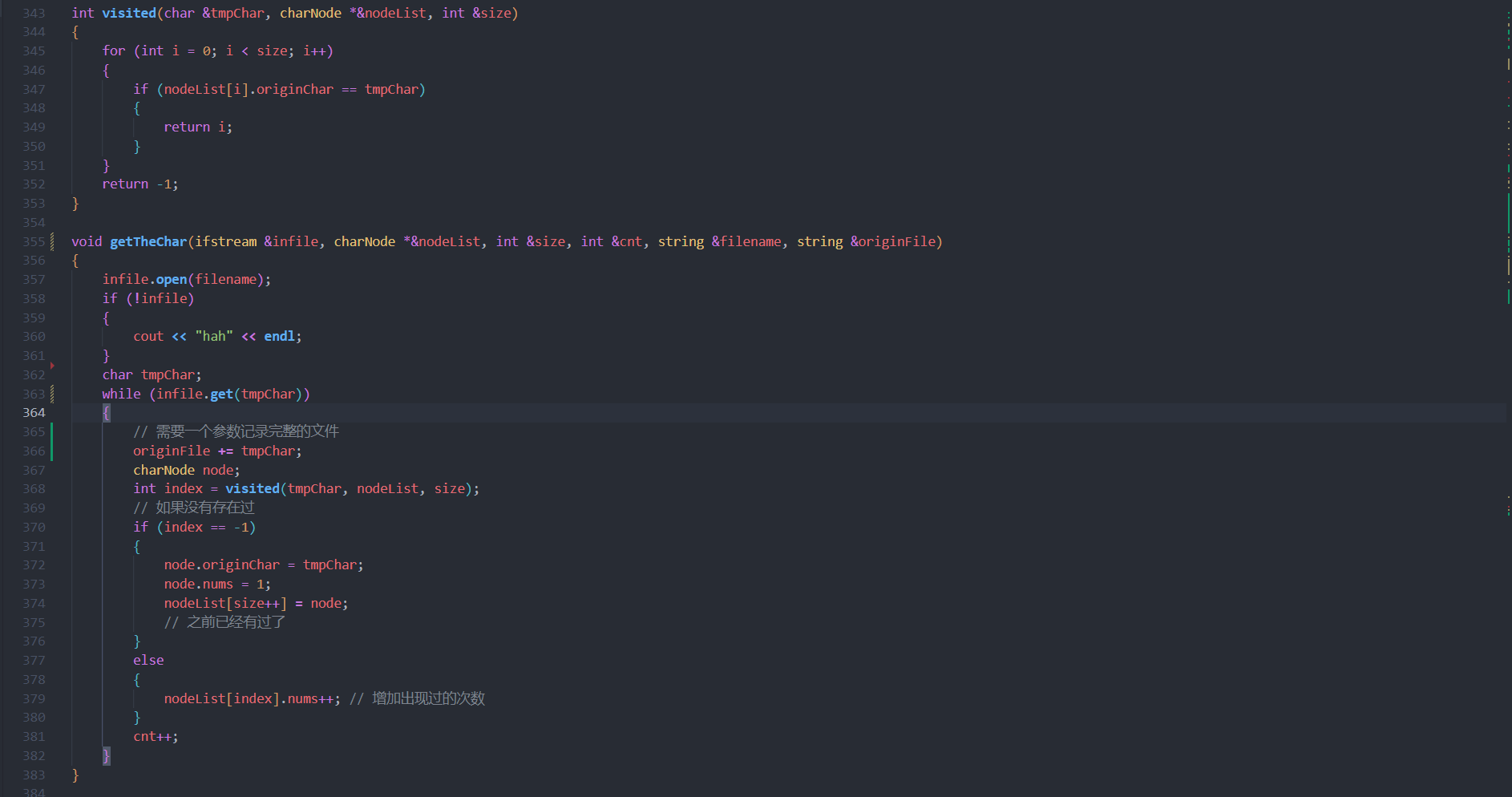


* **主要函数的实现**

压缩主体操作函数：



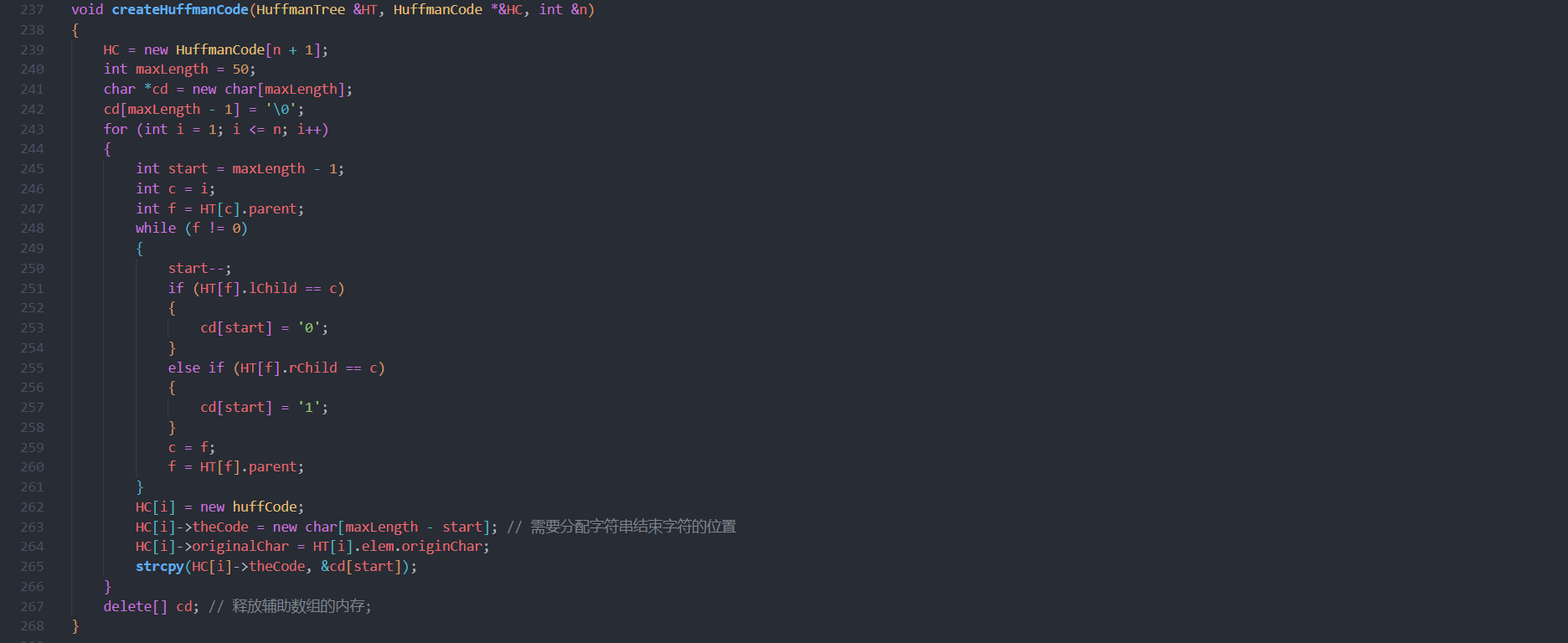
GetTheChar()函数的实现，主要得到了文件中的所有字符类型并记录。

****

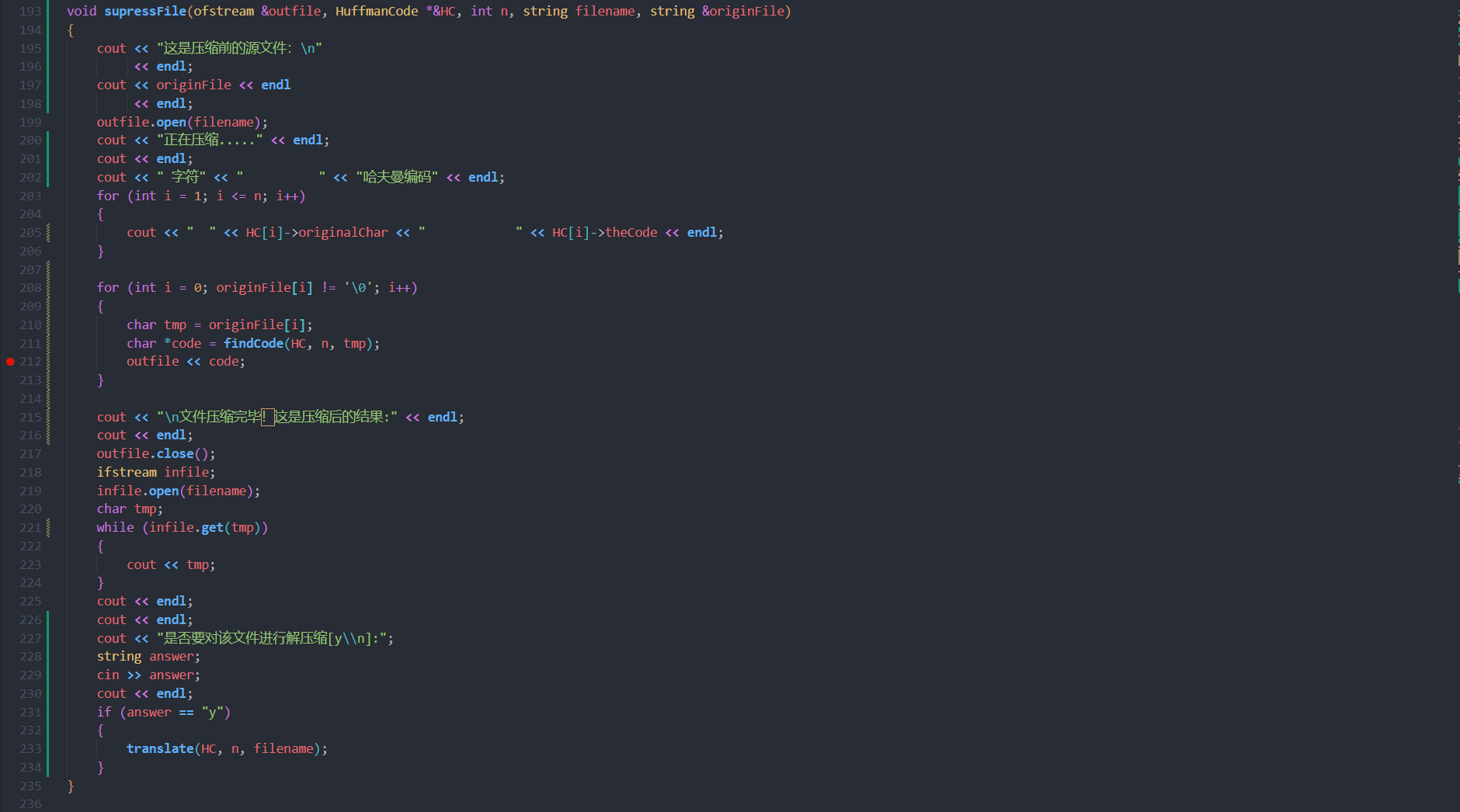
哈夫曼树创建算法的实现：



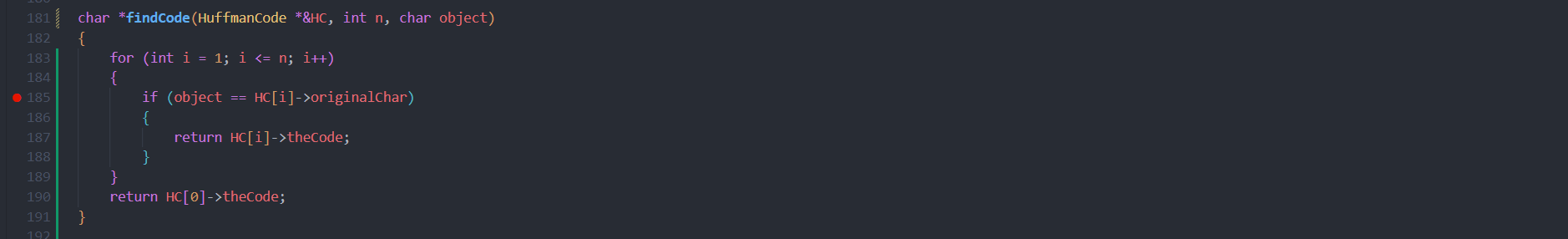
哈夫曼编码算法的实现：



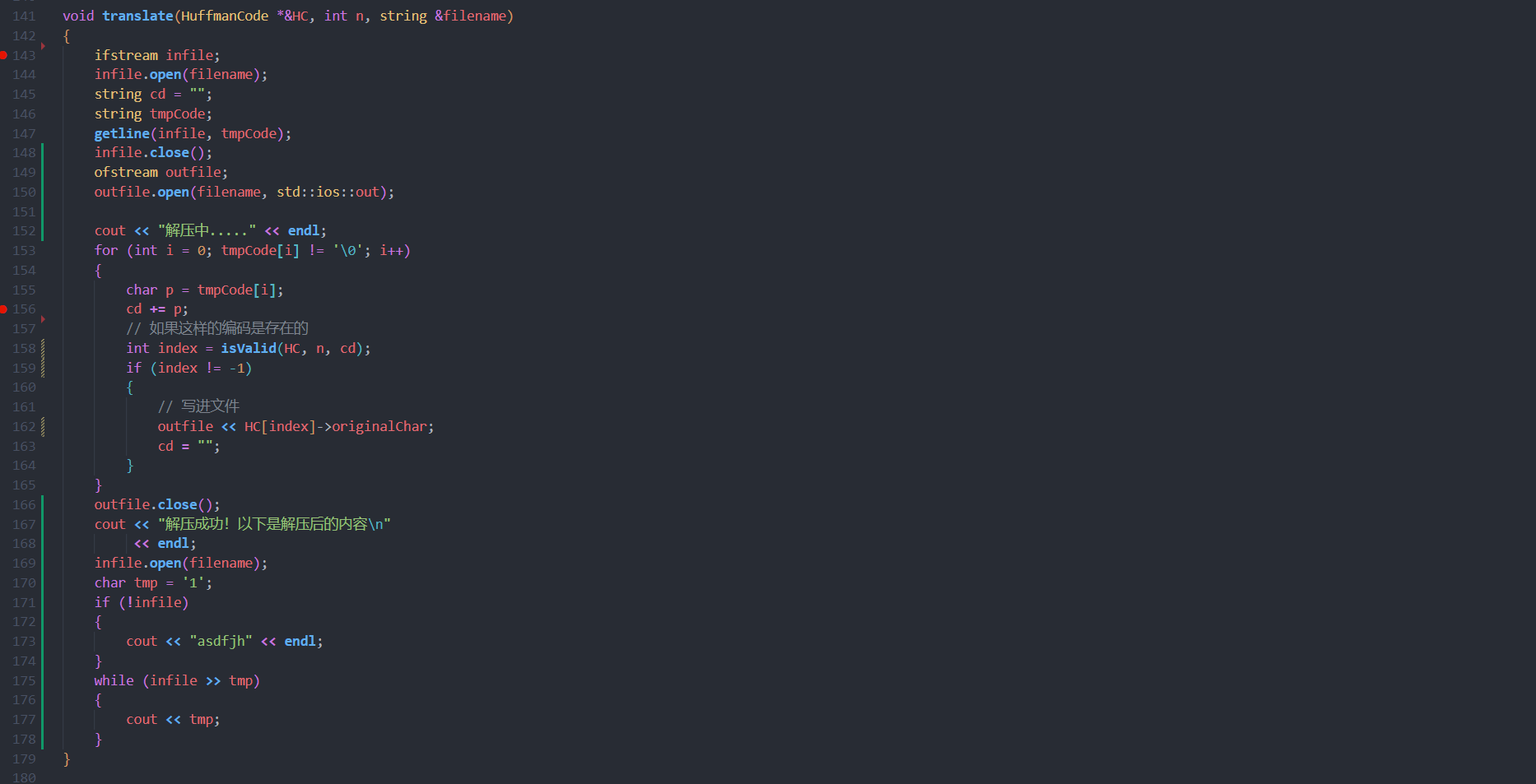
压缩文件的主体函数实现：



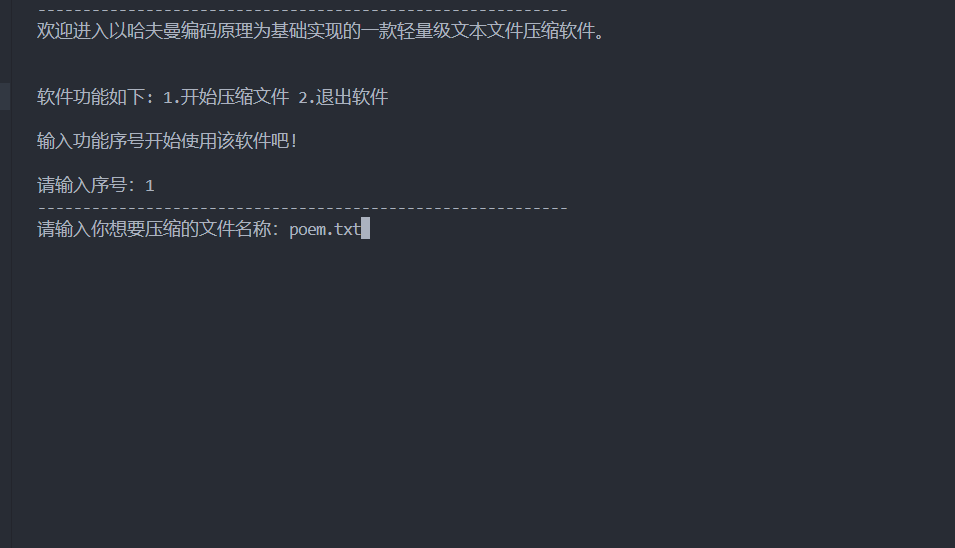
查找编码算法：



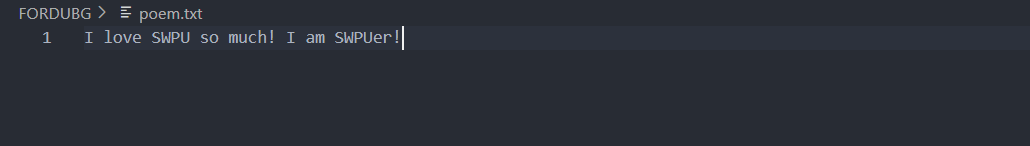
译码函数操作的实现：



* 代码运行过程和结果



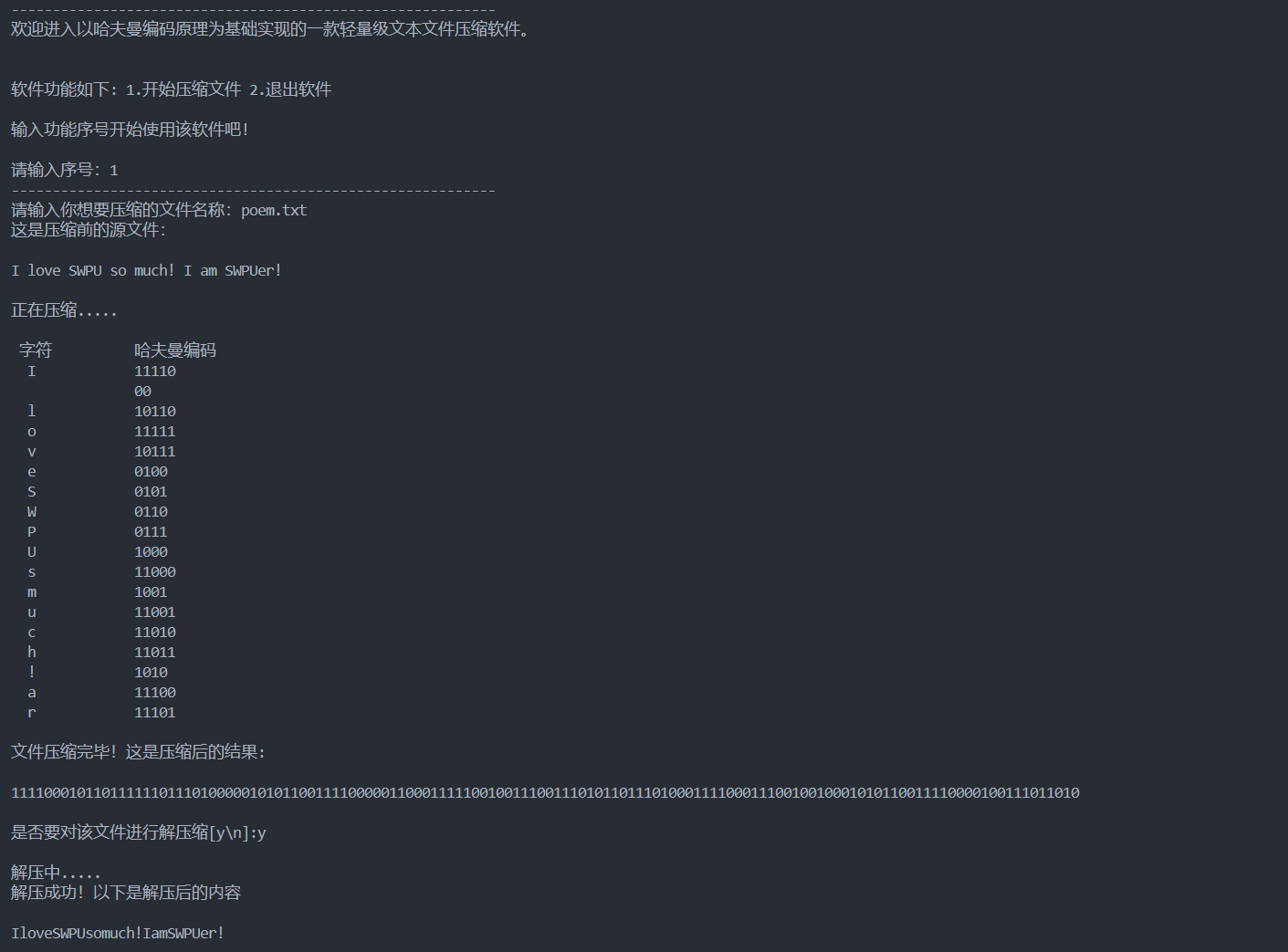
Poem.txt源文件如下：

****

压缩完成：

****

开始解压：



**退出软件：**

**(4)分析代码的时间复杂度和空间复杂度**

整个程序里存在着，多个顺序存储结构，包括存储字符的nodeList，存储哈夫曼编码的哈夫曼编码表数组HuffmanCode，下面将依次根据各个函数来计算时间复杂度和空间复杂度。

**getTheChar()函数：**

**时间复杂度:**

- 遍历文件中的所有字符: O(n)，其中 n 是文件中的字符总数。

- 检查字符是否已经存在于 nodeList 中: O(size)，最坏情况下是 O(n)。

**空间复杂度:**

- nodeList 数组大小: O(n)，存储所有不同的字符信息。

- originFile 字符串的空间复杂度: O(n)，存储文件的全部内容。

getTheWeight()函数：

**时间复杂度:**

- 该函数只需遍历 nodeList 一次，因此时间复杂度为 O(size)，最坏情况下为 O(n)。

**空间复杂度:**

- 不需要额外的空间，空间复杂度为 O(1)。

**CreateHuffmanTree函数：**

**时间复杂度:**

- 总共有 2n-1 个节点，每次选择两个最小权重的节点进行合并。选择两个最小权重的节点的操作（Select 函数）时间复杂度为 O(n)，执行 n 次，因此构建哈夫曼树的时间复杂度为 O(n^2)。

**空间复杂度:**

- 需要 O(2n) 的空间来存储哈夫曼树节点。

**createHuffmanCode函数：**

**时间复杂度:**

- 生成每个字符的哈夫曼编码的时间复杂度取决于树的深度。哈夫曼树的深度为 O(log n)，因此时间复杂度为 O(n log n)。

**空间复杂度:**

- 哈夫曼编码数组的空间复杂度为 O(n)。

**supressFile函数:**

**时间复杂度:**

- 遍历整个文件的每个字符，并查找对应的哈夫曼编码，这两个操作的时间复杂度为 O(n log n)。

**空间复杂度:**

- 不需要额外的空间，空间复杂度为 O(1)。

**translate函数:**

**时间复杂度:**

- 遍历压缩文件的每个字符，每次需要查找编码是否在哈夫曼编码数组中，最坏情况下时间复杂度为 O(n log n)。

**空间复杂度:**

- 不需要额外的空间，空间复杂度为 O(1)。

**总结：**

最耗时的部分是构建哈夫曼树，整体的时间复杂度是O(n^2)。整体的空间复杂度是O(n)。

**作者：2023级计算机类10班 邢国富**