**数据结构课程设计报告**

**题目** 哈夫曼编/译码

**班级**  42210905

**姓名** 屈阳

**学号**  422109071303

**指导老师**  李宁

**完成日期** 2023 年 12 月 29 日

一、需求分析

1. 本程序中，通过给出的字符频率创建顺序表，根据创建的顺序表中的字符和权值来创建哈夫曼树，并将树保存在文件hfmTree中，利用已经建好的哈夫曼树（若无则用hfmTree中的），对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。利用已经建好的哈夫曼树将文件CodeFile中的代码进行译码，结果存入文件TextFile中

2. 将代码文件CodeFile以紧凑格式显示在终端上，每行50个代码。将已在内存中的哈夫曼树以直观的形式（凹入表）显示在终端上，同时将此形式的哈夫曼树储存在文件TreePrint中。

3. 程序执行的算法过程：

(1)使用顺序存储构造哈夫曼树并保存在文件hfmTree中 ；

(2)利用哈夫曼树对文件ToBeTran中的正文进行编码，结果存入文件CodeFile中；

(3) 利用哈夫曼树将文件CodeFile中的代码进行译码，结果存入文件TextFile中；

(4)打印代码文件CodeFile；

(5)以凹入表的形式打印哈夫曼树并保存在文件TreePrint中；

(6) 结束。

4. 测试数据

(1)用下表给出的字符集和频度的实际统计数据建立哈夫曼树 , 并实现以下报文的编码和译码："THIS PROGRAM IS MY FAVORITE"。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字符 |  | A | B | C | D | E | F | G | H | I | J | K | L | M |
| 频度 | 186 | 64 | 13 | 22 | 32 | 103 | 21 | 15 | 47 | 57 | 1 | 5 | 32 | 20 |
| 字符 | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |  |
| 频度 | 57 | 63 | 15 | 1 | 48 | 51 | 80 | 23 | 8 | 18 | 1 | 16 | 1 |  |

(2)根据（1）中哈夫曼树，对英文文件中的内容进行编码和译码，文件内容为IZWDVEWPTWDUJQJIUAXNQSGPJZFJVJSHKKHNXWXYCLXKDNJVXZKFBZODYJNPSPBUZENYFUV XGGEYVNZQMQGTOMMW MUYCOKHDFI

二、概要设计

为实现上述程序功能，应以顺序结构创建哈夫曼树。为此，需要的抽象数据类型： 哈夫曼树，结构体数组。

1. 哈夫曼树的抽象数据类型定义为：

ADT OrderedList

{

**数据对象：**D={ai|ai∈CharSet，i=1,2,...,n, n≥ 0}

**数据关系：**Rl={<ai-1,ai>|ai-1,ai∈D,ai-1<ai,i=2,...,n}

**基本操作：**

initHuffmanTree(&HT)：

初始条件：存在HT变量。

操作结果：构造一个空的哈夫曼树 HT。

buildHuffmanTree(HT, FreqList)：

初始条件：哈夫曼树 HT 已存在，FreqList 包含字符及其频率信息。

操作结果：根据频率信息构建哈夫曼树 HT。

encode(HuffmanTree, Message，num)：

` 初始条件：哈夫曼树 HT 已存在，Message 是待编码的字符串。

` 操作结果：将 Message 编码为哈夫曼编码。

decode(HuffmanTree, EncodedMessage, num)：

初始条件：哈夫曼树 HT 已存在，EncodedMessage 是哈夫曼编码的字符串。

操作结果：将哈夫曼编码解码为原始消息。

printFile():

` 初始条件：要打开的文件存在且存储的为二进制字符。

操作结果：将文件内容直接输出在终端。

displayHuffmanTree()：

初始条件：哈夫曼树 HT 已保存在文件中。

操作结果：以凹入表的形式显示哈夫曼树的结构。

}

2.结构体数组的抽象数据类型定义为：

**数据对象：**D={ai|ai∈CharSet，i=1,2,...,n, n≥ 0}

**数据关系：**Rl={<ai-1,ai>|ai-1,ai∈D,ai-1<ai,i=2,...,n}

**基本操作：**

extern FreqList F[27]：

操作结果：创建结构体数组；

3. 本程序包含三个模块

1) 主程序模块

void main(){

do

{

创建空哈夫曼树；

哈夫曼树初始化；

编码并保存；

译码并保存；

打印编码后的文件；

以凹入表显示哈夫曼树；

退出；

}while(!退出)

｝

2) 哈夫曼树模块 实现哈夫曼树的抽象数据类型；

3) 结构体数组模块 实现结构体的抽象数据类型；

关系调用图如下：

主程序模块

结构体数组模块

哈夫曼树模块

三、详细设计

1. 存储字符频率的结构体

typedef struct {

char data;

int weight;

}FreqList;

FreqList F[] {

/\*字符和频率构成的数组\*/

{字符, 频率},{字符, 频率}……

}

2.哈夫曼树的结构体

typedef struct {

char data;

int weight;

int parent, lchild, rchild;

}HTNode, \*HuffmanTree; /\*结点类型,指针类型\*/

void initHuffmanTree(HuffmanTree &HT) {

/\*建立空树\*/

HT = NULL;

}

void buildHuffmanTree(HuffmanTree &HT, FreqList \*F, int n) {

/\*判断想要传入的树是否满足要求，不符合则结束该操作，符合则创建哈夫曼树并将其保存在文件hfmTree中\*/

HT = (HuffmanTree)malloc((m + 1) \* sizeof(HTNode));

// 以字符及频率为基础创建哈夫曼树

for (int i = 1; i <= n; ++i) {

HT[i].data = F[i].data;

HT[i].weight = F[i].weight;

}

}

void encode(HuffmanTree HT, const string &input, int n) {

/\*遍历传入的字符串，根据哈夫曼树对其进行编码\*/

/\*编码后的结果拼接到encodedChar中\*/

/\*将encodedChar保存在文件CodeFile中\*/

}

void decode (HuffmanTree HT, const string &input, int n){

/\*遍历传入的字符串，根据哈夫曼树对其进行译码\*/

/\*编码后的结果拼接到decodedChar中\*/

/\*将decodedChar保存在文件TextFile中\*/

}

void printFilet() {

/\*打开要输出的文件\*/

while ((ch = fgetc(fp)) != EOF)

{

if (ch != ' ') {

printf("%c", ch);

count++;

}

if (count == 50) {

printf("\n");

count = 0;

}

}

}

void displayHuffmanTree(){

/\*通过“- ”调整缩进将哈夫曼树以凹入表的形式打印在终端\*/

for (int i = 0; i < depth; ++i) {

cout << "- "; // 这里可以调整缩进

}

}

3. 主函数和其他函数的伪码算法

void main()

{// 主函数

/\*创建HT变量，计算给定的结构体数组F中的元素个数n\*/

/\*循环创建菜单\*/

do{

initHuffmanTree(HT)；

buildHuffmanTree(HT, F, n)；

/\*使用c++中的函数读取文件内容，再进行编码译码操作\*/

encode(HT, input);

decode(HT, input);

printFilet();

displayHuffmanTree()；

退出操作；

}while(num != 7) ;

}//main

void Select(HuffmanTree HT, int n, int &s1, int &s2){

/\*选出两个最小值\*/

}

void huffmanTreeCode(HuffmanTree HT, int n){

/\*创建哈夫曼编码，并保存在文件里\*/

fp = fopen(“”);

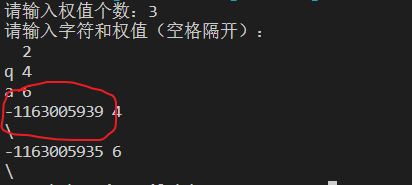
fprintf(fp,””);

fclose(fp);

}

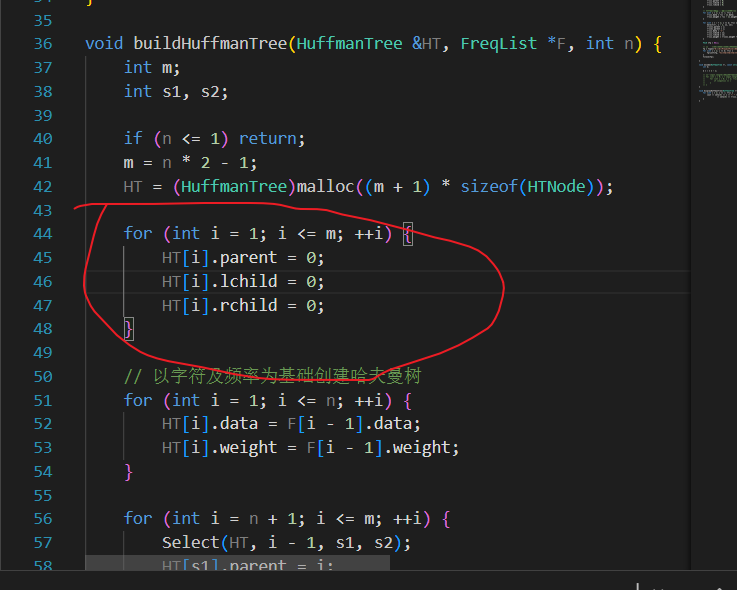
四、调试分析

1. 创建哈夫曼树时需要输入空格字符，不能使用scanf(“ %c”)这种形式在输入空格字符时会导致输入不匹配，应用getchar()，如图表1所示：



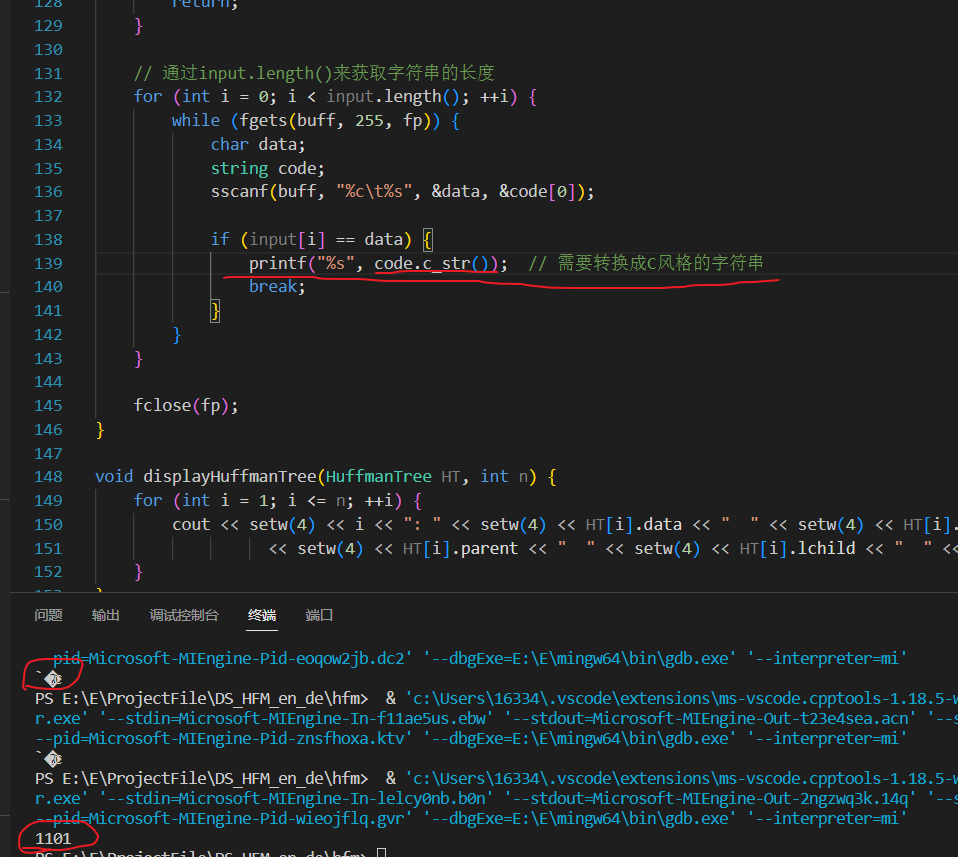
图表 1

2. 在创建哈夫曼树时需要先进行对父节点，子节点的初始化，否则数据会异常，如图表2所示：



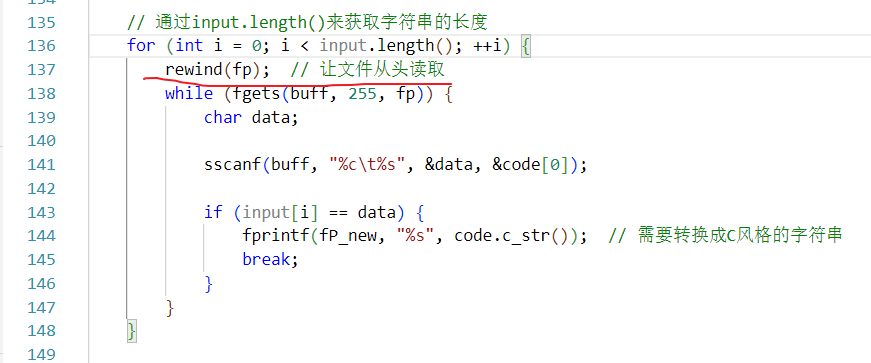
图表 2

3. 在输出字符串的时候需要使用C风格的字符串形式，否则会乱码，如图表3所示：



图表 3

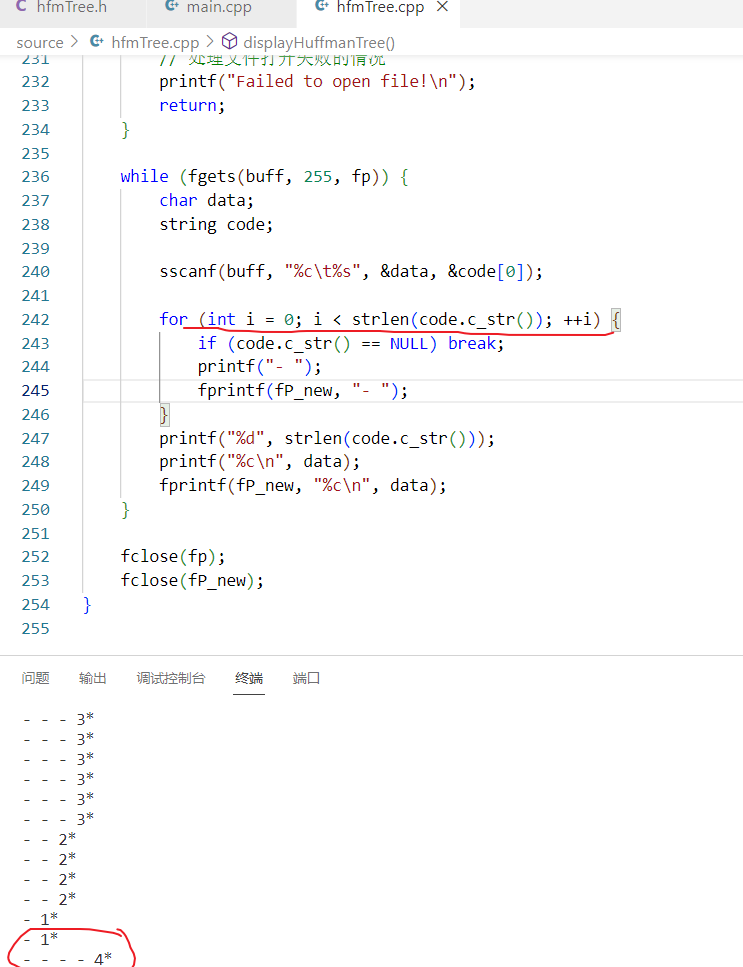
4. 为防止文件只读取一遍而不能完整编码，如图标4所示：



图表 4

5. 在保存编码成功的文件时应为“ %s”这样空格在前的格式，防止译码时多输出一次结尾字符。

6. 使用 strlen(code.c\_str()) 来获取字符串长度时可能会在读取时根据 \t 为 code 字符串添加结束符导致多判断字符，如图表5所示：



图表 5

7. 算法的时间复杂度分析

哈夫曼树中所用函数的时间复杂度分析如下：

Select函数用于在创建哈夫曼树时排序，依次调用需要一次循环，因此时间复杂度为O(n),n为哈夫曼树大小。

在创建哈夫曼树时buildHuffmanTree调用多个循环完成创建，时间复杂度也为O(n)，n为初始字符的数量。

在对文件内容进行编码encode时使用嵌套循环的方式进行文件内容与哈夫曼编码表一一对照，因此时间复杂度为O(n\*m)，n为读取到的文件内容中的字符长度，m为保存的哈夫曼编码的字符的数量。

译码decode与编码encode同理，时间复杂度一样为O(n\*m)，n为读取到的文件内容中的字符长度，m为保存的哈夫曼编码的字符的数量。

在printFilet进行输出文件内容时使用一层循环遍历文件内容，时间复杂度为O(n),n为文件内容大小。

在displayHuffmanTree进行以凹入表的形式打印并保存哈夫曼树时，使用嵌套循环控制缩进长短，因此时间复杂度为O(n2)，n为哈夫曼树中结点的总数。

主函数main使用while循环来生产菜单，因此时间复杂度为O(n),n为程序运行次数。

8. 本实习作业采用数据抽象的程序设计方法，将程序划分为四个层次结构；元素结点、哈夫曼树、结构体数组和主控模块。使得设计时思路清晰，实现时调试顺利。各模块具有较好的可重用性。了解到了c语言和c++中对于文件的读取和存入操作，同时学习到了更多的库函数的使用，学会合理的区分头文件和源文件并可以熟练的将二者联系在一起，对于模块化操作有了更深的理解，得到了一次良好的程序设计训练。

五、测试结果

执行命令 1后创建一个空的树

执行命令 2 根据给出的字符和频率构建哈夫曼树并将编码保存在文件中

执行命令 3 对指定文件中的内容进行编码，并保存在另一个文件中

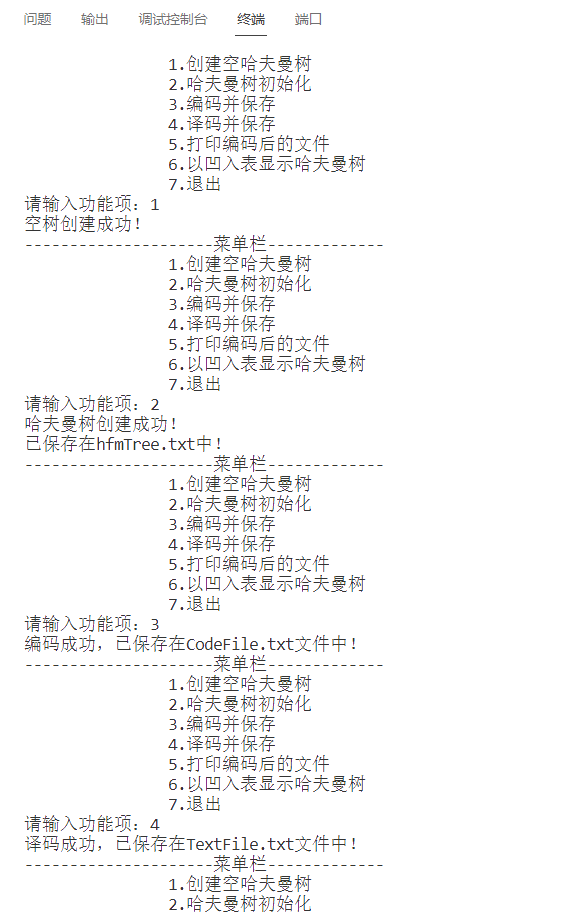
执行命令 4 对指定文件中的内容进行译码，并保存在另一个文件中

执行命令 5 将哈夫曼编码以紧凑格式打印纸终端上

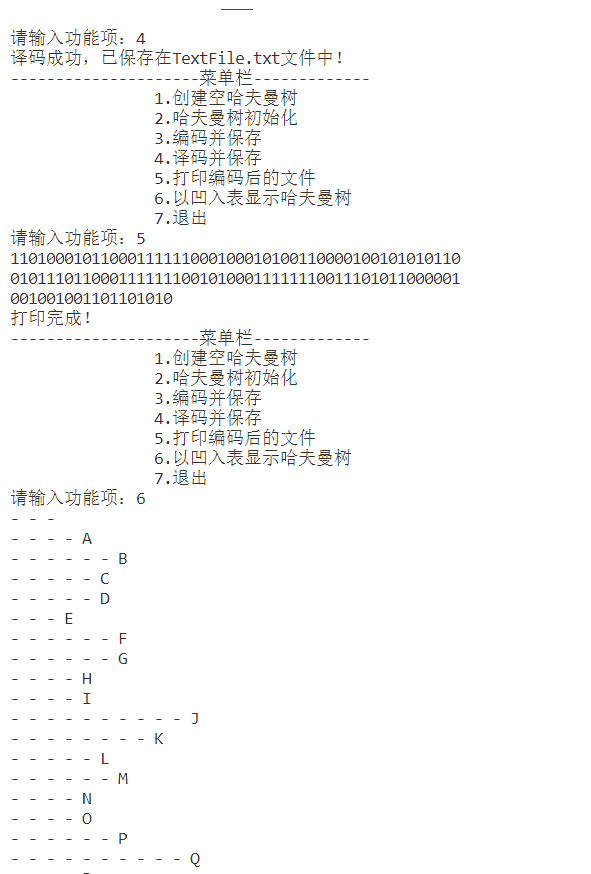
执行命令 6 将哈夫曼树以凹入表的形式打印在终端上并保存在文件里

执行命令 7 退出程序

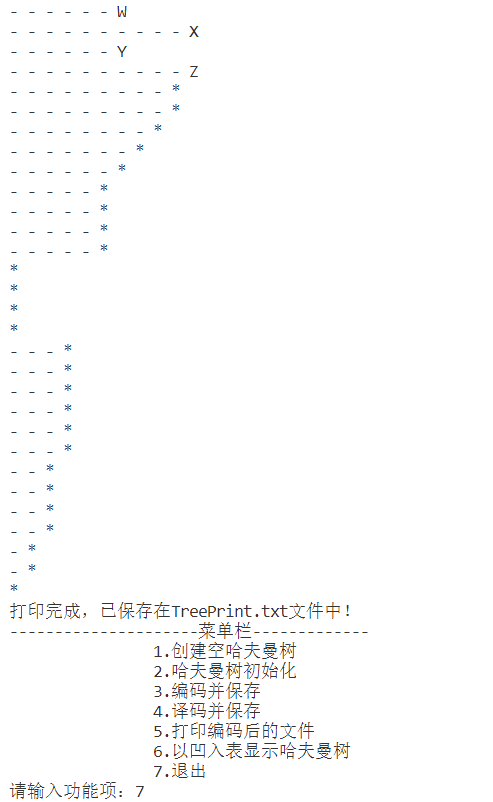
操作流程如下列图表所示：



图表 6



图表 7



图表 8

六、附录 源程序文件名清单

freqList.h // 字符和权值的结构体数组

hfmTree.h // 哈夫曼树

hfmTree.cpp // 哈夫曼树中函数的实现

main.cpp // 主函数

hfmTree.txt // 哈夫曼树构建完成后的内容

ToBeTran.txt // 存有待编码内容的文件

CodeFile.txt // 编码后的文件

TextFile.txt // 译码后的文件

TreePrint.txt // 以凹入表展示的哈夫曼树