**数据结构与算法 第二次实验**

学号：22920212204392 姓名：黄勖

1. **实验目的**
2. 了解二叉树的基础实现方法与原理，理解二叉树的基本操作的代码编写方式
3. 学会灵活按照二叉树的存储内容自由编写二叉树的存储结构
4. 在二叉树的基础上进而理解编写哈夫曼树的基础实现方法，理解哈夫曼树的基本操作的代码编写方式
5. 通过实验探索不同类型的树的相似点与区别，发现在操作实现上的不同
6. **实验内容**

2-1 设二叉树的存储结构如下：

typedef struct BiTNode

{

Type data; //数据域；Type: 用户定义数据类型

struct BiTNode \*Lchild; //左指针域

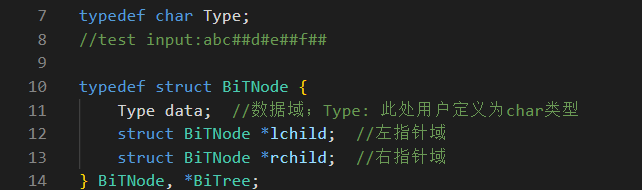
struct BiTNode \*Rchild; //右指针域

} BiTNode, \*BiTree;

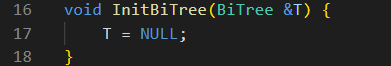
实现二叉树的基本操作和遍历操作。

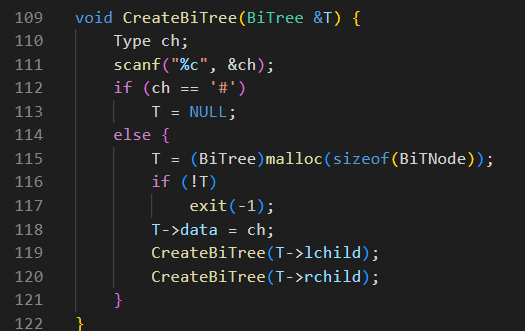
**实验内容：**

* **设计二叉树存储方式**

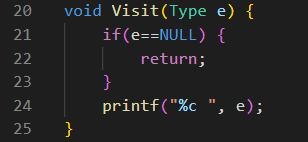


* **创建二叉树**

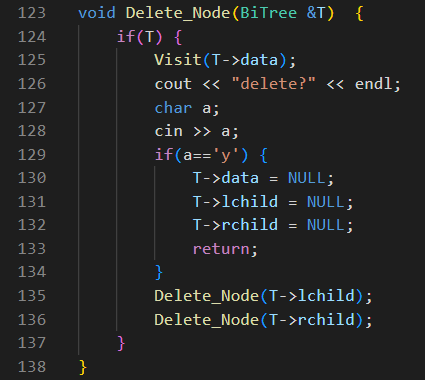




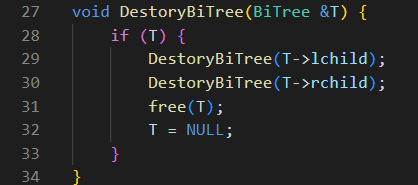
* **访问节点**



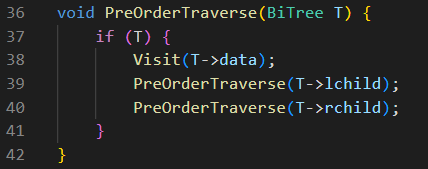
* **删除节点**



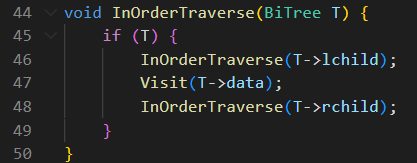
* **释放二叉树**



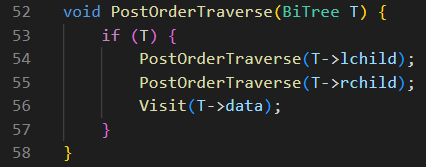
* **前序遍历**



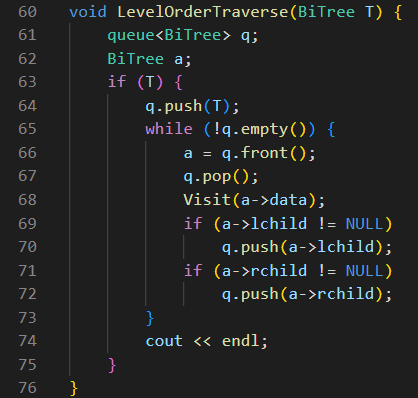
* **中序遍历**



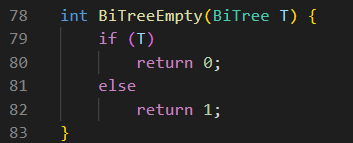
* **后序遍历**



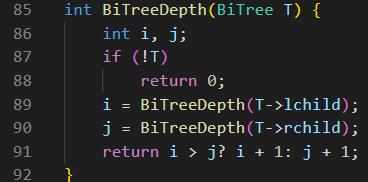
* **层序遍历**



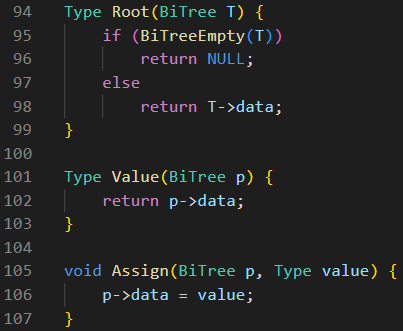
* **判断二叉树是否为空**



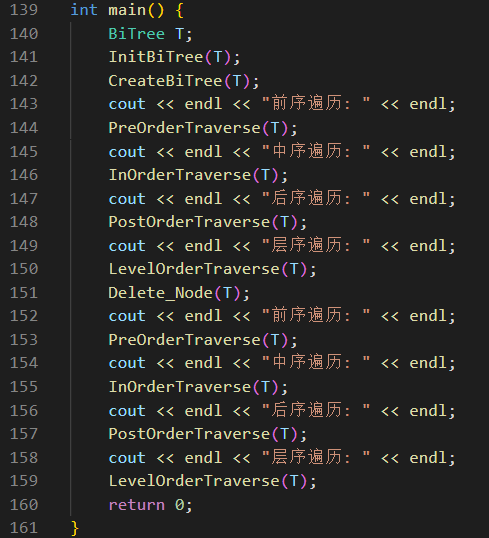
* **计算二叉树深度**



* **其他的一些小功能（读根、值、设定值）**



* **主函数交互**



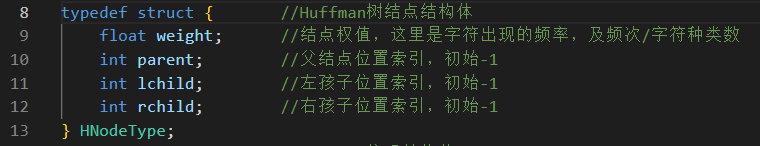
2-2 设计算法实现：

(1) 构造哈夫曼树；

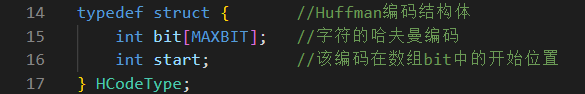
(2) 求解哈夫曼编码。

**实验内容：**

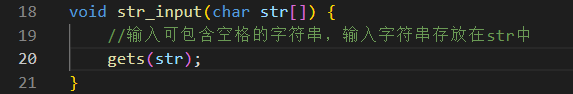
* 哈夫曼结点，放在一个数组中，即 HNodeType HuffNodes[]



* 哈夫曼编码结构，也采用顺序存储结构（数组）



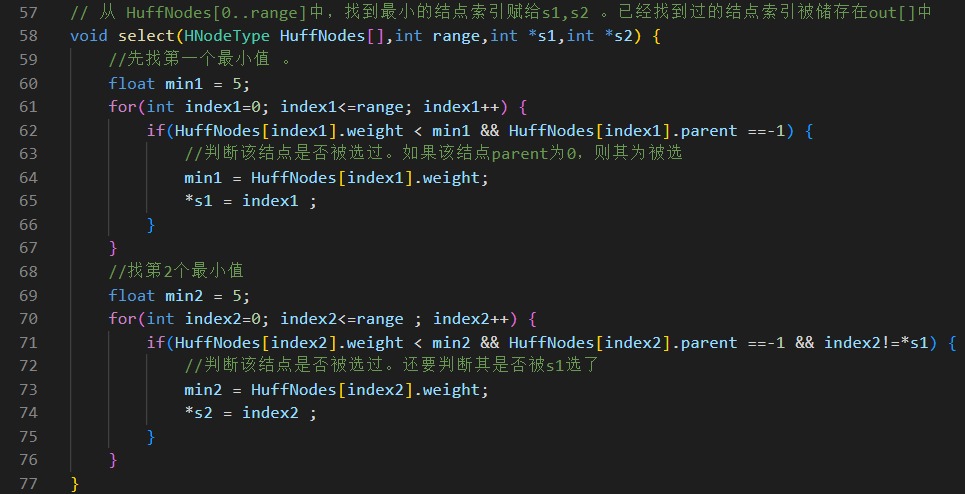
* 接受字符串



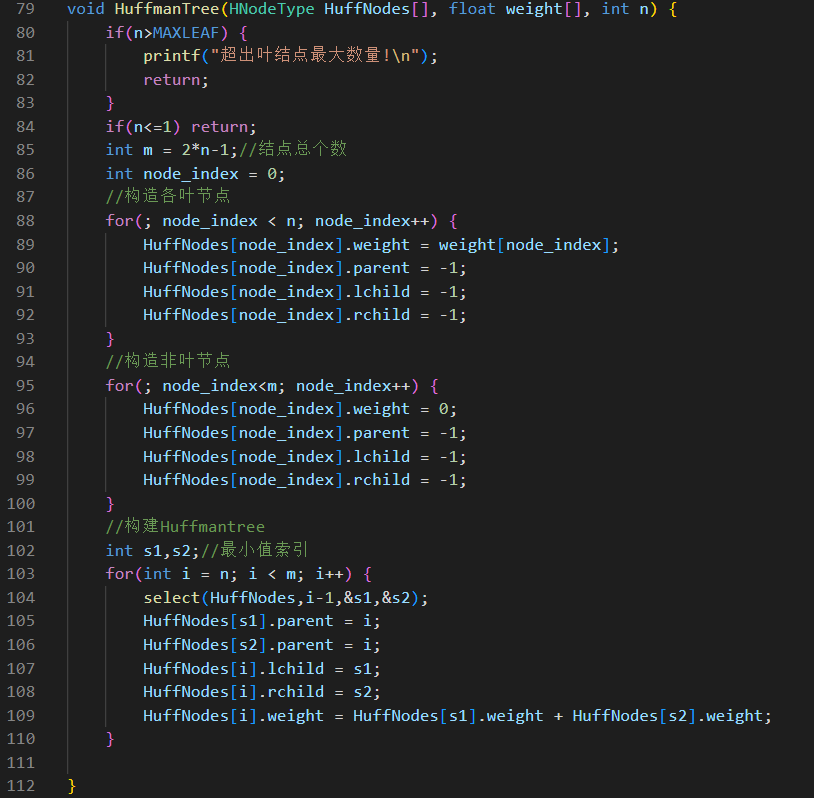
* 统计字符频次



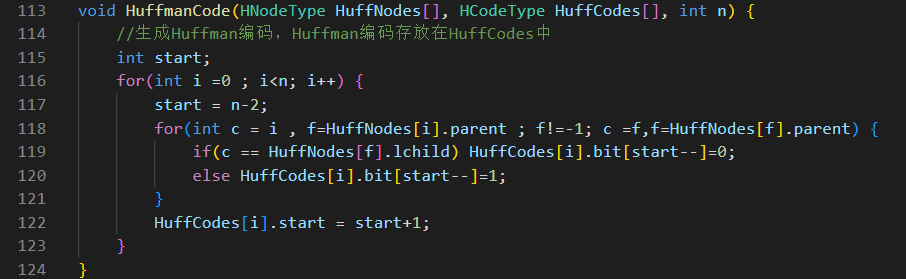
* 找到权值最小的两个结点



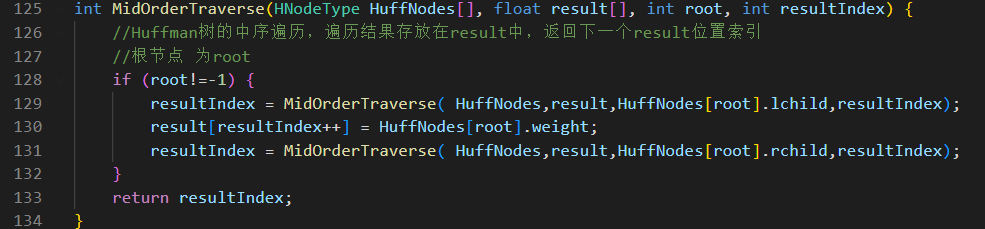
* 构造哈夫曼树



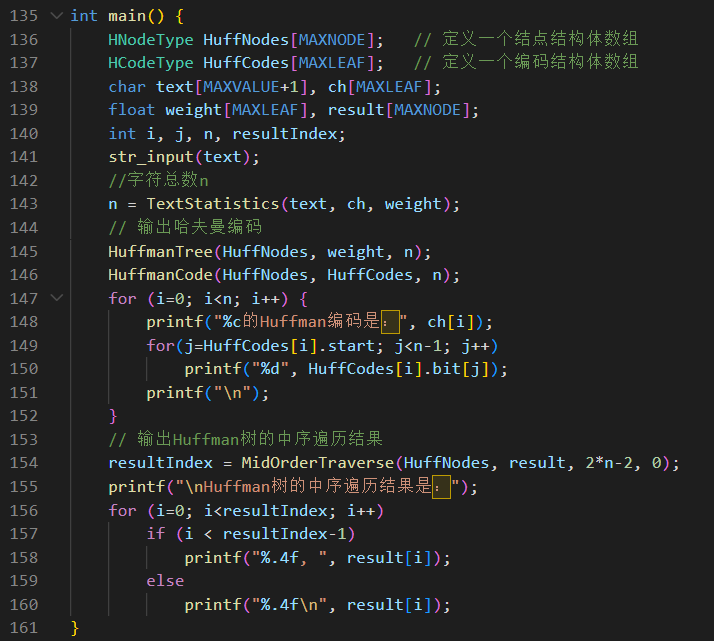
* 生成哈夫曼编码



* 遍历哈夫曼树，使用队列递归方式，在本次实验中使用的中序遍历

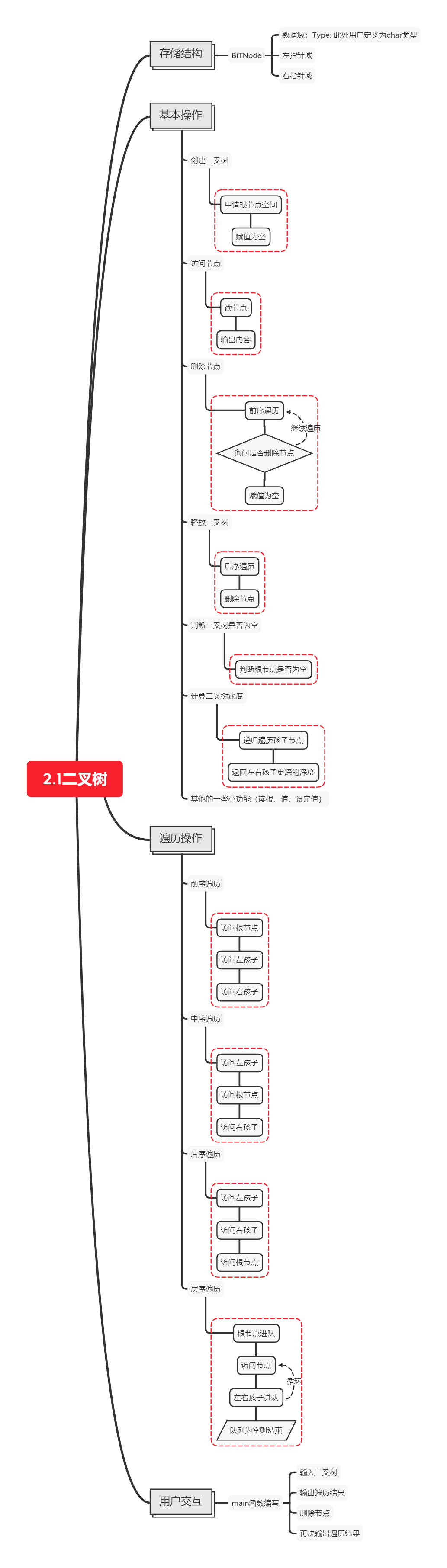


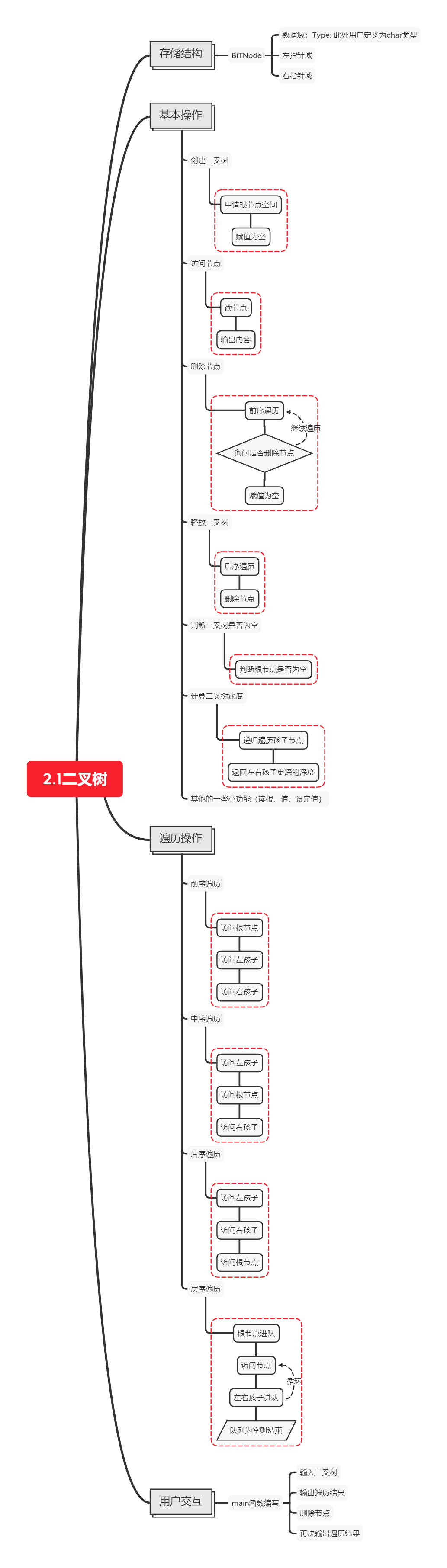
* 主函数交互



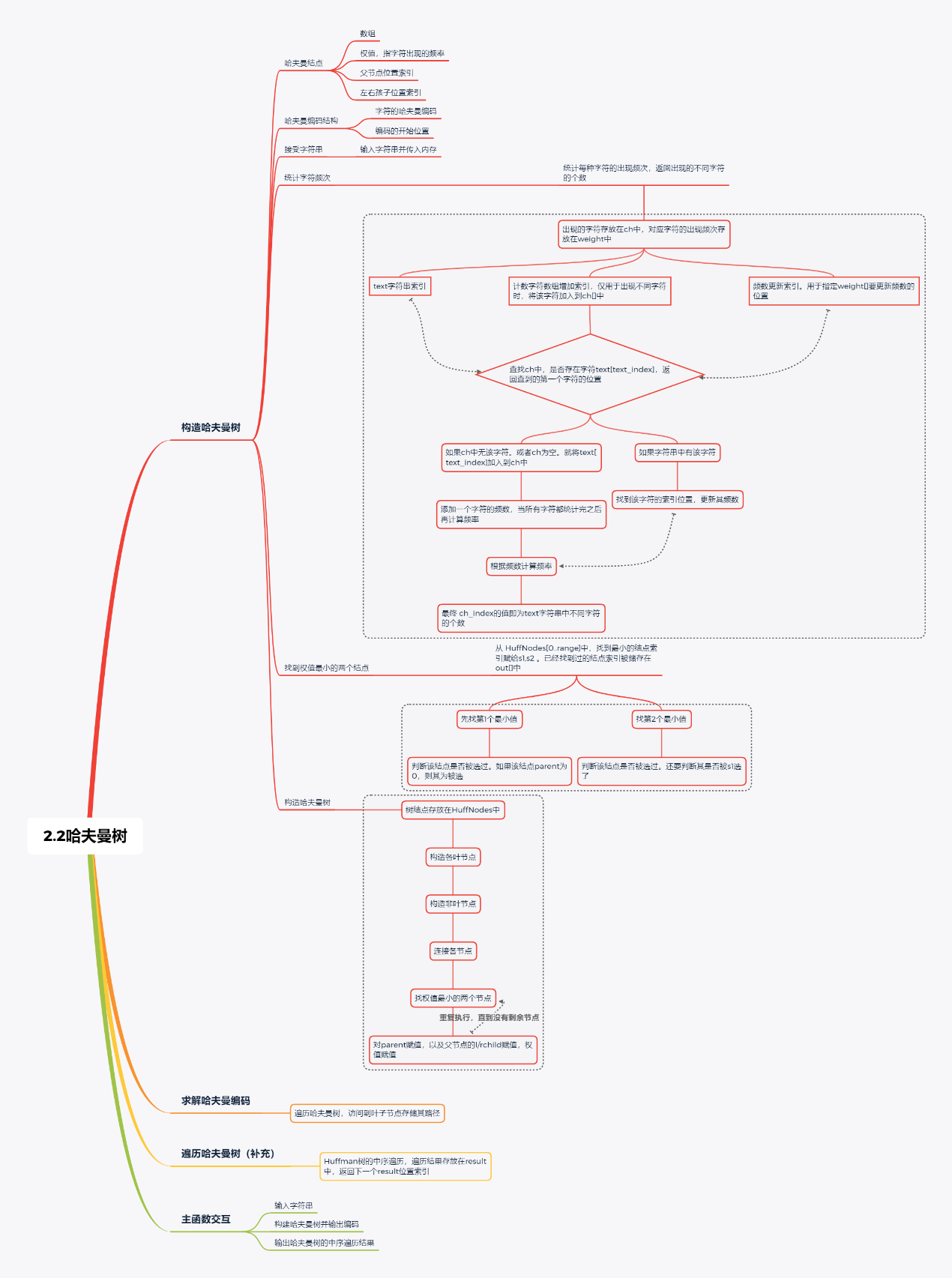
1. **主要算法流程图**

**2-1算法流程（图片太小可见目录下的2-1二叉树.png）**

****

****

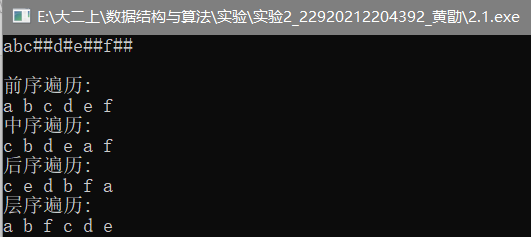
**2-2算法流程**

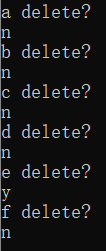
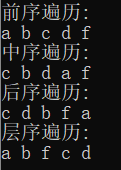
****

1. **实验结果：**

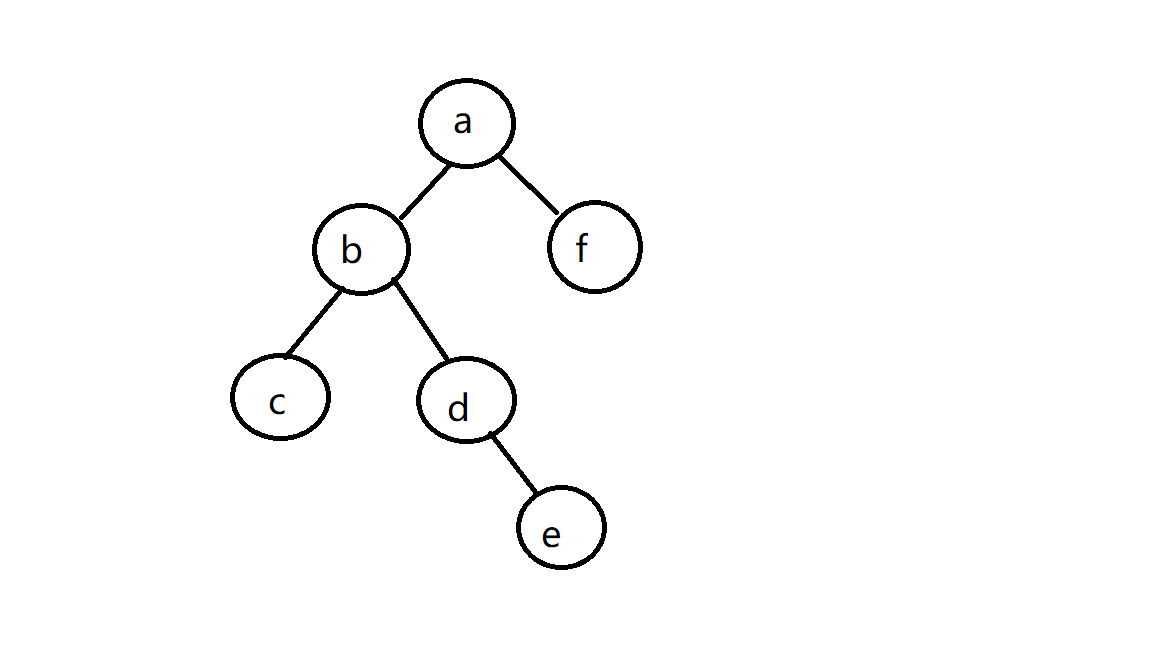
（结合截图说明算法的输入输出）

1. 关于2-1的输入与输出：



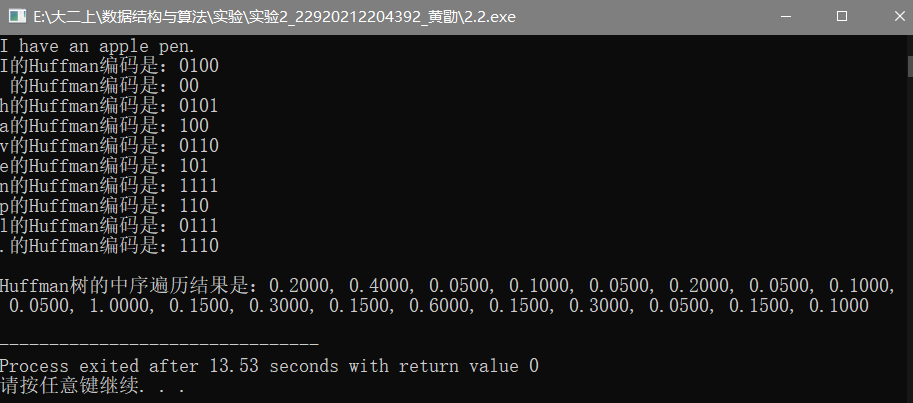
 

在实际运行中，我创建了

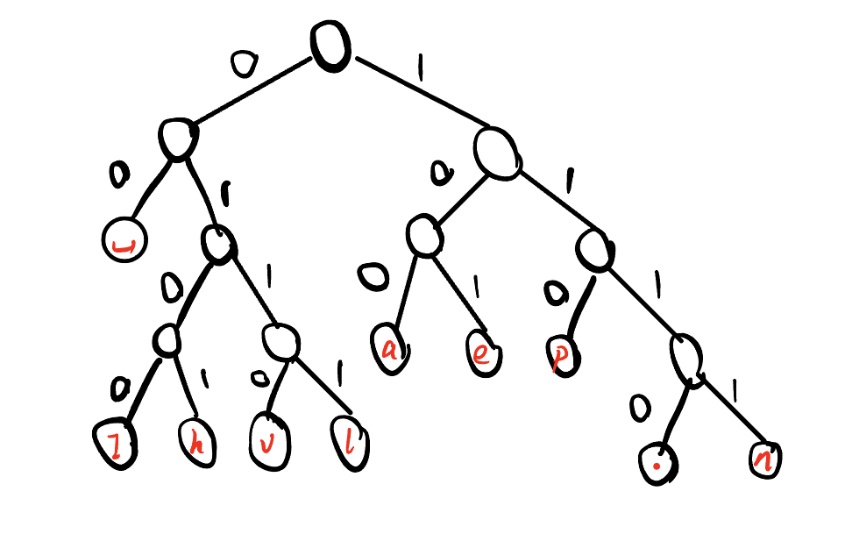


这样一个二叉树，并在后续删除了e，可以看出遍历的结果都是正确的。

1. 关于2-2的输入与输出



在实际运行中，我创建了



这样的哈夫曼树，并且利用中序遍历得到了每个节点的值（即出现概率）。

1. **实验小结（即总结本次实验所得到的经验与启发等）：**

在本次实验中，我尝试具体运用了二叉树，在实体机的实验中我能够更深刻地理解对这一部分数据结构的执行方式与特点，并且在编写代码的过程中，我通过不断的调试去寻找语句之间的问题和不足，在潜移默化中提高了我的代码编写能力，这是一次完成效果良好的实验！