**西南石油大学实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目编号** | 001063000503 | **项目名称** | 二叉树的构造和遍历 | **成绩** |  |
| **专业年级** | **软件工程2022级** | | | **指导教师** | **胡卫东** |
| **姓名** | **李浩楠** | **学号** | **202231060920** | **实验日期** | **2023/6/3** |

1. **实验目的**
2. 掌握二叉树的定义。
3. 掌握二叉树的相关操作。
4. 使用C/C++是相关操作。
5. **实验工具**

PC微机、Windows、PTA或Dev C++。

1. **实验步骤**
2. 后缀式。

本题要求实现一个函数，输出二叉树表示的表达式的后缀式。输入一个表达式的前缀形式（该表达式二叉树的扩展的先序序列），构造该二叉树，并输出其后序序列即为该表达式的后缀形式。

1. 求二叉树的高度。

输入二叉树的先序序列以创建二叉树，并求出该二叉树的高度。

1. 哈夫曼编码。

编写函数实现哈夫曼编码。输入结点个数(保证个数>1)及各结点的权值，为各结点进行编码。其中 HT 为哈夫曼树，n 为叶子结点个数， HC 为哈夫曼编码。

1. **实验结果**
2. 测试用例：+-A##\*F##G##C##

代码：void creat(BiTree &Tree)//构建二叉树

{

char a;

scanf("%c",&a);

if(a=='#')

Tree=NULL;

else

{

Tree = new BiTNode;

Tree->data=a;

creat(Tree->lchild);

creat(Tree->rchild);

}

}

void PostOrder(BiTree Tree)//输出后缀表达式

{

if(Tree)

{

PostOrder(Tree->lchild);

PostOrder(Tree->rchild);

printf("%c",Tree->data);

}

}

运行结果：后缀表达式为：AFG\*-C+

1. 测试用例：AB##CD##EF###

代码：int Depth(BiTree Tree)

{

int m,n;

if(!Tree)

return 0;

m=Depth(Tree->lchild);

n=Depth(Tree->rchild);

if(m>n)

return m+1;

else

return n+1;

}

运行结果：4

1. 测试用例：3

1 2 3

代码：void CreateHuffman\_tree(HuffmanTree &HT,int n)/\*建立n个叶子结点的哈夫曼树\*/

{

if(n<=1) return;

int m,i,s1,s2;

m=2\*n-1;

HT=new HTNode[m+1];

for(i=0;i<=m;i++)

{

HT[i].parent=0;

HT[i].lchild=0;

HT[i].rchild=0;

}

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

{

scanf("%d",&HT[i].weight);

}

for(i=n+1;i<=m;i++)

{

Select(HT,i-1,s1,s2);

HT[s1].parent=i;

HT[s2].parent=i;

HT[i].lchild=s1;

HT[i].rchild=s2;

HT[i].weight=HT[s1].weight+HT[s2].weight;

}

}

void Huffman\_code(HuffmanTree HT,HuffmanCode &HC,int n)//求哈夫曼编码

{

int i,f,start,c;

char \*cd;

HC=new char\*[n+1];

cd=new char[n];

cd[n-1]!='\0';

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

{

start=n-1;

c=i;f=HT[i].parent;

while(f!=0)

{

start--;

if(HT[f].lchild==c)

cd[start]='0';

else

cd[start]='1';

c=f;f=HT[f].parent;

}

HC[i]=new char[n-start];

strcpy(HC[i],&cd[start]);

}

delete cd;

}

运行结果：编码是：10

编码是：11

编码是：0

1. **实验总结**  
   1.二叉树是一种常用的树结构，二叉树具有一些特殊的性质，而满二叉树和完全二叉树又是两种特殊形态的二叉树。
2. 二叉树有两种存储表示：顺序存储和链式存储。顺序存储就是把二叉树的所有节点按照层次顺序存储到连续的存储单元中，这种存储更适用于完全二叉树。链式存储又称二叉链表，每个节点包括两个指针，分别指向其左孩子和右孩子。链式存储是二叉树常用的存储结构。
3. 树的存储结构有3种：双亲表示法、孩子表示法和孩子兄弟表示法。孩子兄弟表示法是常用的表示法，任意一棵树都能通过孩子兄弟表示法转换为二叉树进行存储。森林与二叉树之间也存在相应的转换方法，通过这些转换，可以利用二叉树的操作解决一般树的有关问题。
4. 二叉树的遍历算法是其他运算的基础，通过遍历可得到二叉树中节点访问的线性序列，实现了非线性结构的线性化。根据访问节点的次序不同有3种遍历，即先序遍历、中序遍历、后序遍历，其时间复杂度均为O（n）。
5. 在线索二叉树中，利用二叉链表中的n+1个空指针域来存放指向某种遍历次序下的前驱节点和后继节点的指针，这些附加的指针称为"线索"。引入二叉线索树的目的是加快查找  
   节点前驱或后继的速度。
6. 哈夫曼树在通信编码技术上有广泛的应用，只要构造了哈夫曼树，按分支情况在左路径上写代码0，在右路径上写代码1，然后从上到下叶节点相应路径上的代码序列就是该叶节点的最优前缀码，即哈夫曼编码。