计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：朱龙康 学号：1824120002 专业：计算机科学与技术\_ 年级：\_2018级

课程：数据结构 主讲教师：\_袁彩虹\_ 辅导教师：\_\_袁彩虹\_\_

实验时间：\_\_2020\_年 \_3\_月 \_\_27\_日 \_上\_午\_8\_时至\_10\_时，实验地点\_\_\_\_\_\_\_

实验题目： 二叉树的操作

实验目的： （1）掌握二叉树的二叉链表存储方式及二叉树的特征；

（2）验证二叉树在二叉链表存储结构下遍历操作的实现；

（3）掌握赫夫曼树的构造方法和赫夫曼编码的方法。

实验环境（硬件和软件） CodeBlocks

实验内容：

*1）创建二叉链表结构的二叉树；*

*2）二叉树的先序、中序、后序和层序遍历；*

*3）计算二叉树的高度和叶子结点个数；*

*4）赫夫曼树的构造方法和赫夫曼编码的实现方法。*

实验步骤：

**1）创建二叉链表结构的二叉树；**

typedef struct BiTNode{

char data;//数据域

struct BiTNode \*lchild,\*rchild;//左右孩子

}BiTNode,\*BiTree;

//先顺序遍历创建二叉链表

void CreateBiTree(BiTree &T){

cout<<"输入"<<endl;

char integer; //待输入数据

cin>>integer;

if(integer=='#') {T=NULL; return;} //若输入# 表明不再创建节点

else{

T=new BiTNode; //节点初始化

T->data=integer;

cout<<"建好一个"<<endl;

CreateBiTree(T->lchild); //看是不是还要左孩子

CreateBiTree(T->rchild);//看是不是还要右孩子

}

}

**2）二叉树的先序、中序、后序和层序遍历；**

//中序遍历二叉链表(递归形式）

void InOrderTraverse(BiTree T){

if(T==NULL) return;

else{ //T指针不为空

InOrderTraverse(T->lchild);

cout<<T->data;

InOrderTraverse(T->rchild);

}

}

//前序遍历二叉链表(递归形式）

void FrontTraverse(BiTree T){

if(T==NULL) return;

else{ //T指针不为空

cout<<T->data;

InOrderTraverse(T->lchild);

InOrderTraverse(T->rchild);

}

}

//后序遍历二叉链表(递归形式）

void RearTraverse(BiTree T){

if(T==NULL) return;

else{ //T指针不为

InOrderTraverse(T->lchild);

InOrderTraverse(T->rchild);

cout<<T->data;

}

}

//层序遍历二叉树

//先定义队列存储结构

typedef struct

{

BiTree \*base;//存储空间的基地址

int front;//头指针

int rear;//尾指针

}SqQueue;

bool InitQueue(SqQueue &q){

q.base=new BiTree[MAXSIZE];//申请空间

if(!q.base)return false;//申请失败

q.front=q.rear=0;//队列空 头尾指针置零

return true;

}

bool InQueue(SqQueue &q,BiTree e){

if((q.rear+1)%MAXSIZE==q.front)return false;//队满

q.base[q.rear]=e;//将e入队

q.rear=(q.rear+1)%MAXSIZE;//尾指针移动

return true;

}

bool OutQueue(SqQueue &q,BiTree &m){

if(q.rear==q.front)return false;//队空

m=q.base[q.front];//将m传给参数带出

q.front=(q.front+1)%MAXSIZE;//头指针改变

return true;

}

void levelTree(BiTree b){

SqQueue s;

InitQueue(s);

BiTree tem;

InQueue(s,b);

while(s.front!=s.rear){

OutQueue(s,tem);

cout<<tem->data;

if(tem->lchild) InQueue(s,tem->lchild);

if(tem->rchild)InQueue(s,tem->rchild);

}

}

**3）计算二叉树的高度和叶子结点个数；**

//计算二叉树深度

int Depth(BiTree T){

int m,n;

if(T==NULL) return 0;

else{

m=Depth(T->lchild);

n=Depth(T->rchild);

if(m>n)return (m+1);

else return (n+1);

}

}

//统计二叉树中节点个数

int CountNode(BiTree T){

if(T==NULL)return 0;

else return CountNode(T->lchild)+CountNode(T->rchild)+1;

}

**4）赫夫曼树的构造方法和赫夫曼编码的实现方法。**

//哈夫曼树

//哈夫曼树的存储结构

typedef struct{

int weight;

int parent,lchild,rchild;

}HTNode,\*HTree;

//构造哈夫曼树

void Select(HTree H,int n,int &s1,int &s2){//找到权重最小的两个结点 将下表给s1 s2

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

if(H[i].parent==0){

s1=i;

}

}

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

if(H[i].parent==0){

if(H[i].weight<H[s1].weight){s1=i;}

}

}

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

if(H[i].parent==0&&i!=s1){

s2=i;

}

}

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

if(H[i].parent==0&&i!=s1){

if(H[i].weight<H[s2].weight){s2=i;}

}

}

}

void CreateHTree(HTree &H,int n){

if(n<=1) return;

int m=2\*n-1;

H=new HTNode[m+1];//不使用0号单元

for(int i=1;i<=m;i++){//初始化每一个节点

H[i].parent=0;H[i].lchild=0;H[i].rchild=0;

}

for(int j=1;j<=n;j++){//输入权重

cin>>H[j].weight;

}

int s1,s2;

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

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

H[s1].parent=i;H[s2].parent=i;

H[i].lchild=s1;H[i].rchild=s2;

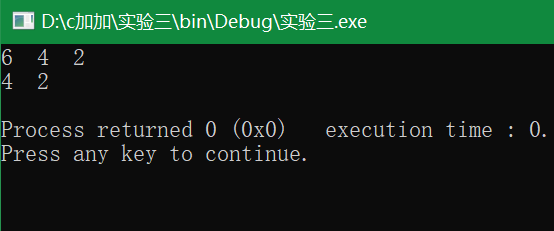
H[i].weight=(H[s1].weight+H[s2].weight);

}

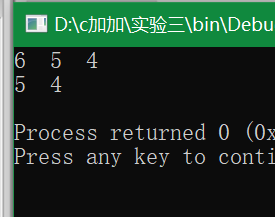
}

实验数据记录：

顺序栈 入栈6 4 2 出栈6

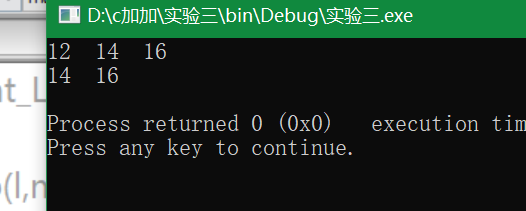


链栈 入栈 6 5 4 出栈 6



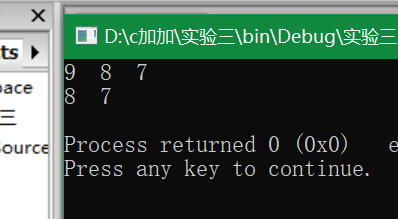
循环队列 入队 12 14 16

出队 12



链队列 入队9 8 7

出队 9



问题讨论：