

**数据结构实验报告**

**实验报告题目：** 二叉树的创建与遍历

**学生姓名**： 刘凌杰

**专 业： 软件工程**

**学 号**： 201720180411

**指导教师： 邹 国 华**

**2018年 11 月 12 日**

**一. 实验内容：**

* **二叉树的创建**
* **前序遍历**
* **中序遍历**
* **后序遍历**
* **查找节点深度**
* **查找元素的地址**
* **获取节点个数**
* **获取叶节点个数**
* **计算最大深度**

**二. 实验算法实现：**

**1.结构体。**

typedef struct Tree

{

DataType data;

struct Tree \*lchild;

struct Tree \*rchild;

}Tree;

**2.创建二叉树。**

//创建二叉树

Tree\* createTree(DataType Data[])

{

Tree \*T,\*Ntree[MAXSIZE];

Ntree[0]=(Tree\*)malloc(sizeof(Tree));

Ntree[0]->data=Data[0];

Ntree[0]->lchild=NULL;

Ntree[0]->rchild=NULL;

T=Ntree[0];

for(int i=1;i<MAXSIZE;i++)

{

if(Data[i]=='\0')

break;

if(Data[i]!='0')

{

Ntree[i]=(Tree\*)malloc(sizeof(Tree));

Ntree[i]->data=Data[i];

Ntree[i]->lchild=NULL;

Ntree[i]->rchild=NULL;

int j=(i-1)/2;

if(2\*j+1==i)

Ntree[j]->lchild=Ntree[i];

else

Ntree[j]->rchild=Ntree[i];

}

}

return T;

}

**3.前，中，后序遍历。**

//前序遍历

void preorderTree1(Tree \*T)

{

if(T)

{

cout<<T->data<<" ";

preorderTree1(T->lchild);

preorderTree1(T->rchild);

}

}

//中序遍历

void preorderTree2(Tree \*T)

{

if(T)

{

preorderTree2(T->lchild);

cout<<T->data<<" ";

preorderTree2(T->rchild);

}

}

//后序遍历

void preorderTree3(Tree \*T)

{

if(T)

{

preorderTree3(T->lchild);

preorderTree3(T->rchild);

cout<<T->data<<" ";

}

}

**4. //求叶节点**

int yeNum(Tree\* T)

{

if(!T)

return 0;

if(T->lchild==NULL&&T->rchild==NULL)

return 1;

return yeNum(T->lchild)+yeNum(T->rchild);

}

**5. //求节点数**

int jieNum(Tree\* T)

{

if(!T)

return 0;

if(T)

return jieNum(T->lchild)+jieNum(T->rchild)+1;

}

**6. //求深度**

int depth(Tree\* T)

{

// static int dep=0;

if(!T)

return 0;

int right=depth(T->lchild );

int left=depth(T->rchild );

if(left>right)

return left+1;

else

return right+1;

}

**7. //获取节点所在深度**

//返回节点所在层次

int reaDepth(Tree \*T,DataType e,int len)

{

int depth;

if(!T)

return 0;

else

if(T->data==e)

return len;

else

{

if((depth=reaDepth(T->lchild,e,len+1))!=0)

return depth;

if((depth=reaDepth(T->rchild,e,len+1))!=0)

return depth;

return 0;

}

}**8. //返回节点指针**

Tree\* node(Tree \*T,DataType e)

{

Tree\* pr;

if(T->data==NULL)

return NULL;

else

{

if(T->data==e)

return T;

else

{

if(pr=reasearch(T->lchild,e))

return pr;

else if(pr=reasearch(T->rchild,e))

return pr;

return NULL;

}

}

}

**8.main函数。**

void main()

{

Tree \*T;

DataType data[MAXSIZE];

for(int i=0;i<MAXSIZE;i++)

data[i]='0';

cout<<"请输入数据创建二叉树："<<endl;

cin>>data;

T=createTree(data);

cout<<"前序遍历：";

preorderTree1(T);cout<<endl;

cout<<"中序遍历：";

preorderTree2(T);cout<<endl;

cout<<"后序遍历：";

preorderTree3(T);cout<<endl;

cout<<"节点数为"<<jieNum(T)<<endl;

cout<<"叶节点数为"<<yeNum(T)<<endl;

cout<<"深度为："<<depth(T)<<endl;

char ch;

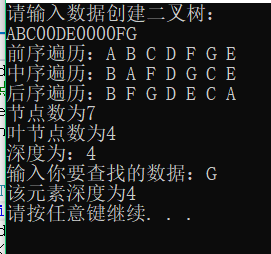
cout<<"输入你要查找的数据：";

cin>>ch;

cout<<"该元素深度为"<<reaDepth(T,ch,1)<<endl;

system("pause");

}



**三．实验总结：**

经过一段时间的学习，二叉树算法的实现基本掌握了，就是代码打得还不太熟练，还有待提高，有些功能不太熟要问同学或查资料，尤其在做遍历运算时。以后在这方面我会多花点时间，让自己更加熟练。