**江西师范大学计算机信息工程学院学生实验报告（7）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **专业：** | **数据科学与大数据技术2班** | **姓名：** | **赖丽婷** | **学号：** |  | **日期：** | **2021.11.8** |

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 数据结构 | 实验室名称 | 计算机综合实验室 |
| 实验名称 | 二叉树 | | |
| 指导教师 | 李云清 | 成绩 |  |

1. **实验目的**（结出本次实验所涉及并要求掌握的知识点）

1.1掌握二叉树的动态存储结构及表示方法

1.2掌握二叉树的前序，中序和后序遍历的递归与非递归算法，掌握二叉树的层次遍历算法

1.3运用二叉树的三种遍历算法求解基于二叉树的相关问题

**2.实验内容**（结出实验内容具体描述）

2.1实现二叉树t的非递归前序遍历

2.2实现二叉树的层次遍历

2.3编写函数分别返回二叉树t的前序遍历下的最后一个节点地址和后序遍历下的第一个节点的地址

2.4编写一个函数求值为x的节点在二叉树中的层数

2.5编写函数，将一颗给定二叉树中的所有结点的左子女和右子女互换

**3.算法描述及实验步骤**（用适当的形式表达算法设计思想与算法实现步骤）

3.1

利用栈stack,当栈不为空或者树还没访问完的时候执行程序，如果左子树不为空，把左 子树的值挨个打印出来，并将左子树进栈，如果为空，将上一个子树出栈，并访问他的 右子树，以此循环，直到栈为空并且树访问完

3.2

利用队列quene,从根部开始将树进队列，然后将这时的左右子树进队列，打印出队首的 值，队首出队列，队首后移一位，通过这样的程序循环直到队首等于队尾，就能实现二 叉树的层次遍历了

3.3

前序遍历的最后一个节点值可以从右子树开始查找，如果没有右子树叶子节点，则去找 左子树，直到找到第一个叶子节点，返回此叶子节点的值

后序遍历的第一个节点值可以假设在左子树的叶子节点上，如果没有左子树的叶子节 点，则去访问右子树，然后再按照从左往右的顺序找到找到第一个叶子节点，并将地址 返回

3.4

利用递归算法，按照前序遍历类似的算法，先访问左子树，当找不到值为x的时候返回-1，找到了返回1，因为要求层次，则找到了，不断往上加一，没找到，则开始往右子树找，依然按照从左往右的顺序查找值为x的节点，如果在右子树找到了不断往上加一，由m记入上一层遍历左子树的层数，由n记录上一次遍历右子树的层数，最终将左子树的层数和右子树的层数加起来就是x在的层数。

3.5

可以利用头递归或者尾递归的算法将左右子树互换，按照从左往右的顺序直到遍历完所有树节点。

1. **调试过程及运行结果**（详细记录在调试过程中出现的问题及解决方法。记录实验执行的结果）

**5. 总结**（对实验结果进行分析，问题回答，实验心得体会及改进意见）

第三题虽然利用遍历的方法能做出来，但是没有特别有必要，如果要找前序遍历的最后一个节点值，可以假设在右子树的最底层的那个叶子节点，如果没有右子树再往左子树找，然后再看左子树是否有右子树的叶子节点，后序的第一个节点思路也差不多

**6.附录**（程序源代码等）

6.1

#include "bintree.h"

char \*a = "ABC##D#E##F##"; /\*?????????????t?????????\*/

/\*????preorder1()??????????????????????t??????????????????????????\*/

void preorder1(bintree t) {

seqstack s;

Init(&s);

while(s.top != -1 || t != NULL)

{

if(t != NULL)

{

printf("%5c", t->data);

push(&s, t);

t = t->lchild;

}

else

{

t = pop(&s);

t = t->rchild;

}

}

}

int main() {

bintree t;

t = creatbintree(); /\*??????????t??洢??\*/

printf("??????????????????\n");

preorder1(t); /\*???????????????\*/

return 0;

}

6.2

#include "bintree.h"

char \*a = "ABC##D#E##F##"; /\*扩充二叉树序树t的前序序列\*/

void levelbintree(bintree t)

{

bintree quene[100];

int front, top;

front = 0;

quene[front] = t;

top = 1;

while (front < top)

{

if (t->lchild)

{

quene[top++] = t->lchild;

}

if (t->rchild)

{

quene[top++] = t->rchild;

}

printf("%5c", t->data);

t = quene[++front];

}

}

int main()

{

bintree t;

t = creatbintree(); /\*建立二叉树t的存储结构\*/

printf("二叉树的层次序列为：\n");

levelbintree(t); /\*层次遍历二叉树\*/

return 0;

}

6.3

#include "bintree.h"

char \*a="ABC##D##EF#G###"; /\*扩充二叉树序树t的前序序列\*/

bintree prelast(bintree t)

{

seqstack s;

bintree end;

init(&s);

while(s.top != -1 || t !=NULL)

{

if(t)

{

end = t;

push(&s, t);

t = t->lchild;

}

else

{

t = pop(&s);

t = t->rchild;

}

}

return end;

}

bintree prelast1(bintree t)

{

while(t){

if(t->rchild){

t= t->rchild;

}else if(t->lchild){

t = t->lchild;

}else{

return t;

}

}

}

bintree postfirst(bintree t)

{

while(t){

if(t->lchild){

t = t->lchild;

}else if(t->rchild){

t = t->rchild;

}else{

return t;

}

}

}

int main()

{ bintree t,p,q;

t=creatbintree(); /\*建立二叉树t的存储结构\*/

p=prelast1(t);

q=postfirst(t);

if (p!=NULL)

{ printf("前序遍历最后一个结点为：%c\n",p->data);

printf("后序遍历第一个结点为：%c\n",q->data);

}

else printf("二叉树为空！");

return 0;

}

6.4

#include "bintree.h"

char \*a="ABC##D##EF#G###"; /\*扩充二叉树序树t的前序序列\*/

/\*

函数Depth，功能：求结点x所在的层次

\*/

int Depth(bintree t,char x)

{

int m, n;

if(!t) return -1;

if(t->data == x) return 1;

m = Depth(t->lchild, x);

if( m != -1){

return m+1;

}else{

n = Depth(t->rchild, x);

if( n != -1){

return n + 1;

}else{

return -1;

}

}

}

int main()

{ bintree root;

char x;

int k=0;

root=creatbintree();

printf("请输入树中的1个结点值：\n");

scanf("%c",&x);

k=Depth(root,x);

printf("%c结点的层次为%d\n",x,k);

}

6.5

#include "bintree.h"

char \*a="ABC##D##EF#G###"; /\*扩充二叉树序树t的前序序列\*/

/\*请将本函数补充完整，并进行测试\*/

void change(bintree t)

{

bintree temp; /\*将t的左右结点交换\*/

if (t)

{

if( t->lchild ) change( t->lchild ); /\*交换左子树\*/

if( t->rchild ) change( t->rchild ); /\*交换右子树\*/

temp = t->lchild;

t->lchild = t->rchild;

t->rchild = temp;

}

}

int main()

{ bintree root;

root=creatbintree();

change(root);

preorder(root);

}