山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验题目：二叉树操作 | | | |
| 实验学时：4 | | 实验日期：2021年11月23日星期二 | |
| 实验目的：  掌握二叉树的基本概念，二叉树的存储结构使用链表。 | | | |
| 硬件环境：  lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz | | | |
| 软件环境：  CLion 2020.3.3 x64 | | | |
| 实验步骤与内容：  1、 输入一个完全二叉树的层次遍历字符串，创建这个二叉树，输出这个二叉树的前序遍历字符串、中序遍历字符串、后序遍历字符串、结点数目、二叉树高度(上述每一个结果独立一行显示)。  2、 输入二叉树前序序列和中序序列(各元素各不相同)，创建这个二叉树，输出该二叉树的后序序列、层次遍历。 | | | |
| 结论分析与体会：  分析：  1.通过层次遍历添加二叉树的方法：  添加变量：返回的结果根；临时过渡值；二叉树数组  指向父节点的int 一开始是1，指向现在的节点的int，一开始是0，在循环里进行递增  每一次循环： 先将数值存在temp节点内，其左右子树都是空。把现在的节点++，让它代表的序号是当前的第n个节点。把它放到数组的第n个位置。如果是第一个，就把它放到root里面。如果不是，  分析这个节点的位置now，如果是奇数，一定是右子树；如果是偶数，则是左子树。是谁的：父节点的。父节点是谁：之前添加到了数组里的索引为1开始，每次now是奇数的时候，父节点加一。一开始的nodearray【1】是根节点，然后向后一次链接。  2.如何打印最后一个不是“，”：利用静态变量counter，但是记得每次循环之后要重置为0；  3.利用层次遍历：  利用队列queue：  打印根节点数字，如果左子树不空，则将其放入queue里面；右子树不空也放进去；利用temp.front方法先弹出左子树作为tree的新值，再循环，打印右子树  队列：先进先出 | | | |
|  | | | |

**本次实验主要代码：**

#include <iostream>

#include<cmath>

#include <queue>

using namespace std;

struct treeNode{

char ele;

treeNode \* left;

treeNode \* right;

treeNode(char \_ele,treeNode\* \_left,treeNode\* \_right){

ele = \_ele;

left = \_left;

right = \_right;

}

};

treeNode\* createByLevelOrder(char \*ele,int length){

treeNode \* root=NULL;

treeNode \* temp=NULL;

treeNode \* nodeArr[length+1];

int father=1,now = 0;

for (int i = 0; i < length; ++i) {

temp = new treeNode(ele[i],NULL,NULL);

now++;

nodeArr[now] = temp;

if (now==1)root = temp;

else{

if (nodeArr[father]){

if (now%2==0) nodeArr[father]->left=temp;

else nodeArr[father]->right=temp;

}

//每加两个就换一个父亲

if (now%2==1)father++;

}

}

return root;

}

void preView(treeNode \* tree,char end){

if (tree!=NULL){

if (tree->ele!=end)cout << tree->ele<<",";

else cout << tree->ele<<endl;

preView(tree->left,end);

preView(tree->right,end);

}

}

static int mycount = 0;

void inView(treeNode \* tree,int length){

if (tree!=NULL){

inView(tree->left,length);

if (mycount<length-1){

cout << tree->ele<<",";

mycount++;

}

else cout << tree->ele<<endl;

inView(tree->right,length);

}

}

static int mycount1 = 0;

void postView(treeNode \* tree,int length){

if (tree!=NULL){

postView(tree->left,length);

postView(tree->right,length);

if (mycount1<length-1){

cout << tree->ele<<",";

mycount1++;

}

else cout << tree->ele<<endl;

}

}

static int mycount2 = 0;

void levelView(treeNode \*tree,int length){

queue<treeNode\*> temp;

while (tree!=NULL){

if (mycount2<length-1){

cout << tree->ele<<",";

mycount2++;

} else {

cout << tree->ele<<endl;

return;

}

if (length==1)return;

if (tree->left!=NULL){

temp.push(tree->left);

}

if (tree->right!=NULL){

temp.push(tree->right);

}

try {

tree=temp.front();

}catch(error\_code){return;}

temp.pop();

}

}

treeNode\* createByPAndI(char \*pre,char \*in,int length){

if (length==0)return NULL;

treeNode \*result = new treeNode(pre[0],NULL,NULL);

if (length==1)return result;

int left1=0,right1=0,left2=0,right2=0;

char leftIn[length],rightIn[length],leftPre[length],rightPre[length];

for (int i = 0; i < length; ++i) {

if (i <= left1 && in[i] != pre[0]) {

leftIn[left1++] = in[i];//left1:左子树结点个数

} else if (in[i] != pre[0]) {

rightIn[right1++] = in[i];

}

}

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

if (i<left1+1){

leftPre[left2++]=pre[i];

} else{

rightPre[right2++]=pre[i];

}

}

result->left= createByPAndI(leftPre,leftIn,left1);

result->right= createByPAndI(rightPre,rightIn,right1);

return result;

}

int main() {

cout <<"Input1"<<endl;

string enter;

cin >> enter;

int size = enter.size();

char result[size/2+1];

int j=0;

for (int i = 0; i < size; ++i) {

if (enter[i]!=','){

result[j]=enter[i];

j++;

}

}

treeNode\* node1;

node1 = createByLevelOrder(result,j);

cout <<"Output1"<<endl;

preView(node1,result[j-1]);

inView(node1,j);

postView(node1,j);

cout << j<< endl;

cout << (int)(log(j)/log(2))+1<<endl;

mycount=0;mycount1=0;mycount2=0;

cout << "Input2"<<endl;

string enter1;string enter2;

cin >>enter1;

cin >> enter2;

char\* result1 = &enter1[0];

char\* result2 = &enter2[0];

cout << "Output2"<<endl;

treeNode\* node2 = createByPAndI(result1,result2,enter1.size());

postView(node2,enter1.size());

levelView(node2,enter1.size());

cout <<"End0"<<endl;

return 0;

}