**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

3、二叉树的应用 (必做)（二叉树）

[问题描述]

编程实现二叉树的建立，先序、中序、后序、层序遍历（递归和非递归方法），二叉树的高度，交换左右子树，统计叶子节点的数目，判断是否为完全二叉树。

[基本要求]

（1） 从文件中读入建树信息，树的节点数目不小于20个，树的高度不小于4。

（2） 采用二叉链表结构。

采用的数据结构：

二叉链表结构

算法思想：

二叉树的遍历

源程序：

#include<iostream>

#include<stack>

#include<fstream>

#include<queue>

using namespace std;

#define N 40

struct binary\_tree{

int data;

binary\_tree \* left;

binary\_tree \* right;

binary\_tree(int a = 0):data(a),left(NULL),right(NULL) {}

};

fstream f1,f2;

void CreateBiTree(binary\_tree\* &BT)

{

int ch;

if(!f1.eof())

{

f1>>ch;

cout<<ch<<" ";

if(ch==0) BT=NULL;

else{

BT=new binary\_tree;

BT->data=ch;

// cout<<"left "<<ch<<endl;

CreateBiTree(BT->left);

// cout<<"right "<<ch<<endl;

CreateBiTree(BT->right);

}

}

}

void insert(binary\_tree \*cur,binary\_tree \* p)

{

if(cur->data>p->data)

{

if(cur->left==NULL)

{

cur->left=p;

}

else

{

insert(cur->left,p);

}

}else

{

if(cur->right==NULL)

{

cur->right=p;

}

else

{

insert(cur->right,p);

}

}

}

void bfs(binary\_tree \*root)

{

queue<binary\_tree \*> q;

q.push(root);

while(!q.empty())

{

binary\_tree \* p;

p=q.front();

q.pop();

cout<<p->data<<" ";

if(p->left!=NULL) q.push(p->left);

if(p->right!=NULL) q.push(p->right);

}

cout<<endl;

}

binary\_tree\* read\_num(binary\_tree \* root)

{

int tmp;

binary\_tree \*p,\*q;

root=NULL;

p=root;

int n=N;

while(n--)

{

tmp=rand()%10000;

if(n==N-1)

tmp=3000;

cout<<tmp<<endl;

q = new binary\_tree;

q->data=tmp;

if(p!=NULL){

insert(root,q);

}

else{

root=q;

p=root;

}

}

return root;

}

void pre\_read(binary\_tree \* root)

{

if(root!=NULL)

{

cout<<root->data<<" ";

pre\_read(root->left);

pre\_read(root->right);

}

}

void mid\_read(binary\_tree \* root)

{

if(root!=NULL)

{

mid\_read(root->left);

cout<<root->data<<" ";

mid\_read(root->right);

}

}

void post\_read(binary\_tree \* root)

{

if(root!=NULL)

{

post\_read(root->left);

post\_read(root->right);

cout<<root->data<<" ";

}

}

void mid\_read\_while(binary\_tree \*root)

{

binary\_tree \*p;

p=root;

stack<binary\_tree\*>q;

while(!q.empty()||p)

{

while(p)

{

q.push(p);

// f2<<p->data<<" ";

p=p->left;

}

// f2<<'0'<<" ";

p=q.top();

q.pop();

cout<<p->data<<" ";

p=p->right;

}

cout<<endl;

}

void pre\_read\_while(binary\_tree \*root)

{

binary\_tree \*p;

p=root;

stack<binary\_tree\*>q;

while(!q.empty()||p)

{

while(p)

{

cout<<p->data<<" ";

f2<<p->data<<" ";

q.push(p);

p=p->left;

}

f2<<'0'<<" ";

p=q.top();

q.pop();

p=p->right;

}

f2<<'0'<<" ";

cout<<endl;

}

void post\_read\_while(binary\_tree \*root)

{

binary\_tree \*p,\*last;

p=root;

stack<binary\_tree\*>q;

last=NULL;

while(p)

{

q.push(p);

p=p->left;

}

while(!q.empty())

{

p=q.top();

q.pop();

if(p->right==NULL||p->right==last)

{

cout<<p->data<<" ";

last=p;

}

else

{

q.push(p);

p=p->right;

while(p)

{

q.push(p);

p=p->left;

}

}

}

cout<<endl;

}

int check\_is\_sorted(binary\_tree \*root)

{

binary\_tree \*p;

int last;

p=root;

int flag=0;

stack<binary\_tree\*>q;

while(!q.empty()||p)

{

while(p)

{

q.push(p);

p=p->left;

}

p=q.top();

q.pop();

if(!flag)

{

last=p->data;

flag=1;

}

// cout<<p->data<<" ";

if(p->data>=last)

{

last=p->data;

}

else

{

return 0;

}

p=p->right;

}

cout<<endl;

return 1;

}

int test\_file()

{

binary\_tree \*root;

f1.open("bin3.txt",ios::in);

// f2.open("bin3.txt",ios::out);

root=read\_num(root);

CreateBiTree(root);//两种模式

/\* cout<<"bfs: "<<endl;

bfs(root);

cout<<"pre\_read: "<<endl;

cout<<"post\_read: "<<endl;

post\_read\_while(root);

\*/ cout<<"mid\_read: "<<endl;

mid\_read\_while(root);

// pre\_read\_while(root);

if(!check\_is\_sorted(root))

cout<<"this tree is not sorted"<<endl;

else

cout<<"this tree is sorted"<<endl;

// f2.close();

f1.close();

return 0;

}

void bin\_h(binary\_tree \*root,int &max\_height,int cur)

{

if(cur>max\_height)

max\_height=cur;

if(root->left)

{

bin\_h(root->left,max\_height,cur+1);

}

if(root->right)

{

bin\_h(root->right,max\_height,cur+1);

}

}

int bin\_height(binary\_tree \*root)

{

int max\_height=0;

bin\_h(root,max\_height,1);

return max\_height;

}

void exchange(binary\_tree \*root)

{

if(root)

{

binary\_tree \*p;

p=root->left;

root->left=root->right;

root->right=p;

exchange(root->left);

exchange(root->right);

}

}

void exchange\_bin\_tree(binary\_tree \*root)

{

exchange(root);

}

void count\_leave(binary\_tree \*root,int &count)

{

// cout<<root->data<<endl;

if(root->left)

{

count\_leave(root->left,count);

}

if(root->right)

{

count\_leave(root->right,count);

}

if(!root->left&&!root->right)

{

count++;

// cout<<endl<<root->data<<" ";

}

}

int count\_leaves(binary\_tree \*root)

{

int count=0;

count\_leave(root,count);

return count;

}

int main()

{

binary\_tree \*root;

f1.open("bin3.txt",ios::in);

cout<<"Create Tree :"<<endl;

CreateBiTree(root);

cout<<endl<<"Recurrene: "<<endl;

cout<<endl<<"pre\_read:"<<endl;

pre\_read(root);

cout<<endl<<"mid\_read:"<<endl;

mid\_read(root);

cout<<endl<<"pre\_read:"<<endl;

post\_read(root);

cout<<endl<<"non Recurrence: "<<endl;

cout<<endl<<"pre\_read\_while:"<<endl;

pre\_read\_while(root);

cout<<endl<<"mid\_read\_while:"<<endl;

mid\_read\_while(root);

cout<<endl<<"post\_read\_while:"<<endl;

post\_read\_while(root);

cout<<endl<<"binary\_tree\_height: "<<endl;

cout<<bin\_height(root)<<endl;

cout<<endl<<"exchange\_left\_and\_right\_tree: "<<endl;

exchange\_bin\_tree(root);

cout<<endl<<"mid\_read:"<<endl;

mid\_read(root);

cout<<endl<<endl<<"count\_binary\_tree\_leaves:"<<endl;

cout<<count\_leaves(root)<<endl;

cout<<"check\_if\_the\_tree\_is\_sorted: "<<endl;

if(!check\_is\_sorted(root))

cout<<"this tree is not sorted"<<endl;

else

cout<<"this tree is sorted"<<endl;

exchange\_bin\_tree(root);

if(!check\_is\_sorted(root))

cout<<"this tree is not sorted"<<endl;

else

cout<<"this tree is sorted"<<endl;

return 0;

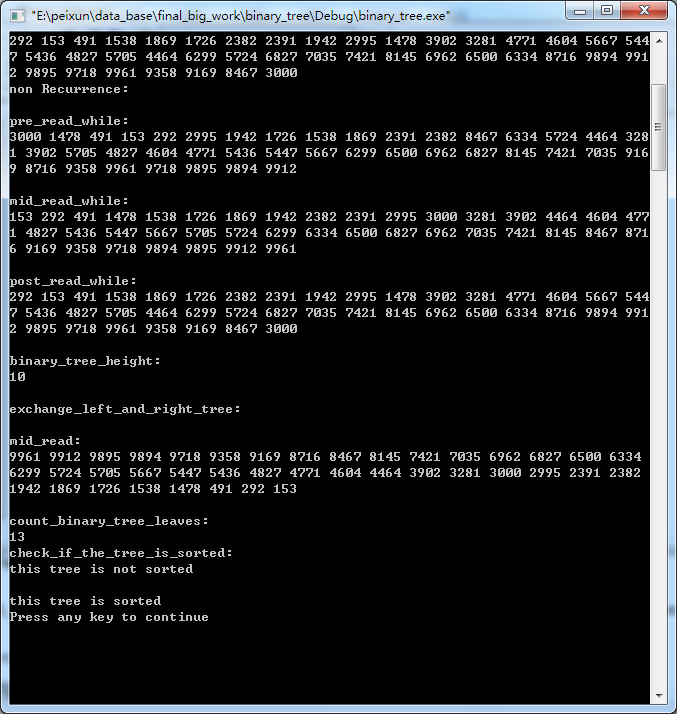
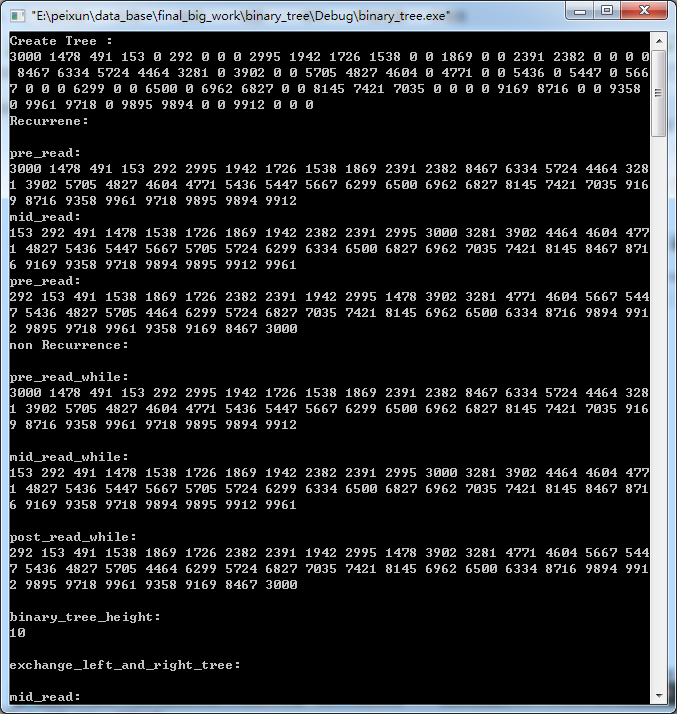
}

测试数据：

bin3.txt

3000 1478 491 153 0 292 0 0 0 2995 1942 1726 1538 0 0 1869 0 0 2391 2382 0 0 0 0 8467 6334 5724 4464 3281 0 3902 0 0 5705 4827 4604 0 4771 0 0 5436 0 5447 0 5667 0 0 0 6299 0 0 6500 0 6962 6827 0 0 8145 7421 7035 0 0 0 0 9169 8716 0 0 9358 0 9961 9718 0 9895 9894 0 0 9912 0 0 0

测试结果：



代码行数：375行