二叉树的构建（递归法）+中序遍历

#include <iostream>

using namespace std;

typedef char DataType;

//二叉树数据结构

struct node

{

    DataType info ; //存放结点数据

    struct node \*lchild , \*rchild ; //指向左右孩子的指针

};

typedef struct node \*BiTree ;

/\*创建二叉树

  函数名：createBiTree

  参数：无

  返回值：二叉树根结点指针

  \*/

BiTree createBiTree(void)

{

    //请在此处填写代码， 完成二叉树和创建，返回值是二叉树的根结点指针

    /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/

    BiTree node = new struct node;

    cin >> node->info;

    if(node->info=='#'){

        delete node;

        return nullptr;

    }

    else{

        node->lchild=createBiTree();

        node->rchild=createBiTree();

    }

    return node;

    /\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*/

}

void visit(BiTree T) //输出结点T的数据

{

    cout<<T->info ;

}

void inOrder(BiTree root)

{

    if(root == NULL) return ;

    inOrder(root->lchild);

    visit(root);

    inOrder(root->rchild);

}

int main(void)

{

    BiTree root = createBiTree();

    inOrder(root);

}

后序遍历得到 树叶的模板

int countLeaf(BiTree root)

{

    //请在此处填写代码， 计算二叉树中树叶的个数

    /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/

    if(root==nullptr){

        return 0;

    }

    if(root->lchild==NULL&&root->rchild==NULL){

        return 1;

    }

    int lef=countLeaf(root->lchild);

    int rig=countLeaf(root->rchild);

    return lef+rig;

    /\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/

}

void changeLR(BiTree root)

{

    //请在此处填写代码， 完成二叉树左右子树互换

    /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/

    if(root==nullptr){

        return ;

    }

    BiTree lef= root->rchild ;

    BiTree rig=root->lchild;

    root->lchild=lef;

    root->rchild=rig;

    changeLR(root->lchild);

    changeLR(root->rchild);

    return ;

    /\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/

}

int countFullNode(BiTree root)

{

    //请在此处填写代码，计算二叉树中满结点的个数

    /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/

    if(root==nullptr){

        return 0;

    }

    if(root->lchild==nullptr&&root->rchild==nullptr){

        return 0;

    }

    int lef=countFullNode( root->lchild);

    int rig=countFullNode(root->rchild);

    if(root->lchild!=nullptr&&root->rchild!=nullptr){

       return lef+rig+1;

    }

    return lef+rig;

    /\*\*\*\*\*\*\*\*\*\*\* End-\*\*\*\*\*\*\*\*\*\*/

}

二叉树的顺序结构构建方法模板

 #include <stdio.h>

 /\*

 定义顺序存储结构

  \*/

  #define  MAX 1024

  struct seqTree

  {

    int n;

    int idx;

    char node[MAX];

  };

//按照层序遍历来初始化顺序存储结构

/\*辅助功能：逐个输出顺序表的元素，元素之间以空格为界\*/

void printTree(struct seqTree \*T)

{

   for(int i=0;i<T->n;i++){

      printf("%c ",T->node[i]);

   }

 return ;

}

/\*第一关\*/

struct seqTree \*createSeqTree()

{

  //此处编写代码实现二叉树的初始化，包括输入数据

 struct seqTree \* root= (struct seqTree \*) malloc(sizeof(struct seqTree ));

 root->n=0;

 root->idx=0;

 int i=0;

 char key;

 scanf("%c",&key);

 while(key!='\n'){

   root->node[i++]=key;

   if(key!=' ') root->idx++;

   root->n++;

  scanf("%c",&key);

 }

return root;

}

/\*第二关，返回二叉树的根结点的值，若二叉树为空，则返回#\*/

char root(struct seqTree \*T)

{

 if(T->idx==0) return '#';

 return T->node[0];

}

/\*第二关，求二叉树T中指定结点ch的双亲结点，返回值是双亲结点的下标，若双亲不存在，则返回-1\*/

int parent(struct seqTree \*T ,char ch)

{

     int key=0;

    for(int i=0;i<T->idx;i++){

      if(T->node[i]==ch){

        key=i/2-1;

        break;

      }

    }

    if(key>=0){

      return key;

    }

    return -1;

}

/\*第二关，求二叉树T中指定结点ch的左孩子的下标，若左孩子不存在，则返回-1\*/

int leftChild(struct seqTree \*T ,char ch)

{

 int key=0;

    for(int i=0;i<T->idx;i++){

      if(T->node[i]==ch){

        key=2\*i+1;

        break;

      }

    }

    if(key<T->idx){

      return key;

    }

    return -1;

}

/\*第二关，求二叉树T中指定结点ch的右孩子的下标，若左孩子不存在，则返回-1\*/

int rightChild(struct seqTree \*T ,char ch)

{

int key=0;

    for(int i=0;i<T->idx;i++){

      if(T->node[i]==ch){

        key=2\*i+2;

        break;

      }

    }

      return key;

}

/\*第三关：层序遍历二叉树，输出遍历得到的结点，结点之间不需要空格\*/

void levelOrder(struct seqTree \*T )

{

for(int i=0;i<T->n;i++){

     if(T->node[i]!=' '){

 printf("%c",T->node[i]);

     }

   }

return ;

}

/\*第四关：先序遍历二叉树，输出遍历得到的结点，结点之间不需要空格\*/

void preOrder(struct seqTree \*T ,int child){

  if(child>=T->n) return ;

  if(T->node[child]==' ') return;

  printf("%c",T->node[child]);

  preOrder(T ,child\*2+1);

preOrder(T ,child\*2+2);

return;

}

int main(void)

{

  struct seqTree \*T = createSeqTree();

  /\*printTree(T); \*///测评第一关时，把本行代码放开

  /\*printf("%c\n",root(T));  //测评第二关时，把该代码块放开

    printf("%d\n",leftChild(T,'A'));

    printf("%d\n",rightChild(T,'A'));

    printf("%d\n",parent(T,'A'));\*/

   /\*levelOrder(T);\*///测评第三关时，把本行代码放开

   preOrder(T,0); //测评第四关时，把本行代码放开

}