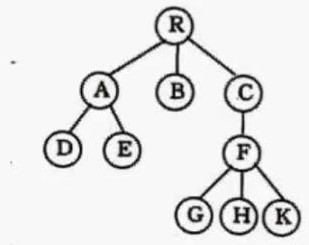
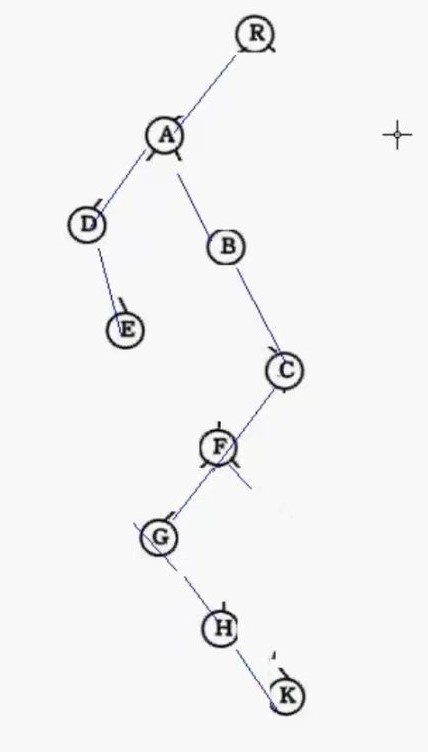
**树与森林以及与二叉树的转换（孩子兄弟方法实现）**

**一、普通的树如何转化成二叉树？**

孩子往左边放，兄弟往右边放

** **

森林同理，将根结点视为兄弟

**二、树的方法实现：**

#include <iostream>

#include<malloc.h>

#include<assert.h>

using namespace std;

#define ElemType char

typedef struct TreeNode {

ElemType data;

struct TreeNode\* firstChild; //第一个孩子结点（成为该节点的左树）

struct TreeNode\* nextSibling; //下一个兄弟结点（成为该结点的右树）

}TreeNode;

typedef struct Tree {

TreeNode\* root;

ElemType refvalue;

};

void InitTree(Tree&tree,ElemType value) {

tree.root = NULL;

tree.refvalue = value;

}

//是以树等价的二叉树来创建

//在变成之前先将普通树或森林转化成二叉树

void CreatTree(Tree &tree, TreeNode\*&t,const char\*&str) { //这里结点指针要带引用

if (\*str == tree.refvalue) {

t = nullptr;

}

else {

t = (TreeNode\*)malloc(sizeof(TreeNode));

assert(t != nullptr);

t->data = \*str;

CreatTree(tree, t->firstChild, ++str);

CreatTree(tree, t->nextSibling, ++str);

}

}

void CreatTree(Tree& tree,const char\*str) {

CreatTree(tree,tree.root,str);

}

//求根结点

TreeNode\* Root(Tree& tree) {

return tree.root;

}

//求整棵树的第一个孩子结点

TreeNode\* FirstChild(TreeNode\* t) {

if (t == nullptr) {

return NULL;

}

}

TreeNode\* FirstChild(Tree& tree) {

return FirstChild(tree.root);

}

//求整棵树的第一个兄弟结点

TreeNode\* NextSibling(TreeNode\* t) {

if (t == nullptr) {

return NULL;

}

else {

return t->nextSibling;

}

}

TreeNode\* NextSibling(Tree& tree) {

return NextSibling(tree.root);

}

//查找结点是否存在(还是从二叉树出发)

TreeNode\* Find(TreeNode\* t, ElemType key) {

if (t == nullptr) {

return NULL;

}

if (t->data == key) {

return t;

}

TreeNode\* p = Find(t->firstChild, key);

if (p != nullptr) {

return p; //先查找孩子树，如果找到了，直接返回，不用再查找兄弟树了

}

return Find(t->nextSibling, key); //如果不在左树并且存在 肯定在右树 直接返回即可

}

//重点

//找结点的父结点(此时不能将树转化成二叉树做)

//左树好搞，因为左树都是孩子结点

//但是右树都是兄弟结点

TreeNode\* Parent(TreeNode\* t, TreeNode\* p) {

if (t == nullptr || p == nullptr || p == t) {

return nullptr;

}

TreeNode\* q = t->firstChild;

TreeNode\* parent;

while (q != nullptr && q != p) {

parent = Parent(q, p);

if (parent != nullptr) {

return parent;

}

q = q->nextSibling; //如果左树找不到，则进入右兄弟里找

}

if (q != NULL && q == p) {

return t;

}

return NULL;

}

TreeNode\* Parent(Tree tree, TreeNode\* p) {

return Parent(tree.root,p);

}

int main() {

const char\* str = "RAD#E##B#CFG#H#K#####";

Tree mytree;

InitTree(mytree,'#');

CreatTree(mytree, str);

}