* **题目：先序递归过程建立二叉树**
* **班级：2021211312 姓名：杨天宇 学号：2021211281** 分工： **1,2**
* **班级：2021211312 姓名：王昊钰 学号：2021211282** 分工： **5,6**
* **班级：2021211312 姓名：李志茂 学号：2021211289** 分工： **3,4**

1. 需求分析

本实验需要完成的任务为通过输入的数据，按照先序递归序列建立并打印出一棵二叉树；其中，输入形式为一个长度不超过10000的字符串，该字符串需为二叉树的先序遍历序列，当某结点的左子树或右子树为空时，输入应为“\*”，如果输入格式正确，程序会纵向输出输入二叉树的形态，否则输出”error input”

1. 概要设计

程序主要通过调用两个递归子函数实现功能，第一个子函数为：void createtree(tree \*root);其输入为二叉树的根节点指针，作用为根据二叉树的先序序列建立二叉树，第二个子函数为：void printree(tree \*root,int level);输入为二叉树的根节点指针和结点的层次（根节点的层次为0，其余根据层次依次加1），函数作用为纵向输出二叉树的形态，主函数通过依次调用这两个子函数实现程序功能

1. 详细设计

typedef struct node //定义树的结点

{

struct node\* lc;

struct node\* rc;

char data;

} tree;

char \*num=(char\*)malloc(sizeof(char)\*10000);

int judge=1;

void createtree(tree \*root);

void printree(tree \*root,int level);

int main () //主函数

{

gets(num);

tree \*root=(tree\*)malloc(sizeof(tree));

createtree(root);

if(judge==1)

printree(root,0);

system("pause");

return 0;

}

void createtree(tree \*root) //创建二叉树

{

if(!root) return;

root->data=\*num;

num++;

if(\*num!='\*'&&\*num!=0)

{

root->lc=(tree\*)malloc(sizeof(tree));

createtree(root->lc);

}

else if(\*num==0)

{

printf("error input");

exit(0);

}

else

{

root->lc=NULL;

num++;

}

if(\*num!='\*'&&\*num!=0)

{

root->rc=(tree\*)malloc(sizeof(tree));

createtree(root->rc);

}

else if(\*num==0)

{

printf("error input");

exit(0);

}

else

{

root->rc=NULL;

num++;

}

}

void printree(tree\* root,int level) //横向打印二叉树

{

int i;

if(!root)

{

for(i=0;i<level;i++)

printf(" ");

printf("\n");

return;

}

printree(root->rc,level+1);

for(i=0;i<level;i++)

printf(" ");

printf("%c\n",root->data);

printree(root->lc,level+1);

}

调用关系：

main()

printree()

createtree()

4、调试分析报告

调试过程中，发现打印出的树的长宽比例不协调，通过控制printree()中打印的空格个数加以解决，对程序的改进设想主要为：通过细化打印二叉树的步骤使打印出的二叉树更为对称，更加美观

算法的时间复杂度为O(n)，n为二叉树结点个数，空间复杂度为O(e)，e为二叉树结点的最大层次

5、用户使用说明

1. 启动程序

2. 输入数据按先序遍历所得序列输入，当某结点左子树或右子树为空时，输入‘\*’号

3. 纵向输出二叉树的形态

4. 按任意按键结束程序

6、测试结果

输入 a

输出error input

输入 a\*

输出 error input

输入 a\*\*

输出

a

Press any key to continue . . .

输入abc\*\*d\*\*e\*\*

输出

e

a

d

b

c

Press any key to continue . . .

输入 abcdefg\*\*\*\*\*\*\*\*

输出

a

b

c

d

e

f

g

Press any key to continue . . .

输入abd\*\*e\*\*cf\*\*g\*\*

输出

g

c

f

a

e

b

d

Press any key to continue . . .

输入abd\*\*e\*\*cf\*\*g\*

输出 error input

输入abd\*\*e\*\*c\*g\*\*

输出

g

c

a

e

b

d

Press any key to continue . . .