### 问题点一：

CreateBiTree(BiTree \*T)

{

char c;

scanf("%c",&c);

if(' '==c)

{

\*T = NULL;

}

else

{

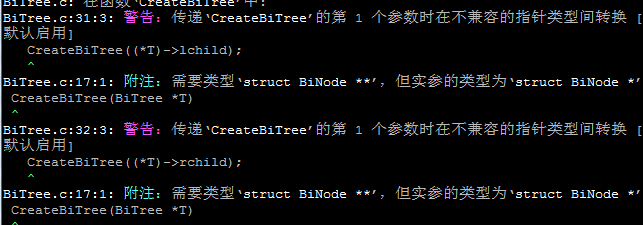
\*T = (BiNode\*)malloc(sizeof(BiNode));

(\*T)->data = c;

CreateBiTree((\*T)->lchild);错误！

CreateBiTree((\*T)->rchild);错误！

}

}

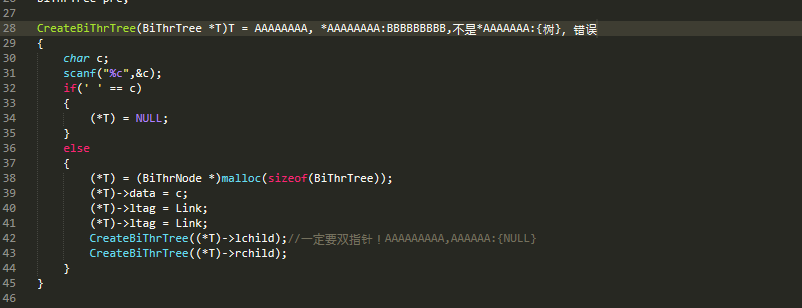
lchild和rchild就是一个地址，地址传入给星号的，应该可以呀？

在vc中编译，就会给提醒，而不一定错误

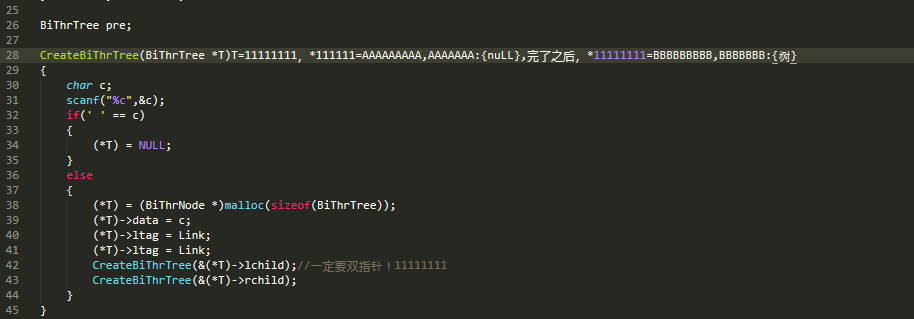
## 为什么这里要用多重指针？

凡是涉及到你传进来的（\*T）我需要重新分配指针的情况，都要二级指针；

为啥，因为重新分配的话，如果是一级指针aaaaaaaa,T=xxxxx,新的bbbbbbbb会把aaaaaaaa给替换掉，而aaaaaaaa里面的值没改变



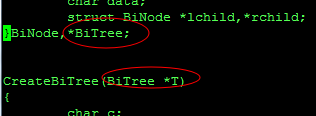
如果是11111111，T=1111111111，\*111111111=bbbbbbbb,就是aaaaaa



为什么会报错？有平衡检测机制的

不报错，可以加了星号逻辑是不是走不通了？通了！

## 如何判断需要类型‘struct BiNode \*\*’？



看得出，BiTree 本来就是一个指针了

小甲鱼的技巧：利用编译器的提示，看看指针平不平衡

代码：

/\*\*

\*

\* BiTree.c 二叉树的建立和遍历

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2018-2019

\*/

#include <stdio.h>

#include <stdlib.h>

typedef struct BiNode

{

char data;

struct BiNode \*lchild,\*rchild;

}BiNode,\*BiTree;

CreateBiTree(BiTree \*T)

{

char c;

scanf("%c",&c);

if(' '==c)

{

\*T = NULL;

}

else

{

\*T = (BiNode\*)malloc(sizeof(BiNode));

(\*T)->data = c;

CreateBiTree(&(\*T)->lchild);

CreateBiTree(&(\*T)->rchild);

}

}

void visit(char c,int level)

{

printf("%c located in %d layer\n",c,level);

}

void PreOrderTraverse(BiTree T,int level)

{

if(T)

{

visit(T->data,level);

PreOrderTraverse(T->lchild,level+1);

PreOrderTraverse(T->rchild,level+1);

}

}

void MidOrderTraverse(BiTree T,int level)

{

if(T)

{

MidOrderTraverse(T->lchild,level+1);

visit(T->data,level);

MidOrderTraverse(T->rchild,level+1);

}

}

int main()

{

printf("BiTree.c Test\n");

BiTree T;

int level=1;

CreateBiTree(&T);

printf("\n PreOrderTraverse :\n");

PreOrderTraverse(T,level);

level =1;

printf("\n MidOrderTraverse :\n");

MidOrderTraverse(T,level);

return 0;

}

