#include<stdio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

/\*定义二叉树结构体\*/

/\*word表示节点，left指针及right指针均表示左右子节点\*/

struct tnode{

char \*word;

int count;

struct tnode \*left;

struct tnode \*right;

};

/\*用于分配结构体内存\*/

struct tnode \*talloc(void)

{

return (struct tnode \*) malloc (sizeof(struct tnode));//注意此处有一个强制转换，原本是返回一个无类型的指针

}/\*用于赋值字符串到返回值指针上面\*/

char \*strdup1(char \*s)

{

char \*p;

p=(char \*)malloc(strlen(s)+1);//分配一定空间，注意此处要加上最后一个字符串结束符的空间1

if(p!=NULL)

strcpy(p,s);

return p;

}/\*打印树\*/

void treeprint(struct tnode \*p)

{

if(p!=NULL) {

treeprint(p->left);

printf("%4d %s\n",p->count,p->word);

treeprint(p->right);

}

}/\*添加一个节点给一棵树上\*/

struct tnode \*addtree(struct tnode \*p,char \*w)

{

int cond;

if(p==NULL) {

p=talloc();//若p是第一个根节点，则为其分配内存

p->word=strdup1(w);//赋值给指针

p->count=1;//计数为1

p->left=p->right=NULL;//初始化左子树右子树均为NULL

}

else if(strcmp(w,p->word)==0)//若这个字符串与已有的节点相同，则计数加1

p->count++;

else if(cond<0)//若比它小，则加到左子树上，递归调用

p->left=addtree(p->left,w);

else //若比它大，则加到右子树，递归调用

p->right=addtree(p->right,w);

return p;

}/\*主函数，加字符串到二叉树中，并且打印该二叉树\*/

int main(void)

{

// char \* word;

struct tnode \*root;

root=NULL;

printf("请输入'+'添加一个节点字符串，输入'!'退出程序\n\n");

int c;

while((c=getchar())!='!') {

if(c=='+') {

printf("输入要添加的字符串,不得超过10个字符\n");

char s[10];

scanf("%s",s);

root=addtree(root,s);

printf("打印已有的树结构如下:\n");

treeprint(root);

printf("输入'+'继续添加，输入'!'退出程序\n\n");

}

}

}