数据结构实验报告实验四-二叉树

一、实验环境：

Windows 10 64位 MinGW GCC 4.8.1

IDEA：CLoin

二、实验内容：

编写程序实现一棵二叉树，并实现实现二叉树的创建，各种遍历（递归和非递归），求深度，求节点数量等功能

三、算法流程：

输入一颗二叉树的先序遍历序列，’#’代表空节点，其他字符为非空节点元素。

利用输入的先序遍历序列递归的建立一颗树。

递归中序遍历树。

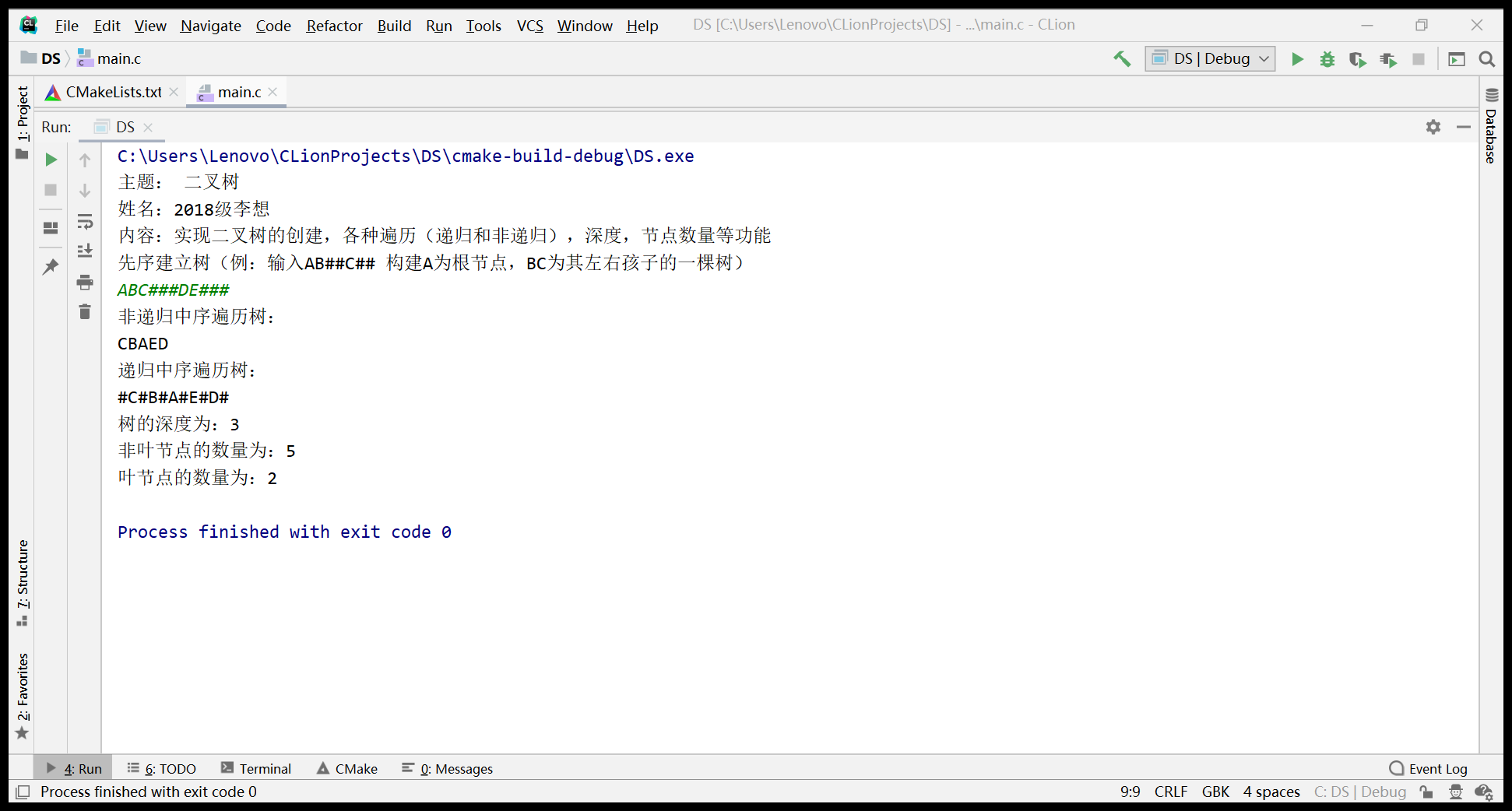
手动实现一个栈来实现二叉树非递归的中序遍历。

通过递归方法来求得二叉树的深度，非叶节点的数量，叶节点的数量。

四、程序代码：

#define **MAXSIZE** 10  
#define **INCRE** 2  
#define **OVERFLOW** -1  
#include**<stdlib.h>**#include**<stdio.h>  
typedef struct** BitNode \* ElemType;  
**typedef struct** stack{  
 ElemType \*data;  
 **int** top;  
 **int** maxsize;  
}stack,\*Stack;  
**typedef struct** BitNode{  
 **char** data;  
 **struct** BitNode \*lchild,\*rchild;  
}BiTNode,\*BiTree;  
Stack initstack(){  
 Stack s;  
 s= (Stack)malloc(**sizeof**(Stack));  
 s->data= (ElemType\*)malloc(**MAXSIZE**\***sizeof**(ElemType));  
 s->top=0;  
 s->maxsize=**MAXSIZE**;  
 **return** s;  
}  
**int** stackempty(Stack s){  
 **if**(s->top==0)  
 **return** 1;  
 **else  
 return** 0;   
}  
**int** IsFull(Stack s){  
 **if**(s->top==s->maxsize)  
 **return** 1;  
 **else   
 return** 0;  
}  
**void** push(Stack s,ElemType i){  
 ElemType \*newbase;  
 **if**(IsFull(s)){  
 newbase=(ElemType\*)realloc(s->data,(s->maxsize+**INCRE**)\***sizeof**(ElemType));  
 **if**(!newbase)exit(**OVERFLOW**);  
 s->data= newbase;  
 s->maxsize+=**INCRE**;  
 s->data[s->top]=i;  
 s->top++;  
 printf(**"full\n"**);  
 }   
 **else**{  
 s->data[s->top]=i;  
 s->top++;  
 }  
}  
ElemType pop(Stack s){  
 ElemType e;  
 **if**(!stackempty(s)){  
 s->top--;  
 e=s->data[s->top];  
 **return** e;  
 }  
 **else** printf(**"empty\n"**);  
}  
BiTree CreateTree(){  
 BiTree T;  
 **char** ch;  
 scanf(**"%c"**,&ch);  
 **if**(ch==**'#'**)  
 T=**NULL**;  
 **else**{  
 **if** (!(T=(BiTNode\*)malloc(**sizeof**(BiTNode))))  
 exit(**OVERFLOW**);  
 T->data=ch;  
 T->lchild=CreateTree();  
 T->rchild=CreateTree();  
 }  
 **return** T;  
}  
**void** inorder1(BiTNode\* T){  
 **if**(T!=**NULL**){  
 inorder1(T->lchild);  
 printf(**"%c"**,T->data);  
 inorder1(T->rchild);  
 }  
 **else** printf(**"#"**);  
}  
**void** inorder(BiTNode\* T){  
 Stack s;  
 BiTree p,q;  
 p=T;  
 s=initstack();  
 **if** (!(q=(BiTNode\*)malloc(**sizeof**(BiTNode))))  
 exit(**OVERFLOW**);  
 **while**(p!=**NULL**||!stackempty(s)){  
 **if**(p!=**NULL**){  
 push(s,p);  
 p=p->lchild;  
 }  
 **else**{  
 q=pop(s);  
 printf(**"%c"**,q->data);  
 p=q->rchild;  
 }  
 }  
}  
**int** Depth(BiTree t){  
 **int** m,n;  
 **if**(!t)  
 **return** 0;  
 **else**{  
 m=Depth(t->lchild);  
 n=Depth(t->rchild);  
 **return** m>n?m+1:n+1;  
 }  
}  
**int** NodeCount(BiTree t){  
 **if**(!t)  
 **return** 0;  
 **else**{  
 **return** NodeCount(t->lchild)+NodeCount(t->rchild)+1;  
 }  
}  
**int** LeafCount(BiTree t){  
 **if**(!t)  
 **return** 0;  
 **else**{  
 **if**(!t->lchild&&!t->rchild)  
 **return** 1;  
 **else**{  
 **return** LeafCount(t->lchild)+LeafCount(t->rchild);  
 }  
 }  
}  
**int** main(){  
 BiTree t;  
 **int** depth,nodecount,leafcount;  
 printf(**"主题： 二叉树\n"**);  
 printf(**"姓名：2018级李想\n"**);  
 printf(**"内容：实现二叉树的创建，各种遍历（递归和非递归），深度，节点数量等功能\n"**);  
 printf(**"先序建立树（例：输入AB##C## 构建A为根节点，BC为其左右孩子的一棵树）\n"**);  
 t=CreateTree();  
 printf(**"非递归中序遍历树：\n"**);  
 inorder(t);  
 printf(**"\n"**);  
 printf(**"递归中序遍历树：\n"**);  
 inorder1(t);  
 printf(**"\n"**);  
 depth=Depth(t);   
 printf(**"树的深度为：%d\n"**,depth);  
 nodecount=NodeCount(t);   
 printf(**"非叶节点的数量为：%d\n"**,nodecount);  
 leafcount=LeafCount(t);   
 printf(**"叶节点的数量为：%d\n"**,leafcount);  
 **return** 0;  
}

五、运行截图



六、注意事项

在利用递归来遍历一颗二叉树时，有时会因为递归次数过多造成栈溢出，所以就需要自己来实现一个栈，来用迭代方法遍历二叉树。