南京信息工程大学 数据结构 实验(实习)报告

实验(实习)名称 实验(实习)日期 得分 指导老师

系专业 班级 姓名 学号

一、实验目的

掌握二叉树遍历的基本操作，并对其进行简单应用。

1. 实验内容与步骤

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct BinaryTreeNode{

char data;

struct BinaryTreeNode \*lchild, \*rchild;

}BinaryTreeNode, \*pBinaryTreeNode;

typedef struct {

pBinaryTreeNode \*stack;

int top;

int stacksize;

}MyStack;

void initstack(MyStack \*s,int stacksize)

{

s->stacksize = stacksize;

s->top = -1;

s->stack = (pBinaryTreeNode\*)malloc(sizeof(pBinaryTreeNode)\*(s->stacksize));

}

int StackEmpty(MyStack \*s)

{

if (s->top < 0) return 1;

return 0;

}

void StackPush(MyStack \*s, pBinaryTreeNode pdata)

{

if (s->top == s->stacksize - 1)

exit(1);

else{

s->stack[++(s->top)] = pdata;

}

}

pBinaryTreeNode StackTop(MyStack \*s)

{

if (StackEmpty(s) == 1)

exit(1);

else{

return s->stack[s->top];

}

}

void StackPop(MyStack \*s)

{

if (StackEmpty(s) == 1)

exit(1);

else{

s->top--;

}

}

void inorder(pBinaryTreeNode root){

MyStack s;

initstack(&s, 10);

pBinaryTreeNode current = root;

while (current || !StackEmpty(&s)){

if (current){

StackPush(&s, current);

current = current->lchild;

}

else{

current = StackTop(&s);

StackPop(&s);

putchar(current->data);

// printf("%d",current->data);

current = current->rchild;

}

}

}

pBinaryTreeNode CreatBinaryTree(){

pBinaryTreeNode node;

char data;

data=getchar();

if (data =='-')

return NULL;

node = (pBinaryTreeNode)malloc(sizeof(BinaryTreeNode));

node->data = data;

node->lchild = CreatBinaryTree();

node->rchild = CreatBinaryTree();

return node;

}

int main(){

pBinaryTreeNode rootDyn = NULL;

rootDyn = CreatBinaryTree();

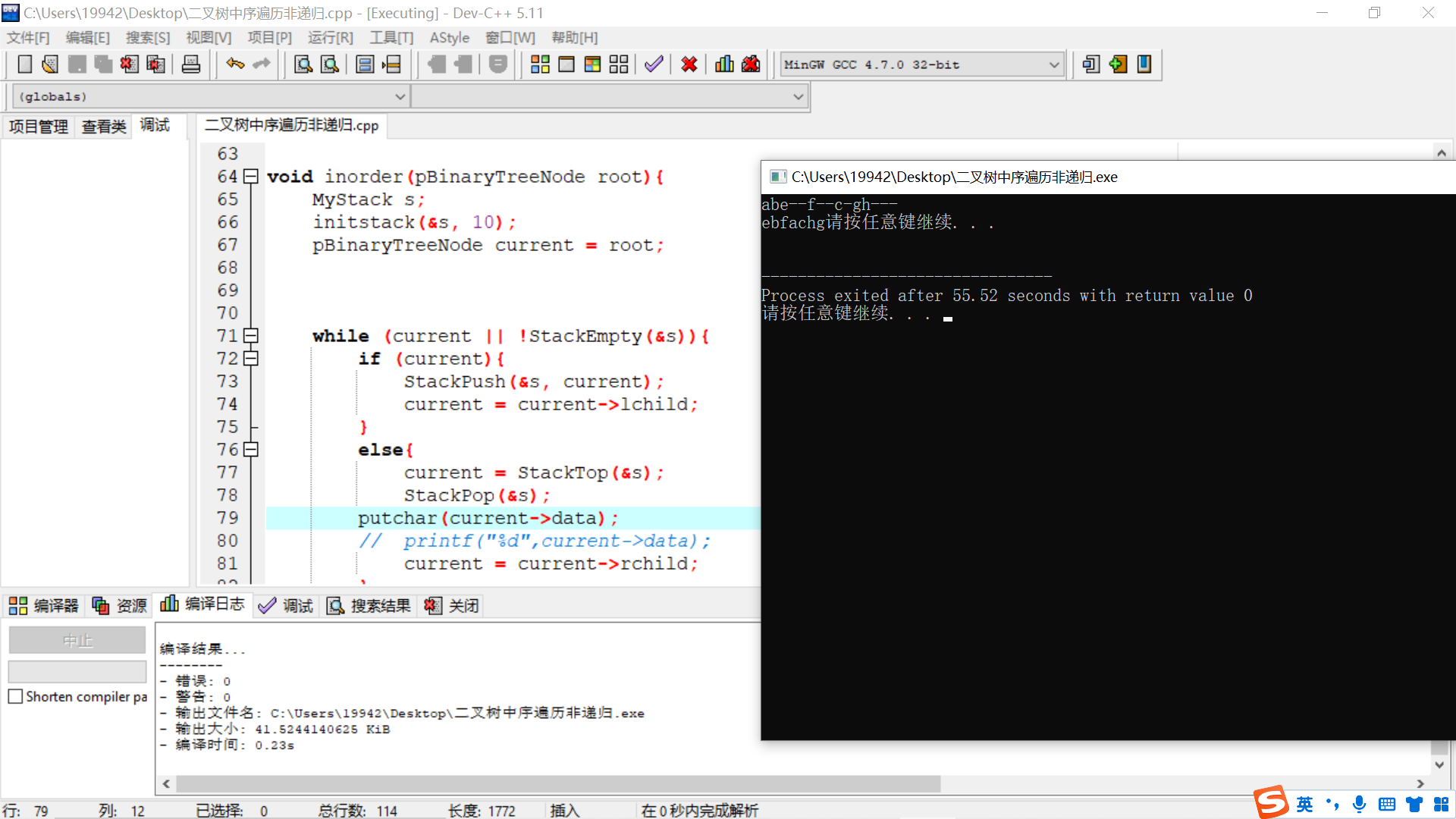
inorder(rootDyn);

system("pause");

printf("\n");

return 0;

}



三、体会和总结

非递归要借助顺序栈，递归算法更加简洁。