

二叉树实验报告

课程名称：数据结构与算法

学 院： 计算机与信息安全学院

专业班级：19003603

学 号：1900301517

姓 名：陆洪业

报告日期：2021年 01 月 10 日

目录

**[1. 二叉树的创建及遍历输出](#_Toc475353216_WPSOffice_Level1)** **[1](#_Toc475353216_WPSOffice_Level1)**

[1.1. 选用储存结构](#_Toc569951487_WPSOffice_Level2) [1](#_Toc569951487_WPSOffice_Level2)

[1.2. 实验代码](#_Toc59923446_WPSOffice_Level2) [1](#_Toc59923446_WPSOffice_Level2)

[1.3. 系统测试与结果](#_Toc1996587104_WPSOffice_Level2) [13](#_Toc1996587104_WPSOffice_Level2)

[1.4. 实验小结](#_Toc211562106_WPSOffice_Level2) [13](#_Toc211562106_WPSOffice_Level2)

**[2. 二叉树的应用](#_Toc569951487_WPSOffice_Level1)** **[14](#_Toc569951487_WPSOffice_Level1)**

[2.1. 实验任务](#_Toc1834168774_WPSOffice_Level2) [14](#_Toc1834168774_WPSOffice_Level2)

[2.2. 实验代码](#_Toc1931779814_WPSOffice_Level2) [14](#_Toc1931779814_WPSOffice_Level2)

[2.3. 系统测试与实验结果](#_Toc971520243_WPSOffice_Level2) [19](#_Toc971520243_WPSOffice_Level2)

[2.4. 实验小结](#_Toc1956411600_WPSOffice_Level2) [20](#_Toc1956411600_WPSOffice_Level2)

# 二叉树的创建及遍历输出

完成二叉树的创建及遍历输出。

## 选用储存结构

树

## 实验代码

#include <iostream>

using namespace std;

typedef char DataType;

//二叉树结点定义

struct node

{

DataType data; //存放结点数据

struct node \*lchild, \*rchild ; //左右孩子指针

};

typedef struct node BiTree;

typedef struct node \*ptree;

// 栈

/\*---------begin----------\*/

struct stack\_node

{

ptree tnode;

int flag; // 0为无孩子,1为有左孩子无右孩子

struct stack\_node \*next;

};

typedef struct stack\_node \*PNode;

PNode createNullStack( )

{

PNode s = (PNode)malloc(sizeof(struct node));

if (NULL == s) {

printf("Out of Space!\n");

return NULL;

}

s->next = NULL;

s->tnode = NULL;

s->flag = 0;

return s;

}

int isNullStack(PNode s)

{//判断栈是否为空，若为空，返回值为1，否则返回值为0,若栈不存在，则返回-1

if (NULL == s)

return -1;

else if (NULL == s->next)

return 1;

else

return 0;

}

int push(PNode s ,ptree tnode)

{//在栈中插入数据元素x，若插入不成功，返回0；插入成功返回值为1

PNode p = (PNode)malloc(sizeof(struct node));

if (NULL == p)

return 0;

p->tnode = tnode;

p->flag = 0; // 新入栈的结点没有孩子

p->next = s->next;

s->next = p;

return 1;

}

ptree pop(PNode s)

{//弹栈并返回删除元素，若栈为空，则返回NULL

if (NULL == s || NULL == s->next)

return NULL;

PNode p = s->next;

ptree tnode = p->tnode;

s->next = s->next->next;

free(p);

p = NULL;

return tnode;

}

/\*---------end----------\*/

//函数可直接使用，功能：输出结点数据

void print(DataType d)

{

cout<<d<<" ";

}

/\*

函数名：createBiTree

函数功能：创建二叉树，并返回二叉树的根结点指针

参数：无

返回值：二叉树的根结点指针

\*/

BiTree \*createBiTree() {

//请在此处填写代码，完成创建二叉树并返回二叉树根结点指针的功能

/\*-------begin--------\*/

PNode s = createNullStack();

ptree tree = NULL;

ptree root = NULL;

int i = 0;

DataType str[80] = {};

DataType data;

cin >> str;

while ((data = str[i++]) != '\0') {

if(data == '#') {

root = NULL;

} else {

root = new BiTree;

if (NULL == tree)

tree = root;

root->data = data;

root->lchild = NULL;

root->rchild = NULL;

}

if(!isNullStack(s)) {

switch(s->next->flag) {

case 0:

s->next->tnode->lchild = root;

s->next->flag++;

break;

case 1:

s->next->tnode->rchild = root;

pop(s); // 若左右孩子均确定, 则该节点确立, 弹出构建栈

break;

}

}

// 如果新建结点不为空, 那么将该结点压入构建栈中

if (NULL != root)

push (s, root);

}

return tree;

/\*

ptree root = NULL;

DataType data;

cin >> data;

if (data == '#') {

return NULL;

} else {

root = new BiTree;

root->data = data;

root->lchild = createBiTree();

root->rchild = createBiTree();

}

return root;

\*/

/\*-------end--------\*/

}

/\*

函数名：preOrder

函数功能：先根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

void preOrder(BiTree \*T)

{

//请在此处填写代码，完成先根遍历二叉树功能

/\*-------begin--------\*/

if (T == NULL)

return;

ptree c = T;

PNode s = createNullStack();

do {

while (c != NULL) {

print(c->data);

push(s, c);

c = c->lchild;

}

while ((c == NULL) && (!isNullStack(s))) {

c = s->next->tnode->rchild;

pop(s);

}

} while (c != NULL);

/\*-------end--------\*/

}

/\*

函数名： inOrder

函数功能：中根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

void inOrder(BiTree \*T)

{

//请在此处填写代码，完成中根遍历二叉树功能

/\*-------begin--------\*/

if (T == NULL)

return;

ptree c = T;

PNode s = createNullStack();

do {

while (c != NULL) {

push(s, c);

c = c->lchild;

}

while ((c == NULL) && (!isNullStack(s))) {

c = s->next->tnode->rchild;

print(pop(s)->data);

}

} while (c != NULL);

/\*-------end--------\*/

}

/\*

函数名：postOrder

函数功能：后根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

void postOrder(BiTree \*T)

{

//请在此处填写代码，完成后根遍历二叉树功能

/\*-------begin--------\*/

if (T == NULL)

return;

ptree c = T;

PNode s = createNullStack();

do {

while (c != NULL) {

push(s, c);

if (s->next->flag >= 1) // 如果左孩子访问过了, 那么跳过左孩子的访问

break;

// 访问左孩子

c = c->lchild;

s->next->flag++;

}

while (!isNullStack(s)) {

if (1 == s->next->flag) { // 如果是从左孩子进来的, 那么访问右孩子

// 访问右孩子

c = s->next->tnode->rchild;

s->next->flag++;

break;

} else { // 如果已经访问完了左右孩子

// 访问该结点

print(s->next->tnode->data);

pop(s);

}

}

} while (!isNullStack(s));

/\*-------end--------\*/

}

int main(void)

{

BiTree \*T;

T = createBiTree( ); //调用创建二叉树功能，得到二叉树的根结点指针

preOrder( T );//调用先根遍历二叉树，按先根遍历顺序输出二叉树结点功能

cout<<endl; //换行

inOrder(T);//调用中根遍历二叉树，按中根遍历顺序输出二叉树结点功能

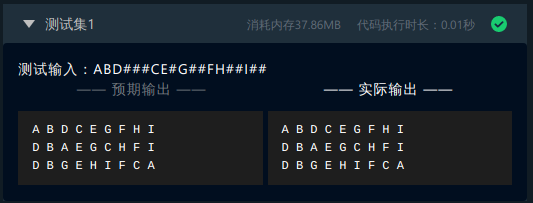
cout<<endl;

postOrder(T);//调用后根遍历二叉树，按后根遍历顺序输出二叉树结点功能

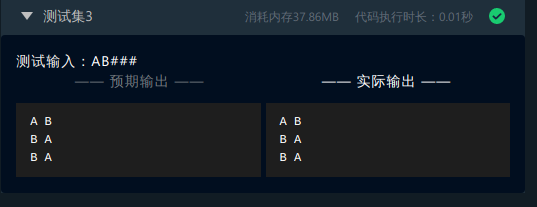
return 1;

}

## 系统测试与结果







上述测试及结果证明了系统测试通过,各项操作函数能正常执行,并达到了实验要求的目标,实现了使用栈来创建和遍历二叉树.

## 实验小结

通过本次实验, 我学会了用栈创建和遍历二叉树. 对将递归化成栈又有了进一步的了解.

# 二叉树的应用

## 实验任务

假设我们用一棵二叉树存储你的家族族谱： 对于任意一个结点来说，左孩子代表长子或长女，右孩子代表兄弟或姐妹。 结点存储的是每个人的代号。 本关任务要求：输入某人的代号，输出此人的所有兄弟姐妹。

## 实验代码

#include <iostream>

using namespace std;

#include <stdlib.h>

typedef char DataType;

//二叉树结点定义

//此处进行二叉树结构定义

/\*-------begin------\*/

typedef struct node BiTree;

struct node {

char data;

BiTree \* parent;

BiTree \* lchild;

BiTree \* rchild;

};

/\*-----end---------\*/

void print(DataType d)

{

cout<<d<<" ";

}

//初始化二叉树的结点序列

char treeData[] ="ABC##D##E#F#G##";

/\*

函数名：createBiTree

函数功能：读取treeData 数组中的字符序列进行二叉树创建二叉树，并返回二叉树的根结点指针

参数：无

返回值：二叉树的根结点指针

\*/

//在此处填入代码

/\*----------begin-------------\*/

BiTree \*createBiTree() {

BiTree \*root = NULL;

DataType data;

static int i = 0;

data = treeData[i++];

if (data == '#') {

return NULL;

} else {

root = new BiTree;

root->data = data;

root->parent = NULL;

root->lchild = createBiTree();

if (NULL != root->lchild)

root->lchild->parent = root;

root->rchild = createBiTree();

if (NULL != root->rchild)

root->rchild->parent = root;

}

return root;

}

/\*----------end-------------\*/

void Sibling(BiTree \*T )

{

//在此处填入代码,输出结点T的所有兄弟姐妹

/\*----------begin-------------\*/

BiTree \*p = NULL;

for (p = T; NULL != p->parent && p->parent->rchild == p; p = p->parent);

for (p = T; NULL != p; p = p->rchild)

cout << p->data << " ";

/\*----------end-------------\*/

}

/\*

函数名：preOrder

函数功能：先根遍历二叉树 ，并找到值为ch的结点指针

参数：根结点指针 BiTree \*T ,需查找的结点值 ch

返回值：无

\*/

//在此处填入代码,利用先序遍历，找到结点值ch后调用函数sibling输出该结点的所有兄弟姐妹，以空格分界

/\*----------begin-------------\*/

void preOrder(BiTree \*T, char ch)

{

if (T == NULL)

return;

if (T->data == ch)

Sibling(T->rchild);

preOrder(T->lchild, ch);

preOrder(T->rchild, ch);

}

/\*----------end-------------\*/

int main(void)

{

BiTree \*T;

T= createBiTree(); //创建二叉树

char ch ;

cin>>ch; //输入某人的代号

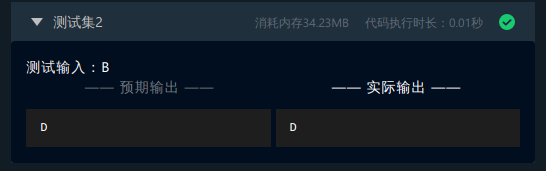
preOrder(T, ch); //调用函数输出ch的所有兄弟姐妹

return 1;

}

## 系统测试与实验结果





上述测试及结果证明了系统测试通过,各项操作函数能正常执行,并达到了实验要求的目标,实现了所有兄弟姐妹的输出.

## 实验小结

通过本次实验, 我学会了将栈使用在一些具体的问题上.