**西南石油大学实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目编号** | **001063000503** | **项目名称** | **二叉树的构造和遍历** | **成绩** |  |
| **专业年级** | **计算机科学与技术2022级** | | | **指导教师** | **胡卫东** |
| **姓名** | **王磊** | **学号** | **202231060435** | **实验日期** | **2023.06.03** |

1. **实验目的**

1、掌握二叉树链式存储的类型定义及实现。

2、掌握二叉树链式存储的各种基本运算方法。如二叉树的创建、遍历、查找等运算。

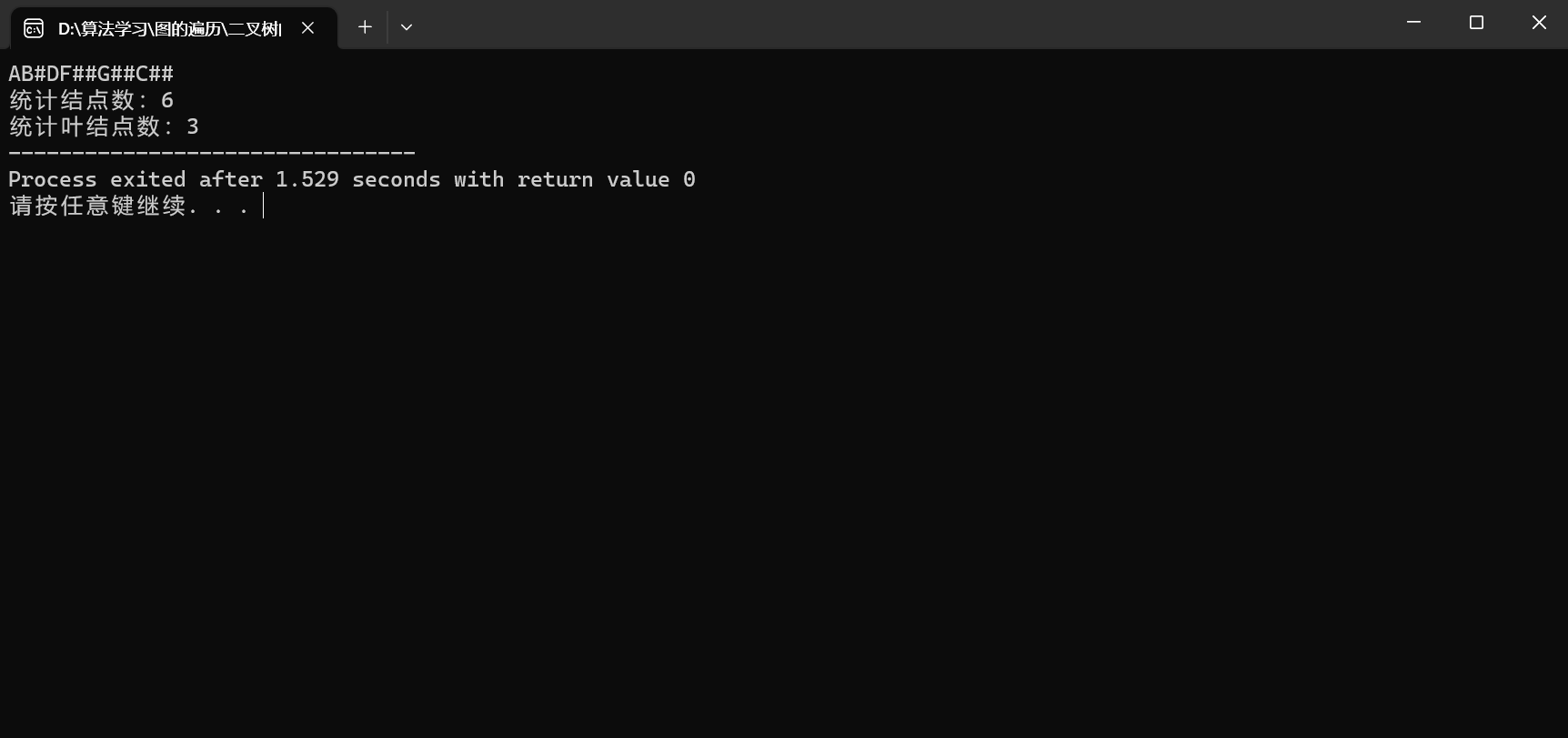
3、掌握二叉树用不同方法表示所对应的不同输入形式。

4、掌握二叉树中各重要性质在解决实际问题中的应用。

1. **实验工具**

PC微机，Windows，DOS，Turbo C或Visual C++

1. **实验步骤**
2. 先用双链表及createTree函数构建二叉树
3. 然后用countNode和countLeaves函数分别统计结点数和叶结点数
4. **实验结果**

****

1. **实验总结**

该代码使用递归实现了创建二叉树的函数 createTree，并提供了两个辅助函数 countNode 和 countLeaves，用于统计节点和叶节点的数量。通过递归遍历树的方式，对每个节点进行检查，根据节点是否有子节点来进行计数。

**2、求二叉树的宽度**

**（1）问题描述**

对任意建立的一颗二叉树求其中的结点数和叶结点数。

**（2）基本要求**

在输入数据创建二叉树的过程中，要求以二叉树遍历的先根顺序输入。统计结点总数和页结点总数的功能分别用一个函数编程实现，在主函数中调用相应子函数。在实现统计时，注意局部变量和全局变量对统计结果正确性的影响。

1. **算法描述**

**createTree 函数用于创建二叉树。首先从输入中读取一个字符 ch，如果 ch 为 '#'，表示当前节点为空，将节点设为 NULL，并返回。**

**如果 ch 不为 '#'，则创建一个新节点，并将 ch 赋值给节点的数据域。**

**递归调用 createTree 函数创建左（右）子树，将返回的节点赋值给当前节点的左（右）孩子指针 lchild（rchild）。**

**countNode 函数用于统计节点的数量。如果当前节点为空（NULL），返回 0。**

**如果当前节点的左孩子为空，但右孩子不为空，说明当前节点只有右子树，递归调用 countNode 函数统计右子树的节点数量，并加上当前节点本身（+1）。**

**如果当前节点的左孩子不为空，但右孩子为空，说明当前节点只有左子树，递归调用 countNode 函数统计左子树的节点数量，并加上当前节点本身（+1）。**

**如果当前节点既有左子树又有右子树，递归调用 countNode 函数分别统计左子树和右子树的节点数量，并将两者相加，再加上当前节点本身（+1）。**

**countLeaves 函数用于统计叶节点的数量。如果当前节点为空（NULL），返回 0。**

**如果当前节点不为空，并且左孩子和右孩子都为空，说明当前节点是叶节点，将统计叶节点的变量 ret 加一。**

**递归调用 countLeaves 函数分别统计左子树和右子树的叶节点数量。**

**最后返回统计叶节点的变量 ret。**

**这段代码的主函数中调用了上述三个函数，分别创建二叉树，并输出统计的节点数和叶节点数**

1. **算法实现——示例程序**

**#include <stdio.h>**

**#include <stdlib.h>**

**typedef struct node {**

**char data;**

**struct node \*lchild;**

**struct node \*rchild;**

**} \*Tree, TREE;**

**void createTree(Tree &T);**

**int countNode(Tree T);//统计结点**

**int countLeaves(Tree T);//统计叶结点**

**int ret = 0;**

**int main() {**

**Tree T;**

**createTree(T);**

**printf("统计结点数：%d\n", countNode(T));**

**printf("统计叶结点数：%d", countLeaves(T));**

**return 0;**

**}**

**void createTree(Tree &T) {**

**char ch;**

**scanf("%c", &ch);**

**if (ch == '#') {**

**T = NULL;**

**return ;**

**}**

**T = new TREE;**

**T->data = ch;**

**createTree(T->lchild);**

**createTree(T->rchild);**

**}**

**int countNode(Tree T) {**

**if (T == NULL)**

**return 0;**

**int i = 0;**

**if (T->lchild == NULL && T->rchild != NULL) {**

**return countNode(T->rchild) + 1;**

**} else if (T->lchild != NULL && T->rchild == NULL) {**

**return countNode(T->lchild) + 1;**

**}**

**return countNode(T->rchild) + countNode(T->lchild) + 1;**

**}**

**int countLeaves(Tree T) {**

**if (T == NULL) {**

**return 0;**

**}**

**if (T != NULL && T->lchild == NULL && T->rchild == NULL) {**

**ret++;**

**}**

**if (T != NULL) {**

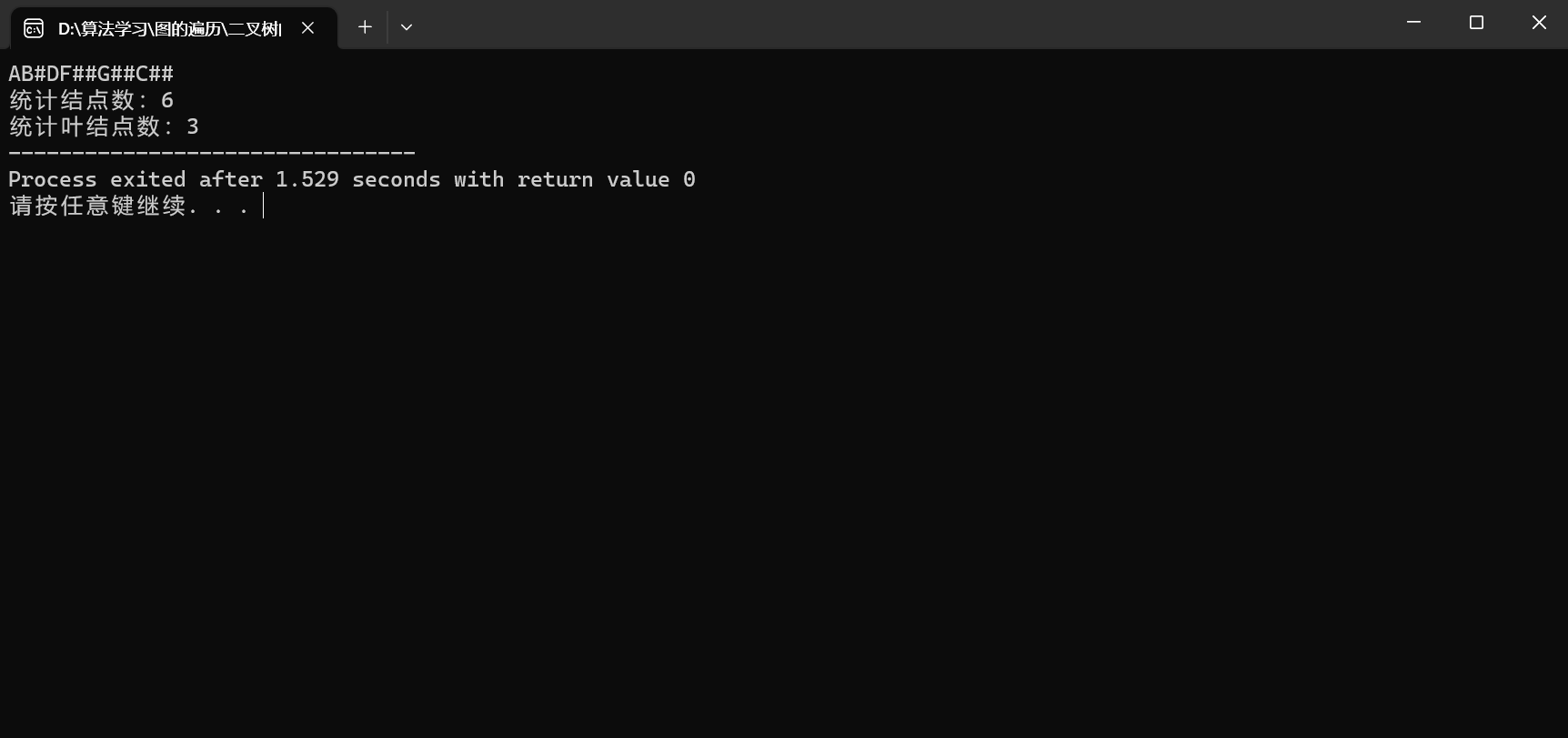
**countLeaves(T->lchild);**

**countLeaves(T->rchild);**

**}**

**return ret;**

**}**

****