**实验三：二叉树的实现**

**学号：B21080526 班级：B210416 姓名：单家俊**

1. **实验目的**

加深对二叉树以及其相关操作的理解。

1. **实验任务**

定义二叉树，其中每个元素的数据类型为int。实现二叉树的插入、遍历（先序、中序、后序任选其一），并按照得到的遍历顺序打印所有结点。

1. **实验设备**

Visual Studio

1. **实验内容（步骤）**

1、编写二叉树的数据结构。

2、通过用户输入实现二叉树的创建。

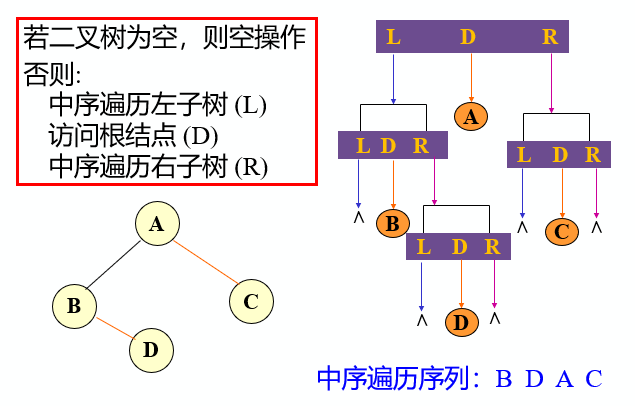
3、通过中序遍历遍历二叉树并将遍历结果打印出来。

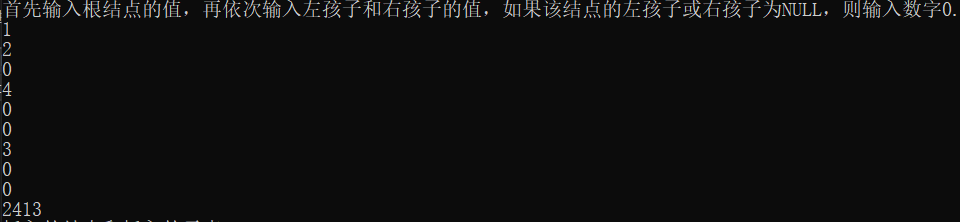
4、插入元素，将插入元素后的二叉树进行中序遍历并将遍历结果打印。

1. **实验结果（总结、分析）**

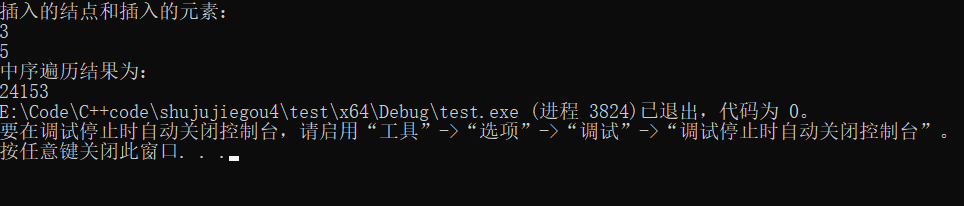
**实验结果分析：**

实现如下图所示二叉树

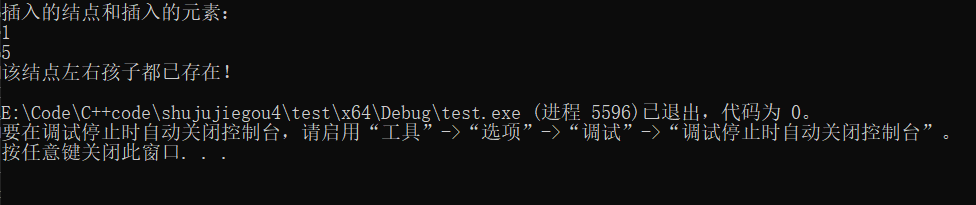




创建如图所示二叉树，根据提示依次输入120400300，创建成功后程序自动进行中序遍历，并将结果打印，结果为2413，正确。



若想在结点3上插入5（默认插在该节点的左孩子上，若左孩子已存在，则插在右孩子上），根据提示依次输入3和5，进行中序遍历的结果为24153，结果正确，表示成功插入。



如果我们选择操作的结点的左右孩子都已经存在，则会告诉我们插入失败。

**实验过程中遇到的问题：**

1、在编写完程序准备编译时，发现编译报错，告诉我访问冲突，经过查询资料后发现，发现原来是该指针一开始指向NULL，对该指针进行操作发生访问冲突。

2、在主函数中调用递归函数，遇到return我以为是直接返回主函数，查询资料后得知是返回上一层递归。

**体会：**

经历过几次数据结构的实验，我觉得，数据结构这门课程，首先是很有意思的，各种数据之间迷人的关系，树、哈夫曼树等有趣的结构。但是它们代码实现部分，是非常需要我们去细细琢磨的，每一个地方，该用什么样的结构体，该定义指针还是普通变量，返回的是一个什么类型的返回值，序号的计算过程是不是写对了，+1或是-1有没有考虑等等等等。如今学期已接近尾声，期末考试也即将来临，每一次实验都在考察我们对之前所学知识掌握的程度如何。我想通过每一次实验，我们都应该反思自己的真实水平，去复习巩固，去查漏补缺。每一次实验都是对我们的一次锤炼，我想我们应该珍惜每一次机会，认认真真地完成实验。

1. **附录（代码）**

#define OK 1

#define WRONG -1

typedef int TElemType;

typedef int Status;

#include<iostream>

using namespace std;

//定义二叉树的结构类型

typedef struct BiNode

{

TElemType data;

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

}BiNode, \* BiTree;

//建立二叉树

void CreateBiTree(BiTree& T)

{

int ch;

cin >> ch;

if (ch == 0) T = NULL; //递归结束，建空树

else

{

T = new BiNode;

T->data = ch; //生成根结点

CreateBiTree(T->lchild); //递归创建左子树

CreateBiTree(T->rchild); //递归创建右子树

}

}

//中序遍历

Status InOrderTraverse(BiTree T)

{

if (T == NULL) return OK; //空二叉树

else {

InOrderTraverse(T->lchild); //递归遍历左子树

cout << T->data; //访问根结点

InOrderTraverse(T->rchild); //递归遍历右子树

}

}

Status Find(BiTree &T,int k,int value)

{

if (T == NULL) return OK; //空二叉树

else {

Find(T->lchild,k,value); //递归遍历左子树

if (T->data == k)

{

if (T->lchild == NULL)

{

BiNode\* p = new BiNode;

p->data = value;

p->lchild = NULL;

p->rchild = NULL;

T->lchild = p;

return OK;

}

else

{

if (T->rchild == NULL)

{

BiNode\* p = new BiNode;

p->data = value;

p->lchild = NULL;

p->rchild = NULL;

T->rchild = p;

return OK;

}

else

return WRONG;

}

}

Find(T->rchild,k,value); //递归遍历右子树

}

}

int main()

{

int k, value, j;

BiTree T;

cout << "首先输入根结点的值，再依次输入左孩子和右孩子的值，如果该结点的左孩子或右孩子为NULL，则输入数字0." << endl;

CreateBiTree(T);

InOrderTraverse(T);

cout << endl;

cout << "插入的结点和插入的元素：" << endl;

cin >> k >> value;

j = Find(T, k, value);

if (j == OK)

{

cout << "中序遍历结果为：" << endl;

InOrderTraverse(T);

}

else

cout << "该结点左右孩子都已存在！" << endl;

return 0;

}