上机内容：第六章上机训练第二题：以二叉链表作存储结构，编写求二叉树中叶子结点数目的递归函数

姓名：杨阳　　学号：20121703

开发环境版本：Visual Studio 2019　　日期：2022/2/27

头文件个数：5　　源文件个数：1　　工程文件名：上机大作业.sln

1 实现原理

题目要求编写递归函数求二叉树中叶子结点的数目，故首先传入二叉树的根节点，判断所构造的二叉树的根节点是否存在，如果根节点为空，则返回叶子结点数目为0；如果根节点不为空，则判断其左右孩子结点是否为空，如果都为空，则返回叶子结点数目为1,；否则递归调用该函数继续向下查找判断二叉树的每一个结点是否有左右孩子，来求出叶子结点的数目，从而达到题目的要求。

以下为具体的代码实现：

```
int BinaryTree<ElemType>::CountLeaf(const BinTreeNode<ElemType>* r) const {
    if (r == NULL)
        return 0;
    else if (r->leftChild == NULL && r->rightChild == NULL)
        return 1;
    else
        return CountLeaf(r->leftChild) + CountLeaf(r->rightChild);
}
```

2 实现技术

2.1 使用 Assitance.h

在定义链队列类时，运用到了辅助软件包中的自定义类型，SUCCESS、OVER_FLOW、RANGE_ERROR等；并在测试程序中运用到了其中的辅助函数如SWAP()、Write()、Display()、等函数。

2.2 使用和改善 Binarytree.h及BinTreeNode.h

在教材程序的基础上删除了一些对于该题目要求冗余的功能，因为目的只是为了在测试程序时对构造的二叉树进行正确性验证所以不必大费周章；并且加入了CountLeaf()这一函数来实现题目的要求，具体实现细节已在上文的实现原理中阐明，这里不多赘述。

而在二叉树结点类编写时采用类模板的形式，这样可以更加方便地构造出不同数据类型的二叉树。

2.3 使用LinkQueue.h及Node.h

题目要求是以二叉链表作存储结构，因此这两个头文件分别以类模板的形式定义了链结点类和链队列类，为构造二叉链表类提供便利的同时，可以更加方便地构造出不同数据类型的二叉链表。

2.4 编写TestBinaryTree.cpp

笔者在测试程序中给出了以下两个序列：

```
char pre[]={'A','B','D','E','G','H','C','F','I'};    // 先序序列
char in[]={'D','B','G','E','H','A','C','F','I'};     // 中序序列
```

通过以上两个序列来构造二叉树，同时设计了插入、删除元素等操作，并通过先序遍历、中序遍历、求结点个数、二叉树高度来验证之前的操作的正确性。最后就是测试题目要求的求二叉树叶子结点个数的函数，具体的测试流程将在下文测试数据和结果中展示，

3 文件结构说明

文件名	文件类型	功能简介	备注
Assistance.h	CPP Header File	辅助软件包	无
BinaryTree.h	CPP Header File	定义二叉树类及对其功能的实现	无
BinTreeNode.h	CPP Header File	定义二叉树结点类	无
LinkQueue.h	CPP Header File	定义链队列类及对其功能的实现	无
Node.h	CPP Header File	定义链结点类	无
TestBinaryTree.cpp	CPP Source File	构造二叉树并测试所需功能	无

表1 文件结构说明

4 运行实测试数据和输出结果

4.1 由先序和中序序列构造二叉树

```
构造二叉树
由先序:A,B,D,E,G,H,C,F,I和中序:D,B,G,E,H,A,C,F,I构造的二叉树:
      A
   B      C
 D   E      F
   G   H      I

请按任意键继续. . .
```

4.2 在D处插入新元素Z后求其先序序列和中序序列

```
1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):1

输入被插入元素的值:D

输入插入元素的值:Z

1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):3

A  B  D  Z  E  G  H  C  F  I
```

```
1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):4

Z  D  B  G  E  H  A  C  F  I
```

4.3 求此时二叉树的结点个数和二叉树的高度

```
1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):5

二叉树的结点数为：10

1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):6

二叉树的高度为：4
```

4.4 求此时二叉树的叶子结点个数

```
1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):7

二叉树中叶子结点数目为：4
```

4.5 在C结点处插入新元素X并求此时叶子结点个数

```
1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):1

输入被插入元素的值:C

输入插入元素的值:X

1. 插入左孩子.
2. 删除右子树.
3. 先序遍历.
4. 中序遍历.
5. 求二叉树的结点数.
6. 求二叉树的高度.
7. 二叉树中叶子结点数目.
0. 退出
选择功能(0~7):7

二叉树中叶子结点数目为：5
```

5 个人体会建议

5.1 题目的难度

该题目的难度在于递归函数的设计，以及递归函数终止条件的设计，完成了这两部分的设计后，就是构造一个二叉树进行功能的验证。笔者在课本代码的基础上做了删减，已达到突出一些与该题相关的二叉树功能函数，以及统计叶结点树的这一题目要求。其次难度在于构造数据进行功能的验证和调试。

5.2 调试过程

笔者首先构造一个字母集，给出了他们的先序和中序序列，以此来构造一个二叉树，并在控制台上输出了构造完成后的树形结构，并一一试验功能是否可以成功进行，具体调试过程中产生的图片已在前文中展示，这里便不再赘述。

5.3 有关课程及研讨的建议

5.3.1 课程建议

数据结构课上除了原理的讲解，可能也需要加入一些代码的讲解，有助于同学们的理解，但可能是受限于课程进度的关系，删去了大部分代码讲解的过程，仅仅对一些难以理解的代码进行讲解分析，这也是可以理解的。

5.3.2 研讨建议

课程研讨的分工安排由学生代替安排时会出现分组、选题的小矛盾，如果老师能够在小组人数，小组组数等方面进行硬性的规定，可以很大程度上节省分组花费的时间，减少同学们因为分组而产生的不愉快情绪。另外，课程研讨扎堆在最后几周进行可以说是有利有弊，利在课程的持续性，弊在同学们的拖延导致的延后性以及完成研讨后同学上课的漫不经心。