5.2 中序线索二叉树

二叉树的非线性，给二叉树遍历带来麻烦。线索二叉树可能给遍历带来方便。

一、程序设计简介

本验证程序包括两个源程序文件 InThrBiTree.h 和 ThrBiTree.cpp。

（1）InThrBiTree.h

该文件中包括：

① 线索二叉树的结点 ThrBTNode 的定义；

② 基于上述结点定义的二叉树的建立 CreateBiTree()、显示 DispBiTree()和销毁 DestroyThrBiTree()；

③ 中序线索化二叉树（算法 5.13）、中序线索二叉树的中序遍历（算法 5.15）。

（2）主文件 ThrBiTree.cpp

主文件 LinkList.cpp 包括以下 3 个内容：

① 菜单定义 dispmanu()，用于用户功能选择的菜单显示函数

② 测试用例，二叉树创建的输入序列极易出错导致程序运行异常，所以，源程序中给出了 4 个测试用例，具体如下：

```
string  fbt="a b d # # e # # c f # # g # #";    // 满二叉树
string  cbt="a b d # # e # # c # #";        // 完全二叉树
string  gbt="a b # d # # c e # # #";        // 一般二叉树
string  obt="a b c d # # # # #";            // 左斜树
```

③ 主函数 main()，通过调用 InThrBiTree.h 中定义的操作实现程序的各功能，用菜单提供交互界面。

程序功能结构如图 1.5.3 示。

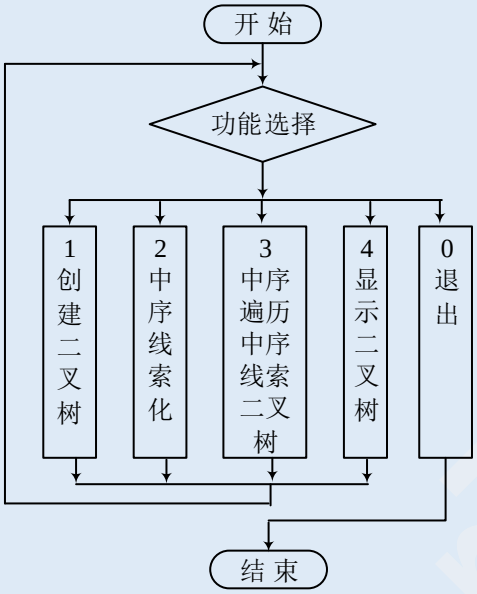

图 1.5.3 单链表验证程序结构示意

二、运行说明

运行程序，显示如图 1.5.4 所示界面。

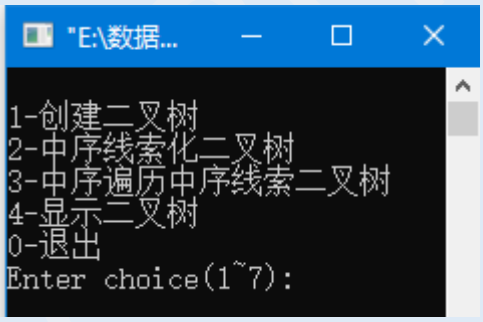

图 1.5.4 单链表验证程序运行界面

case 1：键入“1”，选择功能 1，创建二叉树。

按屏幕提示，输入所创建二叉树的先序遍历序列。创建成功，屏幕显示逆时钟转了 90 度的二叉树

case 2：键入“2”，选择功能 2，中序线索化二叉树

屏幕显示 “中序线索化成功”

case 3：键入“3”，选择功能 3，中序遍历中序线索化二叉树

屏幕显示二叉树的中序遍历序列

注：必须先执行功能 2，方可执行该操作。

case 4：键入“4”，选择功能 4，显示线索二叉树

按屏显示一棵转了 90 度的二叉树。

case 0：键入“0”，选择功能 0，结束运行

屏幕显示“结束运行 bye-bye!”，按任意键，结束程序运行。

三、思考题

1. 研读源程序，回答下列问题：

（1）基于线索二叉树结点创建二叉树、销毁二叉树、显示二叉树与二叉链表二叉树的创建、

销毁和显示算法上有区别吗？能共用吗？

（2） 函数 ThrBTNode<DT> * CreateInThread(ThrBTNode<DT> *&bt)的功能是什么？

（3） 中序线索二叉树的头指针指向根结点吗？

（4）中序遍历中序线索二叉树算法 void InThrBiTree(ThrBTNode<DT> * bt)是非递归算法，但是也没有用到栈，为什么？

（5）参照本验证程序的头文件“InThrBiTree.h”，编写先序线索二叉树的相关操作及算法 5.14 源码放于头文件“PreThrBiTree.h”中，验证算法 5.14 的正确性。

2. 运行程序，回答下列问题：

（6）创建一棵深度为 4 的拥有最多结点数二叉树，观察功能 2 和 3 的运行结果。

（7）创建给出一棵深度为 4 的拥有最少结点数二叉树，观察功能 2 和 3 的运行结果。

（8）创建一棵深度为 4 右斜树，观察功能 2 和 3 的运行结果。

（9）对于一棵一般二叉树，观察功能 2 和 3 的运行结果。

（10）如果没有先执行功能 “ 2-中序线索化二叉树”，直接执行功能“3-中序遍历中序线索二叉树”，会出现什么现象？分析原因。