Lab5

黄子骕 21307130013

本实验实现了如下函数：

Insert 函数与 insert_sub 函数：二者大致一样，区别是后者带有一个保存父结点的参数便于对插入的结点添加指向父节点的指针，insert 中有对于目标为空树的处理，若不为空树，则继续调用 insert_sub。

build_BST 函数：从终端读取键盘输入的数据，并利用 insert 函数插入数据。

Search_recursion 与 search_iterative 函数：分别用递归、迭代方法查找，若查不到则返回-1，查到了则返回 key 对应的 value。

Minimum 与 maximum 函数：返回以给定根结点子树中最小/最大 key 的得 value

Predecessor 与 successor 函数：根据两种 case 情况分别讨论找到前驱/后继，返回对应的 value

Successor_node 函数：方便写 delete 函数而写，与 successor 函数相同，区别是返回对应的 node

_delete 函数：根据 3 种 case 分别处理，无孩/1 孩/2 孩。

Preorder_treewalk、inorder_treewalk、Postorder_treewalk 函数：三种顺序遍历，区别为打印的时间点

实验结果：

将根节点设置为 key = 4， value = 6；初始化输入 2 3 1 9 6 7；测试 insert 与 delete 函数的 node，key = 5， value = 10；

```
PS D:\FDU\Sophomore_semester1\Data Structure\code_cpp> cd "d:\FDU\Sophomore_sem
ester1\Data Structure\code_cpp\" ; if ($?) { g++ BST.cpp -o BST } ; if ($?) { .
\BST }
2 3 1 9 6 7
^Z
minimum is: 9
maximum is: 7
result of search_recursion: 6
result of search_iterative: 6
the value of its successor is: 7
the value of its predecessor is: 3
10
-1
6397
9367
9376
PS D:\FDU\Sophomore_semester1\Data Structure\code_cpp>
```

以上均为以 root 结点为根的运行结果