

Binary Tree 实验报告

赵宸 2021201645

本次试验基本完成实验要求，选做任务选择了实现查找兄弟节点和查找祖先节点。基本实现了对错误输入的人性化处理。

内存泄漏检查

经过valgrind的内存泄漏检查，本程序没有出现内存泄漏现象,可放心食用。

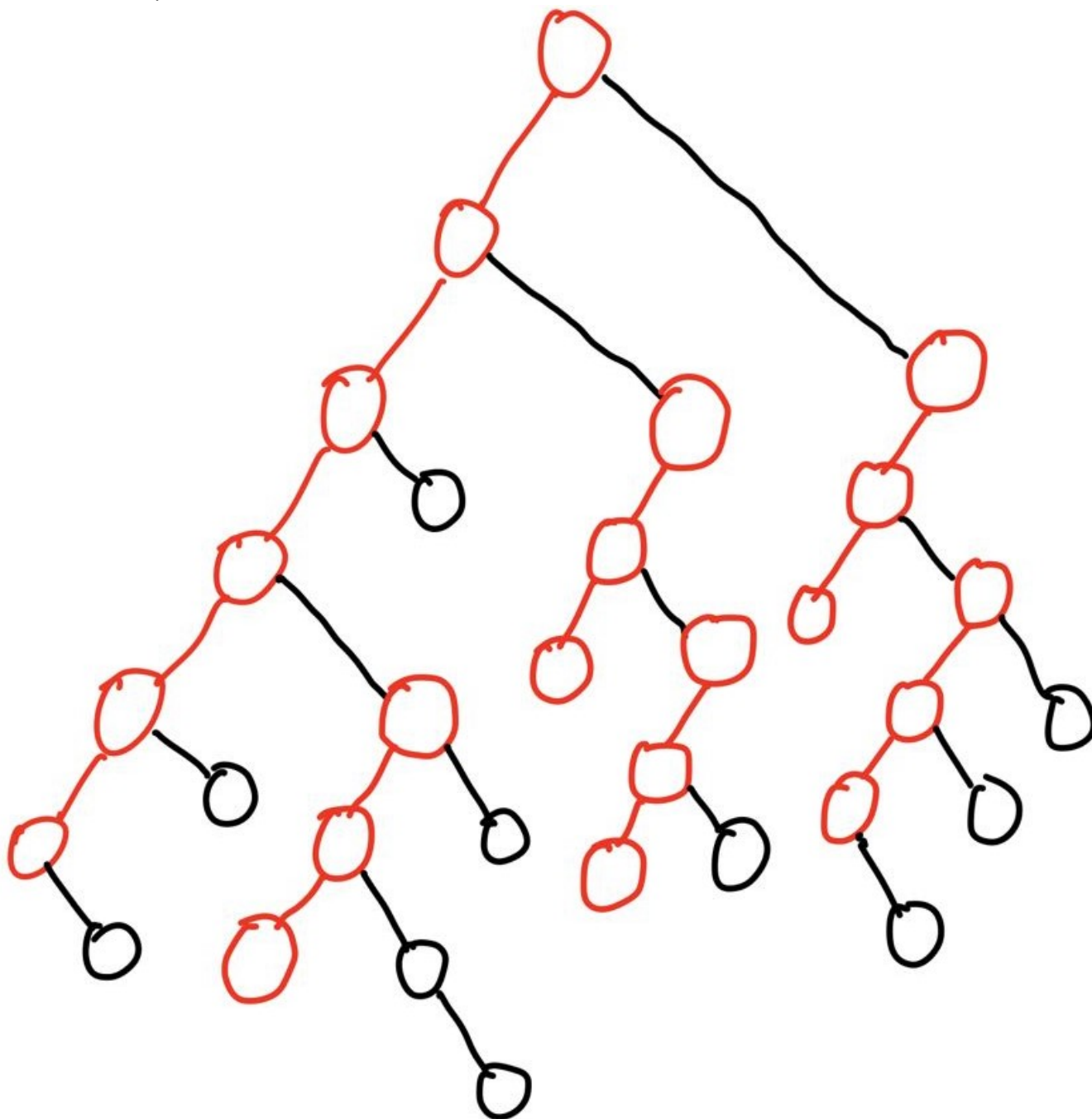
```
student@student-VirtualBox: ~/valgrind-3.19.0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
student@student-VirtualBox:~/valgrind-3.19.0$ sudo chmod -R 777 binarytree.cpp
[sudo] student 的密码:
student@student-VirtualBox:~/valgrind-3.19.0$ valgrind --tool=memcheck --leak-check=full ./binarytree.cpp
==2461== Memcheck, a memory error detector
==2461== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==2461== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==2461== Command: ./binarytree.cpp
==2461==
./binarytree.cpp: 6: ./binarytree.cpp: using: not found
: not found.cpp: 6: ./binarytree.cpp:
./binarytree.cpp: 21: ./binarytree.cpp: typedef: not found
: not found.cpp: 22: ./binarytree.cpp: {
: not found.cpp: 23: ./binarytree.cpp: RB_RED,
: not found.cpp: 24: ./binarytree.cpp: RB_BLACK
./binarytree.cpp: 25: ./binarytree.cpp: Syntax error: "}" unexpected
==2461==
==2461== HEAP SUMMARY:
==2461==    in use at exit: 1,929 bytes in 61 blocks
==2461==   total heap usage: 81 allocs, 20 frees, 9,193 bytes allocated
==2461==
==2461== LEAK SUMMARY:
==2461==    definitely lost: 0 bytes in 0 blocks
==2461==    indirectly lost: 0 bytes in 0 blocks
==2461==    possibly lost: 0 bytes in 0 blocks
==2461==    still reachable: 1,929 bytes in 61 blocks
==2461==    suppressed: 0 bytes in 0 blocks
==2461== Reachable blocks (those to which a pointer was found) are not shown.
==2461== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==2461==
==2461== For lists of detected and suppressed errors, rerun with: -s
==2461== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@student-VirtualBox:~/valgrind-3.19.0$
```

后序遍历迭代版实现思路

由于要先遍历所有的子节点然后再搞父节点，因此我便采用了老师上课所讲的“藤缠树：模型，也就是持续的向左遍历并压栈知道nullptr；弹栈的时候先查询右节点再遍历自身；如果右节点有叶子节点，那就继续压栈。重复以上过程直到栈空为止。

为了实现这个目的，首先我要写一个向左节点延伸的“爬藤”函数，然后向右分析叶子节点，分析到非空的节点

再次爬藤直到nullptr为止。下图是以上算法的**简略**图示：



parent和sibling的查询

由于没有特定的遍历函数要求，我在这里便使用`coutData`函数来代表遍历函数了。parent的寻找我采用递归的方法，一层一层的寻找压栈然后print，如果他根本没有parent那我就cout啥都没有来做特判；sibling的寻找也是同理，直接找他有没有parent，如果有那再找他的左右孩子节点是否全满，最后判断哪个是我输入的x，另一个便是我兄弟。

使用方法及运行结果

建立二叉树

首先要输入根节点的数值，然后会允许你选择：延伸左节点，延伸右节点，返回父节点，如果左右的data输入到已有元素的节点中会报错并退出。运行截图如下：

```

student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
student@student-VirtualBox:~/桌面$ ./using
Please give your root value.
1
*****
Please set your binary tree:
if you want to give the left-child value, cin 1.
if you want to give the right-child vlaue,cin 2.
if you want to come back the parent palce, cin 3.
if you want to quit, cin 4.
1
this is the left child.
please cin value:2
*****
Please set your binary tree:
if you want to give the left-child value, cin 1.
if you want to give the right-child vlaue,cin 2.
if you want to come back the parent palce, cin 3.
if you want to quit, cin 4.
3
now you have back to the parent place.
*****
Please set your binary tree:
if you want to give the left-child value, cin 1.
if you want to give the right-child vlaue,cin 2.
if you want to come back the parent palce, cin 3.
if you want to quit, cin 4.

```

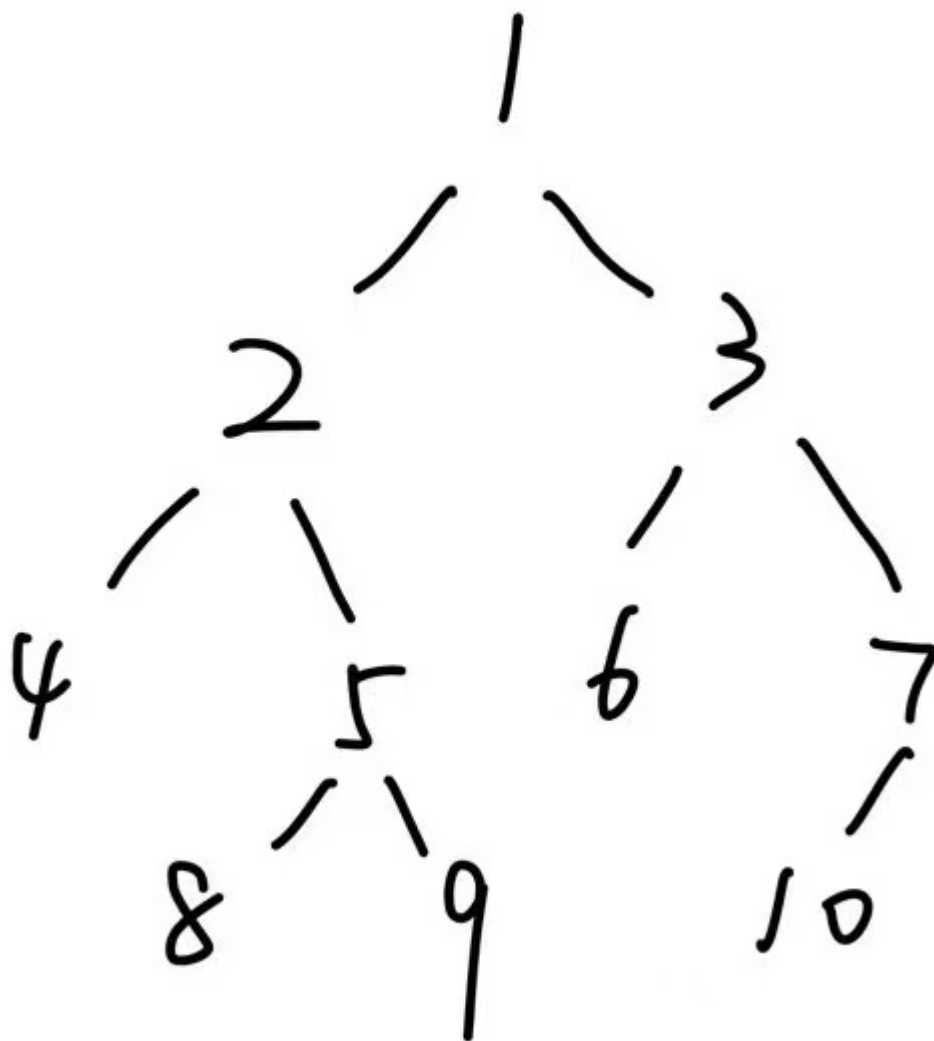
下面我将以图中的树为例，介绍各个功能的运行，输入数据：

```

1
1 2
1 4
3
2 5
1 8
3
2 9
3
3
3
2 3
1 6
3
2 7
1 10
4

```

其中1表示延伸左节点，后边的数据为Data；2表示右节点，3表示返回父节点，4表示退出二叉树编辑。以此我就得到了这样一颗二叉树：



我将以此树为样例演示操作。

先序遍历和中序遍历

先输入迭代版，再输出递归版。

```

student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.
1
this is the iteration model:
1 2 4 5 8 9 3 6 7 10
this is the recursion model:
1 2 4 5 8 9 3 6 7 10
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.
2
this is the iteration model:
4 2 8 5 9 1 6 3 10 7
this is the recursion model:
4 2 8 5 9 1 6 3 10 7
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.

```

后序遍历和层次遍历

```

student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if you want to quit, cin 7.
3
this is the iteration model:
4 8 9 5 2 6 10 7 3 1
this is the recursion model:
4 8 9 5 2 6 10 7 3 1
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.
4
the answer is:
1 2 3 4 5 6 7 8 9 10
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.

```

parent查找

以6为例，显然祖先为1和3.

```
student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.
5
you may need to choose the iterator's place:
the iterator's height is: 3 and the data is: 1
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
2
the iterator's height is: 2 and the data is: 3
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
1
the iterator's height is: 0 and the data is: 6
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
4
the answer is:
1 3
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the uncle data, cin 6.
if you want to quit, cin 7.
```

uncle查找

如果没有兄弟节点，返回null of this :

```
student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the sibiling data, cin 6.
if you want to quit, cin 7.
6
you may need to choose the iterator's place:
the iterator's height is: 3 and the data is: 1
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
2
the iterator's height is: 2 and the data is: 3
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
2
the iterator's height is: 1 and the data is: 7
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
1
the iterator's height is: 0 and the data is: 10
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
4
null of this
```

正常返回值 :


```
student@student-VirtualBox: ~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
4
null of this
if you want to Preorder traversal, cin 1.
if you want to Middle order traversal,cin 2.
if you want to Postorder traversal, cin 3.
if you want to level traversal, cin 4.
if you want to get the parent data, cin 5.
if you want to get the sibiling data, cin 6.
if you want to quit, cin 7.
6
you may need to choose the iterator's place:
the iterator's height is: 3 and the data is: 1
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
2
the iterator's height is: 2 and the data is: 3
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
1
the iterator's height is: 0 and the data is: 6
cin 1 means goto left child;2 means right one;3 means its parent;or 4 means quit.
4
the answer is:7
```

以上就是本次实验的全部内容，通过本次lab，我加深了对二叉树的理解，实验很有意义。