《数据结构》课程实践报告

院、系	计算机学院	年级专业	21 计算机科学与技术	姓名	赵鹏	学号	2127405037
实验布置日期	2022.10.25		提交日期	2022.11.23		成绩	

课程实践实验 8：二叉树

一、问题描述及要求

产生一个菜单驱动的演示程序，用以说明二叉树的使用。元素由单个键组成，键为单个字符。用户能演示的二叉树基本操作至少包括：构造二叉树，按先序、中序、后序、层序遍历这棵二叉树，求二叉树的深度、宽度，统计度为 0，1，2 的结点数等。

二叉树采用链式存储结构。

对二叉查找树做上述工作，且增加以下操作：插入、删除给定键的元素、查找目标键。

二、概要设计

1.问题分析

本次实验的内容是设计一个由菜单驱动的说明二叉树使用的演示程序。程序首先需要实现二叉树和二叉查找树的基本功能，并在其基础上设计与用户交互的菜单。

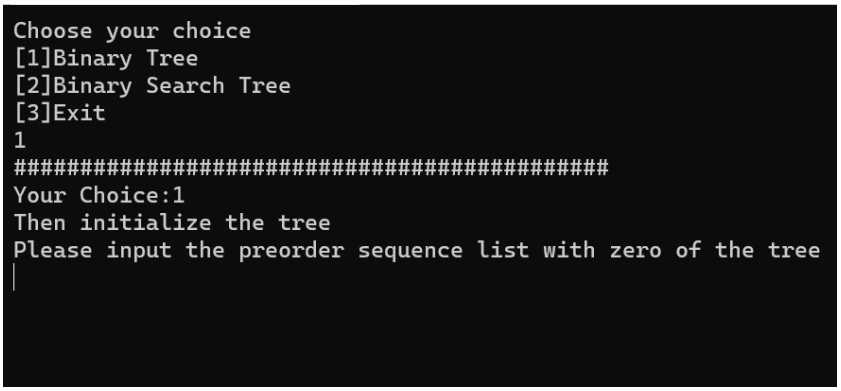
2.系统功能设计

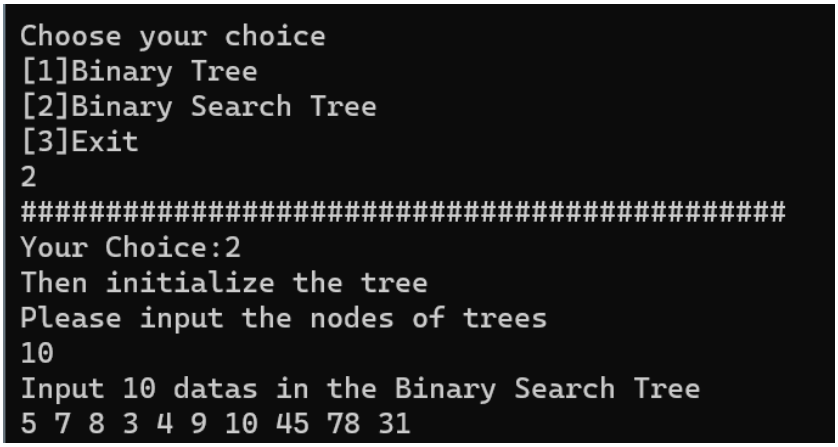
本次实验首先需要完成以下功能：

1. 用户选择二叉树/二叉查找树
2. 由用户创建二叉树
3. 按先序、中序、后序、层序遍历二叉树
4. 统计树上指定度数的节点数
5. 在二叉查找树中查找、插入、删除元素

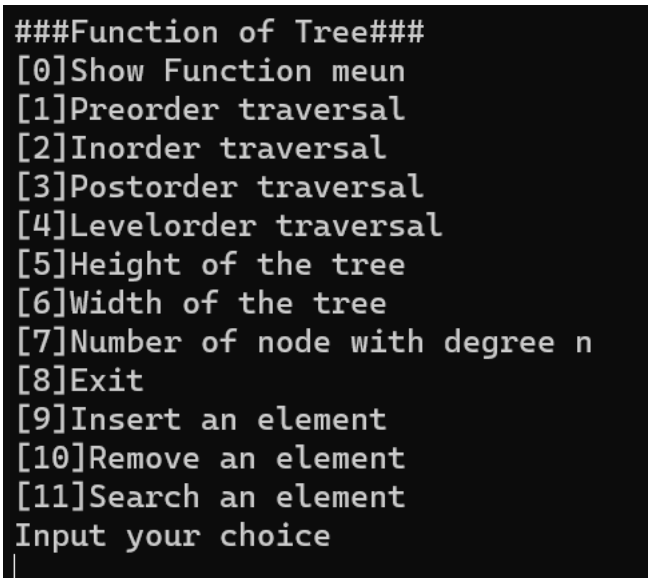
3.用户界面设计

首先由用户选择树的种类，如图所示：





随后显示功能菜单，如图所示：



其中功能 9-11 仅在用户选择生成二叉搜索树时显示。

在用户输入了合法选项编号后程序会调用二叉树/二叉搜索树的相应方法，输出相应信息，如下图所示：

```
Input your choice
1
The Preorder traversal of the tree is:
5 3 4 7 8 9 10 45 31 78
-------------------------------------------
Input your choice
2
The Inorder traversal of the tree is
3 4 5 7 8 9 10 31 45 78
-------------------------------------------
Input your choice
3
The Postorder traversal of the tree is:
4 3 31 78 45 10 9 8 7 5
-------------------------------------------
Input your choice
4
The Levelorder traversal of the tree is:
5 3 7 4 8 9 10 45 31 78
-------------------------------------------
Input your choice
5
The height of the tree is 9
-------------------------------------------
Input your choice
6
The width of the tree is 2
-------------------------------------------
Input your choice
7
Input the degree of nodes
1
Number of node with degree 1 is 5
-------------------------------------------
Input your choice
```

4. 程序结构设计

程序在设计过程中定义了二叉树结点类 BinaryNode、二叉树类 BinaryTree、二叉查找树类 BinarySearchTree，结点类供二叉树以及二叉查找树存储信息使用并将三个类均封装为模板类。由于二叉查找树是二叉树的一类，所以二叉查找树类可以直接由二叉树类派生而来。

三个类的具体方法如下：

BinaryNode	
Public	
方法	功能
BinaryNode()	默认构造函数
BinaryNode(DataType d)	构造函数

BinaryTree	
Public	
方法	功能
BinaryTree()	默认构造函数
~BinaryTree()	默认析构函数
void CreateTreeByPreOrder()	由特殊前序序列建树
void PreOrder()	前序遍历
void InOrder()	中序遍历
void PostOrder()	后序遍历
void LevelOrder()	层序遍历

int GepDepth()	获取树的深度
int GetWidth()	获取树的高度
int CalcDegree(int degree)	计算度数为 degree 的结点数
Protected	
方法	功能
void DestoryTree(BinaryNode<DataType>* rt)	递归销毁树
void PreOrderRecursion(BinaryNode<DataType>* rt)	前序遍历递归函数
void InOrderRecursion(BinaryNode<DataType>* rt)	中序遍历递归函数
void PostOrderRecursion(BinaryNode<DataType>* rt)	后序遍历递归函数
int GetDepthRecursion(BinaryNode<DataType>* rt)	计算深度递归函数
int CalcDegreeRecursion(BinaryNode<DataType>* rt, int degree)	计算特定度数结点数递归函数
BinaryNode<DataType>* Create()	建树函数

BinarySearchTree	
方法	功能
Public	
BinarySearchTree(DataType a[], int n)	构造函数
~BinarySearchTree()	析构函数
bool InsertBST(const DataType x)	插入元素
BinaryNode<DataType>*SearchBST (DataType x)	查找元素
bool Remove(const DataType& x)	删除元素
Private	
bool InsertRecursion (BinaryNode<DataType>*& bt, const DataType& x)	插入元素递归函数
BinaryNode<DataType>* SearchBST(BinaryNode<DataType>* bt, DataType x)	查找元素递归函数
bool RemoveRecursion(BinaryNode <DataType>*& rt, const DataType& x)	删除元素递归函数
void RemoveRoot(BinaryNode <DataType>*& rt)	删除元素辅助函数

对于菜单部分，采取了面向过程的设计方法。在主函数中设计与用户交互的过程。基本思路为首先由用户选择树的种类并引导用户输入信息对树进行初始化。随后采取循环的方式使用户不断输入操作，并调用对应树的相应方法。

三、详细设计

（一）、二叉树部分算法实现

1. 二叉树的创建

对于简单二叉树，程序采取使用带有特殊值的前序遍历序列进行创建，由用户输入指定格式的前序序列，创建二叉树。

2. 二叉树的遍历

对于二叉树的前、中、后序遍历，都可采取递归式遍历实现。具体如下：

前序遍历：①访问根结点；②递归访问左子树；③递归访问右子树

中序遍历：①递归访问左子树；②访问根结点；③递归访问右子树

后序遍历：①递归访问左子树；②递归访问右子树；③访问根结点

三种遍历方式的基本思路均相同，只是访问根结点的时间不同。由于递归的简洁性，实现也较为简单。

3. 二叉树的层序遍历

层序遍历的基本思路为按二叉树的层次关系由浅到深进行遍历。基本思路为由上到下，由左到右。为了保持访问结点的这种次序关系，可以采取队列来实现层次遍历。在实现过程中复用了链栈这一数据结构，代码如下：

```
1. Queue<BinaryNode<DataType>* >Q;
2. Q.push(root);
3. while (!Q.empty())
4. {
5. BinaryNode<DataType>* t = Q.front(); Q.pop();
6. std::cout << t->data << " ";
7. if (t->lchild != NULL)Q.push(t->lchild);
8. if (t->rchild != NULL)Q.push(t->rchild);
9. }
10.std::cout << std::endl;
```

4. 计算二叉树的深度

二叉树上某一节点的深度，都等于其左右子树的最大深度+1。因此在计算深度时同样可以采取递归的方式，递归访问每个点的左右子树(注意空树直接返回)并求出最大值即可。

代码实现如下：

```
return rt == NULL ? 0 :
1  + GetDepthRecursion(rt->lchild) + GetDepthRecursion( rt->rchild)
```

5. 求解二叉树的宽度

二叉树的宽度的定义是同一层的节点数目的最大值。因此可以采取与层序遍历类似的方法进行统计。但需要注意的是，若直接采取层序遍历的方法会出现问题：由于层序遍历是加入新节点和弹出旧节点交替进行，因此对于加入队列的点不能够知道其位于二叉树的某一层，因此也就无法统计同一层的结点数。为了解决这一问题，我们可以循环访问当前队列中同一层的节点，将其不为空的左右儿子直接加入队列中，而循环前当前队列的大小 size 也就是当前层的结点数。在当前层的子节点均入队过程中，弹出队列中前 size 个节点即可。

代码实现如下：

```
1.     Queue<BinaryNode<DataType>* >Q;
2.     int maxWidth = 0;
3.     if (root != NULL)Q.push(root);
4.     while (!Q.empty())
5.     {
6.         int levelWidth = Q.size();
7.         maxWidth = max(maxWidth, levelWidth);
8.         for (int i = 0; i < levelWidth; i++)
9.         {
10.            BinaryNode<DataType>* p = Q.front();
11.            if (p->lchild != NULL)Q.push(p->lchild);
12.            if (p->rchild != NULL)Q.push(p->rchild);
13.            Q.pop();
14.        }
15.    }
16.    return maxWidth;
```

6. 统计二叉树指定度数的节点数

当前二叉树中度数为 d 的节点数等于左右子树中节点数的和，如果当前根节点的度也等于 d 则额外加一即可。

代码实现如下：

```
1.     if (rt == NULL)return 0;
2.     int child = 0;
3.     if (rt->lchild != NULL)child++;
4.     if (rt->rchild != NULL)child++;
5.     return degree == child ? 1 + CalcDegreeRecursion(rt->lc
   hild, degree) + CalcDegreeRecursion(rt->rchild, degree)
6.         : CalcDegreeRecursion(rt->lchild, degree) + CalcDeg
   reeRecursion(rt->rchild, degree);
```

（二）、二叉查找树部分算法实现

1.在二叉查找树中查找与插入节点

在二叉查找树中插入与删除节点的大体思路基本相同。即从根节点开始访问，若根节点为空则查找失败，返回信息或者插入节点。否则判断 root->data 与 target 的关系。若 root->target 则查找成功。若 root->data>target 则递归访问左子树，否则递归访问右子树，若递归到子树中空指针即为插入位置，也代表查找失败。

2. 在二叉查找树中删除节点

与插入查找节点类似，要删除二叉查找树中的节点首先需要找到节点的位置。但与查找和插入节点总是在叶子节点不同，被删除的节点可能是叶子节点也可能是分支节点，当删除分支结点时，会破坏原有节点的关系，因此需要调整指针，使得节点在删除后仍为二叉查找树。

可以分为以下两种情况进行讨论：

（1） 被删除的节点的度数小于等于 1，将其替换为非空子树后删除即可。

（2） 删除节点的度为 2 时需要找到一个节点替换这个节点。由于替换后需要保持节点的值比左子树所有节点小，比右子树所有节点大。所以可以选择左子树中最右下的节点。所以需要先向左移动，再不断向右移动直至当前节点的 rchild 指针为空为止。然后交换该节点和待删除结点的数据域后删除待删除结点即可。

3.二叉查找树的创建

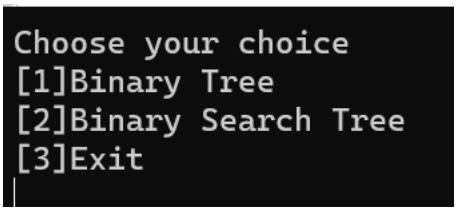
二叉查找树可以由序列直接创建。创建过程基于插入操作，遍历这个序列并且依次对其执行插入操作即可。

(三)、菜单的设计

菜单的设计总体采用面向过程的设计方法。

1. 树的选择与初始化

首先输出信息请求用户选择生成二叉树/二叉搜索树。随后根据用户输入的编号生成对应的树并进入下一阶段。



2. 树的操作

由于用户可以选择执行多次操作，故使用循环的方式请求用户不断输入操作。为了程序的简洁，可将功能列表和功能的执行封装为单独的函数。功能列表将根据根据树的

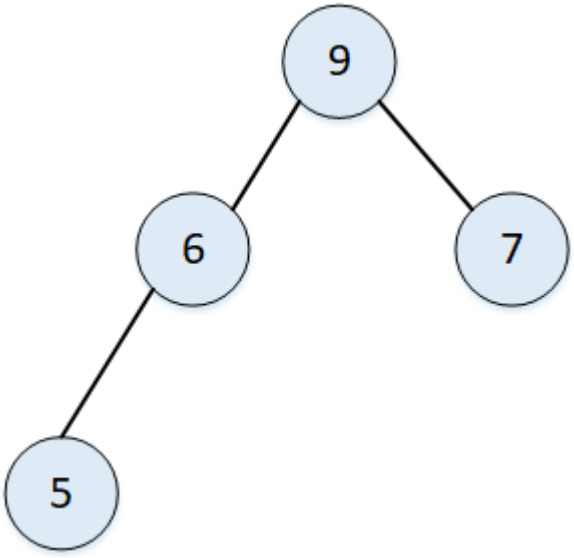
类型输出不同的功能。对于操作执行函数，在其内部采取 swtich case 结构，由根据用户输入的操作序号执行对应的操作。

```
Your Choice:2
Then initialize the tree
Please input the nodes of trees
3
Input 3 datas in the Binary Search Tree
1 2 3
###Function of Tree###
[0]Show Function meun
[1]Preorder traversal
[2]Inorder traversal
[3]Postorder traversal
[4]Levelorder traversal
[5]Height of the tree
[6]Width of the tree
[7]Number of node with degree n
[8]Exit
[9]Insert an element
[10]Remove an element
[11]Search an element
Input your choice
```

四、实验结果

(一)二叉树的测试与使用

运行程序输入 1 选择生成二叉树。随后输入特殊前序序列 9 6 5 0 0 0 7 0 0 生成如下图所示的二叉树。



```
Choose your choice
[1]Binary Tree
[2]Binary Search Tree
[3]Exit
1
###############################################
Your Choice:1
Then initialize the tree
Please input the preorder sequence list with zero of the tree
9 6 5 0 0 0 7 0 0
###Function of Tree###
[0]Show Function meun
[1]Preorder traversal
[2]Inorder traversal
[3]Postorder traversal
[4]Levelorder traversal
[5]Height of the tree
[6]Width of the tree
[7]Number of node with degree n
[8]Exit
Input your choice
```

1. 输入 1 查看前序遍历

```
Input your choice
1
The Preorder traversal of the tree is:
9 6 5 7
----------------------------------------
```

结论:通过测试

2. 输入 2 查看中序遍历

```
Input your choice
2
The Inorder traversal of the tree is
5 6 9 7
----------------------------------------
```

结论：通过测试

3. 输入 3 查看后序遍历

```
Input your choice
3
The Postorder traversal of the tree is:
5 6 7 9
```

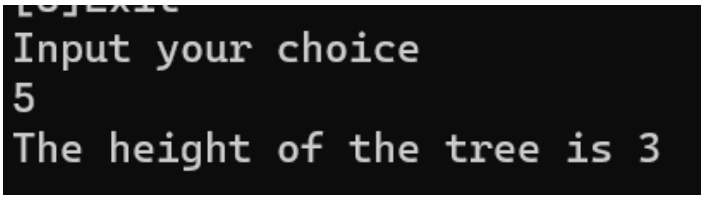
结论：通过测试

4. 输入 4 查看层序遍历

```
Input your choice
4
The Levelorder traversal of the tree is:
9 6 7 5
----------------------------------------
```

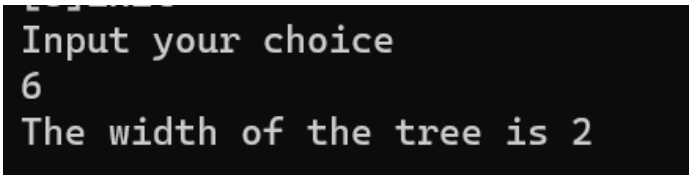
结论：通过测试

5. 输入 5 查看树的高度



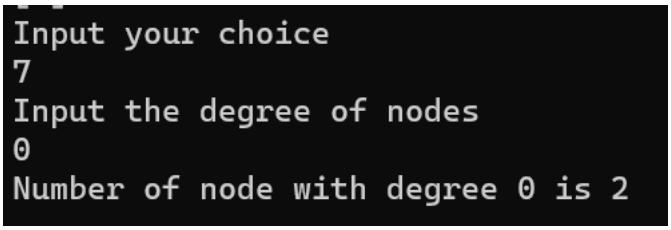
结论：通过测试

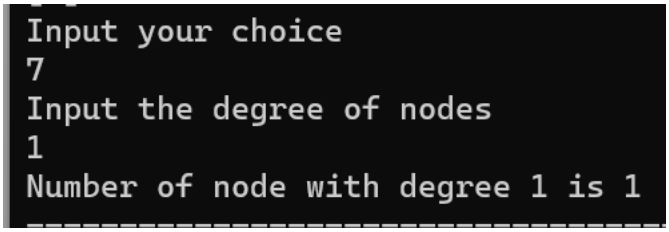
6. 输入 6 查看树的宽度

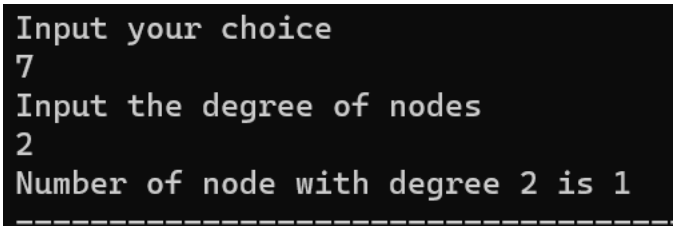


结论：通过测试

7. 输入 7 并分别输入 0，1，2 统计特定度数的结点数







结论：通过测试

(二)二叉搜索树的测试与使用

运行程序输入 2 选择生成二叉树。随后输入数据 10 12 48 35 7 5 33 64 85 99 100 生成一棵二叉搜索树。

1. 测试中序遍历是否为有序序列

输入 2 输出中序遍历序列

```
----------------------------------------
Input your choice
2
The Inorder traversal of the tree is
5 7 12 33 35 48 64 85 99 100
----------------------------------------
```

结论：测试通过

2. 测试删除结点

输入 10 12 删除数据为 12 的结点，通过中序遍历判断删除是否正确

```
Input your choice
10
Input the number to be removed in the tree
12
Remove completed!
----------------------------------------
Input your choice
2
The Inorder traversal of the tree is
5 7 33 35 48 64 85 99 100
```

结论：测试通过

3.测试查找结点

输入 11 10 测试查找失败的情况，输入 11 33 测试查找成功的情况

```
----------------------------------------
Input your choice
11 10
Input the number to be searched
The number does not exist in the tree
----------------------------------------
Input your choice
11 33
Input the number to be searched
The number exists in the tree
```

结论：测试通过

4.测试插入结点

输入 9 2 插入数值为 2 的结点，通过中序遍历判断插入是否正确

```
----------------------------------------
Input your choice
9
Input the number to be inserted
2
Insert completed!
----------------------------------------
Input your choice
2
The Inorder traversal of the tree is
2 5 7 33 35 48 64 85 99 100
----------------------------------------
```

结论：测试通过

五、小结

通过此次实验，我对二叉树与二叉搜索树的理解有了进一步的加深。对于树的遍历过程中的代码实现也使得我对递归算法的掌握有所增，树的层序遍历也加起了我对队列这一数据结构的应用能力。