**实 验 报 告**

**课程名称 数据结构（c）（英文）**

**实验项目** 树的建立和遍历

**实验仪器** 计算机

**系 别** 计算机学院学院

**专 业** 计算机大类

**班级/学号** 计算机类1801 2018011137

**学生姓名**  武文斌

**实验日期**  2019.11.10

**成 绩**

**指导教师**  蔡英

实验三

1. 实验目的
2. 学会使用二叉链表
3. 学会使用递归和非递归算法，进行前序遍历
4. Swap函数交换节点的左右指数
5. 实验内容

1） 生成一棵以二叉链表存储的二叉树bt（不少于15个结点）。

2） 分别用递归和非递归方法前序遍历bt，并以缩格形式打印bt上各结点的信息。

3） 编写算法，交换bt上所有结点的左、右子树，并以缩格形式打印出交换前后的bt结点信息。

1. 设计思路
2. 创建一棵二叉树：

创建一个树，每个树根以“#”做结，创建一个根节点，判断节点是否为空，若根节点为空，返回空；若根节点不为空输出根节点

2、利用递归算法前序遍历二叉树

3、利用非递归算法前序遍历二叉树：

（根左右）先访问根节点，若有根节点且堆栈不为空，把根节点压栈，首先把根节点pop出栈并打印，若同时有左右孩子，则把右孩子压入堆栈，将左孩子压入堆栈后pop出栈并打印，以此类推，直到前序遍历所有的节点。

1. 交换二叉树的左右子树：

直接用swap函数交换根节点的左右孩子

四、源代码：

#include <iostream>  
#include <stack>  
  
using namespace std;  
typedef struct Node{  
 char data;//结点里存的信息  
 struct Node\* lchild,\*rchild;//创建左右孩子  
}  
Tree;  
Tree\* createBinTree(){//创建一个二叉树  
 char ch;  
 Tree\* root;//树根  
 if ((ch = getchar()) != '#'){//以“#”做结  
 root = (Tree\*)malloc(sizeof(Tree));//创建一个节点  
 if (root == NULL){//判断根节点是否为空，若根节点为空，返回空；若根节点不为空输出根节点  
 return NULL;  
 }  
 root->data = ch;  
 root->lchild = createBinTree();  
 root->rchild = createBinTree();  
 }  
 else{  
 return NULL;  
 }  
 return root;  
}  
void createBinTree(Tree\* &t){  
 char ch;  
 if ((ch = getchar()) != '#'){  
 t = (Tree\*)malloc(sizeof(Tree));//创建一个节点  
 t->data = ch;  
 createBinTree(t->lchild);  
 createBinTree(t->rchild);  
 }  
 else{  
 t = NULL;  
 }  
}  
void preOrderTraverse(Tree\* &t){//前序遍历根左右,递归算法  
 if (t != NULL){  
 cout << t->data;  
 preOrderTraverse(t->lchild);//先访问根节点，再遍历左孩子  
 preOrderTraverse(t->rchild);//最后遍历右孩子  
 }  
}  
void preOrderTraverseNonRecur(Tree\* t){//非递归算法  
 stack<Tree\*> stack;  
 Tree\* p = t;  
 while (p != NULL || !stack.empty()){//有根节点且堆栈不为空，先将头节点压入堆栈  
 while (p != NULL){//有根节点时，先把根节点的左孩子压入堆栈  
 cout << p->data << " ";  
 stack.push(p);  
 p = p->lchild;  
 }  
 if (!stack.empty()){//当堆栈不为空时，先把栈顶孩子弹出，并打印，若栈顶孩子的右孩子不为空，再把右孩子压入堆栈  
 p = stack.top();//如果栈顶孩子的左孩子不为空，则将左孩子压入堆栈中  
 stack.pop();  
 p = p->rchild;  
 }  
 }  
}  
void ExchangeTree(Tree\* t)//交换二叉树的左右子树  
{  
 if (t == NULL)  
 {  
 return;  
 }  
 swap((t)->lchild, (t)->rchild);//直接用swap函数交换根节点的左右孩子  
 ExchangeTree(t->lchild);  
 ExchangeTree(t->rchild);  
}  
int main(){  
 Tree\* t = NULL;  
 createBinTree(t);  
 cout << "Preorder Traversal: ";  
 preOrderTraverseNonRecur(t);  
 cout << "\nExchanged Preorder Traversal: ";  
 ExchangeTree(t);  
 preOrderTraverseNonRecur(t);  
}

五、实验心得

在做了本次实验三树的建立和遍历后，充分的了解了用二叉链表存储树的过程和思想，并且明确了对于树的前序遍历的思想，以及根节点子树的交换。

1. 进一步的改进和设想

后续会添加中序和后续的遍历算法。