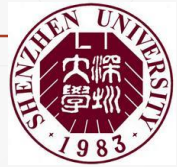
实验四
二叉树遍历实验

一、实验目的

- 掌握二叉树的“先根”遍历存储表示原理
- 掌握二叉树“先根”遍历存储表示，转换为二叉链表表示原理和方法
- 掌握采用二叉链表表示的二叉树的先根遍历、中根遍历、后根遍历实现方法

二、实验要求

- 熟悉C++语言编程
- 掌握二叉树遍历原理



三、实验内容

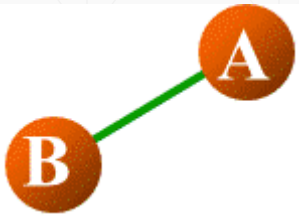
1、问题描述

- 给定一棵按“先根”遍历存储表示的二叉树，请先根遍历、中根遍历、后根遍历这棵二叉树。

三、实验内容

2、算法

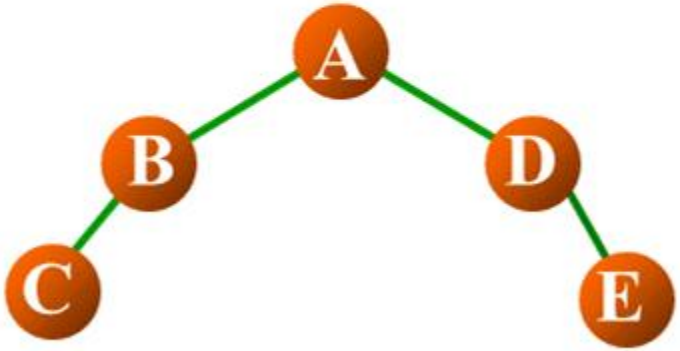
- 按“先根”遍历存储表示的二叉树中，如果在遍历过程中，发现子树为空，输出0
- 例如：A有左子树B，没有右子树，其“先根”遍历存储表示为A B 0 0 0

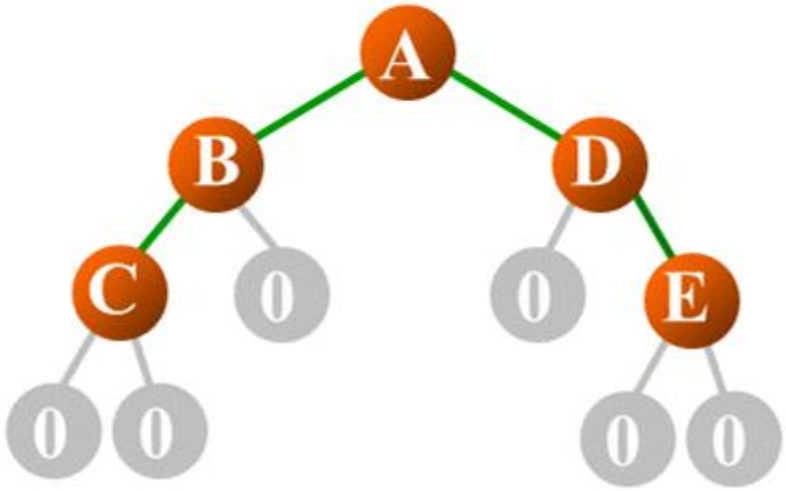


三、实验内容

2、算法

- 例如：A为根，B为A的左孩子；D为A的右孩子，C为B的左孩子，其“先根”遍历存储表示为
  A B C 0 0 0 D 0 E 0 0





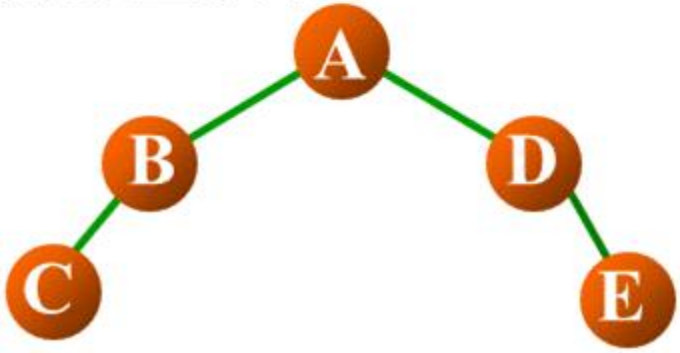
三、实验内容

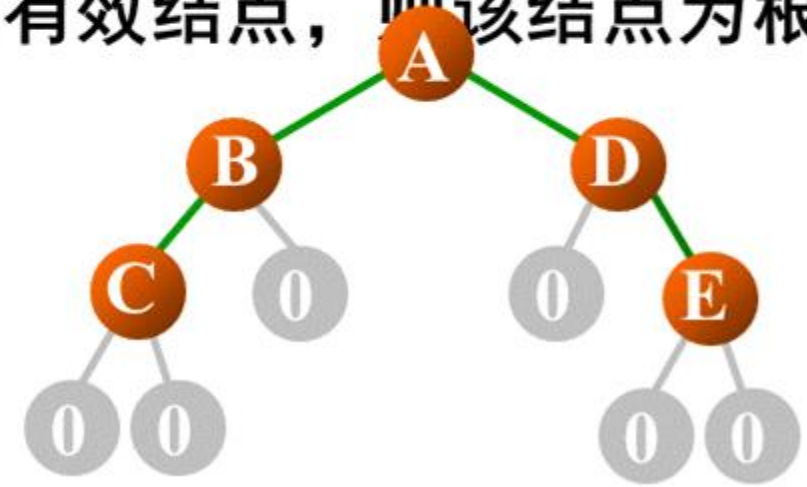
2、算法[“先根”存储遍历表示到二叉链表表示的转换]

- “先根”遍历存储表示A B C 0 0 0 D 0 E 0 0中：

1、根结点后面结点如果为空，则没有左子树；如果再后面结点有效，则为根的右孩子；否则根也是叶子（即后跟2个空结点）

2、根结点后，如果结点为有效结点，则该结点为根结点的左孩子





三、实验内容

3、输入

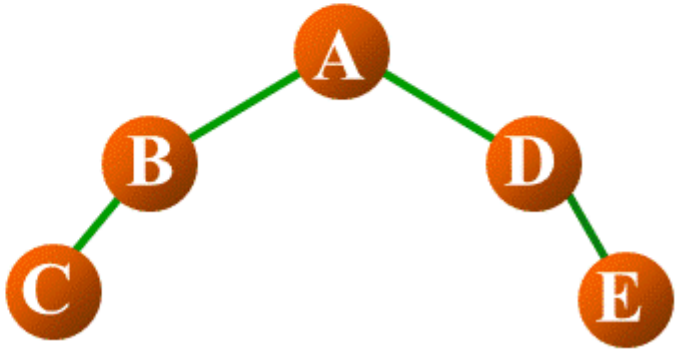
- 第一行：先根遍历的二叉树结点数目(包括空结点)
- 第二行：n个“先根”遍历存储表示的二叉树字符序列(用空格隔开)(字符0表示空结点)

三、实验内容

4、输入样本

11

A B C 0 0 0 D 0 E 0 0



三、实验内容

5、输出

- 第1行：二叉树的先根遍历序列
- 第2行：二叉树的中根遍历序列
- 第3行：二叉树的后根遍历序列

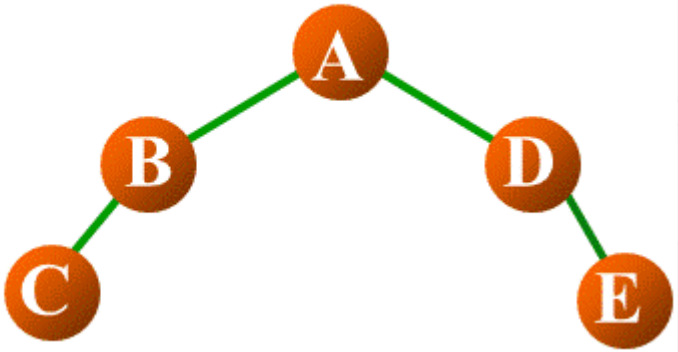
三、实验内容

6、输出样本

A B C D E

C B A D E

C B E D A



四、实验步骤

1、二叉链表定义

2、“先根”遍历表示，到二叉链表表示的转换

3、先根遍历函数

4、中根遍历函数

5、后根遍历函数

6、主程序

四、实验步骤

1、二叉链表定义

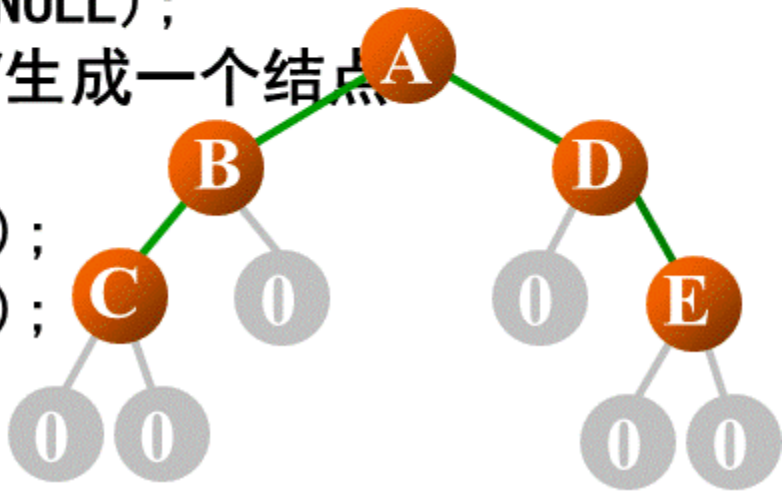
```
struct BiNode{
    char      data;
    BiNode  *lchild, *rchild;
};                                  // 定义二叉树的结点
BiNode *BiTree;
```

四、实验步骤

2、“先根”遍历表示到二叉链表表示的转换[函数]

```
int NodeID;          //全局变量，表示“先根”遍历存储表示序列序号
BiNode *CreateBiTree(char *c, int n)
{ BiNode *T;
  NodeID++;
  if (NodeID > n) return(NULL);
  if (c[NodeID] == '0') return(NULL);
  T = new BiNode;                    //生成一个结点
  T->data = c[NodeID];
  T->lchild = CreateBiTree(c, n);
  T->rchild = CreateBiTree(c, n);
  return(T);
}
```

1	2	3	4	5	6	7	8	9	10	11
11	A	B	C	0	0	0	D	0	E	0	0



四、实验步骤

3、先根遍历算法

```
void PreOrderTraverse(BiNode *T)
{
    :
}
```

四、实验步骤

4、中根遍历算法

```
void InOrderTraverse(BiNode *T)
{
   :
}
```

四、实验步骤

5、后根遍历算法

```
void PostOrderTraverse(BiNode *T)
{
   :
}
```

四、实验步骤

6、主函数

```
int main()
{  int i, SampleNum;
   char c[100];
   cin >> SampleNum;                                    //输入二叉树结点样本数目
   for (i=1; i<=SampleNum; i++) cin >> c[i];//输入二叉树结点数据
   NodeID = 0;
   BiTree = CreateBiTree(c, SampleNum);       //创建二叉树
   PreOrderTraverse(BiTree); cout << endl;   //先根遍历二叉树
   InOrderTraverse(BiTree);  cout << endl;   //中根遍历二叉树
   PostOrderTraverse(BiTree);cout << endl;   //后根遍历二叉树
   return 0;
}
```