```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 树的次树：最大MAXM次，树最多有MAXN个结点
#define MAXN 100
#define MAXM 10

typedef struct nodeParent{
    char data;
    int parent;
}NODE_PARENT;

typedef struct linkNode {
    char data;
    struct linkNode *child[MAXM];
    struct linkNode *parent;
    //根据应用的需要添加字段，例如可以添加结点的层次，子结点的个数等。
}LINK_NODE;

LINK_NODE *addr_NODE[MAXN];

int AddChild(LINK_NODE *parent,LINK_NODE *child,int m)
//m次树中，parent下插入一个新的子结child
{
    //int j=-1;
    //if (parent==NULL || child== NULL) return -1;
    //while (parent->child[++j]!=NULL && j<m);
    //……
```

```
}

LINK_NODE *Node_initail(int m)
{
    //m次树结点初始化，根据结点类型，完成结点的初始化操作
}


LINK_NODE *creatTreeFromParent(NODE_PARENT inputTree[],int m,int n)
//用一个数组addr_NODE来记录每个结点的指针
{
}
//分别 加入 前序，后序，层次遍历的函数，  前序和后续分别有递归实现和非递归实现
void r_preorder(LINK_NODE *t, int m) //递归前序遍历，t:根，m次数
{
}
void r_postorder(LINK_NODE *t, int m)
{
}
void hierachicalorder(LINK_NODE *t, int m)
{
}
void preorderNoRecursion(LINK_NODE *t, int m)
{
}
void postorderNoRecursion(LINK_NODE *t, int m)
{
}
int main()
```

```
{
    LINK_NODE *root;
    NODE_PARENT inputTree[MAXN];

//将输入数据读入到inputTree数组，建树，分别前序，后序，层次输出。（包括递归和非递归）、
    //root = creat_tree_fromParent(inputTree,3,10);
    //r_preorder(root,3);……
return 1;
}
```