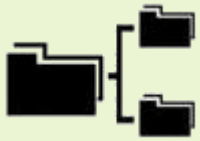
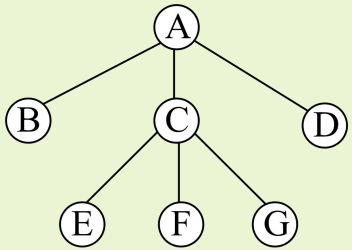


本节主题：

树的存储结构

双亲存储结构

- 做法
  - 一种顺序存储结构，用一组连续空间存储树的所有节点，
  - 同时在每个节点中附设一个伪指针指示其双亲节点的位置

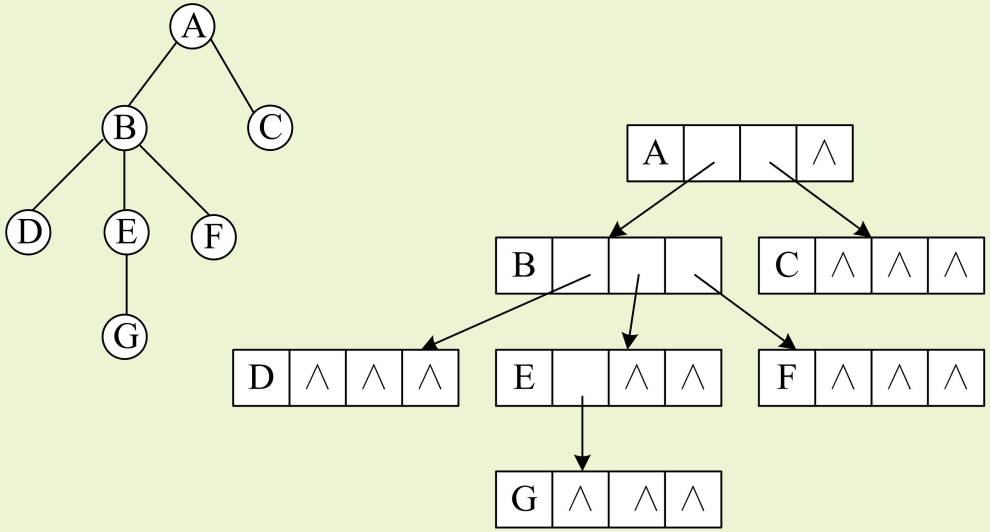


[0]	A	-1
[1]	B	0
[2]	C	0
[3]	D	0
[4]	E	2
[5]	F	2
[6]	G	2
[7]	...	

```
typedef struct
{
    ElemType data;
    int parent;
} PTree[MaxSize];
```

孩子链存储结构

- 方法
  - 存储每个节点的值，以及所有孩子的链接
  - 按树的度（即树中所有节点度的最大值）设计节点的孩子节点指针域个数。

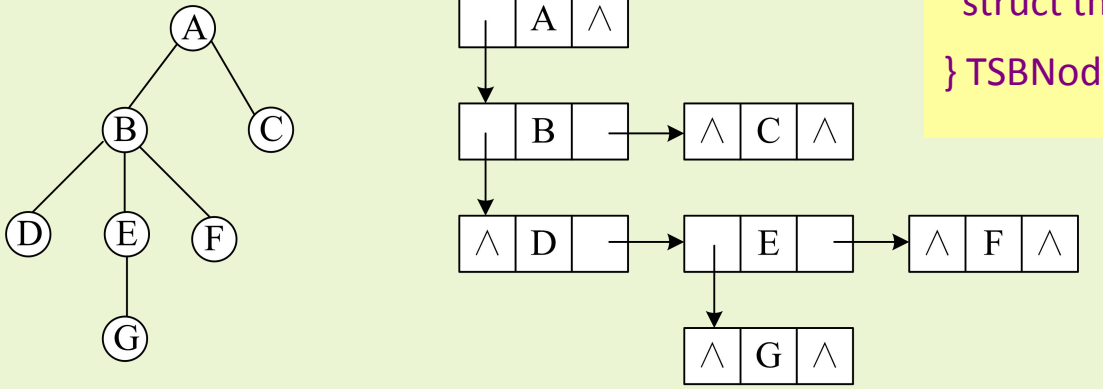



```
typedef struct node
{
   ElemType data;
   struct node *sons[MaxSons];
} TSonNode;
//MaxSons为最多的孩子节点个数
```

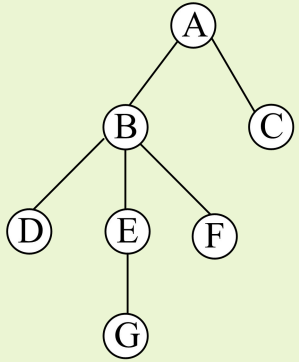
孩子兄弟链存储结构

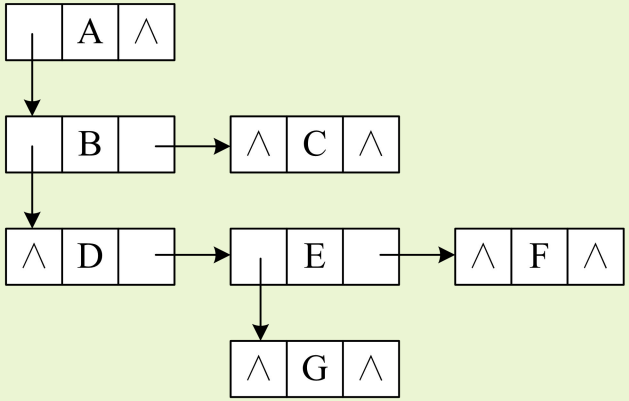
- 孩子兄弟链存储结构是为每个节点设计三个域
  - 一个数据元素域
  - 一个该节点的第一个孩子节点指针域
  - 一个该节点的下一个兄弟节点指针域。

```
typedef struct tnode
{
  ElemType data;        //节点的值
  struct tnode *hp;     //指向兄弟
  struct tnode *vp;     //指向孩子节点
} TSBNode;
```



例：以孩子-兄弟链作为存储结构，求树的高度





- 设f(t)为树t的高度，有

$$f(t)=\begin{cases}0 & ,\ t\text{为NULL}\\ 1 & ,\ t\text{没有孩子节点}\\ \max\limits_{p\text{为}t\text{的孩子}}\{f(p)\}+1 & ,\ \text{其他孩子}\end{cases}$$

```
p=t->vp;
while(p!=NULL)
{
   m=TreeHeight(p);
   if(max<m)
      max=m;
   p=p->hp;
}
```

```
int TreeHeight(TSNode *t)
{
   TSNode *p;
   int m, max = 0;
   if(t==NULL)
      return(0);
   else if(t->vp==NULL)
      return(1);
   else
   {
      //求t的子树的最大高度max
      return(max+1);
   }
}
```

关于实现的讨论

```
typedef struct tnode
{
  ElemType data;        //节点的值
  struct tnode *hp;     //指向兄弟
  struct tnode *vp;     //指向孩子节点
} TSBNode;
int TreeHeight(TSNode *t);
void TreeCreate(TSNode *&t);
void TreeDisp(TSNode *t);
```

```
int main()
{
    TSNode *tree;
    TreeCreate(tree);
    TreeDisp(tree);
    printf("Height: %d\n", TreeHeight(tree));
    return 0;
}
```