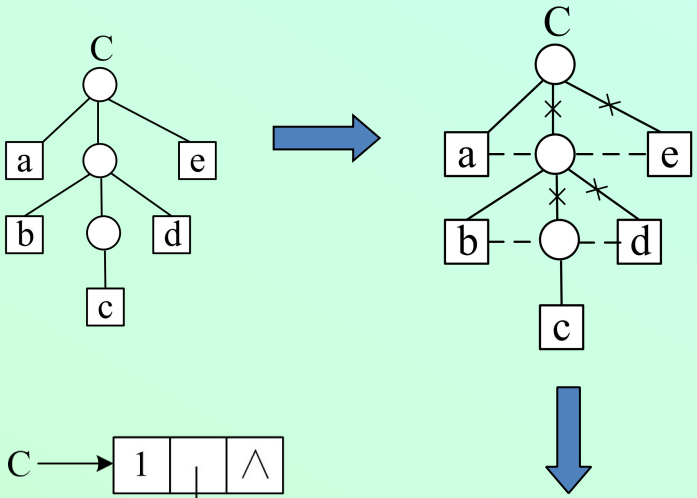
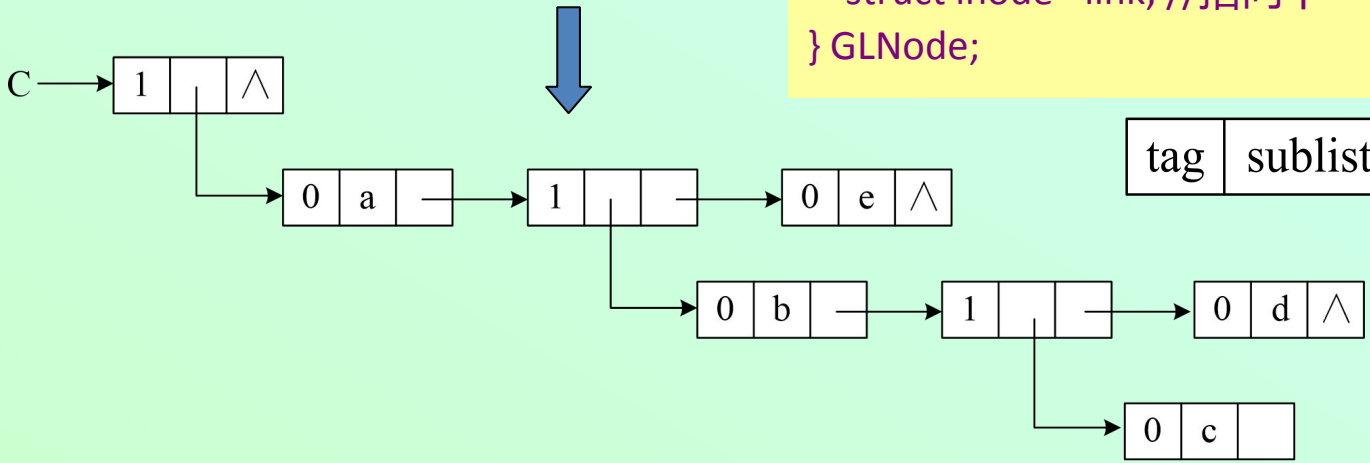
本节主题：

广义表的存储结构及基本运算的实现

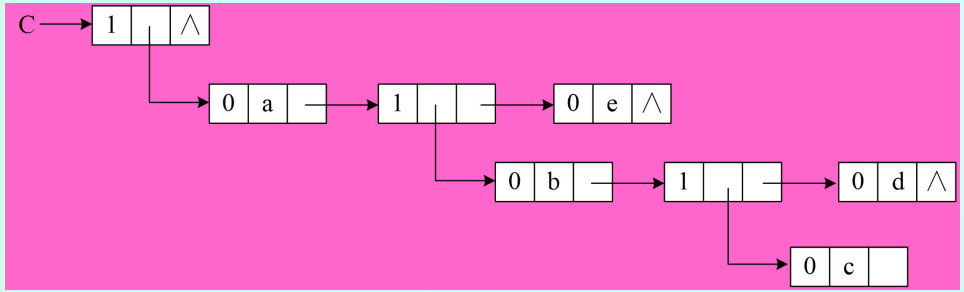
广义表的存储结构




```
typedef struct lnode
{
   int tag;   //节点类型标识：1 - 子表，0 - 原子
   union
   {
      ElemType data;
      struct lnode *sublist;
   } val;
   struct lnode *link; //指向下一个元素
} GLNode;
```

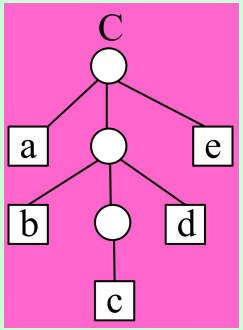


广义表链式存储实例



空表 g1 → 1 ∧ ∧

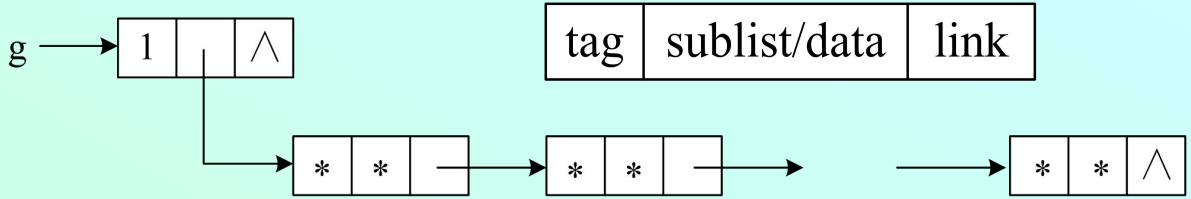
非空表 g2 → 1 ∧ → * * → * * → * * ∧



子表节点 1 * *

a 原子节点 0 a *

求广义表的长度



- 原理
  - 在广义表中，同一层次的每个节点是通过link域链接起来的，所以可把它看做是由link域链接起来的单链表
- 算法
  - 求广义表的长度就是第一层单链表的长度，求单链表长度得到广义表长度

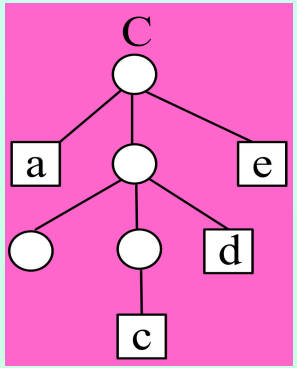
```
typedef struct lnode
{
   int tag;   //1 - 子表，0 - 原子
   union
   {
      ElemType data;
      struct lnode *sublist;
   } val;
   struct lnode *link; //指向下一个元素
} GLNode;
```

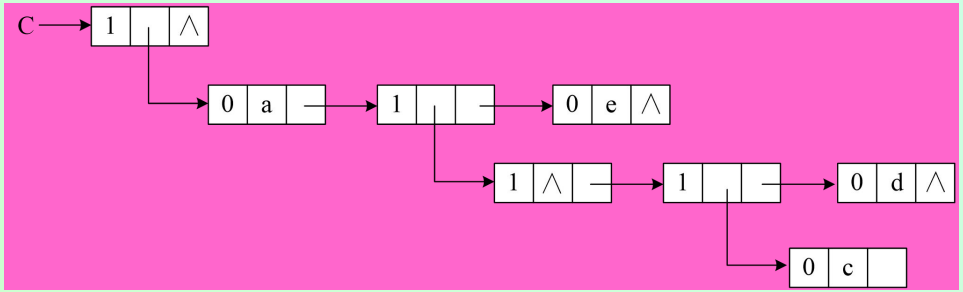
```
int GLLength(GLNode *g)
{
   int n=0;
   g=g->val.sublist;
   while (g!=NULL)
   {
      n++;
      g=g->link;
   }
   return n;
}
```

求广义表的深度

- 原理
  - 对于带头节点的广义表g，其深度等于所有子表中的最大深度加1；若g为原子，其深度为0；g为空表，其深度为1。
- 广义表深度f(g)的递归定义

$$f(g)=\begin{cases}0 & g\text{为原子}\\ 1 & g\text{为空表}\\ \max\{f(subg)\}+1 & g\text{有子表}\end{cases}$$

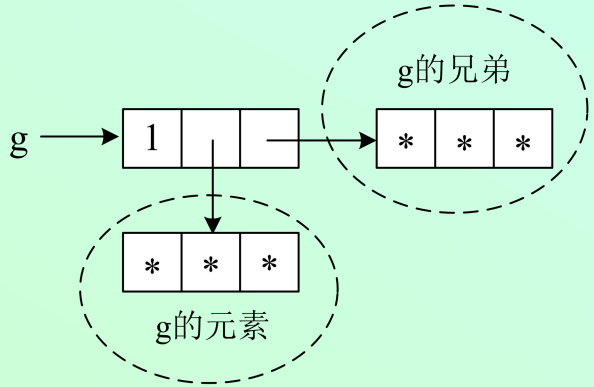




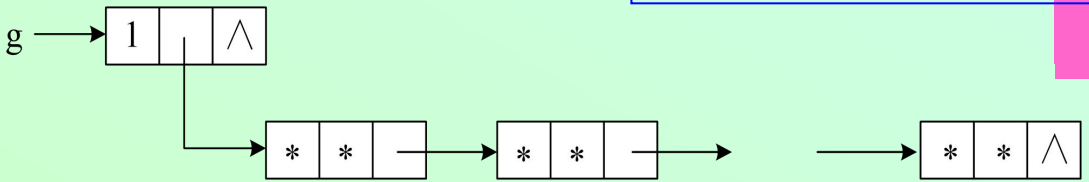

```
int GLDepth(GLNode *g)
{
   int max=0,dep;
   if (g->tag==0) return 0;
   g=g->val.sublist;
   if  (g==NULL) return  1;
   while (g!=NULL)
   {
      if (g->tag==1)
      {
         dep=GLDepth(g);
         if (dep>max)
            max=dep;
      }
      g=g->link;
   }
   return(max+1);
}
```

输出广义表

- 方案
  - 对子表递归式输出



(a,(b,c,d),(#))



tag	sublist/data	link

```
void DispGL(GLNode *g)
{
   if (g!=NULL)
   {
      //先处理g的元素
      //再处理g的兄弟
   }
}
```

```
if (g->tag==0)
   printf("%c", g->val.data);
ese
{
   printf("(");
   if (g->val.sublist==NULL)
      printf("#");
   else
      DispGL(g->val.sublist);
   printf(")");
}
```

```
if (g->link!=NULL)
{
   printf(",");
   DispGL(g->link);
}
```

建立广义表的链式存储结构

- 输入
  - 由括号表示法表示的s的广义表
  - 原子值为字符
  - 空表为#
- 返回
  - 指向链式结构的指针
- 输入样例
  - (a,(b,c,d),(#))

```
GLNode *CreateGL(char *&s)
{
   GLNode *g;
   char ch=*s++;
   if (ch!='\0')
   {
      g=(GLNode *)malloc(sizeof(GLNode));
      if (ch=='(')
      {   g->tag=1;
          g->val.sublist=CreateGL(s);   新节点作表头节点
      }
      else if (ch==')')
         g=NULL;  //遇到')'字符，子表结束，g置为空
      else if (ch=='#')
          g=NULL;  //遇到'#'字符，为空表，g置为空
      else
      {   g->tag=0;
          g->val.data=ch;   新节点作为原子节点
      }
   }
   else
      g=NULL;  //串结束,g置为空
   //继续处理后续字符
   return g;
}
```

```
ch=*s++;
if (g!=NULL)
   if (ch==',')
      g->link=CreateGL(s);
   else
      g->link=NULL;
```

思考题

- 多项式运算：是否可以采用广义表的方法存放多项式，如何实现相关算法？