DATA STRUCTUE

GENERALIZED LIST

广义表的定义

- 广义表（简称表）是线性表的推广，是有限个元素的序列，其逻辑结构采用括号表示法表示如下：

$$GL=(a_1,\ a_2,\ \ldots,\ a_i,\ \ldots,\ a_n)$$

广义表的基本概念

$$GL=(a_1,\ a_2,\ \ldots,\ a_i,\ \ldots,\ a_n)$$

- 若n=0时称为空表
- a_i为广义表的第i个元素。
- 如果a_i属于原子类型，称之为广义表GL的原子(atom)
- 如果a_i又是一个广义表，称之为广义表GL的子表

广义表的特性

1. 广义表中数据元素是有**相对次序**的
2. 广义表的长度：最外层包含元素的个数
3. 广义表的深度：所含括弧重数；原子深度为0，空表深度为1
4. 表头：首元素a_1
5. 表尾：$(a_2, \ldots, a_n)$其也是一个广义表
6. 广义表可以共享，一个广义表可被其他广义表共享，共享广义表又叫“再入表”
7. 广义表可以是一个递归的表，可以使自己的字表。称“递归表”。depth = ∞ ; length = C

广义表的存储结构

A=()

B=(e)

C=(a, (b, c, d))

D=(A, B, C)

E=((a, (a, b), ((a, b), c)))

A=()

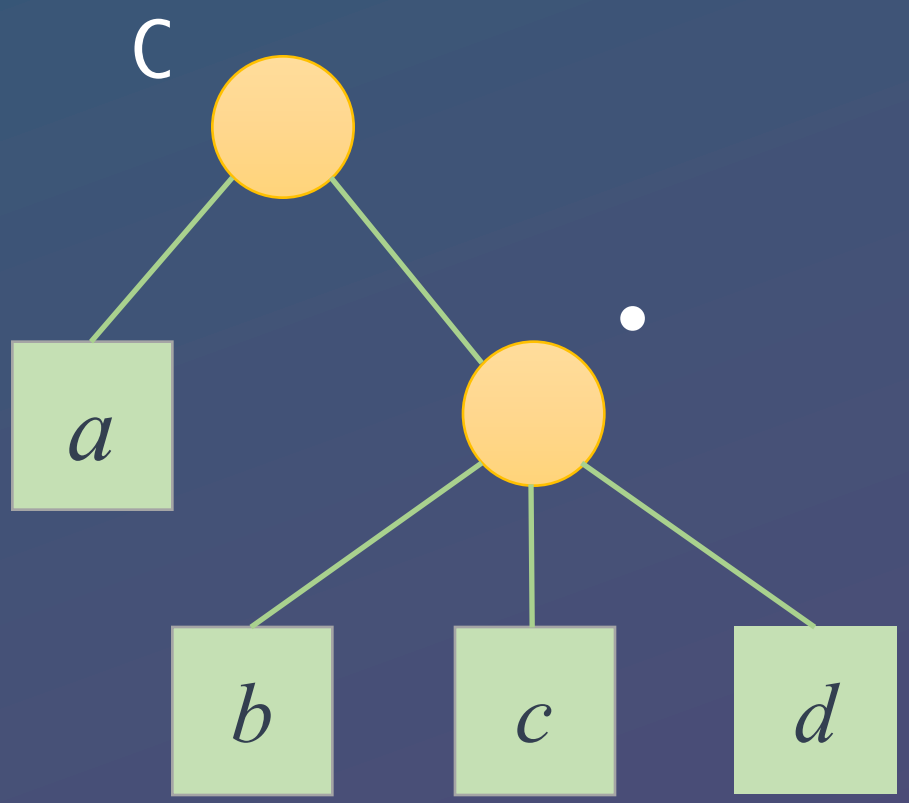
B=(e)

C=(a, ·(b, c, d))

D=(A(), B(e), C(a, ·(b, c, d))))

E=(·(a, ·(a, b), ·(·(a, b), c)))

广义表的图形表示

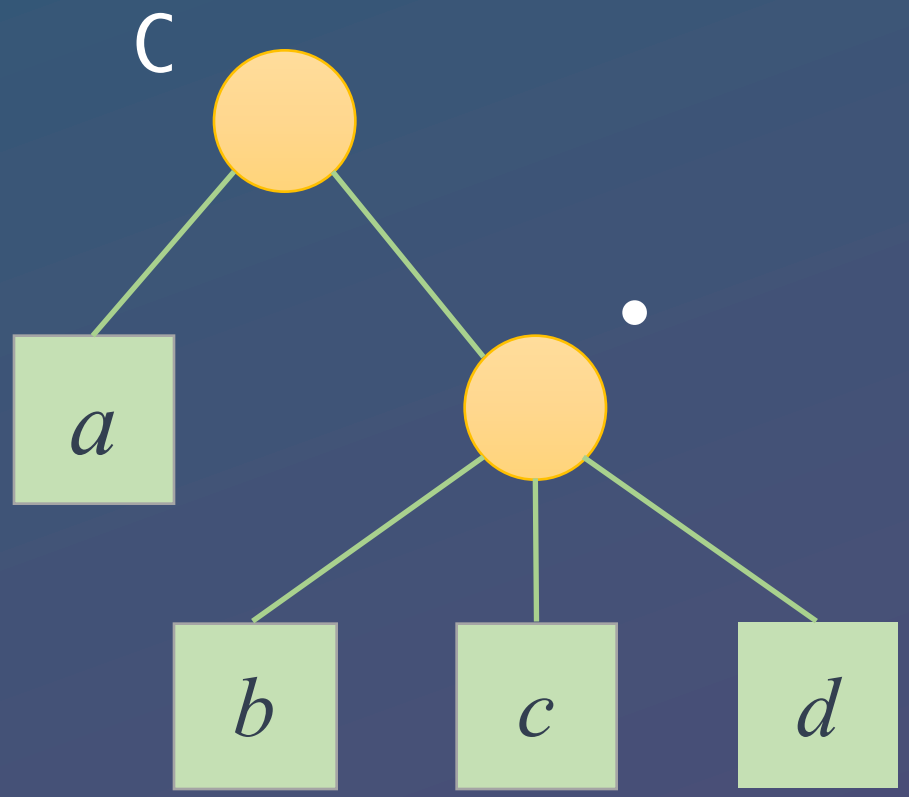


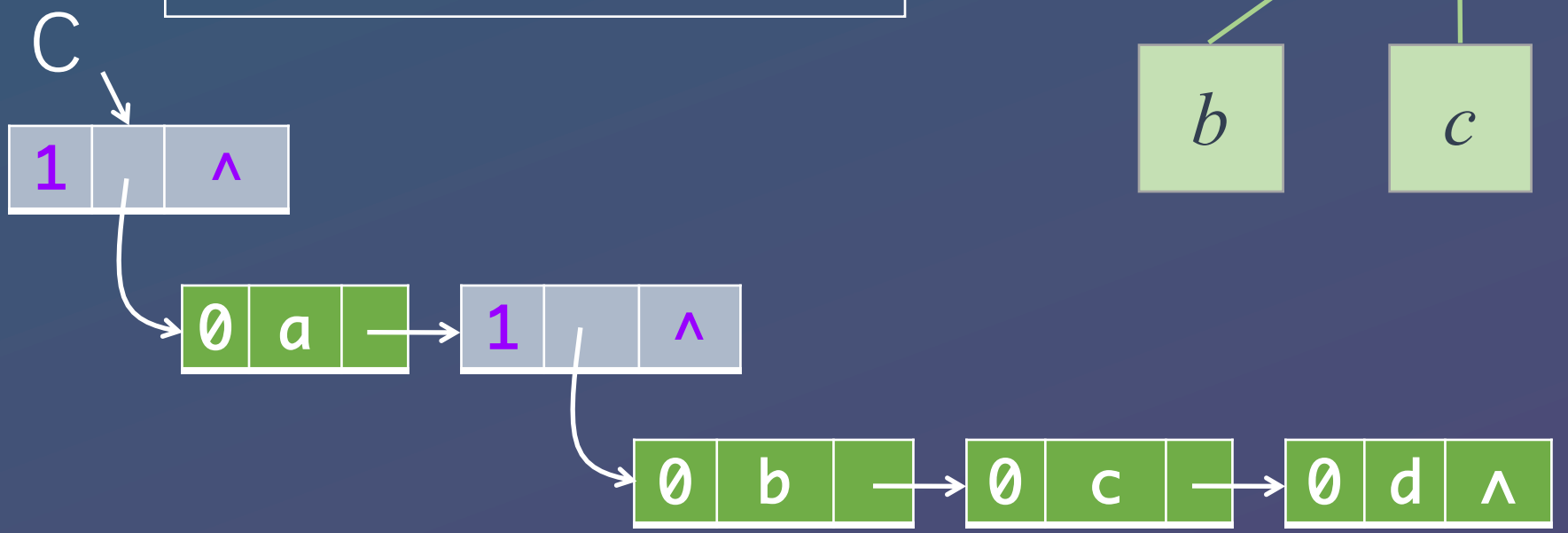
广义表的存储

tag	sublist/data	link

若tag=0，表示该结点为原子结点

若tag=1，表示该结点为表/子表结点

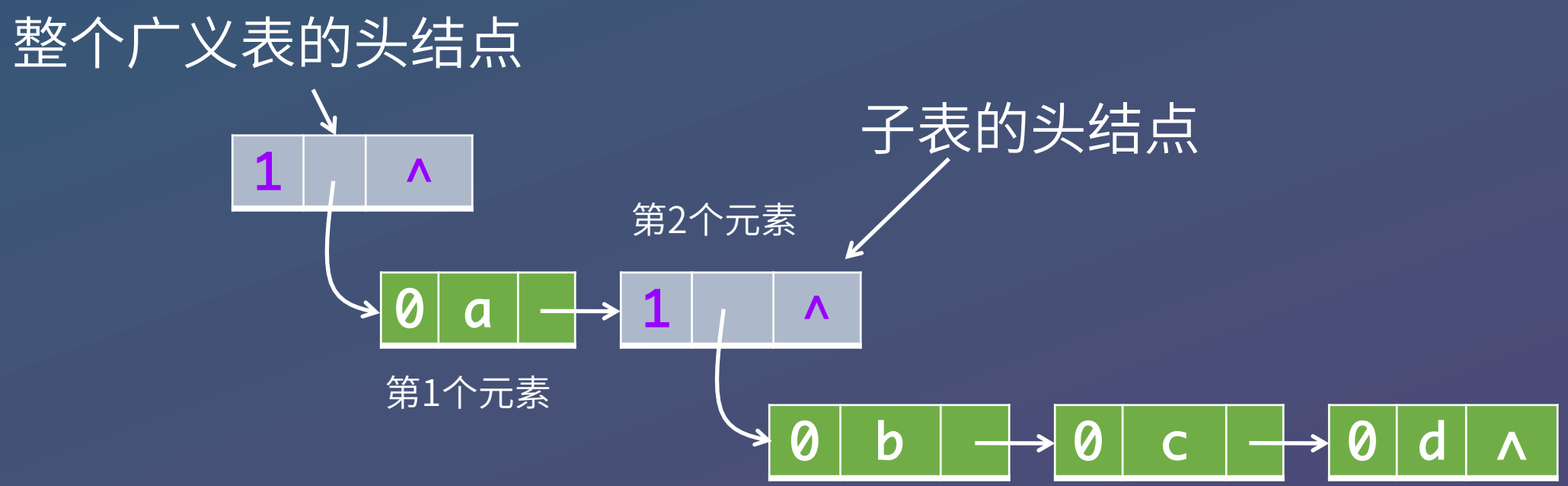




广义表的节点定义

```
typedef struct lnode  {
    int tag;            //结点类型标识
    union{
        ElemType data;       //存放原子值
        struct lnode *sublist;  //指向子表的指针
    } val;
    struct lnode *link;      //指向下一个元素
}  GLNode;
```

广义表算法设计方法 1



广义表的处理算法 1

思路：

1. 循环 + 递归
2. 以递归的方式处理子表。子表和整个广义表处理是相似的。
3. 处理方式类似树的 **深度优先搜索（DFS）每当遇到子表时，就先将剩余未处理的兄弟放入系统栈，等处理完子表再回溯。**

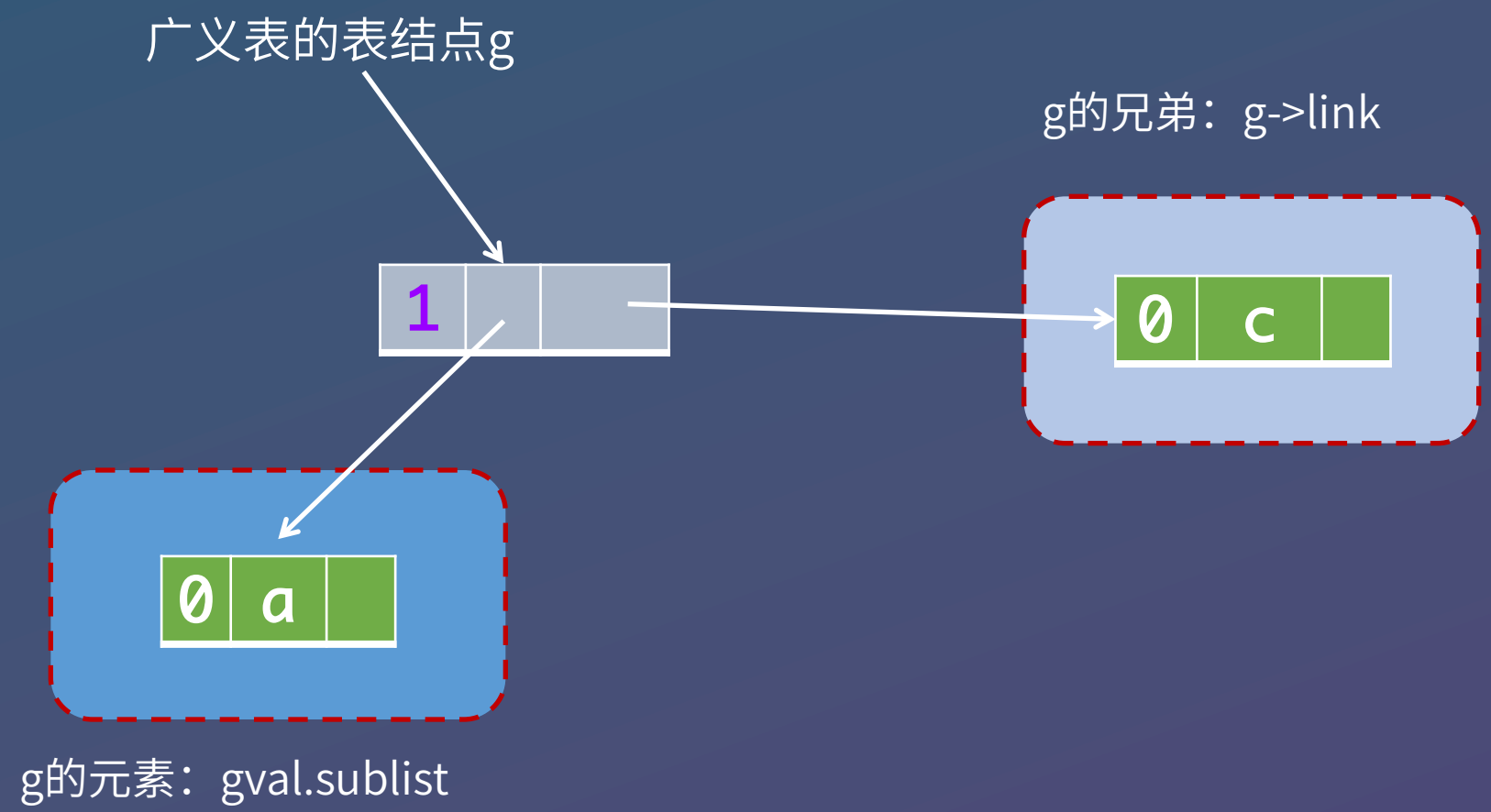
广义表的处理算法 1

C++ 广义表算法1.cpp

```cpp
void fun1(GLNode *g) {                  //g为广义表头结点指针
    GLNode *g1 = g->val.sublist;        //g1指向第一个元素
    while (g1 != NULL) {                //元素未处理完循环
        if (g1->tag==1)                 //为子表时
        fun1(g1);                       //递归处理子表
        else                            //为原子时
            原子处理语句;                //实现原子操作
        g1=g1->link;                    //处理兄弟
    }
}

```

广义表算法设计方法 2



广义表的处理算法 2

思路：

1. 递归
2. 以递归的方式处理子表和兄弟。子表/兄弟和整个广义表处理是相似的。
3. 处理方式类似树的 **深度优先搜索（DFS）每当遇到子表时，就先将剩余未处理的节点放入系统的栈，等处理完子表再回溯。**

广义表的处理算法 2

C++ 广义表算法2.cpp

```cpp
void fun2(GLNode *g) {                    //g为广义表结点指针
    if (g!=NULL) {
        if (g->tag==1)                    //为子表时
            fun2(g->val.sublist);         //递归处理其元素
        else                              //为原子时
            原子处理语句;                  //实现原子操作
        fun2(g->link);                    //递归处理其兄弟
    }
}

```

THANKS FOR YOUR ATTENTION!

You can download the pdf document from

https://github.com/JetRunner/Generalized-List