2.3线性表的链式存储结构（续）

DS/SS

链表---线性表的链式存储

内涵：

线性表的链式存储指用任意的存储单元存放线性表中的元素，每个元素与其前驱和（或）后继之间的关系用指针来存储。这称为链表。

术语：

* 结点
* 数据域
* 指针域
* 头指针
* 头结点

分类：

* 单链表
* 循环链表
* 双向链表
* 双向循环链表
* 静态链表

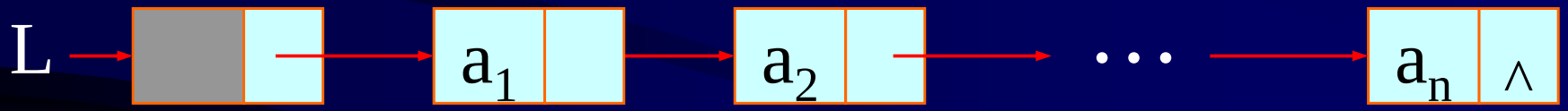
2.3线性表的链式存储结构（续）

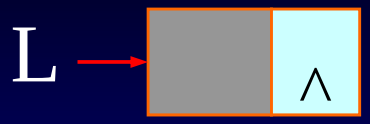
DS/SS

单链表

链表中，如果每个结点中只包含一个指针域，称这种链表为线性链表或单链表。

单链表可由头指针唯一确定。





2.3线性表的链式存储结构（续）

DS/SS

单链表的数据类型描述

用高级语言中的指针类型描述线性表的链式存储

```
//----------用结构指针描述-----------
typedef  struct  LNode{
        ElemType        data      //数据域
        struct LNode    *next     //指针域
} LNode, *LinkList
```

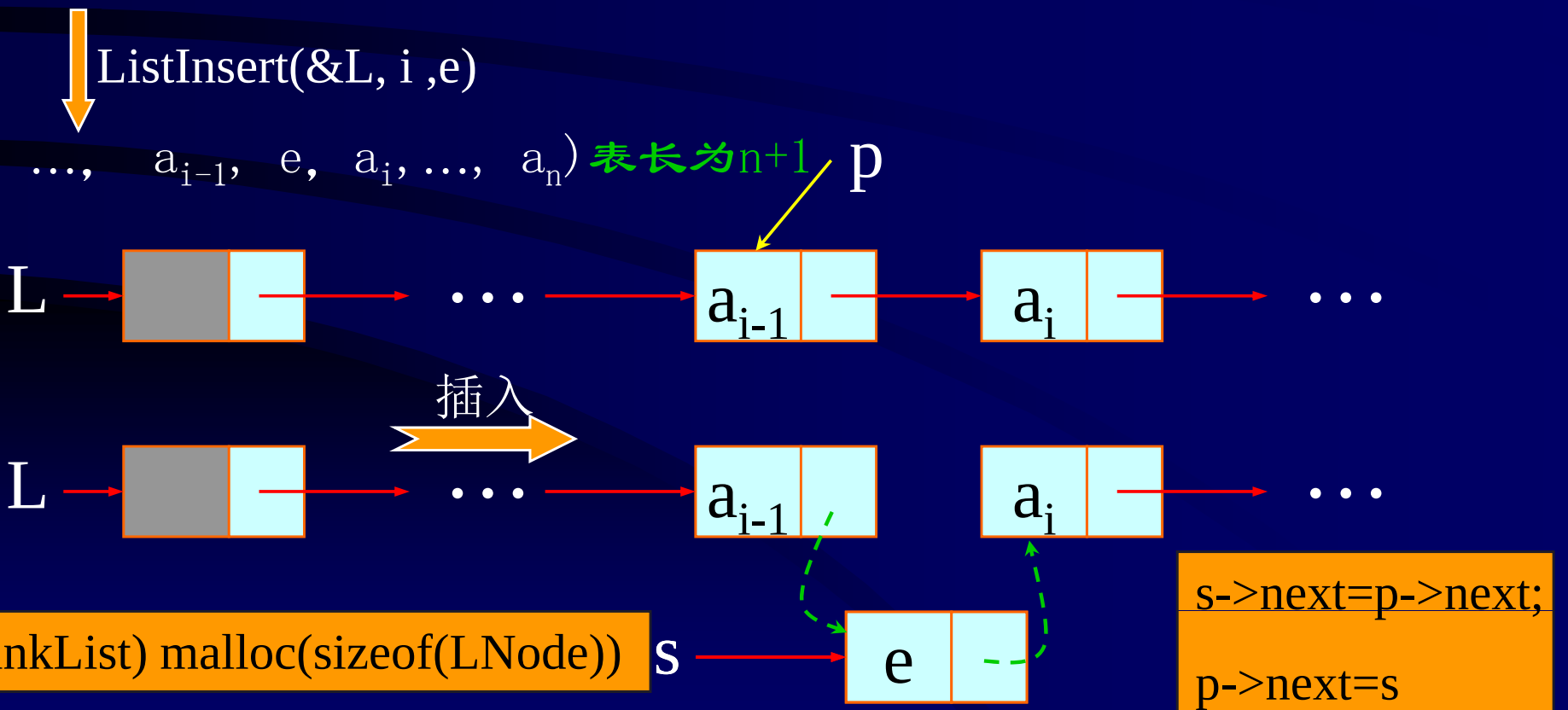
2.3线性表的链式存储结构（续）

单链表上插入运算的实现（一）

$(a_1, \ldots, a_i, a_{i+1}, \ldots, a_n)$ 表长为n

ListInsert(&L, i ,e)

$(a_1, \ldots, a_{i-1}, e, a_i, \ldots, a_n)$ 表长为n+1



2.3线性表的链式存储结构（续）

DS/SS

单链表上插入运算的实现（二）

Status ListInsert_L(LinkList &L, int i, ElemType e) {//
在带头结点的单链表L的第i个元素之前插入

第一步：判断参数是否合法合理，否则出错；

第二步：在物理空间中找到插入位置(i-1)；

第三步：插入前的准备工作;

第四步：插入；

} //ListInsert_L

2.3线性表的链式存储结构（续）

单链表上插入运算效率分析

分析：

从算法流程上看，找到插入位置后插入元素的时间花费是常数级，而算法执行时间主要花在插入前插入位置的查找上，而查找时间与插入位置i有关。具体而言，在第i个元素之前插入需要找到指向第i-1个元素的指针。

$T(n)=O(n)$

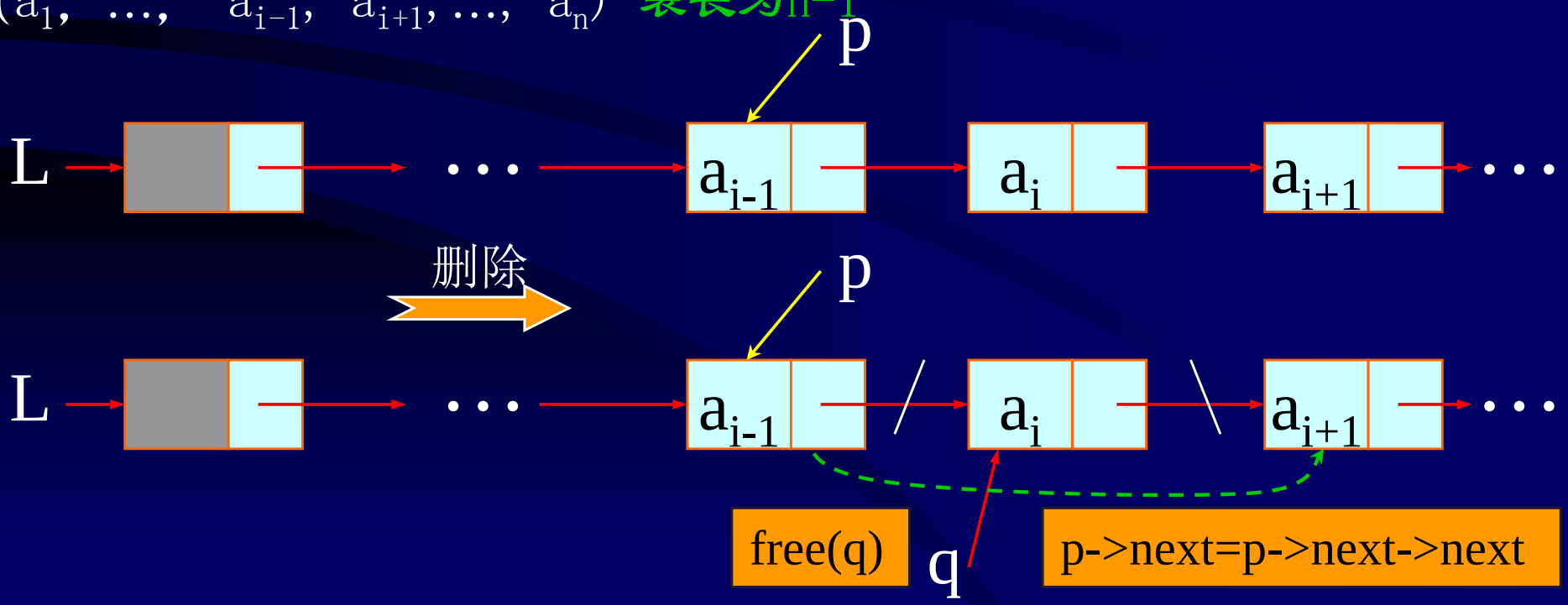
2.3线性表的链式存储结构（续）

单链表上删除运算的实现（一）

$(a_1, \ldots, a_i, a_{i+1}, \ldots, a_n)$ 表长为n

ListDelete(&L, i ,&e)

$(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$ 表长为n-1



2.3线性表的链式存储结构（续）

DS/SS

单链表上插入运算的实现（二）

```
Status ListDelete_L(LinkList &L, int i, ElemType &e) {
    第一步：判断参数是否合法合理，否则出错；
    第二步：在物理空间中找到删除位置(i-1)；
    第三步：删除；
    第四步：删除后的善后工作
} // ListDelete_L
```

2.3线性表的链式存储结构（续）

单链表上删除运算效率分析

分析：

从算法流程上看，找到删除位置后删除元素的时间花费是常数级，而算法执行时间主要花在删除前删除位置的查找上，而查找时间与删除位置i有关。具体而言，删除第i个元素需要找到指向第i-1个元素的指针。

$$T(n)=O(n)$$