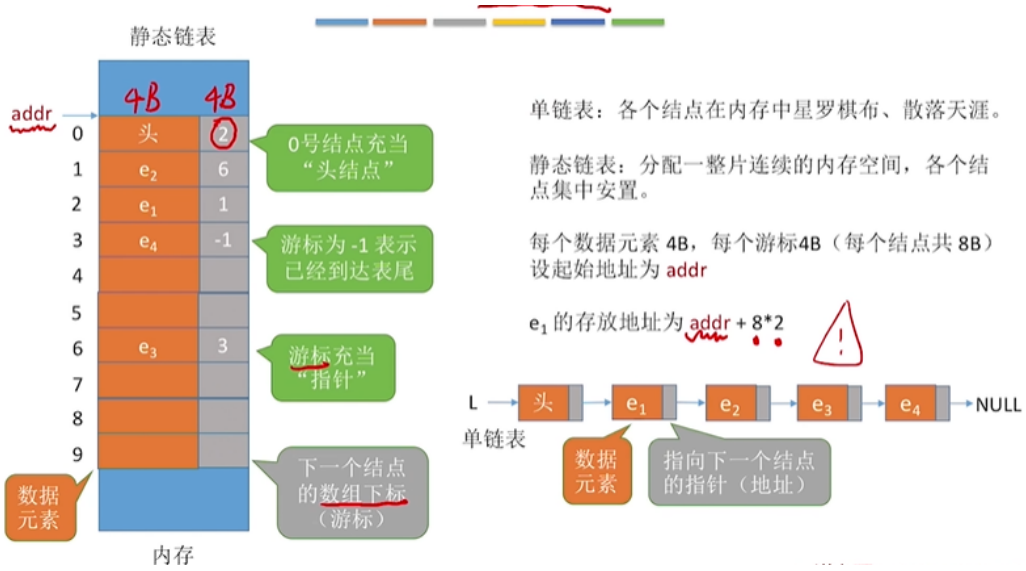
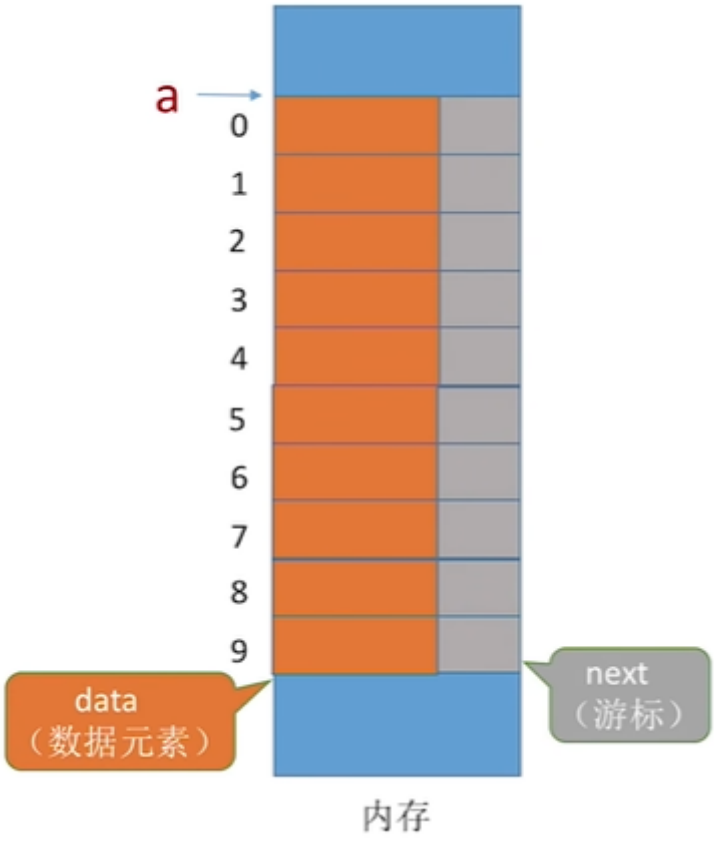
静态链表

- **什么是静态链表**



- **用代码定义一个静态链表**

```c
#define MAxSize 10    //静态链表的最大长度
struct Node           //静态链表结构类型的定义
{
    ElemType data;    //存储数据元素
    int next;         //下一元素的数组下标
};

void testSLinkList()
{
    struct Node a[MAxSize];     //数组a作为静态链表
    /*...后续代码...*/
}
```




```c
#define MAxSize 10
typedef struct
{
    ElemType data;
    int next;
}SLinkList[MaxSize];
//等价于
#define MaxSize 10
struct Node
{
    ElemType data;
    int next;
};
typedef struct Node SLinkList[MaxSize];  //可用SLinkList定义“一个长度为MaxSize的Node型数组”

void testSLinkList()
{
    SLinkList a;        //a是一个静态链表
    /*...后续代码...*/
}
//等价于
void testSLinkList()
{
    struct Node a[MaxSize];  //a是一个Node型数组
    /*...后续代码...*/
}
//类似于char *a等价于char a[]的关系
```

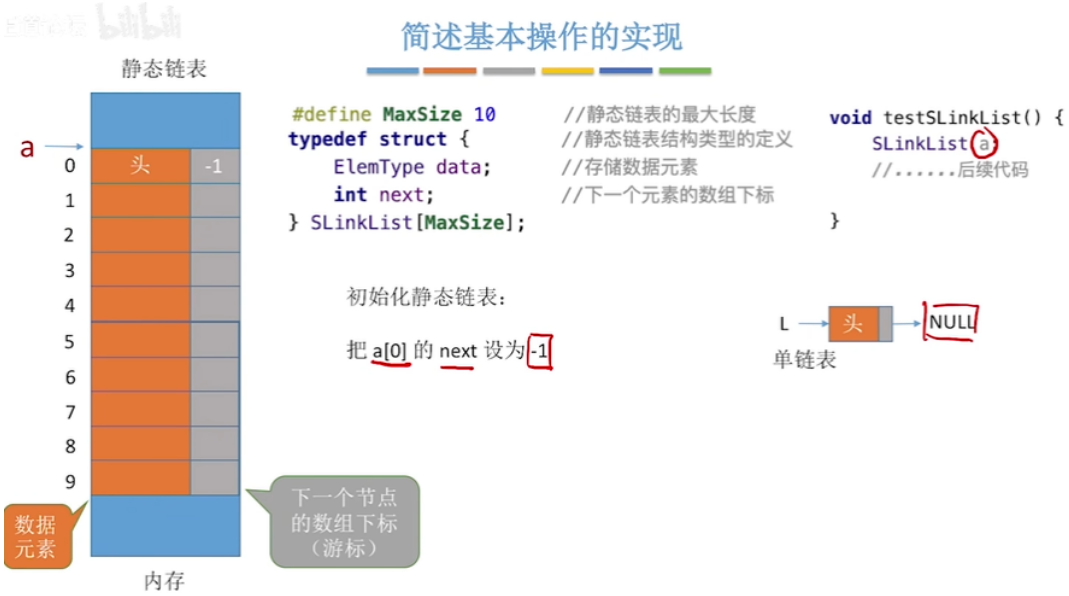
- **对猜想的验证**

```c
#define MAxSize 10    //静态链表的最大长度
struct Node
{
    int data;
    int next;
};
typedef struct
{
    int data;
    int next;
}SLinkList[MaxSize];

void testSLinkList()
{
    struct Node x;
    printf("size x = %d\n",sizeof(x));

    struct Node a[MaxSize];
    printf("size a = %d\n",sizeof(a));

    SLinkList b;
    printf("size b = %d\n",sizeof(b));
}

/*运行结果
size x = 8
size a = 80
size b = 80

Process finished with exit code 0*/
```

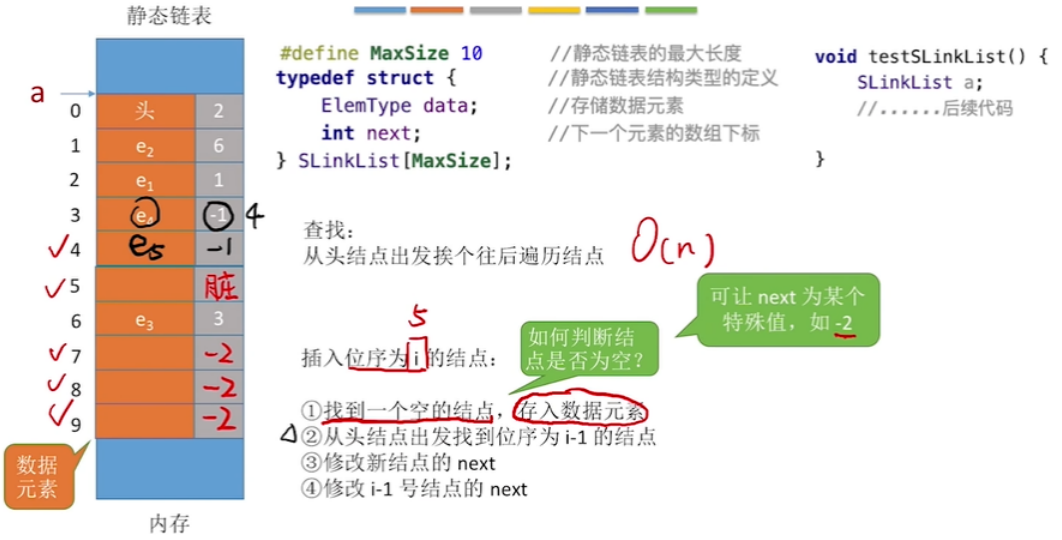
结论：

SLinkList b —— 相当于定义了一个长度为MaxSize的Node型数组

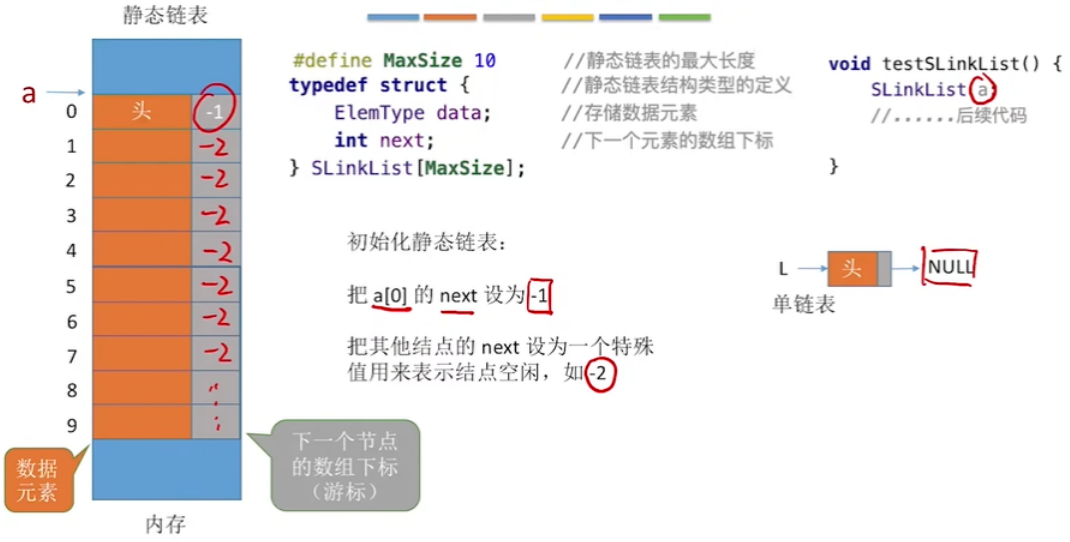
- **简述基本操作的实现**



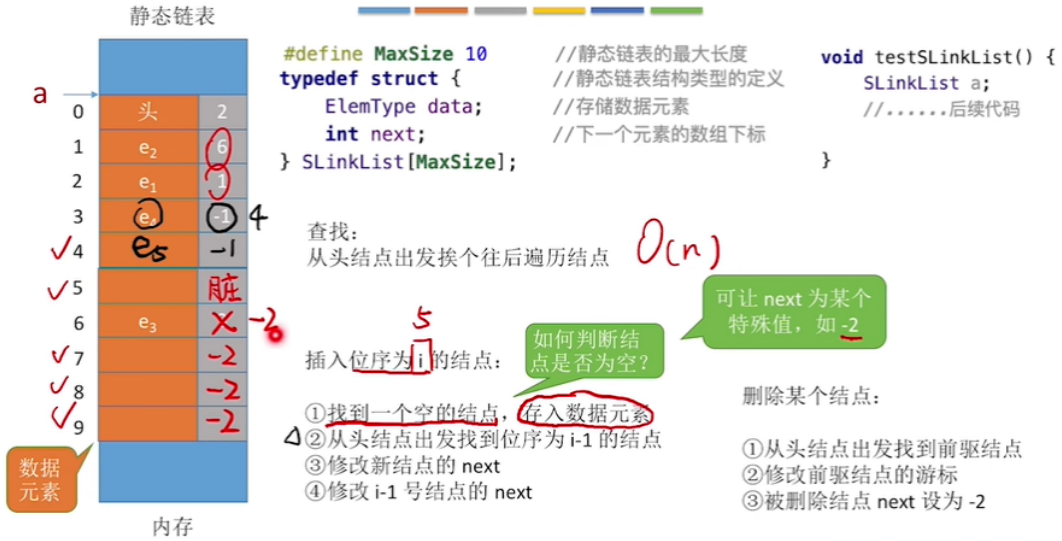
简述基本操作的实现



简述基本操作的实现

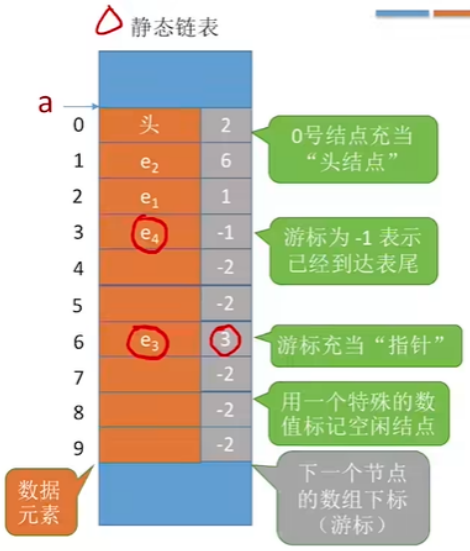


简述基本操作的实现



- **知识总结**

知识回顾与重要考点




```
#define MaxSize 10          //静态链表的最大长度
typedef struct {            //静态链表结构类型的定义
    ElemType data;          //存储数据元素
    int next;               //下一个元素的数组下标
} SLinkList[MaxSize];
void testSLinkList() {
    SLinkList a;
    //......后续代码

}
```

静态链表：用数组的方式实现的链表

优点：增、删 操作不需要大量移动元素
缺点：不能随机存取，只能从头结点开始依次往后查找；容量固定不可变

适用场景：①不支持指针的低级语言；②数据元素数量固定不变的场景（如操作系统的文件分配表FAT）？