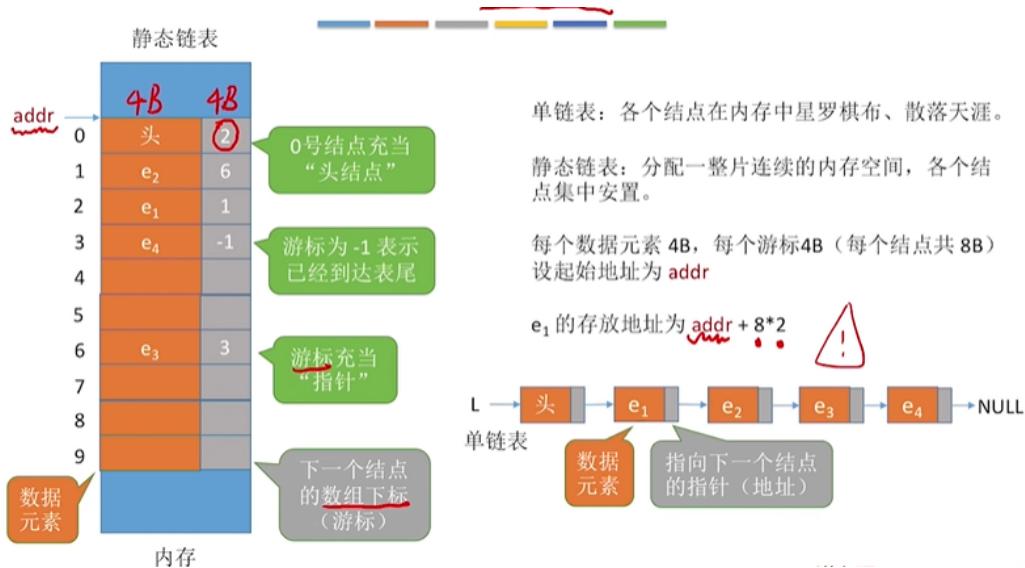


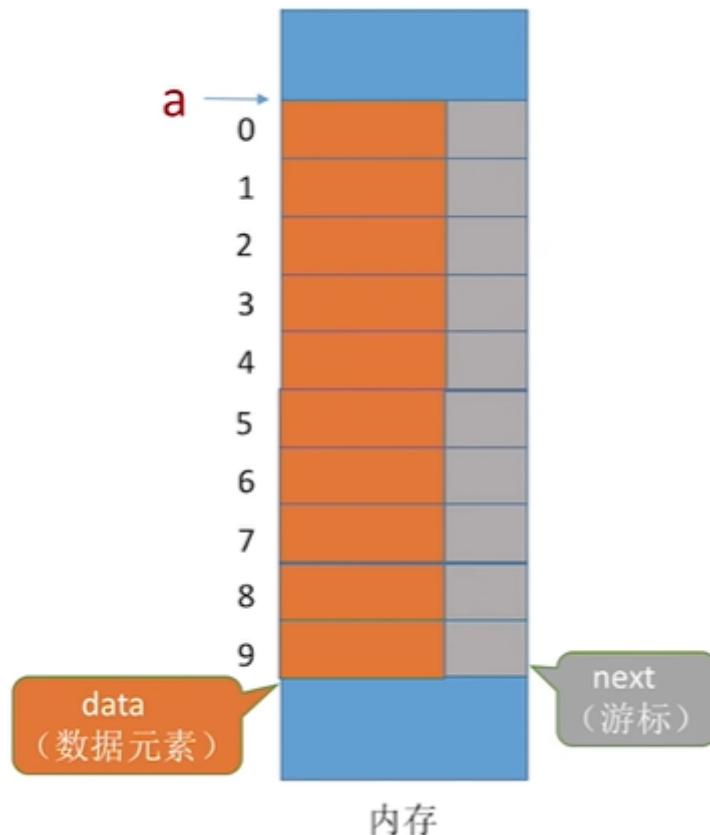
静态链表

- 什么是静态链表



- 用代码定义一个静态链表

```
1 #define MAXSize 10      //静态链表的最大长度
2 struct Node          //静态链表结构类型的定义
3 {
4     ELEMType data;   //存储数据元素
5     int next;        //下一元素的数组下标
6 };
7
8 void testSLinkList()
9 {
10    struct Node a[MAXSize]; //数组a作为静态链表
11    /*...后续代码...*/
12 }
```



```

1 #define MAXSIZE 10
2 typedef struct
3 {
4     ELEMTYPE data;
5     int next;
6 } SLinkList[MAXSIZE];
7 //等价于
8 #define MaxSize 10
9 struct Node
10 {
11     ELEMTYPE data;
12     int next;
13 };
14 typedef struct Node SLinkList[MAXSIZE]; //可用SLinkList定义“一个长度为
MaxSize的Node型数组”
15
16 void testSLinkList()
17 {
18     SLinkList a;           //a是一个静态链表
19     /*...后续代码...*/
20 }
21 //等价于
22 void testSLinkList()
23 {
24     struct Node a[MAXSIZE]; //a是一个Node型数组
25     /*...后续代码...*/
26 }
27 //类似于char *a等价于char a[]的关系

```

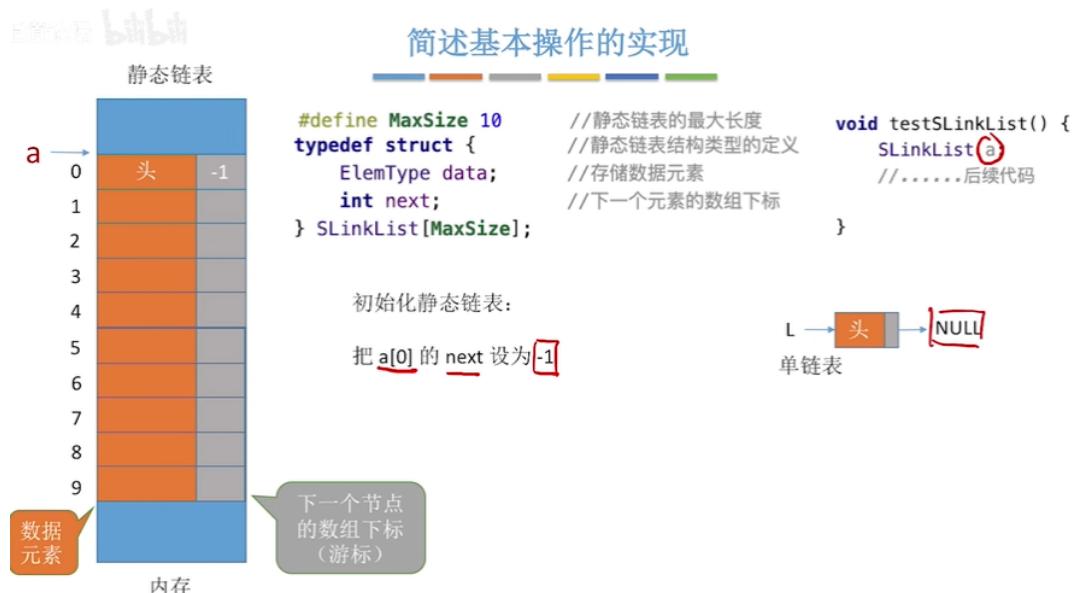
• 对猜想的验证

```
1 #define MaxSize 10      //静态链表的最大长度
2 struct Node
3 {
4     int data;
5     int next;
6 };
7 typedef struct
8 {
9     int data;
10    int next;
11 }SLinkList[MaxSize];
12
13 void testSLinkList()
14 {
15     struct Node x;
16     printf("size x = %d\n", sizeof(x));
17
18     struct Node a[MaxSize];
19     printf("size a = %d\n", sizeof(a));
20
21     SLinkList b;
22     printf("size b = %d\n", sizeof(b));
23 }
24
25 /*运行结果
26 size x = 8
27 size a = 80
28 size b = 80
29
30 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

静态链表

静态链表		
a	0	头 2
	1	e ₂ 6
	2	e ₁ 1
✓	3	e ₄ -1
✓	4	e ₅ -1
✓	5	空
✓	6	e ₃ 3
✓	7	-2
✓	8	-2
✓	9	-2
数据元素		内存

```
#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
}
```

查找:
从头结点出发挨个往后遍历结点

$O(n)$

- 插入位序为 i 的结点:
- ①找到一个空的结点, 存入数据元素
 - ②从头结点出发找到位序为 $i-1$ 的结点
 - ③修改新结点的 $next$
 - ④修改 $i-1$ 号结点的 $next$

如何判断结点是否为空?
可让 $next$ 为某个特殊值, 如 -2

简述基本操作的实现

静态链表

静态链表		
a	0	头 -1
	1	-2
	2	-2
	3	-2
	4	-2
	5	-2
	6	-2
	7	-2
	8	-2
	9	-2
数据元素		内存

```
#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
}
```

初始化静态链表:

把 $a[0]$ 的 $next$ 设为 -1



单链表

把其他结点的 $next$ 设为一个特殊值用来表示结点空闲, 如 -2

下一个节点的数组下标
(游标)

简述基本操作的实现

静态链表

静态链表		
a	0	头 2
	1	e ₂ 6
	2	e ₁ 1
✓	3	e ₄ -1
✓	4	e ₅ -1
✓	5	空
✓	6	e ₃ X -2
✓	7	-2
✓	8	-2
✓	9	-2
数据元素		内存

```
#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
}
```

查找:
从头结点出发挨个往后遍历结点

$O(n)$

- 插入位序为 i 的结点:
- ①找到一个空的结点, 存入数据元素
 - ②从头结点出发找到位序为 $i-1$ 的结点
 - ③修改新结点的 $next$
 - ④修改 $i-1$ 号结点的 $next$

如何判断结点是否为空?
可让 $next$ 为某个特殊值, 如 -2

删除某个结点:

- ①从头结点出发找到前驱结点
- ②修改前驱结点的游标
- ③被删除结点 $next$ 设为 -2

知识总结

知识回顾与重要考点

