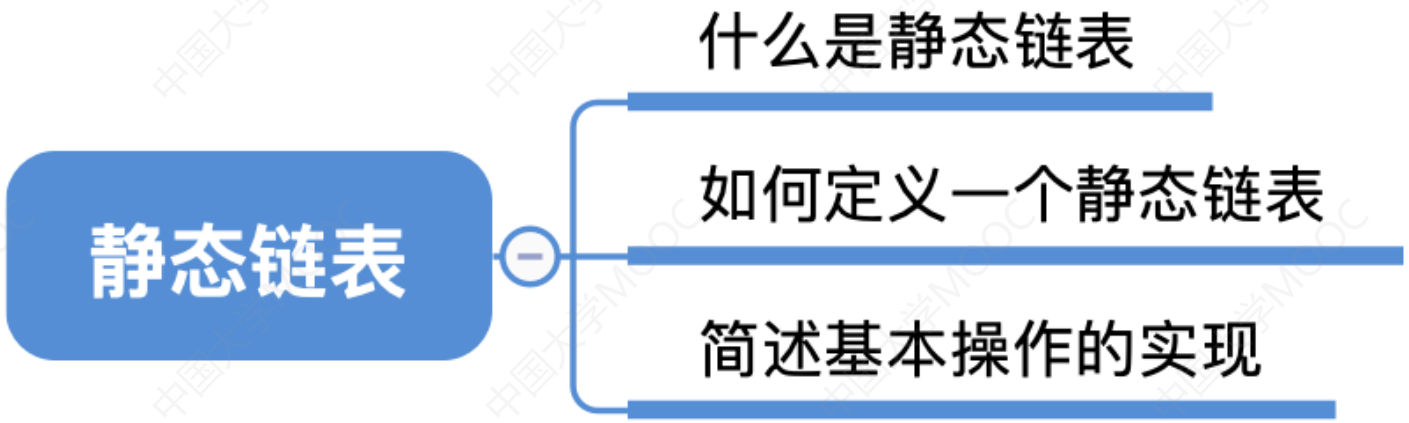
本节内容

静态链表

知识总览



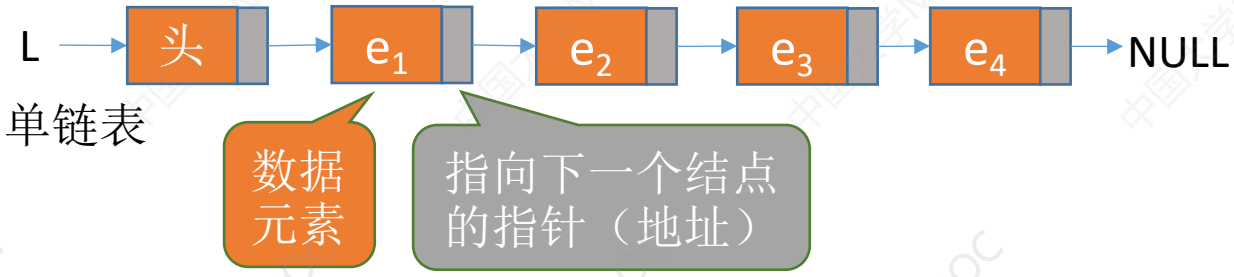
什么是静态链表

单链表：各个结点在内存中星罗棋布、散落天涯。

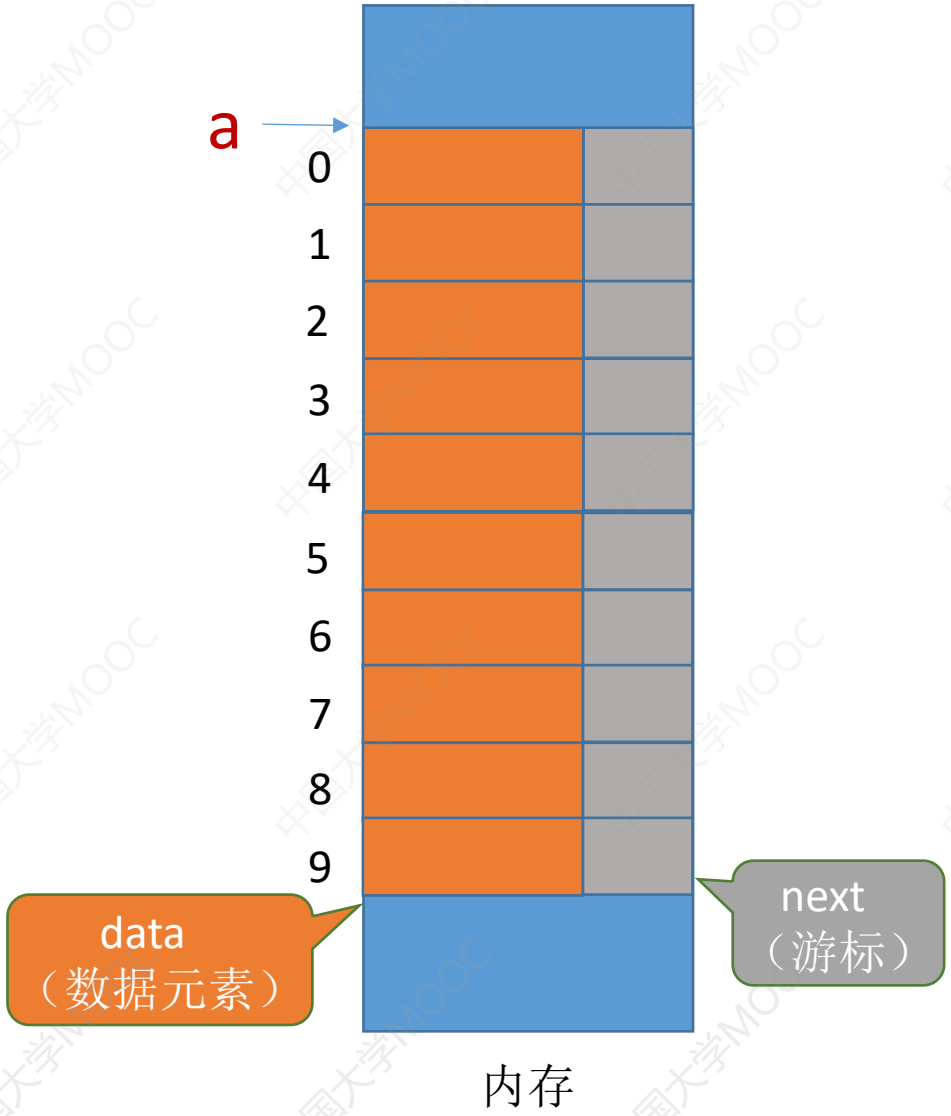
静态链表：分配一整片连续的内存空间，各个结点集中安置。

每个数据元素 4B，每个游标4B（每个结点共 8B）设起始地址为 addr

e_1 的存放地址为 addr + 8*2



用代码定义一个静态链表



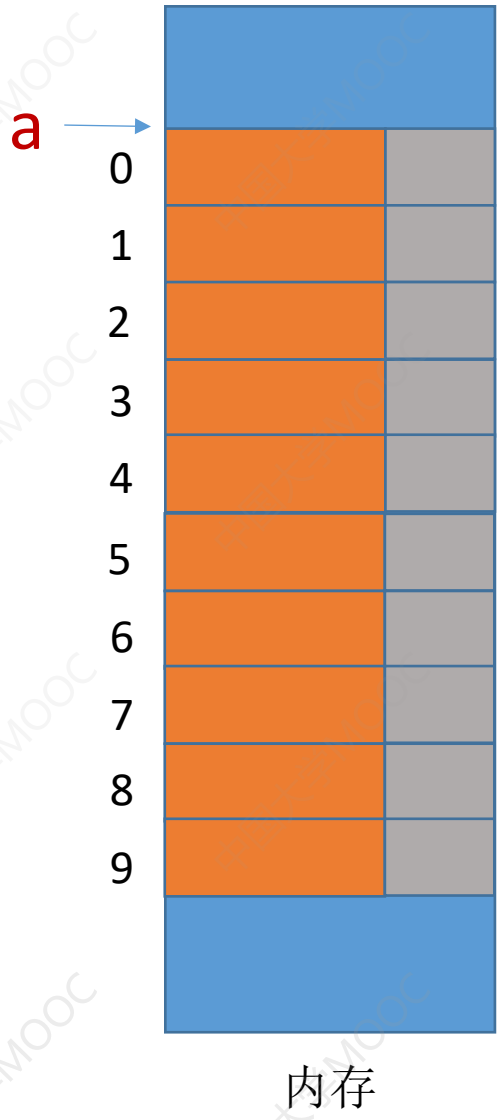

```
#define MaxSize 10          //静态链表的最大长度
struct Node{                //静态链表结构类型的定义
    ElemType data;          //存储数据元素
    int next;               //下一个元素的数组下标
};


void testSLinkList() {
    struct Node a[MaxSize];
    //......后续代码

}
```

数组 a 作为静态链表

用代码定义一个静态链表




```
#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];
```

等价

```
#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;               //下一个元素的数组下标
};
typedef struct Node SLinkList[MaxSize];
```

可用 SLinkList 定义“一个长度为 MaxSize 的Node 型数组”

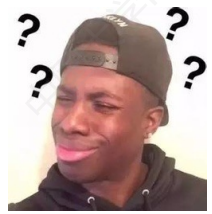
a 是一个静态链表

```
void testSLinkList() {
    SLinkList a;
    //......后续代码
}
```

等价

```
void testSLinkList() {
    struct Node a[MaxSize];
    //......后续代码
}
```

a 是一个Node型数组



对猜想的验证

```
#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
    int data;        //存储数据元素
    int next;        //下一个元素的数组下标
};
typedef struct {     //静态链表结构类型的定义
    int data;        //存储数据元素
    int next;        //下一个元素的数组下标
} SLinkList[MaxSize];

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

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

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

结论：

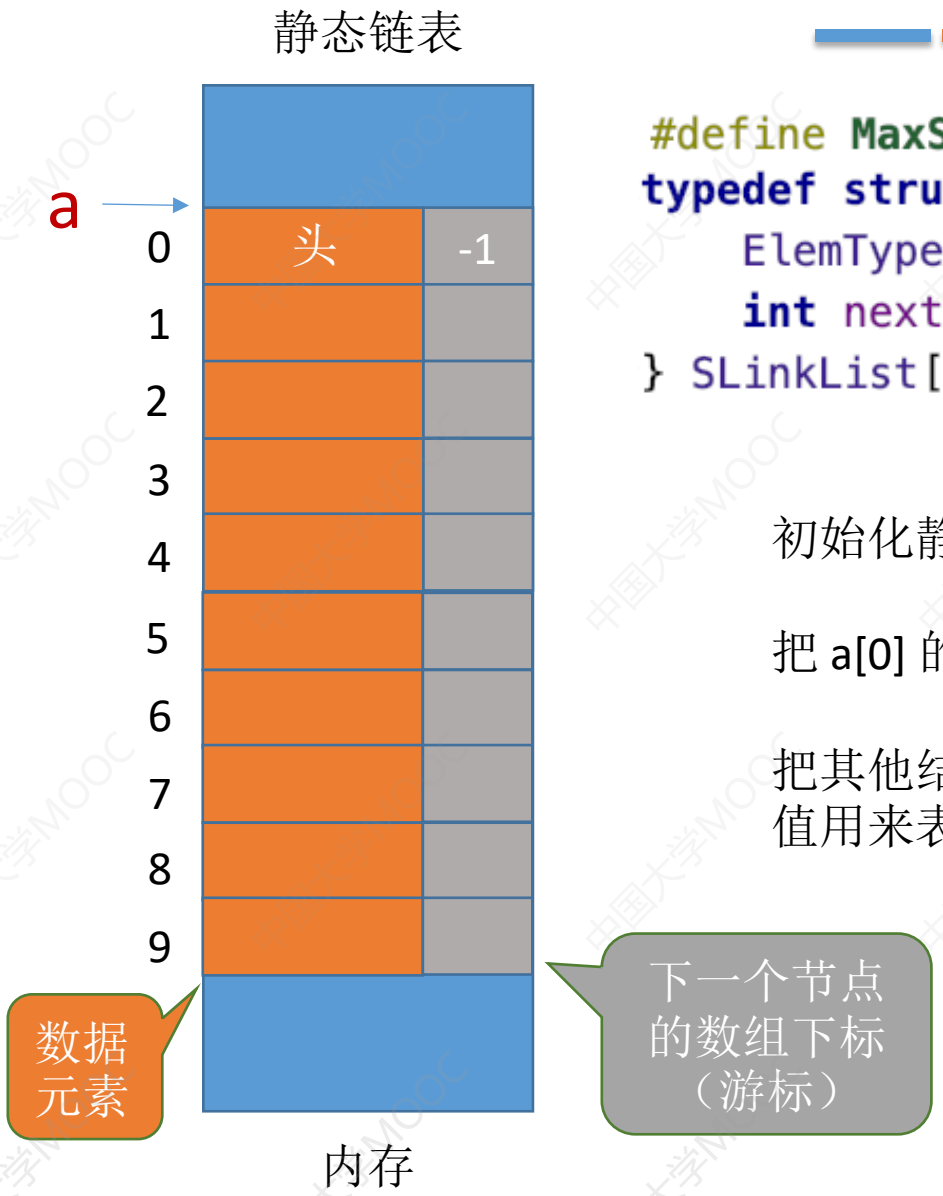
SLinkList b ——相当于定义了一个长度为 MaxSize 的 Node型数组

运行结果

```
sizeX=8
sizeA=80
sizeB=80

Process finished with exit code 0
```

简述基本操作的实现




```
#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：

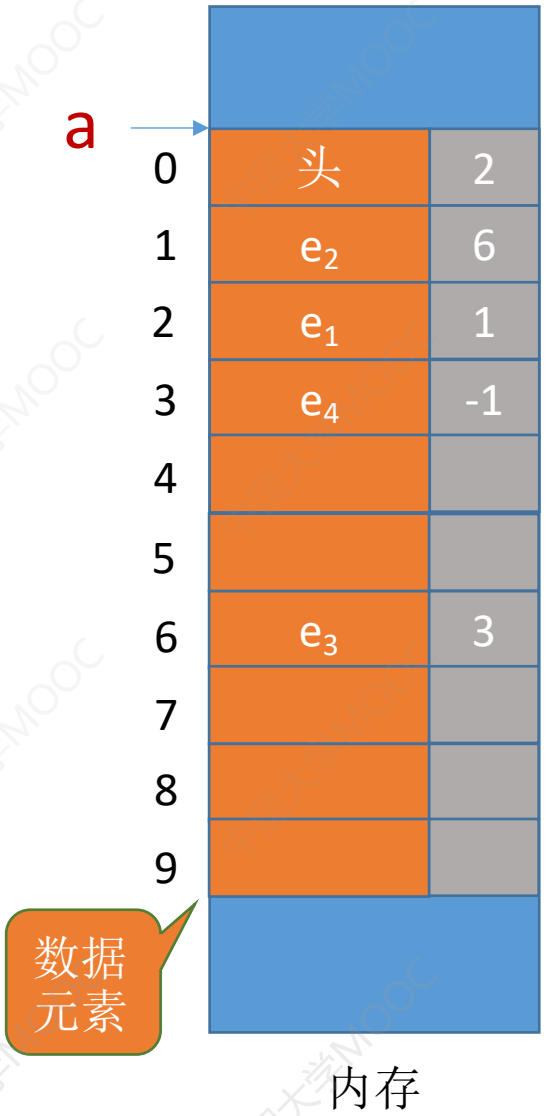
把 a[0] 的 next 设为 -1

把其他结点的 next 设为一个特殊值用来表示结点空闲，如 -2

单链表

简述基本操作的实现

静态链表




```
#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

插入位序为 i 的结点：

如何判断结点是否为空？

可让 next 为某个特殊值，如 -2

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

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）

欢迎大家对本节视频进行评价~

学员评分：2.3.5静态链表

扫一扫二维码打开或分享给好友

– 腾讯文档 –

可多人实时在线编辑，权限安全可控

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研