

本节内容

静态链表

知识总览

静态链表

什么是静态链表

如何定义一个静态链表

简述基本操作的实现

什么是静态链表

静态链表

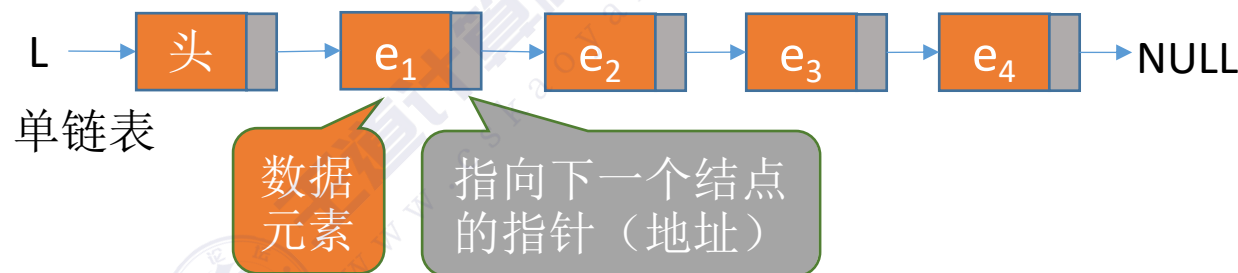


单链表：各个结点在内存中星罗棋布、散落天涯。

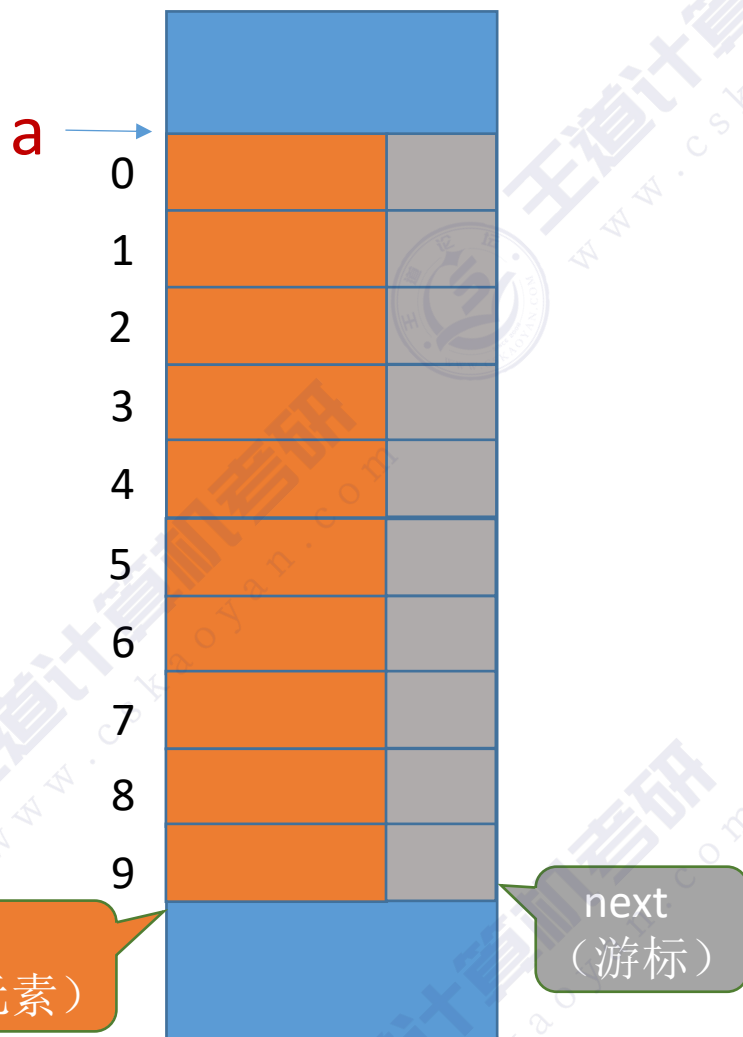
静态链表：分配一整片连续的内存空间，各个结点集中安置。

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**

e₁ 的存放地址为 **addr + 8*2**



用代码定义一个静态链表



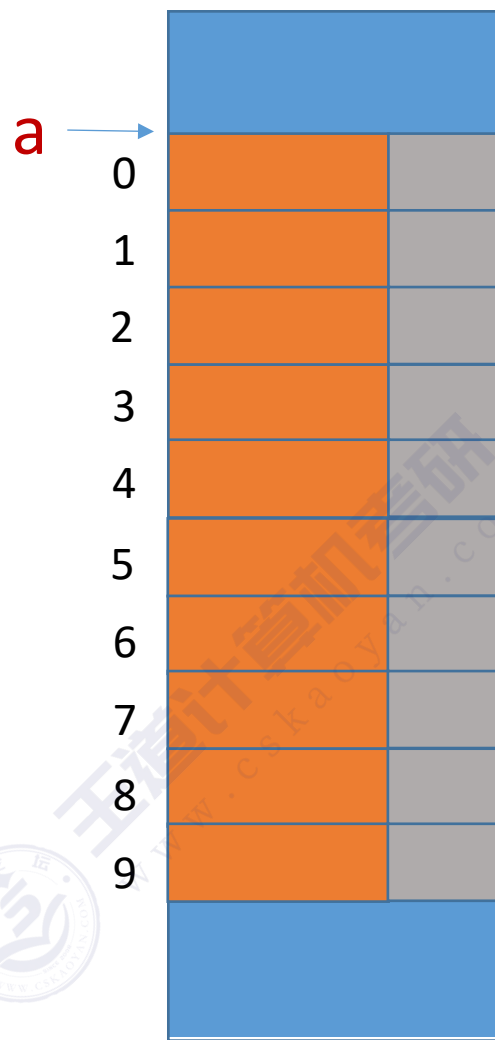
```
#define MaxSize 10  
struct Node{  
    ElemType data;  
    int next;  
};
```

//静态链表的最大长度
//静态链表结构类型的定义
//存储数据元素
//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



内存

```
#define MaxSize 10
typedef struct {
    ElemType data;
    int next;
} SLinkList[MaxSize];
```

//静态链表的最大长度
//静态链表结构类型的定义
//存储数据元素
//下一个元素的数组下标



```
#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”

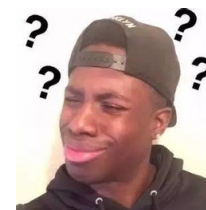
a 是一个静态链表

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



```
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

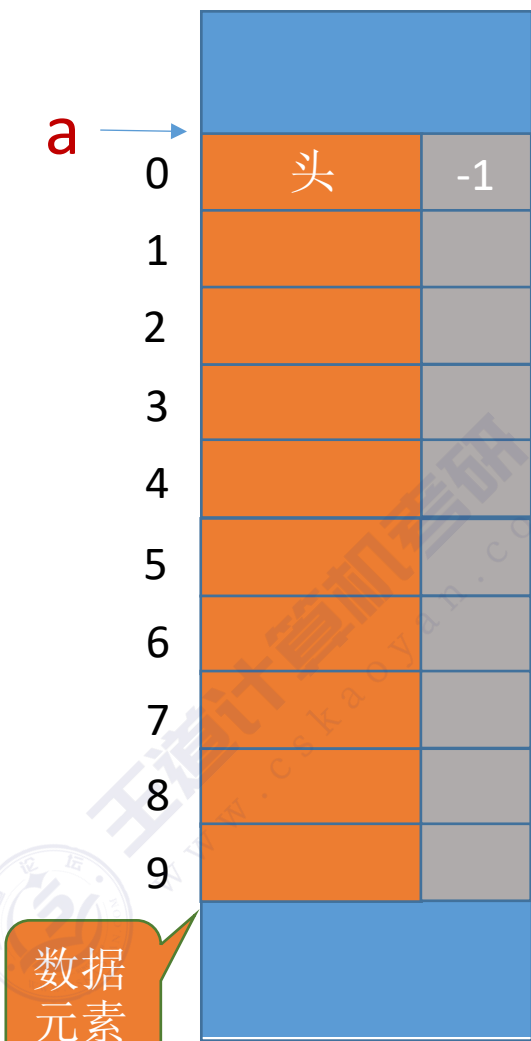
运行结果

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

Process finished with exit code 0

简述基本操作的实现

静态链表



```
#define MaxSize 10
typedef struct {
    ElemType data;
    int next;
} SLinkList[MaxSize];
```

//静态链表的最大长度
//静态链表结构类型的定义
//存储数据元素
//下一个元素的数组下标

```
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



简述基本操作的实现

静态链表

a →

0	头	2
1	e ₂	6
2	e ₁	1
3	e ₄	-1
4		
5		
6	e ₃	3
7		
8		
9		

数据元素

内存

```
#define MaxSize 10
typedef struct {
    ElemType data;
    int next;
} SLinkList[MaxSize];
```

//静态链表的最大长度
//静态链表结构类型的定义
//存储数据元素
//下一个元素的数组下标

```
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:

- 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

知识回顾与重要考点

静态链表

a →

0	头	2
1	e ₂	6
2	e ₁	1
3	e ₄	-1
4		-2
5		-2
6	e ₃	3
7		-2
8		-2
9		-2

数据元素

0号结点充当“头结点”

游标为 -1 表示已经到达表尾

游标充当“指针”

用一个特殊的数值标记空闲结点

下一个节点的数组下标 (游标)

```
#define MaxSize 10
typedef struct {
    ElemType data;
    int next;
} SLinkList[MaxSize];

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

//静态链表的最大长度
//静态链表结构类型的定义
//存储数据元素
//下一个元素的数组下标

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）