

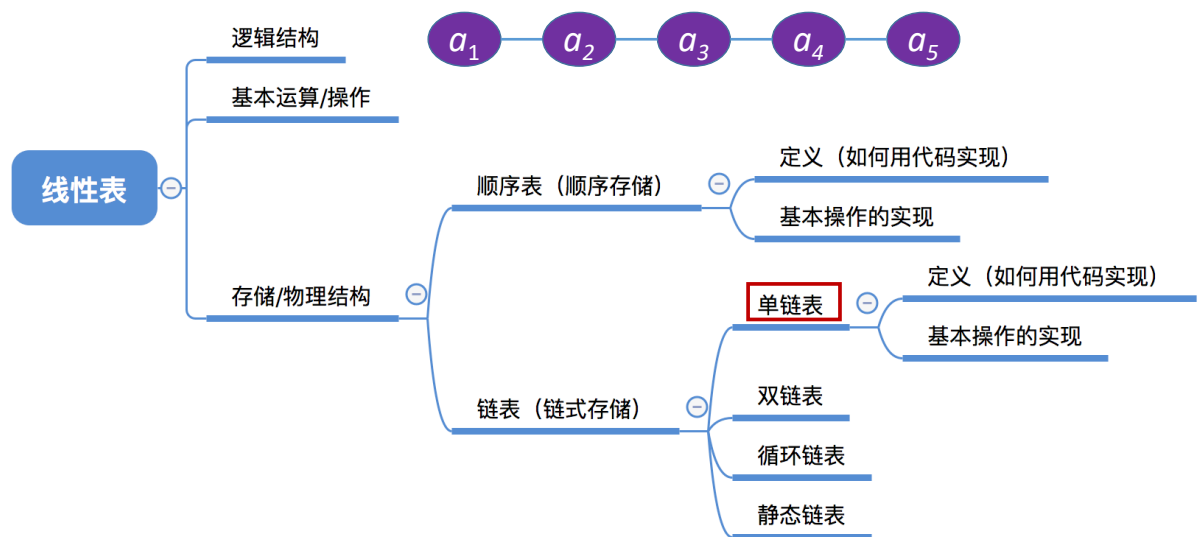
本节内容

# 单链表 定义

王道考研/CSKAOYAN.COM

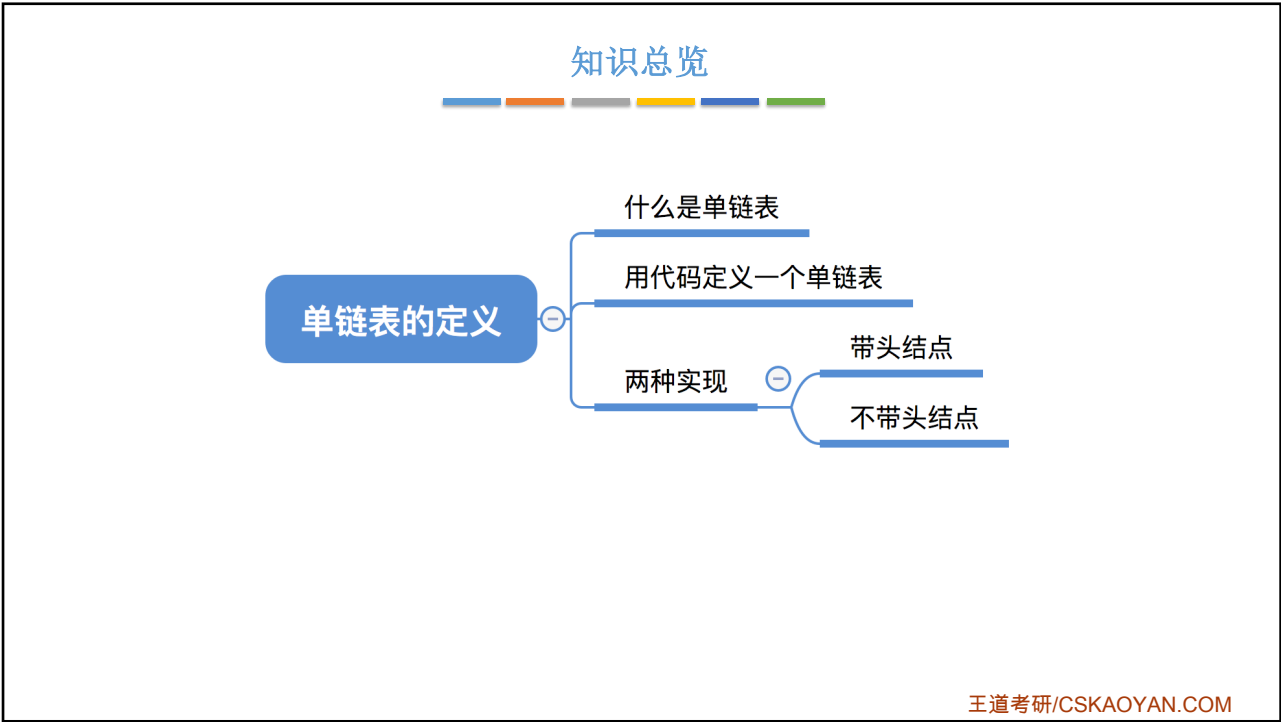
1

## 知识总览

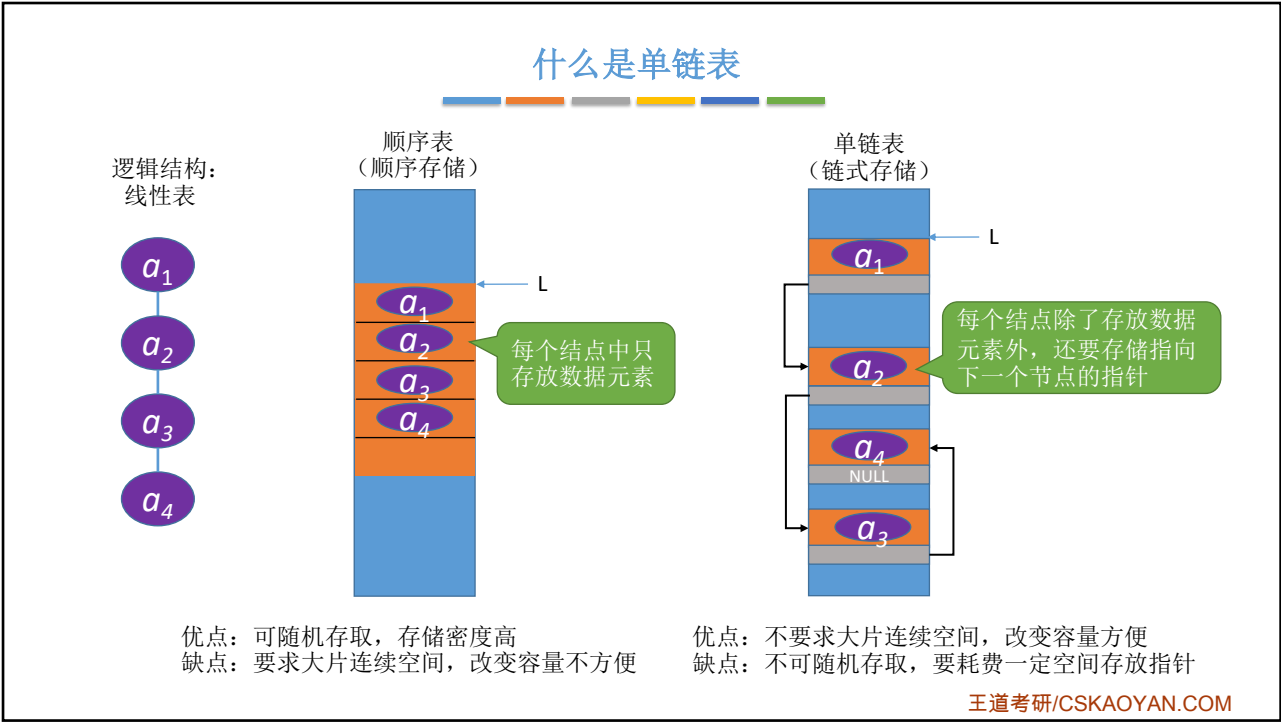


王道考研/CSKAOYAN.COM

2

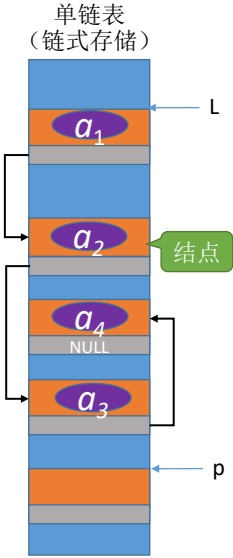


3



4

单链表  
(链式存储)



### 用代码定义一个单链表

结点

```
struct LNode{  
    ElemType data;  
    struct LNode *next;  
};
```

数据域

指针域

//定义单链表结点类型  
//每个节点存放一个数据元素  
//指针指向下一个节点

```
struct LNode * p = (struct LNode *) malloc(sizeof(struct LNode));
```

增加一个新的结点：在内存中申请一个结点所需空间，并用指针 p 指向这个结点

事情并不简单

typedef 关键字 —— 数据类型重命名

```
typedef <数据类型> <别名>  
typedef int zhengshu;  
typedef int *zhengshuzhizhen;  
int x = 1;  
int *p;
```

巴啦啦能量

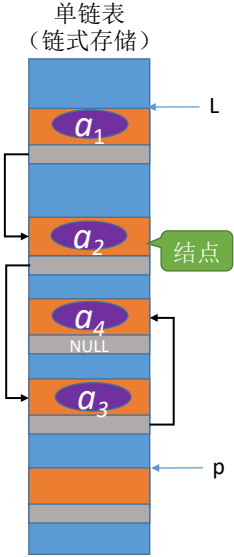
等价

```
zhengshu x = 1;  
zhengshuzhizhen p;
```

王道考研/CSKAOYAN.COM

5

单链表  
(链式存储)



### 用代码定义一个单链表

结点

```
struct LNode{  
    ElemType data;  
    struct LNode *next;  
};
```

数据域

指针域

//定义单链表结点类型  
//每个节点存放一个数据元素  
//指针指向下一个节点

```
struct LNode * p = (struct LNode *) malloc(sizeof(struct LNode));
```

增加一个新的结点：在内存中申请一个结点所需空间，并用指针 p 指向这个结点

事情并不简单

typedef 关键字 —— 数据类型重命名

```
typedef <数据类型> <别名>  
typedef struct LNode LNode;  
LNode * p = (LNode *) malloc(sizeof(LNode));
```

巴啦啦能量

原来如此，简单!

王道考研/CSKAOYAN.COM

6

王道考,cskaoyan.com

公众号：最新考研资料 免费分享

3

### 用代码定义一个单链表

单链表  
(链式存储)

```

typedef struct LNode{
    ElemType data;
    struct LNode *next;
}LNode, *LinkList;

struct LNode{
    ElemType data;
    struct LNode *next;
};

typedef struct LNode LNode;
typedef struct LNode *LinkList;
        
```

//定义单链表结点类型  
//每个节点存放一个数据元素  
//指针指向下一个节点

网络释义  
— 单链表

//定义单链表结点类型  
//每个节点存放一个数据元素  
//指针指向下一个节点

要表示一个单链表时，只需声明一个**头指针**L，指向单链表的第一个结  
**LNode \* L;** //声明一个指向单链表第一个结点的指针  
 或: **LinkList L;** //声明一个指向单链表第一个结点的指针

代码可读性更强

王道考研/CSKAOYAN.COM

7

### 用代码定义一个单链表

单链表  
(链式存储)

```

typedef struct LNode{
    ElemType data;
    struct LNode *next;
}LNode, *LinkList;

LNode *GetElem(LinkList L, int i){
    int j=1;
    LNode *p=L->next;
    if(i==0)
        return L;
    if(i<1)
        return NULL;
    while(p!=NULL && j<i){
        p=p->next;
        j++;
    }
    return p;
}
        
```

//定义单链表结点类型  
//每个节点存放一个数据元素  
//指针指向下一个节点

强调返回的是一个结点

强调这是一个单链表

强调这是一个单链表  
强调这是一个结点

——使用 LinkList  
——使用 LNode \*

王道考研/CSKAOYAN.COM

8

## 用代码定义一个单链表

头插法建立单链表的算法如下：

```
LinkedList List_HeadInsert(LinkedList &L) { // 逆向建立单链表
    LNode *s; int x;
    L = (LinkedList)malloc(sizeof(LNode)); // 创建头结点
    L->next = NULL; // 初始为空链表
    scanf("%d", &x); // 输入结点的值
    while(x != 9999) { // 输入 9999 表示结束
        s = (LNode*)malloc(sizeof(LNode)); // 创建新结点
        s->data = x;
        s->next = L->next;
        L->next = s; // 将新结点插入表中，L 为头指针
        scanf("%d", &x);
    }
    return L;
}
```

强调这是一个单链表  
强调这是一个结点

——使用 LinkedList  
——使用 LNode \*

王道考研/CSKAOYAN.COM

9

## 不带头结点的单链表

```
typedef struct LNode{
    ElemType data;
    struct LNode *next;
}LNode, *LinkedList;
```

// 定义单链表结点类型  
// 每个结点存放一个数据元素  
// 指针指向下一个节点

// 初始化一个空的单链表

```
bool InitList(LinkedList &L) {
```

```
    L = NULL; // 空表，暂时还没有任何结点
```

```
    return true;
```

```
}
```

```
void test(){
```

```
    LinkedList L; // 声明一个指向单链表的指针
```

```
    // 初始化一个空表
```

```
    InitList(L);
```

```
    // .....后续代码.....
```

```
}
```

注意，此处  
并没有创建  
一个结点

防止脏数据

// 判断单链表是否为空

```
bool Empty(LinkedList L) {
```

```
    if (L == NULL)
```

```
        return true;
```

```
    else
```

```
        return false;
```

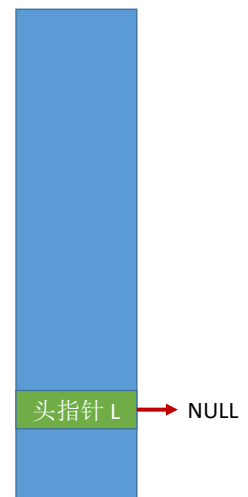
```
}
```

或：bool Empty(LinkedList L) {

```
    return (L == NULL);
```

```
}
```

内存



王道考研/CSKAOYAN.COM

10

## 带头结点的单链表

```

typedef struct LNode{
    ElemType data;           //定义单链表结点类型
    struct LNode *next;      //每个节点存放一个数据元素
                              //指针指向下一个节点
}LNode, *LinkList;

//初始化一个单链表（带头结点）
bool InitList(LinkList &L) {
    L = (LNode *) malloc(sizeof(LNode)); //分配一个头结点
    if (L==NULL)                          //内存不足，分配失败
        return false;
    L->next = NULL;                       //头结点之后暂时还没有节点
    return true;
}

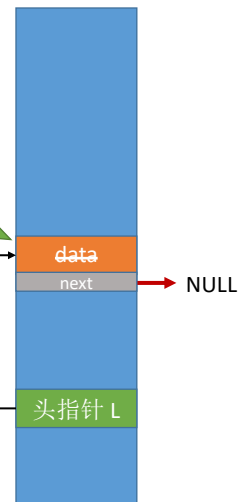
//判断单链表是否为空（带头结点）
bool Empty(LinkList L) {
    if (L->next == NULL)
        return true;
    else
        return false;
}

void test(){
    LinkList L; //声明一个指向单链表的指针
    //初始化一个空表
    InitList(L);
    //.....后续代码.....
}

```

头结点不  
存储数据

内存

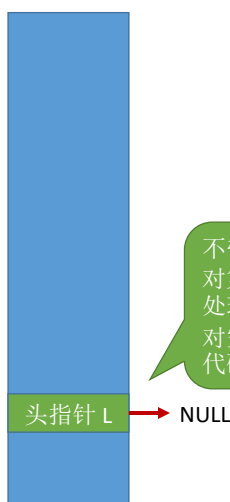


王道考研/CSKAOYAN.COM

11

## 不带头结点 v.s. 带头结点

不带头

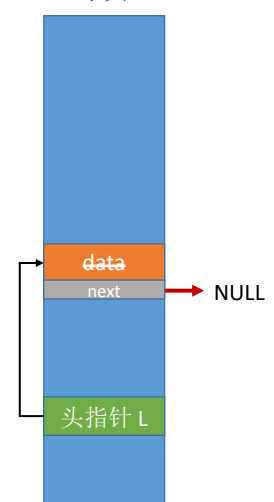


带头结点，写代码更方便，用过都说好



不带头结点，写代码更麻烦  
对第一个数据结点和后续数据结点的  
处理需要用不同的代码逻辑  
对空表和非空表的处理需要用不同的  
代码逻辑

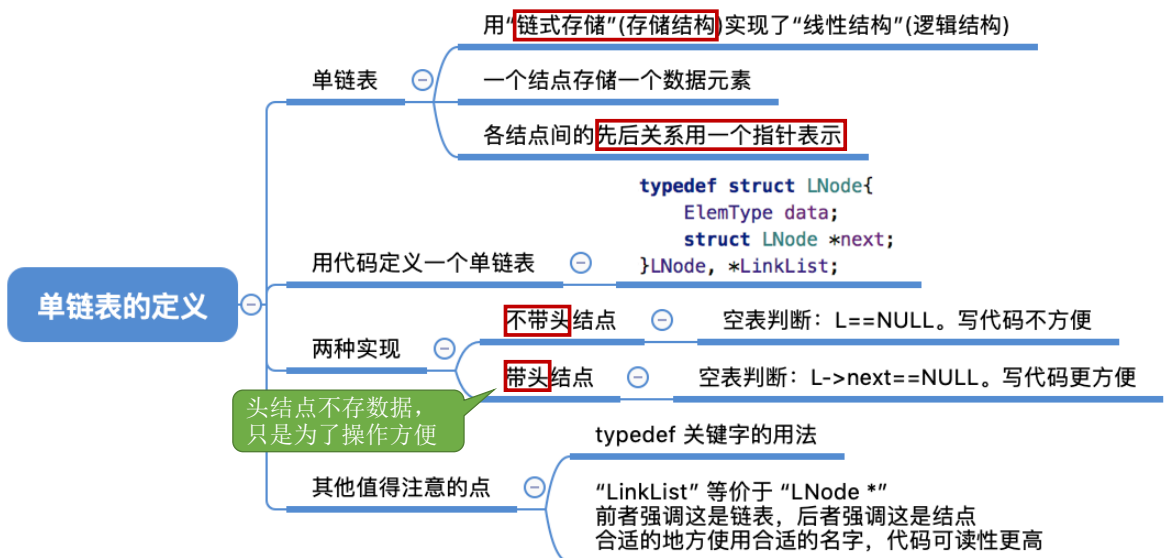
带头



王道考研/CSKAOYAN.COM

12

## 知识回顾与重要考点



王道考研/CSKAOYAN.COM