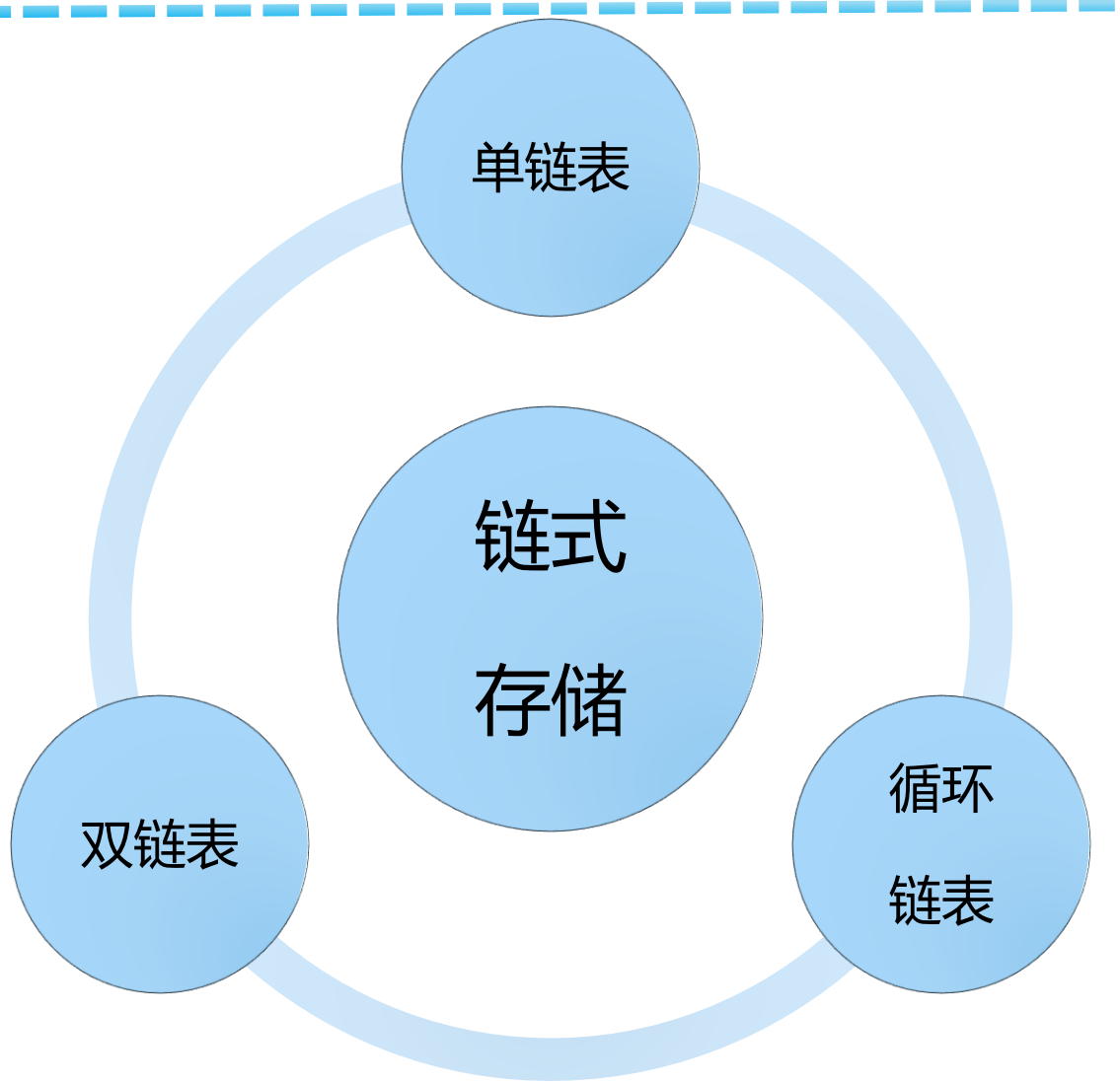
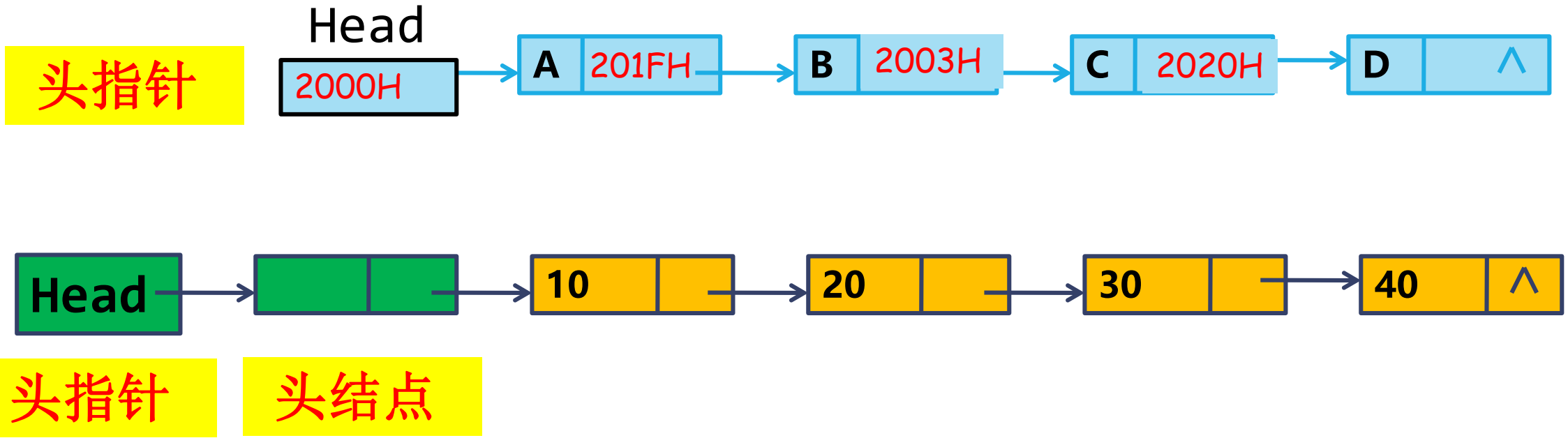
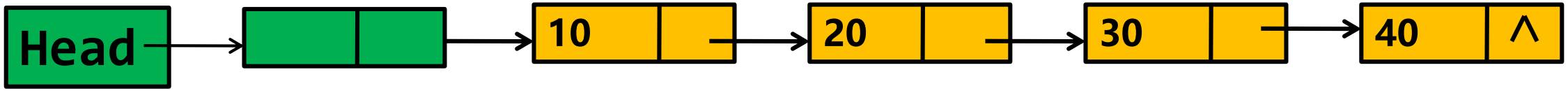
链表



单链表

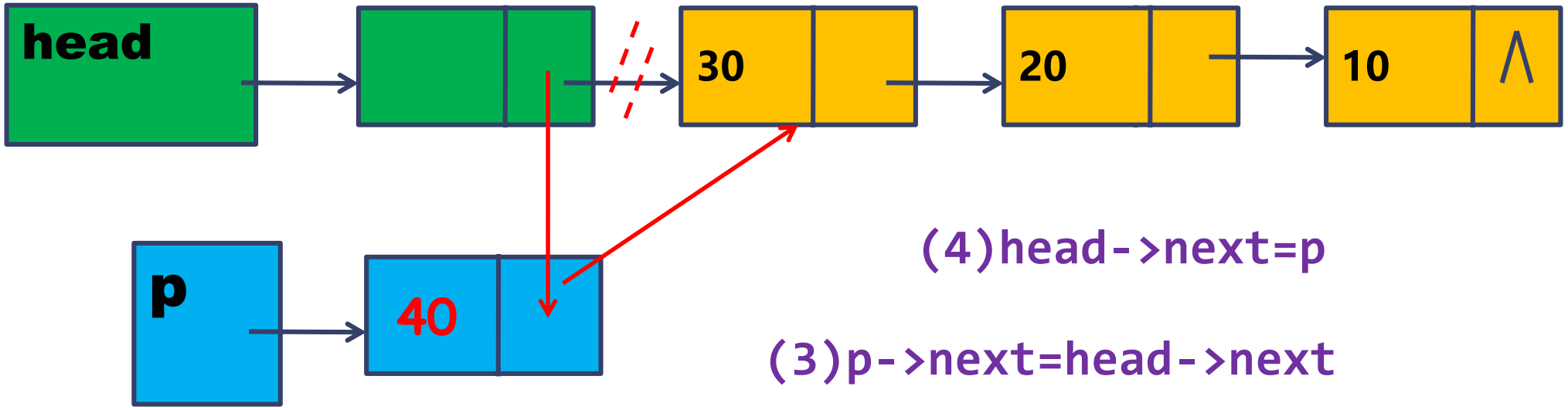



```
struct  Node {
 int    data;
 struct Node  *next;
}
```



建立	插入	删除	查找	输出
头插法	前插法	删除P后继	序号查找	顺序打印
尾插法	后插法	删除P本身	值查找	
递归法				

2.5.3 头插法建立单链表



(4)head->next=p

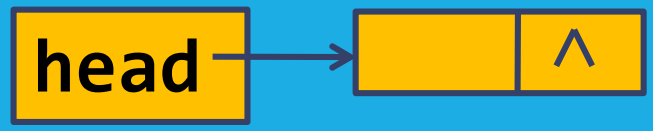
(3)p->next=head->next

(2)p->data=data

(1)p=malloc(sizeof(struct Node))

2.5.1建立空的单链表　　算法2-12，算法2-13

```
1  LinkList SetNullList_Link() //创建带有头结点的空链表
2  {
3          LinkList head =(LinkList)malloc(sizeof(struct Node));//申请头结点空间
4          if (head!=NULL)
5              head->next = NULL;
6          else
7              printf("alloc failure");
8          return head; //返回头指针
9  }
```

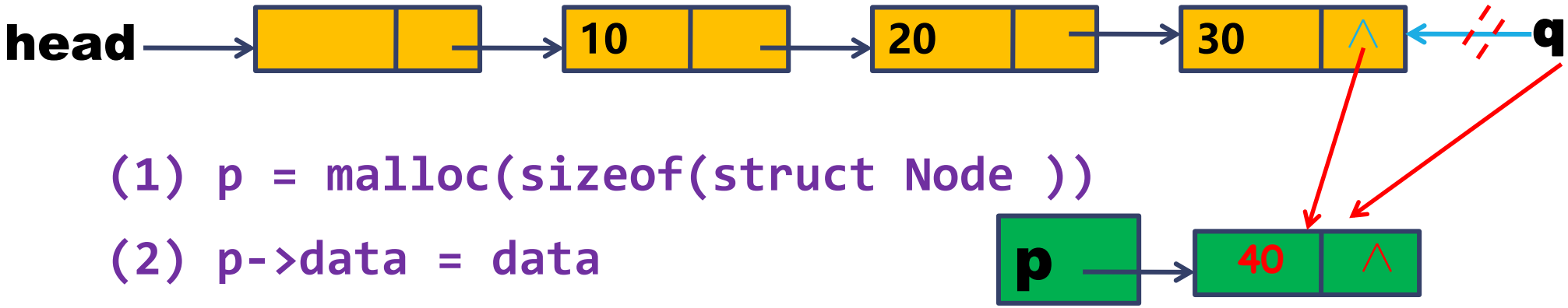



```
1  int IsNull_Link(LinkList head) //判断链表是否为空
2  {
3          return(head->next==NULL);
4  }
```

2.5.3 头插法建立单链表　　算法2-14

```
1   void CreateList_Head(struct Node *head)//头插法建立单链表
2   {
3           PNode p = NULL;  //临时使用
4           int data;
5           printf("请输入整型数据建立链表，以-1结束\n");
6           scanf("%d",&data);
7           while(data!=-1)
8           {
9                   p = (struct Node*)malloc(sizeof(struct Node)); //分配空间
10                  p->data = data;  //对数据域赋值
11                  p->next = head->next;  //next域赋值
12                  head->next = p;
13                  scanf("%d",&data);
14          }
15  }
```

2.5.4尾插法建立单链表




```
(1) p = malloc(sizeof(struct Node ))
(2) p->data = data
(3) p->next = NULL
(4) q->next = p
(5) q = p
```

2.5.4尾插法建立单链表　　算法2-15

```
1  void CreateList_Tail(struct Node* head)// 尾插法建立单链表
2  {
3          struct Node *p = NULL;   struct Node *q = head;
4          int data;  printf("请输入整型数据建立链表，以-1结束\n");
5          scanf("%d",&data);
6          while(data!=-1)
7          {
8                  p = (struct Node*)malloc(sizeof(struct Node)); //分配空间
9                  p->data = data; //数据域赋值
10                 p->next = NULL; //指针域赋值
11                 q->next = p;
12                 q = p;
13                 scanf("%d",&data);
14         }
15 }
```