



链表的构造

教学目标和要求

1.能够实现链表的构造（向前插入法，向后插入法）



1. 链表的构造

1. 构造链表的通用算法

步骤1) 构造“空链表”

步骤2) 读入第一个元素值

步骤3) 当读入的元素值不是“输入结束标记”时，循环执行步骤4~7

步骤4) 申请一个新结点

步骤5) 将读入的元素值存入新结点的值域

步骤6) 将新结点“插在”链表中

步骤7) 读入“下一个”元素值，转步骤3

步骤8) 构造完毕，返回首指针



1. 链表的构造（向前插入法）

步骤1) 构造“空链表”

步骤2) 读入第一个元素值

步骤3) 当读入的元素值不是“结束标记”时，循环执行步骤4~7

步骤4) 申请一个新结点

步骤5) 将读入的元素值存入新结点的值域

步骤6) 将新结点“插在”链表的**表头**处

步骤7) 读入“下一个”元素值，转步骤3

步骤8) 构造完毕，返回首指针



1. 链表的构造（向前插入法）

```
ptr creatlinkedA()  
{  
    ptr head, p; int x;  
    head= NULL; //将表头指针置空  
    scanf("%d", &x); //读入第一个元素  
    while (x!=End_elm) //当读的不是结束标记时循环  
    {  
        p=(ptr)malloc(sizeof(snode)); //申请一个存储结点  
        p->data=x; //置结点的值域  
        p->next=head; //插在表头处  
        head=p; //表头指针指向新结点  
        scanf("%d", &x); //读入下一个元素  
    }  
    return(head); //返回表头指针  
}
```



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);



head




x





向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x); 
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

head




x

1



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0) 
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

head


x

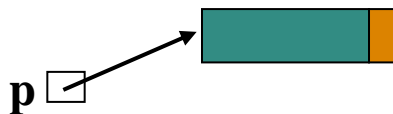
1



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode)); ←
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

head
↑



x
1



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x; ←
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

head
↑

p



x

1



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head; ←
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

head
↑

p



x
1



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);



head



p



x





向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);



head



p




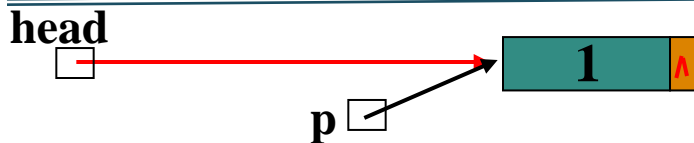
x

2



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0) 
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);



x

2



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode)); ←
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

head



p



x

2



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x; ←
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

head



p



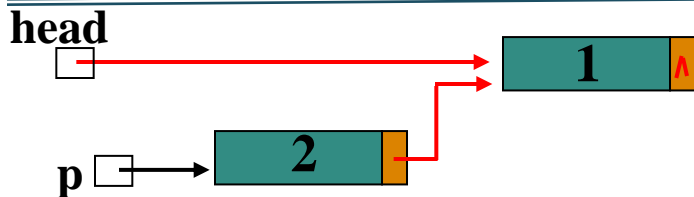
x

2



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head; ←
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);



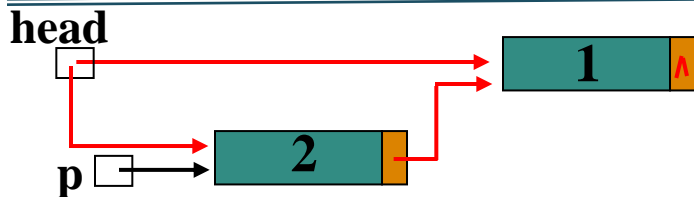
x

2



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p; ←
8. scanf("%d", &x);
9. }
9. return(head);



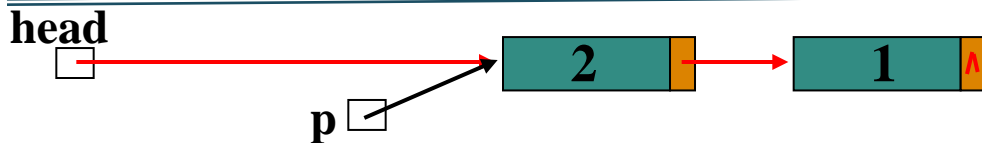
x

2



向前插入构造法示例，输入序列：1 2 3 0


1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x); ←
9. return(head);



x
3



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0) 
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

head



p



x

3



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

head



p



x

3



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);



head



p



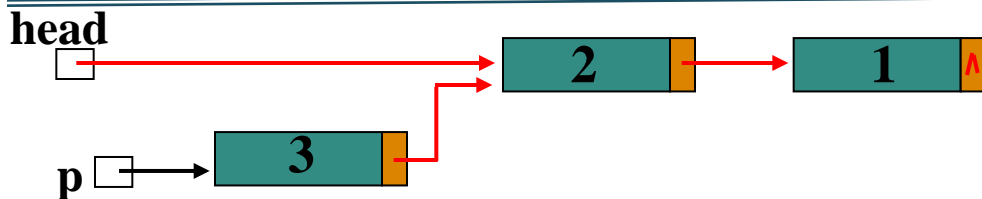
x

3



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head; ←
7. head=p;
8. scanf("%d", &x);
9. }
9. return(head);

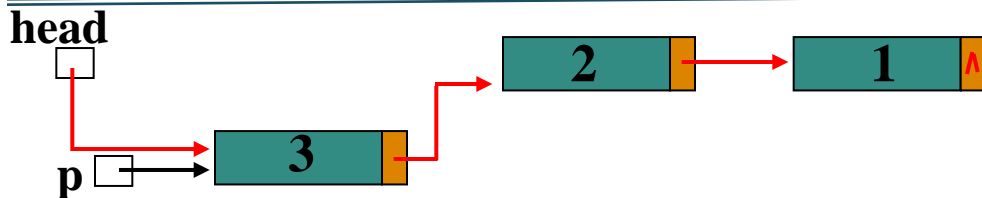


x
3



向前插入构造法示例，输入序列：1 2 3 0

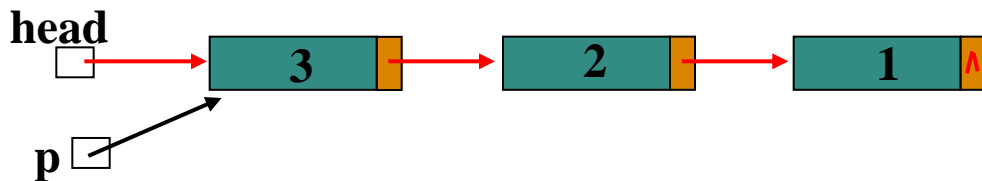
1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p; ←
8. scanf("%d", &x);
9. }
9. return(head);





向前插入构造法示例，输入序列：1 2 3 0


1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

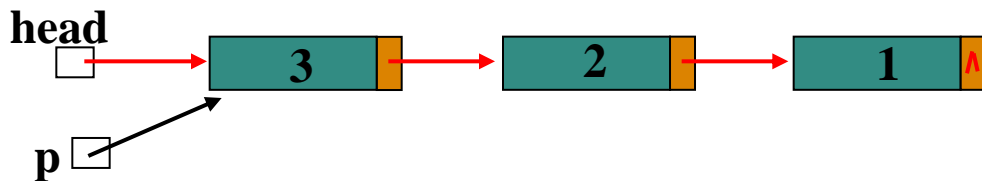


x
0



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0) 
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);



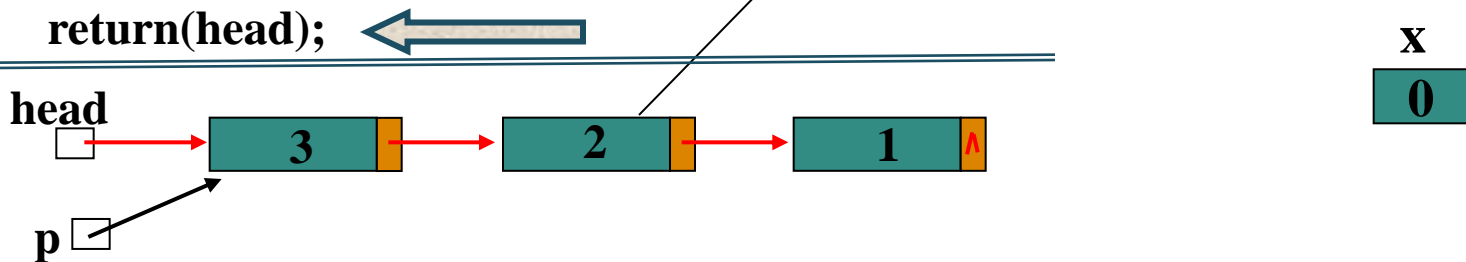
x
0



向前插入构造法示例，输入序列：1 2 3 0

1. head=NULL;
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. p->next=head;
7. head=p;
8. scanf("%d", &x);
9. return(head);

最终得到一个元素排列
次序与输入相反的链表





2. 链表的构造（向后插入法）

步骤1) 构造“空链表”

步骤2) 读入第一个元素值

步骤3) 当读入的元素值不是“结束标记”时，循环执行步骤4~7

步骤4) 申请一个新结点

步骤5) 将读入的元素值存入新结点的值域

步骤6) 将新结点“插在”链表的表尾处

步骤7) 读入“下一个”元素值，转步骤3

步骤8) 构造完毕，返回首指针



2. 链表的构造（向后插入法）

```
ptr creatlinked_B()
{   ptr head,last,p; int x;
    head=NULL;
    scanf("%d",&x);
    while(x!=End_elm)
    {
        p=(ptr)malloc(sizeof(snode)); //申请一个存储结点
        p->data=x;
        if (head==NULL) /空表，修改头指针和尾指针
        {   p->next=head; head=p; last=p;}
        else //非空表，插在表尾，修改尾指针
        {   last->next=p; p->next=NULL; last=p; }
        scanf("%d",&x);//读入下一个元素
    }
    return head;
}
```



2. 链表的构造（向后插入法）

```
ptr creatlinked_B()
{   ptr head,last,p; int x;
    head=last=(ptr)malloc(sizeof(snode)); last->next=NULL; //构造加头空链表
    scanf("%d",&x);
    while(x!=End_elm)
    {
        p=(ptr)malloc(sizeof(snode)); //申请一个存储结点
        p->data=x;
        last->next=p; //插在表尾
        p->next=NULL;
        last=p; //修改尾指针

        scanf("%d",&x); //读入下一个元素
    }
    p=head; head=head->next; free(p); //删除辅助头结点
    return head;
}
```



向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;** ←
2. scanf("%d", &x);
3. while (x!=0)
4. { p=(ptr)malloc(sizeof(snode));
5. p->data=x;
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. scanf("%d", &x);
10. **p=head;head=head->next;free(p); return(head);**

head



last




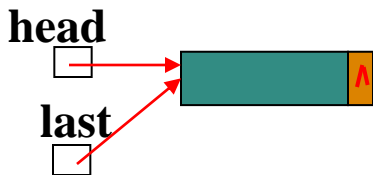
X





向后插入构造法示例，输入序列：1 2 3 0


1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);** 
3. **while (x!=0)**
4. **{ p=(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
- }**
10. **p=head;head=head->next;free(p); return(head);**

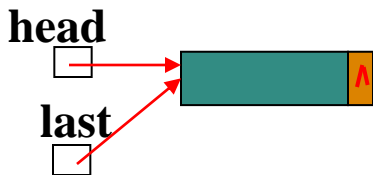


x
1



向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)** 
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

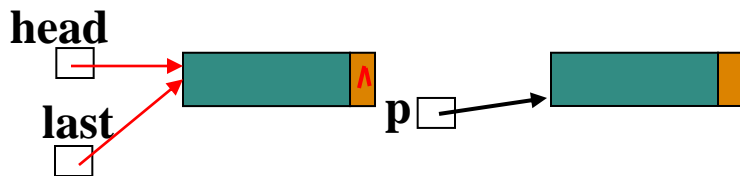


x
1



向后插入构造法示例，输入序列：1 2 3 0


1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));** ←
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

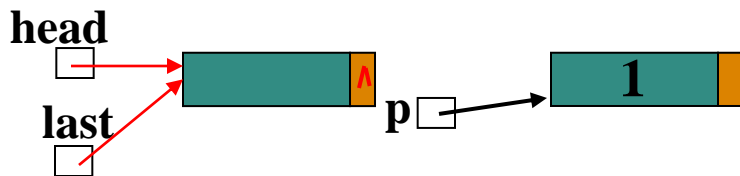


X
1



向后插入构造法示例，输入序列：1 2 3 0


1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;** 
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

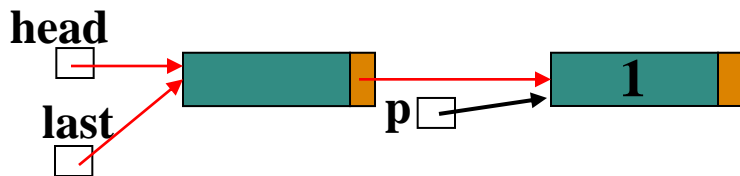


x
1



向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;** 
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

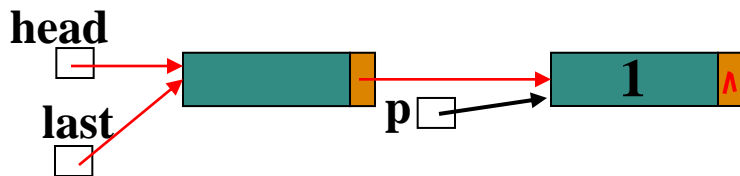


x
1



向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. **{ p=(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;** ←
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**



x
1

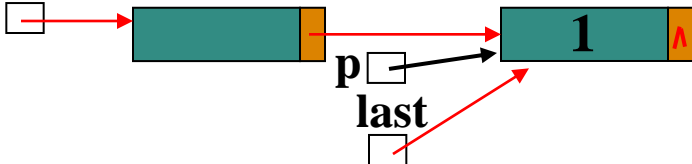


向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode));last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**



head

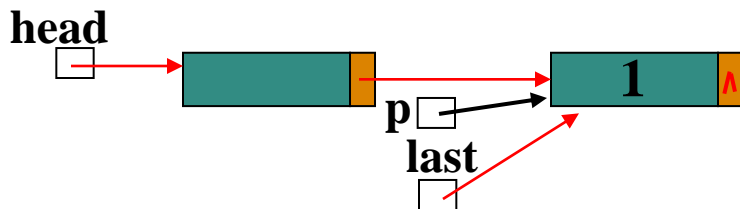


x
1



向后插入构造法示例，输入序列：1 2 3 0


1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);** ←
10. **p=head;head=head->next;free(p); return(head);**

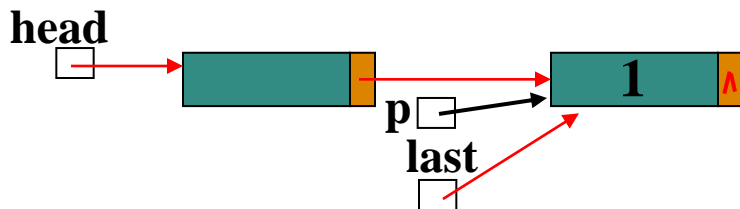


x
2



向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)** 
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**



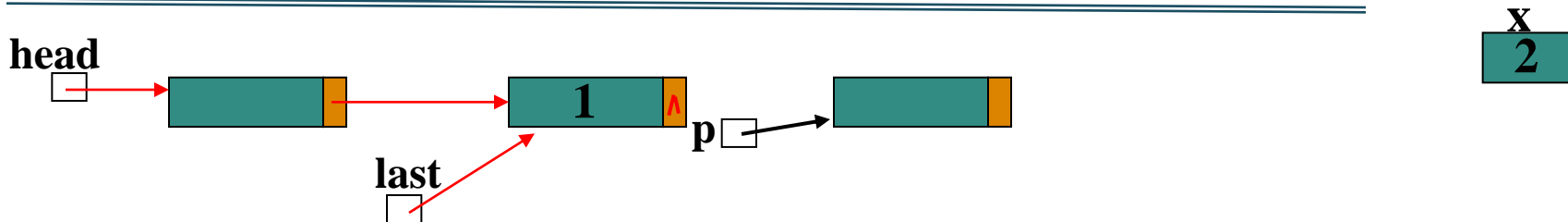
x

2



向后插入构造法示例，输入序列：1 2 3 0

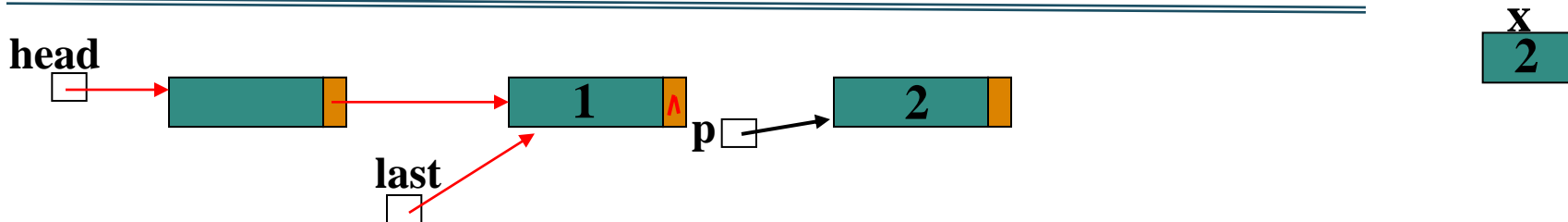
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));** ←
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**






向后插入构造法示例，输入序列：1 2 3 0

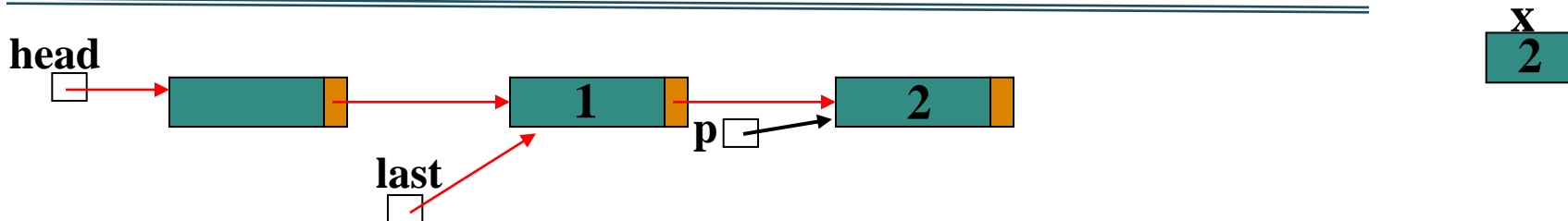
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;** ←
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

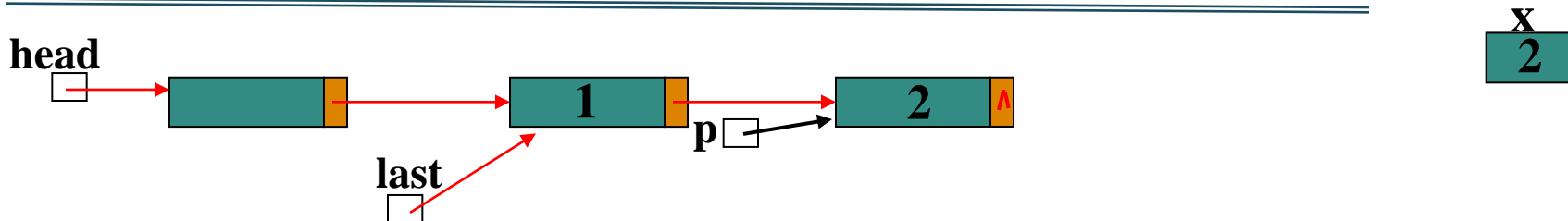
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;** 
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

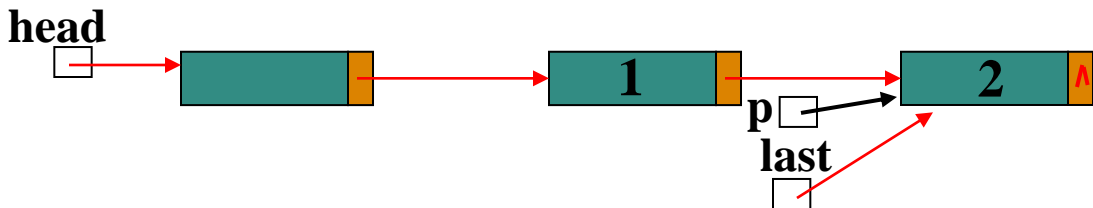
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;** ←
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

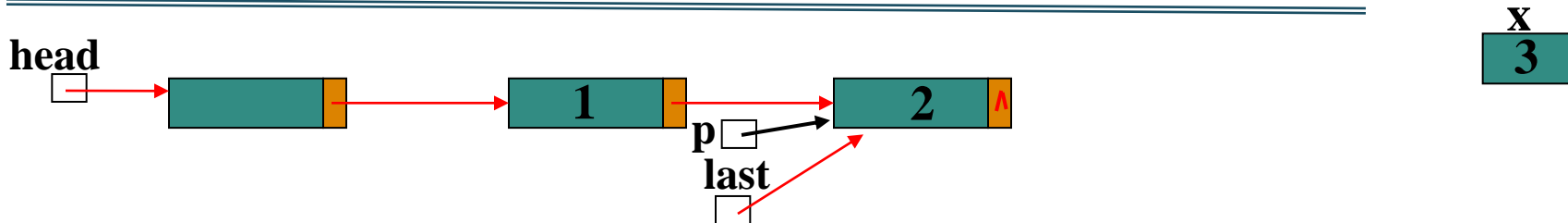


x
2




向后插入构造法示例，输入序列：1 2 3 0

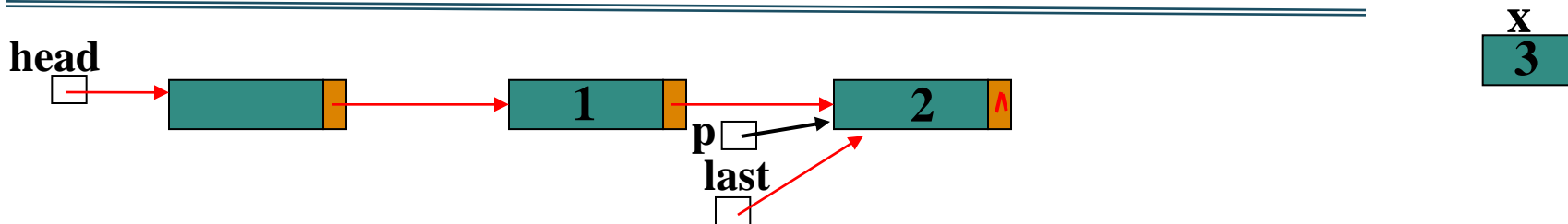
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);** ←
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

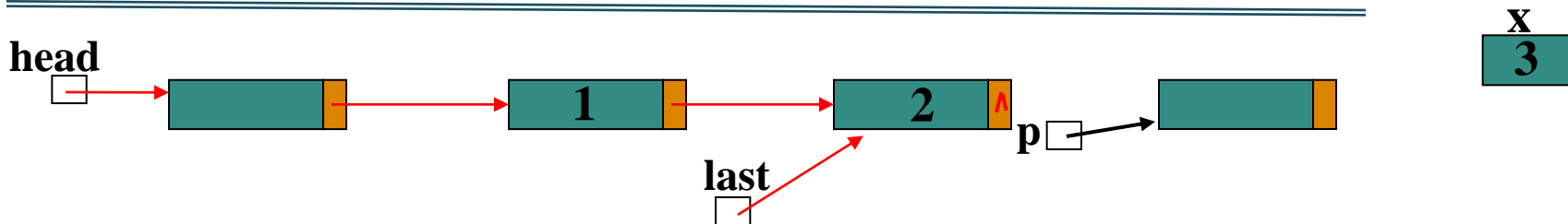
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)** 
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

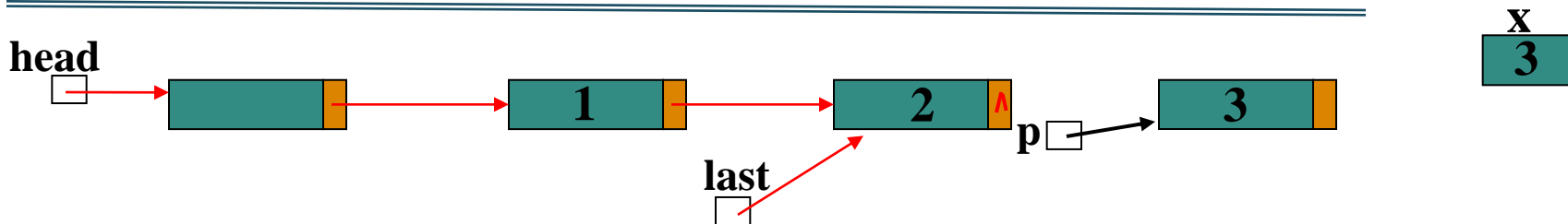
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));** ←
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**






向后插入构造法示例，输入序列：1 2 3 0

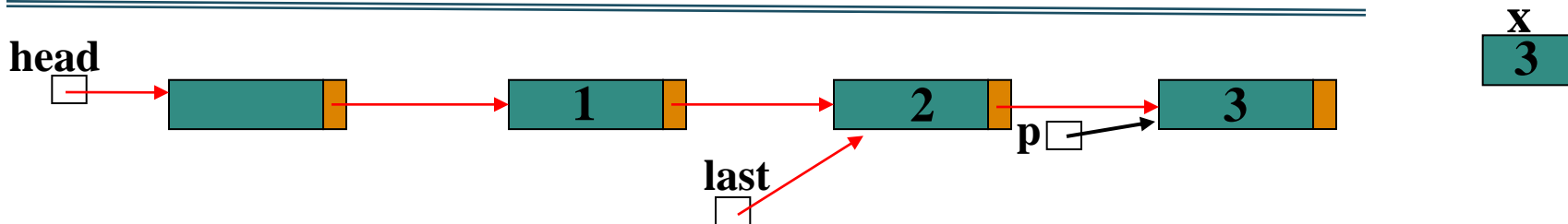
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;** ←
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

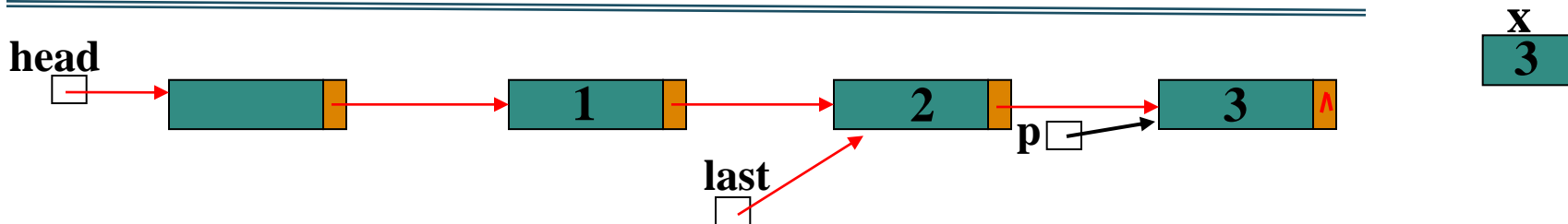
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;** 
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

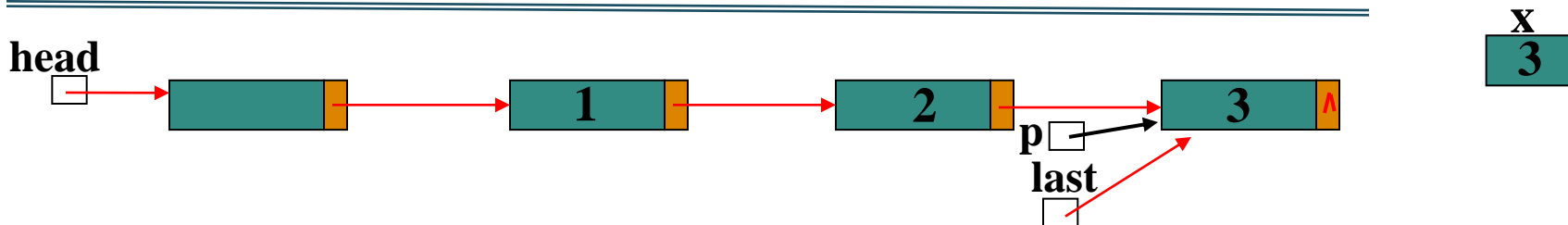
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;** ←
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

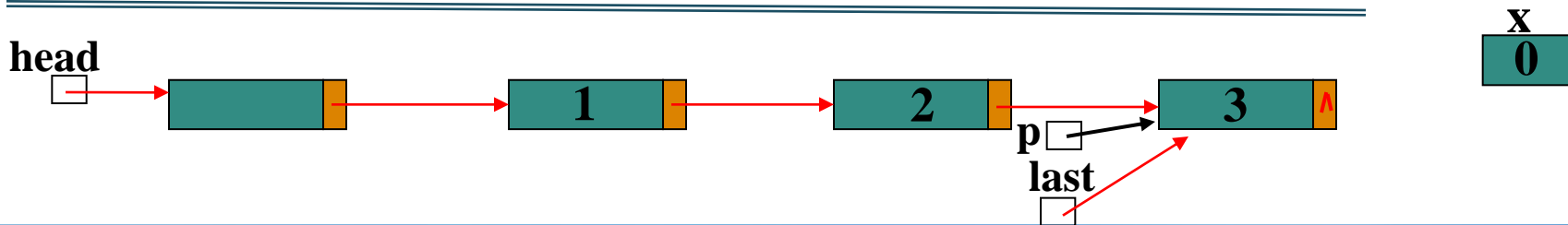
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**






向后插入构造法示例，输入序列：1 2 3 0

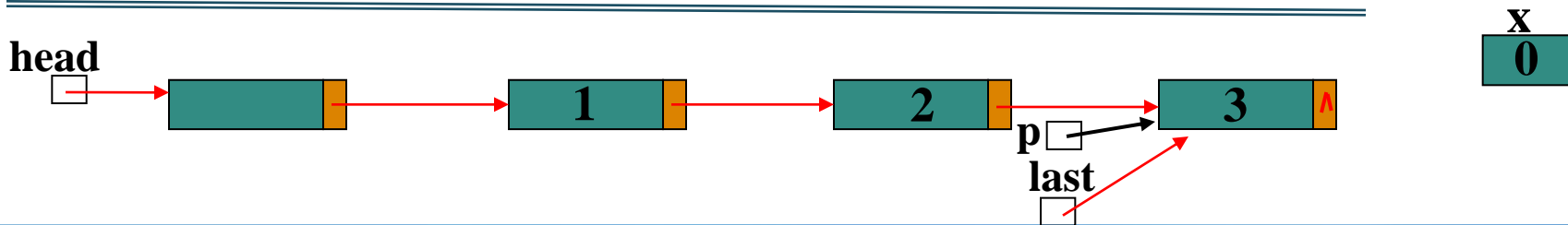
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);** ←
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

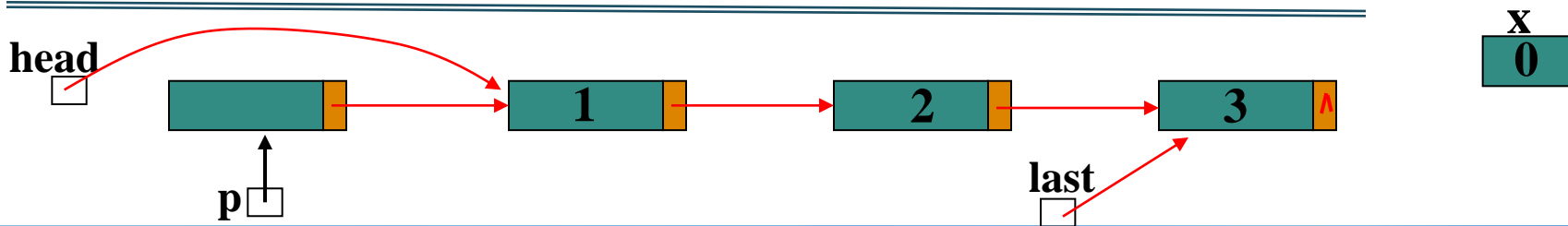
1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)** 
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**





向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. { **p =(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head;head=head->next;free(p); return(head);**

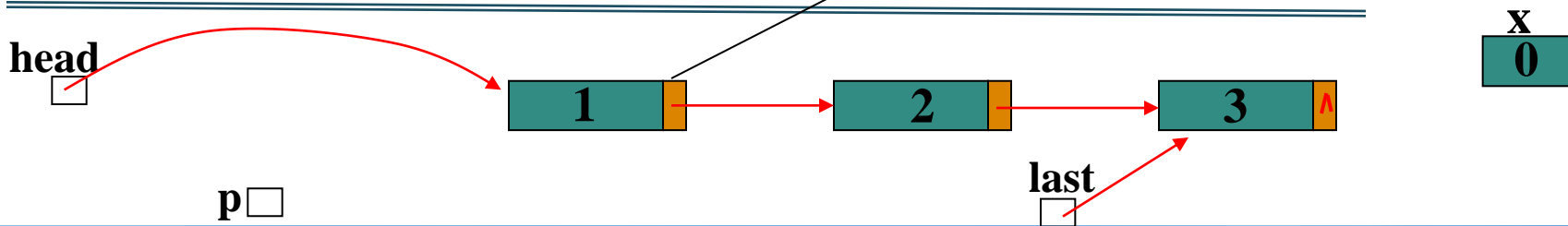




向后插入构造法示例，输入序列：1 2 3 0

1. **head=last=(ptr)malloc(sizeof(snode)); last->next=NULL;**
2. **scanf("%d", &x);**
3. **while (x!=0)**
4. **{ p=(ptr)malloc(sizeof(snode));**
5. **p->data=x;**
6. **last->next=p;**
7. **p->next=NULL;**
8. **last=p;**
9. **scanf("%d", &x);**
10. **p=head; head=head->next; free(p); return(head);**

最终得到一个元素排列
次序与输入相同的链表





3.向前插入和向后插入法的比较

1) 空链表的形式不同

向前插入法:

```
head=NULL; //简单空链表
```

向后插入法:

```
head=last=(ptr)malloc(sizeof(snode));  
last->next=NULL; //加头空链表
```




3.向前插入和向后插入法的比较

- 1) 空链表的形式不同
- 2) 插入部位不同

向前插入法:

```
p->next=head; head=p;//表头插入
```

向后插入法:

```
last->next=p; p->next=NULL;  
last=p;//表尾插入
```



3.向前插入和向后插入法的比较

- 1) 空链表的形式不同
- 2) 插入部位不同
- 3) 使用工作指针个数不同

向前插入法:

head, p

向后插入法:

head, last, p



3.向前插入和向后插入法的比较

- 1) 空链表的形式不同
- 2) 插入部位不同
- 3) 使用工作指针个数不同
- 4) 监督元的处理不同

向前插入法：

不用加监督元

向后插入法：

加监督元，算法不需分情况处理



3.向前插入和向后插入法的比较

- 1) 空链表的形式不同
- 2) 插入部位不同
- 3) 使用工作指针个数不同
- 4) 监督元的处理不同
- 5) 结点排列次序不同

向前插入法：

与输入次序相反

向后插入法：

与输入次序相同