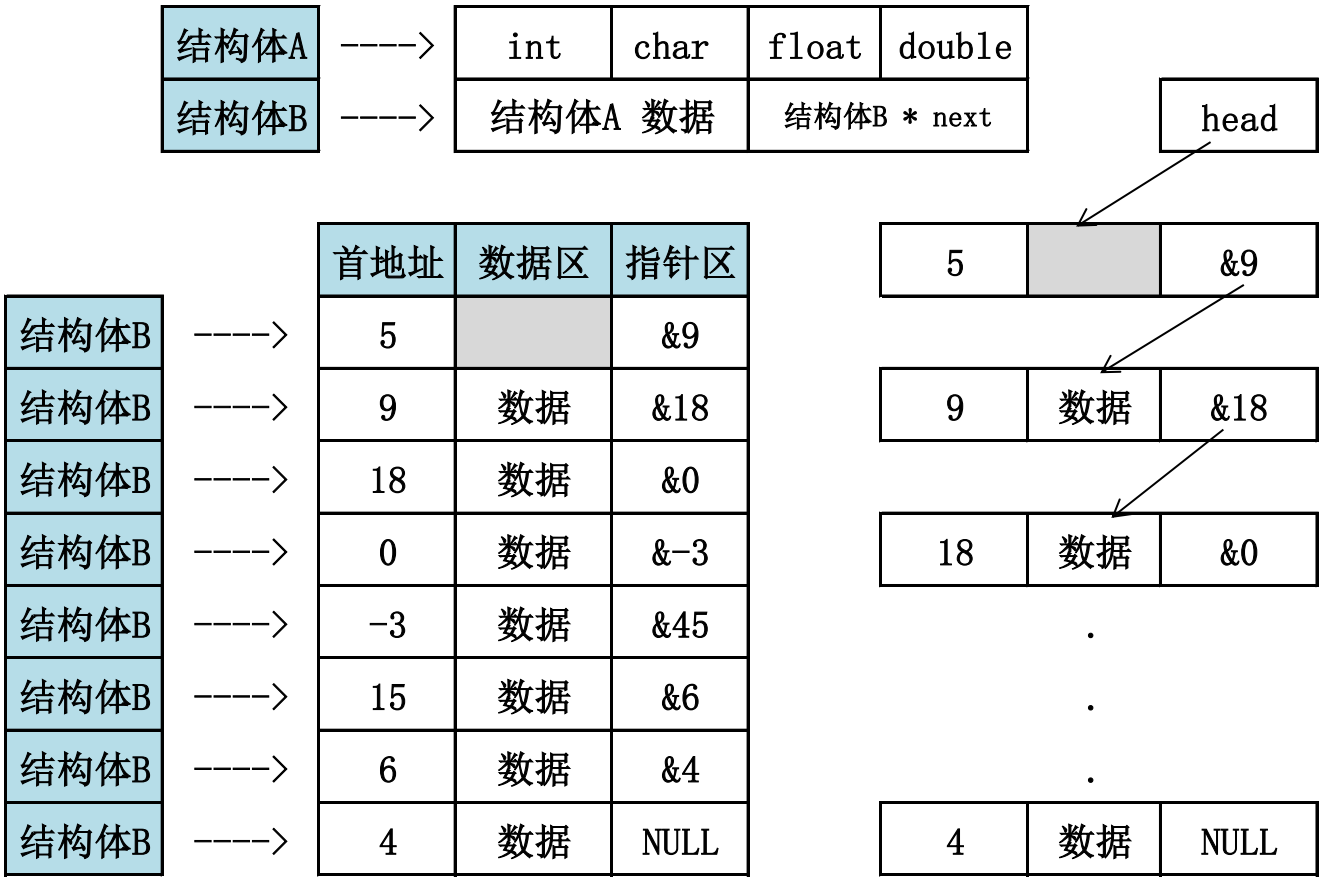
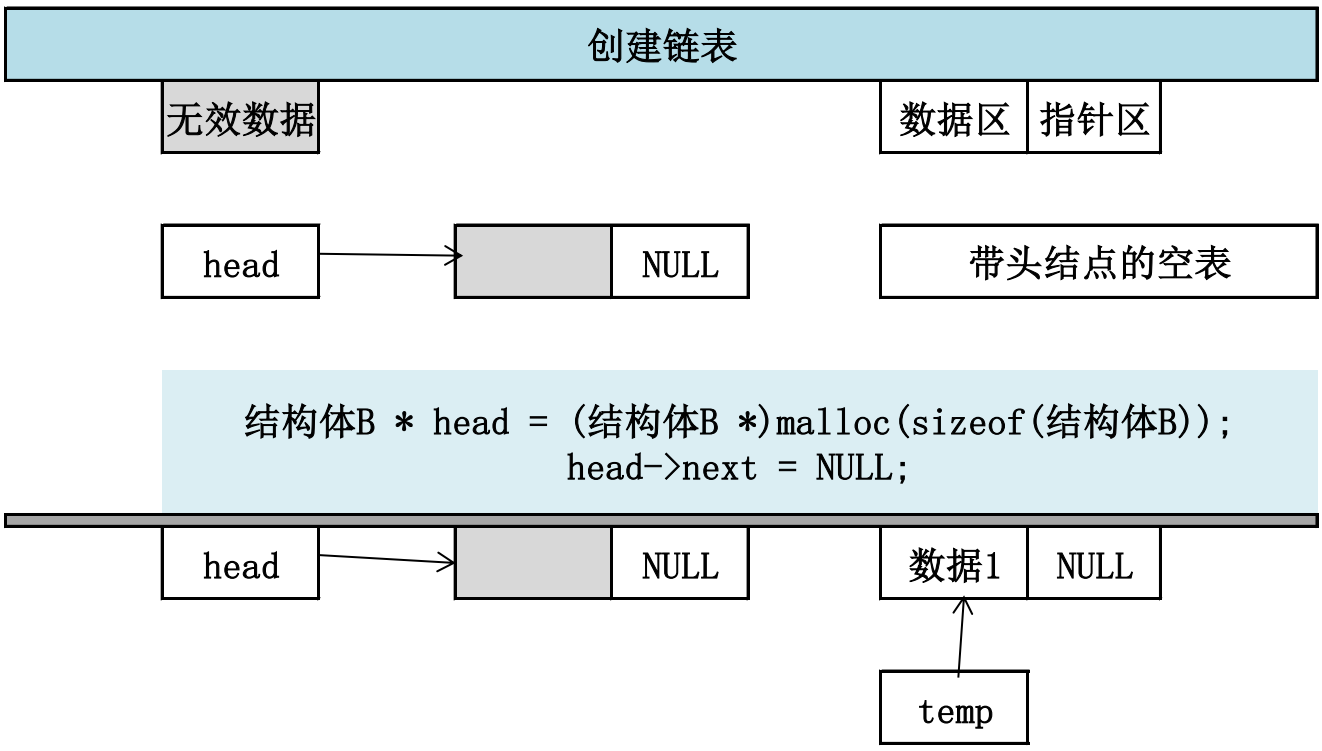
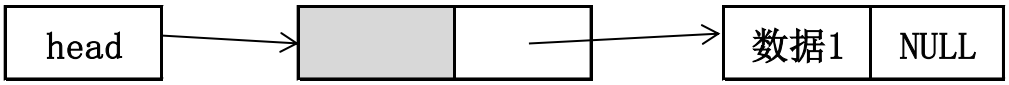
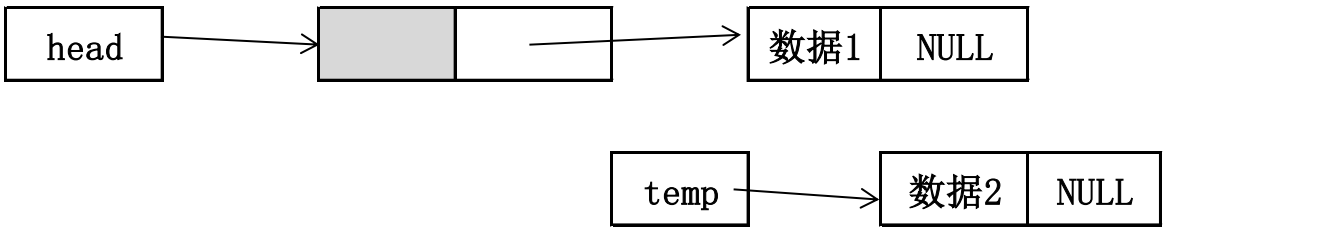

结构体A
结构体B
int
char
float
double
结构体A 数据
结构体B * next
head
首地址
数据区
指针区
结构体B
5
&9
9
数据
&18
18
&0
0
&-3
-3
&45
15
&6
6
&4
4
NULL

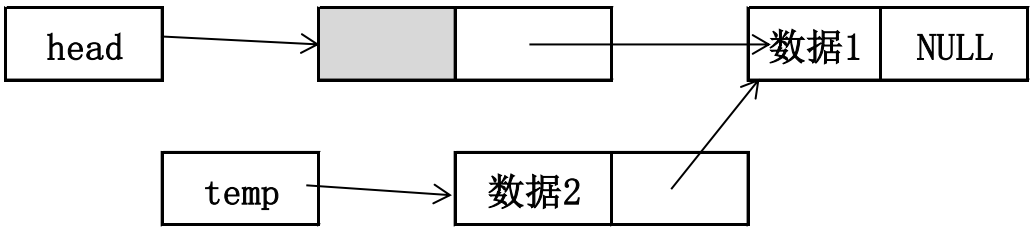

创建链表
无效数据
数据区
指针区
head
NULL
带头结点的空表
结构体B * head = (结构体B *)malloc(sizeof(结构体B));
head->next = NULL;
数据1
temp



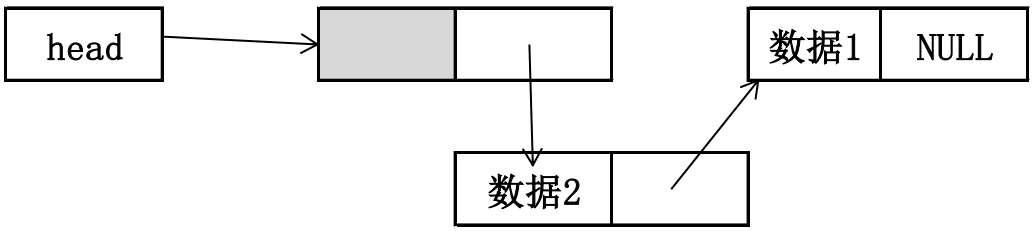

```
结构体B * temp = (结构体B *)malloc(sizeof(结构体B));
temp->数据 = 数据1;  temp->next = NULL;
temp->next = head->next
head->next = temp;
```




```
temp = (结构体B *)malloc(sizeof(结构体B));
temp->数据 = 数据2;  temp->next = NULL;
```

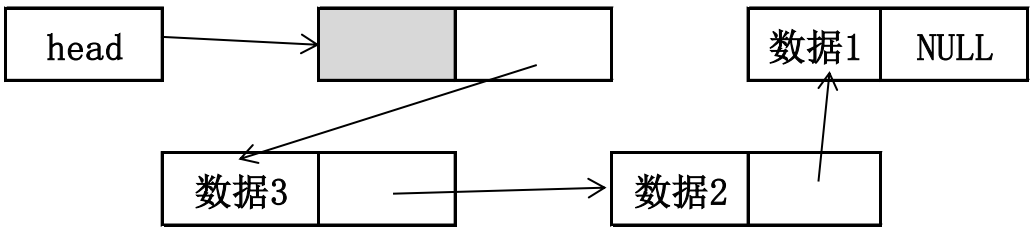



```
temp->next = head->next;
```

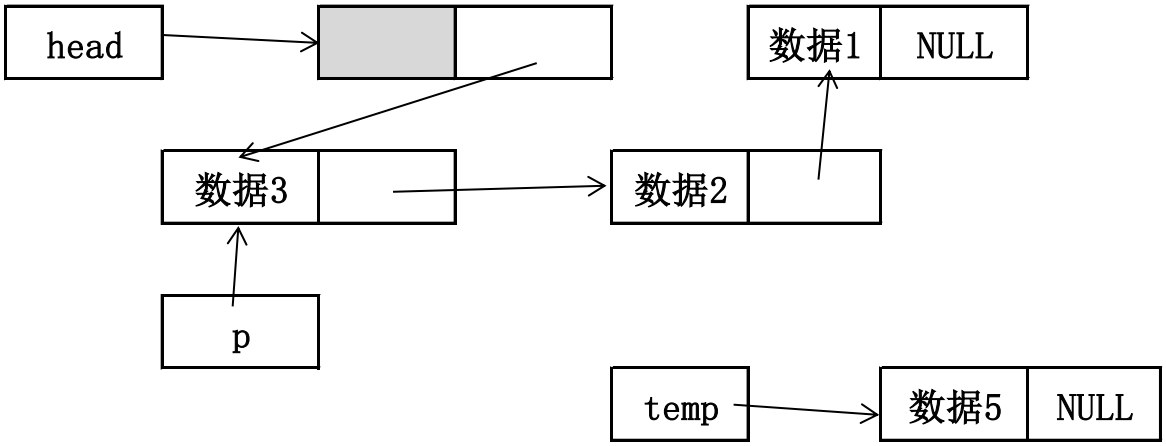



```
head->next = temp;
```

插入节点

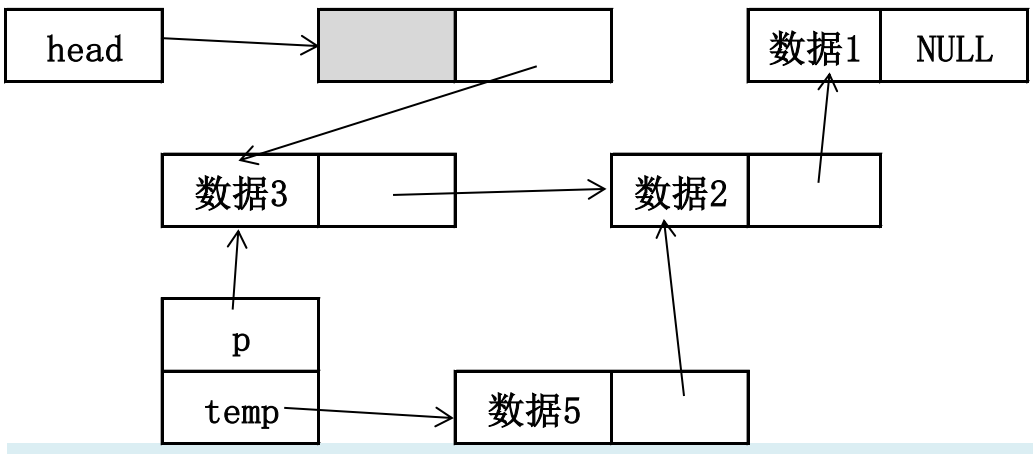


我们将创建一个节点，数据5 插入到数据2前面

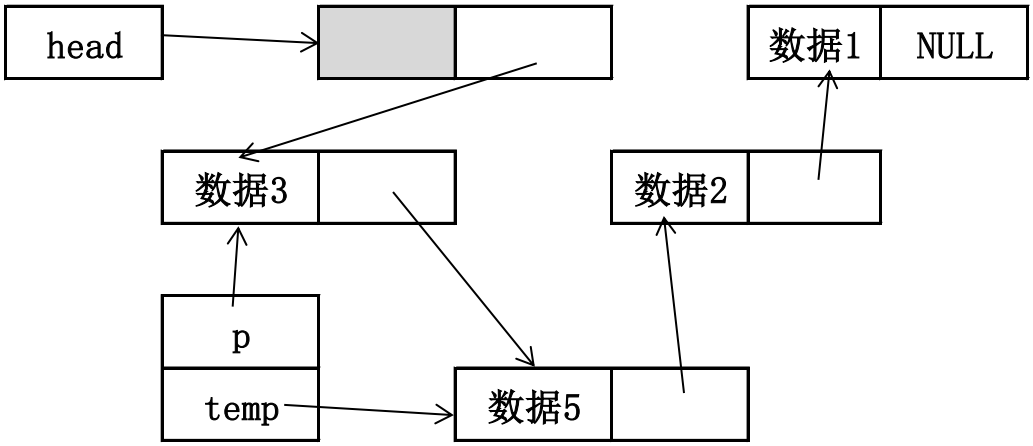


首先动态创建要插入的节点，在查找 数据2 节点的前一个节点

```
结构体B * p = head;
while( ( p->next != NULL) && (p->next->数据 != 数据2) )
{ p = p->next; }
```

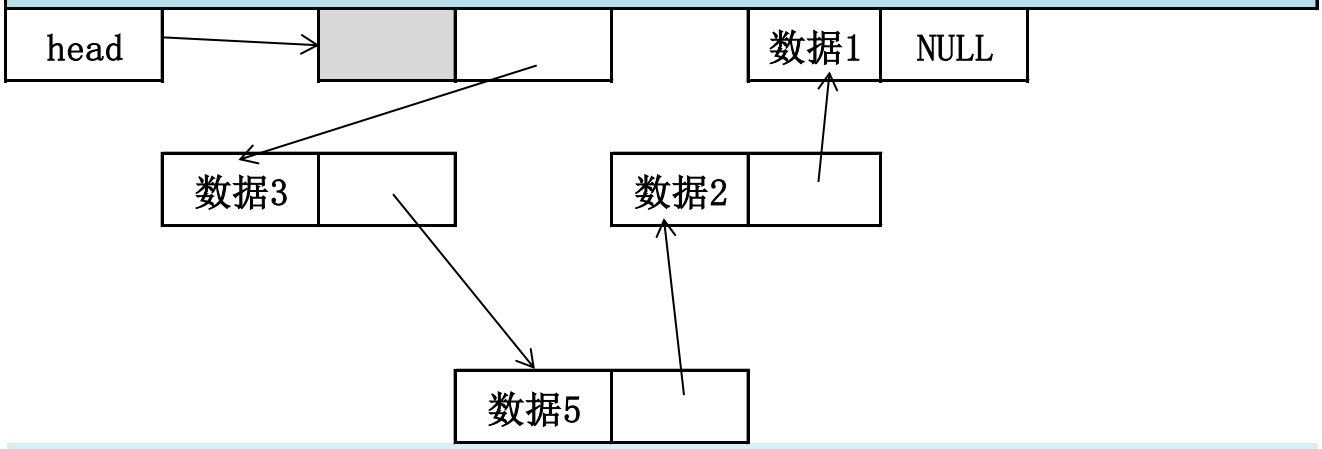



```
temp->next = p->next;
```

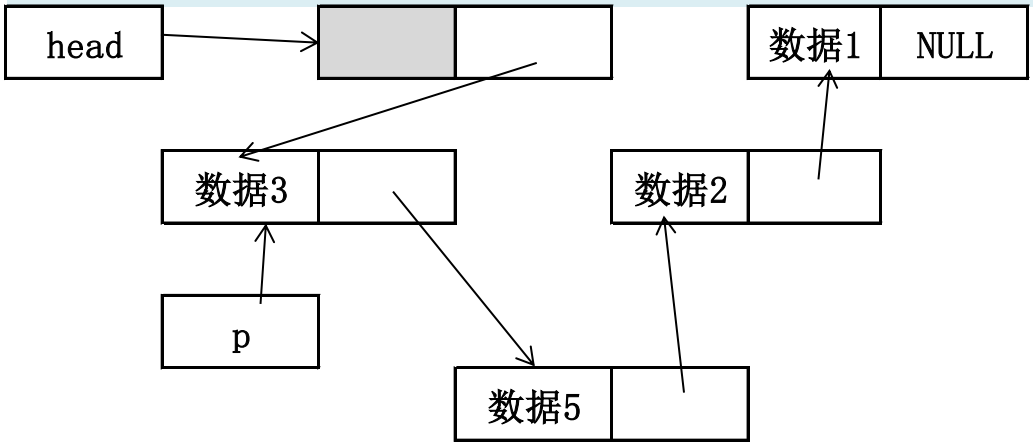



```
p->next = temp;
```

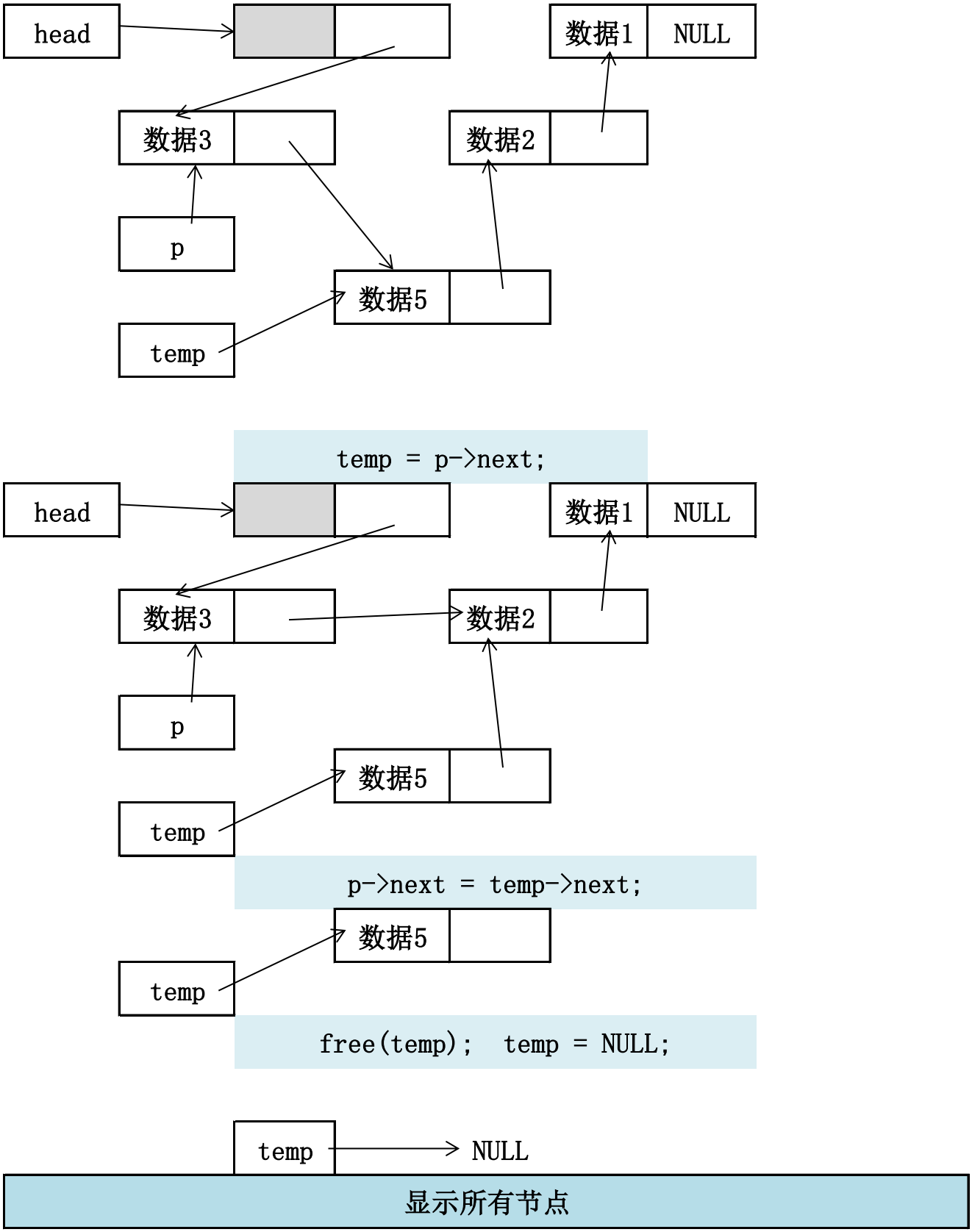
删除节点



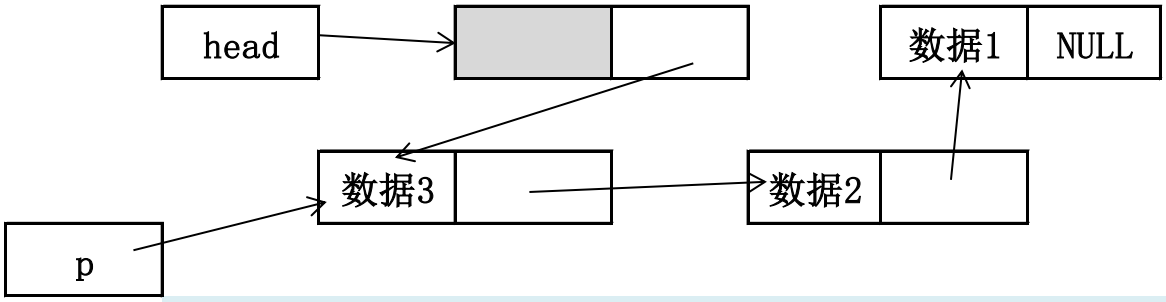
我们将 数据为 数据5的节点删除，首先用p指针找到 数据为数据5的节点的前一个节点




```
p = head;
while( ( p->next != NULL) && (p->next->数据 != 数据5) )
{ p = p->next; }
```



首先用临时节点 p 指向 头结点的下一个节点



然后从当前节点开始显示数据，移动到下一个节点，直到NULL为止

```
p = head->next;
while( p != NULL )
{
     printf("%d %c %f %lf\n", p->数据.int, p->数据.char, p->数据.float, p->数据.double);
     p = p->next;
}
```