阅读：13,140　　作者：解学武

双向链表基本操作（C语言实现）详解

上一节　　下一节

前面学习了如何创建一个双向链表，本节学习有关双向链表的一些基本操作，即如何在双向链表中添加、删除、查找或更改数据元素。

本节知识基于已熟练掌握双向链表创建过程的基础上，我们继续上节所创建的双向链表来学习本节内容，创建好的双向链表如图 1 所示：

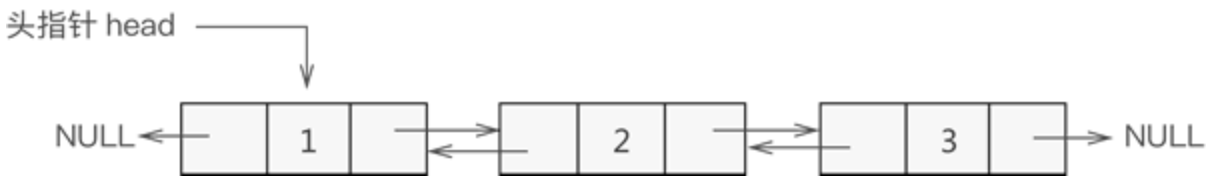

图 1 双向链表示意图

双向链表添加节点

根据数据添加到双向链表中的位置不同，可细分为以下 3 种情况：

1) 添加至表头

将新数据元素添加到表头，只需要将该元素与表头元素建立双层逻辑关系即可。

换句话说，假设新元素节点为 temp，表头节点为 head，则需要做以下 2 步操作即可：
1. temp->next=head; head->prior=temp;
2. 将 head 移至 temp，重新指向新的表头；

例如，将新元素 7 添加至双链表的表头，则实现过程如图 2 所示：

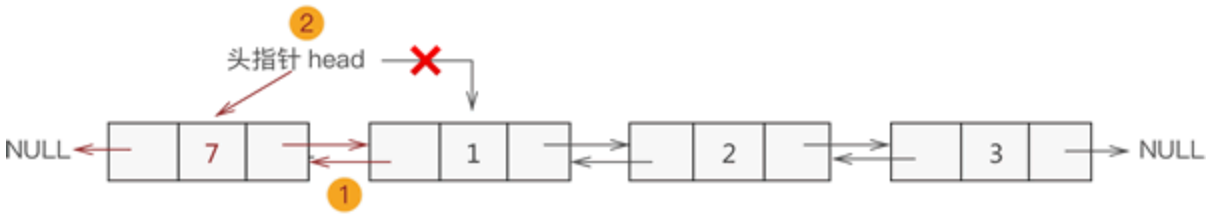

图 2 添加元素至双向链表的表头

2) 添加至表的中间位置

同单链表添加数据类似，双向链表中间位置添加数据需要经过以下 2 个步骤，如图 3 所示：
1. 新节点先与其直接后继节点建立双层逻辑关系；

2. 新节点的直接前驱节点与之建立双层逻辑关系；

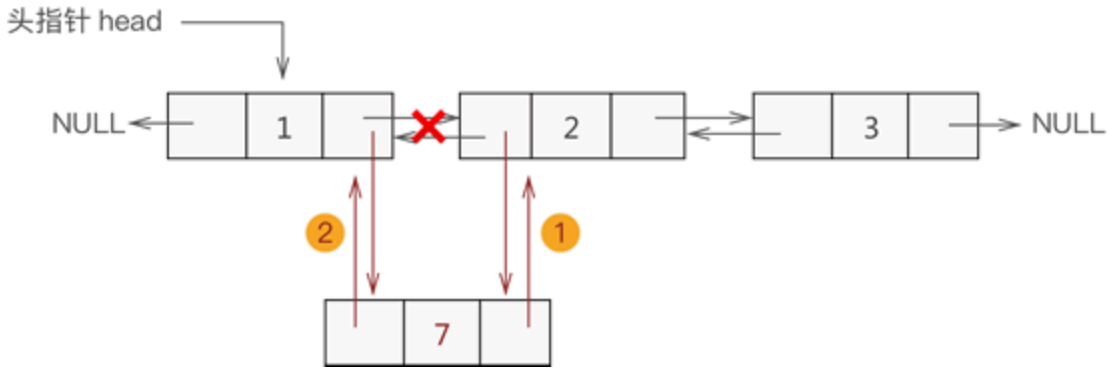

图 3 双向链表中间位置添加数据元素

3) 添加至表尾

与添加到表头是一个道理，实现过程如下（如图 4 所示）：

1. 找到双链表中最后一个节点；
2. 让新节点与最后一个节点进行双层逻辑关系；

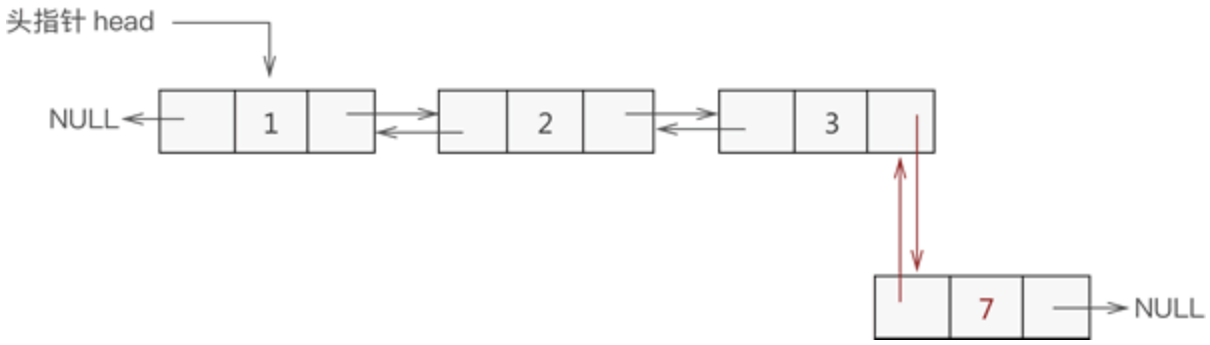

图 4 双向链表尾部添加数据元素

因此，我们可以试着编写双向链表添加数据的 C 语言代码，参考代码如下：

```
01.  //data 为要添加的新数据，add 为添加到链表中的位置
02.  line * insertLine(line * head, int data, int add) {
03.      //新建数据域为data的结点
04.      line * temp = (line*)malloc(sizeof(line));
05.      temp->data = data;
06.      temp->prior = NULL;
07.      temp->next = NULL;
08.      //插入到链表头，要特殊考虑
09.      if (add == 1) {
10.          temp->next = head;
11.          head->prior = temp;
12.          head = temp;
13.      }
14.      else {
15.          int i = 0;
16.          line * body = head;
17.          //找到要插入位置的前一个结点
```

```
            for (i = 1; i < add - 1; i++) {
                body = body->next;
                if (body == NULL) {
                    printf("插入位置有误\n");
                    break;
                }
            }
            if (body) {
                //判断条件为真，说明插入位置为链表尾
                if (body->next == NULL) {
                    body->next = temp;
                    temp->prior = body;
                }
                else {
                    body->next->prior = temp;
                    temp->next = body->next;
                    body->next = temp;
                    temp->prior = body;
                }
            }
        }
        return head;
    }
```

双向链表删除节点

双链表删除结点时，只需遍历链表找到要删除的结点，然后将该节点从表中摘除即可。

例如，从图 1 基础上删除元素 2 的操作过程如图 5 所示：

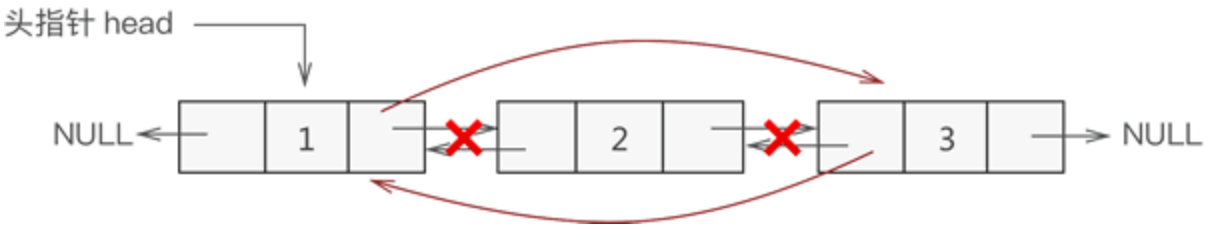

图 5 双链表删除元素操作示意图

双向链表删除节点的 C 语言实现代码如下：

```
//删除结点的函数，data为要删除结点的数据域的值
line * delLine(line * head, int data) {
    line * temp = head;
    //遍历链表
    while (temp) {
```

```
        //判断当前结点中数据域和data是否相等，若相等，摘除该结点
        if (temp->data == data) {
            temp->prior->next = temp->next;
            temp->next->prior = temp->prior;
            free(temp);
            return head;
        }
        temp = temp->next;
    }
    printf("链表中无该数据元素\n");
    return head;
}
```

双向链表查找节点

通常，双向链表同单链表一样，都仅有一个头指针。因此，双链表查找指定元素的实现同单链表类似，都是从表头依次遍历表中元素。

C 语言实现代码为：

```
//head为原双链表，elem表示被查找元素
int selectElem(line * head, int elem) {
    //新建一个指针t，初始化为头指针 head
    line * t = head;
    int i = 1;
    while (t) {
        if (t->data == elem) {
            return i;
        }
        i++;
        t = t->next;
    }
    //程序执行至此处，表示查找失败
    return -1;
}
```

双向链表更改节点

更改双链表中指定结点数据域的操作是在查找的基础上完成的。实现过程是：通过遍历找到存储有该数据元素的结点，直接更改其数据域即可。

实现此操作的 C 语言实现代码如下：

```
//更新函数，其中，add 表示更改结点在双链表中的位置，newElem 为新数据的值
line *amendElem(line * p, int add, int newElem) {
```

```
    int i = 0;
    line * temp = p;
    //遍历到被删除结点
    for (i = 1; i < add; i++) {
        temp = temp->next;
        if (temp == NULL) {
            printf("更改位置有误！\n");
            break;
        }
    }
    if (temp) {
        temp->data = newElem;
    }
    return p;
}
```

总结

这里给出双链表中对数据进行 "增删查改" 操作的完整实现代码：

```
#include <stdio.h>
#include <stdlib.h>
typedef struct line {
    struct line * prior;
    int data;
    struct line * next;
}line;
//双链表的创建
line* initLine(line * head);
//双链表插入元素，add表示插入位置
line * insertLine(line * head, int data, int add);
//双链表删除指定元素
line * delLine(line * head, int data);
//双链表中查找指定元素
int selectElem(line * head, int elem);
//双链表中更改指定位置节点中存储的数据，add表示更改位置
line *amendElem(line * p, int add, int newElem);
//输出双链表的实现函数
void display(line * head);
int main() {
    line * head = NULL;
    //创建双链表
    printf("初始链表为：\n");
    head = initLine(head);
    display(head);
    //在表中第 3 的位置插入元素 7
```

```
27.      printf("在表中第 3 的位置插入新元素 7: \n");
28.      head = insertLine(head, 7, 3);
29.      display(head);
30.      //表中删除元素 2
31.      printf("删除元素 2: \n");
32.      head = delLine(head, 2);
33.      display(head);
34.
35.      printf("元素 3 的位置是: %d\n", selectElem(head, 3));
36.      //表中第 3 个节点中的数据改为存储 6
37.      printf("将第 3 个节点存储的数据改为 6: \n");
38.      head = amendElem(head, 3, 6);
39.      display(head);
40.      return 0;
41.  }
42.  line* initLine(line * head) {
43.      int i = 0;
44.      line * list = NULL;
45.      head = (line*)malloc(sizeof(line));
46.      head->prior = NULL;
47.      head->next = NULL;
48.      head->data = 1;
49.      list = head;
50.      for (i = 2; i <= 3; i++) {
51.          line * body = (line*)malloc(sizeof(line));
52.          body->prior = NULL;
53.          body->next = NULL;
54.          body->data = i;
55.
56.          list->next = body;
57.          body->prior = list;
58.          list = list->next;
59.      }
60.      return head;
61.  }
62.  line * insertLine(line * head, int data, int add) {
63.      //新建数据域为data的结点
64.      line * temp = (line*)malloc(sizeof(line));
65.      temp->data = data;
66.      temp->prior = NULL;
67.      temp->next = NULL;
68.      //插入到链表头，要特殊考虑
69.      if (add == 1) {
70.          temp->next = head;
71.          head->prior = temp;
72.          head = temp;
73.      }
```

```
74.     else {
75.         int i = 0;
76.         line * body = head;
77.         //找到要插入位置的前一个结点
78.         for (i = 1; i < add - 1; i++) {
79.             body = body->next;
80.             if (body == NULL) {
81.                 printf("插入位置有误\n");
82.                 break;
83.             }
84.         }
85.         if (body) {
86.             //判断条件为真，说明插入位置为链表尾
87.             if (body->next == NULL) {
88.                 body->next = temp;
89.                 temp->prior = body;
90.             }
91.             else {
92.                 body->next->prior = temp;
93.                 temp->next = body->next;
94.                 body->next = temp;
95.                 temp->prior = body;
96.             }
97.         }
98.     }
99.     return head;
100. }
101. line * delLine(line * head, int data) {
102.     line * temp = head;
103.     //遍历链表
104.     while (temp) {
105.         //判断当前结点中数据域和data是否相等，若相等，摘除该结点
106.         if (temp->data == data) {
107.             temp->prior->next = temp->next;
108.             temp->next->prior = temp->prior;
109.             free(temp);
110.             return head;
111.         }
112.         temp = temp->next;
113.     }
114.     printf("链表中无该数据元素\n");
115.     return head;
116. }
117. //head为原双链表，elem表示被查找元素
118. int selectElem(line * head, int elem) {
119.     //新建一个指针t，初始化为头指针 head
120.     line * t = head;
```

```
121.      int i = 1;
122.      while (t) {
123.          if (t->data == elem) {
124.              return i;
125.          }
126.          i++;
127.          t = t->next;
128.      }
129.      //程序执行至此处，表示查找失败
130.      return -1;
131.  }
132.  //更新函数，其中，add 表示更改结点在双链表中的位置，newElem 为新数据的值
133.  line *amendElem(line * p, int add, int newElem) {
134.      int i = 0;
135.      line * temp = p;
136.      //遍历到被删除结点
137.      for (i = 1; i < add; i++) {
138.          temp = temp->next;
139.          if (temp == NULL) {
140.              printf("更改位置有误！\n");
141.              break;
142.          }
143.      }
144.      if (temp) {
145.          temp->data = newElem;
146.      }
147.      return p;
148.  }
149.  //输出链表的功能函数
150.  void display(line * head) {
151.      line * temp = head;
152.      while (temp) {
153.          if (temp->next == NULL) {
154.              printf("%d\n", temp->data);
155.          }
156.          else {
157.              printf("%d->", temp->data);
158.          }
159.          temp = temp->next;
160.      }
161.  }
```

程序执行结果为：

```
初始链表为：
1->2->3
在表中第 3 的位置插入新元素 7：
```

1->2->7->3
删除元素 2：
1->7->3
元素 3 的位置是：3
将第 3 个节点存储的数据改为 6：
1->7->6

上一节 下一节