

# 双向循环链表及创建（C语言）详解

我们知道，单[链表](#)通过首尾连接可以构成单向循环链表，如[图 1](#) 所示：



图 1 单向循环链表示意图

同样，[双向链表](#)也可以进行首尾连接，构成双向循环链表。如图 2 所示：



图 2 双向循环链表示意图

当问题中涉及到需要“循环往复”地遍历表中数据时，就需要使用双向循环链表。例如，前面章节我们对约瑟夫环问题进行了研究，其实约瑟夫环问题有多种玩法，每次顺时针报数后，下一轮可以逆时针报数，然后再顺时针.....一直到剩下最后一个人。解决这个问题就需要使用双向循环链表结构。

## 双向循环链表的创建

创建双向循环链表，只需在创建完成双向链表的基础上，将其首尾节点进行双向连接即可。

C 语言实现代码如下：

```
01. //创建双向循环链表
02. line* initLine(line * head) {
03.     int i = 0;
04.     line * list = NULL;
05.     head = (line*)malloc(sizeof(line));
06.     head->prior = NULL;
07.     head->next = NULL;
08.     head->data = 1;
09.     list = head;
10.     for (i = 2; i <= 3; i++) {
11.         line * body = (line*)malloc(sizeof(line));
12.         body->prior = NULL;
```

```

13.         body->next = NULL;
14.         body->data = i;
15.
16.         list->next = body;
17.         body->prior = list;
18.         list = list->next;
19.     }
20.     //通过以上代码，已经创建好双线链表，接下来将链表的首尾节点进行双向连接
21.     list->next = head;
22.     head->prior = list;
23.     return head;
24. }

```

通过向 main 函数中调用 initLine 函数，就可以成功创建一个存储有 {1,2,3} 数据的双向循环链表，其完整的 C 语言实现代码为：

```

01. #include <stdio.h>
02. #include <stdlib.h>
03. typedef struct line {
04.     struct line * prior;
05.     int data;
06.     struct line * next;
07. }line;
08.
09. line* initLine(line * head);
10. void display(line * head);
11. int main() {
12.     line * head = NULL;
13.     head = initLine(head);
14.     display(head);
15.     return 0;
16. }
17. //创建双向循环链表
18. line* initLine(line * head) {
19.     int i = 0;
20.     line * list = NULL;
21.     head = (line*)malloc(sizeof(line));
22.     head->prior = NULL;
23.     head->next = NULL;
24.     head->data = 1;
25.     list = head;
26.     for (i = 2; i <= 3; i++) {
27.         line * body = (line*)malloc(sizeof(line));
28.         body->prior = NULL;
29.         body->next = NULL;
30.         body->data = i;

```

```

31.
32.         list->next = body;
33.         body->prior = list;
34.         list = list->next;
35.     }
36.     //通过以上代码，已经创建好双线链表，接下来将链表的首尾节点进行双向连接
37.     list->next = head;
38.     head->prior = list;
39.     return head;
40. }
41.
42. //输出链表的功能函数
43. void display(line * head) {
44.     line * temp = head;
45.     //由于是循环链表，所以当遍历指针temp指向的下一个节点是head时，证明此时已经循环至链表的最后一个节点
46.     while (temp->next != head) {
47.         if (temp->next == NULL) {
48.             printf("%d\n", temp->data);
49.         }
50.         else {
51.             printf("%d<->", temp->data);
52.         }
53.         temp = temp->next;
54.     }
55.     //输出循环链表中最后一个节点的值
56.     printf("%d", temp->data);
57. }

```

程序输出结果如下：

```
1<->2<->3
```

**联系方式**    **购买教程（带答疑）**