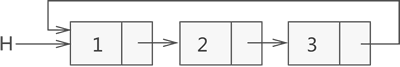
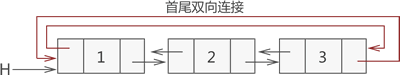
双向循环链表及创建（C语言）详解

我们知道，单[链表](http://data.biancheng.net/view/160.html)通过首尾连接可以构成单向循环链表，如[图](http://data.biancheng.net/view/200.html) 1 所示：

  
图 1 单向循环链表示意图

同样，[双向链表](http://data.biancheng.net/view/166.html)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：

1. //创建双向循环链表
2. line\* initLine(line \* head) {
3. int i = 0;
4. line \* list = NULL;
5. head = (line\*)malloc(**sizeof**(line));
6. head->prior = NULL;
7. head->next = NULL;
8. head->data = 1;
9. 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. list->next = body;
16. body->prior = list;
17. list = list->next;
18. }
19. //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
20. list->next = head;
21. head->prior = list;
22. **return** head;
23. }

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：

1. #include <stdio.h>
2. #include <stdlib.h>
3. **typedef** **struct** line {
4. **struct** line \* prior;
5. int data;
6. **struct** line \* next;
7. }line;
8. line\* initLine(line \* head);
9. void display(line \* head);
10. int main() {
11. line \* head = NULL;
12. head = initLine(head);
13. display(head);
14. **return** 0;
15. }
16. //创建双向循环链表
17. line\* initLine(line \* head) {
18. int i = 0;
19. line \* list = NULL;
20. head = (line\*)malloc(**sizeof**(line));
21. head->prior = NULL;
22. head->next = NULL;
23. head->data = 1;
24. list = head;
25. **for** (i = 2; i <= 3; i++) {
26. line \* body = (line\*)malloc(**sizeof**(line));
27. body->prior = NULL;
28. body->next = NULL;
29. body->data = i;
30. list->next = body;
31. body->prior = list;
32. list = list->next;
33. }
34. //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
35. list->next = head;
36. head->prior = list;
37. **return** head;
38. }
39. //输出链表的功能函数
40. void display(line \* head) {
41. line \* temp = head;
42. //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
43. **while** (temp->next != head) {
44. **if** (temp->next == NULL) {
45. printf("%d\n", temp->data);
46. }
47. **else** {
48. printf("%d<->", temp->data);
49. }
50. temp = temp->next;
51. }
52. //输出循环链表中最后一个节点的值
53. printf("%d", temp->data);
54. }

程序输出结果如下：

1<->2<->3