# C语言实现双向循环链表

循环链表和双向链表的区别，之后再给大家介绍双向循环链表的具体实现。

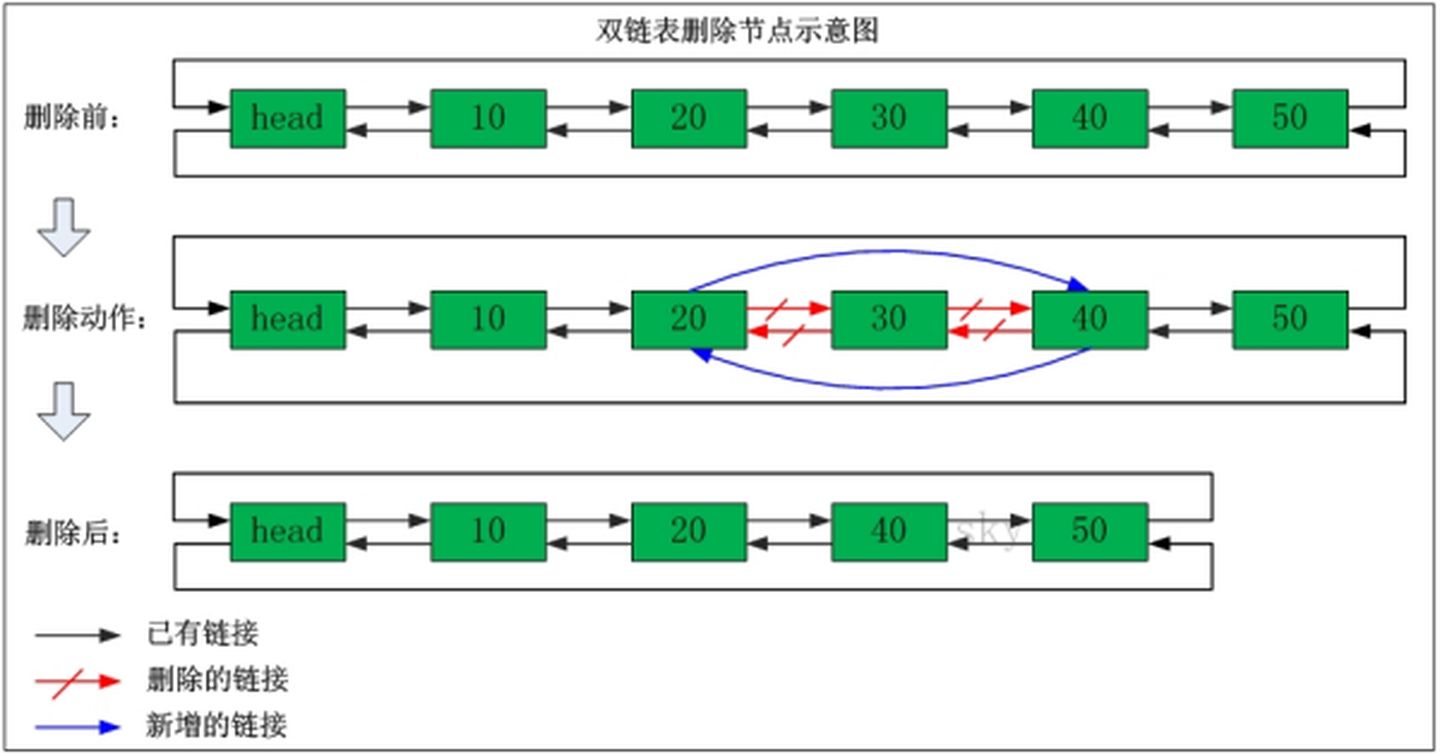
**循环链表和双向链表的区别**

1、最后一个结点指针指向不同

在建立一个循环链表时，必须使其最后一个结点的指针指向表头结点，而不是像双向链表那样置为NULL。此种情况还用于在最后一个结点后插入一个新的结点。

2、判断链域值不同

在判断是否到表尾时，是判断该结点链域的值是否是表头结点，当链域值等于表头指针时，说明已到表尾。而非像单链表那样判断链域值是否为NULL。



3、访问方式：

循环链表：可以从任何一个结点开始，顺序向后访问到达任意结点

双向链表：可以从任何结点开始任意向前向后双向访问

4、操作：

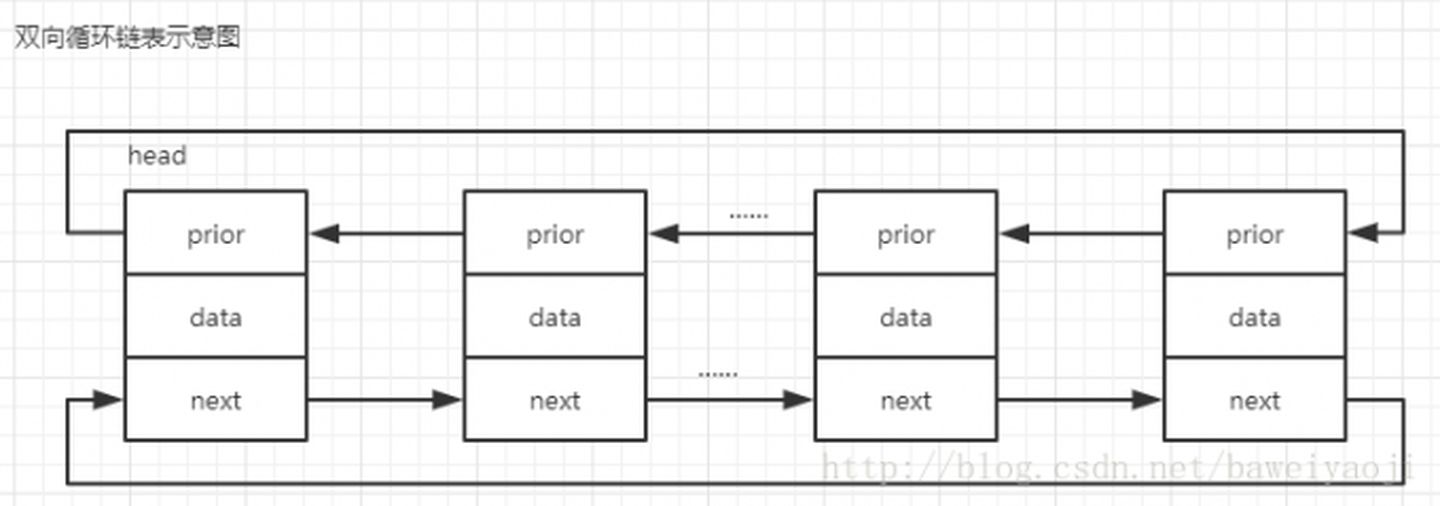
循环链表：只能在当前结点后插入和删除

双链表：可以在当前结点前面或者后面插入，可以删除前趋和后继（包括结点自己）

5、存储：循环链表存储密度大于双链表

**双向循环链表的具体实现**

双向循环链表：最后一个节点的next指向head，而head的prior指向最后一个节点，构成一个环。



由上图可以看出，双向循环链表的结点结构与双向链表的结构是一样的，都是含有三项：前驱指针prior,数据项data,后驱指针next,因此双向循环链表结点结构用C语言实现如下：

**struct** doubleCircularLinkedList

{

**struct** doubleCircularLinkedList**\*** prior;*//结点的前驱指针*

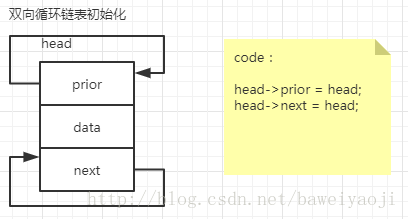
**int** data;*//结点的数据项*

**struct** doubleCircularLinkedList**\*** next;*//结点的后继指针*

};

**双向循环链表的初始化**

只有一个头节点head，就让prior和next都指向自己，形成一个环。



初始化头结点代码实现：

**struct** doubleCircularLinkedList**\*** **createList**()

{

*//创建一个头结点，数据差异化当作表头*

**struct** doubleCircularLinkedList**\*** headNode **=** (**struct** doubleCircularLinkedList**\***)malloc(**sizeof**(**struct** doubleCircularLinkedList));

*//循环链表，所以初始化头指针，尾指针都是指向自身的，data数据域不做初始化*

headNode**->**prior **=** headNode;*//头结点指向自身*

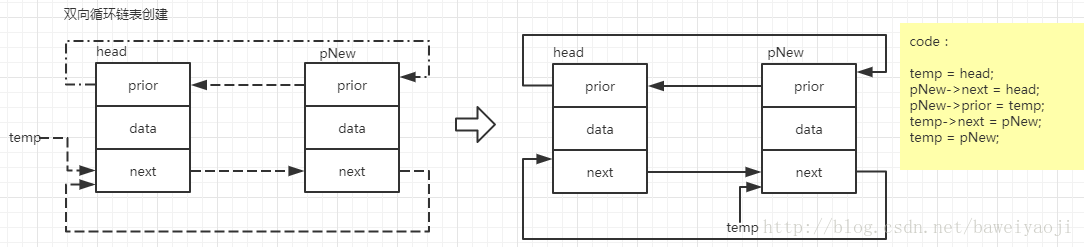
headNode**->**next **=** headNode;*//尾结点指向自身*

**return** headNode;

}

**创建一个新的结点**

与单向循环链表类似的，只是多了一个prior要考虑，为插入做准备。



**struct** doubleCircularLinkedList**\*** **createNode**(**int** data)

{

*//动态申请内存malloc+free c语言的特点*

**struct** doubleCircularLinkedList**\*** newNode **=** (**struct** doubleCircularLinkedList**\***)malloc(**sizeof**(**struct** doubleCircularLinkedList));

*//创建结点过程相当于初始化过程*

newNode**->**data **=** data;*//传入data数值初始化数据域*

newNode**->**prior **=** NULL;*//初始化头结点为null*

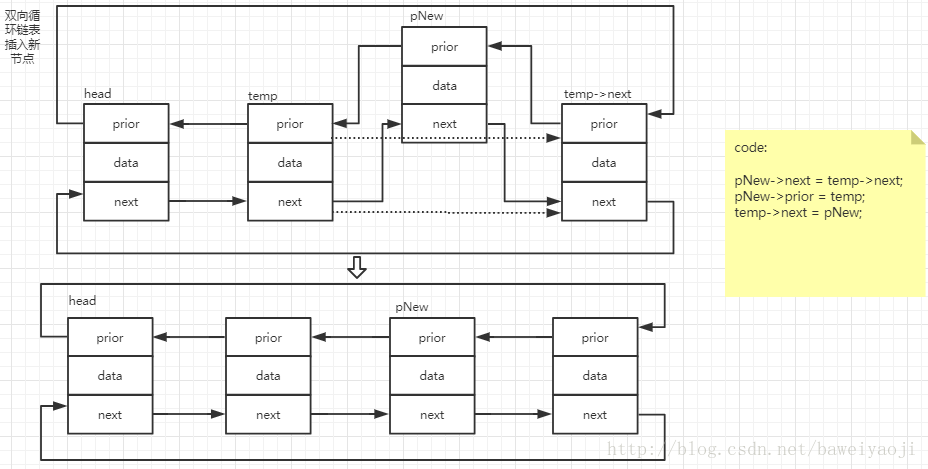
newNode**->**next **=** NULL;*//初始化尾结点为null*

**return** newNode;

}

**插入新的元素**

与单向循环链表类似，只是多了一个prior要考虑。这里就不需判断插入的位置是不是在最后了，已经构成一个环。



**表头插入实现**

**void** **insertNodeByHead**(**struct** doubleCircularLinkedList**\*** headNode,**int** data)

{

*//创建一个新的结点，调用创建新结点的函数*

**struct** doubleCircularLinkedList**\*** newNode **=** createNode(data);

*//修改四个指针变量*

newNode**->**prior **=** headNode;

newNode**->**next **=** headNode**->**next;

headNode**->**next**->**prior**=**newNode;

headNode**->**next **=** newNode;

}

**表尾插入实现**

在表尾插入，比表头插入更容易出错，大家多加注意！首先找到尾部最后一个元素，然后再进行插入操作

**void** **insertNodeBynext**(**struct** doubleCircularLinkedList**\*** headNode,**int** data)

{

**struct** doubleCircularLinkedList**\*** newNode **=** createNode(data);

*//首先找到最后一个结点的位置*

**struct** doubleCircularLinkedList**\*** lastNode **=** headNode;

**while**(lastNode**->**next **!=** headNode)

{

lastNode **=** lastNode**->**next;

}

*//找到之后调整四个指针*

headNode**->**prior **=** newNode;

newNode**->**next **=** headNode;

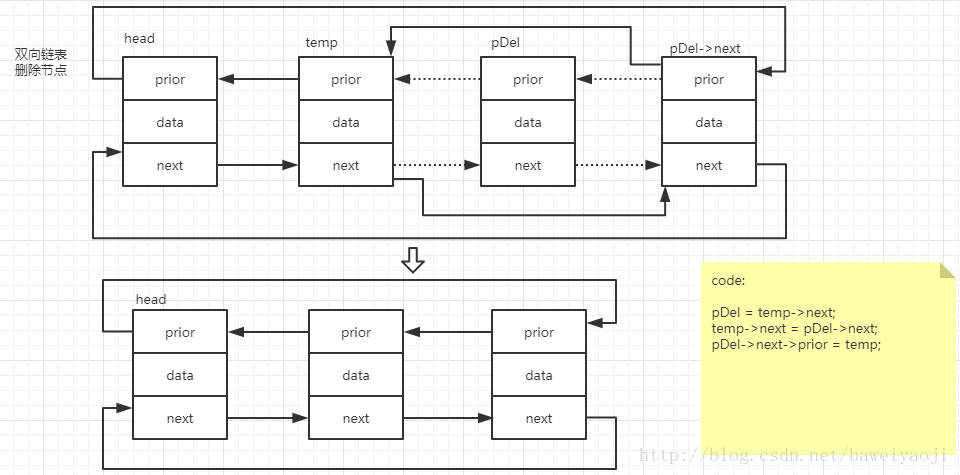
lastNode**->**next **=** newNode;

newNode**->**prior **=** lastNode;

}

**删除指定元素**

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，删除之后不要忘了释放空间哟！



**void** **SpecifyLocationToDelete**(**struct** doubleCircularLinkedList**\*** headNode,**int** posData)

{

**struct** doubleCircularLinkedList**\*** posNode **=** headNode**->**next;*//指定结点指针*

**struct** doubleCircularLinkedList**\*** posNodeprior **=** headNode;*//指定结点前一个结点的指针*

*//找到指定位置*

**while**(posNode**->**data **!=** posData)

{

posNodeprior **=** posNode;

posNode **=** posNodeprior**->**next;

*//如果没有找到特殊处理*

**if**(posNode**->**next **==** headNode)

{

printf("不存在指定位置，无法删除！\n");

**return**;

}

}

posNodeprior**->**next **=** posNode**->**next;

posNode**->**next**->**prior**=**posNodeprior;

free(posNode);*//删除之后，释放空间*

}

**查找指定元素**

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。

**void** **searchSpecifiedElement**(**struct** doubleCircularLinkedList**\*** headNode,**int** posData)

{

**struct** doubleCircularLinkedList**\*** posNode **=** headNode**->**next;*//指定结点指针*

**struct** doubleCircularLinkedList**\*** posNodeprior **=** headNode;*//指定结点前一个结点的指针*

*//找到指定位置*

**while**(posNode**->**data **!=** posData)

{

posNodeprior **=** posNode;

posNode **=** posNodeprior**->**next;

*//如果没有找到特殊处理*

**if**(posNode**->**next **==** headNode)

{

printf("不存在元素！\n");

**return** ;

}

}

printf("该元素存在！\n");

}

**查找指定元素**

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。

**void** **modifySpecifiedElement**(**struct** doubleCircularLinkedList**\*** headNode,**int** posData,**int** elem)

{

**struct** doubleCircularLinkedList**\*** posNode **=** headNode**->**next;*//指定结点指针*

**struct** doubleCircularLinkedList**\*** posNodeprior **=** headNode;*//指定结点前一个结点的指针*

*//找到指定位置*

**while**(posNode**->**data **!=** posData)

{

posNodeprior **=** posNode;

posNode **=** posNodeprior**->**next;

*//如果没有找到特殊处理*

**if**(posNode**->**next **==** headNode)

{

printf("不存在元素！\n");

}

}

posNode**->**data **=** elem;

printf("修改成功！\n");

}

**打印数据**

**void** **printList**(**struct** doubleCircularLinkedList**\*** headNode)

{

*//从第二个结点开始打印，表头不含数据*

*//也可以通过前指针进行打印，只需将next改为prior即可*

**struct** doubleCircularLinkedList**\*** pMove **=** headNode**->**next;

**while**(pMove **!=** headNode)*//如果pMove->next != headNode这样写，最后一个结点是不会打印的*

{

printf("%d ",pMove**->**data);

pMove **=** pMove**->**next;*//移动指针*

}

printf("\n");

}

参考博文： [https://blog.csdn.net/baweiyaoji/article/details/76071053](https://link.zhihu.com/?target=https%3A//blog.csdn.net/baweiyaoji/article/details/76071053)

**坚持就是胜利!**以上就是本次给大家分享的C语言实现双向循环链表，完整的代码已经push到了githubs上面（[传送门](https://link.zhihu.com/?target=https%3A//github.com/xjh199923/Data-structures-in-multiple-languages/blob/master/C/%25E5%258F%258C%25E5%2590%2591%25E5%25BE%25AA%25E7%258E%25AF%25E9%2593%25BE%25E8%25A1%25A8/doubleCircularLinkedList.c)），欢迎各位clone，如果觉得还不错的话，欢迎Star! 如果有哪里有问题，欢迎大家在下面评论区留言，我及时修改更正，坚持就是胜利！