

导航

博客园  
首页  
新随笔  
联系  
订阅XML  
管理

< 2018年9月 >						
日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

统计

随笔 - 278  
文章 - 0  
评论 - 1149  
引用 - 0

搜索

常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

随笔分类 (275)

Android(7)  
Android NDK编程(9)  
Android 系统层(5)  
Android 应用层(46)  
Computer Culture(2)  
Java(111)  
Linux/Ubuntu(5)  
UML(5)  
Windows(1)  
设计模式(1)  
数据结构\_算法(79)  
索引(4)

数组、单链表和双链表介绍 以及 双向链表的C/C++/Java实现

概要

线性表是一种线性结构，它是具有相同类型的n(n≥0)个数据元素组成的有限序列。本章先介绍线性表的几个基本组成部分：数组、单向链表、双向链表；随后给出双向链表的C、C++和Java三种语言的实现。内容分

数组  
单向链表  
双向链表

- 1. C实现双链表
- 2. C++实现双链表
- 3. Java实现双链表

转载请注明出处：<http://www.cnblogs.com/skywang12345/p/3561803.html>

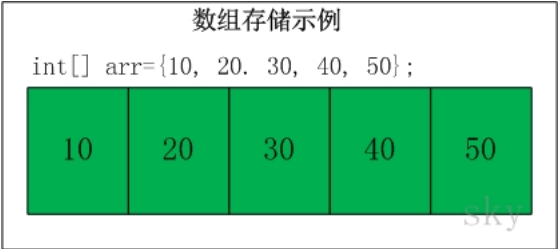
更多内容

[数据结构与算法系列\\_目录](#)

数 组

数组有上界和下界，数组的元素在上下界内是连续的。

存储10,20,30,40,50的数组的示意图如下：



数组的特点是：**数据是连续的；随机访问速度快。**  
数组中稍微复杂一点的是多维数组和动态数组。对于C语言而言，多维数组本质上也是通过一维数组实现的。至于动态数组，是指数组的容量能动态增长的数组；对于C语言而言，若要提供动态数组，需要手动实现；提供了Vector；对于Java而言，Collection集合中提供了ArrayList和Vector。

单 向 链 表

单向链表(单链表)是链表的一种，它由节点组成，每个节点都包含下一个节点的指针。

1. Re:Java 集合系列11之-Hashtable  
详细介绍(源码解析)和使用示例  
为什么是 two one three 顺序??  
不应该是three two one 顺序打印吗?  
--我是你粉丝啊

2. Re:Java 集合系列12之-TreeMap  
详细介绍(源码解析)和使用示例  
受教了, 真厉害  
--猜不透、迷

3. Re:Java hashCode() 和 equals()  
的若干问题解答  
准去的说应该是: == 和 equals 的区别  
前者是比较两个引用地址中的值是否  
相等, 后者是比较两个引用地址是否相  
等吧?  
--宅毛豆

4. Re:[转载] 散列表(Hash Table) 从  
理论到实用 (下)  
厉害厉害, 虽然还是没看明白  
--迷路的猫

5. Re:Java 集合系列10之-HashMap  
详细介绍(源码解析)和使用示例  
后面注释不会执行的地方, 是会执行  
的, 你理解错了.// putForNullKey()的  
作用是将"key为null"键值对添加到  
table[0]位置179 private V  
putForNul.....  
--凤\_鸣

#### 阅读排行榜

1. 红黑树(一)之-原理和算法详细介绍  
(214359)

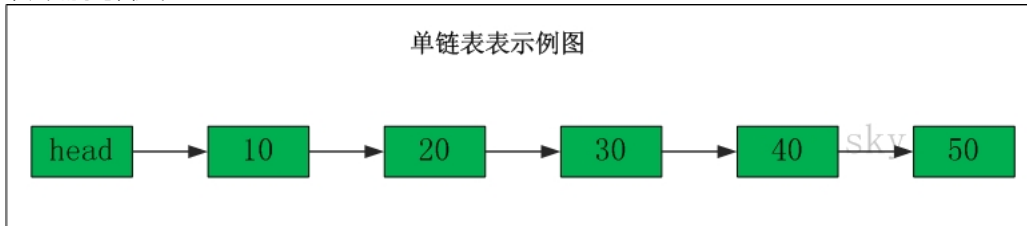
2. Java 集合系列10之-HashMap详细  
介绍(源码解析)和使用示例(126698)

3. Java 集合系列03之-ArrayList详细  
介绍(源码解析)和使用示例(97743)

4. 数据结构与算法系列 目录(87722)

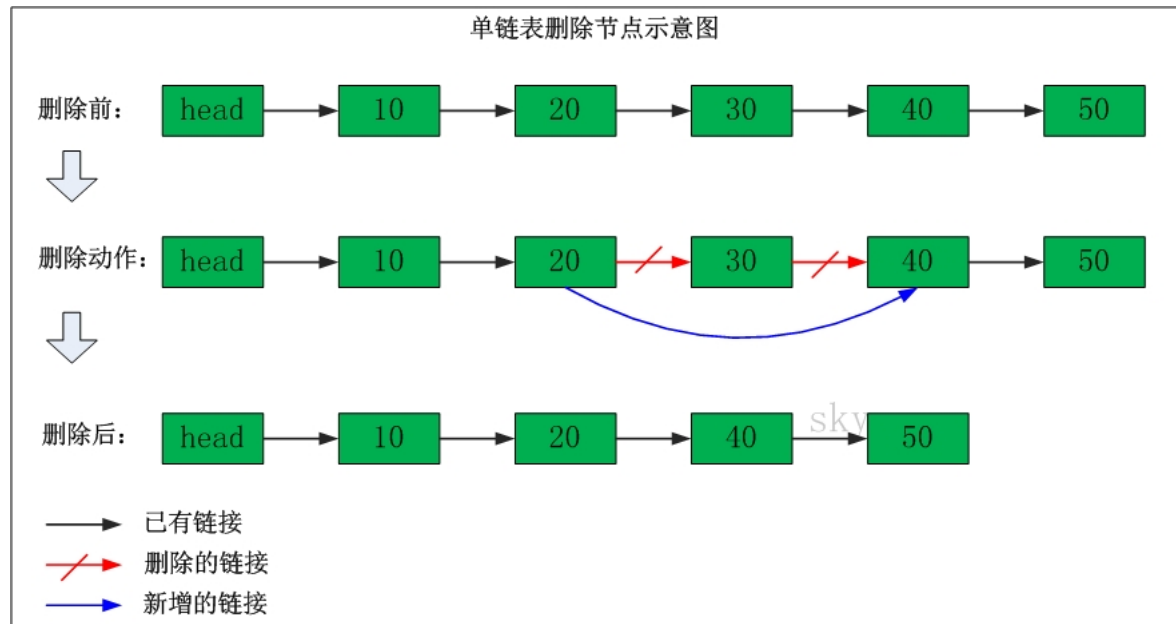
5. Java 集合系列12之-TreeMap详细介  
绍(源码解析)和使用示例(83346)

单链表的示意图如下:



表头为空, 表头的后继节点是"节点10"(数据为10的节点), "节点10"的后继节点是"节点20"(数据为10的节点), ...

### 单链表删除节点

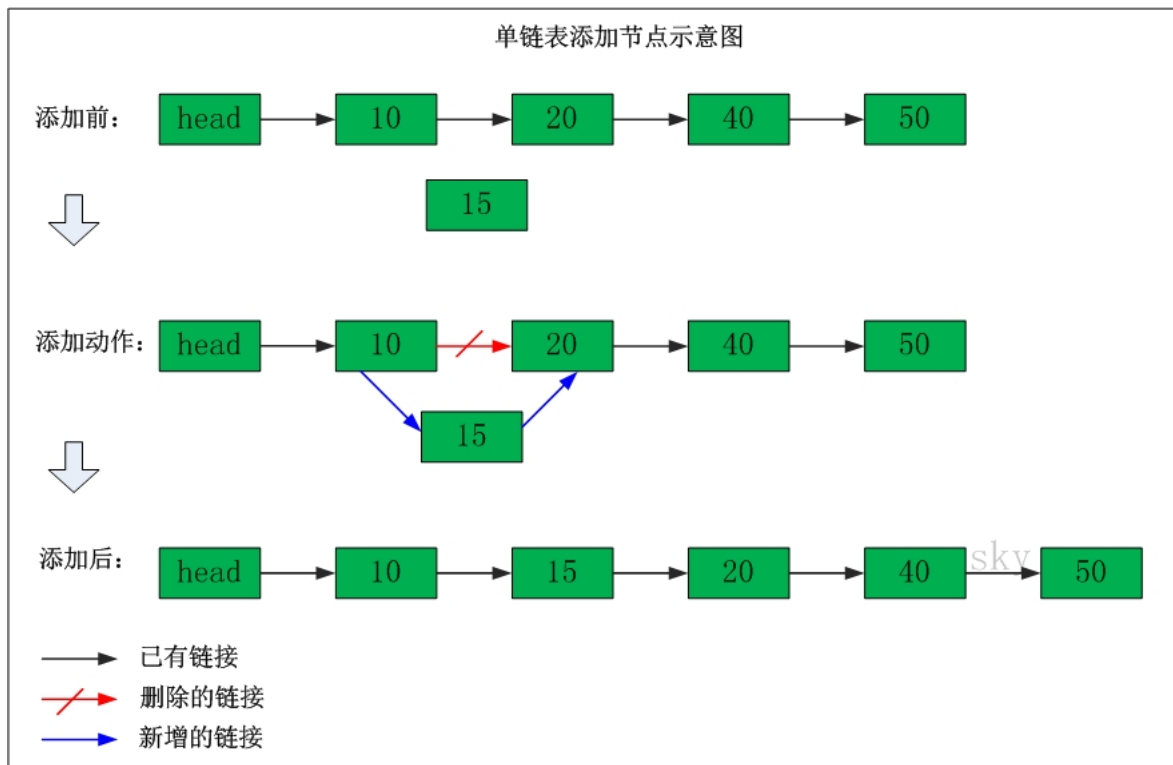


删除"节点30"

删除之前: "节点20" 的后继节点为"节点30", 而"节点30" 的后继节点为"节点40"。

删除之后: "节点20" 的后继节点为"节点40"。

### 单链表添加节点



在"节点10"与"节点20"之间添加"节点15"

**添加之前:** "节点10" 的后继节点为"节点20"。

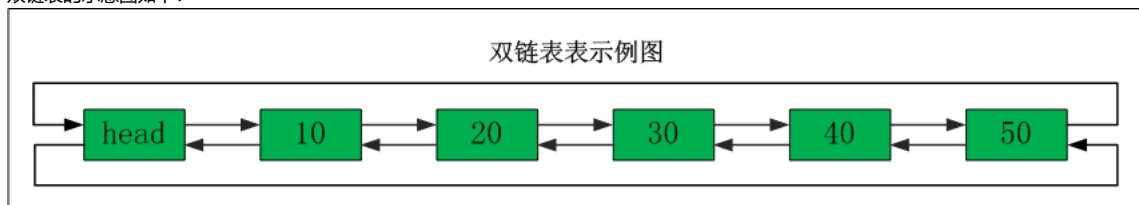
**添加之后:** "节点10" 的后继节点为"节点15", 而"节点15" 的后继节点为"节点20"。

单链表的特点是: 节点的链接方向是单向的; 相对于数组来说, 单链表的随机访问速度较慢, 但是单链表删除/添加数据的效率很高。

## 双向链表

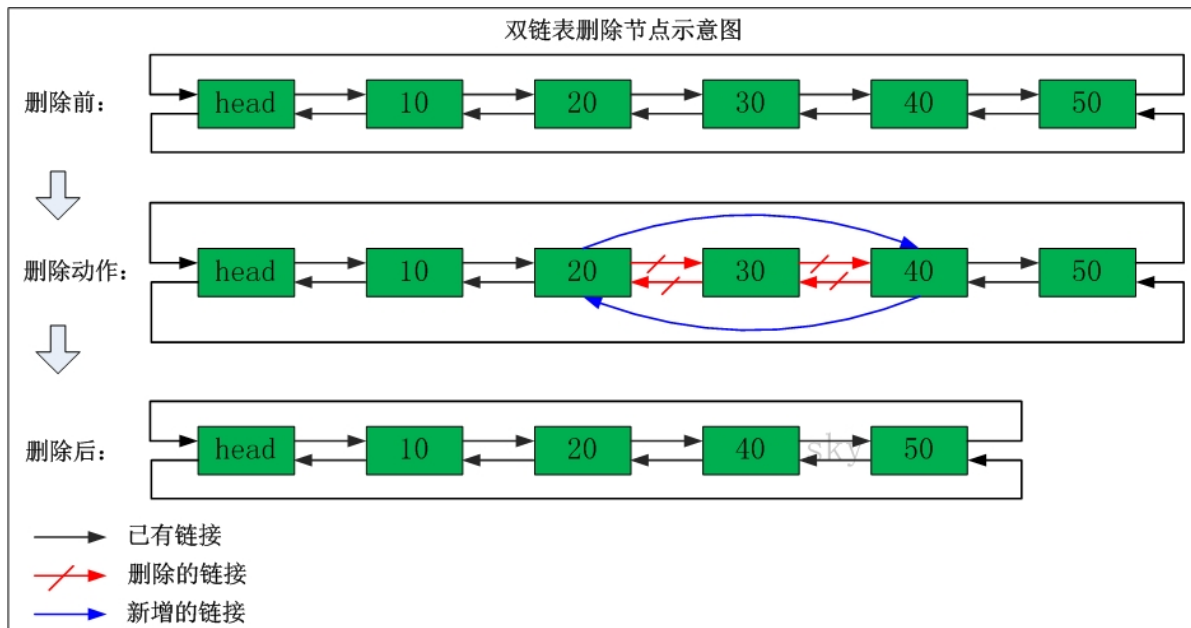
双向链表(双链表)是链表的一种。和单链表一样, 双链表也是由节点组成, 它的每个数据结点中都有两个指针, 分别指向直接后继和直接前驱。所以, 从双向链表中的任意一个结点开始, 都可以很方便地访问它的前驱节点。我们构造双向循环链表。

双链表的示意图如下:



表头为空, 表头的后继节点为"节点10"(数据为10的节点); "节点10"的后继节点是"节点20"(数据为10的节点), "节点20"的前继节点是"节点10"; "节点20"的后继节点是"节点30", "节点30"的前继节点是"节点20"; ... 节点50是表头。

## 双链表删除节点

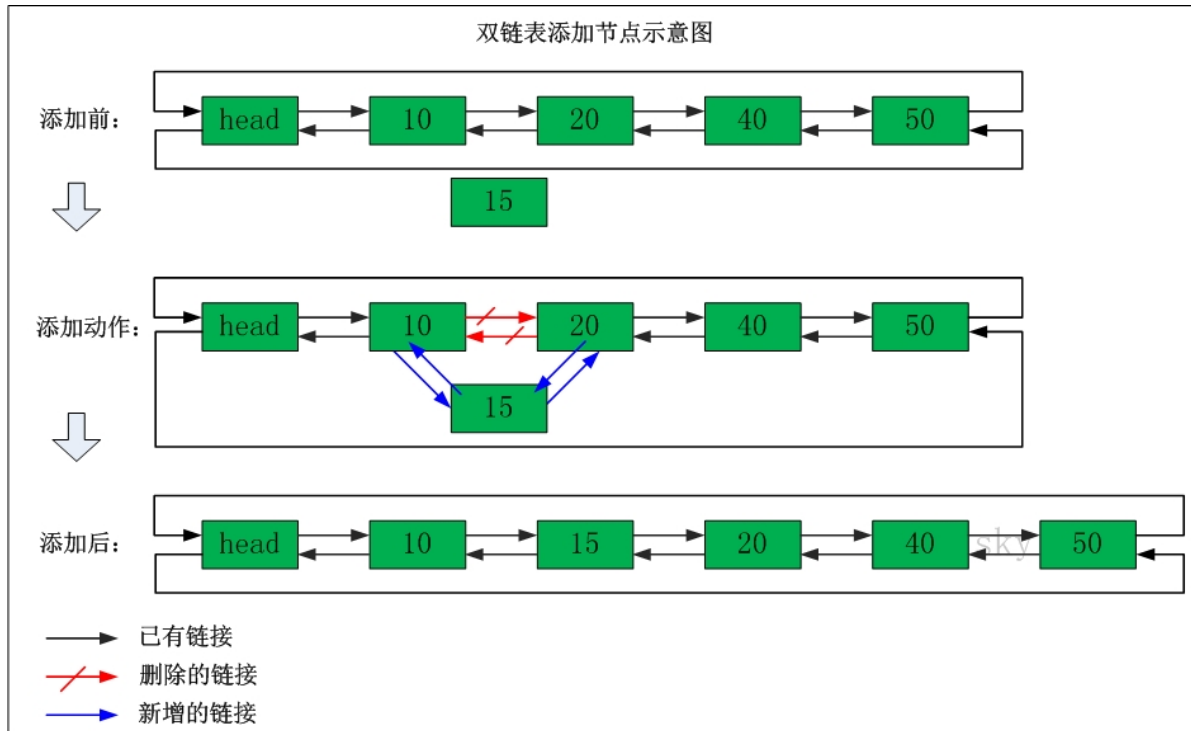


删除"节点30"

**删除之前:** "节点20"的后继节点为"节点30", "节点30" 的前继节点为"节点20"。"节点30"的后继节点为"节点40", "节点40" 的前继节点为"节点30"。

**删除之后:** "节点20"的后继节点为"节点40", "节点40" 的前继节点为"节点20"。

## 双链表添加节点



在"节点10"与"节点20"之间添加"节点15"

**添加之前:** "节点10"的后继节点为"节点20", "节点20" 的前继节点为"节点10"。

**添加之后:** "节点10"的后继节点为"节点15", "节点15" 的前继节点为"节点10"。"节点15"的后继节点为"节点20", "节点20" 的前继节点为"节点15"。

下面介绍双链表的实现，分别介绍C/C++/Java三种实现。

## 1. C实现双链表

### 实现代码

双向链表头文件(double\_link.h)

[View Code](#)

双向链表实现文件(double\_link.c)

[View Code](#)

双向链表测试程序(dlink\_test.c)

[View Code](#)

### 运行结果



```
----int_test----
dlink_is_empty()=0
dlink_size()=3
dlink_get(0)=30
dlink_get(1)=20
dlink_get(2)=10

----string_test----
dlink_is_empty()=0
dlink_size()=3
dlink_get(0)=thirty
dlink_get(1)=twenty
dlink_get(2)=ten

----object_test----
dlink_is_empty()=0
dlink_size()=3
dlink_get(0)=[30, vic]
dlink_get(1)=[20, jody]
dlink_get(2)=[10, sky]
```



## 2. C++实现双链表

### 实现代码

双向链表文件(DoubleLink.h)

View Code

双向链表测试文件(DlinkTest.cpp)

View Code

### 示例说明

在上面的示例中，我将双向链表的"声明"和"实现"都放在头文件中。而编程规范告诫我们：将类的声明和实现分离，在头文件(.h文件或.hpp)中尽量只包含声明，而在实现文件(.cpp文件)中负责实现！

那么为什么要这么做呢？这是因为，在双向链表的实现中，采用了模板；而C++编译器不支持对模板的分离式编译！简单点说，如果在DoubleLink.h中声明，而在DoubleLink.cpp中进行实现的话；当我们在其他头象时，会编译出错。具体原因，可以参考[为什么C++编译器不能支持对模板的分离式编译](#)。

### 运行结果

```

----int_test----
is_empty()=0
size()=3
pdlink(0)=30
pdlink(1)=20
pdlink(2)=10

----string_test----
is_empty()=0
size()=3
pdlink(0)=thirty
pdlink(1)=twenty
```

```
pdlink(2)=ten

----object_test----
is_empty()=0
size()=3
pdlink(0)=[30, vic]
pdlink(1)=[20, jody]
pdlink(2)=[10, sky]
```



### 3. Java实现双链表

实现代码

[双链表类\(DoubleLink.java\)](#)

按 Ctrl+C 复制代码

```
}

// 返回链表是否为空
public boolean isEmpty() {
    return mCount==0;
}

// 获取第index位置的节点
private DNode<T> getNode(int index) {
    if (index<0 || index>=mCount)
        throw new IndexOutOfBoundsException();

    // 正向查找
    if (index <= mCount/2) {
        DNode<T> node = mHead.next;
        for (int i=0; i<index; i++)
            node = node.next;

        return node;
    }

    // 反向查找
    DNode<T> rnode = mHead.prev;
    int rindex = mCount - index -1;
    for (int j=0; j<rindex; j++)
        rnode = rnode.prev;

    return rnode;
}

// 获取第index位置的节点的值
public T get(int index) {
    return getNode(index).value;
}
```

按 Ctrl+C 复制代码

[测试程序\(DlinkTest.java\)](#)

按 Ctrl+C 复制代码

```
/**
 * Java 实现的双向链表。
 * 注: java自带的集合包中有实现双向链表, 路径是:java.util.LinkedList
 *
 * @author skywang
 * @date 2013/11/07
 */

public class DlinkTest {

    // 双向链表操作int数据
    private static void int_test() {
        int[] iarr = {10, 20, 30, 40};

        System.out.println("\n---int_test---");
        // 创建双向链表
        DoubleLink<Integer> dlink = new DoubleLink<Integer>();

        dlink.insert(0, 20);    // 将 20 插入到第一个位置
        dlink.appendLast(10);   // 将 10 追加到链表末尾
        dlink.insertFirst(30);  // 将 30 插入到第一个位置


        // 双向链表是否为空
        System.out.printf("isEmpty()=%b\n", dlink.isEmpty());
        // 双向链表的大小
        System.out.printf("size()=%d\n", dlink.size());

        // 打印出全部的节点
        for (int i=0; i<dlink.size(); i++)
            System.out.println("dlink(\"+i+\")="+ dlink.get(i));
    }

    private static void string_test() {
```

按 Ctrl+C 复制代码

## 运行结果



```
---int_test---
isEmpty()=false
size()=3
dlink(0)=30
dlink(1)=20
dlink(2)=10

---string_test---
isEmpty()=false
size()=3
dlink(0)=thirty
dlink(1)=twenty
dlink(2)=ten

---object_test---
isEmpty()=false
size()=3
dlink(0)=[30, vic]
```



```
dlink(1)=[20, jody]
dlink(2)=[10, sky]
```



生活的悲欢离合永远在地平线以外，而眺望是一种青春的姿态...

PS.文章是笔者分享的学习笔记，若你觉得可以、还行、过得去、甚至不太差的话，可以“推荐”一下的哦。就此谢过！

[好文要顶](#)[关注我](#)[收藏该文](#)

如果天空不死

[关注](#) - 9

[粉丝](#) - 1940

[+加关注](#)

« [上一篇：数据结构与算法系列 目录](#)

» [下一篇：Linux内核中双向链表的经典实现](#)

posted on 2014-03-24 09:43 [如果天空不死](#) 阅读(40234) 评论(23) [编辑](#) [收藏](#)

#### Comments

##### #1楼

☆Ronny\

Posted @ 2014-03-24 10:09

挺不错的！希望博主坚持把这个系列写好~

##### #2楼

死神一护

Posted @ 2014-03-24 10:56

这我必须推荐，感谢楼主的讲解，很详细~~

##### #3楼

雨霖林

Posted @ 2014-03-24 12:16

双链表添加节点中，你的蓝色箭头是不是画少了吧

##### #4楼[楼主]

如果天空不死

Posted @ 2014-03-24 13:43

@ 雨霖林

嗯，的确是少画了。

原文已修正，谢谢指出。

##### #5楼

john23.net

Posted @ 2014-03-25 14:49

感谢分享

##### #6楼

唐小喵

Posted @ 2014-05-15 15:06

C++实现:

T DoubleLink<T>::get(int index) 函数没有判断获取节点为NULL的情况。

#7楼

Cyning

Posted @ 2014-05-20 22:25

Java 中的 public void insert(int index, T t)

感觉是这个吧:

```
DNode<T> inode = getNode(index);
DNode<T> newNode = new DNode<T>(t, inode.prev, inode);
inode.prev.next = newNode;
inode.prev = newNode;
```

#8楼

老船长

Posted @ 2014-05-29 15:31

博主是个认真的人，你的几个系列，我都来来回回看了好几遍。感谢博主辛苦了把这些心得领悟的东西分享出来。

#9楼

左右斋

Posted @ 2014-06-18 17:05

楼主:

```
// 将节点插入到第index位置之前
96 DNode<T> tnode = new DNode<T>(t, inode.prev, inode);
97 inode.prev.next = tnode;
98 inode.next = tnode;
最后应该为 inode.prev = tnode;
```

在96行的new中，已经为tnode分配了prev和next的引用。链表中需要重新建立的引用应该为inode.prex.next和inode的prev。每一个结点指出的链接数为2，被指入的链接数也应该为2，所以应该是有next和prev执行new出来的点在链表中的关系才能被正确维护。

#10楼

JunnySmile

Posted @ 2014-06-20 11:47

@ 死神一护

ewf

#11楼

TealerProg

Posted @ 2014-06-24 08:58

@ 左右斋

同意你的观点，刚才不小心点成反对了

#12楼

老船长

Posted @ 2014-09-27 14:31

@ Cyning

引用

```
Java 中的 public void insert(int index, T t)
感觉是这个吧:
DNode<T> inode = getNode(index);
DNode<T> newNode = new DNode<T>(t, inode.prev, inode);
inode.prev.next = newNode;
inode.prev = newNode;
```

对，我亲自debug过，这里有问题的

#### #13楼

[kaiscript](#)

Posted @ 2016-07-17 10:08

双链表JAVA版 98行应该是

inode.prev = tnode;吧

把index后面节点的前驱指向新结点

#### #14楼

[战星星](#)

Posted @ 2016-08-12 11:31

代码写的很漂亮，我这种水B感觉压力很大

#### #15楼

[Mihai](#)

Posted @ 2016-09-16 12:41

只能说666

#### #16楼

[安息茴香](#)

Posted @ 2016-10-24 23:11

不错，通俗易懂

#### #17楼

[Yyyyyyyyyh](#)

Posted @ 2017-03-14 15:07

// 反向查找

```
DNode<T> rnode = mHead.prev;
```

```
int rindex = mCount - index - 1; (这里的rindex应该=mCount-index+1吧? )
```

```
for (int j = 0; j < rindex; j++)
```

```
rnode = rnode.prev;
```

```
return rnode;
```

#### #18楼

[yabay2208](#)

Posted @ 2017-10-26 11:29

hello，神，你的双向链表理论有错，**双向链表有头尾，只是一条链，首尾是不连接的，双向循环链表首尾才会连接**

#### #19楼

[陈东的博客](#)

Posted @ 2017-11-09 15:58

DNode是自己创建的类么

?

#### #20楼

[SinghamYuan](#)

Posted @ 2018-03-13 10:28

表头为空，表头的后继节点是"节点10"(数据为10的节点)，"节点10"的后继节点是"节点20"(数据为10的节点)，...

应该修正为：

表头为空，表头的后继节点是"节点10"(数据为10的节点)，"节点10"的后继节点是"节点20"(数据为20的节点)，...

#### #21楼

小强fly888

Posted @ 2018-03-16 16:26

java代码insert () 方法中:inode.next=tnode应该为inode.prev=tnode;

#### #22楼

yuwen2424

Posted @ 2018-08-09 20:01

java版本

```
public void insert(int index, T t) {  
    if (index==0) {  
        DNode<T> node = new DNode<T>(t,mHead,mHead.next);  
        mHead.next.prev = node;  
        mHead.next = node;  
        mCount++;  
        return ;  
    }  
}
```

这是在干吗？ node 两次给不同的对象赋值(mHead.next.prev = node;  
mHead.next = node;)

还有 mHead先获取next再获取prev的意义何在(mHead.next.prev)

#### #23楼

doudouw

Posted @ 2018-08-16 22:21

// 将节点插入到第index位置之前

```
public void insert(int index, T t) {  
    if (index==0) {  
        DNode<T> node = new DNode<T>(t, mHead, mHead.next);  
        mHead.next.prev = node;  
        mHead.next = node;  
        mCount++;  
        return ;  
    }  
}
```

```
DNode<T> inode = getNode(index);  
DNode<T> tnode = new DNode<T>(t, inode.prev, inode);  
inode.prev.next = tnode;  
inode.next = tnode;  
mCount++;  
return ;  
}  
inode.next = tnode;应该是inode.prev = tnode;
```

刷新

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。



#### 最新IT新闻:

- 苹果进军电影行业: 购买两部影片版权
- 马云要辞职: 外媒“炸”了 印度这个国家急了
- 特斯拉再融46亿 融资鬼才马斯克六年八次施展绝技
- 盖茨: 我为扎克伯格提供建议 教他如何应对国会听证
- 机场安检塑料盒“最脏”: 包含的病毒比卫生间还多

» [更多新闻...](#)



华为全联接大会 | 上海 | 2018.10.10-12



[ 大会门票+云服务器 ] 专属套餐0.35折起

#### 最新知识库文章:

- 为什么说 Java 程序员必须掌握 Spring Boot ?
- 在学习中, 有一个比掌握知识更重要的能力
- 如何招到一个靠谱的程序员
- 一个故事看懂“区块链”
- 被踢出去的用户

» [更多知识库文章...](#)

---

Powered by:

[博客园](#)

Copyright © 如果天空不死