Linked list用画图的方式比较好理解。

首先，我们设定一个node（节点），这个节点是一个pointer。它指向一个地址。Linked list 就是指，将多个node（节点）连接成一串糖葫芦（单链表），或者一串佛珠（双链表），糖葫芦上每一颗葫芦就是一个node。

这里有个重要概念，如果要设置linked list，他的结尾和开头应该设置一个 nullptr。这不是必须的，但为了不使程序错误，或者起到哨兵的作用，我们会设置一个头链表。（注：逻辑上cs-210作业4并没有设置头链表）

以CS-210作业4为例子. 我们先设置一个structure

struct NodeType

{

long id;

string fname;

string lname;

double balance;

NodePtr flink;

NodePtr blink;

};

Typedef NodeType \*Nodeptr

NodeType 本来是一个struct的定义，如果我们平时在main里用的话

Int main（）｛

NodeType data;

｝

那么我们就可以使用 data.id， data.fname，等来储存或者调去在struct NodeType里的数据。

但是我们用了Typedef NodeType \*Nodeptr

在这里，我们通过NodeType这个struct 再次定义了一个新的定义代码 – Nodeptr

Nodeptr在这里就变成了定义连接到 NodeType 这个struct的point。

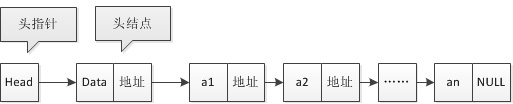
之后只需要用: Nodeptr head。 这个head就变成了NodeType的一个point。

为了区分链表point 和 普通调用struct的区别，我们使用 head->id，来调用struct里的数据

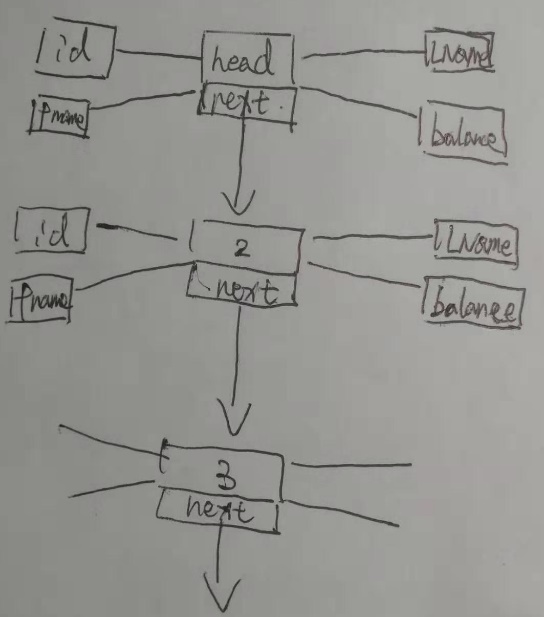
这时head->id 其实等同于 head.id

单链表的意思是 设置一个head point，这个head point = nullptr，然后设置一个head->next, 等于在设立一个新的point，把这个head->next指向另一个point，这样类推，就形成了一个串链接。

（注：c++是可以辨识 head->next->next这样的语句的，不用多次定义，只要链表形成，->next就是下一个节点的代码）



但在用了struct以后，每个point内分别还有不同的数据内容，在这样的情况下，以作业4为例子，单链表的大概图像是这样的



这就是单链表的大概工作原理。 Next的作用是移动当前point去下一个node。

所以我们在读取链表的时候，会再设置一个point

比如： temp = newptr; Temp = head,

这时，temp和head的地址一样了，他可以读取这个point链接的所有数据

比如： temp->id temp->fname 等

如何读取下一个？ Temp = head->next，这时temp等于head->next这个point了，当然，我们也可以使用 temp=temp->next，效果是一样的，把这个node移动到下一个node。也就是图上的“2”。

这样就可以遍历整个单链表。

双链表。双链表其实就是在node的上下都设置一个链接。

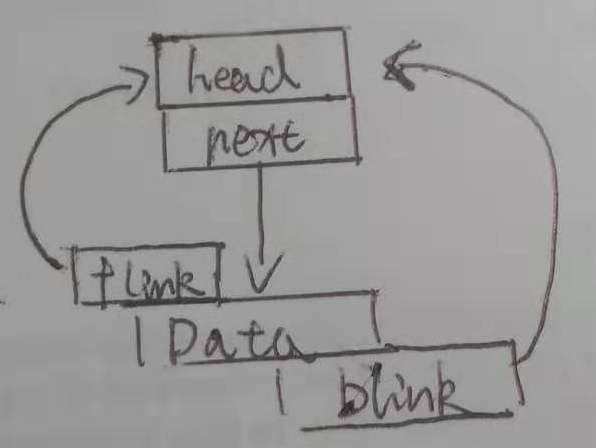
Head = nullptr；data = newptr；

Head->next = Data；

Data->blink = head；

Data->flink = head；

上面的代码就形成了一个双链表



那么，如果我们需要在当中添加新的节点的话，我们就需要让head也有flink和blink

data2 = newptr；

data->flink = data2；

data2->blink = data；

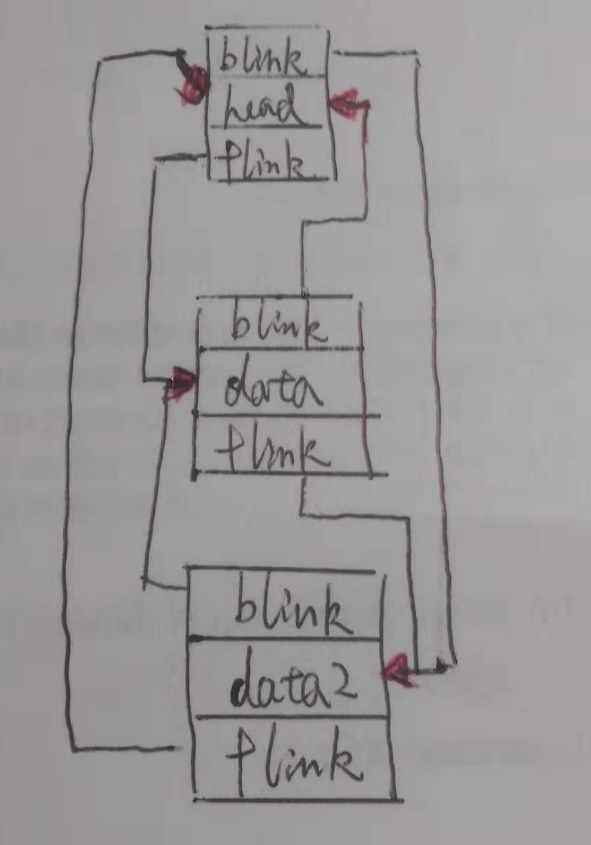
data2->flink = head；

head->blink = data2；

head->flink = data；

data->blink = head；

这就是双链表的关系

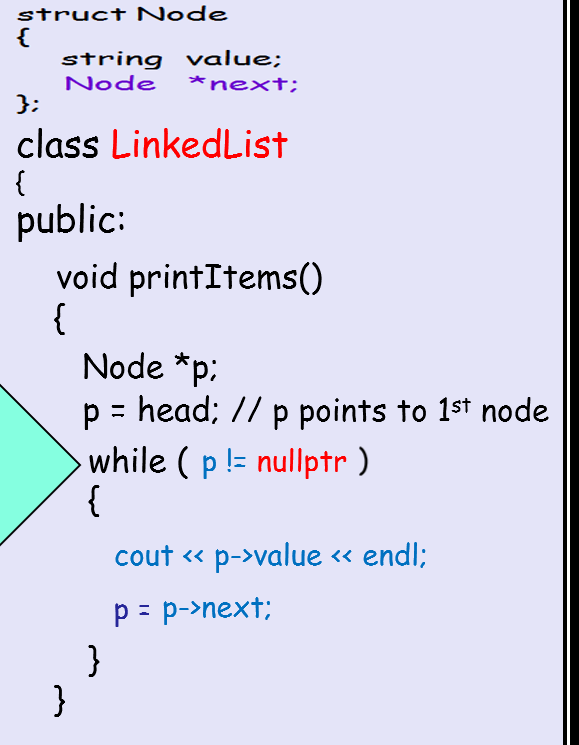


Node的使用是配合struct来的，这样能够更好的分配内存。如果要做到封闭的话，需要class，这时我们就可以用class和struct配合的方式

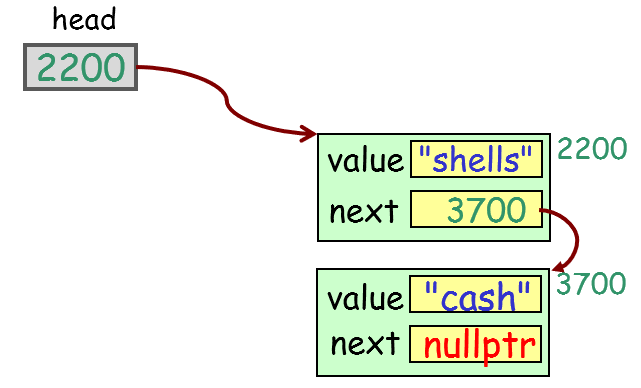
Cs-210作业4的方式就是class和struct配合，用两个struct来储存数据，一个用来记录初始数据，另一个用来记录数据的pointer，也就是形成链表。形成链表后，数据不会被覆盖，所以第一个struct用来单个记录数据，然后储存为第二个struct的链表文件。

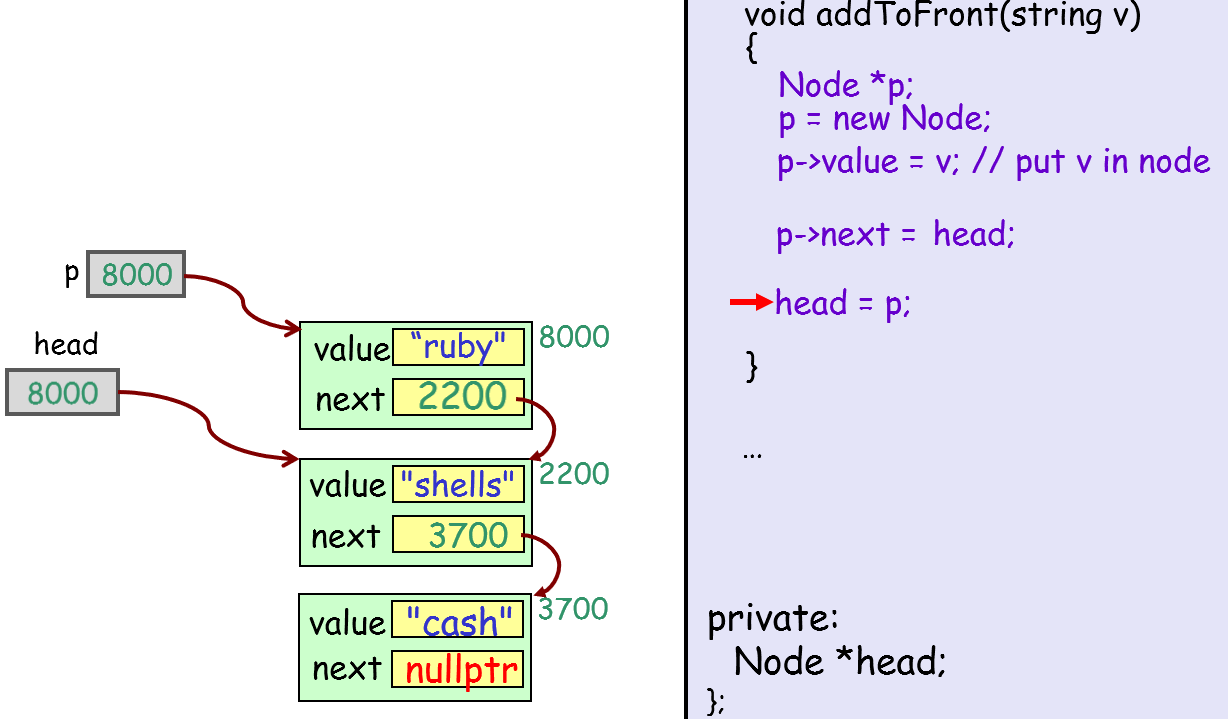
Class的作用，是把struct里的链表point作为私密代码，然后将function放在class里进行操作。

遍历整个node的方法不能用for loop，因为不能定义循环数，所以我们用while loop，去寻找一个节点，这就是为什么需要head为空的原因，因为头尾为空的话，我们可以设置while loop在读取到空pointer时停止循环



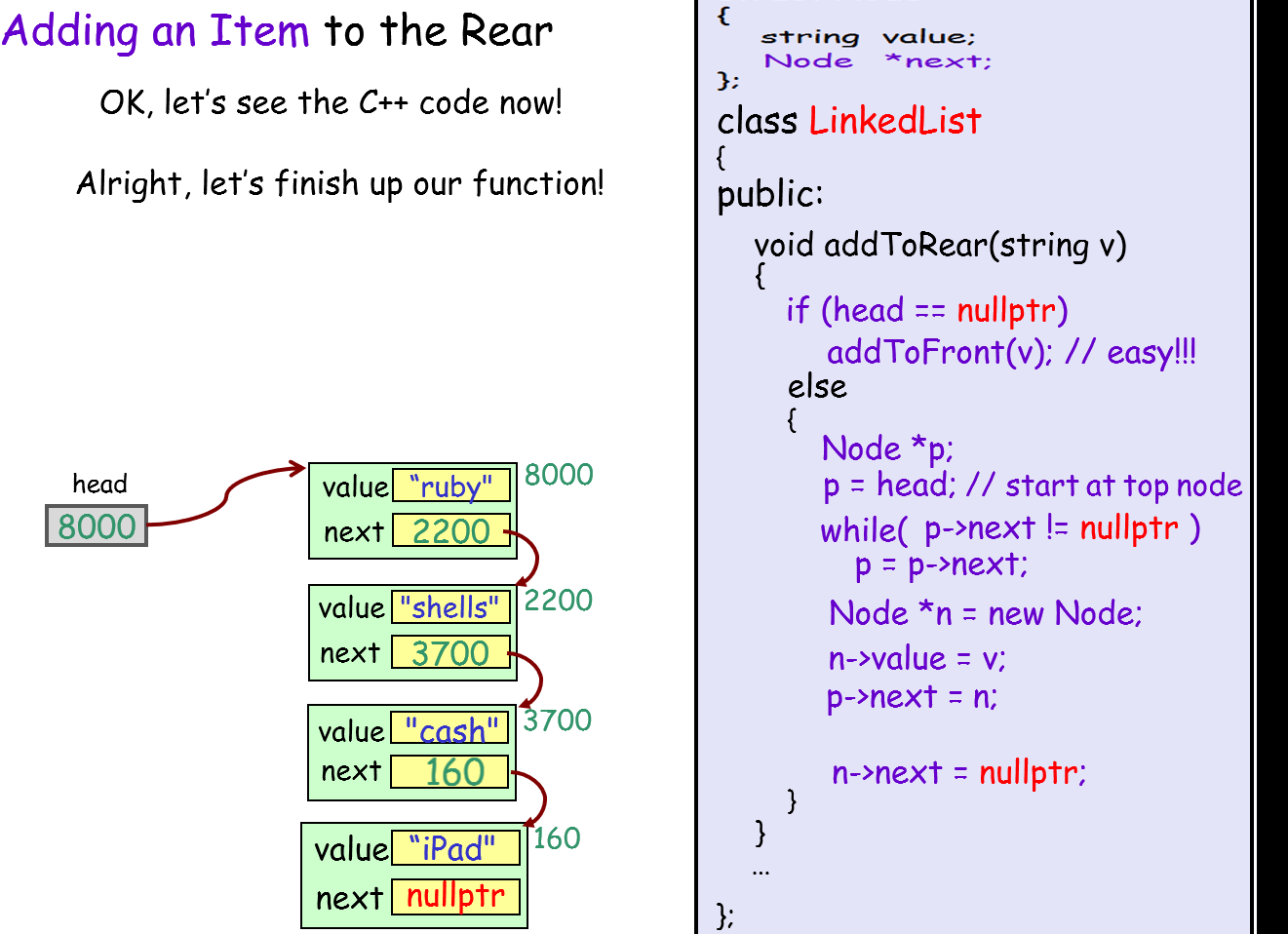
添加新的node（节点）

**初始链表**  加入P这个新的node。

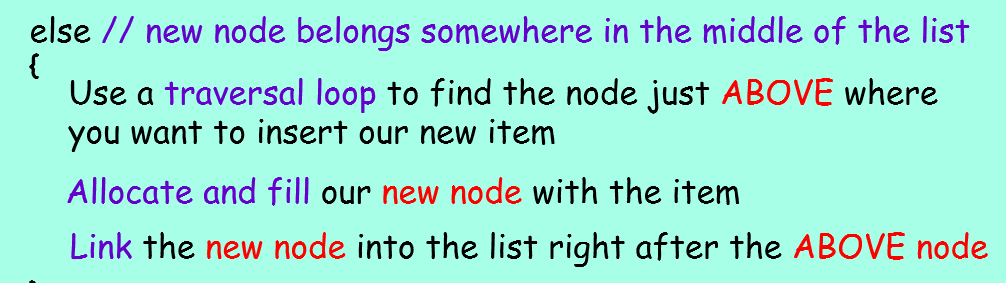
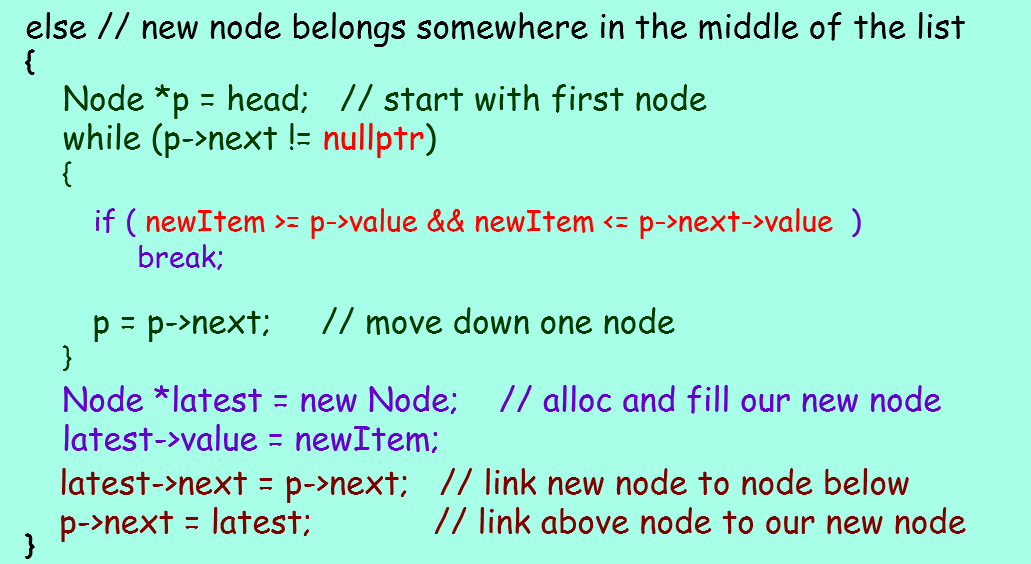


然后让P->next 链接上原先的head，最后让head等于P。新的节点就加入了，而head->next就变成了原先的head.

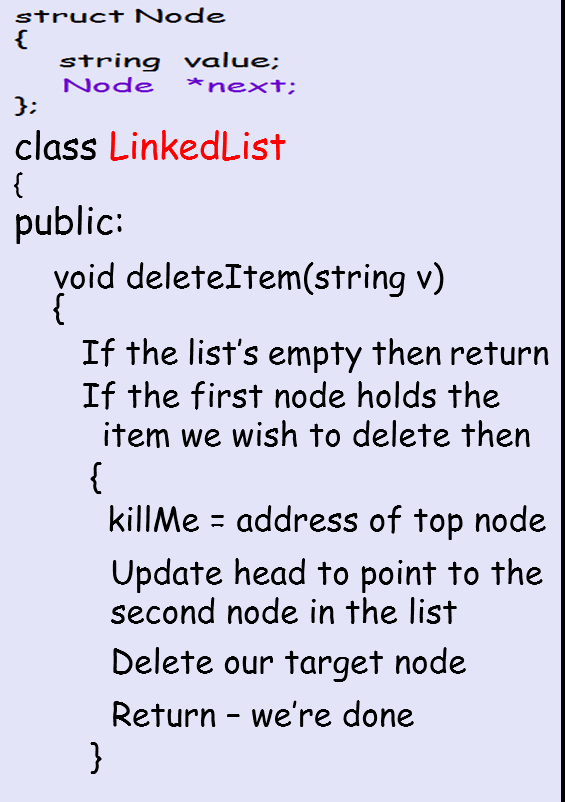
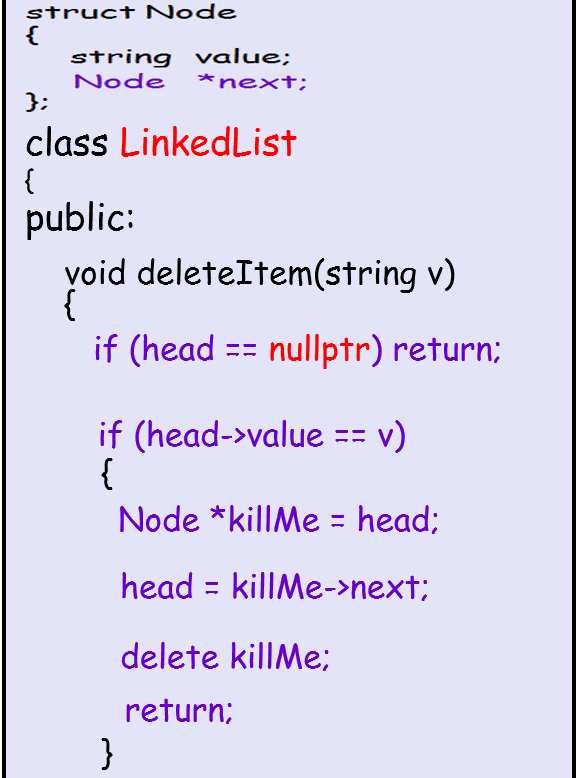
添加到最后面



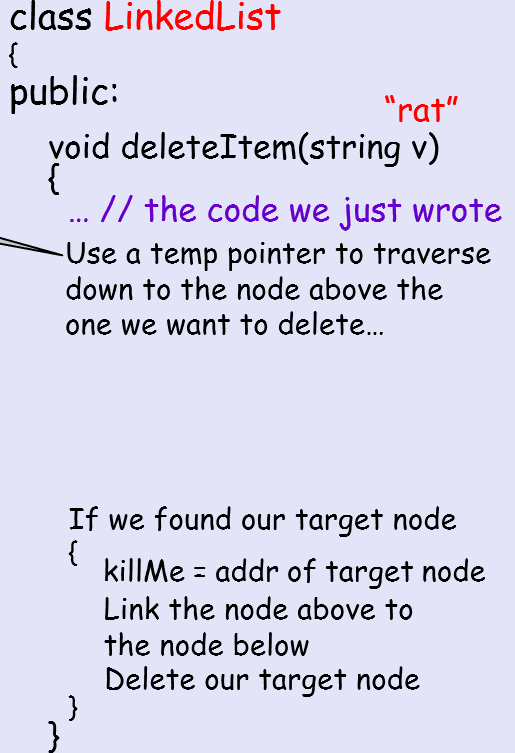
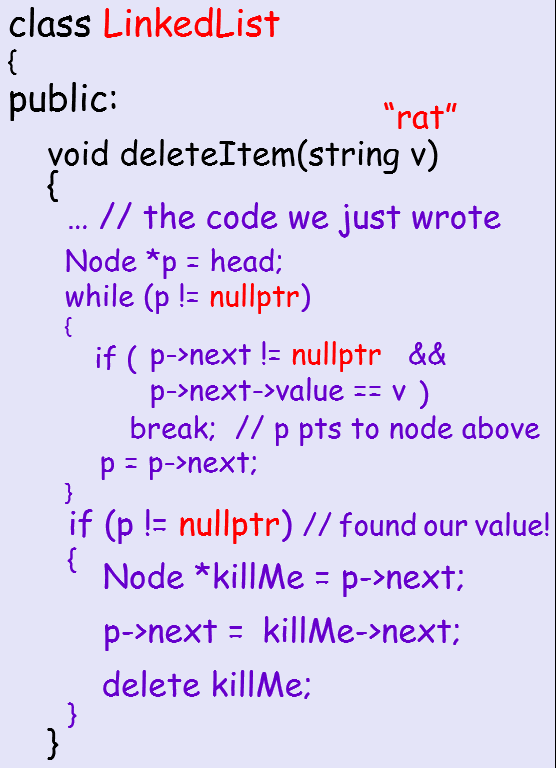
**在中间插入节点**

**Linked list 设计思路**

**删除节点的思路**

Class 能做到使用完linked list后销毁这个list。

**Destructing Linked list**

