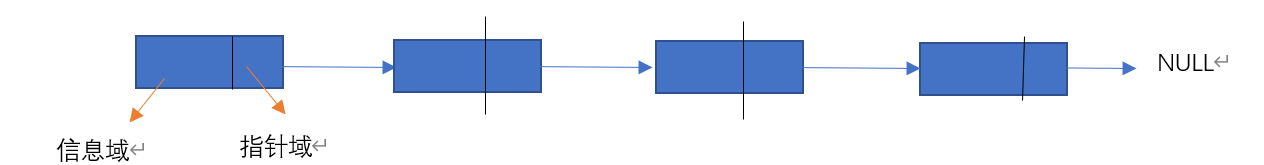
单链表的常用代码

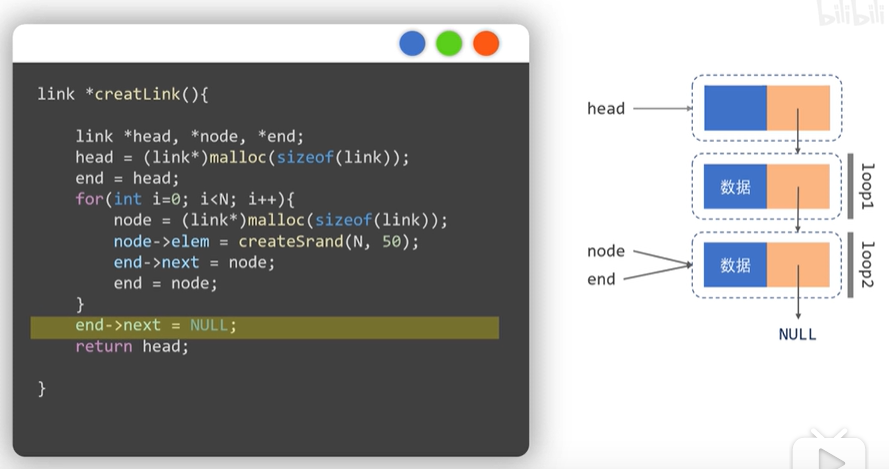
概念：链表是结构体变量与结构体变量连在一起。

动态内存申请

结构体指针---------------->结构体变量

(结构体\*）malloc(sizeof(结构体));





**创建表**

结构体\* createList()

{

结构体\* headNode=(结构体\*)malloc(sizeof(结构体))； //变成结构体变量

headNode->next=NULL;

return headNode;

}

（这样在主函数就可以用 结构体\* 名字=createList();就可以构造一个链表）

**创建节点**

结构体\* createNode(int x)

{

结构体\* newNode(名字)=动态分配；

newNode->data=x;

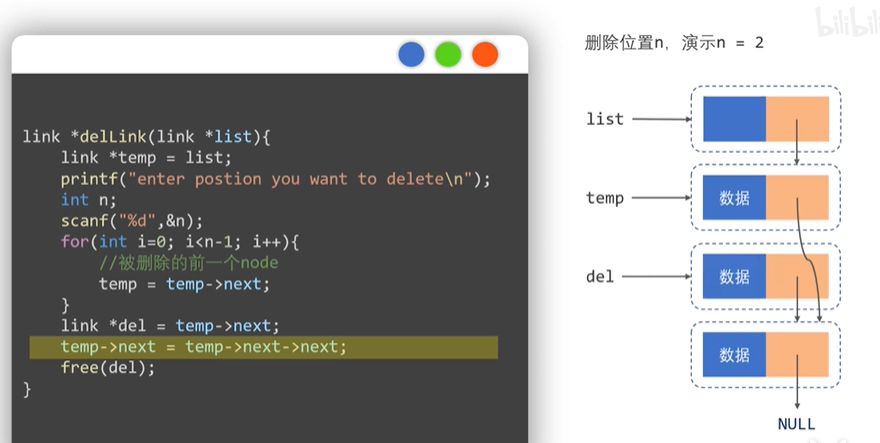
newNode->next=NULL;

return newNode;

}

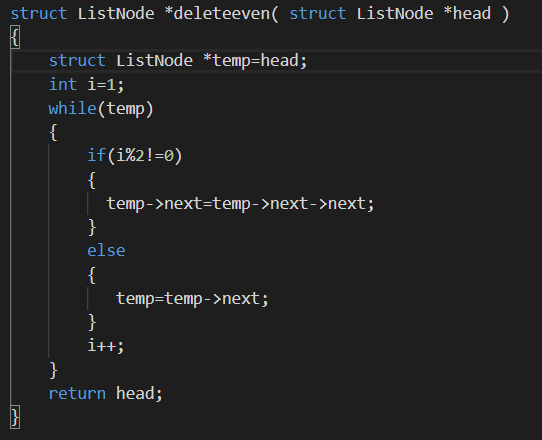


**链表的插入**

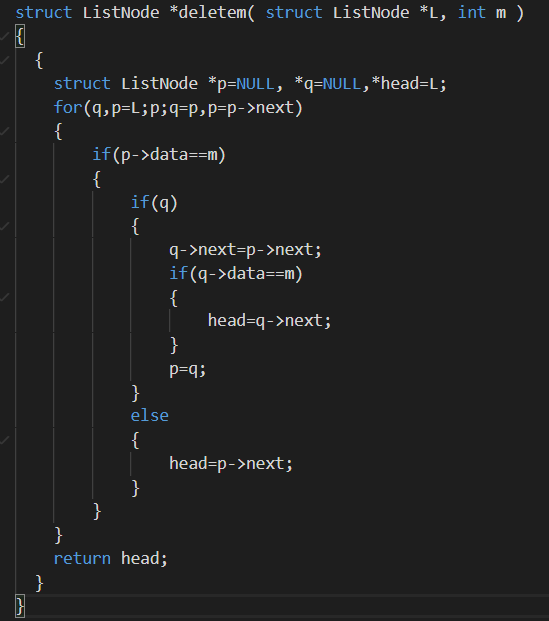


**链表节点的删除**

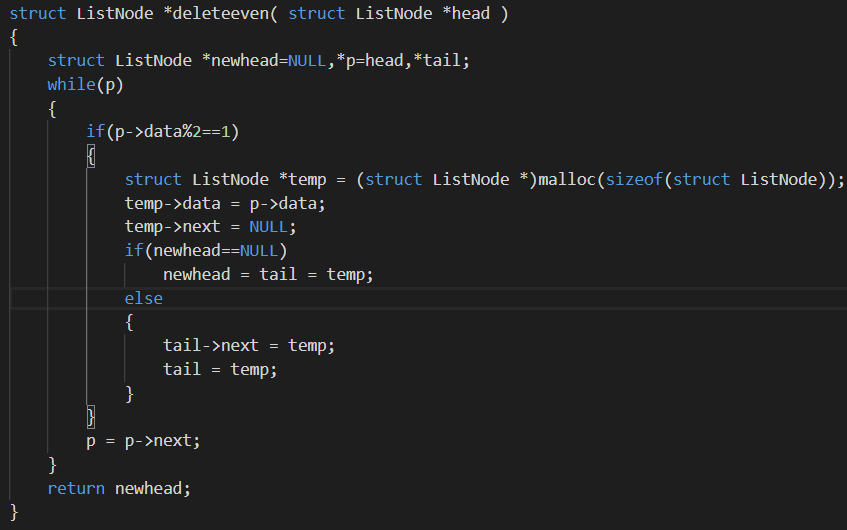
**链表的偶数位节点的删除（跳过）**



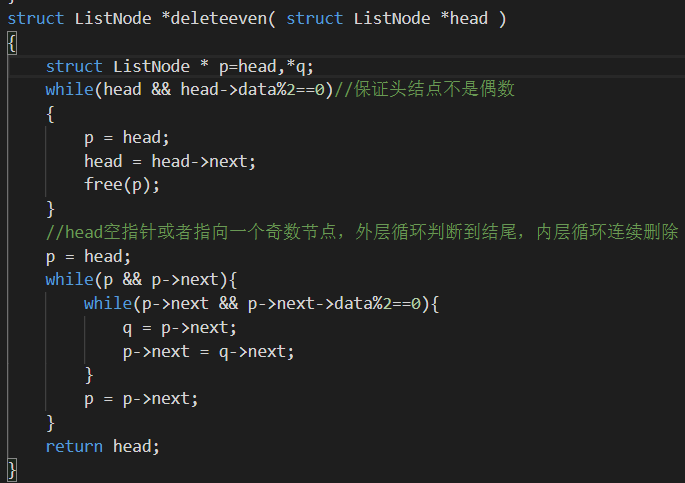
**链表的指定值的删除**



**链表的偶数值的节点的删除（方法1）（创建一个新链表储存奇数的）**



**链表的偶数值的节点的删除（方法2）（删除是偶数的头结点，其他的跳过）**



**链表的打印**

Void printList(结构体\* headNode)

{

结构体\* Pmove=headNode;->next;

While(Pmove)

{

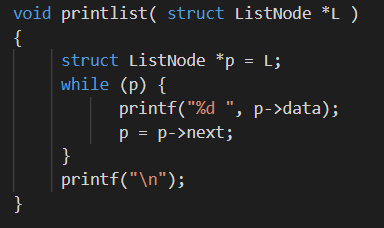
Printf(“%d ”,Pmove->data);

Pmove=pmove->next;

}

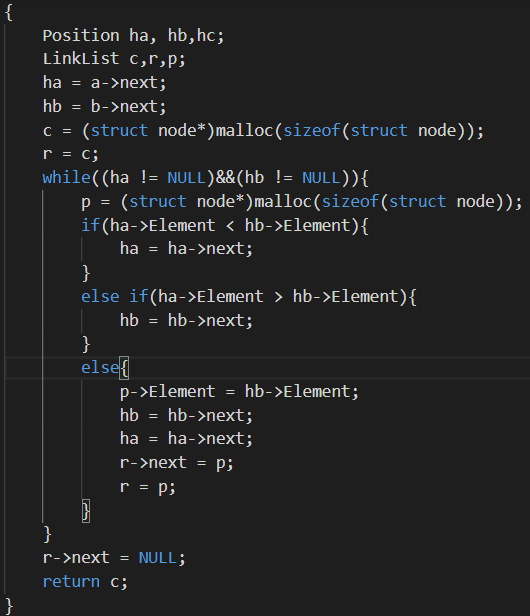
Printf(“\n”);

}

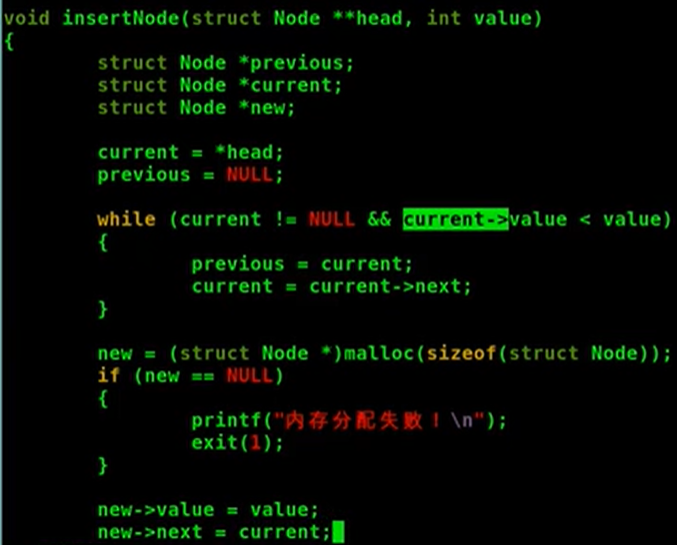
（最好创建一个指针，代替传进来的链表）

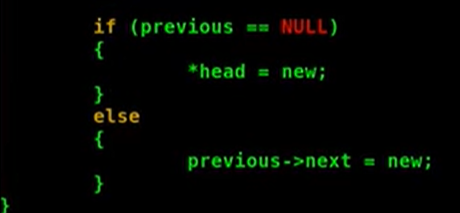
**链表的交集**

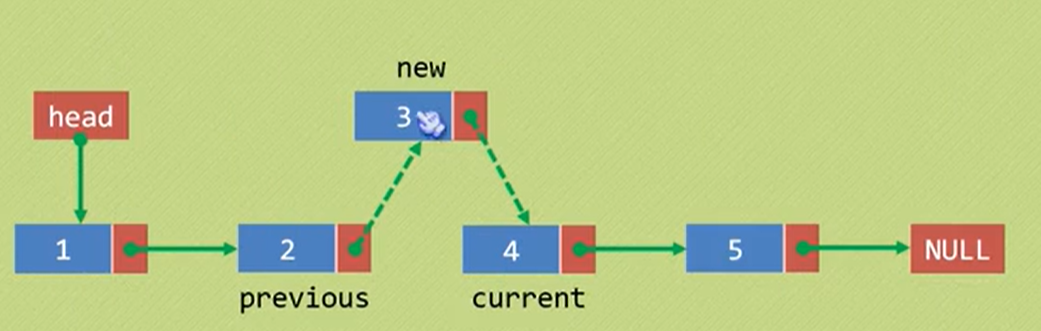




**把一个数插入到有序链表中，仍有序**

（新建两个指针，一个表示现在的节点，一个表示前一个的）

（如果链表一开始就为空）

**（概念图）**

**内存池**

**创建一个内存池**

MAX为内存池的最大容量

**单链表创建新节点时，看看内存池里边是否有之前被放在池里的内存**



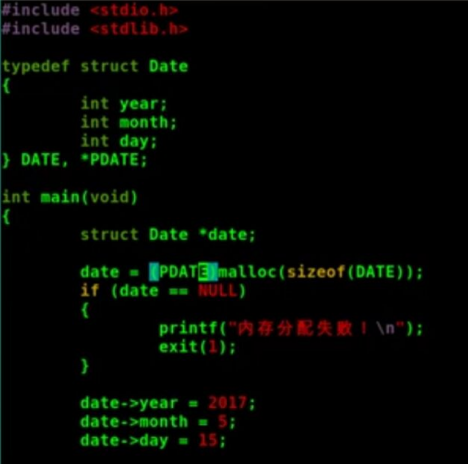
**单链表删除节点时，把节点放入内存池里边，如果内存池满了，就把节点释放**

****

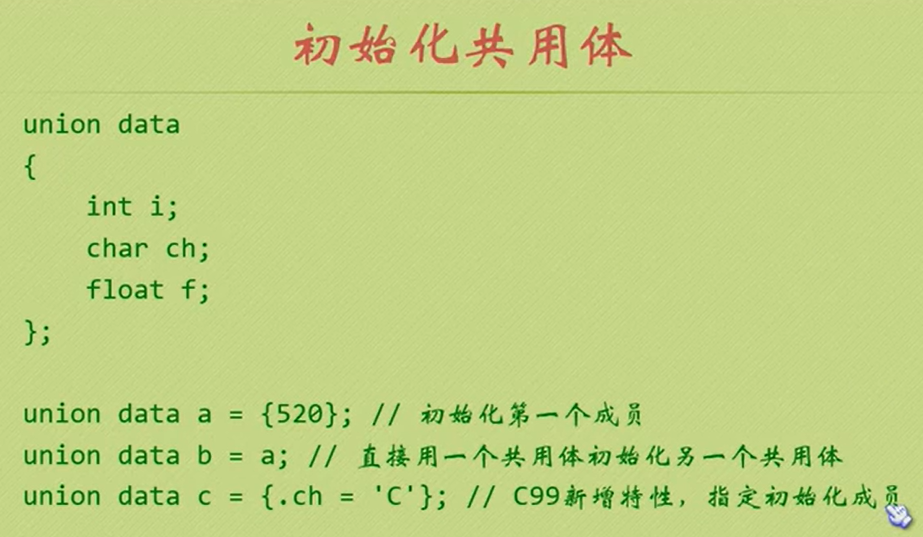
**Typedef**

**起别名的命令**

**可以给结构体变量起别名**



**共用体**

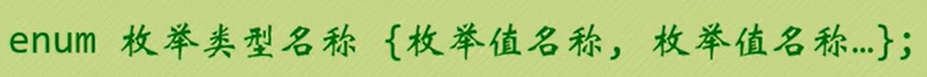


**共用体的成员的地址都是一样的，所以同时赋值会产生覆盖**

**引用时想结构体一样(a.i=1;)**

**枚举类型**

**枚举类型的声明**



**例子**



**一般情况sun为int型，值为0，后面依次加一。**

**位域（把一个字节（8个二进制位）拆开用 ||int是四个字节）（char int bool可以用）**

