**结构体别名是指针**

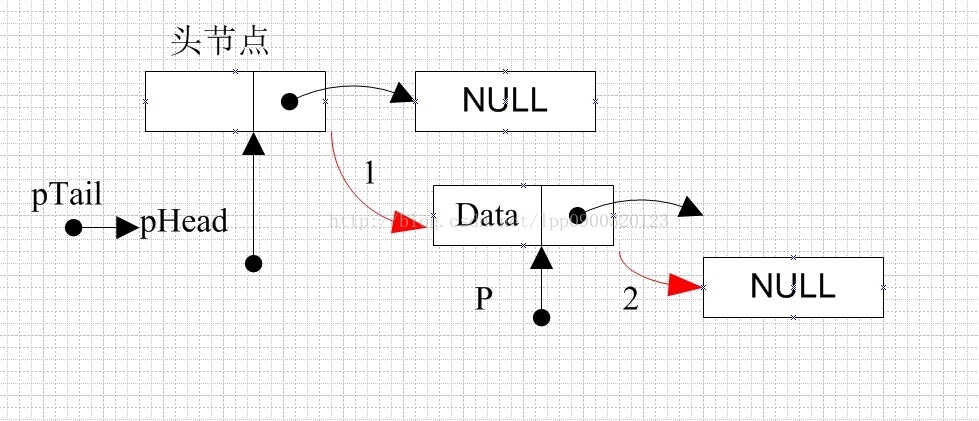


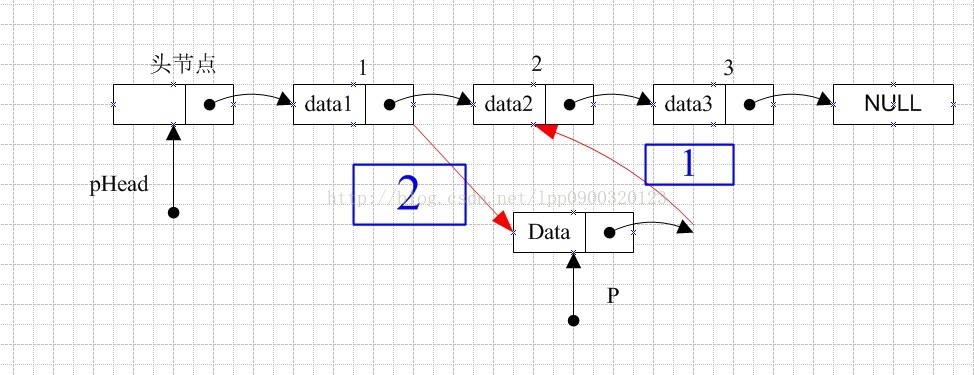
[千篇不一律](https://blog.csdn.net/qq_41063141) 2018-10-04 06:12:07  3534  收藏 15

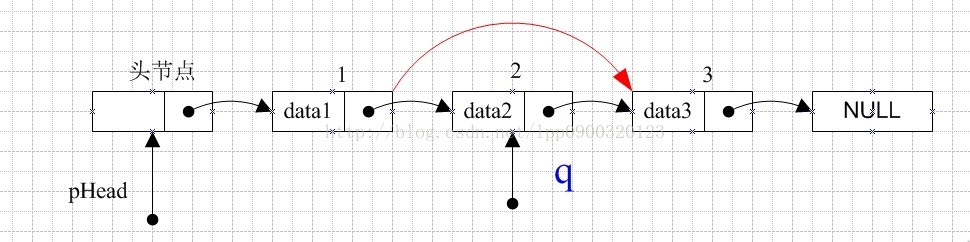
分类专栏： [c/c++](https://blog.csdn.net/qq_41063141/category_9025173.html)

1.什么是链表      链表我的理解要包含以下特征：（1）.由n个节点离散分配；（2）.每个节点通过指针连接（3）每一个节点由一个前驱节点和一个后驱节点（4）.首节点没有前驱节点，尾节点没有后驱节点；      满足上面的4条，我们就称为链表；链表既然由很多个节点，那节点又由什么组成？节点由两个部分组成，一是数据域，用来存放有效数据；二是指针域，用来指向下一个节点；下面用C语言来构建链表数据结构，首先应该构造出节点，然后再把所有的节点连起来，就构成了链表； （1）节点的构造

typedef struct Node  
{  
    int data;//数据域，用来存放数据域；  
    struct Node \*pNext;//定义一个结构体指针，指向下一次个与当前节点数据类型相同的节点  
}NODE,\*PNODE;  //NODE等价于 struct Node; PNODE等价于struct Node \*； 此处用大写是为了与变量区分，可以让人容易变出是个数据类型







 首节点：存放第一个有效数据的节点；    尾节点：存放最后一个有效数据的节点；    头节点：头节点的数据类型与首节点的数据类型相同，并且头节点是首节点前面的那个节点，并不存放有效数据；头节点的存在只是为了方便链表的操作。    头指针：指向头节点的指针；    尾指针：指向尾节点的指针； 首先，我们应该创建一个头节点，并用头指针指向它，用C语言描述：用malloc向计算机申请一块内存，并定义一个指向与头节点数据类型相同的指针（一定要判断申请内存是否成功）； 然后，要知道要创建链表的长度，用一个循环来每次创建一个节点，并把每个节点连在一起；

假如我们要在头节点phead后面插入节点p： （1）把头节点的指针域指向P节点，即pHead->pNext=p; （2）把p节点的指针域指向NULL，即p->pNext=NULL; 这样就可以了吗？ 想想我们就可以发现，当我们要插入多个节点时，头节点始终指向最后添加的一个数据，以前的节点通过头指针此时已经找不到了；我们定义一个尾指针pTail，始终用来指向链表的结尾，每次只在pTail后面添加节点。 伪算法： （1）定义一个尾指针pTail，并初始化，使它指向头节点，即pTail=pHead； （2）在pTail后面添加节点，修改指针：         pTail->pNext=p;         p->pNext=NULL;       pTail=p;              //使pTail指向链表最后一个元素

链表是一种物理[存储单元](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%8D%95%E5%85%83/8727749)上非连续、非顺序的[存储结构](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84/350782)，[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313)的逻辑顺序是通过链表中的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88/2878304)链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0)的数据域，另一个是存储下一个结点地址的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88/2878304)域

[线性表](https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E8%A1%A8)的链式存储表示的特点是用一组任意的[存储单元](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%8D%95%E5%85%83)存储[线性表](https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E8%A1%A8)的[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0)（这组存储单元可以是连续的，也可以是不连续的）。因此，为了表示每个[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0) 与其直接后继数据元素 之间的逻辑关系，对数据元素 来说，除了[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8)其本身的信息之外，还需存储一个指示其直接后继的信息（即直接后继的存储位置

单链表是一种链式存取的数据结构，用一组地址任意的[存储单元](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%8D%95%E5%85%83/8727749)存放线性表中的[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313)。链表中的数据是以结点来表示的，每个结点的构成：元素([数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313)的映象) + [指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88/2878304)(指示后继元素存储位置)，元素就是存储数据的存储单元，指针就是连接每个结点的地址数据。

链表中的数据是以结点来表示的，每个结点的构成：元素([数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0)的映象) + [指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)(指示后继元素存储位置)，元素就是存储数据的存储单元，指针就是连接每个结点的地址数据。

以“结点的序列”表示线性表称作[线性链表](https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E9%93%BE%E8%A1%A8)（单链表），单链表是链式存取的结构

data域--存放结点值的数据域

next域--存放结点的直接后继的地址（位置）的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)域（链域）

链表通过每个结点的链域将线性表的n个结点按其逻辑顺序链接在一起的，每个结点只有一个链域的链表称为单链表（Single Linked List）。

**头指针head和终端结点**

单链表中每个结点的存储地址是存放在其前趋结点next域中，而开始结点无前趋，故应设头指针head指向开始结点。链表由头[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)唯一确定，单链表可以用头指针的名字来命名。

终端结点无后继，故终端结点的指针域为空，即NULL。

typedef struct node{ //结点类型定义

DataType data; //结点的数据域

struct node \*next;//结点的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)域

}ListNode;

typedef ListNode \*LinkList;

ListNode \*p;

LinkList head; [1] 

注意：

①LinkList和ListNode是不同名字的同一个指针类型（命名的不同是为了概念上更明确）

②\*LinkList类型的[指针变量](https://baike.baidu.com/item/%E6%8C%87%E9%92%88%E5%8F%98%E9%87%8F)head表示它是单链表的头指针

③ListNode类型的指针变量p表示它是指向某一结点的指针

**指针变量和结点变量**

|  |  |  |
| --- | --- | --- |
|  | 指针变量 | 结点变量 |
| 定义 | 在变量说明部分显式定义 | 在程序执行时，通过标准函数malloc生成 |
| 取值 | 非空时，存放某类型结点 | 实际存放结点各域内容的地址 |
| 操作方式 | 通过指针变量名访问 | 通过指针生成、访问和释放 |

①生成结点变量的标准[函数](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0/301912)

p=( ListNode \*)malloc(sizeof(ListNode))；

//函数malloc分配一个类型为ListNode的结点变量的空间,并将其首地址放入[指针变量](https://baike.baidu.com/item/%E6%8C%87%E9%92%88%E5%8F%98%E9%87%8F)p中

②释放结点变量空间的标准函数

free(p)；//释放p所指的结点变量空间

③结点分量的访问

利用结点变量的名字\*p访问结点分量

方法一：(\*p).data和(\*p).next

方法二：p-﹥data和p-﹥next

④指针变量p和结点变量\*p的关系

[指针变量](https://baike.baidu.com/item/%E6%8C%87%E9%92%88%E5%8F%98%E9%87%8F)p的值——结点地址

结点变量\*p的值——结点内容

(\*p).data的值——p[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)所指结点的data域的值

(\*p).next的值——\*p后继结点的地址

\*((\*p).next)——\*p后继结点 [1]

注意：

① 若指针变量p的值为空（NULL），则它不指向任何结点。此时，若通过\*p来访问结点就意味着访问一个不存在的变量，从而引起程序的错误。

② 有关指针类型的意义和说明方式的详细解释

可见，在链表中插入结点只需要修改指针。但同时，若要在第 i 个结点之前插入元素，修改的是第 i-1 个结点的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)。

因此，在单链表中第 i 个结点之前进行插入的基本操作为:

找到线性表中第i-1个结点，然后修改其指向后继的指针。

**动态存储**

编辑

链表操作中[动态存储](https://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E5%AD%98%E5%82%A8)分配要使用[标准函数](https://baike.baidu.com/item/%E6%A0%87%E5%87%86%E5%87%BD%E6%95%B0)，先介绍一下这些函数。

(1)malloc(size)

在内存的动态存储区申请一个长度为size字节的连续空间。

(2)calloc(n，size)

在内存的动态存储区申请n个长度为size字节的连续空间，函数返回值为分配空间的首地址。若此函数未被成功执行，函数返回值为0。

(3)free(p)

释放由[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)p所指向的存储单元，而存储单元的大小是最近一次调用malloc()或calloc()函数时所申请的存储空间。

在头文件\"stdlib．h”中包含了这些函数的信息，使用这些函数时需在程序开头用文件包含指令#include“stdlib．h”指明。

调用动态存储分配函数返回的指针是指向void类型或char类型的指针，在具体使用时，要根据所指向的数据进行[强制类型转换](https://baike.baidu.com/item/%E5%BC%BA%E5%88%B6%E7%B1%BB%E5%9E%8B%E8%BD%AC%E6%8D%A2)。

