typedef struct Node

{

DataType data;

struct Node \*next;

}ListNode,\*LinkList;

void InitList(LinkList \*h)

/\*单链表的初始化\*/

{

if((\*h=(LinkList)malloc(sizeof(ListNode)))==NULL) /\*为头结点分配一个存储空间\*/

exit(-1);

(\*h)->next=NULL; /\*将单链表的头结点指针域置为空\*/

}

/////

**头结点**

head

h

\*h

***注意！***

**初始化后得到的内存空间情况如图，其中\*h是头指针，头指针的值是头节点的地址，由于头节点的地址是动态变量，随机生成，故是变化的，怎样抓住这种变化，及时获得头节点的情况（这才是目的）。引入了一个h指针，（h是指针的指针，h的值是头指针的地址，头指针的地址不会变化，头指针的值会变。）\*h就是头指针。头指针\*h指向的地址的内容就是头节点，可以通过\*\*h来反应。（头指针的值发生变化时，头指针会指向不同的地址。这种变化可以通过\*\*h来直接反应头节点的变化，从而绕过了头指针值的不断变化的问题。）**

**说了一大通，归根结底就是，如果是一个普通的变量，要通过函数该其值，就需要将其地址传进去；如果是一个指针变量需要该其值，（此处的值是地址，也就是指针的指向的变量的地址）就需要将指针的地址传进去。即是二重指针，指针的指针。**

int ListEmpty(LinkList h)

/\*判断单链表是否为空\*/

{

if(h->next==NULL) /\*如果链表为空\*/

return 1; /\*返回1\*/

else /\*否则\*/

return 0; /\*返回0\*/

}

ListNode \*Get(LinkList h,int i)

/\*查找单链表中第i个结点。查找成功返回该结点的指针，否则返回NUL\*/

{

ListNode \*p;

int j;

if(ListEmpty(h)) /\*查找第i个元素之前，判断链表是否为空\*/

return NULL;

if(i<1) /\*判断该序号是否合法\*/

return NULL;

j=0;

p=h;

while(p->next!=NULL&&j<i)

{

p=p->next;

j++;

}

if(j==i) /\*如果找到第i个结点\*/

return p; /\*返回指针p\*/

else; /\*否则\*/

return NULL ;/\*返回NULL\*/

}