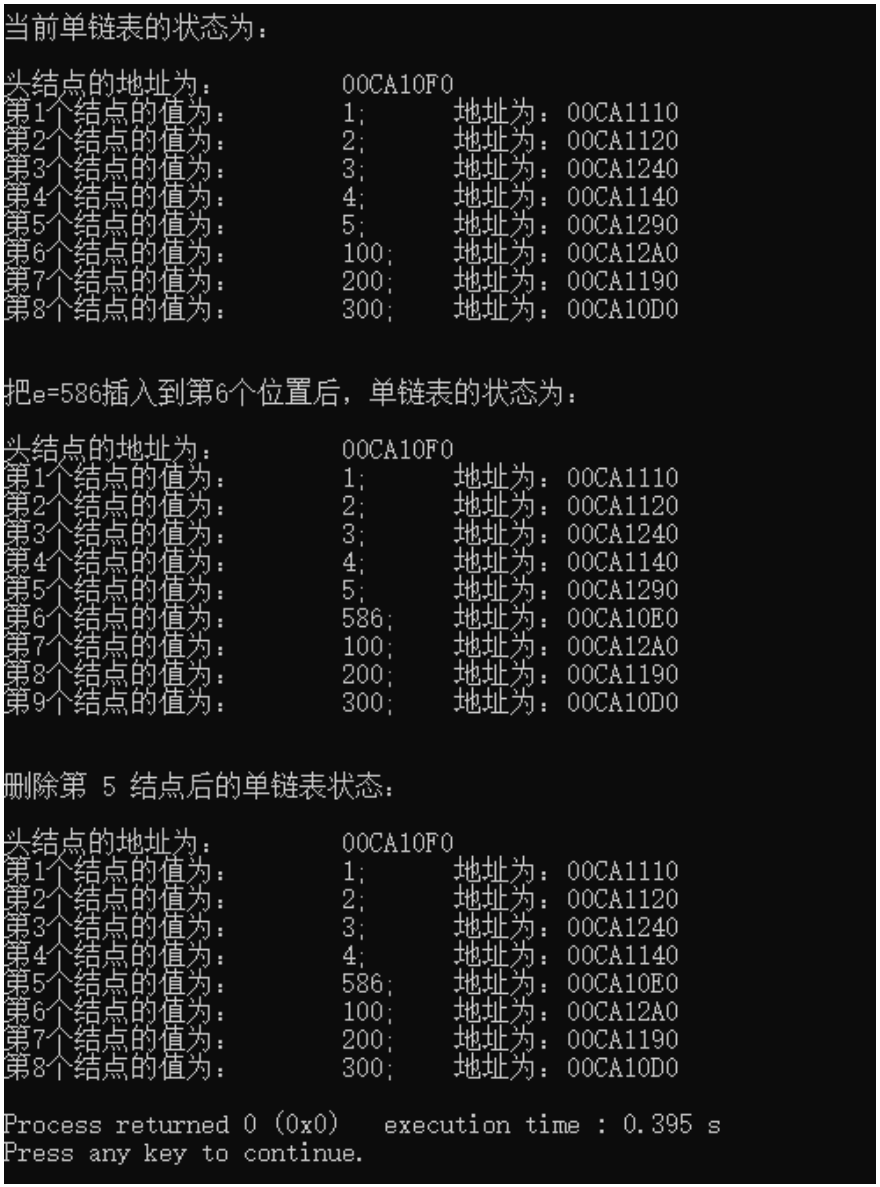
单链表算法设计示例

【示例-1】课本中插入算法与删除算法的演示。

```
#include <bits/stdc++.h>
using namespace std;
/* 添加其他算法代码 */
void ShowAddress(LinkList L)
{
    int i;
    LinkList p = L->next;
    if(p != NULL) {
        printf("\n头结点的地址为: \t%p\n", L);
        for(i = 1; p != NULL; i++) {
            printf("第%d个结点的值为: \t%d;\t地址为: %p\n", i, p->data, p);
            p = p->next;
        }
    }
}
```

```
int  main()
{
    intA[8] = {1, 2, 3, 4, 5, 100, 200, 300};
    inti, j, e = 586;
    LinkList  head, p;
    InitList_L(head);
    for(i = 1, j = 0; i <= 8; i++, j++)
        ListInsert_L(head, i, A[j]);
    printf("\n\n当前单链表的状态为: \n");
    ShowAddress(head);
    i = 6;     //插入位置
    printf("\n\n把e=%d插入到第%d个位置后, 单链表的状态为: \n", e, i);
    ListInsert_L(head, i, e);
    ShowAddress(head);
    i = 5;     //删除位置
    printf("\n\n删除第 %d 结点后的单链表状态: \n", i);
    ListDelete_L(head, i, e);
    ShowAddress(head);
}
```

运行结果


当前单链表的状态为：
头结点的地址为：00CA10F0
第1个结点的值为：1; 地址为：00CA1110
第2个结点的值为：2; 地址为：00CA1120
第3个结点的值为：3; 地址为：00CA1240
第4个结点的值为：4; 地址为：00CA1140
第5个结点的值为：5; 地址为：00CA1290
第6个结点的值为：100; 地址为：00CA12A0
第7个结点的值为：200; 地址为：00CA1190
第8个结点的值为：300; 地址为：00CA10D0
把e=586插入到第6个位置后，单链表的状态为：
头结点的地址为：00CA10F0
第1个结点的值为：1; 地址为：00CA1110
第2个结点的值为：2; 地址为：00CA1120
第3个结点的值为：3; 地址为：00CA1240
第4个结点的值为：4; 地址为：00CA1140
第5个结点的值为：5; 地址为：00CA1290
第6个结点的值为：586; 地址为：00CA10E0
第7个结点的值为：100; 地址为：00CA12A0
第8个结点的值为：200; 地址为：00CA1190
第9个结点的值为：300; 地址为：00CA10D0
删除第 5 结点后的单链表状态：
头结点的地址为：00CA10F0
第1个结点的值为：1; 地址为：00CA1110
第2个结点的值为：2; 地址为：00CA1120
第3个结点的值为：3; 地址为：00CA1240
第4个结点的值为：4; 地址为：00CA1140
第5个结点的值为：586; 地址为：00CA10E0
第6个结点的值为：100; 地址为：00CA12A0
第7个结点的值为：200; 地址为：00CA1190
第8个结点的值为：300; 地址为：00CA10D0
Process returned 0 (0x0) execution time : 0.395 s
Press any key to continue.

【示例-2】设计一个算法，通过一趟遍历确定单链表*L*（至少含两个数据结点）中第一个元素值最大的结点。

解：算法思路

用*p*遍历单链表，在遍历时用maxp指向data域值最大的结点（maxp的初值为*p*）。当单链表遍历完毕，最后返回maxp。

```
LNode *MaxNode(LNode *L)
{
   LNode *p=L->next, *maxp=p;
   while (p!=NULL)    //遍历所有的结点
   {  if (maxp->data<p->data)
         maxp=p;      //当p指向更大的结点时，将其赋给maxp
      p=p->next;      //p沿next域下移一个结点
   }
   return maxp;
}
```

【示例-3】设计一个算法，删除一个单链表L（至少含两个数据结点）中第一个元素值最大的结点。

解：算法思路

在单链表中删除一个结点先要找到它的前驱结点。

- 以p遍历单链表，pre指向p结点的前驱结点。
- 在遍历时用maxp指向data域值最大的结点，maxpre指向maxp结点的前驱结点。
- 当单链表遍历完毕，通过maxpre结点删除其后的结点，即删除了元素值最大的结点。

```
void DelMaxNode(LNode *&L)
{
   LNode *p=L->next,*pre=L,*maxp=p,*maxpre=pre;
   while (p!=NULL)
   {  if (maxp->data<p->data)
      {  maxp=p;
         maxpre=pre;
      }
      pre=p;          //pre、p同步后移，保证pre始终为p的前驱结点
      p=p->next;
   }
   maxpre->next=maxp->next;   //删除maxp结点
   free(maxp);                //释放maxp结点
}
```

【示例-4】设计一个算法，将一个单链表L（至少含两个数据结点）中所有结点逆置，并分析算法的时间复杂度。

解：算法思路

- 先将单链表L拆分成两部分，一部分是只有头结点L的空表，另一部分是由p指向第一个数据结点的单链表。
- 然后遍历p，将p所指结点逐一采用头（前）插法插入到L单链表中，由于头插法的特点是建成的单链表结点次序与插入次序正好相反，从而达到结点逆置的目的。

```
void ListReverse(LNode *&L)
{
   LNode *p=L->next, *q;
   L->next=NULL;
   while (p!=NULL)              //遍历所有数据结点
   {    q=p->next;              //q临时保存p结点之后的结点
        p->next=L->next;        //将结点p插入到头结点之后
        L->next=p;
        p = q;
   }
}
```

【示例-5】已知一个带有表头结点的单链表，结点结构为：

假设该单链表只给出了头指针list。在不改变链表的前提下，请设计一个尽可能高效的算法，查找链表中倒数第*k*个位置上的结点（k为正整数）。若查找成功，算法输出该结点的data域的值，并返回1；否则，只返回0。要求：

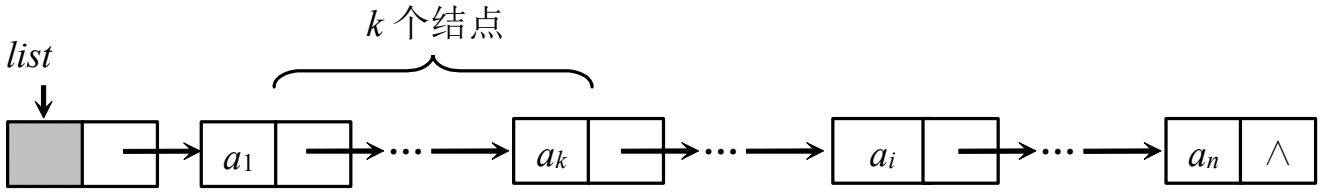
（1）描述算法的基本设计思想。

（2）描述算法的详细实现步骤。

（3）根据设计思想和实现步骤，采用程序设计语言描述算法（使用C、C++或Java语言实现），关键之处请给出简要注释。

本题为2009年全国考研题。

解：算法思路



- （1）算法的基本设计思想：定义两个指针变量p和q，初始时均指向头结点的下一个结点。p指针沿链表移动；当p指针移动到第k个结点时，q指针开始与p指针同步移动；当p指针移动到链表尾结点时，q指针所指元素为倒数第k个结点。

  以上过程对链表仅进行一遍扫描。

➢ （2）算法的详细实现步骤如下：

- ①置count=0，p和q指向链表表头结点的下一个结点。
- ②若p为空，则转向⑤。
- ③若count等于k，则q指向下一个结点；否则，置count=count+1。
- ④置p指向下一个结点，转向②。
- ⑤若count等于k，则查找成功，输出该结点的data域的值，返回1；否则，查找失败，返回0。
- ⑥算法结束。

（3）算法实现如下：

```
typedef struct Node
{  int data;
   struct Node  *link;
} LNode, *LinkList;

int  Searchk(LinkList list,int k)
{
    LinkList  p, q;
    int count=0;
    p=q=list->link;
    while (p!=NULL && count<k)        //查找第k个结点*p
    {   count++;
        p=p->link;
    }
    if (p==NULL)                      //没有时返回0
        return 0;
    else
    {  while (p!=NULL)                //p和q同步后移直到p=NULL
       {   q=q->link;
           p=p->link;
       }
       printf("%d",q->data);
       return 1;
    }
}
```

END