1.单链表的定义

```
typedef struct LNode{                    //定义单链表结点类型
    ElemType data;                       //数据域
    struct LNode *next;                  //指针域
}LNode, *LinkList;
```

```
LinkList Reverse_1(LinkList L){
//L是带头结点的单链表，本算法将L就地逆置

    LNode *p, *r;                    //p为工作指针，r为p的后继，以防断链
    p=L->next;                       //从第一个元素结点开始
    L->next=NULL;                    //先将头结点L的next域置为NULL
    while(p!=NULL){                  //依次将元素结点摘下
        r=p->next;                   //暂存p的后继
        p->next=L->next;             //将p结点插入到头结点之后
        L->next=p;
        p=r;
    }
    return  L;
}
```

```
void Del_X_1(Linklist &L,ElemType x){
//L 为带头结点的单链表，本算法删除 L 中所有值为 x 的结点
    LNode *p=L->next,*pre=L,*q;//置 p 和 pre 的初始值
    while(p!=NULL){
        if(p->data==x){
            q=p;                          //q 指向该结点
            p=p->next;
            pre->next=p;                  //删除*q 结点
            free(q);                      //释放*q 结点的空间
        }
        else{                             //否则，pre 和 p 同步后移
            pre=p;
            p=p->next;
```

例1：素数的判断算法。

```
Void prime( int n)
    /*  n是一个正整数  */
    {   int i=2 ;
        while ( (n% i)!=0 && i*1.0< sqrt(n) )   i++ ;
        if (i*1.0>sqrt(n) )
            printf("&d 是一个素数\n" , n) ;
        else
            printf("&d 不是一个素数\n" , n) ;
    }
```

嵌套的最深层语句是i++；其频度由条件((n% i)!=0 && i*1.0< sqrt(n)) 决定，显然i*1.0< sqrt(n) ，时间复杂度$O(n^{1/2})$。

例2：冒泡排序法。

```
Void bubble_sort(int a[]，int n)
    {  change=false;
       for (i=n-1; change=TURE; i>1 && change; --i)
           for (j=0; j<i; ++j)
               if (a[j]>a[j+1])
                   {   a[j] ←→a[j+1] ;  change=TURE ; }
    }
```

– 最好情况：0次

– 最坏情况：1+2+3+…+n-1=n(n-1)/2

– 平均时间复杂度为： $O(n^2)$