顺序表算法 16个

```
void del_min(sqList &L,int &value){
    //删除顺序表L中最小值元素结点，并通过引用型参数value返回其值
    //若删除成功返回true,否则返回false
    if( L.length==0 )
        return false;
    value = L.data[0];
    int pos = 0;
    for(int i=1;i<L.length;i++){
        if( L.data[i]<value ){
            value=L.data[i];
            pos=i;
        }
    }
    L.data[pos]=L.data[L.length-1];
    L.length--;
    return true;
}

/* 顺序表L所有元素逆置，要求算法空间复杂度为O(1)*/
void Reverse(SqList &L){
    Elemtype temp;  //辅助变量
    for ( i=0;i<L.length;i++ ){
        temp = L.data[i];   //交换L.data[i]与L.data[L.length-i-1]
        L.data[i] = L.data[L.length-i-1];
        L.data[L.length-i-1]=temp;
    }
}
```

```
/* 顺序表长度 n,删除该表中值为x的数据元素,要求：时间复杂度:O(n),空间复杂度O(1) */
/**
 *解法一：
 *用k记录L中不等于x的数据元素个数（即需要保存的元素个数），
 *边扫描L边统计K,并将不等于x的元素向前移动k个位置，最后修改L的长度.
*/
void del_x_1(Sqlist &L,Elemtype x){
    int k=0;
    for(i=0;i<L.length;i++){
        if(L.data[i] != x){
            L.data[k] = L.data[i];
            k++;
        }
    }
    L.length=k;
}

/**
 * 解法二：
 * 用k记录L中等于x的元素，边扫描边统计k，并将不等于x的元素前移k个位置，
 * 最后修改l的长度
*/
void del_x_2(Sqlist &L, Elemtype x){
    int k=0;
    int i=0;
    while(i<L.length){
        if(L.data[i]==x){
            k++;
        }else{
            L.data[i-k] = L.data[i]; //当前元素前移 k 个位置
            i++;
        }
    }
    L.length = L.length-k;
}
```

```
/* 删除 x,  s< x <t */
//同上题算法一
void del_s_t_1( SqList *l, int s, int t ){
    int k=0,i=0;
    while(i<l->length){
        if(l->data[i]>s && l->data[i]<t){    //如果条件加上'=', 则s, t也会被删除
            k++;
        }else{
            l->data[i-k] = l->data[i];
        }
        i++;
    }
    l->length = l->length-k;
}


/* 删除 x ,  s<x<t 或 s<=x<=t*/
// 同上题算法二
void del_s_t_2( SqList *L, int s, int t ){
    int k=0;
    for(int i=0;i<L->length;i++){
        if(L->data[i] <= s || L->data[i] >= t){ //将等号去掉, 则s, t也将被删除
            L->data[k] = L->data[i];
            k++;
        }
    }
    L->length = k;
}
//删除 s<=x<=t
bool Del_s_t2(SqList &L,ElemType s, ElemType t){
    int i,j;
    if(s>=t || L.length==0)
        return false;
    for(i=0;i<L.length && L.data[i]<s;i++);  //寻找大于等于s的第一个元素
    if(i>=L.length)
        return false;
    for(j=i;j<L.length&&L.data[j]<=t;j++);  //寻找值大于t的第一个元素
    for( ; j<L.length; i++, j++ )
        L.data[i] = L.data[j];
    L.length = i;
    return true;
}
```

```
/**
 * 题目描述：有序表删除所有重复的元素，使表中所有元素的值均不同
 * 算法思想：初始时将第一个元素看作是没有重复的顺序表，依次遍历L后序元素，
 * 遇到不同，插入，遇到相同，跳过，继续遍历。最后更新表的长度
*/
bool Del_same(SqList &L){
    if(L.length==0)
        return false;
    int i,j;
    for(i=0,j=1;j<L.length;j++){
        if(L.data[i]!=L.data[j])
            L.data[++i] = L.data[j];
        L.length=i+1;
        return true;
    }
}

/**
 * 题目描述：合并有序顺序表，新表保持有序
*/
bool Merge(SqList A,SqList B,SqList &C){
    if(A.length+B.length>C.MaxSize)
        return false;
    int i=0, j=0, k=0;
    while(i<A.length && j<L.length){
        if(A.data[i]<=B.data[j])
            C.data[k++]=A.data[i++];
        else
            C.data[k++]=B.data[j++];
    }
    while(i<A.length)
        C.data[k++]=A.data[i++];
    while(j<B.length)
        C.data[k++]=B.data[j];
    C.length=k;
    return true;
}
```

```
/**
 * 题目描述:A[m+n]中存放 a1~am,b1~bn,使用算法将其转为b1~bn,a1~am
 * 算法思想:现将整个顺序表逆置为 bn~b1,am~a1
 * 再将A的前n项和后m项分别逆置。
*/
typedef int DataType;
void Reverse ( DataType A[], int left, int right, int arraySize ){
    if( left>=right || arraySize==0 )
        return;
    int min=( left+right )/2;
    for( int i=0; i<mid; i++ ){
        DataType tmp = A[left+i];
        A[left+i] = A[right-i];
        A[right-i] = tmp;
    }
}

void Exchange(DataType A[], int m, int n, int arrarySize){
    Reverse(A, 0,m+n-1,arraySize);
    Reverse(A,0,n-1,arraySize);
    Reverse(A,n,m+n-1,arraySize);
}
```

```
/**
 * 题目描述：递增顺序表，查找x，若找到，与其后继元素交换位置，
 * 若没找到，则插入
 * 算法思想：二分查找
*/
void SearchExchangeInsert(ElemType A[],ElemType x){
    int low=0,high=A.size()-1,mid;
    while( low <= high ){
        mid = (low+high)/2;
        if(A[mid]<x){
            low = mid+1;
        }else{
            high = mid-1;
        }
    }
    if(A[mid]==x && mid!=n-1){  //若和最后一个元素相等，则不存在与其后继交换的操作
        ElemType tmp = A[mid];
        A[mid] = A[mid+1];
        A[mid+1] = tmp;
    }
    if(low>high){       //查找失败
        for(int i=A.size()-1;i>high;i--) A[i+1] = A[i];
        A[i+1] = x;
    }
}

/**
 * 题目描述：数组左移p个单位
 * 算法思想：将问题视为把数组ab转换成数组ba， a代表数组的前p个元素，b代表数组中余下的n-p个元素
 * 现将a逆置为a^-1,将b转为b^-1,再将整体转为ba
*/
void Reverse(int R[], int from, int to){
    int i, temp;
    for(i=0;i<(from+to)/2;i++){
        temp = R[from+i];
        R[from+i] = R[to-i];
        R[to-i] = temp;
    }//Reverse
    void Converse(int R[], int n, int p){
        Reverse(R,0,p-1);
        Reverse(R,p,n-1);
        Reverse(R,0,n-1);
    }
}
```

```
/**
 * eg: S1= (11, 13, 15, 17, 19) ; s2(2,4,6,8,20),给出定义 中位数（长度）L/2 向上取整。eg中s1和s2的中位数是 11
 * 算法思想：二路归并排序，排到第(L+1)/2个停止，即为s1和s2的中位数
*/
int sear_mid(SqList &s1,SqList &s2){
    int count=0;
    int i,j;
    while( i<s1.size() && j<s2.size() && count < (s1.size()+s2.size()+1)/2 ){
        if(s1.data[i]<s2.data[j]){
            i++;
            count++;
        }else{
            j++;
            count++;
        }
    }
    if(s1.data[i]<s2.data[j])
        return s1.data[i];
    else
        return s2.data[j];
}

int Majority(vector<int> nums){
unordered_map<int ,int> m;
    for(auto& v : nums){
        m[v]++;
    }
    int key = m.begin()->first;
    int value = m.begin()->second;
    for(auto it=m.begin();it!=m.end();it++){
        //cout << it->first <<":"<<it->second << endl;
        if(it->second>value){
            key = it->first;
            value = it->second;
        }
    }
    if(value > nums.size()/2)
        return value;
    else
        return << -1;
}
```

```
/**
 * 题目描述：数组中未出现的最小正整数
 * 王道算法：
*/
int findMissMin(int A[],int n){
    int i,*B;
    B = (int*)malloc(sizeof(int)*n);
    memset(B,0,sizeof(int)*n);
    for(i=0;i<n;i++){
        if(A[i]>0 && A[i]<=n)
            B[A[i-1]]=1;
    }
    for(i=0;i<n;i++){
        if(B[i]==0)
            break;
    }
    return i+1;
}

/**
 * 我的算法（未知对错）
*/
int Max_afterZero(vector<int> v){
    int key=1;
    for(auto it=v.begin(); it!=v.end();it++){
        if( key < *it ){
            key=*it;
        }
        if(*it <= 0 && it==v.end()){
            key = 0;
            break;
        }
    }
    return key+1;
}
```