

顺序表

代码题 1

题干

- 题目：

顺序表默认结构体：

```
typedef struct{
    int data[maxsize];
    int length;
} Sqlist;
```

1、顺序表递增有序，插入元素 x，仍递增有序

- 总结：

- 第一步：找到目标位置，返回位置下标
- 第二步：从顺序表的**最后一个**元素开始，将下标后元素全部后移一位（`l.data[j+1]=l.data[j]`），然后插入元素；

解答

```
typedef struct
{
    int data[maxsize];
    int length;
}Sqlist;

int find(Sqlist l,int x)
{
    int i = 0;

    for( ; i<l.length ; i++)
    {
        if(x>l.data[i])
        {
            break;
        }
    }

    return i;
}

void insert(Sqlist& l,int x)
{

```

```

// 当前x需要插入的位置
int index = find(l,x);

// 这个j是大于等于index，因此原本处于index位置的元素同样需要被后移
for(int j = l.length-1 ; j>=index ; j--)
{
    // 把index后的所有元素后移一位
    l.data[j+1] = l.data[j];
}

l.data[index] = x;

// 顺序表中插入元素后，顺序表长度+1
++(l.length);
}

```

代码题 2

题干

- 题目：
 - 2、用顺序表最后一个元素覆盖整个顺序表中最小元素，并返回该最小元素
- 总结：
 - 第一步：遍历、找到最小元素，获得其下标
 - 第二步：将最后一个元素放到最小元素的下标处，并使顺序表长度减一

解答

```

int DeleteTheMin(Sqlist& l)
{
    int min = l.data[0];
    int minIndex = 0;

    for(int i = 0;i<l.length;i++)
    {
        if(l.data[i]<min)
        {
            min = l.data[i];
            minIndex = i;
        }
    }

    l.data[minIndex] = l.data[length-1];
    l.length--;

    return min;
}

```

代码题 3

题干

- 题目:

3、将顺序表中的元素逆置

- 总结:

- 两个指针 `i` 和 `j`，一个指向顺序表头、一个指向顺序表尾，往中间移动；
- 当指针 `i` 指向超过 `mid` 时，停止循环；

解答

```
void Invert(Sqlist& l)
{
    int i = 0;
    int j = l.length - 1;

    for(; i < (l.length)/2; ++i, --j)
    {
        int temp = l.data[j];
        l.data[j] = l.data[i];
        l.data[i] = temp;
    }
}
```

代码题 4

题干

- 题目:

4、将 $(a_1, a_2, a_3, \dots, a_m, b_1, b_2, \dots, b_n)$ 转换成 $(b_1, b_2, \dots, b_n, a_1, a_2, a_3, \dots, a_m)$

- 总结:

- 如果想要在原地、将两个子序列交换位置，可以先将两个子序列分别原地逆置，然后将整个顺序表逆置；

- 过程:

- 先将前 `m` 个元素逆置: `am, am-1, ..., a2, a1`
- 再将后 `n` 个元素逆置: `bn, bn-1, ..., b2, b1`
- 此时: `am, am-1, ..., a2, a1, bn, bn-1, ..., b2, b1`
- 将整个顺序表逆置: `b1, b2, ..., bn, a1, a2, ..., am`

解答

```
void Invert(Sqlist& l, int m, int n)
{
```

```

int i = m;
int j = n;
int mid = (i+j)/2;

for(;i<mid;++i,--j)
{
    int temp = l.data[j];
    l.data[j] = l.data[i]
    l.data[i] = temp;
}

// 原地排列
void Convert(Sqlist &l,int m,int n)
{
    Invert(l,0,m-1);
    Invert(l,m,n-1);
    Invert(l,0,n-1);
}

```

代码题 5

题干

- 题目：

5、删除顺序表中所有值为 x 的元素(两种方法)

- 总结：

- 删除所有值为 x 的元素等价于将所有值不为 x 的元素往前覆盖；
- 两个指针：i 指针，用于遍历；k 指针，用于指向当前覆盖的位置；

- 注意：

- k 指向的是下一次要覆盖的位置；
- 如果当前值等于 x 时，k 停在 x 的位置不走。因此 i 往后走后、会将其不等于 x 的数值放到 k 的位置；

- 实现：

- 要删除 4，一共：1,4,5,6
- 第一次：i 指向 1，不等于 4，将 k 指向的 1 覆盖到 1，并且 k++、指向 4；
- 第二次：i 指向 4，等于 4，k 不移动、依然指向 4
- 第三次：i 指向 5，不等于 4，将 5 覆盖到 k 指向的位置，即 4；

解答

```

void DeleteNumber(Sqlist& l,int x)
{
    int i = 0;
    int k = 0;

    for(;i<l.length;i++)

```

```

{
    if(l.data[i]!=x)
    {
        l.data[k] = l.data[i];
        k++;
    }
    if(l.data[i]==x)
    {
        // 如果当前指向的元素为x时，则k指针不跟着往前走，而是停在元素x的位置；
        // 当下一个元素不是x时，另l.data[i]的数值覆盖至k的位置
    }
}

l.length = k;
}

```

代码题 6

题干

- 题目：
6、从顺序表中删除给定值在 s 到 t 之间（包含 s 和 t）的所有元素
- 总结：
 - 和代码题 5 的思想一致，只需要更改其判断是否覆盖的逻辑；

解答

```

void DeleteNumber(Splist& l,int s,int t)
{
    int i = 0;
    int k = 0;

    for(;i<l.length;i++)
    {
        if(l.data[i]<s || l.data[i]>t)
        {
            l.data[k] = l.data[i];
            k++;
        }
    }

    l.length = k;
}

```

代码题 7

题干

- 题目:

• 7、从有序表中删除所有值重复的元素

- 总结:

- 有两根指针，指针 i 用于遍历所有元素，指针 k 指向当前覆盖的位置；
- 当 i 指针指向的值和 k 指针指向的值内容不一样时，表示为不重复的元素，放入 k 的下一个位置；
- 当 i 指针指向的值和 k 指针指向的内容一样时，表示为重复的元素，`i++`，`k` 不变；

解答

```
void DeleteSameNumber(Sqlist& l)
{
    int i=1;
    int k=0;

    if(l.length <=1)
    {
        return;
    }

    for( ; i<l.length ; i++)
    {
        if(l.data[i] != l.data[k])
        {
            k++;
            l.data[k] = l.data[i];
        }
    }

    l.length = k+1; // 由于当前k指向的是当前顺序表经过删除之后的最后一个元素，因此这里要+1才表示长度
}
```

代码题 8

题干

- 题目:

• 8、两个递增有序表合并成一个递增有序表

- 总结:

- 三根指针，指针 i 指向有序表 1，指针 2 指向有序表 2，指针 3 指向新表；
- 指针 i 和 j 指向的内容进行比较，谁小、就把谁放进指针 3 指向的新表中；

解答

```
Sqlist* Merge(Sqlist& l1,Sqlist& l2,Sqlist& l3)
{
```

```

l3.length = l1.length + l2.length;
int i = j = k = 0;

for(; i<l1.length && j<l2.length ;)
{
    if(l1.data[i]<l2.data[j])
    {
        l3.data[k] = l1.data[i];
        i++;
        k++;
    }
    else
    {
        l3.data[k] = l2.data[j];
        j++;
        k++;
    }
}

while(i<l1.length)
{
    l3.data[k] = l1.data[i];
    k++;
    i++;
}

while(j<l2.length)
{
    l3.data[k] = l2.data[j];
    k++;
    j++;
}

l.length = k;
}

```

代码题 9

题干

- 题目：
 - 9、求两个递增序列合并后的中位数(两种方法)
- 总结：
 - 第一步：将两个序列合并为一个递增序列
 - 第二步：计算两个序列的长度相加除以二，得到中位数的位置；
- 注意：
 - 计算两个序列的长度相加时，还要减去 1，因为下标是从 0 开始的；

解答

```

void Merge(Sqlist& l1,Sqlist& l2,Sqlist& l3)
{
    int i = j = k = 0;
    while(i<l1.length&& j<l2.length)
    {
        if(l1.data[i]<l2.data[j])
        {
            l3.data[k] = l1.data[i];
            i++;
            k++;
        }
        else
        {
            l3.data[k] = l2.data[j];
            j++;
            k++;
        }
    }
    while(i<l1.length)
    {
        l3.data[k] = l1.data[i];
        i++;
        k++;
    }
    while(j<l2.length)
    {
        l3.data[k] = l2.data[j];
        j++;
        k++;
    }
    l3.length = k;
}

int GetMid(Sqlist l1,Sqlist l2)
{
    Sqlist l3;
    l3.length = l1.length+l2.length;

    Merge(l1,l2,l3);
    int mid = (l1.length+l2.length-1)/2;
    return mid;
}

```

代码题 10

题干

- 题目：

10、设计一个时间上尽可能高效的算法，找出数组中未出现的最小正整数

41. (13 分)给定一个含 $n(n \geq 1)$ 个整数的数组，请设计一个在时间上尽可能高效的算法，找出数组中未出现的最小正整数。例如，数组 $\{-5, 3, 2, 3\}$ 中未出现的最小正整数是 1；数组 $\{1, 2, 3\}$ 中未出现的最小正整数是 4。要求：

- (1) 给出算法的基本设计思想。
- (2) 根据设计思想，采用 C 或 C++ 语言描述算法，关键之处给出注释。
- (3) 说明你所设计算法的时间复杂度和空间复杂度。

• 总结：

- 如果当前数组的长度为 n ，则当前能够出现的最大最小正整数只能是 $n+1$
 - 比如当前数组长度为 5，元素为 $1, 2, 3, 4, 5$ ，则此时最小正整数为 6；
 - 比如当前数组长度为 5，元素为 $1, 2, 100, 4, 5$ ，则此时最小正整数为 3；
 - 比如当前数组长度为 5，元素为 $100, 200, 300, 400, 500$ ，则此时最小正整数为 1；
- 因此可以用一个长度为 n 的记录数组，用来记录大小在 $1 \sim n$ 这个范围内的元素有没有出现。如果出现，就在对应的下标处将其记为 1。最后只要遍历这个记录数组，找到的第一个值为 0 的元素即为最小正整数；

解答

```
int GetNullMinNumber(int a[], int n)
{
    int* record = new int[n];
    for(int i = 0; i < n; i++)
    {
        record[i] = 0;
    }

    for(int i = 0; i < n; i++)
    {
        // 由于是返回最小正整数，因此a[i]需要大于0
        // 并且当前n代表的是长度，因此n就代表了当前n个元素可以存在的最大最小整数
        // 比如当n=3时，当前有3个元素，因此最大的最小正整数为3，因此要小于n+1
        if(a[i] < n+1 && a[i] > 0)
        {
            record[a[i]-1] = 1;
        }
    }

    int minNumber = 0;
    for(; minNumber < n; minNumber++)
    {
        // 当a[minNumber]==0时，表示当前下标为minNumber的数组元素其没有出现，因此其为
        // 最小未出现整数
        if(a[minNumber] == 0)
        {
            break;
        }
    }
}
```

```

    }

    Delete[] Record;
    return (maxNumber+1);
}

```

代码题 11

题干

- 题目:

11、若一个整数序列中有过半相同元素，则称其为主元素，设计算法找出数组 $A(a_0, a_1, \dots, a_{n-1})$ 的主元素。（其中 $0 < a_i < n$ ）若存在主元素则输出，否则返回 -1

- 总结:

- 思想和代码题 10 类似;
- 因为数组 A 中的元素大小都不大于数组长度 n ，因此可以用一个长度为 n 的记录数组，记录数组 A 中**每个元素的出现次数**;
- 最后只要遍历这个数组，检查记录数组中有没有元素大于 $n/2$ 即可;

解答

```

int MainNumber(int a[],int n)
{
    int* record = new int[n];
    for(int i = 0;i<n;i++)
    {
        record[i] = 0;
    }

    for(int i = 0;i<n;i++)
    {
        record[a[i]-1]++;
    }

    for(int i = 0;i<n;i++)
    {
        if(record[i]>(n/2))
        {
            Delete[] record;
            return i+1;
        }
    }

    Delete[] record;
}

```

```
return -1;  
}
```

获取笔记持续更新可加微信: WandD2277