顺序表和链表

1.线性表

2.顺序表

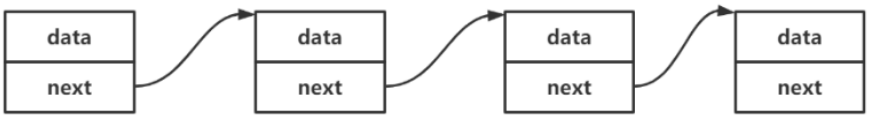
3.链表

4.顺序表和链表的区别和联系

1.线性表

> 线性表 *(linear list)* 是n个具有相同特性的数据元素的有限序列。 线性表是一种在实际中广泛使用的数据结构，常见的线性表：顺序表、链表、栈、队列、字符串...
>
> 线性表在逻辑上是线性结构，也就说是连续的一条直线。但是在物理结构上并不一定是连续的，线性表在物理上存储时，通常以数组和链式结构的形式存储。

顺序表



2.顺序表

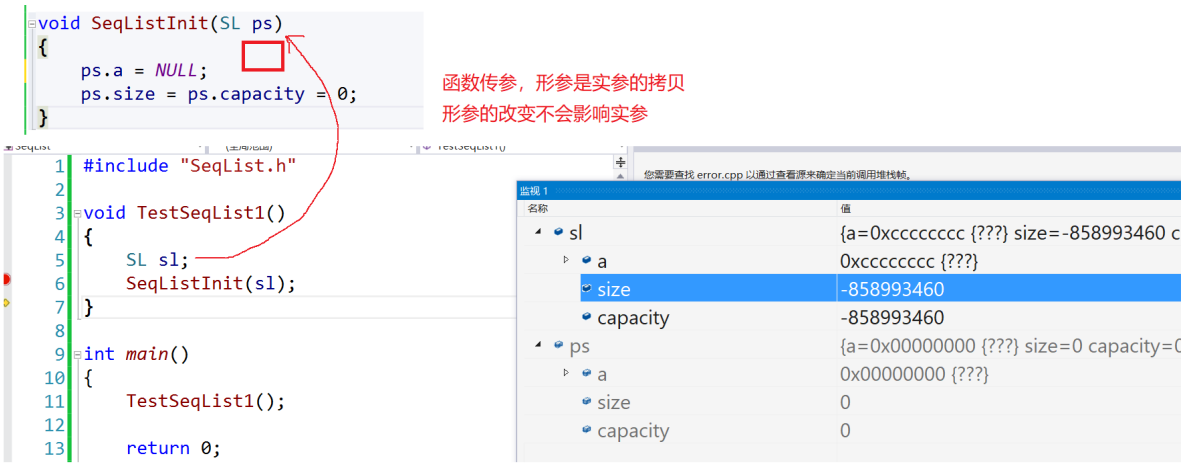
2.1概念及结构

结构是数组，但是对数组还有一点要求

顺序表就是数组，但是在数组的基础上，它还要求数据是从头开始存连续存储的，不能跳跃间隔。

顺序表是用一段物理地址连续的存储单元依次存储数据元素的线性结构，一般情况下采用数组存储。在数组 上完成数据的增删查改。 顺序表一般可以分为： 1. 静态顺序表：使用定长数组存储元素。

```c
void SeqListInit(SL* ps)
{
    ps->a = NULL;
    ps->capacity = ps->size = 0;
}
void SeqListInit(SL* ps);
//void SeqListInit(SL ps); //形参的改变不会影响实参
void TestSeqList1()
{
    SL s1;//改变结构体需要传结构体地址过去
    SeqListInit(&s1);
}
```



调试帮助你观察程序，最终分析的还是你自己。

调试过程中如果有循环很大在循环处下面F9打一个断点，F5运行到那里

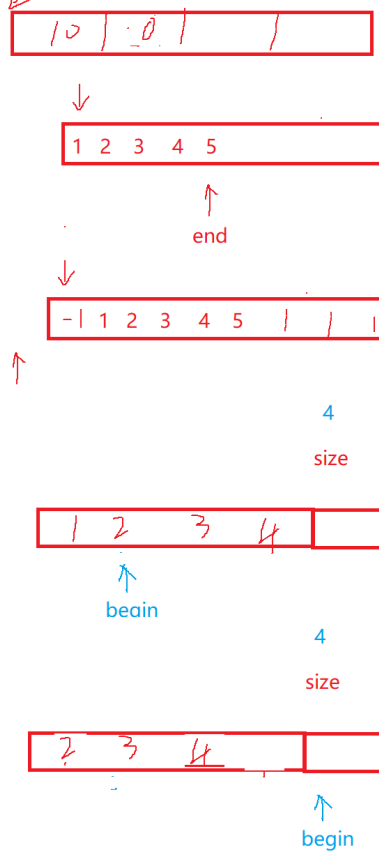
调试时不小心错过了某行代码，怎么返回上一句，用鼠标将它拖回上一行，但是覆水难收，程序并不会退回到原来的那里，

```c
SLDataType* tmp = (SLDataType*)realloc(ps->a, sizeof(SLDataType) *
newcapacity);
        if (tmp == NULL)
        {
            printf("realloc fail\n");
            exit(-1);//开辟失败直接终止程序
        }
```

意义在于比如malloc 开辟内存返回的指针总是不为空指针，里面的程序也无法进入，拖动执行点可以进入代码里面，看一看执行的结果

一般情况下不要生拉硬拽，比如说那一行拖回去基本不影响，赶紧拖回去重新来。错过了就错过了 覆水难收，非常之时才会用它

```
typedef struct SeqList
{
    SLDataType* a;
    int size;
    int capacity;
}SL;
```




```
void SeqListPushFront(SL* ps, SLDataType x)
{
    SeqListCheckCapacity(ps);

    // 挪动数据
    int end = ps->size - 1;
    while (end >= 0)
    {
        ps->a[end + 1] = ps->a[end];
        --end;
    }
    ps->a[0] = x;
    ps->size++;
}
```

```
void SeqListPopFront(SL* ps)
{
    assert(ps->size > 0);

    // 挪动数据
    int begin = 1;
    while (begin < ps->size)
    {
        ps->a[begin - 1] = ps->a[begin];
        ++ begin;
    }

    ps->size--;
}
```

oj

服务器会有主函数和接口函数合并编译链接执行，报错会把运行报错返回来

OJ分类和原理

1、IO型
2、接口型

1、不需要写头文件、主函数等等
2、提交了以后，会跟oj服务器上他准备好的代码合并，再编译运行。

27. 移除元素

难度 简单 1046

给你一个数组 `nums` 和一个值 `val`，你需要 原地 移除所有数值等于 `val` 的元素，并返回移除后数组的新长度。

不要使用额外的数组空间，你必须仅使用 `O(1)` 额外空间并 原地 **修改输入数组**。

元素的顺序可以改变。你不需要考虑数组中超出新长度后面的元素。

```
1
2
3 int removeElement(int* nums, int numsSize, int val){
4
5 }
```

接口型

测试用例：通过参数过来的
结果：一般是通过返回值拿的，也有可能是输出型参数

可能会有多组测试用例

```
int* singleNumbers(int* nums, int numsSize, int* returnSize){
    int* arr = (int*)malloc(sizeof(int)*2);
    *returnSize = 2;

    return arr;
}

int main()
{
    int arr[10] ={...};
    int size = 0;
    int* a = singleNumbers(arr,10, &size);
```

输出型参数

返回数组长度，都默认不知道数组长度，力扣有几百道oj题，为了统一进行测试，返回数组长度。

KY11 二叉树遍历

算法知识视频讲解

较难 通过率：33.86% 时间限制：1秒 空间限制：64M

知识点：树 搜索

题目 题解(8) 讨论(130) 排行

校招时部分企业笔试将禁止编程题跳出页面，为提前适应，练习时请使用在线自测，而非本地IDE。

描述

编一个程序，读入用户输入的一串先序遍历字符串，根据此字符串建立一个二叉树（以指针方式存储）。
例如如下的先序遍历字符串： ABC##DE#G##F### 其中“#”表示的是空格，空格字符代表空树。建立起此二叉树以后，再对二叉树进行中序遍历，输出遍历结果。

输入描述：

输入包括1行字符串，长度不超过100。

输出描述：

可能有多组测试数据，对于每组数据，输出将输入字符串建立二叉树后中序遍历的序列，每个字符后面都有一个空格。每个输出结果占一行。

示例1

输入：abc##de#g##f###

输出：c b e g d f a

1、我们要自己写头文件、main函数等等
2、测试用例输入：我们要去scanf获取
3、测试结果：我们printf输出

C ACM模式

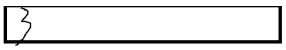
```
1
```

数据结构需要先思考，思考出多种做法，选择最优解，

面试过程面试官就是不断的向你提各种各样的要求，直到你答不出来或者没有更优解，这种方式压力面试，不断优化时间复杂度和空间复杂度，不断给你压力的情况下，看你能力的深度是多少。

一步步慢慢优化， 早期的安卓系统很挫，现在安卓系统很可以。做不出最好的先把最暴力的做出来，在一步步尝试看懂别人的，不断学习，在积累中不断进步。不要想着一口吃一个大胖子，

3



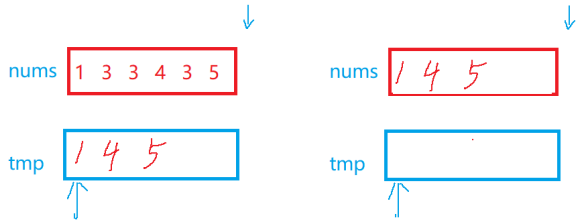
思路1：找到所有的val，一次挪动数据覆盖删除

时间复杂度：O(N^2) 最坏的情况：数组中大部分值甚至全部都是val

n-1
n-2
n-3
...

能否时间复杂度优化到O(N)　　空间复杂度：O(N)

思路2：一次遍历nums数组，把不是val的值，放到tmp数组，再把tmp数组的值拷贝回去



能否时间复杂度优化到O(N)　空间复杂度优化为O(1)?

思路3：

1、src去找nums数组中不等于val的值，放到dst指向的位置去，再++src，++dst


```
2
3  int removeElement(int* nums, int numsSize, int val){
4      int src = 0, dst = 0;
5      while(src < numsSize)
6      {
7          if(nums[src] != val)
8          {
9              nums[dst] = nums[src];
.0             src++;
.1             dst++;
.2         }
.3         else
.4         {
.5             src++;
.6         }
.7     }
.8
.9     return dst;
!0 }
```

提交记录

113 / 113 个通过测试用例

执行用时: **0 ms**

内存消耗: **5.8 MB**