计算概论习题课

韩喆

WIP@ICST

20151221

简介

讲最近三次上机内容

综合练习

~~转输出矩阵~~
~~不能一起吃的食物~~
例题 (15.5) 算 24 (1103)
进制转换
~~计算反序数~~
~~放苹果问题~~
~~习题 (14-3) 奇数单增序列~~
~~1090 分解因数~~
习题 (15-1) 前缀表达式 (1010)
~~文字排版~~
~~验证子串~~

结构体与链表

删除数组中的元素（链表）
统计学生信息（使用动态链表完成）
~~谁拿了最多奖学金~~
班级学生成绩统分
链表合并排序

模拟

~~满足条件的数累加~~
字母次数
~~求均方差 (用指针完成)~~
提取数字串按数值排序
二维数组回形遍历
扩号匹配问题
连续整数区域上确界

期末上机模拟

— 1214

字母次数

题目 - 字母次数

来源	元培-From Whf

描述

有一串符号序列（不超过300个符号），请统计其中的英文字母各自出现的次数，并按字母顺序输出出现的英文字母及其出现次数，大小写分别对待。如果没有字母，则输出 No.

关于输入

一串符号，在一行内输入。

关于输出

每行输出一个字母及其次数，其间，以等号连接。如果没有英文字母，则输出 No

例子输入

```
ab123c52,f/b8AD9b#c
```

例子输出

字母次数

题目 - 字母次数

来源	元培-From Whf
描述	
有一串符号序列（不超过300个符号），请统计其中的英文字母各自出现的次数，并按字母顺序输出出现的英文字母及其出现次数，大小写分别对待。如果没有字母，则输出 No.	
关于输入	
一串符号，在一行内输入。	
关于输出	
每行输出一个字母及其次数，其间，以等号连接。如果没有英文字母，则输出 No	
例子输入	
ab123c52,f/b8AD9b#c	
例子输出	

- 如何判断字符等价？判读字符在某个范围？

```
ain()

ar str[301], str1[52]={'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','
t i, j, n, log, a[52]={0};
```

字母次数

```
int A[30],a[30];//分别存放大写字符，小写字符的出现次数
for(int i = 0 ; i < 30; i ++)
        a[i] = A[i] = 0;//别忘了初始化
bool ex = false;//判断是否有字母出现
for(int i = 0 ; c[i] != '\0'; i ++){
        if(c[i] >='a' && c[i] <='z'){
                a[c[i]-'a'] ++;
                ex =true;
        }
        if(c[i] >='A' && c[i] <='Z'){
                A[c[i]-'A'] ++;
                ex = true;
        }
}
```

- A[1]=A['B'-'A'] 表示'B'的出现次数
- a[3]=A['d'-'a'] 表示'd'的出现次数

提取数字串按数值排序

题目 - 提取数字串按数值排序

来源	元培-From Whf

描述

输入一串不超过300个字符的符号序列，请将其中的所有数字串提出，并将每个数字串作为整数看待（假设可以用int表示），按从小到大顺序输出结果，输出的整数之间以逗号间隔。如果没有数字，则输出0；例如：
*1234.345#6781ad9jk81-11101?aght88ir09kp，其中的整数包括：1234，345，6781，9，81，11101，88，9，从小到大排序后，应该输出：
9，9，81，88，345，1234，6781，11101

关于输入

在一行内输入一串符号

关于输出

从小到大排序的整数序列，如果没有数字，则输出0；

例子输入

```
*1234.345#6781ad9jk81-11101?aght88ir09kp
```

例子输出

```
9,9,81,88,345,1234,6781,11101
```

提取数字串按数值排序

- 一个指针顺序扫每个字符
- 一个变量保存当前的数字大小（小于 0 表示没有数字）
- 扫到数字之后的下一个字符时把刚刚存储的数字加入数组，并把变量置为-1
- 最后对数组排序

```
int t = -1;//记录当前的数值
int m[400],n=0;
for(int i = 0 ; c[i] != '\0'; i ++){
    if(c[i] >='0' && c[i] <='9'){
        if(t < 0)//第一个数字
            t = 0;
        t = t*10 + c[i]-'0';//前面已经有数字了
    }
    else if(t>=0){//数字读完了，加入数组
        m[n++] = t;
        t=-1;//恢复初始值
    }
}
if(t>0)//最后结尾是数字，需要再加入数组
    m[n++] = t;
```

提取数字串按数值排序

- 这是我随便找的**没有过的例子**
- 这个同学至少犯了以下**错误**

1. 数组开的不够大（至少 **301**，建议330+）
2. 数组 a没有初始化
3. 字符和整数的关系没有搞明白
   - 'c'-'a'=2
   - 'a'-2=95
   - str[i] = '0'=48
4. $a[j+1] = a[j] * 10 + \ldots$ 两者实际上没关系，这个式子肯定有问题
5. 输出是 a[0] 还是 a[1] 开始？

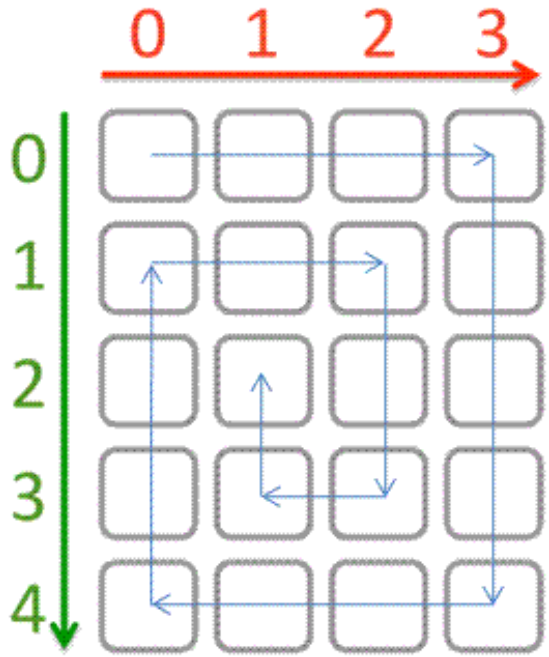
```
#include<stdio.h>
#include<string.h>
int main()
{
    int n, t, i, j, a[300];
    char str[300];
    scanf("%s",&str);
    n=strlen(str);
    j=0;
    for(i=0;i<n;i++){
       if(str[i]>=0&&str[i]<=9){
          a[j+1]=a[j]*10+str[i];
       }else j++;
    }
    n=j;
    for(i=1;i<=n;i++){
       for(j=i+1;j<=n;j++){
          if(a[j]<a[i]){ t=a[i];a[i]=a[j];a[j]=t;}
        }
     }
     printf("%d",a[1]);
     for(i=2;i<=n;i++) printf(",%d",a[i]);
     return 0;
}
```

5. 二维数组回形遍历

来源	北京大学2009年医学部练习题

描述

给定一个row行col列的整数数组array，要求从array[0][0]元素开始，按回形从外向内顺时针顺序遍历整个数组。如图所示：

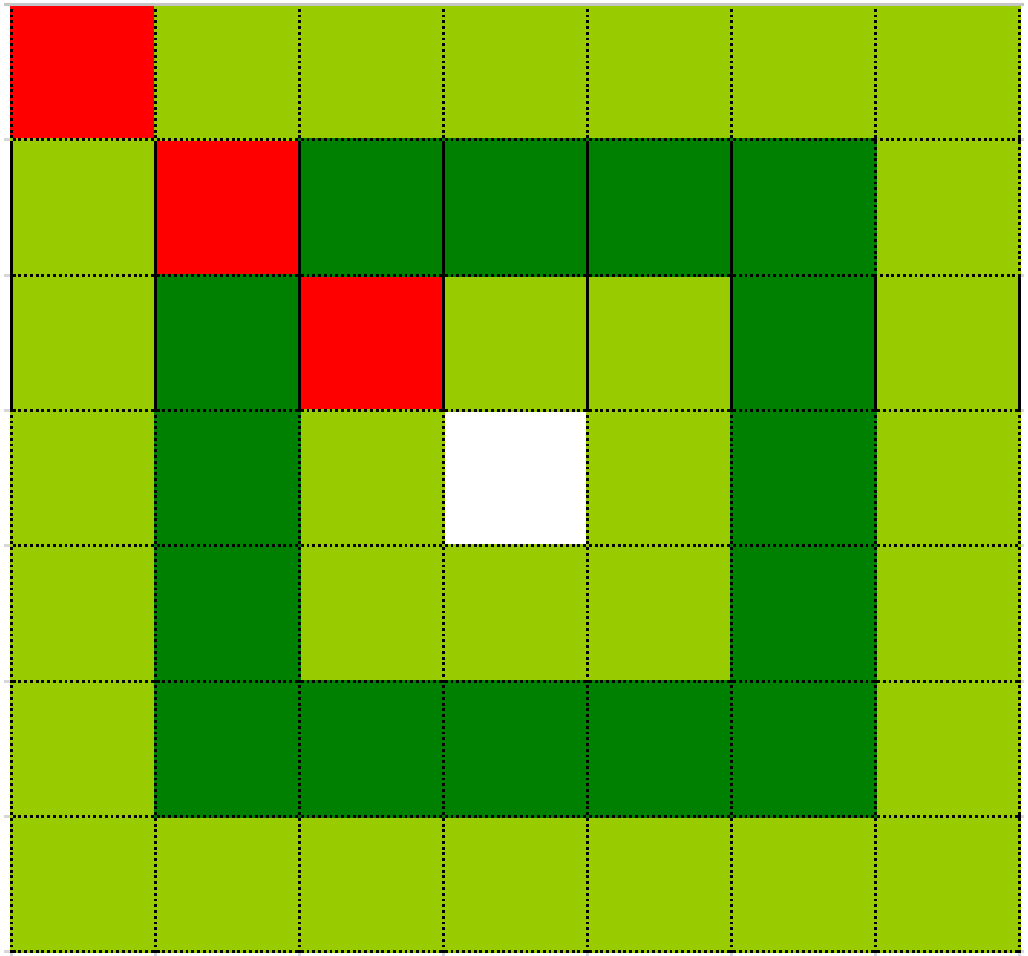


关于输入

输入的第一行上有两个整数，依次为row和col。
余下有row行，每行包含col个整数，构成一个二维整数数组。
（注：输入的row和col保证0<row<100, 0<col<100）

5. 二维数组回形遍历

- 每次从左上角开始顺时针扫描一圈
- 移动起始点，继续做
- 考虑清楚结束条件
- 考虑清楚每圈的行坐标，列坐标



5. 二维数组回形遍历

- 模拟真实行走过程
- 开始位置 + 方向
- 如果改方向下一个格子不在棋盘内或者下一个格子走过了，改变方向
- 如何改变方向？(下一个方向是?)
  - $(0,1)->(1,0)->(0,-1)->(-1,0)->(0,1)$
- 判断在棋盘内且没有走过?
- 停止条件?
- 全局变量: 函数共用

```
int a[210][210];//记录每个点的数值
bool v[210][210];//记录改点是否走过
int r,c;//行数，列数
// (x,y)是刚刚走过的点，(dx,dy)是当前方向，cnt是走过的格子数
void find(int x, int y, int dx, int dy, int cnt){
  int nx = x+dx;
  int ny = y+dy;
  if(cnt >= r*c)
    return;
  // 下一个点可以走
  if( nx>=0 && nx<r && ny >=0&& ny<c && v[nx][ny] == false){
    printf("%d\n", a[nx][ny]);
    v[nx][ny] = true;
    find(nx, ny, dx, dy, cnt+1);//递推继续走
  }
  else{//不能走，需要转换方向
    // 0,1 -> 1,0 -> 0,-1->-1,0  ->0,1
    if(dx == 0 && dy == 1){
      dx = 1; dy = 0;
    }
    else if(dx == 1 && dy == 0){
      dx = 0; dy = -1;
    }
    else if(dx == 0 && dy == -1){
      dx = -1; dy = 0;
    }
    else {
      dx = 0; dy = 1;
    }
//    int ody = dy;
//    dy = dy == 0 ? -dx: 0;
//    dx = ody;
    find(x,y,dx,dy,cnt);
  }
}
```

6. 扩号匹配问题

在某个字符串（长度不超过100）中有左括号、右括号和大小写字母；规定（与常见的算术式子一样）任何一个左括号都从内到外与在它右边且距离最近的右括号匹配。写一个程序，找到无法匹配的左括号和右括号：首先输出原来字符串，下一行是和原字符串等长的一行，标出不能匹配的括号，其中不能匹配的左括号用"$"标注,不能匹配的右括号用"?"标注

关于输入

第一行一个正整数n,表示数据的组数。后面n行，每组数据一行，包含一个字符串，只包含左右括号和大小写字母，字符串长度不超过100

关于输出

对每组输出数据，输出两行，第一行包含原始输入字符，第二行由"$","?"和空格组成，与第一行等长，"$"和"?"表示与之对应的左括号和右括号不能匹配。

注意：即使所有括号都匹配，第二行也要输出等长的一行空格

例子输入

```
2

((ABCD(x)

)(rttyy())sss)(
```

例子输出

```
((ABCD(x)
```

6. 扩号匹配问题

- 记录左括号的位置
- 每找到一个右括号，去查看之前的所有左括号，找最近的没有匹配的左括号匹配

```
#include <stdio.h>
int main(){
  int k;
  scanf("%d",&k);
  char c[200];
  while(k--){
    scanf("%s",c);
    int li[300];//l[i]表示第i个左括号的状态，1为未匹配，2为匹配
    int ri[300];//ri[i]表示第i个左括号所在位置
    int mark[300];//mark[i]表示第i个字符的状态，1表示被匹配了
    for(int i = 0 ; i < 300; i ++)
      li[i] = ri[i] = mark[i] = 0;
    int ln =0, rn = 0;
    for(int i = 0  ; c[i] != '\0'; i++){
      if(c[i] == '('){//记录找到的左括号
        ri[ln] = i;//记录位置
        li[ln++] = 1;//初始化为未匹配
      }
      else if(c[i] == ')'){//找到右括号，开始匹配
        int tl = ln-1;//从后往前查找没有被匹配的左括号
        while(tl >=0 && li[tl] == 2)
          tl --;
        if(tl >=0){//找到了没有被匹配的左括号
          li[tl] = 2;
          mark[ri[tl]] = 1;//更新左括号为“匹配了“
          mark[i] = 1;//更新右括号为”匹配了“
        }
      }
    }
    printf("%s\n",c);
    for(int i = 0 ; c[i] !='\0'; i ++){
      if(c[i] == '(' && mark[i] == 0)
        printf("$");
      else if(c[i] == ')' && mark[i] == 0)
        printf("?");
      else printf(" ");
    }
    printf("\n");
  }
}
```

7. 连续整数区域上确界

已知集合中共有n 个不同元素(最小元素为1)。从中最多取k次元素（可以取相同元素，也可以少于k次），计算能够**覆盖从1开始的最大连续整数区域的上确界**，也就是最大值（中间不能间断）。

如4个元素的集合{1,3,5,6}，k=2，则连续区域为 1 - 12，中间的任意整数都可以覆盖，因此，上确界为12。因为：
1=1 ：直接取1
2=1+1：取2次1
3=3：直接取3
4=3+1：取1和3
5=5：直接取5
6=6：直接取6
7=6+1：取6和1
8=5+3：取5和3
9=6+3：取6和3
10=5+5：取2次5
11=5+6：取6和5
12=6+6：取2次6

关于输入

第1行为元素个数n和 k值，其间以逗号间隔。假设元素个数不多于20。
其后有若干行，每行表示一个集合的 n 个元素(元素已按从小到大排列)，元素间用空格间隔
最后1行为0，表示结束

关于输出

每行集合对应的最大连续区间的最大值

7. 连续整数区域上确界

- 从 sum=1 开始，如果使用 k 次数组得不到和为 sum，则停止
- 递归：至多使用 k 次等价于至多使用 k-1 次得到和为 sum-a[i]

```
int n, k;
int a[30];
// n个数字放在在a数组，最多用times次能否拼成sum
bool get(int sum, int times){
  if(sum == 0 && times >= 0)
    return true;
  // 次数不够了 or 全部用最大的都不够
  if(times <=0 || sum > times*a[n-1])
    return false;
  // 递推查找
  for(int i = n-1; i >=0 ; i--)
    if(sum >= a[i] && get(sum-a[i], times-1))
      return true;
  return false;
}
```

结构体与链表

— 1207

1. 删除数组中的元素（链表）

给定N个整数，将这些整数中与M相等的删除
假定给出的整数序列为：1,3,3,0,-3,5,6,8,3,10,22,-1,3,5,11,20,100,3,9,3
应该将其放在一个链表中，链表长度为20
要删除的数是3，删除以后，链表中只剩14个元素：1 0 -3 5 6 8 10 22 -1 5 11 20 100 9

要求：必须使用链表，不允许使用数组,也不允许不删除元素直接输出
程序中必须有链表的相关操作：建立链表，删除元素，输出删除后链表中元素，释放链表
不符合要求的程序即使通过，也会算作0分

关于输入

输入包含3行：
第一行是一个整数n(1<=n<=200000)，代表数组中元素的个数。
第二行包含n个整数，代表数组中的n个元素。每个整数之间用空格分隔；每个整数的取值在32位有符号整数范围以内。
第三行是一个整数k，代表待删除元素的值（k的取值也在32位有符号整数范围内）。

关于输出

输出只有1行：
将数组内所有待删除元素删除以后，输出数组内的剩余元素的值，每个整数之间用空格分隔。

例子输入

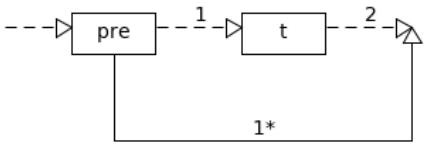
```
20

1 3 3 0 -3 5 6 8 3 10 22 -1 3 5 11 20 100 3 9 3

3
```

例子输出

```
1 0 -3 5 6 8 10 22 -1 5 11 20 100 9
```

1. 删除数组中的元素（链表）

- 删除元素需要直到前一个元素
- 删除的是第一个元素呢？




```
#include <stdio.h>
#include <stdlib.h>
struct node{//链表结构
  int v;
  node * next;
};
int main(){
  int k;
  scanf("%d", &k);
  node *head=NULL, *t, * pre=NULL;
  while(k--){//读取每个元素
    t = (node*)malloc(sizeof(node));
    if(pre == NULL){//是第一个元素
      pre = head = t;//需要初始化head
    }
    else pre->next = t;
    scanf("%d", &(t->v));
    pre = t;
  }
  t->next = NULL;//最后一个元素的next置NULL
  int dv;
  scanf("%d", &dv);//读取待删除元素
  // 删除1：删除开头的连续dv
  while(head != NULL && head->v == dv){
    t = head->next;
                free(head);
    head = t;
  }
  //这个时候head一定不是dv,一定会输出
  printf("%d", head->v);
  pre = head;
  t = head->next;
  while(t != NULL){//删除剩下的元素
    if(t->v == dv){
      pre->next = t->next;
      free(t);
    }
    else{
      pre = t;
      printf(" %d", t->v);
    }
    t = pre->next;
  }
  return 0;
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
                free(head);
                head = t;
        }
        pre = head;
        if(head == NULL)
                return 0 ;
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
                        pre->next = t->next;
                        free(t);
                        t = pre->next;
                }
                else{
                        pre = t;
                        t = t->next;
                }
        }
        // output
        printf("%d", head->v);
        head = head->next;
        while(head != NULL){
                printf(" %d", head->v);
                head = head->next;
        }
        return 0;
}
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
                free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0 ;
        }
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
                        pre->next = t->next;
                        free(t);
                        t = pre->next;
                }
                else{
                        pre = t;
                        t = t->next;
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
                free(head);
                head = t;
        }
        pre = head;
        if(hea    NULL){
                printf("bo");
                return 0 ;
        }
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
                        pre->next = t->next;
                        free(t);
                        t = pre->next;
                }
                else{
                        pre = t;
                        t = t->next;
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出
3. 还是没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出
3. 还是没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        t->next = NULL;
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0 ;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出
3. 还是没输出
4. 还是没输出

```
int main(){
        int k;
        scanf("%d", &k);
        node *head, *t, * pre;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        t->next = NULL;
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0 ;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出
3. 还是没输出
4. 还是没输出

```
};
int main(){
        int k;
        scanf("%d", &k);
        node *head=NULL, *t, * pre=NULL;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        t->next = NULL;
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0 ;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
```

1. 删除数组中的元素（链表）

我是怎么过的

1. ‘Empty output file‘没输出
2. 还是没输出
3. 还是没输出
4. 还是没输出
5. 过了

```
};
int main(){
        int k;
        scanf("%d", &k);
        node *head=NULL, *t, * pre=NULL;
        while(k--){
                t = (node*)malloc(sizeof(node));
                if(pre == NULL){
                        head = t;
                        pre = t;
                }
                else pre->next = t;
                scanf("%d", &(t->v));
                pre = t;
        }
        t->next = NULL;
        int dv;
        scanf("%d", &dv);
        while(head != NULL && head->v == dv){
                t = head->next;
//              free(head);
                head = t;
        }
        pre = head;
        if(head == NULL){
                printf("bo");
                return 0 ;
        }
        printf("%d", head->v);
        t = head->next;
        while(t != NULL){
                if(t->v == dv){
```

2. 统计学生信息（使用动态链表完成）

利用动态链表记录从标准输入输入的学生信息（学号、姓名、性别、年龄、得分、地址）

关于输入

包括若干行，每一行都是一个学生的信息，如：
00630018 zhouyan m 20 10.0 28#460

输入的最后以"end"结束

关于输出

将输入的内容倒序输出
每行一条记录，按照

学号 姓名 性别 年龄 得分 地址

的格式输出

例子输入

```
00630018 zhouyan m 20 10 28#4600

0063001 zhouyn f 21 100 28#460000

0063008 zhoyan f 20 1000 28#460000

0063018 zhouan m 21 10000 28#4600000
```

2. 统计学生信息（使用动态链表完成）

- 结构体，内部存每一行的信息和之前结构体的地址

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct info{
        char c[300];
        info * front;
};
int main(){
        info * front= NULL;
        while(true){
                info * t = (info*)malloc(sizeof(info));
                if(front != NULL)
                        t->front = front;
                else t->front = NULL;
                gets(t->c);
                front = t;
                if(strcmp(t->c, "end") == 0){
                        break;
                }
        }
        front = front->front;
        while(front != NULL){
                printf("%s\n", front->c);
                front = front->front;
        }
        return 0;
}
```

2. 统计学生信息（使用动态链表完成）

- 为什么我按每个元素读就又是 ‘Empty output file‘

```
struct stu{
        char sex[50];
        int age;
        int grade;
        char num[50];
        char name[50];
        char ad[50];
        struct stu *next;
        };

int main(){
        char str[50];
        struct stu *head, *cnew, *p;
        int i=1;

        while(i){
                cnew = (struct stu *)malloc(LEN);
                scanf("%s", cnew->num);
                if(strcmp(cnew->num, "end")==0) break;
                cnew->next = head;
                head = cnew;
                scanf("%s %s %d %d %s", head->name, head->sex,
                if(i==1) head->next = NULL;
                i++;
        }
        p=head;
        do{
            printf("%s %s %s %d %d %s", p->num, p->name, p->se
                p=p->next;
        }while (p!=NULL);

        return 0;
}
```

2. 统计学生信息（使用动态链表完成）

- 为什么我按每个元素读就又是 ‘Empty output file‘
- 打印第二组样例出来看一下？

```
struct Node
{
       char val[200];
       struct Node *next;
};

int main()
{
    char st[200];
    gets(st);

    struct Node *head,*p;
    head=NULL;
    int nm=0;

    while (strcmp(st,"end"))
    {
          p=(struct Node *) malloc(SIZE);
          strcpy(p->val,st);
          if(nm==0){
              nm = 1;
              if(strcmp(st, "00630018 zhouyan m 20 10 28#4600")){
                  printf("sss\n");
              }
          }

          if (head==NULL)
          {
             head=p;
             head->next=NULL;
          }
          else
          {
              p->next=head;
              head=p;
          }

          gets(st);
    }

    p=head;
    while (p != NULL)
    {
          printf("%s\n",p->val);
          p=p->next;
    }
    return 0;
}
```

2. 统计学生信息（使用动态链表完成）

- 为什么我按每个元素读就又是 ‘Empty output file‘
- 打印第二组样例出来看一下？

```
Case 0: Time = 3ms, Memory = 264kB.
Case 1: Time = 2ms, Memory = 264kB.

Different: Different at line 1.
Std: '12346177 abcdefghijklmnopq m 17 85.3 sfajk28#745'
Out: 'sss'

Input of in case 1:
12345678 abcdefghijklmnopq m 1 99.9 sfajk28#246
12345679 abcdefghijklmnopq f 2 99.8 sfajk28#247
12345680 abcdefghijklmnopq m 3 99.7 sfajk28#248
12345681 abcdefghijklmnopq m 4 99.6 sfajk28#249
12345682 abcdefghijklmnopq f 5 99.5 sfajk28#250
12345683 abcdefghijklmnopq m 6 99.4 sfajk28#251
12345684 abcdefghijklmnopq f 7 99.3 sfajk28#252
12345685 abcdefghijklmnopq m 8 99.2 sfajk28#253
12345686 abcdefghijklmnopq m 9 99.1 sfajk28#254
12345687 abcdefghijklmnopq f 10 99 sfajk28#255
```

4. 班级学生成绩统分

- 注意保序（等于时不替换）
- 全排序一定会超时
- 冒泡 3 次得到前三大的数字后结束

5. 链表合并排序

现在给出两个整数数列，先要将其合并为一个数列，并且合并后整个数列有序（从小到大），例如：
数列1： 3 1 2 5 9
数列2： 8 4 7 10 11
合并后得到数列：1 2 3 4 5 7 8 9 10 11

关于输入

输入包括4行
第一行是1个整数N(N<=5000)，表示数列1包含的整数个数
第二行是N个整数，表示数列1中的整数
第三行是1个整数M(M<=5000)，表示数列2包含的整数个数
第四行是M个整数，表示数列2中的整数

关于输出

输出包括1行：合并后的序列

例子输入

```
5

3 1 2 5 9

5

8 4 7 10 11
```

例子输出

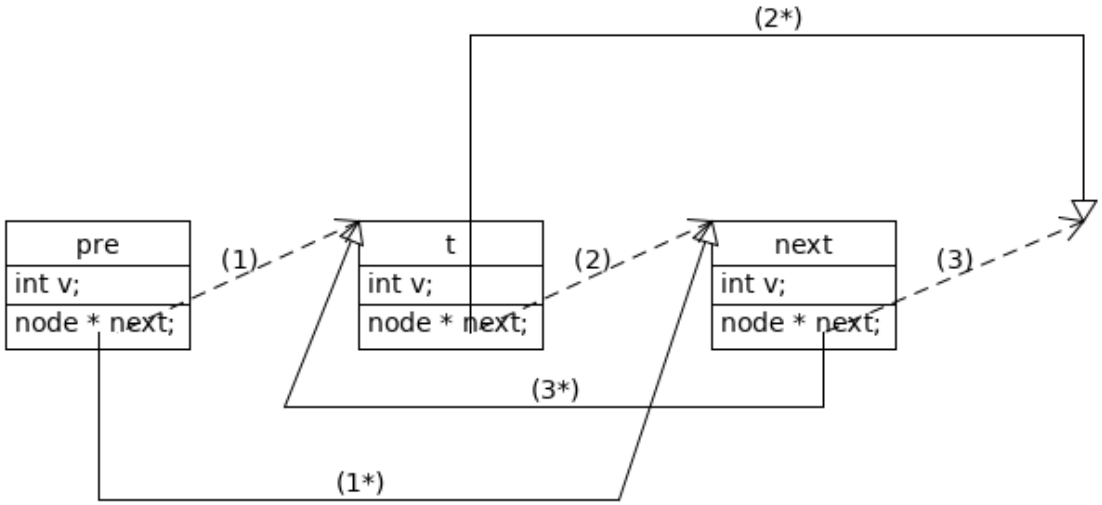
```
1 2 3 4 5 7 8 9 10 11
```

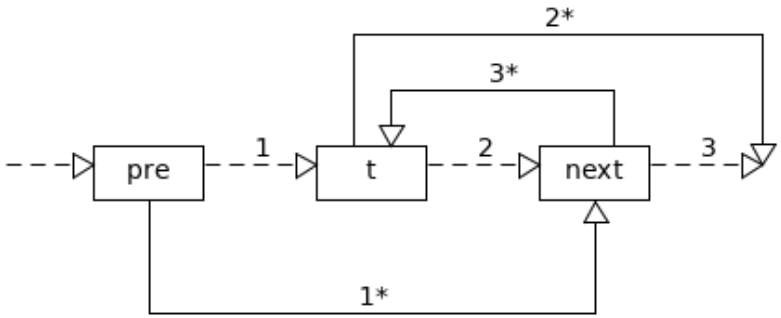
提示

先将两个链表排序，然后再合并
也可以先合并两个链表再排序
注意：是链表操作，不是数组操作。输出是从表头开始遍历每一个元素。

5. 链表合并排序

- 先合并，再排序
  item 冒泡排序需要知道前一个节点





5. 链表合并排序

我是怎么调试的

```
#include <stdio.h>
#include <stdlib.h>
struct node{
        int v;
        node * next;
};
int main(){
        int k;
        node * head = NULL, * pre = NULL;
        int sum = 0;
        scanf("%d", &k);
        sum += k;
        while(k--){
                node * t = (node*)malloc(sizeof(node));
                scanf("%d", &t->v);
                if(head == NULL){
                        head = pre = t;
                }
                else pre->next = t;
                pre = t;
        }
        scanf("%d", &k);
        sum += k;
        while(k--){
                node * t = (node*)malloc(sizeof(node));
                scanf("%d", &t->v);
                if(head == NULL){
                        head = pre = t;
                }
                else pre->next = t;
                pre = t;
        }
//      node * thead = head;
//      printf("%d", thead->v);
//      while(thead->next != NULL){
//              printf(" %d", thead->next->v);
//              thead = thead->next;
//      }
//      printf("\n");
```

1.读取完毕后输出看是否正确

```
        //sort
        for(int i = sum-1; i>=1; i--){
                node * t = head;
                pre = NULL;
//              printf("%d\n", i);
                int k = i;
                while(k--){
//                      printf("i%d\n", i);
                        if(t->v > t->next->v){
//                              printf("i%d\n", i);
                                // pre, t, t->next, t->next->next
                                node * nn = t->next->next;
                                node * n = t->next;
                                if(pre != NULL)
                                        pre->next = n;
                                else head = n;
                                n->next = t;
                                t->next = nn;
                                pre = n;
                        }
                        else{
                                pre = t;
                                t = t->next;
                        }
                }
//              node * thead = head;
//              printf("%d", thead->v);
//              while(thead->next != NULL){
//                      printf(" %d", thead->next->v);
//                      thead = thead->next;
//              }
//              printf("\n");
        }
        printf("%d", head->v);
        while(head->next != NULL){
                printf(" %d", head->next->v);
                head = head->next;
        }
        return 0;
}
```

2. 冒泡一轮，输出一次当前的序列

综合练习

— 1201

1. 旋转输出矩阵

- 和上机模拟的 ‘二维数组回形遍历 ‘类似
- 先从一个方向把二维数组输出到一维数组中
- 再头尾两个指针输出（两端夹逼）

3. 算 24

给出4个小于10个正整数，你可以使用加减乘除4种运算以及括号把这4个数连接起来得到一个表达式。现在的问题是，是否存在一种方式使得得到的表达式的结果等于24。

这里加减乘除以及括号的运算结果和运算的优先级跟我们平常的定义一致（这里的除法定义是实数除法）。

比如，对于5，5，5，1，我们知道5 * (5 ? 1 / 5) = 24，因此可以得到24。又比如，对于1，1，4，2，我们怎么都不能得到24。

关于输入

输入数据包括多行，每行给出一组测试数据，包括4个小于10个正整数。最后一组测试数据中包括4个0，表示输入的结束，这组数据不用处理。

关于输出

对于每一组测试数据，输出一行，如果可以得到24，输出“YES”；否则，输出“NO”。

例子输入

```
5 5 5 1

1 1 4 2

0 0 0 0
```

例子输出

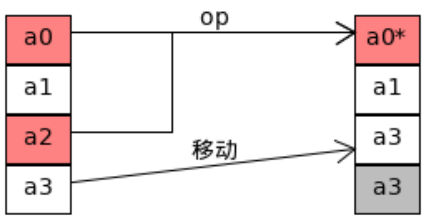
```
YES

NO
```

3. 算 24

- 有同学是这样过的

3. 算 24

- 我把 1500011105 同学的代码注释了作为样例
- 递归，每次做一步运算，枚举每种情况
- 运算前保留原状态，运算结束后还原，为下一次运算做准备
- 理论上是近似解: 和 24 的差值小于 10^{-9}




```
#include<stdio.h>
#include<math.h>
double nums[4];
// 表示数字i,j使用运算符op得到的运算结果
double eval(double i,double j,int op){
  switch(op){
    case 0:return i+j;
    case 1:return i-j;
    case 2:return i*j;
    case 3:return i/j;
    case 4:return j-i;
    case 5:return j/i;
  }
  return 0;
}
int calc(int n){//calc(n)表示(n+1)个数字算24点
  int i,j,k;
  if(n==0){//只剩一个数字，看看是不是24
    return(fabs(nums[0]-24)<1e-9);
  }
  for(i=0;i<n;i++){
    for(j=i+1;j<=n;j++){//任取两个数字，做一次运算
      double a=nums[i];
      double b=nums[j];
      nums[j]=nums[n];/*把nums[i],nums[j]的运算结果放到第i个位置，把原来
      的最后一个数字放到第j个位置*/
      for(k=0;k<6;k++){//求出这两个数字做运算得到的所有可能
        nums[i]=eval(a,b,k);
        if(calc(n-1))//递归看是否可能得到24点
          return 1;
      }
      nums[i]=a;//为了计算下一个nums[i],nums[j]还原数组nums为原来的值
      nums[j]=b;
    }
  }
  return 0;
}
int main()
{
  for(;;){
    scanf("%lf%lf%lf%lf",&nums[0],&nums[1],&nums[2],&nums[3]);
    if(nums[0]<=0){
      break;
    }
    printf("%s\n",calc(3)?"YES":"NO");
  }
  return 0;
}
```

3. 算 24

- 再贴一个我过的样例，
  不是特别好
- 每个函数都被其下面紧
  邻的函数调用

```
// a,b进行不考虑顺序的任意运算是否可以得到tar
bool find(double a, double b, double tar){
  return fabs(a*b - tar) < 1E-4 || fabs(a+b - tar) < 1E-4 || (b!=0 && fabs(a -tar*b)< 1E-4)
         || fabs(a-b - tar) < 1E-4 || fabs(b-a - tar) < 1E-4 || (a!=0 && fabs(b-a*tar) < 1E-4);
}
//a,b,c三个数进行运算是否可以得到t，其中a,b先进行运算
bool find(double a, double b, double c, double t){
  return  (c!=0&&find(a,b, t/c)) || (c!=0 &&find(a,b, t*c)) || find(a,b,t-c) || find(a,b,t+c)
          || find(a,b,c-t)|| (c!=0 &&find(a,b,c/t));
}
//a,b,c三个数进行任意顺序的运算，是否可以得到t
bool find2(double a, double b, double c, double t){
  return find(a,b,c,t) || find(a,c,b,t) || find(b,c,a,t);
}
//a,b,c,d四个数，a和b先进行运算，之后任意顺序，是否可以得到t
bool find(double a, double b, double c, double d, double t){
  return find2(a*b, c,d, t) || find2(a+b, c,d, t) || find2(a-b, c,d, t) ||
         find2(b-a, c,d, t) || (b!=0 && find2(a/b, c,d, t)) ||(a!=0 && find2(b/a, c,d, t));
}
//a,b,c,d四个数进行任意顺序的运算是否可以得到t
bool find2(double a, double b, double c, double d, double t){
  return find(a,b,c,d,t) || find(a,c,b,d,t) || find(a,d,c,b,t) ||
         find(b,c, a, d,t) || find(b,d, a,c,t) || find(c,d,a,b,t);
}
```

4. 进制转换

- 先将 14 进制数转化成十进制，然后将十进制转换成 7 进制打印

8. 因式分解

给出一个正整数a，要求分解成若干个正整数的乘积，即a = a1 * a2 * a3 * ... * an，并且1 < a1 <= a2 <= a3 <= ... <= an，问这样的分解的种数有多少。注意到a = a也是一种分解。

关于输入

第1行是测试数据的组数n，后面跟着n行输入。每组测试数据占1行，包括一个正整数a (1 < a < 32768)

关于输出

n行，每行输出对应一个输入。输出应是一个正整数，指明满足要求的分解的种数

例子输入

```
2

2

20
```

例子输出

```
1

4
```

8. 因式分解

- 方法见代码

```
#include <stdio.h>
// 整数n的满足每个因子都不大于max的因式分解个数
int find(int n, int max){
  if(n < 2)// n=1无因式分解
    return 0;
  int re = 0;
  for(int i = 2; i <= n/2 && i <= max; i++)
    if(n % i == 0)// i是n的因子，递归查找
      re += find(n/i, i);
  re += (n <= max) ? 1:0;// n=n本身也是一种因式分解
  //debug用的
  //printf("n:%d m:%d re:%d\n", n, max, re);
  return re;
}
```

9. 前缀表达式

前缀表达式是一种把运算符前置的算术表达式，例如普通的表达式2 + 3的前缀表示法为+ 2 3。前缀表达式的优点是运算符之间不必有优先级关系，也不必用括号改变运算次序，例如(2 + 3) * 4的前缀表示法为* + 2 3 4。本题求解前缀表达式的值，其中运算符包括+ - * /四个。

关于输入

输入为一行，其中运算符和运算数之间都用空格分隔，运算数是浮点数。

关于输出

输出为一行，表达式的值。
可直接用printf("%f\n", v)输出表达式的值v。

例子输入

```
* + 11.0 12.0 + 24.0 35.0
```

例子输出

```
1357.000000
```

提示

可使用atof(str)把字符串转换为一个double类型的浮点数。atof定义在stdlib.h中。
此题可使用函数递归调用的方法求解。

9. 前缀表达式

- 思路 1：按照要求来：出现连续两个整数，就应该用他们加上他们之前的运算符进行运算，将三个元素替换成一个元素，然后不断做，直到只剩一个元素，就是我们要的结果

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(){
  char c[1000];//表达式
  gets(c);
  char op[500];//运算符数组
  double v[500];//数值数组
  int opn=0, vn=0;//opn为运算符的长度，vn为数值数组的长度
  bool type[500];//type[i]=1表示第i个元素是运算符，否则表示是数值
  int tn = 0;//type数组的长度
  char * o = strtok(c, " ");//按空格split元素
  while(o != NULL){
//              printf("%s\n", o);
    if(o[0] == '*' || o[0] == '-' ||
       o[0] == '+' || o[0] == '/'){//是运算符
      op[opn++] = o[0];
      type[tn++] = 1;//记录第tn个元素是运算符
    }
    else {//是数字
      v[vn++] = atof(o);
      type[tn++] = 0;
      //最近的三个元素是(运算符，数字，数字),type[tn-1]已经是当前这个数字了
      // 不能能出现连续三个数字，如果最近的两个是数字，前一个一定是运算符
      while(tn > 1 && type[tn-2] == 0){
        double v1 = v[--vn];
        double v2 = v[--vn];
        switch(op[opn-1]){
          case '+':
            v[vn++] = v1+v2;break;
          case '-':
            v[vn++] = v2-v1;break;
          case '*':
            v[vn++] = v2*v1;break;
          case '/':
            v[vn++] = v2/v1;break;
        }
        tn -= 3;//将这次运算用到的三个元素删除
        type[tn++] = 0;//添加这次运算得到的新元素
//                            printf("%f %f %f\n", v1, v2, v[vn-1]);
        opn --;//删除刚刚使用的运算符
      }
    }
    o = strtok(0, " ");//取出下一个元素
  }
  printf("%f\n", v[0]);
  return 0;
}
```

9. 前缀表达式

- 思路 2：递归
  - block = (+) (subBlock1) (subBlock2)
  - subBlock1 = (-) (subBlock11) (subBlock12)

```
double expr()
{
    static char buffer[64];
    scanf("%s", buffer);
    switch(buffer[0])
    {
        case '+':
            return expr() + expr();
        case '-':
            return expr() - expr();
        case '*':
            return expr() * expr();
        case '/':
            return expr() / expr();
        default:    /* is number */
            return atof(buffer);
    }
}
```

几个问题/建议

- 想好怎么做（过程）再写代码
- 提高 = 读懂别人的代码 +**自己实现**
- 常用函数的头文件背好
- 本地运行过不了样例的，提交上去 99.999999999% 过不了
- 很奇怪的数/字符（-1208899956，烫烫烫）：数组越界！
- ‘Empty output file‘没输出:数据没读完（程序没有足够的内存记录待读入的数据） 或者程序中有死循环
- *intnum*[300] = {0} 的含义: 只初始化 *nums*[0] = 0，之后的数据值未知

最后