**1.已知循环缓冲区是一个可以无限循环读写的缓冲区，当缓冲区满了还继续写的话就会覆盖我们还没读取到的数据。下面定义了一个循环缓冲区并初始化，请编写它的Write函数：**

typedef struct RingBuf {

char \*Buf;

unsigned int Size;

unsigned int RdId;

unsigned int WrId;

}RingBuf;

void Init(RingBuf \*ringBuf, char \*buf, unsigned int size) {

memset(ringBuf, 0, sizeof(RingBuf));

ringBuf->Buf = buf;

ringBuf->Size = size;

ringBuf->RdId = 0;

ringBuf->WrId = strlen(buf);

}

void Write(RingBuf \*ringBuf, char \*buf, unsigned int len) {

unsigned int pos = ringBuf->WrId;

while (pos + len > ringBuf->Size) {

memcpy(ringBuf->Buf + pos, buf, ringBuf->Size - pos);

buf += ringBuf->Size - pos;

len -= ringBuf->Size - pos;

pos = 0;

}

memcpy(ringBuf->Buf + pos, buf, len);

ringBuf->WrId = pos + len;

}

void Print(RingBuf \*ringBuf) {

for (int i = 0; i < ringBuf->Size; i++) {

cout << ringBuf->Buf[i];

}

cout << endl;

}

int main()

{

RingBuf \*rb = (RingBuf \*)malloc(sizeof(RingBuf));

char init\_str[] = "ABC";

int size = 6;

Init(rb, init\_str, size);

char p[] = "1234567";

Write(rb, p, 7);

Print(rb);

return 0;

}

2. 已知两个已经按从小到大排列的数组，将它们中的所有数字组合成一个新的数组，要求新数组也是按照从小到大的顺序。请按照上述描述完成函数：

int merge(int \*array1, int len1, int \*array2, int len2, int \*array3);

解答：这道题本质上就是一道合并排序，网上一大堆的程序案例，就不多介绍了。下面这段程序是我自己写的，并不是从网上贴的，如果有一些BUG，还请指出。

int merge(int \*array1, int len1, int \*array2, int len2, int \*array3) {

int retn = len1 + len2;

if ((\*array1 < \*array2 || len2 == 0) && len1 > 0) {

\*array3 = \*array1;

merge(++array1, --len1, array2, len2, ++array3);

}

if ((\*array1 >= \*array2 || len1 == 0) && len2 > 0) {

\*array3 = \*array2;

merge(array1, len1, ++array2, --len2, ++array3);

}

return retn;

}

2. 有10000个正整数，每个数的取值范围均在1到1000之间，编程找出从小到大排在第3400（从0开始算起）的那个数，将此数的值返回，要求不使用排序实现。（总分10分）

解答：10000个正整数找出从小到大的第3400个（从0开始算起），第一个想到的就是排序（冒泡排序、插入排序、选择排序……），或者使用桶排序。但这些显然都不满足这个题目的要求。

关键的点是：正整数，每个数的取值均在1-1000之间，这是本题一个特殊性。

本题思路：维护一个数组count[1000]，分别存储1-1000每个数字的出现次数。

#include <iostream>

using namespace std;

#define TOTAL 10000

#define RANGE 1000

#define REQUIRED 3400

int main()

{

int number[TOTAL] = { 0 };

int count[RANGE] = { 0 };

int i, sum = 0;

for (i = 0; i < 10000; i++) {

number[i] = (rand() % 1000) + 1; /\*产生10000个1-1000之间的随机数\*/

}

for (i = 0; i < 10000; i++) {

count[number[i] - 1]++; /\*计算10000个整数出现次数\*/

}

for (i = 0; i < 1000; i++) {

sum += count[i];

if (sum >= REQUIRED + 1) {

cout << i + 1 << endl;

break;

}

}

return 0;

}

3. 双向循环[链表](https://www.nowcoder.com/jump/super-jump/word?word=%E9%93%BE%E8%A1%A8)怎么创建？用两句话说清楚（两句话没有说清楚，不用再说了）

## 4. 小米面试题

# 一个乱序数组a[0…n-1]，求a[j]-a[i]的最大值，其中i<j

# （[小米面试题]一个乱序数组a[0...n-1]，求a[j]-a[i]的最大值

）

5.



6. 以下代码是把一个字符串倒序，如“abcd”倒序后变成“dcba”

方法2：

main()

{

char str[]="hello,world";

int len=strlen(str);

char t;

for(int i=0; i{

t=str[i];

str[i]=str[len-i-1]; str[len-i-1]=t;

}

printf("%s",str);

return 0;

}

7. 函数checkstr判断一个字符串是不是对称的，如：“abccba”是对称的，"abccbaa"则不是对称的。函数声明如下，其中msg为输入的字符串，对称返回0，不对称返回-1，请实现该函数。

int checkstr(const char \*msg);

答：

int checkstr(const char \*msg)

{

int i = 0, len;

while(msg[i] != '\0')i++;

len = i;

for(i = 0; i < len/2; i++)

if(msg[i] != msg[len-i-1])

return -1;

return 0;

}

8. 给出一个单向链表的头指针，输出该链表中倒数第k个节点的指针。链表的倒数第0个节点为链表的尾结点（尾结点的next成员为NULL）。

函数findnode实现上述功能，链表节点定义及函数声明如下，请实现函数findnode。

typedef struct Node

{

struct Node \*next;

}Node;

Node\* findnode(Node \*head, unsigned int k);

Node\* findnode(Node \*head, unsigned int k)

{

int cnt = 0;

Node \*p = head;

while(p)

{

cnt++;

p = p->next;

}

if(cnt < k)

return NULL;

cnt = cnt - k - 1;

p = head;

while(cnt--)

p = p->next;

return p;

}

9. struct SC{int a,b,c;};

struct SD{int a,b,c,d;};

int main()

{

struct SC c1[] = {{3}, {4}, {5}, {6}};

struct SD \*c2 = (struct SD\*)c1 + 1;

cout<<c2->a<<c2->b<<c2->c<<c2->d<<endl;

return 0;

}

请问其运行结束时的打印信息是

答：0050

c1的存储方式是这样的：

3 0 0 | 4 0 0 | 5 0 0 | 6 0 0

c1[0] | c1[2] | c1[3] | c1[4]

c1转换为CD结构后是这样的：

3 0 0 4 | 0 0 5 0 | 0 6 0 0

c1[0] | c1[1] | c1[2]

c2 = c1 + 1，因此c2指向转换后的c1[1]，即0050

10. 给一个整数数组，然后计算出第二大的数   
2、使用递归方法判断整数数组是否为递增序列   
3、阶乘

11. int func(int\* arr,int len)

{

//max是最大的数，\_max是第二大的数

int max = arr[0],\_max = 0;

for(int i=1;i<len;i++)

{

if(arr[i] > max)

{

\_max = max;

max = arr[i];

}

else if(arr[i] > \_max) //这个条件是来判断后面出现的数会不会比max小但是比\_max大

{

\_max = arr[i];

}

}

return \_max;

}

12. 给定一个超大的文本文件，有很多行，怎么用C语言实现把这个文件按行倒序输出

13. 给一本书，让你统计每个词出现的频率   
如果要查看词频前10个怎么做

14. int foo ( int x , int n)　  
{  
int val;  
val =1;

if (n>0)  
{  
if (n%2 == 1) val = val \*x;

val = val \* foo(x\*x , n/2);

}  
return val;  
}

x^n (x的n次幂)

15. Linux网络编程主机字节序和网络字节序的转换函数的实现

16. **如何申请一块超大连续内存？**

(类型说明符\*)calloc(n,size)  
功能：在内存动态存储区中分配n块长度为“size”字节的连续区域。函数的返回值为该区域的首地址。

# 17. 100个人回答五道试题，有81人答对第一题，91人答对第二题，85人答对第三题，79人答对第四题，74人答对第

100个人回答五道试题，有81人答对第一题，91人答对第二题，85人答对第三题，79人答对第四题，74人答对第五题，答对三道题或三道题以上的人算及格， 那么，在这100人中，至少有（ ）人及格。

相当于总共答了：100×5=500道

答对的题有：81+91+85+79+74=410道

则答错的题有：500-410=90道

题目问至少多少人及格，那么当不及格的人最多时，即符合题意。

答错3道即为不及格，则不及格的人最多为90÷3=30人

及格的人数至少有：100-30=70人

# 18. 有7g，2g砝码各一个，天平一只，如何只用这些物品三次将140g的盐分成50g，90g各一份？

1. 不用砝码，将 140 g 盐用天平分成 70g + 70g
2. 不用砝码，将 70 g盐用天平分成 35g + 35g
3. 用天平将一份 35g 盐分为 20g盐 + 2g砝码 = 15g盐 + 7g砝码
4. 将 20g与 70g盐并在一起就是 90g，剩下的是 50g

19.

计算一颗二叉树的深度

深度的计算函数：

int depth(BiTree T)

{

if(!T) return 0; //判断当前结点是否为叶子结点

int d1= depth(T->lchild); //求当前结点的左孩子树的深度

int d2= depth(T->rchild); //求当前结点的右孩子树的深度

} return (d1>d2?d1:d2)+1;

20一个整数数组，可能是降序或升序，也可能是先升序再降序，求最大值？（几秒后就让我赶紧说思路）

21二进制0的个数

int CountZeroBit(int num)

{

int count = 0;

while (num + 1)

{

count++;

num |= (num + 1); //算法转换

}

return count;

}

int main()

{

int value = 25;

int ret = CountZeroBit(value);

printf("%d的二进制位中0的个数为%d\n",value, ret);

system("pause");

return 0;

}

22给一个整数数组，然后计算出第二大的数

const int MINNUMBER = -32767;

int find\_sec\_max(int data[], int count)

{

int maxnumber = data[0];

int sec\_max = MINNUMBER;

for(int i = 1; i < count; i++) {

if(data[i] > maxnumber) {

sec\_max = maxnumber;

maxnumber = data[i];

} else {

if(data[i] > sec\_max) sec\_max = data[i];

}

}

return sec\_max;

}

23使用[递归方法](https://so.csdn.net/so/search?q=%E9%80%92%E5%BD%92%E6%96%B9%E6%B3%95)判断整数数组是否为递增序列

bool fun(int a[], int n)

{

if(n == 1)

return true;

if(n == 2)

return a[n-1] >= a[n-2];

return fun(a, n-1) && (a[n-1] >= a[n-2]);

}

24一个函数实现给定字符串,去除前面和后面的空格,比如“ ab cd ”,最后得到的结果是

25字符串移位，给出字符串abc##dfg##gh，实现将所有#移至字符串串头。输出####abcdfggh（个人认为可以用后向移位，减少移位次数）

26 数组找前k大的数字

27两个有序链表合并

List **Merge**( List L1, List L2 ){

List Pa, Pb, Pc, L;

L **=** (List)malloc(**sizeof**(**struct** Node)); *// 定义首节点*

Pa **=** L1**->**Next; Pb **=** L2**->**Next; *// 分别用于遍历L1，L2*

Pc **=** L; *// 用于接续合并后的新链表*

**while**( Pa **&&** Pb ){ *// 只要L1和L2中有一个遍历完就退出循环*

**if**( Pa**->**Data **<** Pb**->**Data ){ *// 如果Pa指向节点数据小于Pb指向节点数据*

Pc**->**Next **=** Pa; *// 合并Pa指向的节点*

Pc **=** Pa; *// Pc移动到L当前最后的节点*

Pa **=** Pa**->**Next; *// Pa移动到L1当前最后的节点*

}**else**{

Pc**->**Next **=** Pb;

Pc **=** Pb;

Pb **=** Pb**->**Next;

}

Pc**->**Next **=** Pa**?**Pa:Pb; *// 考虑结尾情况：L1遍历完或者L2遍历完*

L1**->**Next **=** NULL; *// 使得L1、L2为空链表*

L2**->**Next **=** NULL; *//*

}

**return** L;

}

28Linux网络编程主机字节序和网络字节序的转换函数的实现

#define BigLittleSwap32(A) ((((unsigned int)(A) & 0xff000000) >> 24) | (((unsigned int)(A) & 0x00ff0000) >> 8) | (((unsigned int)(A) & 0x0000ff00) << 8) | (((unsigned int)(A) & 0x000000ff) << 24))

int checkCPUendian()

{

union{

unsigned long int i;

unsigned char s[4];

}c;

c.i=0x12345678;

return (0x12 == c.s[0]);

}

1. quint32 hostnetl(unsigned short int a)
2. {
3. return checkCPUendian() ? a : BigLittleSwap32(a);
4. }

29**计算一颗二叉树的深度**

int depth(BiTree T)

{

if(!T) return 0; //判断当前结点是否为叶子结点

int d1= depth(T->lchild); //求当前结点的左孩子树的深度

int d2= depth(T->rchild); //求当前结点的右孩子树的深度

} return (d1>d2?d1:d2)+1;

## 30

## 100大小的数组放了1-99 100个数，只有一个数重复了，找出重复的那个数。

* 答:解法很简单，把这100个数加起来，然后逐一剪掉1-99 剩下的那个数就是重复的数。

31