**2014 C语言程序设计-笔试**

总体感觉：和去年的题有重复，时间紧，所以大家还是要好好准备，

一、指出程序中的错误，说明原因并修正。

1、

Int \*p,\*q;

p=(int\*)malloc(sizeof(int)\*20);

q=(int\*)malloc(sizeof(int)\*10);

…

q=p;//修改的话可以删除这句话

…

free(p);

free(q);// 错了，因为p=q后，就指向同一块地址了。 内存泄露（p分配了空间没释放）重复释放同一内存会报错，还剩一块内存没有释放，内存泄露

一：由于malloc函数返回值为void \*，所以还是应该强制转换为（int \*）,虽然我在运行时没有错，但一般都要指定指针类型。

二：由于p=q,两个指针指向同一个地址,free(p)之后，则该内存空间被释放，则q此时就没有具体指向一个地址，free(q)就会出错。

内存泄露：  
一般我们常说的内存泄漏是指堆内存的泄漏。堆内存是指程序从堆中分配的，大小任意的（内存块的大小可以在程序运行期决定），使用完后必须显式释放的内存。应用程序一般使用malloc，calloc，realloc，new等函数从堆中分配到一块内存，使用完后，程序必须负责相应的调用free或delete释放该内存块，否则，这块内存就不能被再次使用，我们就说这块内存泄漏了。  
内存泄漏是常见的问题。当以前分配的一片内存不再需要使用或无法访问时，但是却并没有释放它，那么对于该进程来说，会因此导致总可用内存的减少，这时就出现了内存泄漏。尽管优秀的编程实践可以确保最少的泄漏，但是根据经验，当使用大量的函数对相同的内存块进行处理时，很可能会出现内存泄漏。尤其是在碰到错误路径的情况下更是如此。

2、交换两个字符串(和去年一样)

Void swap(char \*p, char \*q)// 和去年一样

{

Char \*temp;

\*temp=\*p;

\*p=\*q;

\*q=\*temp;

}

1.函数类型大小写错误。

2. 交换形参指针地址不能修改实参的值。

Void swap(char \*p, char \*q)

{

         Char  temp[100];

         Strcpy(temp,p);

         Strcpy(p,q);

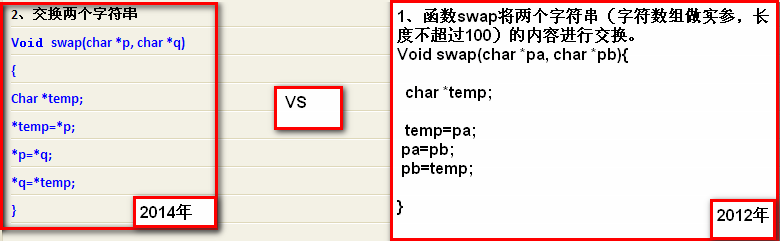
Strcpy(q,temp);

}

**正确修改法：见下**

**(解释)实参是str1和str2的地址 ，函数里改变了\*p,\*q的值，也就是str1和str2中存放的字符串的地址做了改变。所以str1里存放的str2所指字符串的地址**

![`KOD](0IN$MV4370D}N@43S](data:image/jpeg;base64,)



3、for(num=N;arr[num]!=item;num--);

printf("%d",num);

问异常情况

结果可能是输出一个负值，因为item是个变量，分配了存储空间， arr[-2]也是可以输出的，因为arr只是个地址，经过测试会找到item，printf他的地址

首先如果item在数组中，则不会出错，可以输出所在的位置。如果不在，则num最后会变为负值，由于C语言没有自动对数组进行下标检查，则会产生数组下标越界，而且会继续运行，出现死循环。如果编译器检查，则会产生数组下标越界出错。



二、简答题

1、arr为整型数组，N为数组长度-1，enumb为整型变量，下列函数负责找出arr数组中等于enumb的元素所在位置。

For(i=N-1;arr[i]!=enumb;--i)

Printf(“%d”,i);

指出程序会出现的3种异常，并说明原因。

*I为负数且运行结束，i为正数且运行结束,i一直递减直到堆栈溢出*

2、if(B) s1 else s2; 请问是什么结构，请用显式结构语言表示该程序段，用伪代码写出来(注明条件跳转和强制跳转)。

*if(b) goto L1;*

*s2; goto L2;*

*L1: s1;*

*L2:;*

//

选择结构。

Begin

If b！=0 条件跳转

Do S1;

否则 Do S2;

3、C语言中，常量存储在哪儿？static 全局变量和static 局部变量存储在哪儿？

//

全局变量(外部变量)的说明之前再冠以static 就构成了静态的全局变量。全局变量本身就是静态存储方式，

静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序，

当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。

而静态全局变量则限制了其作用域，

即只在定义该变量的源文件内有效，

在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，

因此可以避免在其它源文件中引起错误。

static全局变量与普通的全局变量有什么区别：static全局变量只初使化一次，防止在其他文件单元中被引用;

static局部变量和普通局部变量有什么区别：static局部变量只被初始化一次，下一次依据上一次结果值；

//

1. 局部变量，静态局部变量，静态全局变量的存储位置

局部变量： 栈区  
局部静态变量：静态区  
全局变量： 静态区的常量区  
全局静态变量：静态区

网上的资料，科普下：  
 1、栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。   
 2、堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。   
 3、全局区（静态区）（static）—，全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。 - 程序结束后由系统释放   
 4、文字常量区 —常量字符串就是放在这里的。 程序结束后由系统释放   
 5、程序代码区—存放函数体的二进制代码。

1、栈，就是那些由编译器在需要的时候分配，在不需要的时候自动清楚的变量的存储区。里面的变量通常是局部变量、函数参数等。

2、堆，就是那些由new分配的内存块，他们的释放编译器不去管，由我们的应用程序去控制，一般一个new就要对应一个delete。如果程序员没有释放掉，那么在程序结束后，操作系统会自动回收。

3、自由存储区，就是那些由malloc等分配的内存块，他和堆是十分相似的，不过它是用free来结束自己的生命的。

4、全局/静态存储区，全局变量和静态变量被分配到同一块内存中，在以前的c语言中，全局变量又分为初始化的和未初始化的，在c++里面没有这个区分了，他们共同占用同一块内存区。

5、常量存储区，这是一块比较特殊的存储区，他们里面存放的是常量，不允许修改。

三、填空题

1. 选择排序法的链表实现

(答案大概是 while(p->next) ..p=p->next head=head->next return head)

//

P=(struct node\*)malloc(sizeof());

p->next=head;

head=p;//新建一个头结点

p=head->next;

k=q=head;

while(...)

{

//初始化，记不得了，大体结构就这样

while(p)

{

if(q->next->data<p->data)

q=p;

p=p->next;

}

If(q!=k)

{

r=q-;

q->next=r->next;

p->next=r;

}

}

P=head;

head=head->next;

free(p)

return head;

//

(答案大概是 while(p->next) ..p=p->next head=head->next return head)

1)每次将最小的元素挪到链表的头部。即min.next=head;

while(p->next) {

for(p=head;p->next!=NULL;p=p->next){

if(p->next->data<q->data){

min=p->next;

}}

Min->next=head;

Head=min;

}

2)先建立一个newHead，循环原链表找到最小的元素，用连接到newHead的后面，相当于拆成了两条链。之后找到的最小值都加入新链表的末尾。代码差不多吧。

2. 快速排序法求某个数组前n个元素第k大的数

(答案大概是 i<j break low=i+1 high=i-1 ...return a[i] )

Low=0;high=n-1;

do{

I=low;j=high;t=a;

do{

While(a[--j]>t);

While(a[++i]<t);

If(i<j)swap(a[j],a);//具体代码实在忘了，反正不是空空，我这样写运行是一样的

}while(i<j);

If(i==k)

Returnt;

If(i>k)

high=i-1;

if(i<k)

low=i+1;

}while(low<high);

Return a[low];//这个空我可能错了

//

也有2种方法，最简单的，从大到小快速排序完以后，直接取第K个数。

int partions(int l[],int low,int high)   
{  
int prvotkey=l[low];  
l[0]=l[low];  
while (low<high)  
{  
 while (low<high&&l[high]>=prvotkey)  
 --high;  
 l[low]=l[high];  
 while (low<high&&l[low]<=prvotkey)  
 ++low;  
 l[high]=l[low];  
}  
l[low]=l[0];  
return low;  
}

void qsort(int l[],int low,int high)  
{  
int prvotloc;  
if(low<high)  
{  
 prvotloc=partions(l,low,high); //将第一次排序的结果作为枢轴  
 qsort(l,low,prvotloc-1); //递归调用排序 由low 到prvotloc-1  
 qsort(l,prvotloc+1,high); //递归调用排序 由 prvotloc+1到 high  
  
}  
}  
int quicksort(int l[],int n，int k)  
{  
qsort(l,1,n); //第一个作为枢轴 ，从第一个排到第n个

Return L[k-1];//返回第K大的数。  
}

看起来高级一点的方法（网上看的）：

基本思想：

以最后一个元素x为轴，把数组分为两部分Sa和Sb。Sa中的元素大于等于X，Sb中元素小于X。这时有两种情况：

1.Sa中元素的个数小于k，则Sb中的第k-|Sa|个元素即为第k大数；

2.Sa中元素的个数大于等于k，则返回Sa中的第k大数。

时间复杂度近似为O(n)

int partition(int \*a,int l,int r)

{

int i=l-1,j=l;

int x=a[r];

int temp;

for(j=l;j<r;j++)

{

if(a[j]>=x) //把比x大的数往前放

{

exchange(a[j],a[i+1]);

i++;

}

}

exchange(a[r],a[i+1]);

return i+1;

}

int k\_element(int \*a,int l,int r, int k)

{

if(l>=r)

return a[l];

int q=partition(a,l,r);

if(q==k-1)

return a[q];

else if(q>=k)

return k\_element(a,l,q-1,k); //Sa中元素个数大于等于k

else

return k\_element(a,q+1,r,k-(q+1)); //Sa中元素个数小于k

}

//

**#include<stdio.h>**

**2**

**3**

**4 void QuickSort(int arr[],int L,int H)**

**5 {**

**6 int ipivot;**

**7 int itemp;**

**8 int icount=1;**

**9 int low = L; //像我这样把得到枢纽的位置和快速排序放在一起，则L，H在这一趟排序中不能变，而改用low,high的变化求出枢纽的位置，因为L，H还要为下一次循环用。**

**10 int high = H;**

**11 itemp=arr[low];**

**12**

**13 if(L < H) //一定要有L < H判断，不然会有H为负值出现**

**14 {**

**15 while(low != high)**

**16 {**

**17 while((low < high)&&(arr[high] > itemp))**

**18 --high;**

**19 if(low < high)**

**20 {**

**21 arr[low] = arr[high];**

**22 ++low;**

**23 }**

**24 while((low < high)&&(arr[low] < itemp))**

**25 ++low;**

**26 if(low <high)**

**27 {**

**28 arr[high] = arr[low];**

**29 --high;**

**30 }**

**31 }**

**32**

**33**

**34**

**35 ipivot = low; //====ipivot = high,而不是=high+1**

**38 arr[ipivot] = itemp;**

**39**

**40 QuickSort(arr,L,ipivot-1);**

**41 QuickSort(arr,ipivot+1,H);**

**42 }**

**43 }**

**44**

**45 int Max\_K(int arr[],int length,int k)**

**46 {**

**47 QuickSort(arr,0,length-1);**

**48 return arr[k-1];**

**49 }**

**50**

**51**

**52 ////////////////后面的写答案时，不要，舍去，只是为了验证。**

**53 void main()**

**54 {**

**55 int arr[]={23,1,3,6,5,78,45,7};**

**56 int length;**

**57 int max\_k;**

**58 int k;**

**59**

**60 length = sizeof(arr)/sizeof(int);**

**61 printf("please input the data of K!\n");**

**62 scanf("%d",&k);**

**63 printf("length:%d\n",length);**

**64**

**65 max\_k = Max\_K(arr,length,k);**

**printf("OK,the %dth of arr is %d\n",k,max\_k);**

**68**

**69**

**70**

**71**

**72 }**

**===========**

**======我的法二：找枢纽位置与快速排序分开（参照下面的）**

**#include<stdio.h>**

**2**

**3**

**4 //-----yitangkuaisupaixu**

**5 int Postion(int arr[],int low,int high)**

**6 {**

**7 int itemp = arr[low];**

**8 while(low < high)**

**9 {**

**10 while((low < high)&&(arr[high] >= itemp))**

**11 --high;**

**12 arr[low] = arr[high];**

**13 while((low < high)&&(arr[low] <= itemp))**

**14 ++low;**

**15 arr[high] = arr[low];**

**16 }**

**17**

**18 arr[low] = itemp;**

**19 return high;**

**20**

**21 }**

**22**

**23**

**24**

**25 void QuickSort(int arr[],int low,int high)**

**26 {**

**27 int ipivot;**

**28 if(low < high)**

**29 {**

**30 ipivot = Postion(arr,low,high);**

**QuickSort(arr,low,ipivot-1);**

**32 QuickSort(arr,ipivot+1,high);**

**33 }**

**34 }**

**35**

**36**

**37 int Max\_k(int arr[],int length,int k)**

**38 {**

**39 QuickSort(arr,0,length-1);**

**40 return arr[k-1];**

**41 }**

**42**

**43 void main()**

**44 {**

**45 int arr[]={2,1,3,5,4,9,10,6,7,8};**

**46 int length;**

**47 int max\_k;**

**48 int k,i;**

**49 int \*pa=arr;**

**50**

**51 length = sizeof(arr)/sizeof(int);**

**52 printf("input\n");**

**53 scanf("%d",&k);**

**54**

**55 max\_k = Max\_k(arr,length,k);**

**56 for(i=0;i < length;i++)**

**57 printf(" %d",arr[i]);**

**58**

//=========33333,的到枢纽的位置，即一趟快速排序

**int partions(int l[],int low,int high)**   
{  
int prvotkey=l[low];  
l[0]=l[low];  
while (low<high)  
{  
 while (low<high&&l[high]>=prvotkey)  
 --high;  
 l[low]=l[high];  
 while (low<high&&l[low]<=prvotkey)  
 ++low;  
 l[high]=l[low];  
}  
l[low]=l[0];  
return low;  
}

//=========22222，递归调用快速排序

**void qsort(int l[],int low,int high)**{  
int prvotloc;  
if(low<high)  
{  
 prvotloc=partions(l,low,high); //将第一次排序的结果作为枢轴  
 qsort(l,low,prvotloc-1); //递归调用排序 由low 到prvotloc-1  
 qsort(l,prvotloc+1,high); //递归调用排序 由 prvotloc+1到 high  
  
}  
}

//=========1111，对数组进行快速排序，并取第K个元素  
**int quicksort(int l[],int n，int k)**{  
qsort(l,1,n); //第一个作为枢轴 ，从第一个排到第n个

Return L[k-1];//返回第K大的数。  
}

看起来高级一点的方法（网上看的（未看））：

基本思想：

以最后一个元素x为轴，把数组分为两部分Sa和Sb。Sa中的元素大于等于X，Sb中元素小于X。这时有两种情况：

1.Sa中元素的个数小于k，则Sb中的第k-|Sa|个元素即为第k大数；

2.Sa中元素的个数大于等于k，则返回Sa中的第k大数。

时间复杂度近似为O(n)

**int partition(int \*a,int l,int r)**

**{**

**int i=l-1,j=l;**

**int x=a[r];**

**int temp;**

**for(j=l;j<r;j++)**

**{**

**if(a[j]>=x) //把比x大的数往前放**

**{**

**exchange(a[j],a[i+1]);**

**i++;**

**}**

**}**

**exchange(a[r],a[i+1]);**

**return i+1;**

**}**

**int k\_element(int \*a,int l,int r, int k)**

**{**

**if(l>=r)**

**return a[l];**

**int q=partition(a,l,r);**

**if(q==k-1)**

**return a[q];**

**else if(q>=k)**

**return k\_element(a,l,q-1,k); //Sa中元素个数大于等于k**

**else**

**return k\_element(a,q+1,r,k-(q+1)); //Sa中元素个数小于k**

**}**

3. 约瑟夫环数组实现，即100个人围成圈，从第一个人开始，1~3报数，数到3的人退出，问剩下来的人编号是多扫？

约瑟夫环数组实现约瑟夫环是一个数学的应用问题：已知n个人（以编号1，2，3...n分别表示）围坐在一张圆桌周围。从编号为k的人从1开始报数，数到m的那个人出列；他的下一个人又从1开始报数，数到m的那个人又出列；依此规律重复下去，直到圆桌周围的人全部出列。

(答案大概是 p=a+N ,k=N, p>a+N, i<3 ,i%3, a[i]!=0 )

(答案大概是 p=a+N ,k=N, p>a+N, (\*p)!=0 ,i%3, a[i]!=0 )

//

3. 100个人围成圈，从第一个人开始，1~3报数，数到3的人退出，问剩下来的人编号是多扫？

约瑟夫环数组实现(答案大概是 p=a+N ,k=N, p>a+N, i<3 ,i%3, a!=0 )

//此题我理解为0号那个人不算，只算1-100人

Int a[N+1];

Int \*p=a,i;

For(i=0;i<N+1;i++)

\*p=i;

P=a+1;

K=N;

For(j=1,i=0;k!=1;j++)

{

If(p>a+N) p=a+1;

If(\*p!=0) i++;

If((i%3)==0) { k--;i=0;\*p=0;}

}

For(i=0;i<N+1;i++)

{

If(p!=0)

Printf(.);

}

//

int main()

{int m, n;

while (cin >> n >> m)

{int\* a = new int[n];

for (int i = 0; i < n; i++)

a[i] = i + 1;

int j = 1; //用于报数

int k = 0;//遍历数组

int l = n; //跟踪剩余人数

while (l > 1) {//剩余人数大于1，继续报数

if (a[k])

{if (j++ == m) { //报到m出局

j = 1; //重新报数

a[k] = 0; //记下位置

l--;//人数减1

}

}

if (++k == n)

k = 0; //循环数组

}

while (a[--i] == 0); //找到最后一个人，即为答案

cout << a[i] << endl;

delete[] a;

}return 0;}

//

==========回忆版

a[N+1]的数组

\*p=i 循环从0到100赋值

0不考虑的

for(j=1,i=0;k!=1;j++)

i就是报数的单位

if(p>a+n) p=a+1

跳过了a，因为他是0， 而这里是1-100人

if(\*p!=0) i++; 如果有人就报数

i从0开始的

4. 完美乘法，a\*b=c,abc中只出现0~9的数字，且每个数字在这个等式中只出现一遍。

(答案大概是 c=a\*b while(y>0) y=y/10; if(f[x]!=1) t=1; if(t==0) n++ 这题看着麻烦，其实最简单)

//

for(i=12;i<999;i++)

{

t=0;

for(x=0;x<10;x++)

f[x]=0;

for(j=345;j<9999;j++)

{

c=a\*b;

s[0]=a;s[1]=b;s[2]=c;

for(x=0;x<3;x++)

{

y=s[x];

while(y>0)

{

Int t=y%10;

f[t]++;

y=y/10;

}

for(x=0;x<10;x++)

{

if(f[x]!=1) t++;

}

if(t==0)

n++;

}

}

}

//

看他那看都看不懂的答案来说，有可能是求1000以内的完美数？那就是：

void main()

{

int j,k,sum = 0;

for(k=2;k<=1000;k++)

{

sum=0;

for(j=1;j<k;j++)

if(k%j==0)

sum=sum+j;

if(sum==k)

printf("%d ",k);

}

}

或者题是判断一个数是不是完美数？那就是：

要判断m是否是完全数

for(k=1;k<m;k++)

{

if(m%k==0)

sum=sum+k;

}

If(sum==m)

Return true;

四、程序题

1、编写函数，实现字符串的逆序，不允许申请新的数组空间

Void reverse(char \*str)

//

Void reverse(char \*str)

{

Int i=0,len=strlen(str);

For(;i<len/2;i++)

{

Char a=str;//突然想到我当时写成了Int，，。日

str=str[len-i-1];

str[len-i-1]=a;

}

}

//

void ReverseStr(char \*s)

{

int i,j;

char c;

j=strlen(s);

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

{ c=s[i];

s[i]=s[j-1-i];

s[j-1-i]=c; }

}

//

**#include<stdio.h>**

**2 #include<string.h>**

**3**

**4 #define MAX 40**

**5**

**6 void Reverse\_2(char \*pstr)**

**7 {**

**8 int length;**

**9 int i;**

**10 char ctemp;**

**11**

**12 length = strlen(pstr);**

**13 for(i=0;i < length/2;i++)**

**14 {**

**15 ctemp = pstr[length-i-1];**

**16 pstr[length-i-1] = pstr[i];**

**17 pstr[i] = ctemp;**

**18**

**19 }**

**20 }**

**21**

**22 void Reverse(char \*pstr)**

**23 {**

**24 int length;**

**25 int i;**

**26 char ctemp;**

**27**

**28 length = strlen(pstr);**

**29**

**30 for(i=0;i < length/2;i++**)

31 {

**ctemp = \*(pstr+length-i-1);**

**33 \*(pstr+length-i-1) = \*(pstr+i);**

**34 \*(pstr+i) = ctemp;**

**35**

**36 }**

**37**

**38 }**

**39**

**40 void main()**

**41 {**

**42 char cstr[MAX];**

**43 gets(cstr);**

**44**

**45 puts(cstr);**

**46 //Reverse(cstr);**

**47 Reverse\_2(cstr);**

**48 printf("%s\n",cstr);**

**49**

**50 }**

**32 ctemp = \*(pstr+length-i-1);**

2、身份证长度是18，其中第7至14个数字是生日，编写完整程序判断2个身份证号的出生日期先后。

//

Int judge(char \*s1,char\*s2)

{

Int low=6,high=14,i;

For(i=low;i<high;i++)

{

If(s1>s2)

Return 1;

Else if(s1<s2)

Return -1;

}

Return 0;

}

//

身份证日期比较(写个函数即可，比较简单)先比较年，然后月，然后日

If(我的出生日期>你的)

Return 1;

If(我的出生日期和你一样)

Return 0;

If(我的出生日期<你的)

Return -1;

//

**#include<stdio.h>**

**2**

**3**

**4 //-----1: str1 2:str2 0:==**

**5 int Compare(char str1[],char str2[])**

**6 {**

**7 int j=6;**

**8 int i;**

**9 int a\_1,a\_2;**

**10**

**11 for(i=0; i< 8;i++)**

**12 {**

**13 //---xianzhuanhuanweishuzi**

**14 a\_1 = \*(str1+j+i)-'0';**

**15 a\_2 = \*(str2+j+i)-'0';**

**16**

**17 if(a\_1 > a\_2)**

**18 return 1;**

**19 else**

**20 if(a\_1 < a\_2)**

**21 return 2;**

**22**

**23 continue;**

**24 }**

**25 return 0;**

**26**

**}**

**28**

**29**

**30 void main()**

**31 {**

**32 char str1[18],str2[18];**

**33 int ireturn;**

**34**

**35 scanf("str1=%s\nstr2=%s",str1,str2);**

**36**

**37 ireturn = Compare(str1,str2);**

**38 if(1== ireturn)**

**39 printf("the max is str1:%s\n",str1);**

**40 else**

**41 if(2== ireturn)**

**42 printf("the max is str2:%s\n",str2);**

**43 else**

**44 printf("oh,sir,the two people are the same birthday!,%s\n",str1);**

**45**

**46**

**47 }**

3、编写完整程序，计算1-x+x^2/2!-x^3/3!+…+x^n/n!，只允许有Main函数，不允许定义其他函数，x是单浮点数，n是整数。

//

Floatsum=0,x=1,res=0,res1=0;

Int i,j,k;

for(i=1;i<n;i++){ //循环n次

for(j=1;j<i;j++){ //求阶乘

sum=sum\*j;}

for(k=0;k<i;k++){ //求次方

x=x\*x;}

res=x/sum; //得到结果

if(i%2==1) //处理符号

res=0-res;

res1=res+res1;

}

//

去年原题 1-x+x^2/2!+... 注意不能自己写自定义函数，要在main函数里完成，建议写2个for循环，内层写阶乘和x的n次方。

阶乘的代码：for (i = 2; i <= n; i++) sum = sum \* i;

这个其他也没什么好写的。

//

**#include<stdio.h>**

**2**

**3 void main()**

**4 {**

**5 int n,total=0;**

**6 float x;**

**7 int itemp\_a=x,itemp\_b=1;**

**8 int i,j;**

**9**

**10 printf("please input the data of x,n\n");**

**11 scanf("x=%f\nn=%d",&x,&n);**

**12**

**13 for(i=0;i <= n;i++)**

**14 {**

**15 if(0 == i)**

**16 {**

**17 total +=1;**

**18 }**

**19 else**

**20**

**21 if(1 == i)**

**22 {**

**23 total = total-x;**

**}**

**25 else**

**26 {**

**27 //===oushu**

**28 if((0 == i%2))**

**29 {**

**30 itemp\_a =1;**

**31 itemp\_b =1;**

**32 for(j=i;j >=1;j--)**

**33 {**

**34 itemp\_a \*=x;**

**35 itemp\_b \*=j;**

**36 }**

**37 total=total+(itemp\_a/itemp\_b);**

**38 }**

**39 else**

**40 //===jishu**

**41 {**

**42 itemp\_a =1;**

**43 itemp\_b =1;**

**44 for(j=i;j >=1;j--)**

**45 {**

**46 itemp\_a \*=x;**

**47 itemp\_b \*=j;**

**48**

**49 }**

**50 total=total-(itemp\_a/itemp\_b);**

**51 }**

**//===shuchu**

**57 printf("total is %d\n",total);**

**58**

**59**

**60 }**

4、编写完整程序，一个链表，找出其中数据项最大的结点，然后将其移动到链表尾部（结点node由整型data和节点指针next构成），不允许申请新的结点。

//

Struct node\*p,\*q,\*r;

P=q=head;

while(p->next)

{

if(q->next->data<p->next->data)

q=p;

p=p->next;

}

r=q->next;

q->next=r->next;

p->next=r;

//

1. 求链表最大的数，并插入链表尾部(代码参照填空题第一题选择排序法，半分钟能鲁完)

复习下冒泡排序

LinkList Sort(LinkList &SL){

LinkList p,q;

int temp;

for(p=SL;p!=NULL;p=p->next){

for(q=p->next;q!=NULL;q=q->next){

if(p->data>q->data){

temp=q->data;

q->data=p->data;

p->data=temp;}

}

}}

//

======我的（不要此法，不太符合题意，正确见法二）

**#include<stdio.h>**

**2 #include<stdlib.h>**

**3**

**4 typedef struct node{**

**5**

**6 int data;**

**7 struct node \*next;**

**8 }LNode,\*PLNode;**

**9**

**10**

**11 void MoveMax(LNode \*L)**

**12 {**

**13 LNode \*pa,\*pb,\*pre; //==pre指向pb的前一个节点，最后指向最后一个节点**

**14 int itemp;**

**15**

**16 pa=pb=L->next;**

**17 pre=L;**

**18 printf("2====\n");**

**19 //--找到最大值节点**

**20 while(pb)**

**21 {**

**22 if(pa->data < pb->data)**

**23 pa=pb;**

**24 pre = pre->next;**

**25 pb = pb->next;**

**26 }**

**27**

**28 printf("3====\n");**

**29 //-----最大值节点pa和最后一个节点pre交换数据**

**30 itemp = pre->data;**

**pre->data = pa->data;**

**32 pa->data = itemp;**

**33 }**

**34**

**35**

**36 void main()**

**37 {**

**38 LNode \*Head,\*pa;**

**39 int n,i;**

**40**

**41 //---chushihuadanlianbiao**

**42 Head = (PLNode)malloc(sizeof(LNode));**

**43 Head->next = NULL;**

**44**

**45 //---jianlidanlianbiao**

**46 printf("please input the data of n!\n");**

**47 scanf("%d",&n);**

**48 for(i=0;i < n;i++)**

**49 {**

**50 pa = (PLNode)malloc(sizeof(LNode));**

**51 printf("input the data:\n");**

**52 scanf("%d",&pa->data);**

**53 pa->next = Head->next;**

**54 Head->next = pa;**

**55 }//end of for**

**56**

**57 printf("1====\n");**

**58 MoveMax(Head);**

**printf("4====\n");**

**61 pa = Head->next;**

**62 while(pa)**

**63 {**

**64 printf(" %d",pa->data);**

**65 pa = pa->next;**

**66 }**

**67 printf("\n");**

**68**

**69 //---shifang**

**70 free(Head);**

**71**

**72**

**73 }**

======================法二

**void MoveMax(LNode \*L)**

**12 {**

**13 LNode \*pa,\*pb; // pa用来指向最大值节点的前一个节点**

**LNode \*pre,\*ptemp; //pre用来指向pb的前一个节点，最终指向尾节点 ptemp:用来指 向最大值节点**

**14 int itemp;**

**16 pa=pb=L->next;**

**17 pre=L;**

**19 //--找最大值节点**

**20 while(pb)**

**21 {**

**22 if((pa->next->data) < (pb->data))**

**23 pa->next=pb;**

**24 pre = pre->next;**

**25 pb = pb->next;**

**26 }**

**30 //-----将最大值节点移动到链表尾部**

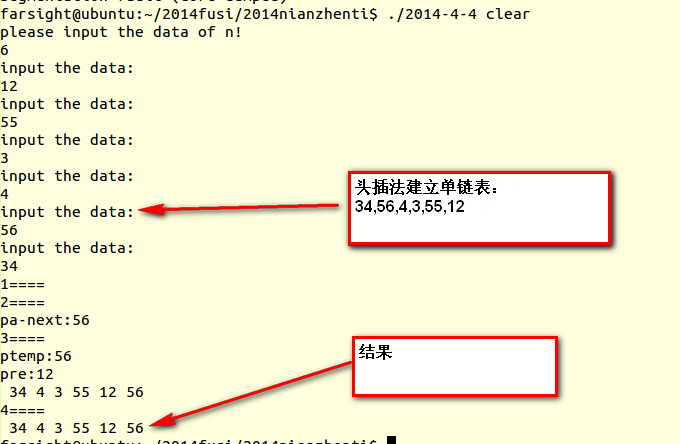
**31 ptemp = pa->next; //ptemp 暂时记下最大值节点**

**33 pa->next =pa->next->next; //将最大值节点从当前位置删掉，未真正删除**

**34 ptemp->next = pre->next; //连接到尾部**

**35 pre->next = ptemp;**

**}**



**======一位女生的，更简洁**

**void maxInsert(Lnode \*L)**

**{**

**Lnode \*p,\*q,\*pre;//pre记录最大值节点的前驱指针**

**int max;**

**max=L->data;**

**q=L;//工作指针**

**p=L;//记录最大值结点**

**while（p->next!=null)**

**{**

**if(p->next ->data >max)**

**{**

**max=p->next ->data ;**

**q=p->next ;**

**pre=p;**

**}**

**p=p->next ;**

**}**

**pre->next =q->next ;//删除q结点即最大值结点**

**p->next =q;**

**a->next=null;**

**}**

LinkList Sort(LinkList &SL){

LinkList p,q;

int temp;

for(p=SL;p!=NULL;p=p->next){

for(q=p->next;q!=NULL;q=q->next){

if(p->data>q->data){

temp=q->data;

q->data=p->data;

p->data=temp;}

}

}}

五、从提高程序执行效率的角度，说明C语言采取哪些措施并指出原因。

老掉牙的C语言如何提高程序执行效率

C语言的措施：

1：指针，可以操作内存

2：位移，可以作位运算

3：API，可以调用系统API，接近底层

4：宏define可以编译的时候替换

