2019 重庆大学 C 上机真题

一、2011 年复试真题

1. 水仙花数。读入文件 number.in 数字每行一个，以 0 输入结束。判断是否是水仙花数，并输入到 number.out 文件中。

如 输入： 153 输出： T

9 F

10 F

0

```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int isNarci(int num);
int main()
{
    int num;
    FILE* in,*out;
    in  = fopen("number.in","r");
    out = fopen("number.out","w");
    if(!in || !out)
    {
        printf("failture to open file!");
        exit(1);
    }
    //循环读写文件+判断
    while(1)
    {
        fscanf(in,"%d",&num);
        if(num == 0)  //如果无 0 判断结束，也可 while(!feof(in))
            break;
        if(isNarci(num))
            fprintf(out,"%s","T\n"); //再错 0:\n 用%s 而不是%c
        else
            fprintf(out,"%s","F\n");
    }
    fclose(in);  //再错 1：不关闭文件流，输出文件不会显示
    fclose(out);
    return 0;
```

```
}
//n 位水仙花数（n>=3）：该数 = 个位数^n + 十位数^n + 百位数^n + ...
int isNarci(int num)
{
    int n = 0;  //判断几位数，使用前要置 0!
    int temp = num;
    int sum = 0; //计算水仙花数
    while(temp != 0) //[背]：获取整数每一位(从低位开始)，并操作模板
    {
        n++;   //操作
        temp = temp / 10;
    }
    if(n <3 )
        return 0;  //易忘
    temp = num;     //再错 2:下面改变 num,上面 temp 也不再是 num,重新赋值
    while(temp != 0) [背]：获取整数每一位(从低位开始)，并操作模板
    {
        //temp = temp % 10; //再错 3:temp 只会改变去低位! temp/10!
        sum += pow(temp%10,n);  //操作
        temp = temp / 10;
    }
    return num == sum;
}
```

2.定义一个正值数组，值由用户输入。删除其最大最小值，并求此时数组平均值。如:

输入： 1 2 3 4 5　　输出：平均值：3

```c
#include<stdio.h>
#define MAX 100

void del(int A[],int len, int p1 , int p2);
int main()
{
  int A[MAX] = {0}; //ATTENTION 0 : {0}是全部初始 0，{1}只 A[0]=1 其余 0
  int i,min,max,len;
  float average , sum = 0;  //易错 1：不给 sum 初始化使用会是奇怪的数
  i = max = min = len = 0;
  /*
    1.ATTENTION 1：连续赋值，ctrl+c 退出
    2.使用 while + scanf 连续赋值必须 getchar()读取空格/回车
    3.可能会有愚蠢的 ox4001005 错误，高冷忽略。
  */
  while((scanf("%d",&A[i++])) != EOF)    getchar();
  len = i - 1;  //玄：输入 5 个数字 i=6
  for(i = 1; i < len ; i++)//[背]：一次遍历寻找到最大和最小
  {
     if(A[i] < A[min])       min = i;
     else if(A[i] > A[max])  max = i;//注意 else if
  }
  del(A,len,min,max);
  len = len - 2; //记录删除后数组长度
  for(i = 0; i < len ; i++)
    sum += A[i];
  average = sum / len;
  printf("平均值 = %f",average);
  return 0;
}
//[背]：o(n)删除数组满足条件的元素，这里指的删除下标 p1、p2 对应数
void del(int A[], int len,int p1 , int p2)
{
  int i , k = 0;
  for(i = 0; i < len ; i++)
  {
    if( i == p1 || i == p2)
        continue;     //结束这次循环，i 指针后移，k 指针不变
    A[k++] = A[i] ; //如果不是要删除的就赋值，i、k 一起后移
  }
}
```

3.实现一个单链表完成基本的：初始化、插入、删除等操作。(写的太丑优化代码待补)

1.可先参照：https://blog.csdn.net/zer1123/article/details/54954014

2.C++真题链表实现，同。

二、2012 年复试真题

1. 判断是否对称素数 。

输入： 121　　输出 ： 121 不是对称素数!

```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int isPrime(int num);
int isSym  (int num);
int main()
{
  int num;
  int is4;  //四位数不会对称素数
  printf("请输入一个数字:\n");
  scanf("%d",&num);
  is4 = num >=1000 && num <= 9999;
  //错误 1: isPrime && isSym 不传参数
  if(!is4 && isPrime(num) && isSym(num))
      printf("%d 是对称素数! \n",num);
  else
      printf("%d 不是对称素数! \n",num);
  return 0;
}

/*
      判断是否素数：一个数若可以进行因数分解，那么分解时得到的两个数一定是
  一个小于等于 sqrt(n)，一个大于等于 sqrt(n)，据此，代码遍历到 sqrt(n)即可，
  因为若 sqrt(n)左侧找不到约数，那么右侧也一定找不到约数
*/
 int isPrime(int num)
 {
     int i ;
     for(i = 2; i <= sqrt(num);i++)
     {
         if(num % i == 0)
             return 0;          //注意如果不是就直接 return 退出函数
     }
     return 1 ;
 }
 //判断是否对称:计算该数的回文数，如 321 计算出 123 ，比较相等否
 //[背]：获取整数每一位(从低位开始)，并操作模板
```

```
39. int isSym  (int num)
40. {
41.     int temp = num;
42.     int sum  =0;
43.     while(temp != 0)
44.     {
45.         //每次计算出该数最低位 temp%10，就加上之前计算的最低位*10 进一位
46.         sum = sum*10 + temp%10 ; //操作
47.         temp = temp /10 ;
48.     }
49.     return sum==num;    //错误 3:不要写 sum == temp! !
50. }
```

2. 因式分解。如：

输入： 90　　输出 ： 90 = 2*3*3*5

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

int* factorization(int num);
int main()
{
 int* factors;    //存储因式分解的因子
 int num ,i;
 printf("请输入想分解的数字：");
 scanf("%d",&num);
 factors = factorization(num);
 printf("因式分解 %d = %d" ,num,factors[0]);//注意先写好 A[0]
 for(i = 1; factors[i] != 0 ;i++)
     printf(" * %d",factors[i]);
 return 0;
}
/*[背]：计算因式分解
  为什么终止条件： i <= temp ？ 强调 = 号？
    1.保证当此时 temp 不能被分解时(如 5)，可以循环到 temp，i=temp 存起来
       注： i <= num ，也可但没必要。
    2.同时下轮循环 temp = 1；不满足 i = 2 <= temp ，退出循环。
*/
int* factorization(int num)
{
    int i , j = 0 ,*factors,temp;
    factors = (int*)malloc(sizeof(int)*MAX);
    memset(factors,0,sizeof(int)*MAX);  //初始为 0，也是终止条件
    temp = num;
    for(i = 2 ; i <= temp ; i++)     //不用两次循环
        {
            if(temp % i == 0)      //找到因子 i
            {
              factors[j++] = i;     //存储因子 i
              temp = temp / i;     //得到被整除后的（temp/i）
              i = 1;    //再错 1: i 重置 1，i++下次循环会变成 2,寻找因子
            }
        }
    return factors;
}
```

3.※从输入文件中输入一串由 26 个字母和数字组成的字符串，统计其各个字符频率并输入至输出文件。

如：输入文件：zcda112　　输出：z:1 c:1　d：1　a：1　1:2　2:1

分析：

1.每个字符对应 ASCII 码 0~255。可设数组 num[256] ，初始为 0。

下标 i 对应字符 ASCII 码值((int)s[i])，值 num[i]对应出现次数。

2.遍历字符串 s，即可统计每个字符的频率

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

int* strFrequency(char s[]);

int main()
{
    FILE *in,*out;
    char s[MAX];
    int* nums,i; //记录字符频率，指针即可
    in = fopen("string.in","r");
    out = fopen("string.out","w");
    if(!in || !out)
    {
        printf("打开文件失败！");
        exit(1);
    }
    fscanf(in,"%s",s);
    nums = strFrequency(s);
    for(i = 0 ; i < 256;i++)//写入文件 out
    {
        if(nums[i]) //如果字符 i（i 是其 ASCII 码值）频率不为 0 就执行
            fprintf(out,"%c:%d\n",(char)i,nums[i]);
    }
    fclose(in);
    fclose(out);
    return 0;
}

//[背]统计字符串频率:利用 ASCII 码 0~256
//返回 char*类型，不能用 printf 不好写，返回数组 int*
```

```c
int* strFrequency(char s[] )
{
    int* nums = (int*)malloc(sizeof(int) * 256 );
    int i ;
    int len = strlen(s); // 玄：下面不能直接用 strlen(s)
    memset(nums,0,sizeof(int) * 256 );//易错 1：初始化前 256*int 字节 0
    for(i = 0 ;i < len ; i++)
        nums[(int)s[i]] ++; //如:A 的 ASCII 码对十进制 64,num[64] = 1;
    return nums;
}
```

4.十进制转二进制。

如输入：11.75　　输出：1011.11

```c
#include<stdio.h>
#include<stdlib.h>
#define MAX 100

char* dectoB(int dec); //ATTENTION 0:这里用 char*比 int*好
int main()
{
    int dec;
    printf("请输入一个十进制数字：");
    scanf("%d",&dec);
    printf("对应二进制数：%s",dectoB(dec));
    return 0;
}

//[背]整数转二进制：除 2 取余，判商赋商，逆序排列
//再错 1:数字 0 1 记得转换成 char 类型,且只：binary[i] = r + '0';
//      binary[i] = (char)r;  错误!
char* dectoB(int dec)
{
   int r; //余数
   int q; //商
   int i , j;
   char* binary = (char*)malloc(sizeof(char)*MAX),temp;
   for(i = 0 ; q != 0 ; i++ )
   {
       r = dec % 2;
       binary[i] = r + '0';
       q = dec / 2;
       dec = q;
   }
   binary[i] = '\0';  //再错 2：字符数组一定要记得设置'\0'
   i = i-1;  //i-1 是此时 binary 最大下标
   for(j = 0 ; j < i ; j++,i-- )  //[背]反转 binary：实现二进制正序
   {
       temp = binary[j];
       binary[j] = binary[i];
       binary[i] = temp;
   }
   return binary;
}
```

5.（找不到）矩阵操作，矩阵字母由中心向四周扩散

6.非负十六进制相加，如：

输入： A B　　输出：15

C D　　19

E F　　1D

分析：

1.可定义 int A B。但接收的时候选择 %X 接收为 16 进制

2.相加时 A+B 其结果 ：(1)%d 输出是对应十进制；(2) %X 是十六进制

3.由于容易溢出，可考虑__int64：

(1)但输出是前面会多 CCC..(这里没有！放眼)

(2)十进制转十六进制那里有[C 练习谭-3 题]，也许是一开始就没按十六进制接收?

```
#include<stdio.h>

/*
  //ATTENTION 1：如果改为%X 输出-1A,-12=ffffffd4 而非 ffffffffffffd4
*/
int main()
{
  __int64 A,B,sum;  //ATTENTION 2:__int64 等价于 longlong,但 vc6 不支持
  while(1)
  {
  printf("请输入两个十六进制，中间空格隔开：-1A +2A\n");
  scanf("%I64X %I64X",&A,&B); //错误 1：%I64X 用来接收长整型 16
  sum = A+B;
  printf("\n%I64X\n",sum);
  }
  return 0;
}
```

三、2013 年复试真题

1. 输出五位数以内全部对称素数。

输出：0 1 2 3 5 7 11 101 131 151 181 191 313 353 373
383 727 757 787 797 919 929

```c
#include<stdio.h>
#include<math.h>

int isPrime(int num);
int isSym(int num);
int main()
{
  int is4 , i;
  printf("5 位数以内的对称素数: \n");
  //四位数不会是对称素数:
 //abba,即 1000a+100b+10b+a=1001a+110b = 11(91a+10b),具有因子 11 合数。
  for(i = 0 ;i <= 9999;i++) //对称素数从几位开始算?
  {
    is4 = i>=1000 && i<=9999;
    if(!is4 && isPrime(i) && isSym(i))
        printf("%d\t",i);
  }
  return 0;
}
//判断是否对称素数
int isPrime(int num)
{
    int i;
    for(i = 2 ; i <= sqrt(num) ;i++ )
        if(num % i == 0) return 0;
    return 1;
}
//判断是否回文数
int isSym(int num)
{
    int temp = num , sum = 0;
    while(temp != 0)
    {
      sum = sum * 10 + temp %10 ;
      temp = temp / 10;
    }
    return sum == num ;
}
```

2. 有一个天平有 6 种砝码，分别重 1,2,3,5,10,20,克，每种砝码各有 5,3,2,2,1,1,个，计算天平能称出的重量和种类。（下面算法实现指定任意砝码个数计算，限制<=1000 克）

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAXKG 1001 //ATTENTION 1: 不是 1000，因为要求还要考虑 1000g
int* farmarWeighing(int n1,int n2,int n3,int n5,int n10,int n20);
int main()
{
  int n1,n2,n3,n5,n10,n20 ,i,* N ;
  printf("请输入 1，2,3,5,10,20 克砝码个数,如：5 3 2 2 1 1\n");
  scanf("%d %d %d %d %d %d",&n1,&n2,&n3,&n5,&n10,&n20);
  //N =farmarWeighing(5,3,2,2,1,1);
  N =farmarWeighing(n1,n2, n3, n5, n10, n20);
  for(i = 1 ;i < MAXKG ; i++ )
  {
      if(N[i]) //重量 i 对应称重方式不为 0
         printf(" %d 克:共有%d 种方式称重\n",i,N[i]);
  }
  return 0;
}
/*
  题目要求求出能称出的重量不大于 1000 克和方式个数，如 1 克 有 3 种方式
  N* :  下标存能称的重量，值存方式个数
*/
int* farmarWeighing(int n1,int n2,int n3,int n5,int n10,int n20)
{
    int i,j,k,l,m,n,sum;
    int* N = (int*)malloc(sizeof(int) * MAXKG);
    memset(N,0,sizeof(int) * MAXKG);
    for(i = 0 ; i<= n1 ;i++) //易错 1: 小于等于号!
        for(j = 0 ; j<= n2 ;j++)
            for(k = 0 ; k<= n3 ;k++)
                for(l = 0 ; l<= n5 ;l++)
                    for(m = 0 ; m<= n10 ;m++)
                        for(n = 0 ; n<= n20 ;n++)
                        {
                          sum = i*1 + j*2 + k*3 + l*5 + m*10 + n*20;
                          if(sum)//易错 2: if(0)不执行，其余正负都执行
                              N[sum] ++; //sum 克对应称重方式+1
                        }
    return N;
}
```

3.输入 n 个由'0'和'1'组成的字符串并排序输出，n 由用户输入，规定比较规则如下：

a.字符串长的比较大

b.字符串等长的含'0'多比较小

c.字符串等长，且含'1'相等，该字符串相等

分析：

1.char *s[10] 和 char (*s)[10] 区别?

(1) char *s[10]：[]优先级更高先和 s 结合，表示 s 是数组，有 10 个 char*

(2) char (*s)[10]：s 先和*结合表示 s 是指针，指向有 10 个元素的数组。

2.为什么定义指针数组？为什么又不定义 char* s[n]?

比较时交换指针即可。数组名是常量就只能 strcpy 开销大；n 要动态指定。

3.为什么冒泡排序逆序?

从小到大输出，冒泡排序从末尾开始筛选最小到前面；正序只能筛选出最大。

但无论正序、逆序都可实现从小到大输出。

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

int strcmp01(char* s1,char* s2);
int main()
{
    char** s,*temp;
    int n;
    int i,j;
    printf("请指定输入 01 字符串个数：");
    scanf("%d",&n);
    s = (char**)malloc(sizeof(char*)*n); //再错 1：指定二维数组大小
    for( i = 0 ; i < n ; i++)
    {
        s[i] = (char*)malloc(sizeof(char)*MAX);
        printf("请输入第%d 串：",i+1);
        scanf("%s",s[i]);
    }
    //[背]尾冒泡排序：不写函数？1.简单 2.每一轮可确定一个数直接输出
    for(i = 0 ; i < n ; i++) //[注]：不写 n-1，方便输出第 n 大字符串
    {
        for(j = n-1 ; j-i > 0 ; j--)//j-i 表示执行 j-i 次，每轮递减
        {
            if(strcmp01(s[j],s[j-1]) == -1)
```

```
            {
                temp = s[j];
                s[j] = s[j-1];
                s[j-1] = temp;
            }
        }
        printf("%s\n",s[i]); //输出每轮最小
    }
    return 0;
}

//比较 01 字符串大小
int strcmp01(char* s1,char* s2)
{
    int len1,len2,s1_cout1 = 0,s2_cout1 = 0;
    int i,j;
    len1 = strlen(s1);
    len2 = strlen(s2);
    if(len1 > len2)  return 1;
    if(len1 < len2)  return -1;
    //字符等长，开始比较 1 个数
    for(i = 0 ; i < len1 ; i++)
        if(s1[i] == '1')  s1_cout1++;
    for(j = 0 ; j < len2 ; j++)
        if(s2[j] == '1')  s2_cout1++;
    if(s1_cout1 == s2_cout1)  return 0;
    if(s1_cout1 >  s2_cout1)  return 1;
    if(s1_cout1 <  s2_cout1)  return -1;
}
```

四、2014 年复试真题

1.利用一维数组打印等腰杨辉三角。

如输入：4　　输出：

```
        1
      1   1
    1   2   1
  1   3   3   1
```

分析：

1.定义一个数组 A[100]={1 0 0 0 ....0}。每层最左 1 不用计算，从第二层开始。

2.从最右侧 1 逆向开始计算(右侧 1 也恰一起计算，符合表达式)：

正向求时：A[1] = A[1] + A[0]　A[2] = A[2] + A[1]　此时 A[1]已被修改会出错

逆向求时：A[2] = A[2] + A[1]　A[1] = A[1] + A[0]　A[2]虽被修改但不影响

3.考虑第 N 层，最右侧 1 对应 A[N-1]，每层需计算 N-1 个数字。那么内循环：

```
for( i = N -1 ; i > 0 ; i-- )
```

```c
#include<stdio.h>
#define MAX 100

int main()
{
    int A[MAX] = {1}, N; //A[0] = 1，其余为 0
    int n , i , space;
    printf("请指定杨辉三角的层数：");
    scanf("%d",&N);
    for(n = 1 ; n <= N ; n++)
    {
        //ATTENTION 1: 第 N 层右移 0 单位，第 N-1 层右移 1 单位..
        for(space = N-n ; space > 0 ; space--)
            printf("  "); //1 单位 = 2 空格
        for(i = n-1 ; i > 0 ; i--) //第 n 层计算杨辉数字
        {
            A[i] = A[i] + A[i-1];
            printf("%5d",A[i]);//从最右 1 开始打印，但对称等价从最左
        }
        printf("%5d\n",A[0]); //再错 1：注意换行
    }
    return 0;
}
```

2.一百位以内大数相加。

输入：112233445566778899 998877665544332211

输出：1111111111111111110

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

char* bigNumAdd(char* a,char* b);
int main()
{
    char a[MAX], b[MAX] ,*result;
    printf("请输入大数 a b，空格隔开：");
    scanf("%s %s",&a,&b);
    result = bigNumAdd(a,b);
 //result = bigNumAdd("112233445566778899","998877665544332211");
    printf("结果为：%s",result);//注：结果第一位可能为 0
    return 0;
}
/*
  大数相加：
  1.[背 1]如何将字符串 a 右移（已知右移后长度 lm）
     (1) 设置 i = la-1,k = lm-1 分别指向移动前/后末尾
     (2) 当 i 大于等于 0 时：a[k] = a[i] //再错 1：注意=号
         当 i 大于等于 0 时：//其他操作如填 0
  2.[背 2]尾数起加，每次存 1，其余进位
*/
char* bigNumAdd(char* a,char* b)
{
    char* result = (char*)malloc(sizeof(char)*(MAX+1));
    int la,lb,lm; //长度
    int i,j,k;    //位移控制
    int ta,tb,t,carry = 0;  //低位相加临时保存
    la = strlen(a);  lb = strlen(b);
    lm = la > lb ? la : lb;
    lm = lm+1;   //再忘 1：此时 lm 表示化为标准后的长度
    for(i=la-1,j=lb-1,k=lm-1; k>=0 ;i--,j--,k--) //不是 k<0
    {
        if( i >= 0)  a[k] = a[i];
        else         a[k] = '0'; //再错 1：字符串添 0 加''
        if( j >= 0)  b[k] = b[j];
        else         b[k] = '0';
```

```
    }
    //已化标准，开始相加，只保存计算的数字最低一位，其余进位
    for(i = lm-1 ; i >=0  ; i--)
    {
        ta = a[i]-'0';
        tb = b[i]-'0';
        t = ta + tb + carry;
        result[i] = t % 10 + '0'; //只保存最低一位
        carry = t / 10;   //其余进位
    }
    a[lm] = b[lm] = result[lm] = '\0'; //再错 2：字符串一定设置末尾
    return result;
}
```

五、（暂缺）2015 年复试真题

六、2016 年复试真题

1. 求每一行数字的最大公约数和最小公倍数。输入文件 numberi.in,输出文件 number.out。

从输入文件中读取数字，每一行两个数字，有多行。

如输入文件：20 8　　输出文件：4 40

15 5　　5 15

8 7　　1 56

分析：

1.最大公约数：穷举法求出 a b 最大公约 g

2.最小公倍数：a*b/g

```c
#include<stdio.h>

int gcd(int num1,int num2);
int main()
{
  FILE* in,*out;
  int num1,num2,g,lcm;
  in = fopen("number.in","r");
 out = fopen("number.out","w");//ATTENTION 0:以"w"每次打开都会清除文件
                    // 因为，"w"打开无论文件是否存在都会重新创建并覆盖
  if(!in || !out)
  {
      printf("打开文件失败！\n");
      exit(1);
  }
  while(!feof(in))
  {
      //错误 1:fscanf 不用指明"r"
     fscanf(in,"%d %d",&num1,&num2);//空格分隔数字
     g = gcd(num1,num2);
     lcm = num1 * num2 / g;//最大公倍
     fprintf(out,"%d %d\n",g,lcm);
  }
  fclose(in);//错误 2：不关闭文件流
  fclose(out);
  return 0;
}
/*
  [背]求最大公约数：穷举法找出最大能被两数整除的
  判断条件： i < sqrt(min) 错误，只能说明不是素数
```

```
                不能找到最大整除数。应该 i <= min
 */
int gcd(int num1,int num2)//ATTENTION 1:不实现 build 会报错
{
  int min,g = 1,i ; //ATTENTION 2:初始 g 为 1，防止没有最大公约
  min = num1 < num2? num1:num2;
  for(i = 2; i <= min;i++ ) //错误 3:<=! 最小数是最大公约情况
  {
    if( num1 % i == 0 && num2 % i == 0 )
    {
      g = i; //寻找最大公约数
    }
  }
  return g;
}
```

七、2017 年复试真题

1.字符串循环加密(1)

(相似例题)输入一串字符（长度不超过 100）和一个正整数 k，将其中的英文字母加密并输出加密后的字符串。其中 a-z 、A-Z 循环加密，如 z +偏移 k=1 时，是 a。Z+偏移 k= 1 时 ，是 A；非英文字母不变。

输入： 12kjsdZjk 280 输出： 12edmxTde

分析：

ASCII 值： 41 42 43 44 45 46 ... 96

加密字符串： A B C D E F ... Z

相对位置： 0 1 2 3 4 5 ... 25

1.str[i] - 'A' == str[i]的 ASCII 值 - 'A'的 ASCII 值 ，如： 45(E) - 41(A) = 4

（ **char 本质是整型**，存储的是 ASCII 值只有在输出%c 才显示对应字符）

2.但不是-'0'！我们要获得相对 A 位置偏移，将 ASCII 码值可视为下标则：

[1] str[i]存的就是字符数组下标,-'A'下标使得第一个下标为 0,%26 就可以在 0-25 取值（A-Z 整体下标左移 ‘A’个单位）。

[2]同时因为存的是下标，不会出现 int 数组：A[i] = A[(i+K)%n]

A[0] = A[0+K] A[1] = A[1+K]... A[n] = A[0]

此时 A[0]值已被覆盖成 A[K]问题。因为我们只计算出下标（ASCII），和值无关。

3.易错 1：+'A' 得到 E 加密后真实下标 （ASCII 值）


```c
#include<stdio.h>
#include<ctype.h>  //ATTENTION 1 :调用判断是否大小写字母
#include<string.h>
#define MAX 100

int main()
{
    char s[MAX];
    int k , i;
    printf("请输入一串字符串和偏移 k，空格隔开：");
    scanf("%s %d",s,&k);
    for(i = 0 ; i < strlen(s) ; i++)
    {
        if(islower(s[i]))  s[i] = (s[i] - 'a' + k ) % 26 + 'a';
        if(isupper(s[i]))  s[i] = (s[i] - 'A' + k ) % 26 + 'A';
    }
    printf("加密后的字符串：%s",s);
    return 0;
}
```

2.字符串循环加密(2)

已知 ASCII 码对应值 [0-255],将用户指定输入的一串字符数组 s 对应每个字符 s[i]进行加密和解密，加密即该字符循环向后偏移 i+5 个 ASCII 码值。如 a、b 对应 ASCII 码值 97、98。f、h 对应 ASCII 码值 102,104。

输入：ab　　输出：加密后：fh

输入：fh　　输出：解密后：ab

分析：

1．此题同前。无外乎计算字符偏移后的 ASCII 码值（下标）：

加密：　s1[i] = s[i] + i + 5... 式 1 //注：不用取余，超过 255 自动取余

解密：　根据加密式 1，可反解加密前 s[i] = s[i] - i - 5

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

char* encrypt(char* s);
char* decode(char* s);
int main()
{
  char s[MAX] , str_encrypt[MAX];
  gets(s);   //gets()可读入空格
  str_encrypt = encrypt(s);
  printf("加密后：%s\n",str_encrypt);
  printf("解密后：%s\n",decode(str_encrypt));
  return 0;
}

//加密字符
char* encrypt(char* s)
{
  char *s1 = (char*)malloc(sizeof(char)*MAX);
  int i;
  for(i = 0 ; i < strlen(s); i++)
  {
      s1[i] = s[i] + i + 5; //超过 255 会自动取余
  }
  s1[i] = '\0';//无数次错误 1：  字符指针末尾加'\0'
  return s1;
}
```

```
30.
31. //解密字符
32. char* decode(char* s)
33. {
34.   char *s1 = (char*)malloc(sizeof(char)*MAX);
35.   int i;
36.   for(i = 0 ; i < strlen(s); i++)
37.   {
38.       s1[i] = s[i] - i - 5;
39.   }
40.   s1[i] = '\0';
41.   return s1;
42. }
```

3.找出指定整数 N 范围内的所有 2 的幂次数，并输出。如：

输入：10　　输出：　2　4　8

分析:

此题，不封装函数，直接在主函数写代码会简单很多。但为了展示 int 数组的返回和不知长度下如何设定终止条件使用，还是不做改变。其余关于位运算细节见注释。

```
1.  #include<stdio.h>
2.  #define MAX 1000
3.
4.  int* findPower2(int N);
5.  int main()
6.  {
7.    int *p2,N ,i;
8.    scanf("%d",&N); //ATTENTION 0：gets(&N)错误，只能是字符串
9.    p2 = findPower2(N);
10.   //错误 1：获取 int 数组不能 strlen(p2),只能 sizeof(数组名)/sizeof(int)
11.   for(i = 0 ; p2[i] != -1; i++)
12.      printf("%4d",p2[i]);
13.   return 0;
14. }
15.
16. /*
17.   函数：返回所有指定 N 内是 2 的幂次数
18.   分析：一个数 a 是 2 幂次，则二进制必是 100..0 形式，a-1 是 011..1 形式
19.         它们按位与 & 则必为 000..0
20. */
21. int* findPower2(int N)
22. {
23.   int *p2 ,i,k = 0 ;
24.   p2 = (int*)malloc(sizeof(int)*MAX);
25.   for( i = 2 ; i < N ; i++)
26.   {
27.       if((i & i-1) == 0)
28.           p2[k++] = i;
29.   }
30.   p2[k] = -1;//设置终止条件
31.   return p2;
32. }
```

4.随机生成不大于 100 的 10 个不重复的数字。如：

输出： 5 6 12 78 66 11 9 5 2 3 1

```c
#include<stdio.h>
#include<time.h> //注意引入时间库

int exist(int r[],int n);
int main()
{
  int r[10];
  int i ,temp;
  srand((int)time(0));//ATTENTION 0:设置随机数种子，防止序列一致
  for(i = 0 ; i < 10 ; i++)
  {
    temp = rand() % 101 ; //101
    if(!exist(r,temp)) //ATTENTION 1 : !0 = 1,!非 0 = 0
    {
       r[i] = temp;
       printf("%5d",r[i]);
    }
  }
  return 0;
}

//判断是否存在重复
int exist(int r[],int n)//[好]这里定义函数比直接写 main 里好
{
    int i;
    for(i = 0; i < 10 ; i++)
    {
      if(n == r[i]) //有重复
           return 1;
    }
    return 0;
}
```

八、2018 年复试真题

1.(改错)由控制台打开程序，附带 n 个命令行参数，将参数升序排序并输出

输入命令行参数： welcome to our new world

输出： E:\Second\Debug\2.exe

new our to welcome world

分析：

1.[背]冒泡循环的两种模板：

for(i = 0 ; i < n-1; i++) //n 数只需 n-1 轮

for(j = 0 ; j < n-1-i ; j++) //执行 n-1-i 次，递减

if(满足条件) //do something

for(i = 0 ; i < n-1; i++) //n-1 轮

for(j = n-1 ; j -i> 0; j--)//执行 j-i 次递减

if(满足条件) //do something

2.[背]设置命令行参数（英文 VC++6.0）

①打开：project -> setting->debug -> program arguments

②输入：welcome to our new world (空格分隔)

argv[i] [1] [2] [3] [4] [5] （args[0]是路径名称默认）


```
#include<stdio.h>
#include<string.h>

int main(int argc,char* argv[]) //二维字符数组
{
  int i , j , flag ;
  char* temp;
  for(i = 1 ; i < argc - 1 ; i++) //冒泡 n-1 轮，这里第 0 次不排
  {
    flag = 0 ; //哨兵
    for(j = argc-1; j - i > 0 ; j--)//执行 argc-1-i 次
    {
      if(strcmp(argv[j-1],argv[j]) > 0)//错误 0：>0,-1 也会执行
      {
        temp = argv[j];argv[j] = argv[j-1];argv[j-1] = temp;
        flag = 1; //哨兵，发生交换
      }
    }
    if(flag == 0)       break;
  }
  for(j = 0 ; j < argc ; j++)   //输出
      printf("%s\n",argv[j]);
  return 0;
}
```

2.从键盘输入一个字符串，并将其中的小写字母全部转换为大写字母，并将结果保存在 E:\test.txt 文件中。如：

输入：abcdABCD　　输出：ABCDABCD

```c
#include<stdio.h>
#include<ctype.h>
#define MAX 100

int main()
{
    FILE* out;
    char s[MAX];
    int i ;
    gets(s);
    for(i = 0 ; s[i] != '\0';i++)
    {
        if(islower(s[i]))   s[i] = toupper(s[i]);
    }
   out = fopen("E:\\test.txt","w"); //ATTENTION 0 :fopen 参数没 out!
    if(!out)
         exit(1);
    //2019.2.20 10:43 我是个傻 zi，调试半小时原来是 printf 没 f,fprintf!
    fprintf(out,"%s",s);
    fclose(out);  //易错 1 : 记得关文件流
    return 0;
}
```

九、面试算法题真题

1.给 n 个整数 x1..xn，找出最小的整数。请用二分递归法来实现-分成两半递归，各解出最小值然后比较。

分析：{5,7,2,4,6}

递归表达分析：n 个数递归查找最小数**（规模 n）**：

(1)前 n/2（向上取整）个数 查最小**（规模 n/2）**

(2)后 n/2（向下取整）个数 查最小**（规模 n/2）**

(3)比较这两部分子串最小值大小，返回最小值

递归参数分析：

(1) 要二分子串则必然需要指示改子串起始，故传 s、r

如 F(0,4) --> F(0,2) F(3,4)

递归终止条件分析：

```
                          F(0,4)
 {5  7  2}        F(0,2)              F(3,4)              {4 6}
{5 7} {2}    F(0,1)    F(2,2)        F(3,4)  F(4,4)          {4} {6}
{5} {7}    F(0,0)  F(1,1)
```

(1)当 s==r :**缩小定位到子串只有 1 个数**，此数便是该子串最小，返回

(2)当 s<r :可再划分两个子串，比较两子串最小，返回更小那个

①如执行 F(0,0) 返回 5；接着执行（**递归同层先执行**）F(1,0) 返回 7

②由于该层都执行完，执行上一层 F(0,1)未执行的**(*)**代码返回最小 5


```c
#include<stdio.h>
int FindMin(int A[],int s ,int r);
void main()
{
    int A[] = {5,7,2,4,6};
    int min;
    min = FindMin(A,0,4);
}
//递归查找最小值
int FindMin(int A[],int s,int r)
{
     int k1 , k2;
     if( s == r)       return A[s];
     k1 = FindMin(A,s, (s+r) / 2); //找前 n/2（向上取整）个数最小
     k2 = FindMin(A,(s+r) / 2 + 1 ,r);
     return k1 < k2 ? k1 : k2; //(*)
}
```

2.编程计算 n 条直线可以将 1 个平面分成多少个子平面？ 如：

输入 ： 100 输出： 5051

分析：

1.设 n 条直线可分为 f(n)个平面；第 n-1 条直线可分为 f(n-1)个平面。

2.n-1 条直线增加第 n 条直线，即可分为 f(n)个平面。若要增加的第 n 条直线划分的子平面最多：第 n 条直线和原 n-1 条直线需产生最大 n-1 个交点。

3.这些交点将第 n 条直线划分出：2 条射线 + n-2 条线段。这些射线和线段，每一条都会将平面多划分出一个子平面(可画图辅助理解)，即 f(n) - f(n-1) = n。

以上，递推表达式为：

f(n) = n + f(n-1) f(1) = 2

```c
#include<stdio.h>

int f(int n);
int main()
{
  int n;
  scanf("%d",&n);
  printf("%d",f(n));
  return 0;
}

//求 n 条直线可将一条平面分成多少个子平面
//递归推
int f(int n)
{
    if(n == 1)  return 2;
    return n+f(n-1);
}
```

3.已升序有序的数组 A[N],指定正整数 M。使用二分查找法，查找在 A[N]中和 M 相等数的位置，返回其下标 k； 如果没有找到则返回 A 中最大的 <M 数的下标 k。如：A[5] = {1 , 3 , 15 ,70 , 108} ,查找 M=20，返回 k = 2。

输入： 20　　输出： 2

分析：

递归表达分析：n 个数递归查找<=M 数下标 k **（规模 n）**：

(1)将 M 和 n 数中值 mid 比较，判断在左/右子串

(1)在左子串查**（规模 n/2）**

(3)else 在右子串查**（规模 n/2）**

递归参数分析：

同 ：九、1

递归终止条件分析：

(1) 查找到 M =A[mid]成功，返回下标 mid（好理解）

(2) else 考虑查找 M 失败时：s > r。**不可能 s==r,缩小定位到子串只有 1 个数查找一定是成功了，**返回最接近<M 的数下标 r(数在 r 右边才会 s>r) 。


```
1.  #include<stdio.h>
2.  int findM(int A[],int M ,int s , int r);
3.  void main()
4.  {
5.      int k , M;  //k 记录位置最近 M 数在 A 中下标
6.      int A[] = {1 , 3 ,15 ,70 , 108};
7.      scanf("%d",&M);
8.      k = findM(A,M,0,4);
9.      printf("%d",k);
10. }
11. //ATTENTION 0：考虑 M=20 s=3 r=4 mid = 3
12. //此时满足左侧，findM(A,M,3,2) 3<2
13. int findM(int A[],int M ,int s , int r)
14. {
15.     int mid = (s+r)/2;
16.     if(A[mid] == M)  return mid;
17.     if(s > r)  return r;
18.     if(M < A[mid]) //M 在左侧
19.         findM(A,M,s,mid-1); //[-1?] 1.mid 已经考虑 & 2.查找失败时会使 s>r
20.     else//右侧
21.         findM(A,M,mid+1,r);
22. }
```