2019　　C 经典 100 例（部分）

100. 有 2 个学生，每个学生有 2 门课的成绩，从键盘输入以上数据（包括学生号，姓名，二门课成绩），计算出平均成绩，况原有的数据和计算出的平均分数存放在磁盘文件"stud"中。

分析:

1. `struct student  s[n];` 要求按 s.float(或者某课程成绩)**排序输出**？
   （1）这里不能交换指针。但可以像元素一样交换 stuct student 变量
   （2） `struct student  temp;` （建议像 stu 一样，旁边再写个 temp）
   `temp = s[j]  s[j] = s[j+1]  s[j+1] = temp`

```c
#include<stdio.h>

struct student
{
    int ID;
    char name[10];
    int cpp;
    int c;
    float average;
}stu;      //ATTENTION 1 : 结构体变量声明方式 1
int main()
{
  FILE* out;
  struct student  s[2]; //ATTENTION 2 : 结构体变量声明方式 2
  int i;
  for(i = 0 ; i < 2 ;i++)
  {
    //易错 1：以逗号为分隔符会产生意料之外的错误，建议空格或*c
    scanf("%d %s %d %d",&s[i].ID,&s[i].name,&s[i].cpp,&s[i].c);
    s[i].average = (s[i].cpp + s[i].c) / 2.0;
  }
  //写入文件
  out = fopen("stu.out","w");
  for(i = 0 ; i < 2 ;i++)
  {
     fprintf(out,"%d,%s,%d,%d,%f\n",s[i].ID,s[i].name,s[i].cpp,s[i].c,s[i].average);
  }
  fclose(out);
  return 0;
}
```

99.有两个磁盘文件 A 和 B,各存放一行字母，要求把这两个文件中的信息合并（按字母顺序排列），输出到一个新文件 C 中。

输入文件：　文件 1：　abAB　　文件 2：　cdCD

输出文件：　ABCDabcd

```c
#include<stdio.h>

void mysort(char* s); //排序传指针方便
int main()
{
  FILE* in1,*in2,*out;
  char s1[100],s2[100];
  in1 = fopen("txt1.in","r");
  in2 = fopen("txt2.in","r");
  out = fopen("txt3.out","w");
  if(!in1 || !in2 || !out)
      exit(0); // 崩溃 1:调试半小时是写成了 exist(0)
  fscanf(in1,"%s",s1);
  fscanf(in2,"%s",s2);
  strcat(s1,s2);   //ATTENTION 1 :strcat()连接字符串
  mysort(s1);
  fprintf(out,"%s",s1);
  fclose(in1);
  fclose(in2);
  fclose(out);
  return 0;
}

//[排序模板]字符冒泡排序升序：按 ASCII 码
void mysort(char* s)
{
    int len = strlen(s);
    int i , j;
    char temp;
    for(i = 0 ; i < len-1;i++)
        for(j = 0 ; j < len-1-i;j++) //ATTENTION 2:不是 len-1
            if(s[j]-s[j+1] > 0)//再错 1:比较字符直接比,别用 strcmp()!
            {                          //因为，strcmp()参数是字符指针
              temp = s[j];  s[j] = s[j+1];  s[j+1] = temp;
            }
    }
}
```

95. 输入一串主串，一串子串，计算子串在主串中出现的频率。如:

输入：ABCCDFEFCDCDFFCD　　输出：2

CDF

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

int match(char* mainstr,char* substr);
int main()
{
    char mainstr[MAX],substr[MAX];
    int times;
    printf("请分别输入主串和子串，空格隔开：\n");
    scanf("%s %s",mainstr,substr);
    times = match(mainstr,substr);
    printf("子串在主串出现的频率是：%d",times);
    return 0;
}
/*
 [背]计算子串在主串中匹配成功的次数：一次遍主，但不到尾。i 赋变 k，减一复 j。
    1.一次遍主，但不到尾。
    2.i 赋变 k。每次 i 进来一次都要用 k=i，遍历子串是否匹配。
    3.减一复 j。判断 j==len，匹配成功 i 位移 slen-1，j 复原到 0（易错）
*/
int match(char* m,char* s)
{
    int mlen = strlen(m),slen = strlen(s);
    //int flag = 1; //标识匹配成功子串[可 j==len 判断,同时免去循环 break]
    int i,j,k;
    int times = 0; //再错 0：注意初始化
    for(i = 0 ; i <= mlen-slen ; i++)//再错 1：一次遍主，但不到尾
    {
        if(m[i] == s[0])//和子串首匹配[此判断可省&&可同时省去 j=0]
        {
         for(k =i+1,j=1; m[k]==s[j] && j<slen;k++,j++);//kj 后移判断
        }
        if(j == slen)
        {
            times++;
            i += slen-1; //ATTENTION 0:这里还会 i++，故减 1
            j = 0; //再错 1：防止下次匹配失败循环错误判断 j==len
        }
```

```
    }
    return times;
}
```

更简洁的代码 2：

```
/*
  更好的代码：统计子串出现次数
    1.i <= mlen - slen，减少匹配次数，同时 for 循环条件不用担心 k>=melen
    2.k=i,不用存 i 初始值，i 值不会被改变。下次循环一定执行下一个
       但是匹配成功时：也要指向下一个，不会 i+子串长度减少匹配(已优化)
    3.同时 s[j] == m[k]写在 for 里，起到不满足就 break 效果，减少代码
  */
int betterCount(char* m,char* s)
{
  int mlen = strlen(m);
  int slen = strlen(s);
  int count =0 , i ,j ,k;  //k 暂存 i 比较
  for(i = 0 ; i <= mlen - slen  ; i++) //小于子串长度就不匹配，注意=
  {
      for(j = 0 , k = i ; j < slen && s[j] == m[k] ; j++,k++) ;
      if(j == slen)
      {
        count++;
        i = i + slen-1; //优化 1：匹配成功,i 直接后移 slen-1.注意-1，因为 i 还会进入循环 i++
      }
  }
      return count;
}
```

86. 回答结果（结构体变量作为形参）

```
#include<stdio.h>

struct student
{
    int x;
    char c;
} a;

int main()
{
    a.x=3;
    a.c='a';
    f(a);
    printf("%d,%c",a.x,a.c);
}
f(struct student b)
{
    b.x=20;
    b.c='y';
}
```


答：

程序运行结果: 3 ,a

结构体变量作形参，是值传递。若想改变值，应该传指针：

```
struct  student f(struct student *b)
{
    b ->x = 20;
    b->c ='y';
}
```

83. 将一个偶数拆分成任意两个素数之和。如：

输入： 8　　输出：3 ， 5

分析：

这里其实就是[**将某数 n 拆分成 ab 之和/积/差等 a 和 b 还要分别满足一些条件**]模板。

[背]处理：遍历 a 所有可能，而 b 利用 n 和 a 计算，在判断 ab 是否满足条件。例如：

1.此题。for(a = 0 ; a <=8;a++) //a 可能为 0-8

if(isprime(a) && isprime(n-a)) //b 是 n-a，且判断二者是否素数

2.华科 OJ：1109.输出所有 abcde/fghij=n 表达式。

//改为遍历 fghij 更好，防止 n/a 小数，结合利用 n 计算 abcde

for(abcde = 12345 ; abcde<=n;abcde ++)

if(isdiffrent(a) && isdiffrent(n/a)) //b 是 n/a，且判断二者是否不同

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

int isPrime(int num);
int main()
{
  int n , i;
  scanf("%d",&n);
  for(i = 2 ; i <= n ; i++)
  {
      if(isPrime(i) && isPrime(n-i))// i+n-i = n,巧妙之处
          printf("%d,%d\n",i,n-i);
  }
  return 0;
}
/*
  判读是否是素数，灵异事件：2019/2/23 12:38
  不加 math.h，sqrt()不报错，但是不生效。最后 i=num%num==0，非素数
*/
int isPrime(int num)
{
    int i;
    for( i = 2 ; i <= sqrt(num) ; i++) //注意=
    {
        if(num % i == 0)
          return 0;
    }
    return 1; //ATTETION 1:1 是素数，这里也会直接返回 1 不用判断
}
```

81.将一个 8 进制数转为 10 进制。如：

输入：1234567　　输出：342391

分析：

此题有两种思路：

1.利用前循环获取低位操作模板（从低位开始）：

```
while(temp != 0)
    sum += (temp%10)pow(8,i++);    temp = temp /10 ;
```

2.循环获取高位（从高位开始）尤其适合 X 进制转 10 进制，需要注意：

(1) 只能用%s 获取字符串，因为高位获取取余无法实现

(2) [背]X 进制转 10 进制模板：

```
while(i < strlen(s)) //改成 for 循环更方便
    sum += sum * X + s[i] -  '0' ;
```

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

int _8to10(char* s);
int main()
{
    char s[100];   //易错 1：直接定义指针不初始化
    scanf("%s",s); //注：获高位%s，且循环获取低位操作模板最好%d(有%、/等操作)
    printf("%d",_8to10(s));
    return 0;
}

/*
  八进制转十进制：01234567（从高位开始操作）
     1.从最高位开始，初始：sum = 0
     2.每一位：  sum = sum*8 + s[i] -'0';
    (相当于每循环增加一低位，前面数字权重*8)
*/
int _8to10(char* s)
{
    int sum = 0; //
    int i;
    for(i = 0 ; i < strlen(s);i++)
    {
       sum = sum * 8 + s[i] -'0';
    }
    return sum;
}
```

76.指针的指针，填空练习。

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char *s[]={"man","woman","girl","boy","sister"};
    char **q;
    int k;
    for(k=0;k<5;k++)
    {
        ___(1)___;      /*在这里填入内容*/
        printf("%s\n",*q);
    }
    return 0;
}
```

答案：

写入：q = &s[k];

这里不把 `char **q` 理解为是二维数组，而且指向指针的指针。如果 q 是字符指针 `char *q` 那么直接输出 q 就行。这里*q，说明 q 是指针并指向了一个字符指针。那么如何指向字符指针？

类似 `int* p = &a;` `q = &s[k];`

75.利用指针函数。编写一个函数，输入 n 为偶数时，调用函数求 1/2+1/4+...+1/n,当输入 n 为奇数时，调用函数 1/1+1/3+...+1/n。如：

输入： 4　　输出：0.75

输入： 5　　输出：1.533333

分析：

1.函数指针/普通指针定义

普通指针： double *ptr;

函数指针： double (*ptr)(int a); //1.增加参数列表 2.*指针增加括号表示 ptr 是指针

(类似: double *p[2] 指针数组 区分 double (*p)[2] 指向数组指针)

.

```c
#include<stdio.h>

double sumEven(int n);
double sumOdd(int n);
int main()
{
  int n;
  double (*psum)(int),result;//注：或定义函数 double sum(double (*f)(int))
  scanf("%d",&n);
  if( n % 2 ==0)     psum = sumEven; //没有括号，加括号只有使用函数时
  else               psum = sumOdd;
  result = (*psum)(n); //再错 1：如何使用函数指针（调用其绑定的函数）！！
  printf("%lf",result);
  return 0;
}
//求偶数和：1/2+1/4+...+1/n
double sumEven(int n)
{
    double sum = 0;
    int i;
    for(i = 2; i <= n; i+=2)//灵异事件:又调试半小时！i+2 错误，应该 i+=2!
        sum += 1.0/i;
    return sum;
}
//求奇数和：1/1+1/3+...+1/n
double sumOdd(int n)
{
    double sum = 0;
    int i;
    for(i = 1; i <= n; i+=2)
        sum += 1.0/i;
    return sum;
}
```

72-74：链表，待填坑

67. 有 n 个整数，使其前面各数顺序向后移 m 个位置，最后 m 个数变成最前面的 m 个数。

输入：1 2 3 4 5 6 （依次输入，且>0）　　输出：4 5 6 1 2 3

3

分析：循环右移动数组 A [6] = {1 2 3 4 5 6}

思路 1：（1）先考虑如何右移 1 个单位？为了避免思考末位 6 移动越界循环移

1.将 6 暂存，12345 往后移动 A[i] = A[i-1]

2.最后 6 右移动一位： A[0] = 6

(2)要移动 m 位，则重复(1)操作 m 次即可。

思路 2：对比字符向后偏移：s1[i] =(s[i]+m)%256; //会自动取余 256 可省

（1）再定义一个数组 A1[6]

（2）A1[i] = A[(i+m)%6]

二者不同：s1、s 相当存字符对应下标(ASCII 值)，s1[i]保存字符偏移后的下标

A1、A 存的是值，A1[i]保存字符偏移后的下标所对应值

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

void move1_m(int A[],int N ,int m);
void move2_m(int A[],int N ,int m);
int main()
{
    int A[MAX] = {0}; //控制终止：1.设置-1 等元素 2.设置长度
    int i ,len, m ;
    /*
     汇总：1.用 ctrl+c 结束会有不可意料错误
          2.可能会输入-1，但只能这么判断。读入字符%c 再转数字只能转 0-9
          3.[注]这里不用 getchar()
    */
    printf("请输入数组元素，输入-1 回车结束，元素空格相隔：");
    for(i = 0 ; scanf("%d",&A[i]) && A[i] != -1 ; i++);
    len = i;
    printf("请指定数组偏移 m: ");
    scanf("%d",&m);
    move1_m(A,len,3);
    //move2_m(A,len,3);
    for(i = 0 ; i < len ; i++)
        printf("%d  ",A[i]);
    return 0;
}
```

```
/*
   指定数组前 N 位循环右移 1：为什么传长度 N，-1 不就可以判断终止吗？
        因为：为了可维护性，如果改变终止条件不是-1 呢？
*/
void move1_m(int A[],int N,int m)
{
    int A1[MAX] = {0};
    int i;
    for(i = 0 ; A[i] != -1 ; i++)
    {
        A1[i] = A[(i+m)%N];
    }
    memcpy(A,A1,sizeof(int)*N);//再错 1: int 数组复制，指定字节
}

//循环右移思路 2
void move2_m(int A[],int N,int m)
{
   m = m % N;  //m 取余
   int temp , i ,j; //temp 记录 A 尾元素 A[N-1]
   for(i = 0 ; i < m ; i++) //偏移 m 移动 m 次
   {
       temp = A[N-1];
       for(j = N-1; j > 0 ; j--)
       {
           A[j] = A[j-1]; //犯错:A[j-1] = A[j]
       }
       A[0] = temp; //尾元素偏移+1 = 首元素
   }
}
```

34. 将用户输入的字符串反转存储，然后输出。如：

输入：2019-CQU-The cannon fodder of the second battalion commander

输出：rednammoc noilattab dnoces eht fo reddof nonnac ehT-UQC-9102

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

void reverse(char s[]);
int main()
{
    char s[100] = "2019-CQU-The cannon fodder of the second battalion commander";
    int i = 0;
    //while( s[i++] != '\0' ); //i 指向末尾'\0'处
    reverse(s);
    while( i < strlen(s) )
        printf("%c",s[i++]);
    return 0;
}

//[背]反转模板
void reverse(char s[])
{
  int len = strlen(s);
  int i , j ;
  char temp;
  //ATTENTION 1: i<j 不是 i<=j
  for(i = 0 , j = len-1 ; i < j ; i++,j--)
  {
    temp = s[i];
    s[i] = s[j];
    s[j] = temp;
  }
}
```

33. 用户输入指定 n，求 1+2!+3!+...+n!的和。如：

输入：4　　输出：33

//注意观察前一项和后一项的关系就好。

current = 1　sum = 0　　i=1

for

current　= current *i

sum　+= current

31.用户输入指定 n，有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13...求出这个数列的前 20 项之和。

输入：2　　输出：3.5

```c
#include <stdio.h>

int main()
{
   int i,j;
   float sum=0;
   float a=1,b=1;
   for(i=1;i<=20;i++)
   {
     j = a+b; //分母等于前两项之和
     sum += j/i;
     a=b;
     b=j;
   }
   printf("%9.6f\n",sum);
}
```

21.打印指定层数的菱形：

输入：3

输出：

```
  *
 ***
*
```

分析：

1.对于上半层第 i 层来说：顺推 * 个数易得：2i-1;逆推 空格 个数：

第 N/2+1 层： 0 空 ；第 N/2 层：1 空 ;---->第 i 层：N/2+1 -i 空格

2.下半层，恰相反。顺推 空格 个数易得：i-(N/2+1);逆推 *个数：

第 N 层：1 个* ；第 N-1 层 3 个* ------> 第 i 层：2 (N-i) +1 个*

```c
include<stdio.h>

void printSpace(int n);
void printSpark(int n);
int main()
{
    int N;
    int i;
    printf("请指定要打印的菱形层数：");
    scanf("%d",&N);
    for(i = 1 ; i <= N ; i++)
    {
        if( i <= N/2+1 )//上半层
        {
            printSpace(N/2+1 - i); //先打印空格
            printSpark(2*i-1);     //再打印*号
        }
        else//下半层
        {
            printSpace(i - (N/2+1)); //先打印空格
            printSpark(2*(N-i)+1);    //再打印*号
        }
    }
    return 0;
}
void printSpace(int n)
{
    int i;
    for(i = 0 ; i < n ; i++)  printf(" ");
}
void printSpark(int n)
{
    int i;
    for(i = 0 ; i < n ; i++)  printf("*");
    printf("\n");  //再错 1：打完*号就要换行了
}
```

18.一个数如果恰好等于它的因子之和（1 是，本身不是），这个数就称为"完数"。例如 6=1 + 2 + 3.编程找出 N 以内的所有完数。

输入：1000　　输出：6　28　496

```c
#include<stdio.h>
#define N 1000
#define MAX 100

int isPerfectNum(int num);
int main()
{
  int i;
  printf("%d 以内的全部完全数是：\n",N);
  for(i = 2 ; i <= N ; i++)
  {
      if(isPerfectNum(i))
          printf("%d  ",i);
  }
  return 0;
}
/*
  [背]判断是否完全数：求完全+数组求和
      [注]这里不是因式分解，而且能被 num 整除的所有数（除本身）
*/
int isPerfectNum(int num)
{
    int  temp = num , factors[MAX];
    int  sum = 0;
    int i , j ;
    for(i = 1 ,j = 0 ; i < temp ; i++)
    {
        if(temp % i == 0)
        {
            factors[j++] = i;
            sum += i;
        }
    }
    return sum == num;
}
```

8.输出 9*9 口诀。如：

输出： 1*1=1

2*1=2 2*2=4

3*1=3 3*2=6 3*3=9

. ....

9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81

分析：

九九乘法表很明显可以判断是两层循环，可以先思考内层循环，并特殊化：

假设打印第 4 层： for(j = 4 , k = 1 ; j > 0 ; j--) //打印 4 组等式

printf(“%d*%d = %d”,j,k,j*k);

那么显然，外层循环就是改变 j，j = 4 ----> j = i.

```c
#include<stdio.h>

int main()
{
    int i , j , k ;
    for( i = 1 ; i <= 9; i++)
    {
        for(j = i, k = 1 ; j > 0 ; j--,k++)  //打印 j 组等式
        {
            printf("%d*%d = %d",i,k,i*k);
            printf(" "); //ATTENTION 1 :打印 1 空格，直接上面\t 打印太宽
        }
        printf("\n");//换行
    }
    return 0;
}
```

1.有 1、2、3、4 个数字，能组成多少个千百十位互不相同无重复数字的三位数？都是多少？

输出：　123　124 ...432

```c
#include<stdio.h>
#define N 5 //再错 1：可能出现的数字是 4 个但要写 5 0-4

int isdifferent(int num);
int main()
{
/*
   int i , j ,k ,n;
  //1.利用砝码问题思想：解决计算千百位不重复三位数
   //之前的问题 1:n = i*100+j*10+k 和 sum = i*5+j*2+1 不同:n 一定不会重复出现!
   for(i = 1 ; i <= 4 ; i++ )//千
       for(j = 1 ; j <= 4 ; j++ ) //百
           for(k = 1 ; k <= 4 ; k++ ) //十
           {
               if(i!=j && i!=k && k!=j ) //ijk 不同 n 一定不会重复
               {
                 n = i*100+j*10+k;
                 printf("%d\n",n);
               }
           }
*/
  int i;
  for(i = 1234 ; i <= 4321 ; i++)
      if(isdifferent(i))
          printf("%d\t",i);
  return 0;
}
//[背]2.判断 num 是否数不同：一位数组+循环获取低位操作模板
int isdifferent(int num)
{
    int A[N] = {0},low;
    int temp = num;
    while(temp != 0)
    {
        low = temp%10;
        if(low==0 ||low>4 ||++A[low]>1)//再错 2：还要判断只能出现 1-4
            return 0;
        temp = temp/10;
    }
    return 1;
}
```

2019　　HUSTOJ 题（选做）

1101.高精度计算，求 n（<100）的阶乘。如：

输入：　19　　输出：121645100408832000

分析：

1.这个问题可以分为两个子问题：

①大数 a 和非大数 b 相乘：

（1）b×a 每一位低位：[和前模板不同，int[]不能取余]循环获取每一位低位+操作

（2）操作：b 乘 a 低，结果存低，其余进位

②多个大数 a 和非大数 b 相乘，循环计算。

2.和前面大数相加不同：【大数相加用 int[]也应该会更好，但没尝试】

（1）这里用 int 数组而非 char 数组保存计算的 a、b 和结果。原因如下：

①免去字符操作数组的转化、设置末尾等操作


```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 1000

int* multiply(int a[],int b);
int main()
{
    int N;
    int i;
    int* result = (int*)malloc(sizeof(int)*MAX);
    memset(result,-1,sizeof(int)*MAX);//ATTENTION 1：终止条件不要设置 0
    printf("请指定计算多少位的阶乘：");
    scanf("%d",&N);
    result[0] = 1; //初始为 1
    for(i = 1 ; i <= N ; i++)
    {
       result = multiply(result,i);
    }
    printf("计算结果：");
    for(i = 0 ; result[i] != -1 ; i++)
        printf("%d",result[i]);
    return 0;
}
//大数 a 和非大数 b 相乘：b 乘 a 低，结果存低，其余进位
```

```
int* multiply(int a[],int b)
{
    int* result = (int*)malloc(sizeof(int)*MAX);
    int len;
    int i,j;
    int temp , carry = 0;
    memset(result,-1,sizeof(int)*MAX);
    for(len = 0 ; a[len] != -1 ; len++); //获取大数 a 长度
    for(i = len-1,j=0 ; i >= 0 ; i--,j++)
    {
        temp = a[i]*b + carry;
        result[j] = temp % 10; //保存最低位
        carry = temp / 10;
    }
    if(carry != 0)
        result[j] = carry; //保存最后一次相乘的进位
    else
        j = j-1; //再错 1：进位为 0 时，把 j 指向数组最后一个位置！
    //将结果反转：反转模板
    for(i = 0 ; i < j ; i++,j--)
    {
        temp = result[i];
        result[i] = result[j];
        result[j] = temp;
    }
    return result;
}
```

1109.输入正整数 n,按从小到大的顺序输出所有形如 abcde/fghij=n 的表达式，其中 a~j 恰好为数字 0~9 的一个排列，2<=n<=79。

输入：62　　输出：79546/01283=62

94736/01528=62

分析：

这个问题就是前面所说的[数 N 被分为 ab 之和、之积、之除，ab 还要满足一定条件]：

1.一般来说，取较大数视为 N 较好，取 abcde 视作 N，遍历 faghi 所有可能，且 abcde 通过 faghi *n 计算得出。（这里 **abcde** 既是被分解的数同时也是**未知数**）

2.验证 abcde 和 faghi 是否满足数不同条件，满足则输出。

```c
#include<stdio.h>

int diffrent(int a , int b);
int main()
{
    int n;
    int abcde,fghij;
    //double abcdefghij ; //ab 将数 a-数 b，合为一数判断数不同(错误 1：数太大)
    printf("请输入指定 n： ");
    scanf("%d",&n);
    //1.0 暴力循环 （98762-1234）* （98762-1234)执行不下去
    //2.0 给了指定 n，遍历 fghij++同时* n ,算出 abcde，abcde,fghij 不同则满足
    for(fghij = 1234 ; fghij*n <= 98765 ; fghij++)
    {
        abcde = fghij * n;
        if(diffrent(abcde,fghij))
            printf("%d / % d = %d \n" ,abcde,fghij,n);
    }
    return 0;
}

//[背]判断数字是否每位都不同：一位数组+循环获取低位操作模板
//建立一个 A[10],下标对应数字 0-9，值记录出现次数。
int diffrent(int abcde , int fghij)
{
    int A[10] = {0};
    int low ; //获取最低位
    //易错 2：是四位数前面 0 也要考虑不要重复
    if(abcde >= 1000 && abcde <= 9999) //是 4 位数
```

```
30.         A[0]++;
31.     if(fghij >= 1000 && fghij <= 9999) //是4位数
32.         A[0]++;
33.     while(abcde) //易错 3 ：while(!abcde) abcde为0才执行，错误!
34.     {
35.         low = abcde % 10;
36.         A[low]++; //标识此时n某位数字low，个数+1
37.         if(A[low] > 1)
38.             return 0; //存在某位数字存在个数>1
39.         abcde = abcde /10;
40.     }
41.     while(fghij)
42.     {
43.         low = fghij % 10;
44.         A[low]++;
45.         if(A[low] > 1)
46.             return 0;
47.         fghij = fghij /10;
48.     }
49.     return 1;
50. }
```

1134.很多字串，有些是对称的，有些是不对称的，请将那些对称的字串按从小到大的顺序输出。字串先以长度论大小，如果长度相同，再以 ASCII 码值为大小标准。

输入：123321　　输出：123321

123454321　　123454321

123

121212

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

int mystrcmp(char* s1,char* s2);
int isSym(char* s);
int main()
{
  char* s[10] ;//ATTENTION 0: 对比声明 char** s,还必指定 1.指针个数和 2.大小
  char* temp;
  int N =0 ; //记录字符串个数
  int i,j,k=0;
  printf("请输入多个字符串，输入 null 结束:\n");
  //ATTENTION 1 : 用户不知道输入多少个字符串
  while(1) //由于{}里面还要分配内存，不能 while 里赋值
  {
     //易错 1：前面是指定二维大小，这里还要分配内存 s[i]
      s[N] = (char*)malloc(sizeof(char)*MAX);
      scanf("%s",s[N]);
      if(strcmp(s[N],"null") == 0)
          break;//易错 2 ： 不能 s[i++] == "null"!
      N++;
  }
  //剔除非对称的字符串
  for(i = 0 ; i < N ; i++)
  {
      if(isSym(s[i]))
          s[k++] = s[i];
  }
  N = k;  //是回文串的字符串长度
  //[背]冒泡排序模板
  printf("排序后输出结果为: \n");
  for(i = 0 ; i < N ; i++) //不写 N-1，为了输出 s[N-1]
  {
```

```
        for(j = N-1 ;j > i;j--) //每一轮把交换左右把最小放前面
        {
            if(mystrcmp(s[j],s[j-1]) < 0)
            {
               temp = s[j];
               s[j] = s[j+1];
               s[j+1] = temp;
            }
        }
        printf("%s\n",s[i]); //输出
    }
    return 0;
}

//比较字符串大小
//ATTENTION 2 :字符数组指针可以获取长度，int 不行!
int mystrcmp(char* s1,char* s2)
{
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    if(len1 > len2) return 1;
    if(len1 < len2) return -1;
    if(len1 == len2) return strcmp(s1,s2);
}

//[背]判断是否回文：反转字符模板
int isSym(char* s)
{
    int len = strlen(s);
    int i = 0 ,j = len-1;
    while(i < j)
    {
        if(s[i++] != s[j--])
            return 0;
    }
    return 1;
}
```

2019　　谭浩强习题练习

1. 有一行电文译文下面规律译成密码：　A->Z a->z B->Y b->y C->X c->x　... 即第一个字母变成第 26 个字母,第 i 个字母变成第 26-i-1 个字母。例如有一行电文译文下面规律译成密码：

A->Z　B->Y　C->X　　a->z　b->y c->x　...

非字母字符不变,要求编程序将密码回原文,并打印出密码和原文.

分析：

1.和前面 s[i] = (s[i] - ‘a’ + k) %26 + ‘a’不同：（[注]这里 i 都是从 0 开始）

a　b　c .... z

ASCII

相对：0　1　2　　25

2.这两题的思路都是：s[i] = a + 相对 a 偏移 m

(1)相对偏移 m = s[i] - a + k , m 会超出 25 要取余

(2)相对偏移 m = 26-i-1, ∈(0,25),无需取余

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define MAX 100
char* code(char* s);
void main()
{
    char* s = (char*)malloc(sizeof(char)*MAX),* c;
    printf("请输入一段字符串：");
    gets(s);
    c = code(s);
    printf("加密后：%s\n",c);
    printf("解密后：%s\n",code(c));    //ATTENTION 1:加密解密一个函数
}

//加密字符:加密后 ASCII 无需取余计算得来
char* code(char* s)
{
    int i , len = strlen(s);
    for(i = 0 ; i < len ; i++)
    {
        if(islower(s[i])) //1.解密字符经过推理算出 s[i] = 下面表达式
            s[i] = 'z' - (s[i] - 'a') ;// s[i] = a + (25 - (s[i] -'a' ))
        else if(isupper(s[i]))
            s[i] = 'Z' - (s[i] - 'A') ;
        else   continue; //非字母不改变
    }
    return s;
}
```

2. 输出一串字符中最长的单词。如:

输入： i am huang wang hui 23 years old　　输出：huang

分析:

[模板]记录字符串中最长单词长度：if(是字符) len++　else **1.操作**　**2.**len =0

应用：此题还多了要输出最长单词，记录初始位置。**在 1.操作位置补上代码**

①遍历数组，如果遇到字母开始计算长度

②遇到空格/数字/...和 max 比较决定是否记住当前单词位置+长度置 0

③重复 1，直至到末尾。

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

int main()
{
  char s[100];
  int i;
  int pos = 0 , max = 0,len = 0;
  printf("请输入一串字符串：");
  gets(s);
  for(i = 0 ; i < strlen(s); i++)
  {
      if(isalpha(s[i]))
      {
          len++;
      }
      else
      {
          if(len > max) //当前记录长度大于 max
          {
              pos = i - len; //该单词首字母位置
              max = len;
          }
          len = 0;
      }
  }
  for(i = 0 ; i < max ; i++)
      printf("%c",s[i+pos]);
  return 0;
}
```

3. 十六进制转为十进制。

输入：ABCD　　输出：43981

```c
#include<stdio.h>

int main()
{
    __int64 n16_10;
    int n10_16;

      //16转10(1)
    printf("请输入一个十六进制数(1)：");
    scanf("%X",&n16_10);
    printf("对应十进制数：%d\n",n16_10); //易错1：lld

      //10转16(1)
    printf("请输入一个十进制数：");
    scanf("%d",&n10_16);
    printf("对应十六进制数：%X",n10_16);
    return 0;
}
```

4.十进制转为十六进制。

输入：43981　　输出：ABCD

分析：除 16 取余，判商赋商，逆序排列

和二进制类似：(1) n % 16 = 商...余

(2) n = 商， 记录此时余数

(3) 循环(1)、(2)直至商为 0

(4) 此时逆序，在逆向反置，使得正序

```c
#include<stdio.h>

char* dectohex(int num);
int main()
{
    int num;
    printf("请输入一个十进制数：");
    scanf("%d",&num);
    printf("%s",dectohex(num));
    return 0;
}

char* dectohex(int num)
{
    char* hex = (char*)malloc(sizeof(char)*100);
    char temp;
    int q; //商
    int r; //余数
    int i = 0 , j = 0;
    while(q != 0)
    {
        q = num / 16;
        r = num % 16;
       //巧：大于 10 char 类型也存不了，刚好存 A..F
        if(r >= 10) hex[i++] = r - 10 + 'A';
        else        hex[i++] = r;
        num = q;
    }
    hex[i] = '\0';   //易错!
    //[模板]反置：交换使得 hex 逆向，从而正序
    for(i = i-1,j =0 ;j < i;i--,j++)//为方便 i 指向最后一个字符，而非'\0'
    {
      temp = hex[i];     hex[i] = hex[j];      hex[j] = temp;
    }
    return hex;
}
```

5.将一个 5x5 的矩阵中最大的元素放在中心，4 个角分别放 4 个最小的元素(顺序为从左到右，从上到下依次从小到大存放)。

分析：

①写出冒泡排序模板：

for(i =0 ; i<n-1;i++)

for(j = 0 ; j <n-1-i;j++)

②对于上述：n 等于 25，访问二维数组 A[I][J]元素： ***(A[I]+J)**(**行指针必须要获得行地址**) --> 线性访问二维数组第 n 个元素： **I = n /5 ， J = n%5**

```
#include<stdio.h>
#define n 5
void sort(int A[][n]); //ATTENTION 0：A[][]非法
void swap(int A[][n],int i,int j,int i1,int j1);
void main()
{
 int A[n][n]={{7,1,5,3,2},{5,4,8,0,9},{1,2,3,4,5},{6,7,8,9,10},{1,7,9,3,5};
    sort(A);
    swap(A,n/2,n/2,n-1,n-1); //中心
    swap(A,0,n-1,0,1); //右上角
    swap(A,n-1,0,0,2); //左下角
    swap(A,n-1,n-1,0,3); //右下角
}
/*
  线性二维数组排序：冒泡模板
  和字符串排序不同： 1.字符串依然相当是一维数组比较
                     2.交换的是地址（指针），这里交换元素。
*/
void sort(int A[][n])
{
    int i ,j,N = n*n,temp ;
    for(i = 0 ; i < N ; i++) //线性访问二维数组
        for(j = 0 ; j < N-1-i ; j++)
        {
           if(A[j/n][j%n]> A[(j+1)/n][(j+1)%n]) //这里交换不了数组地址
               swap(A,j/n,j%n,(j+1)/n,(j+1)%n);
        }
}
void swap(int A[][n],int i,int j,int i1,int j1) //易错 1：注意交会二维数组参数
{
    int temp;
    temp = A[i][j];   A[i][j] = A[i1][j1];   A[i1][j1] = temp;
}
```

6.实现计算 sinx，cosx，e^x ,[a,b]积分的通用函数。

分析：

1. 积分计算 a-------b ： ∑ f(x) * △x：

   △x = (b-a)/n ； x = a + △x * i;

2. 唯一不同：f(x)，可考虑把它当函数指针传入即可。

```c
#include<stdio.h>
#include<math.h>
#define n 1000

//double _sin(double x); //库中有定义，无需我来写
double area (double (*f)(double) , double a ,double b);
int main()
{
    double area_sin ,area_cos ,area_exp;
    area_sin = area(sin,0,1);  //易错 1：传函数指针 sin 别带()!
    area_cos = area(cos,0,1);
    area_exp = area(exp,0,1);
        printf("sin[0,1] 积 分 ： %lf,cos[0,1] 积 分 ： %lf,exp[0,1] 积分：%lf",area_sin,area_cos,area_exp);
    return 0;
}

//通用计算函数积分：函数指针的使用
double area (double (*f)(double) , double a ,double b)
{
    double det = (b -a) / n ; //det 别写 1/n !
    double area = 0;
    int i;
    for(i = 1 ; i <= n ; i++)
    {
        area += (*f)(a+det * i) * det; //易错 2：f 不是函数名，*f 才是!
    }
    return area;
}

```

7.输入一个字符串，内有数字和非数字字符。将其中连续的数字作为一个整数，依次存放到一数组 a 中。例如，123 放在 a[0]，456 放在 a[1]..统计共有多少个整数，并输出这些数：

输入：A123x456 17960? 302tab5876　　输出：123 456 17960 302 5876

分析：

1.这也是：[模板]一串字符求最长单词长度/输出最长单词/输出数字/...

(1)**if(是数字)**　// 操作 1: num = num*10 + s[i] - '.0'

[再错]：最后一个数字 5876，读到 6 结束循环，不会执行(2)被存!

(2)**else** // 操作 2: 存入上一个 num（flag 判断上一次结束的是数字），num = 0。

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

int main()
{
    char s[MAX];
    int nums[MAX] , num = 0;
    int flag = 1;  //易错 0:初始为 1，防止读入非数字进 nums
    int i , j = 0 , len = strlen(s) ;
    printf("请输入一串字符：");
    gets(s);
    for(i = 0 ; i < len ; i++)
    {
       if(isdigit(s[i]))
       {
           flag = 0 ; //标识上次结束的是一串数字而非其它
           num = num * 10 + s[i] - '0';
           //再错 1：此时已经读到了字符末尾，不会执行 else 存 [又错了]
           if(i == len-1)  {  nums[j] = num;  printf("%d\t",nums[j++]);
       }
       else
       {
           //易错 2：上次结束的是数字&&这次读入非数字才存下来
           if(flag == 0)  {  nums[j] = num;  printf("%d\t",nums[j++]);
           flag = 1;
           num = 0;
       }
    }
    return 0;
}
```

8.写一个 strcmp 函数，实现比较字符 ASCII 值，返回比较结果。

输入： BOY BAD　　输出： 1

分析：

1.遍历两者字符串，直至某一字符串遍历完，逐个比较字符 ASCII 码值返回结果。

2.如果遍历完还未比较出，更长的那个更大。

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

int mystrcmp(char* s1,char* s2);
int main()
{
    char s1[100],s2[100];
    printf("请输入两个字符串，中间逗号隔开：\n");
    scanf("%s %s",&s1,&s2);
    printf("结果：%d",mystrcmp(s1,s2));
    return 0;
}

//实现 strcmp
int mystrcmp(char* s1,char* s2)
{
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    int lm = len1 < len2 ? len1:len2;
    int i;
    for(i = 0 ; i < lm ; i++)
    {
        if(s1[i] - s2[i] > 0)
            return 1;
        else if(s1[i] - s2[i] < 0)
            return -1;
        else //相等
            continue;
    }
    //执行到这还没分出来
    if(len1 == len2) return 0;
    return len1 > len2 ? 1:0; //长度大的大
}
```

9. 写一个结构体变量，含有年、月、日，并用 days 函数实现计算某日是当年第多少天？注意闰年问题。

分析：

1.注意闰年有两种情况：(1)能被 4 整除不能被 100 整除

(2)能被 400 整除

```c
include<stdio.h>

struct Date
{
    int year;
    int month;
    int day;
}date; // 易错:这里的 date 是变量，直接用！如果加了 typedef 是类型！

int days(int year,int month,int day);
int main()
{
  int day;
  printf("请输入年 月 日，以空格分隔: ");
  scanf("%d %d %d",&date.year,&date.month,&date.day);
  day = days(date.year,date.month,date.day);
  printf("该天是 %d 年第 %d 天", date.year,day);
  return 0;
}

//计算第几天
int days(int year,int month,int day)
{
   int a[12]={31,28,31,30,31,30,31,31,30,31,30,31};
   int sum,i;
   sum=day;
   for(i=0;i<month-1;i++)
         sum+=a[i];
  //闰年: 1.能被 4 整除但不能被 100 整除 2.能被 400 整除
  if(((year%4==0 && year%100!=0)|| year%400==0) && month>2)
      sum = sum + 1; //加 1 天
  return sum;
}
```

10. 13 个人围成一圈报数，凡是报到为 3 倍数的都要退出圈子。用链表实现，有 del 函数。

分析：

1.参照 C++真题链表的实现！