2011年计算机二级C语言编写程序题及答案解析精选

【4.1】已知银行整存整取存款不同期限的月息利率分别为：

0.315% 期限一年

0.330% 期限二年

月息利率 ＝ 0.345% 期限三年

0.375% 期限五年

0.420% 期限八年

要求输入存钱的本金和期限，求到期时能从银行得到的利息与本金的合计。

【4.2】输入年份 year 和月 month，求该月有多少天。判断是否为闰年，可用如下C语言表达式：year%4==0 && year0!=0 || year@0==0。若表达式成立（即表达式值为1），则year为闰年；否则，表达式不成立（即值为0），year为平年。

【4.3】编写一个简单计算器程序，输入格式为：data1 op data2。其中data1和data2是参加运算的两个数，op为运算符，它的取值只能是+、-、*、/。

【4.4】输入n值，输出如图所示矩形。

```
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

图4.1 n=6时的矩形

【4.5】输入n值，输出如图所示平行四边形。

```
      * * * * * *
    * * * * * *
  * * * * * *
* * * * * *
* * * * * *
```

图4.2 n=6时的平行四边形

【4.6】输入n值，输出如图所示高为n的等腰三角形。

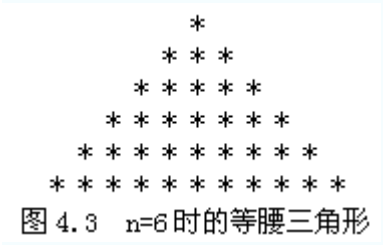

图 4.3　n=6 时的等腰三角形

【4.7】输入 n 值，输出如图所示高为 n 的等腰三角形。

图 4.4　n=6 时的倒等腰三角形

【4.8】输入 n 值，输出如图所示高和上底均为 n 的等腰梯形。

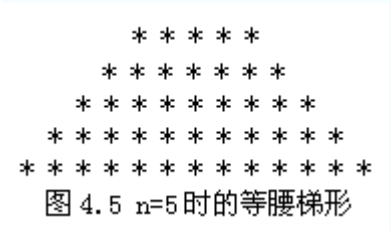

图 4.5 n=5 时的等腰梯形

【4.9】输入 n 值，输出如图所示高和上底均为 n 的等腰空心梯形。

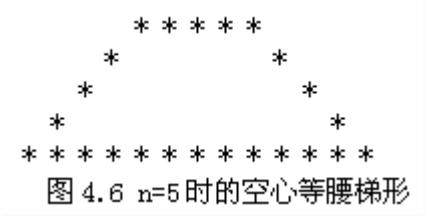

图 4.6 n=5 时的空心等腰梯形

【4.10】输入 n 值，输出如图所示边长为 n 的空心正六边型。

图 4.7 n=5 时的空心正六边型

【4.11】输入 n 值，输出如图所示图形。

```
*       *
 *     *
  *   *
   * *
    *
   * *
  *   *
 *     *
*       *
```

图 4.8 n=5 时的 X 形

【4.12】输入 n 值，输出如图所示图形。

```
* * * * *
      *
     *
    *
* * * * *
```

图 4.9 n=5 时的 Z 形

【4.13】输入 n 值，输出如图所示图形。

```
*   *
* *
*
* *
*   *
```

图 4.10 n=3 时的 K 形

【4.14】输入 n 值，输出如图所示图形。

```
*     *
* *   *
*  *  *
*   * *
*     *
```

图 4.12 n=5 时的 N 形

【4.15】输入 n 值，输出如图所示图形。

图 4.12 n=3时的菱形

【4.16】输入 n 值，输出如图所示图形。（例为 n=6时）

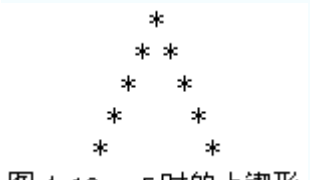

图 4.13 n=5时的上锲形

【4.17】编写程序，输出如图所示 sin(x) 函数0到2π的图形。

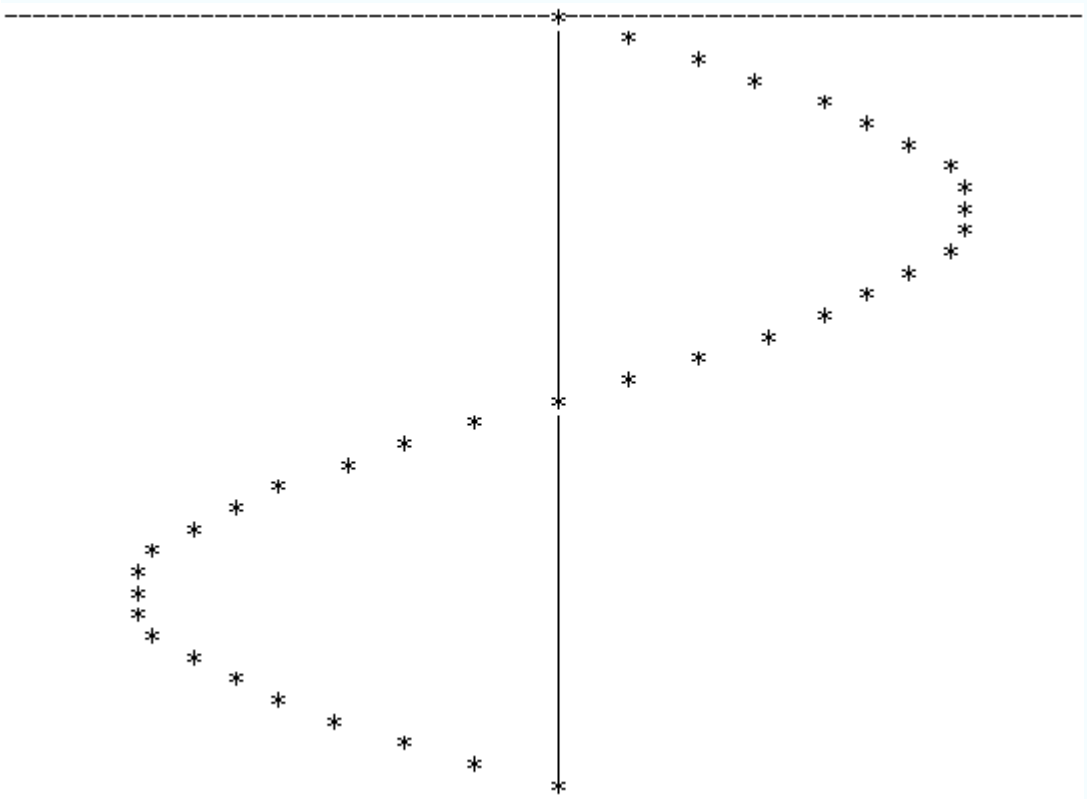

图 4.14　正弦曲线

【4.18】编写程序，在屏幕上输出一个由*号围成的空心圆。

【4.19】编写程序，在屏幕上绘制如图余弦曲线和直线。若屏幕的横向为 x 轴，纵向为 y 轴，　在屏幕上显示0～360度的 cos(x)曲线与直线 x=f(y)=45*(y-1)+31的迭加图形。其中 cos 图形用"*"表示，f(y)用"+"表示，在两个图形的交点处则用 f(y)图形的符号。

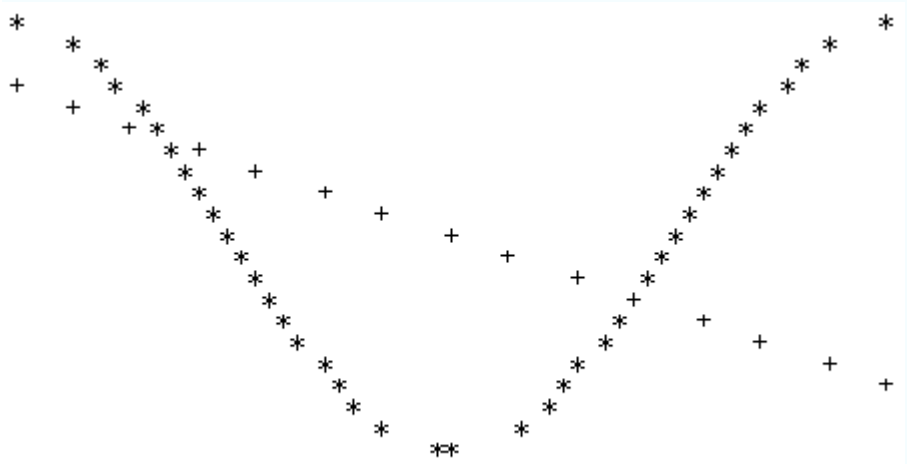

图 4.15　余弦曲线和直线

【4.20】编写程序，输出如图所示高度为 n 的图形。

```
 1   2   3   4   5   6
 7   8   9  10  11  12
13  14  15  16  17  18
19  20  21  22  23  24
25  26  27  28  29  30
31  32  33  34  35  36
```

图 4.16　n=6 时的数字正方形

【4.21】编写程序，输出如图所示高度为 n 的图形。

```
 1   3   6  10  15  21
 2   5   9  14  20
 4   8  13  19
 7  12  18
11  17
16
```

图 4.17 n=6 时的数字倒三角

【4.22】输入 n 值，输出如图所示图形。

```
1  2  3  4  5
1  1  2  3  4
1  1  1  2  3
1  1  1  1  2
1  1  1  1  1
```

图 4.18 n=5 时的数字矩形

【4.23】输入 n 值，输出如图所示的 n×n(n<10)阶螺旋方阵。

```
1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9
```

图 4.19 n=5 时的螺旋方阵

【4.24】输入 n 值，输出如图所示回型方阵。

```
当 n=5 时：
3  2  3  2  3
3  2  2  2  3
3  2  1  2  3
3  3  2  3  3
3  3  3  3  3

当 n=6 时：
3  3  3  3  3  3
3  2  2  2  2  3
3  2  1  1  2  3
3  2  1  1  2  3
3  2  2  2  2  3
3  3  3  3  3  3
```

图 4.20 回形方阵

【4.25】输出如图所示的数字金字塔

```
        1
      1 2 1
    1 2 3 2 1
  1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
  1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
      1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
        1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
```

图 4.21 n=5 时的数字金字塔

【4.26】输入 n 值，输出如图所示图形。

```
      Z
    Y   X
  W       V
U           T
S             R
  R           Q
    P       O
      N   M
        L
```

图 4.22 n=5 时的空心菱形

【4.27】输入顶行字符和图形的高，输出如图所示图形。

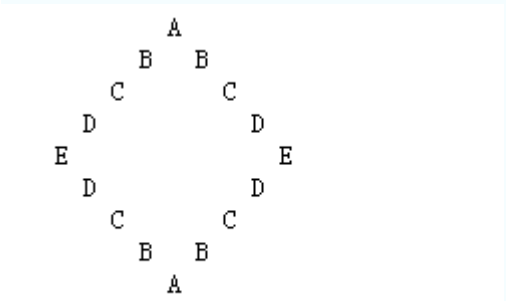

图 4.23 顶行字符为'A'、高为 5 的菱形

【4.28】输入首字符和高后，输出如图所示回型方阵。

A A A A A

A B B B A

A B C B A

A B B B A

A A A A A

图 4.24 首字符为'A'、高为 5 的方阵

【4.29】输入中心字符和高后，输出如图所示回型方阵。

X X X X X

X Y Y Y X

X Y Z Y X

X Y Y Y Y

X X X X X

图 4.25 中心字符为'Z'、高为 5 的方阵

【4.30】编写程序，输出如图所示上三角形式的乘法九九表。

```
 1  2  3  4  5  6  7  8  9
--------------------------
 1  2  3  4  5  6  7  8  9
    4  6  8 10 12 14 16 18
       9 12 15 18 21 24 27
         16 20 24 28 32 36
            25 30 35 40 45
               36 42 48 54
                  49 56 63
                     64 72
                        81
```

图 4.26　上三角乘法九九表

【4.31】编写程序，输出如图所示下三角乘法九九表。

```
 1  2  3  4  5  6  7  8  9
--------------------------
                        81
                     64 72
                  49 56 63
               36 42 48 54
            25 30 35 40 45
         16 20 24 28 32 36
      9 12 15 18 21 24 27
   4  6  8 10 12 14 16 18
1  2  3  4  5  6  7  8  9
```

图 4.27 下三角乘法九九表

【4.32】编写程序，输入三角型的三条边长，求其面积。注意：对于不合理的边长输入要输出数据错误的提示信息。

【4.33】编写程序求出555555的约数中最大的三位数是多少。

【4.34】编写程序计算下列算式的值：

$$C = 1 + \frac{1}{x^1} + \frac{1}{x^2} + \frac{1}{x^3} + \frac{1}{x^4} \cdots\cdots \quad (x>1)$$

直到某一项A<=0.000001时为止。输出最后C的值。

【4.35】从键盘输入任意的字符，按下列规则进行分类计数。

第一类 '0'，'1'，'2'，'3'，'4'，'5'，'6'，'7'，'8'，'9'

第二类 '+'，'-'，'*'，'/'，'%'，'='

第三类 其它字符

当输入字符'\'时先计数，然后停止接收输入，打印计数的结果。

【4.36】对从键盘上输入的行、单词和字符进行计数。我们将单词的定义进行化简，认为单词是不包含空格、制表符(\t)及换行符的字符序列。例如："a+b+c"，认为是1个单词，它由5个字符组成。又如："xy abc"，为2个单词，6个字符。一般用[CTRL+D]作为文件结束标记，其字符码值为-1，当输入[CTRL+D]时表示文件输入结束，停止计数。

【4.37】编写程序计算当x=0.5时下述级数和的近似值，使其误差小于某一指定的值epsilon（例如：epsilon=0.000001）：

$$x - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \cdots\cdots$$

【4.38】编写程序计算下式的值：

$$\sum_{k=1}^{100} k + \sum_{k=1}^{50} k*k + \sum_{k=1}^{10} \frac{1}{k}$$

【4.39】编写程序计算下列序列的值：

$$1+\frac{1}{1\times 2}+\frac{1}{2\times 3}+\frac{1}{3\times 4}+\frac{1}{4\times 5}+\cdots\cdots+\frac{1}{N\times (N+1)}$$

要求最后一项小于0.001时、或者当N=20时尚未达到精度要求，则停止计算。

【4.40】已知求正弦sin(x)的近似值的多项式公式为：

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots\cdots + (-1)^n\frac{x^{2n+1}}{(2n+1)!} + \cdots$$

编写程序，要求输入x和ε，按上述公式计算sin(x)的近似值，要求计算的误差小于给定的ε。

【4.41】从键盘输入十个整数，用插入法对输入的数据按照从小到大的顺序进行排序，将排序后的结果输出。

【4.42】输入一个正整数，要求以相反的顺序输出该数。例如输入12345，输出位54321。

【4.43】编写程序，读入一个整数Ｎ；若Ｎ为非负数，则计算Ｎ到2×Ｎ之间的整数和；若Ｎ为一个负数，则求２×Ｎ到Ｎ之间的整数和。分别利用 for 和 while 写出两个程序。

【4.44】求解爱因斯坦数学题。有一条长阶梯，若每步跨2阶，则最后剩余1阶，若每步跨3阶，则最后剩2阶，若每步跨5阶，则最后剩4阶，若每步跨6阶则最后剩5阶，若每步跨7阶，最后才正好一阶不剩。请问，这条阶梯共有多少阶?

【4.45】一个自然数被8除余1，所得的商被8除也余1，再将第二次的商被8除后余7，最后得到一个商为 a 。又知这个自然数被17除余4， 所得的商被17除余15，最后得到一个商是 a 的2倍。编写程序求这个自然数。

【4.46】编写程序，用二分法求一元二次方程2x3-4x2+3x-6=0在（10，10）区间的根。

【4.47】中国古代科学家祖冲之采用正多边形逼近的割圆法求出了π的值。请编写一程序，采用割圆法求出π的值，要求精确到小数点之后的第十位。

【4.48】A、B、C、D、E 五人在某天夜里合伙去捕鱼，到第二天凌晨时都疲惫不堪，于是各自找地方睡觉。日上三竿，A 第一个醒来，他将鱼分为五份，把多余的一条鱼扔掉，拿走自己的一份。B 第二个醒来，也将鱼分为五份，把多余的一条鱼扔掉，拿走自己的一份。C、D、E 依次醒来，也按同样的方法拿鱼。编写程序求出他们合伙至少捕了多少条鱼。

【4.49】一辆卡车违犯交通规则，撞人逃跑。现场三人目击事件，但都没记住车号，只记下车号的一些特征。甲说：牌照的前两位数字是相同的；乙说：牌照的后两位数字是相同的；丙是位数学家，他说：四位的车号刚好是一个整数的平方。请根据以上线索求出车号。

【4.50】若一个口袋中放有12个球，其中有3个红的，3个白的和6个黑的，每次从中任取8个球，编写程序求出共有多少种不同的颜色搭配。

【4.51】100匹马驮100担货，大马一匹驮 3 担，中马一匹驮 2 担，小马两匹驮 1 担。试编写程序计算大、中、小马的数目。

【4.52】编写程序，输出用一元人民币兑换成1分、2分和5分硬币的不同兑换方法。

【4.53】显示200以内的完全平方数和它们的个数。（完全平方数：A2+B2=C2,求 A、B、C）

【4.54】设 N 是一个四位数，它的9倍恰好是其反序数（例如：123的反序数是321），求 N 的值。

【4.55】将一个数的数码倒过来所得到的新数叫原数的反序数。如果一个数等于它的反序数，则称它为对称数。求不超过1993的最大的二进制的对称数。

【4.56】编写程序求解下式中各字母所代表的数字。

【4.57】一个自然数的七进制表达式是一个三位数，而这个自然数的九进制表示也是一个三位数，且这两个三位数的数码顺序正好相反，求这个三位数。

【4.58】请验证2000以内的哥德巴赫猜想，对于任何大于4的偶数均可以分解为两个素数之和。

【4.59】如果一个正整数等于其各个数字的立方和，则称该数为阿姆斯特朗数（亦称为自恋性数）。如407=43+03+73就是一个阿姆斯特朗数。编写程序求1000以内的所有阿姆斯特朗数。

【4.60】任意输入一个偶数，请将它分解为两个素数之和。

【4.61】如果整数A的全部因子（包括1，不包括A本身）之和等于B；且整数B的全部因子(包括1，不包括B本身)之和等于A，则将整数A和B称为亲密数。求3000以内的全部亲密数。

【4.62】猜数游戏。由计算机"想"一个数请人猜，如果人猜对了，则结束游戏，否则计算机给出提示，告诉人所猜的数是太大还是太小，直到人猜对为止。计算机记录人猜的次数，以此可以反映出猜数者"猜"的水平。

【4.63】编写程序求出1000!后有多少个零。

【4.64】求矩阵 A[2*3] 的转置矩阵 B[3*2]。设矩阵 A 为：

【4.65】十个小孩围成一圈分糖果，老师分给第一个小孩10块，第二个小孩2块，第三个小孩8块，第四个小孩22块，第五个小孩16块，第六个小孩4块，第七个小孩10块，第八个小孩6块，第九个小孩14 块，第十个小孩20块。然后所有的小孩同时将自己手中的糖分一半给右边的小孩；糖块数为奇数的人可向老师要一块。问经过这样几次调整后大家手中的糖的块数都一样？每人各有多少块糖？

【4.66】输入5×5的数组，编写程序实现：

(1)求出对角线上各元素的和；

(2)求出对角线上行、列下标均为偶数的各元素的积；

(3)找出对角线上其值最大的元素和它在数组中的位置。

【4.67】编写程序，以字符形式输入一个十六进制数，将其变换为一个十进制整数后输出。

【4.68】编写程序，输入一个十进制整数，将其变换为二进制后储存在一个字符数组中。

【4.69】编写程序，输出1000以内的所有完数及其因子。所谓完数是指一个整数的值等于它的因子之和，例如6的因子是1、2、3，而6=1+2+3，故6是一个完数。

【4.70】对数组A中的N（0＜N＜100＝个整数从小到大进行连续编号，输出各个元素的编号。要求不能改变数组A中元素的顺序，且相同的整数要具有相同的编号。例如数组是：A=(5,3,4,7,3,5,6) 则输出为：　(3,1,2,5,1,3,4)

【4.71】现将不超过2000的所有素数从小到大排成第一行，第二行上的每个数都等于它"右肩"上的素数与"左肩"上的素数之差。请编程求出：第二行数中是否存在这样的若干个连续的整数，它们的和恰好是1898？假如存在的话，又有几种这样的情况？

第一行：2 3 5 7 11 13 17 ..... 1979 1987 1993

第二行： 1 2 2 4 2 4 ..... 8 6

【4.72】将1、2、3、4、5、6、7、8、9九个数字分成三组，每个数字只能用一次，即每组三个数不许有重复数字，也不许同其它组的三个数字重复，要求将每组中的三位数组成一个完全平方数。

【4.73】一个自然数的七进制表达式是一个三位数，而这个自然数的九进制表示也是一个三位数，且这两个三位数的数码顺序正好相反，求这个三位数。

【4.74】使用数组精确计算M/N(0<=100)的各小数位的值。如果M N是无限循环小数，则计算并输出它的第一循环节，同时要求输出循环节的起止位置（小数的序号）。

为了实现高精度计算结果，可将商M存放在有N（N＞1）个元素的一维数组中，数组的每个元素存放一位十进制数，即商的第一位存放在第一个元素中，商的第二位存放在第二个元素中……，依次类推。这样可使用数组来表示计算的结果。

【4.75】使用数组完成两个超长（长度小于1 0 0）正整数的加法。

为了实现高精度的加法，可将正整数M存放在有N（N＞1）个元素的一维数组中，数组的每个元素存放一位十进制数，即个位存放在第一个元素中，十位存放在第二个元素中……，依次类推。这样通过对数组中每个元素的按位加法就可实现对超长正整数的加法。

【4.76】使用数组完成两个超长（长度小于1 0 0）正整数的加法。

为了实现高精度的加法，可将正整数M存放在有N（N＞1）个元素的一维数组中，数组的每个元素存放一位十进制数，即个位存放在第一个元素中，十位存放在第二个元素中……，依次类推。这样通过对数组中每个元素的按位加法就可实现对超长正整数的加法。

【4.77】使用数组完成两个超长（长度小于1 0 0）正整数的乘法。

【4.78】马步遍历问题：已知国际象棋棋盘有8*8共64个格子。设计一个程序，使棋子从某位置开始跳马，能够把棋盘上的格子走遍。每个格子只允许走一次。

【4.79】八皇后问题：

在一个 8 × 8 的国际象棋盘，有八个皇后，每个皇后占一格；要求棋盘上放上八个皇后时不会出现相互"攻击"的现象，即不能有量个皇后在同一行、列或对角线上。问共有多少种不同的方法。

【4.80】编制一个计算函数 y=f(x)的值程序，其中：

-x + 2.5 0<= x <2

y= 2 - 1.5(x-3)*(x-3) 2<= x <4

x/2 - 1.5 4<= x <6

【4.81】编写程序，实现比较两个分数的大小。

【4.82】求这样一个三位数，该三位数等于其每位数字的阶乘之和。

即： abc = a! + b! + c!

【4.83】已知两个平方三位数 abc 和 xyz，其中数码 a、b、c、x、y、z 未必是不同的；而 ax、by、cz 是三个平方二位数。编写程序，求三位数 abc 和 xyz。任取两个平方三位数 n 和 n1，将 n 从高向低分解为 a、b、c，将 n1从高到低分解为 x、y、z。判断 ax、by、cz 是否均为完全平方数。

【4.84】找出一个二维数组中的鞍点，即该位置上的元素是该行上的最大值，是该列上的最小值。二维数组也可能没有鞍点。

【4.85】将数字1、2、3、4、5、 6 填入一个2行3列的表格中，要使得每一列右边的数字比左边的数字大，每一行下面的数字比上面的数字大。编写程序求出按此要求可有几种填写方法？

【4.86】编写一个函数实现将字符串 str1和字符串 str2合并，合并后的字符串按其 ASCII 码值从小到大进行排序，相同的字符在新字符串中只出现一次。

【4.87】已知计算 x 的 n 阶勒让德多项式值的公式如下：

1 (n=0)

Pn(x) = x (n=1)

((2n-1)*x*Pn-1(x)-(n-1)*Pn-2(x))/n (n>1)

请编写递归程序实现。

【4.88】编写函数，采用递归方法实现将输入的字符串按反序输出。

【4.89】编写函数，采用递归方法在屏幕上显示如下杨辉三角形：

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
…… …… …… …… ……
```

【4.90】编写函数，采用递归方法将任一整数转换为二进制形式。

【4.91】设有字母 a、b、c，请编程用递归的方法产生由这些字母组成的，且长度为 n 的所有可能的字符串。例如，输入 n=2，则输出：

aa ab ac ba bb bc ca cb cc

【4.92】将一个数的数码倒过来所得到的新数，叫作原数的反序数，如果一个数等于它的反序数，则称它为对称数。编写程序，采用递归算法求不超过1993的最大的二进制的对称数。

【4.93】从 1 到 n(n<1000)个自然数中选出 r 个数进行组合，并按指定的格式输出组合的结果。例如：n=5，r=3时，共有10种组合，运行程序，要按下面的格式输出：

```
 1 2 3
     4
     5
   3 4
     5
   4 5
 2 3 4
     5
   4 5
 3 4 5
```

请用递归算法实现。

【4.94】从键盘输入十个整数，用合并排序法对输入的数据按照从小到大的顺序进行排序，将排序后的结果输出。

【4.95】编写程序，读入一个以符号"."结束的长度小于20字节的英文句子，检查其是否为回文（即正读和反读都是一样的，不考虑空格和标点符号）。例如：

读入句子：MADAM I'M ADAM. 它是回文，所以输出：YES

读入句子：ABCDBA). 它不是回文，所以输出：NO

【4.96】编写程序，其中包括一个函数，此函数的功能是：对一个长度为N 的字符串从其第K 个字符起，删去M 个字符，组成长度为N-M 的新字符串(其中N、M<=80,K<=N)。例如输入字符串"We are poor students."，利用此函数进行删除"poor"的处理，输出处理后的字符串是"We are students."。

【4.97】编写函数，通过指针将一个字符串反向。

【4.98】编写一个函数 insert(s1,s2,ch)，实现在字符串 s1中的指定字符 ch 位置处插入字符串 s2。

【4.99】编写程序将输入的两行字符串连接后，将串中全部空格移到串首后输出。

【4.100】编写程序，输入字符串，分别统计字符串中所包含的各个不同的字符及其各自字符的数量。如：输入字符串： abcedabcdcd

则输出：a=2 b=2 c=3 d=3 e=1。

【4.101】利用结构：struct complx

{ int real;

int im;

};

编写求两个复数之积的函数 cmult，并利用该函数求下列复数之积：

⑴ (3+4i)×(5+6i) ⑵ (10+20i)×(30+40i

【4.102】编写成绩排序程序。按学生的序号输入学生的成绩，按照分数由高到低的顺序输出学生的名次、该名次的分数、相同名次的人数和学号；同名次的学号输出在同一行中，一行最多输出10个学号。

【4.103】编写程序，实现输入的时间屏幕显示一秒后的时间。显示格式为 HH:MM:SS。程序需要处理以下三种特殊情况：

⑴ 若秒数加1后为60，则秒数恢复到0，分钟数增加1；

⑵ 若分钟数加1后为60，则分钟数恢复到0，小时数增加1；

⑶ 若小时数加1后为24，则小时数恢复到0。

【4.104】编写程序，从键盘输入3个学生的数据，将它们存入文件 student；然后再从文件中读出数据，显示在屏幕上。

【4.105】编写程序，从键盘输入一行字符串，将其中的小写字母全部转换成大写字母，然后输出到一个磁盘文件"test"中保存。

【4.106】编写程序，读入磁盘上C语言源程序文件"test8.c"，删去程序中的注释后显示。

【4.1】参考答案：

#include main( )

{ int year;

float money,rate,total; /* money:本金 rate:月利率 total:本利合计*/

```
printf("Input money and year =?");
scanf("%f%d", &money, &year);  /* 输入本金和存款年限 */
if(year==1) rate=0.00315;  /* 根据年限确定利率 */
else if(year==2) rate=0.00330;
else if(year==3) rate=0.00345;
else if(year==5) rate=0.00375;
else if(year==8) rate=0.00420;
else rate=0.0;
total=money + money * rate * 12 * year;  /* 计算到期的本利合计 */
printf(" Total = %.2f\n", total);
}
```

【4.2】参考答案：

```
#include main( )
{ int year, month, days;
printf("Enter year and month:");
scanf("%d%d", &year, &month);
switch (month)
{ case 1: case 3: case 5: case 7: case 8: case 10: case 12:
days=31;  break;  /* 处理"大"月 */
case 4: case 6: case 9: case 11:
days=30;  break;  /* 处理"小"月 */
case 2: if(year%4==0&&year0!=0 || year@0==0)
```

```
days=29;  /* 处理闰年平月 */
else days=28;  /* 处理不是闰年平月 */
break;
default: printf("Input error!\n");  /* 月份错误 */
days=0;
}
if( days!=0 )
printf("%d, %d is %d days\n", year, month, days);
}
```

【4.3】参考答案：

```
#include main ( )
{ float data1, data2;  /* 定义两个操作数变量 */
char op;  /* 操作符 */
printf("Enter your expression:");
scanf("%f%c%f", &data1, &op, &data2);  /* 输入表达式 */
switch(op) /* 根据操作符分别进行处理 */
{ case '+' : /* 处理加法 */
printf("%.2f+%.2f=%.2f\n", data1, data2, data1+data2);  break;
case '-' : /* 处理减法 */
printf("%.2f-%.2f=%.2f\n", data1, data2, data1-data2);  break;
case '*' : /* 处理乘法 */
printf("%.2f*%.2f=%.2f\n", data1, data2, data1*data2);  break;
```

```
case '/' : /* 处理除法 */

if( data2==0 ) /* 若除数为0 */

printf("Division by zero.\n");

else

printf("%.2f/%.2f=%.2f\n", data1, data2, data1/data2);

break;

default: /* 输入了其它运算符 */

printf("Unknown operater.\n");

}

}
```

【4.4】分析：打印此图形用两重循环实现。

图形要重复n行，故采用循环结构实现循环n次，循环体内部打印一行'*'号,把上述思路表示为：

```
for(i=1; i<=n; i++)

打印一行'*'号;
```

每行有n个'*'号，再采用循环结构实现n次循环，循环内部用格式输出语句打印一个'*'号，即：

```
for(j=1; j<=n; j++)

printf("*");
```

按照上述思路，实现打印矩形。

参考答案：

```
main()

{ int i,j,n;
```

```
printf("\nPlease Enter n:");

scanf("%d",&n);

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

{ for(j=1; j<=n; j++)

printf("*");

printf("\n");

}

}
```

【4.5】分析：此图形和上题的区别在于在每一行先要打印空格，然后再打印 n 个'*'号,在上题第一层循环体内打印'*'号的循环前面增加一个循环打印空格。每行空格的个数是逐行减少的，由于第一层循环的控制变量 i 是逐行增1，所以用一个固定值的数减去 i 就可实现对空格个数的控制，在此题中固定值可使用变量 n。

参考答案：

```
main( )

{ int i,j,n;

printf("\nPlease Enter n:");

scanf("%d",&n);

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

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

printf(" ");

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

printf("*");

printf("\n");
```

```
}

}
```

【4.6】分析：此题和上题的区别在于每行'*'的数量逐行减少，可以使用上题控制空格个数的思路来控制'*'号的个数，请注意每行'*'的个数都是奇数。

参考答案：

```
main( )

{ int i,j,n;

printf("\nPlease Enter n:");

scanf("%d",&n);

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

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

printf(" ");

for(j=1; j<=2*i-1; j++)

printf("*");

printf("\n");

}

}
```

【4.7】分析：此题图形是第3题图形的垂直反转，在编程上我们可以变换一个思路。对于图形中的第 i 行（1≤i≤n），共需要输出2n-i 个字符，其中前面的 i-1个字符为空格，后面的字符为'*'号。按照这一思路可以编写出如下程序。

参考答案：

```
main( )

{ int i,j,n;
```

```
printf("\nPlease Enter n:");
scanf("%d", &n);
for( i=1; i<=n; i++ ) /* 重复输出图形的n行 */
{ for( j=1; j<=2*n-i; j++ ) /* 重复输出图形一行中的每个字符 */
if(j<=i-1) printf(" "); /* 输出前面的空格 */
else printf("*"); /* 输出后面的*号 */
printf("\n");
}
}
```

【4.8】分析：此题和第3题的区别仅是每行的'*'个数增加n-1个。

参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n:");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=n-i; j++)
printf(" ");
for(j=1; j<=2*i-1+(n-1); j++)
printf("*");
printf("\n");
}
```

```
}
```

【4.9】分析：对于空心图形，我们可以在上题的基础上，对于打印'*'号的循环进行修改，仅在循环开始值(j=1)和循环结束值(j=2*(i-1)+n)时打印'*'号，其它位置都打印空格。另一种思路是将每行打印的空格和'*'的两个循环合为一体考虑，在判断出需要打印'*'的两个位置及第一行和最后一行相应位置外，其余位置都打印空格。

参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n:");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=2*n+i-3; j++)
if(j==n-i+1 || j>n-i+1 && (i==1||i==n)) printf("*");
else printf(" ");
printf("*\n");
}
}
```

【4.10】分析：此图形可以理解为两个空心梯形反向连接而成，因此可以利用上题的思路进行输出。

参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n:");
```

```
scanf("%d",&n);
for(i=1; i<=n; i++) /* 输出图形的上半部分(含中心行) */
{ for(j=1; j<=2*n-i-1; j++)
if(j==i) printf("*");
else printf(" ");
printf("*\n");
}
for(i=1; i{ for(j=1; j<=n+i; j++)
if(j==n-i) printf("*");
else printf(" ");
printf("*\n");
}
}
```

【4.11】分析：此题与上题的区别在于打印'*'号的位置不同，编程时要找出应打印'*'号的位置和两个循环变量 i、j 以及行数 n 的关系。

参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n:");
scanf ("%d", &n);
for(i=1; i<=n; i++) /* 输出图形的上半部分(含中心行) */
{ for(j=1; j<=2*n-i; j++)
```

```
if(j==n-i+1 || j>n-i+1 && i==1) printf("*");
else printf(" ");
printf("*\n");
}
for(i=1; i{ for(j=1; j<=3*(n-1)-i; j++)
if(j==i+1 || j>i+1 && i==n-1) printf("*");
else printf(" ");
printf("*\n");
}
}
```

【4.12】参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n:");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
if(j==n-i+1 || i==1 || i==n) printf("*");
else printf(" ");
printf("\n");
}
}
```

【4.13】参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n: ");
scanf("%d", &n);
for(i=1; i<=n; i++) /* 输出图形的上半部分(含中心行) */
{ for(j=1; j<=n-i; j++)
if(j==1 || j==n-i+1) printf("* ");
else printf(" ");
printf("\n");
}
for(i=1; i{ for(j=1; j<=i+1; j++)
if(j==1 || j==i+1) printf("* ");
else printf(" ");
printf("\n");
}
}
```

【4.14】参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n: ");
scanf("%d",&n);
```

```
for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
if(j==1 || j==i || j==n) printf("*");
else printf(" ");
printf("\n");
}
}
```

【4.15】参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n: ");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=n+i-1; j++)
if(j>n-i) printf("*");
else printf(" ");
printf("\n");
}
for(i=1; i{ for(j=1; j<=2*n-i-1; j++)
if(j>i) printf("*");
else printf(" ");
printf("\n");
```

```
}

}
```

【4.16】参考答案：

```
main( )

{ int i,j,n;

printf("\nPlease Enter n: ");

scanf("%d",&n);

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

{ for(j=1; j<=n+i-2; j++)

if(j==n-i+1) printf("*");

else printf(" ");

printf("*\n");

}

}
```

【4.17】分析：首先对图形进行设计，坐标的 X 轴和 Y 轴分别对应屏幕的列和行，一个正弦函数的周期为0～360度，我们把一个步长定义为10度，打印时每换一行等于函数的自变量增加10度；屏幕的列宽为80，函数值为0对应屏幕的第40列，sin(x)的值在-1～1，变换成列数为以0为中心的-30～30，对应屏幕上第10～70列。设计程序时，控制换行的自变量 i 乘以10得到正弦函数的 X 值，调用库函数 sin()求出函数值再乘以30输出的列宽，因为我们以屏幕的第40列为0点，故再加上40得到应在屏幕上显示的点。

参考答案：

```
#define PAI 3.14159

#include main( )

{ double x;
```

```
int y,i,yy;

for(i=1; i<80; i++) /* 打印图形的第一行 */

if(i==40) printf("*"); /* i 控制打印的列位置 */ else printf("-");

printf("\n");

for(x=10.0; x<=360.0; x+=10.) /* 从10度到360度 */

{ y = 40+30*sin(x*PAI/180.0); /* 计算对应的列 */

yy = 40>y ? 40 : y; /* 下一行要打印的字符总数 */

for (i=1; i<=yy; i++) /* 控制输出图形中的一行 */

{ if(i==y) printf("*"); /* i 控制打印的列位置 */

else if(i==40) printf("|"); /* 打印中心的竖线 */

else printf(" ");

}

printf("\n");

}

}
```

【4.18】分析：首先设计屏幕图形，如果预计圆形在屏幕上打印20行，所以定义圆的直径就是20，半径为10，圆的方程是X2×Y2=R2,因为图形不是从中心开始打印而是从边沿开始，所以Y从10变化到-10，根据方程求出X,对求得的X值再根据屏幕行宽进行必要的调整得到应打印的屏幕位置。

参考答案：

```
#include main( )

{ double y;

int x,m;
```

```
for(y=10; y>=-10; y--) /* 圆的半径为10 */
{ m = 2.5 * sqrt(100-y*y); /* 计算行 y 对应的列坐标 m */
for(x=1; x<30-m; x++)
printf(" "); /* 输出圆左侧的空白 */
printf("*"); /* 输出圆的左侧 */
for(; x<30+m; x++)
printf(" "); /* 输出圆的空心部分 */
printf("*\n"); /* 输出圆的右侧 */
}
}
```

【4.19】参考答案：

```
#include #include main( )
{ double y;
int x, m, n, yy;
for( yy=0; yy<=20; yy++)
{ y = 0.1*yy;
m = acos(1-y)*10;
n = 45 * (y-1)+31;
for( x=0; x<=62; x++ )
if( x==m && x==n ) printf("+");
else if(x==n) printf("+");
else if(x==m || x==62-m) printf("*");
```

```
else printf(" ");

printf("\n");

}

}
```

【4.20】分析：编程的关键为两点，一是使用控制输出的行和列，这方面的内容在前面已经叙述，另一点是输出的数字和所在行、列关系。此题第一行输出的数字恰好是列数，从第二行起每行的数字均比上一行增 n。

参考答案：

```
main( )

{ int i,j,n;

printf("\nPlease Enter n: ");

scanf("%d",&n);

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

{ for(j=1; j<=n; j++)

printf("M",(i-1)*n+j);

printf("\n");

}

}
```

【4.21】分析：此题的关键是找到输出数字和行、列数的关系。审查图形中每行中数字的关系发现，右边数字和前面数字之差逐次增1；同列数字依然是这样的关系，编程的关键转换为找到每一行左方的第一个数字，然后利用行和列的循环变量进行运算就可得到每个位置的数字。用 ai,j 此表示第 i 行第 j 列的数字，则 a11=1；由第 i 行第一列的数字推出第 i+1 行第一列的数字是 ai+1,1 = ai,1+i；同样由第 j 列推出第 j+1列的数字是 ai,j+1 = ai,j+i+j。另外只有当 j

参考答案：

```
main( )
{ int i,j,m,n,k=1;  /* k是第一列元素的值 */
printf("Please enter m=" ");
scanf("%d",&m);
for(i=1; i<=m; i++)
{ n=k;  /* n第i行中第1个元素的值 */
for(j=1; j<=m-i+1; j++)
{ printf("=",n);
n = n+i+j;  /* 计算同行下一个元素的值 */
}
printf("\n");
k=k+i;  /* 计算下一行中第1个元素 */
}
}
```

【4.22】参考答案：

```
main( )
{ int i,j,n;
printf("\nPlease Enter n: ");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
if(j<=i) printf(" 1");
```

```
else printf("=",j-i+1);

printf("\n");

}

}
```

【4.23】分析：可用不同的方案解决此问题，为了开阔读者的思路，这里给出了两个参考答案，其中第二个答案是使用了递归方法。

方案一：

首先寻找数字输出数字和行列的关系。

每圈有四个边，把每边的最后一个数字算为下边的开始，最外圈每边数字个数是n-1个，以后每边比外边一边少两个数字。

因为数字是一行一行输出的，再分析每行数字的规律。实际没有的数字有三种规律：位于对角线之间的数字是上半图增一，下半图减一。对角线左侧的各列，右侧比左侧增加了一圈数字，例如数字39和它左侧的22比较，数字39所在的圈每边4个数字，左侧22加上一圈16个数字在加1就是39。同理，对角线右侧的各列，则减少一圈的数字个数。

根据以上分析，用两个对角线将图形分为四个区域，如下图所示，图中黑斜体字为对角线上的数字。

1 2 3 4 5 6 *7*

24 25 26 27 28 29 *8*

23 *40* *41* *42* *43* 30 9

22 *39* 48 49 44 31 *10*

21 38 *47* 46 *45* 32 *11*

20 37 36 35 34 33 *12*

19 18 17 16 15 14 *13*

为叙述方便我们称四个区域为上、下、左、右区。设i、j为行列号，n为图形的总行数，则满足各区的范围是，上区：j>=i 且 j<=n-i+1 ；下区：j<=i 且 j>=n-i+1 ；左区：ji 且

j>n-i+1 。

现在问题是，如果知道一行在不同区域开始第一个位置的数字，然后该区后续的数字就可利用前面分析的规律得到。

对于右区开始各行第一个数字最易求出，为4*(n-1)-i+1。后续一个和同行前一个数字之差是4*[n-1-(j-1)*2]+1,其中方括号内是每边的数字个数。

对角线上的数字是分区点，对角线上相临数字仍然相差一圈数字个数，读者自行分析得到计算公式。

右区开始的第一个数字可以从上区结束时的数字按规律求出。

下述程序用变量 s 保存分区对角线上的数字。

参考答案一：

```
main()
{ int i,j,k,n,s,m,t;
printf("Please enter n:");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ s=(i<=(n+1)/2)? 1:3*(n-(n-i)*2-1)+1;
m=(i<=(n+1)/2)? i:n-i+1;  /* m-1是外层圈数 */
for(k=1; kfor(j=1; j<=n; j++)
{ if(j>=n-i+1 && j<=i) /* 下区 */
t=s-(j-(n-i))+1;
if(j>=i && j<=n-i+1) /* 上区 */
t=s+j-i;
if(j>i && j>n-i+1) /* 右区 */
```

```
t-=4*(n-2*(n-j+1))+1;

if(j{ if(j==1) t=4*(n-1)-i+2;

else t+=4*(n-2*j+1)+1;

}

printf("M",t);

}

printf("\n");

}

}
```

方案二：

根据本题图形的特点，我们可以构造一个递归算法。我们可以将边长为N的图形分为两部分：第一部分最外层的框架，第二部分为中间的边长为N-2的图形。

对于边长为N的正方型，若其中每个元素的行号为i（1≤i≤N），列号为j（1≤j≤N），第1行第1列元素表示为a1,1（a11=1），则有：

对于最外层的框架可以用以下数学模型描述：

上边： a1,j=a1,1+j-1 （j≠1）

右边： ai,N=a1,1+N+i-2 （i≠1）

下边： ai,1=a1,1+4N-i-3 （i≠1）

左边： aN,j=a1,1+3N-2-j （j≠1）

对于内层的边长为N-2的图形可以用以下数学模型描述：

左上角元素：ai,i=ai-1,i-1+4(N-2i-1) （i＞1）

若令：ai,j=fun(ai-1,i-1+4(N-2i-1)，当：i＜(N+1)/2且j＜(N+1)/2时，min=MIN(i,j)，则有：

a2,2 = fun(a1,1, min, min, n)

ai,j=fun(a2,2, i-min+1, j-min+1, n-2*(min-1))

我们可以根据上述原理，分别推导出 i 和 j 为其它取值范围时的 min 取值。根据上述递归公式，可以得到以下参考程序。

参考答案二:

```
#include #define MIN(x,y) (x>y) ? (y) : (x)

fun ( int a11, int i, int j, int n)

{ int min, a22;

if( i==j && i<=1 ) return(a11);

else if( i==j && i<=(n+1)/2) return( fun(a11,i-1,i-1,n)+4*(n-2*i+3));

else if( i==1 && j!=1) return( a11+j-1 );

else if( i!=1 && j==n) return( a11+n+i-2 );

else if( i!=1 && j==1 ) return ( a11+4*n-3-i );

else if( i==n && j!=1 ) return ( a11+3*n-2-j );

else

{ if(i>=(n+1)/2 && j>=(n+1)/2) min = MIN(n-i+1,n-j+1);

else if(i<(n+1)/2 && j>=(n+1)/2) min = MIN(i,n-j+1);

else if(i>=(n+1)/2 && j<(n+1)/2) min = MIN(n-i+1,j);

else min = MIN(i,j);

a22 = fun(a11,min,min,n);

return(fun(a22, i-min+1, j-min+1, n-2*(min-1)));

}
```

```
}

main()

{ int all=1, i, j, n;

printf("Enter n=");

scanf("%d", &n);

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

{ for(j=1; j<=n; j++)

printf("M", fun(all,i,j,n) );

printf("\n");

}

}
```

【4.24】分析：此题的关键还是要找到输出数字 aij 和行列数 i、j 的关系。为此将图形分为四个区域如下图：

3 3 3 3 3

3 2 2 2 3

3 2 1 2 3

3 2 2 2 3

3 3 3 3 3 (此图 n 为5)

在左上区域，即 i<=(n+1)/2、j<=(n+1)/2时，输出数字为(n+1)/2-i+1和(n+1)/2-j+1中的大者，记为 max{(n+1)/2-i+1,(n+1)/2-j+1}；在右上区，即 i<=(n+1)/2、j>(n+1)/2时，输出数字为 max{(n+1)/2-i+1,j-n/2}；在左下区，即 i>(n+1)/2、j<=(n+1)/2时，输出数字为 max{i-n/2,(n+1)/2-j+1}；在右下区，即 i>(n+1)/2、j>(n+1)/2时，输出数字为 max{i-n/2,j-n/2}。

参考答案：

```
#define max(x,y) ((x)>(y)?(x):(y))

main( )

{ int i,j,n;

printf("\nPlease Enter n:");

scanf("%d",&n);

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

{ for(j=1; j<=n; j++)

if(i<=(n+1)/2)

if(j<=(n+1)/2)

printf("M",max((n+1)/2-i+1,(n+1)/2-j+1));

else

printf("M",max((n+1)/2-i+1,j-n/2));

else if(j<=(n+1)/2)

printf("M",max(i-n/2,(n+1)/2-j+1));

else

printf("M",max(i-n/2,j-n/2));

printf("\n");

}

}
```

【4.25】分析：前面我们已经见到过上下对称的图形，这是一个左右对称的图形，垂直中心线上的数字恰好是行号，在每行位于图形垂直中心线左方的数字是逐渐增加的，而右方是逐渐减小的。j==i 是分区的标志，左方输出数字就是列数 j，而右方的数字从 i 开始逐步减小 1。

参考答案：

```
main()
{ int i,j;
for(i=1; i<=9; i++)
{ for(j=1; j<=9-i; j++) printf(" ");
for(j=1; j<=i; j++) printf("-",j);
for(j=i-1; j>=1; j--) printf("-",j);
printf("\n");
}
}
```

【4.26】分析：这类输出字符的图形和输出数字的图形考虑是近似的，因为字符的ASCII码就是一个整数。在字符码值的变化过程中，应该注意应该判断码值是否超出字符的范围，进行必要的处理，为了保持程序的简洁，本题没有考虑这个问题，在下题里对这个问题进行了处理。

参考答案：

```
main( )
{ char c='Z';
int i,j,n;
printf("\nPlease Enter n:");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ for(j=1; j<=n+i-2; j++)
if(j==n-i+1) printf("%c",c--);
```

```
else printf(" ");

printf("%c\n",c--);

}

for(i=1; i{ for(j=1; j<=2*(n-1)-i; j++)

if(j==i+1) printf("%c",c--);

else printf(" ");

printf("%c\n",c--);

}

}
```

【4.27】分析：此题与上题相近，区别在于输出时字符的ASCII码值的变化在图形的中间一行为最大，同时一行的两个字符是相同的。程序考虑在输入字符时设计了一个循环，保证输入的是英文字母。字符变化后进行了处理，程序中使用条件运算。在字符码值增加的过程中，首先判断是大写还是小写字符，然后判断字符码值是否超出英文字母z(或Z)，如果超出则重新赋为a(或A)；在输出图象下半部分时，ASCII码值减少用同样的思路进行判断。在判断字符大小写（条件语句的第一个判断）时，用的是两个不同的值，请读者自行思考为什么，用同一个值是否可以？

参考答案：

```
main( )

{ char c;

int i,j,n;

do

{ printf("\nPlease Enter n,char:");

scanf("%d,%c",&n,&c);

}while(c<'A'||c>'Z'&&c<'a'||c>'z');
```

```
for(i=1; i<=n; i++)
{ for(j=1; j<=n+i-2; j++)
if(j==n-i+1) printf("%c",c);
else printf(" ");
printf("%c\n",c++);
c=c<'a'?(c>'Z'?'A':c):(c>'z'?'a':c);
}
c-=2;
c=c<'Z'?(c<'A'?'Z':c):(c<'a'? 'z':c);
for(i=1; i{ for(j=1; j<=2*(n-1)-i; j++)
if(j==i+1) printf("%c",c);
else printf(" ");
printf("%c\n",c--);
c=c<'Z'?(c<'A'?'Z':c):(c<'a'? 'z':c);
}
}
```

【4.28】参考答案：

```
#define max(x,y) ((x)>(y)?(x):(y))
main( )
{ char c;
int i,j,n;
do
```

```
{ printf("\nPlease Enter n, char:");
scanf("%d,%c",&n,&c);
}while(c<'A'||c>'Z'&&c<'a'||c>'z');
for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
if(i<=(n+1)/2)
if(j<=(n+1)/2)
printf(" %c",c-max((n+1)/2-i+1,(n+1)/2-j+1)+(n+1)/2);
else
printf(" %c",c-max((n+1)/2-i+1,j-n/2)+(n+1)/2);
else
if(j<=(n+1)/2)
printf(" %c",c-max(i-n/2,(n+1)/2-j+1)+(n+1)/2);
else
printf(" %c",c-max(i-n/2,j-n/2)+(n+1)/2);
printf("\n");
}
}
```

【4.29】参考答案：

```
#define max(x,y) ((x)>(y)?(x):(y))
main( )
{ char c;
```

```
int i,j,n;
do
{ printf("\nPlease Enter n, char:");
scanf("%d,%c",&n,&c);
}while(c<'A'||c>'Z'&&c<'a'||c>'z');
for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
if(i<=(n+1)/2)
if(j<=(n+1)/2)
printf(" %c",c-max((n+1)/2-i+1,(n+1)/2-j+1)+1);
else
printf(" %c",c-max((n+1)/2-i+1,j-n/2)+1);
else
if(j<=(n+1)/2)
printf(" %c",c-max(i-n/2,(n+1)/2-j+1)+1);
else
printf(" %c",c-max(i-n/2,j-n/2)+1);
printf("\n");
}
}
```

【4.30】参考答案：

```
#include main()
```

```
{ int i, j;
for(i=1; i<10; i++)
printf("M",i);
printf("\n----------------------------------\n");
for(i=1; i<10; i++)
{ for(j=1; j<10; j++)
if(j
printf("\n");
}
}
```

【4.31】参考答案：

```
#include main( )
{ int i,j;
for(i=1; i<10; i++)
printf("M",i);
printf("\n----------------------------------\n");
for(i=1; i<10; i++)
{ for(j=1; j<10; j++)
if(j<10-i) printf(" ");
else printf( "M" , (10-i)*j);
printf("\n");
}
```

```
}
```

【4.32】参考答案：

```
#include "math.h"
main()
{ int flag=0;
float a,b,c,s;
do
{ printf("Please enter a b c:");
scanf("%f%f%f",&a,&b,&c);
if(a>b+c || b>a+c || c>a+b)
flag=1;
}while(flag);
s=(a+b+c)/2;
printf("S=%f",s=sqrt((s-a)*(s-b)*(s-c)));
}
```

【4.33】参考答案：

```
#include main( )
{ int j;
long n;  /* 使用长整型变量，以免超出整数的表示范围 */
printf("Please input number:");
scanf("%ld", &n);
for(j=999; j>=100; j--)/* 可能取值范围在999到100之间，j从大到小 */
```

```
if(n%j == 0 ) /* 若能够整除 j，则 j 是约数，输出结果 */
{ printf("The max factor with 3 digits in %ld is: %d.\n",n,j);
break;  /* 控制退出循环 */
}
}
```

【4.34】参考答案：

```
#define E 0.000001
main()
{ float x,y=1,s=0;
printf("Please enter x=");
scanf("%f",&x);
while(1/y>E)
{ s=s+1/y;
y=y*x;
}
printf("S=%f\n",s);
}
```

【4.35】参考答案：

```
#include main( )
{ int class1, class2, class3;
char ch;
class1=class2=class3=0;  /* 初始化分类计数器 */
```

```
do
{ ch=getch( );
switch(ch)
{ case '0': case '1': case '2': case '3': case '4':
case '5': case '6': case '7': case '8': case '9':
class1++;  break;  /* 对分类 1 计数 */
case '+': case '-': case '*': case '/': case '%': case '=':
class2++;  break;  /* 对分类 2 计数 */
default: class3++;  break;  /* 对分类 3 计数 */
}
}while (ch!= '\\');  /* 字符'\'在 C 程序中要使用转义符'\\' */
printf("class1=%d, class2=%d, class3=%d\n", class1, class2, class3);
}
```

【4.36】分析：程序的关键是怎样判断一个单词。由单词的定义已知它是用空格、制表符或换行符分隔开的，两个字符之间没有空格、制表符或换行符，则认为是一个单词中的两个字符。

参考答案：

```
#define EOF -1
#define YES 1
#define NO 0
#include main( ) /* 对输入的行、字符和单词进行计数统计 */
{ int c, nl, nc, nw, inword;
inword=NO;  /* inword=NO 已处理的最后一个字符是空格、\t 或\n */
```

```
/* inword=YES 已处理的最后一个字符不是空格、\t 或\n */
nl=nc=nw=0;  /* 行、字符、字计数器置0 */
while((c=getchar())!= EOF)
{ ++nc;  /* 进行字符计数 */
if(c=='\n' )
++nl;  /* 进行行计数 */
if(c=='\t'||c=='\n'||c==' ')
inword=NO; /* 如果读入的字符是空格、\t 或\n，则置 inword 为 NO */
else /* 读入的字符不是空格、\t 或\n */
if(inword==NO) /* 如果前一个字符是空格、\t 或\n */
{ inword=YES;  /* 则读入的字符为一个单词的第一个字符*/
++nw;  /*置 inword 为 YES，进行单词计数 */
}
}
printf("Lines=%d\nWords=%d\nChars=%d\n",nl,nw,nc);  /* 输出结果 */
}
```

【4.37】参考答案：

```
#define E 0.000001
#include "math.h"
main()
{ int i,k=1;
float x,y,t=1,s,r=1;
```

```
printf("Please enter x=");
scanf("%f",&x);
for(s=x,y=x,i=2; fabs(r)>E; i++)
{ t=t*(i-1);
y=y*x*x;
k=k*(-1);
r=k*y/t/(2*i-1);
s=s+r;
}
printf("S=%f\n",s);
}
```

【4.38】参考答案：

```
main()
{ int i;
float s=0;
for(i=1; i<=100; i++)
s=s+i;
for(i=1; i<=50; i++)
s=s+i*i;
for(i=1; i<=10; i++)
s=s+1.0/i;
printf("Result=%f\n",s);
```

```
}
```

【4.39】参考答案：

```
main()
{ int i;
float s=1;
for(i=1; i<=20 && 1.0/i/(i+1)>0.001; i++)
s=s+1.0/i/(i+1);
printf("Result=%f i=%d\n",s,i);
}
```

【4.40】参考答案：

```
#include main()
{ float x,eps,s,y=0,y0,t;
int n,j;
printf("Enter x & eps:");
scanf("%f%f", &x, &eps);
n=t=j=1;
s=x;
do
{ y0=y;
if(n%2==0) y=y-s/t;
else y=y+s/t;
s *= x*x;  /* 求 x 的乘方 */
```

```
t *= (j+1)*(j+2);  /* 求n! */
j += 2;
n++;
}while( fabs(y0-y) > eps );  /* 控制误差 */
printf("sin(%f)=%f\n",x,sin(x));  /* 输出标准sin(x)的值 */
printf("%d,sin(%f)=%f\n",n,x,y);  /* 输出计算的近似值 */
}
```

【4.41】参考答案：

```
main( )
{ int i,j,num,a[10];
for(i=0; i<10; i++)
{ printf("Enter No. %d:", i+1);
scanf("%d",&num);
for(j=i-1; j>=0&&a[j]>num; j--)
a[j+1]=a[j];
a[j+1]=num;
}
for(i=0; i<10; i++)
printf ("No.%d=%d\n", i+1, a[i]);
}
```

【4.42】参考答案：

```
main()
```

```
{ int n;
printf("Please enter n:");
scanf("%d",&n);
while(n>0)
{ printf("%d",n);
n=n/10;
}
}
```

【4.43】参考答案：

```
main()
{ int i,n;
long s1=0,s2=0;
printf("Please enter N:");
scanf("%d",&n);
if(n>=0)
for(i=n; i<=2*n; i++)
s1=s1+i;
else
for(i=n; i>=2*n; i--)
s1=s1+i;
i=n;
if(i>=0)
```

```
while(i<=2*n)

s2=s2+i++;

else

while(i>=2*n)

s2=s2+i--;

printf("Result1=%ld result2=%ld\n",s1,s2);

}
```

【4.44】分析：据题意，阶梯数满足下面一组同余式：

x≡1 (mod2)

x≡2 (mod3)

x≡4 (mod5)

x≡5 (mod6)

x≡0 (mod7)

参考答案：

```
#include main()

{ int i=1;  /* i为所设的阶梯数 */

while( !((i%2==1)&&(i%3==2)&&(i%5==4)&&(i%6==5)&&(i%7==0)) )

++i;  /* 满足一组同余式的判别 */

printf("Staris_number=%d\n", i );

}
```

【4.45】参考答案：

```
main( )
```

```
{ int i,n,a;
for(i=0;  ; i++)
{ if(i%8==1)
{ n=i/8;
if(n%8==1)
{ n=n/8;
if(n%8==7) a=n/8;
}
}
if(i==4)
{ n=i/17;
if(n==15) n=n/17;
}
if(2*a==n)
{ printf("result=%d\n",i);
break;
}
}
}
```

【4.46】分析：二分法的基本原理是，若函数有实根，则函数的曲线应当在根这一点上与 x 轴有一个交点，在根附近的左右区间内，函数值的符号应当相反。利用这一原理，逐步缩小区间的范围，保持在区间的两个端点处的函数值符号相反，就可以逐步逼近函数的根。

参考答案：

```
#include "stdio.h"
#include "math.h"
main()
{ float x0, x1, x2, fx0, fx1, fx2;
do
{ printf("Enter x1,x2:");
scanf("%f,%f", &x1, &x2);
fx1=2*x1*x1*x1-4*x1*x1+3*x1-6;  /* 求出x1点的函数值fx1 */
fx2=2*x2*x2*x2-4*x2*x2+3*x2-6;  /* 求出x2点的函数值fx2 */
}while (fx1*fx2>0);  /* 保证在指定的范围内有根，即fx的符号相反 */
do
{ x0=(x1+x2)/2;  /* 取x1和x2的中点 */
fx0=2*x0*x0*x0-4*x0*x0+3*x0-6;  /* 求出中点的函数值fx0 */
if((fx0*fx1)<0) /* 若fx0和fx1符号相反 */
{ x2=x0;  /* 则用x0点替代x2点 */
fx2=fx0;
}
else
{ x1=x0;  /* 否则用x0点替代x1点 */
fx1=fx0;
}
}while(fabs((double)fx0)>=1e-5);  /* 判断x0点的函数与x轴的 距离 */
```

```
printf("x=%6.2f\n", x0);

}
```

【4.47】分析：做圆的内接4边形，从圆心和4边形顶点连接形成4个三角形，可以求出每个三角形的面积（r2/2）现在我们知道三角形的面积和两个边长（均为半径 a=r、b=r），可以用公式：S=s(s-a)(s-b)(s-c)求出第三边 c。我们将内接4边形换为内接8边形，原来的三角形被一分为二，故 c/2就是每个三角形的高，面积又是可以求出的。再将三角形一分为二，……。当三角形的面积求出时，内接多边形的面积就可求出。

参考答案：

```
main()

{ int n=4;

double r=10,s,cr,c,p;

s=r*r/2;

do

{ cr=n*s;

p=16*r*r*r*r-64*s*s;

c=(4*r*r-sqrt(p))/2;

c=sqrt(c);

s=c*r/4;

n=2*n;

}while(n*s-cr>1.0e-10);

printf("PAI=%lf\n",cr/r/r);

}
```

【4.48】分析：根据题意，总计将所有的鱼进行了五次平均分配，每次分配时的策略是相同的，即扔掉一条后剩下的鱼正好分为五份，然后拿走自己的一份，余下其它四份。假定鱼的

总数为 x，则 x 可以按照题目的要求进行五次分配：x-1后可被5整除，余下的鱼为4×（x-1）÷ 5。若 x 满足上述要求，则 x 就是题目的解。

参考答案：

```
main( )
{ int n,i,x,flag=1;  /* flag:控制标记 */
for(n=6; flag; n++) /* 采用试探的方法，令试探值 n 逐步加大 */
{ for(x=n,i=1; flag && i<=5; i++) /* 判断是否可按要 */
if((x-1)%5 == 0) x=4*(x-1)/5;  /* 求进行5次分配 */
else flag=0;  /* 若不能分配则置标记 flag=0退出分配过程 */
if(flag) break;  /* 若分配过程正常,找到结果,退出试探的过程 */
else flag=1;  /* 否则继续试探下一个数 */
}
printf("Total number of fish catched = %d\n", n);  /* 输出结果 */
}
```

【4.49】分析：按照题目的要求造出一个前两位数相同、后两位数相同且相互间又不同的整数，然后判断该整数是否是另一个整数的平方。

参考答案：

```
#include "math.h"
main()
{ int i,j,k,c;
for(i=1; i<=9; i++) /* i:车号前二位的取值 */
for(j=0; j<=9; j++) /* j:车号后二位的取值 */
if( i!=j ) /* 判断两位数字是否相异 */
```

```
{ k=i*1000+i*100+j*10+j;  /* 计算出可能的整数 */

for( c=31; c*cif(c*c==k)

printf("Lorry_No. is %d .\n", k);  /* 若是，打印结果 */

}

}
```

【4.50】分析：用穷举法解决此类问题。设任取红球的个数为 i，白球的个数为 j，则取黑球的个数为8-i-j， 据题意红球和白球个数的取值范围是0～3，在红球和白球个数确定的条件下，黑球的个数取值应为8-i-j<=6。

参考答案：

```
main( )

{ int i,j,count=0;

printf(" RED BALL WHITE BALL BLACK BALL\n");

printf("----------------------------------------\n");

for(i=0; i<=3; i++) /* 循环控制变量 i 控制任取红球个数0～3 */

for(j=0; j<=3; j++) /* 循环控制变量 j 控制任取白球个数0～3 */

if((8-i-j)<=6)

printf("-: %d %d %d\n",++count, i,j,8-i-j);

}
```

【4.51】分析：此题采用穷举法。

参考答案：

```
main()

{ int x,y,z,j=0;

for(x=0;  x<=33;  x++)
```

```
for(y=0;  y<=(100-3*x)/2;  y++)
{ z=100-x-y;
if( z%2==0 && 3*x+2*y+z/2==100)
printf("-:l=- m=- s=-\n",++j,x,y,z);
}
}
```

【4.52】分析：此题采用穷举法。

参考答案：

```
main( )
{ int f1,f2,f5,count=0;
for(f5=0; f5<=20; f5++)
for(f2=0; f2<=(100-f5*5)/2; f2++)
{ f1=100-f5*5-f2*2;
if(f5*5+f2*2+f1==100)
printf("No.- >> 5: M 2: - 1: -\n",++count,f5,f2,f1);
}
}
```

【4.53】分析：此题采用穷举法。

参考答案：

```
main( )
{ long int i,j,k,count=0;
for(i=1; i*i<=200; i++)
```

```
for(j=1; j*j<=200; j++)

for(k=1; k*k<=200; k++)

if(i*i==(j*j+k*k))

{ printf("\nA^2==B^2+C^2: %4ld%4ld%4ld",i,j,k);

count++;

}

printf("\ncount=%ld",count);

}
```

【4.54】分析：此题采用穷举法。可设整数N的千、百、十、个位为i、j、k、m，其取值均为0～9，则满足关系式：(i*103+j*102+10k+m)*9=(m*103+k*102+10j+i) 的i、j、k、m即构成N。

参考答案：

```
#include main( )

{ int i;

for(i=1002; i<1111; i++) /* 穷举四位数可能的值 */

if(i*1000+i/10*100+i/100*10+i/1000==i*9 )

printf("The number satisfied states condition is: %d\n", i);

/* 判断反序数是否是原整数的9倍若是则输出 */

}
```

【4.55】分析：此题采用穷举法。

参考答案：

```
main()

{ int i,j,n,k,a[16]={0};
```

```
for(i=1; i<=1993; i++)

{ n=i; k=0;

while(n>0) /* 将十进制数转变为二进制数 */

{ a[k++]=n%2;

n=n/2;

}

for(j=0; jif(a[j]!=a[k-j-1]) break;

if(j>=k)

{ printf(" %d: ",i);

for(j=0; jprintf("-",a[j]);

printf("\n");

}

}

}
```

【4.56】分析：类似的问题从计算机算法的角度来说是比较简单的，可以采用最常见的穷举法解决。程序中采用循环穷举每个字母所可能代表的数字，然后将字母代表的数字转换为相应的整数，代入算式后验证算式是否成立即可解决问题。

参考答案：

```
#include main( )

{ int p,e,a,r;

for(p=1; p<=9; p++) /* 从1到9穷举字母 p 的全部可能取值 */

for(e=0; e<=9; e++) /* 从0到9穷举字母 e 的全部可能取值 */

if(p!=e)
```

```
for(a=1; a<=9; a++) /* 从0到9穷举字母a的全部可能取值 */

if(a!=p && a!=e)

for(r=0; r<=9; r++) /* 从0到9穷举字母r */

if(r!=p && r!=e && r!=a /* 四个字母互不相同 */

&& p*1000+e*100+a*10+r-(a*100+r*10+a)

== p*100+e*10+a )

{ printf(" PEAR %d%d%d%d\n", p, e, a, r);

printf(" - ARA - %d%d%d\n", a, r, a);

printf("-------- ---------\n");

printf(" PEA %d%d%d\n", p, e, a);

}

}
```

【4.57】参考答案：

```
main()

{ int i,n,k,a[3],b[3];

for(i=248; i<=343; i++)

{ for(n=i,k=0; n>0; n/=7)

a[k++]=n%7;

for(n=i,k=0; n>0; n/=9)

b[k++]=n%9;

if(k==3)

for(n=0; nif(a[n]!=b[k-n-1])
```

```
break;
if(n==k)
printf("%d\n",i);
}
}
```

【4.58】参考答案：

```
main()
{ int i,j,k,m,error;
for(i=6; i<=2000; i+=2)
{ error=1;
for(j=2; j
{ for(k=2; kif(j%k==0) /* j能够被小于它的一个数整除就不是素数 */
break;
if(k>=j) /* j是素数 */
{ m=i-j;
for(k=2; kif(m%k==0)
break;
if(k>=m) /* m也是素数，输出结果 */
{ printf("M = M + M\n",i,j,m);
error=0;
break; }
}
```

```
}

if(error)

printf("M error!");

}

}
```

【4.59】分析：可采用穷举法，依次取1000以内的各数（设为 i），将 i 的各位数字分解后，据阿姆斯特朗数的性质进行计算和判断。

参考答案：

```
#include main()

{ int i,t,k,a[4]={0};

printf ("There are following Armstrong number smaller than 1000:\n");

for(i=2; i<1000; i++) /* 穷举要判定的数 i 的取值范围1～1000 */

{ for(t=0,k=1000; k>=10; t++) /* 截取整数 i 的各位(从高位向低位) */

{ a[t]=(i%k)/(k/10);     /* 分别赋给 a[0]～a[3] */

k /= 10;

}

if(a[0]*a[0]*a[0]+a[1]*a[1]*a[1]+a[2]*a[2]*a[2]+a[3]*a[3]*a[3]==i)

printf(" %d ",i);  /* 判断 i 是否为阿姆斯特朗数，*/

/* 若满足条件，则输出 */

}

}
```

【4.60】参考答案：

```
main( )

{ int j,k,n,m;

printf("Please enter n:");

scanf("%d",&n);

for(j=2; j{ for(k=2; kif(j%k==0) break;  /* j 能够被小于它的一个数整除就不是素数
*/

if(k>=j) /* j 是素数 */

{ m=n-j;

for(k=2; kif(m%k==0) break;

if(k>=m) /* m 也是素数，输出结果 */

{ printf("M = M + M\n",n,j,m);

break;

}

}

}

}
```

【4.61】分析：按照亲密数定义，要判断数 a 是否有亲密数，只要计算出 a 的全部因子的累加和为 b，再计算 b 的全部因子的累加和为 n，若 n 等于 a 则可判定 a 和 b 是亲密数。计算数 a 的各因子的算法：用 a 依次对 i(i=1～a/2)进行模运算，若模运算结果等于0，则 i 为 a 的一个因子；否则结束对 a 的因子的计算。

参考答案：

```
#include #include main( )

{ int a, i, m, n;
```

```
printf("Friendly-numbers pair samller than 3000:\n");
for(a=1; a<3000; a++) /* 穷举3000以内的全部整数 */
{ for(m=0,i=1; i<=a/2; i++ ) /* 计算数a的各因子,各因子之和存于m */
if(!(a%i))
m+=i;  /* 计算m的各因子，各因子之和存于n */
for(n=0,i=1; i<=m/2; i++)
if(!(m%i))
n+=i;
if(n==a && aprintf(" M～M",a,m);
}
```

【4.62】参考答案：

```
#include #include main( ) /* 猜数程序 */
{ int magic;  /* 计算机"想"的数 */
int guess;  /* 人猜的数 */
int counter;
magic=rand( );  /* 通过调用随机函数任意"想"一个数 */
guess=magic-1;  /* 初始化变量guess的值 */
counter=0;  /* 计数器清零 */
while(magic != guess)
{ printf("guess the magic number:");
scanf("%d", &guess);              /* 人输入所猜的数 */
counter++;
```

```
if(guess>magic)

printf("**** Wrong **** too hight\n");

else if(guessprintf("**** Wrong **** too low\n");

}

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

printf("guess counter is %d\n", counter);

}
```

【4.63】分析：直接计算阶乘的结果显然超出整型数的范围。此题的关键是如何减少计算中数的规模，注意在计算过程中出现0后，我们可以先行统计0的个数，然后将0从结果中移去，另外，结果仅保存个位数即可，其它位的数不会对0的个数产生影响。

参考答案：

```
main()

{ int i,n=0;

long s=1;

for(i=1; i<=1000; i++)

{ s=s*i;

while(s==0)

{ s=s/10;

n++;

}

s=s;

}

printf("n=%d,s=%d\n",n,s);
```

```
}
```

【4.64】参考答案：

```
main()
{ int i,j,b[3][2];
int a[2][3]={{1,2,3},{4,5,6}};
for(i=0; i<=1; i++)
for(j=0; j<=2; j++)
b[j][i]=a[i][j];
for(i=0; i<=2; i++)
{ for(j=0; j<=1; j++)
printf("%d ",b[i][j]);
printf("\n");
}
}
```

【4.65】参考答案：

```
main()
{ int i,count=0,a[11]={0,10,2,8,22,16,4,10,6,14,20};
while(1)
{ for(i=1; i<=10; i++)
a[i-1]=a[i-1]/2+a[i]/2;
a[10]=a[10]/2+a[0];
for(i=1; i<=10; i++)
```

```
if(a[i]%2==1) a[i]++;

for(i=1; i<10; i++)

if(a[i]!=a[i+1]) break;

if(i==10) break;

else

{ a[0]=0;

count++;

}

}

printf("count=%d number=%d\n",count,a[1]);

}
```

【4.66】参考答案：

```
main()

{ int i,j,s1=0,s2=1,a[5][5];

for(i=0; i<5; i++)

for(j=0; j<5; j++)

{ printf("%d %d: ",i,j);

scanf("%d",&a[i][j]);

}

for(i=0; i<5; i++)

{ for(j=0; j<5; j++)

printf("]",a[i][j]);
```

```
printf("\n");
}
j=0;
for(i=0; i<5; i++)
{ s1=s1+a[i][i];
if(i%2==0) s2=s2*a[i][i];
if(a[i][i]>a[j][j]) j=i;
}
printf("SUN=%d\nACCOM=%d\na[%d]=%d\n",s1,s2,j,a[j][j]);
}
```

【4.67】参考答案：

```
#include "stdio.h"
main()
{ int i,n=0,a[4]={0};
printf("Please enter a digit:");
for(i=0; i<4 && (a[i]=getchar())!='\n'; i++) ;
for(i=0; i<4; i++)
if(a[i]>=48&&a[i]<=57) a[i]=a[i]-48;
else if(a[i]>=65&&a[i]<=69) a[i]=a[i]-55;
else if(a[i]>=97&&a[i]<=102) a[i]=a[i]-87;
else printf("input Error!");
for(i=0; i<4; i++)
```

```
n=n*16+a[i];
printf("%d",n);
}
```

【4.68】参考答案：

```
main()
{ int i,n,k=16,a[16]={0};
printf("Please enter a digit:");
scanf("%d",&n);
while(n>0) /* 将十进制数转变为二进制数 */
{ a[--k]=n%2;
n=n/2;
}
for(i=0; i<16; i++)
printf("-",a[i]);
}
```

【4.69】参考答案：

```
#include main()
{ int i,j,m,s,k,a[100];
for(i=1; i<=100; i++ ) /* 寻找1000以内的完数 */ { m=i; s=0; k=0;
while(m>0) /* 寻找 i 的因子 */
{ for(j=1; jif(m%j==0)
{ s=s+j;
```

```
m=m/j;
a[k++]=j;
}
if(j>=m) break;
}
if(s!=0&&i==s+m)
{ a[k++]=m;
for(j=0; jprintf("M",a[j]);
printf("==M\n",i);
}
}
}
```

【4.70】参考答案：

```
main()
{ int i,j,k,n,m=1,r=1,a[2][100]={0}; printf("Please enter n:");
scanf("%d",&n);
for(i=0; i{ printf("a[%d]= ",i);
scanf("%d",&a[0][i]);
}
while(m<=n) /* m 记录已经登记过的数的个数 */
{ for(i=0; i{ if(a[1][i]!=0) /* 已登记过的数空过 */
continue;
```

```
k=i;
for(j=i; jif(a[1][j]==0 && a[0][j]
a[1][k]=r++;  /* 记录名次，r 为名次 */
m++;  /* 登记过的数增1 */
for(j=0; jif(a[1][j]==0 && a[0][j]==a[0][k])
{ a[1][j]=a[1][k];
m++;
}
break;
}
}
for(i=0; iprintf("a[%d]=%d, %d\n",i,a[0][i],a[1][i]);
}
```

【4.71】参考答案：

```
#include main()
{ int i,j,k=0,m=2,s,r=0,a[500];  printf("M ",m);
for(i=3; i<=2000; i++ )
{ for(j=2; j<=i-1; j++)
if(i%j==0) break;
if(j==i)
{ printf("M ", i );
a[k++]=i-m;
```

```
m=i;

}

}

for(i=0; i{ s=0;

for(j=i; j{ s=s+a[j];

if(s>=1898) break;

}

if(s==1898)

r++;

}

printf("\nresult=%d\n",r);

}
```

【4.72】分析：本问题的思路很多，我们介绍一种简单快速的算法。

首先求出三位数中不包含0且是某个整数平方的三位数，这样的三位数是不多的。然后将满足条件的三位数进行组合，使得所选出的三个三位数的九个数字没有重复。程序中可以将寻找满足条件三位数的过程和对该三位数进行数字分解的过程结合起来。

参考答案：

```
#include main( )

{ int a[20],num[20][3],b[10];  /* a: 存放满足条件的三位数 */

/* num：满足条件的三位数分解后得到的数字，b: 临时工作 */

int i,j,k,m,n,t,flag;

printf("The 3 squares with 3 different digits each are:\n");

for(j=0,i=11; i<=31; i++) /* 求出是平方数的三位数 */
```

```
if(i != 0) /* 若不是10的倍数，则分解三位数 */
{ k=i*i;  /* 分解该三位数中的每一个数字 */
num[j+1][0]=k/100;  /* 百位 */
num[j+1][1]=k/10;  /* 十位 */
num[j+1][2]=k;  /* 个位 */
if(!(num[j+1][0]==num[j+1][1] || num[j+1][0]==num[j+1][2]
|| num[j+1][1]==num[j+1][2]) )
/* 若分解的三位数字均不相等 */
a[++j]=k;  /* j:计数器，统计已找到的满足要求的三位数 */
}
for(i=1; i<=j-2; ++i ) /* 从满足条件的三位数中选出三个进行组合 */
{ b[1]=num[i][0];  /* 取第 i 个数的三位数字 */
b[2]=num[i][1];
b[3]=num[i][2];
for(t=i+1; t<=j-1; ++t)
{ b[4]=num[t][0];  /* 取第 t 个数的三位数字 */
b[5]=num[t][1];
b[6]=num[t][2];
for(flag=0, m=1; !flag&&m<=3; m++) /* flag:出现数字重复的标记 */
for(n=4; !flag&&n<=6; n++) /* 判断前两个数的数字是否有重复 */
if(b[m]==b[n]) flag=1;  /* flag=1:数字有重复 */
if(!flag)
```

```
for(k=t+1; k<=j; ++k)
{ b[7]=num[k][0];  /* 取第 k 个数的三位数字 */
b[8]=num[k][1];
b[9]=num[k][2];
/* 判断前两个数的数字是否与第三个数的数字重复 */
for(flag=0,m=1; !flag&&m<=6; m++)
for(n=7; !flag&&n<=9; n++)
if(b[m]==b[n]) flag=1;
if(!flag) /* 若均不重复则打印结果 */
printf("%d, %d, %d\n",a[i],a[t],a[k]);
}
}
}
}
```

【4.73】参考答案：

```
main()
{ int i,n,k,a[3],b[3];
for(i=248; i<=343; i++)
{ for(n=i,k=0; n>0; n/=7)
a[k++]=n%7;
for(n=i,k=0; n>0; n/=9)
b[k++]=n%9;
```

```
if(k==3)

for(n=0; nif(a[n]!=b[k-n-1])

break;

if(n==k)

printf("%d\n",i);

}

}
```

【4.74】参考答案：

```
#include int pos[101],div[101];

main ()

{ int m, n, i, j;

printf("Please input m/n(<0<=100):");

scanf("%d%d", &m,&n);

printf("%d/%d=0.", m, n);

for(i=1; i<=100; i++)

{ pos[m]=i;

m*=10;

div[i]=m/n;

m=m%n;

if(m==0)

{ for( j=1; j<=i; j++) printf("%d",div[j]);

break;
```

```
}
if(pos[m]!=0)
{ for( j=1; j<=i; j++) printf("%d",div[j]);
printf("\nloop: start=%d, end=%d",pos[m], i);
break;
}
}
printf("\n");
}
```

【4.75】参考答案：

```
#include "stdio.h"
int a[20],b[20];
main()
{ int t=0,*m,*n,*k,*j,z,i=0;
printf("Input number 1:");
do
{ a[++t]=getchar()-'0';
}while(a[t]!=-38);
printf("Input number 2:");
do
{ b[++i]=getchar()-'0';
}while(b[i]!=-38);
```

```
if(t>i)
{ m=a+t; n=b+i; j=a; k=b; z=i;
}
else
{ m=b+i; n=a+t; j=b; k=a; z=t;
}
while(m!=j)
{ (*(--n-1))+=(*(--m)+*n)/10;
*m=(*m+*n);
if (n==k+1 && *k!=1 ) break;
if (n==k+1 && *k)
{ n+=19; *(n-1)=1;
}
if (n>k+z && *(n-1)!=1) break;
}
while (*(j++)!=-38) printf("%d",*(j-1));
printf("\n");
}
```

【4.76】参考答案：

```
#include "stdio.h"
int a[20],b[20];
main()
```

```
{ int t=0,*m,*n,*k,*j,z,i=0;
printf("Input number 1:");
do
{ a[++t]=getchar()-'0';
}while(a[t]!=-38);
printf("Input number 2:");
do
{ b[++i]=getchar()-'0';
}while(b[i]!=-38);
if(t>i)
{ m=a+t; n=b+i; j=a; k=b; z=i;
}
else
{ m=b+i; n=a+t; j=b; k=a; z=t;
}
while(m!=j)
{ (*(--n-1))+=(*(--m)+*n)/10;
*m=(*m+*n);
if(n==k+1 && *k!=1 ) break;
if(n==k+1 && *k)
{ n+=19; *(n-1)=1;
}
```

```
if(n>k+z && *(n-1)!=1) break;
}
while(*(j++)!=-38) printf("%d",*(j-1));
printf("\n");
}
```

【4.77】参考答案：

```
#include "stdio.h"
int a[20],b[20],c[40];
main()
{ int t=0,*m,*n,*k,f,e=0,*j,i=0;
printf("Input number 1:");
do
{ a[++t]=getchar()-'0';
}while(a[t]!=-38);
printf("Input number 2:");
do
{ b[++i]=getchar()-'0';
}while(b[i]!=-38);
j=c;
for(m=a+t-1; m>=a+1; m--,e++)
{ j=c+e;
for(n=b+i-1; n>=b+1; n--)
```

```
{ f=*j+*m * *n;

*(j++)=f;

*j+=f/10;

}

}

while(j>=c) printf("%d",*(j--)) ;

printf("\n");

}
```

【4.78】这是一个使用数组解决较复杂问题的典型题目。

棋盘如左图所示，图中箭头表示一个棋子从[1、1]点跳到[2、3]点。为了下面叙述的方便，将 I、J 表示棋子起跳点的行、列号，X、Y 表示落子点的行、列号。

首先我们讨论如何从起跳点坐标求出可能的落子点的坐标。

从某点起跳，棋子最多可能有八个落子点，例如从 I=3、J=5点起跳，八个可能的落子点的坐标是[4、5]、[5、4]、[5、2]、[4、1]、[2、1]、[1、2]、[1、4]、[2、5]。将起落点的行列坐标分开考虑，则由起点的行坐标分别与下列八个数相加，就可得到可能的八个落子点的行坐标：1、2、2、1、-1、-2、-2、-1，将这八个数存入数组 b，即：

b[]={1,2,2,1,-1,-2,-2,-1},

落子点的行坐标 X 和起跳点行坐标有如下关系：

X=b[k]+I 1≤k≤8

如果由上式计算得到的落子点 X 的坐标值小于0或大于8,则表示落在了棋盘之外,应予舍弃。

同理得到起落点之间的列坐标关系数组是：

d[]={2,1,-1,-2,-2,-1,1,2}。

我们再讨论落子点的度数问题。对于棋盘中的某一点来说，周围最多有8个方向的棋子在这个点落子，把可能的落子数称为度数，棋盘上各点的度数如下图所示。

根据题意，一个点只能落子一次，所以落过子的点的度数应记为0，可跳向度数为0点的度数相应要减1。

根据上述数组，从一个起跳点出发，可能求出数个可以落子点的坐标，跳棋时到底确定落在这些点中的哪一个呢？我们确定一个原则是落在度数最少的点。如果可能落子点中有两个点的度数一样且都为是度数最少时，取后求出的点为落子点。因此，如果改变数组 b、d 中数的存放顺序，遇到两个度数最少点的先后顺序就要改变，整个跳棋路径就可改变。 2 3 4 4 4 4 3 2

3 4 6 6 6 6 4 3

4 6 8 8 8 8 6 4

4 6 8 8 8 8 6 4

4 6 8 8 8 8 6 4

4 6 8 8 8 8 6 4

3 4 6 6 6 6 4 3

2 3 4 4 4 4 3 2

在下面的程序中，将起落子行列关系的两个一维数组合并为一个二维数组，为了提高程序的可读性，不使用下标为0的数组元素。

参考程序：

```
int base[9][3]={0, 0, 0, /* 从起跳点求落脚点的基础系数数组 */

0, 1, 2,

0, 2, 1,

0, 2,-1,

0, 1,-2,

0,-1,-2,

0,-2,-1,
```

```
0,-2, 1,

0,-1, 2 };

main()

{ int a[9][9],object[9][9];

int i,j,k,p,x,y,m,n,cont;

int min,rm1,rm2,rm0=1;

for(cont=1; cont>0; )

{ for(i=0; i<=8; i++) /* 保存个点度数的数组 清零 */

for(j=0; j<=8; j++)

a[i][j]=0;

rm1=base[1][1]; /* 改变基础数组元素排列顺序 */

rm2=base[1][2];

base[1][1]=base[rm0][1];

base[1][2]=base[rm0][2];

base[rm0][1]= rm1;

base[rm0][2]= rm2;

for(i=1; i<=8; i++)

{ for(j=1; j<=8; j++) /* 计算各点度数存入数组a */

{ for(p=1; p<=8; p++)

{ x=i+base[p][1];

y=j+base[p][2];

if(x>=1&&x<=8&&y>=1&&y<=8)
```

```
a[x][y]++;
}
printf(" %d",a[i][j]);  /* 输出度数表 */
}
printf("\n");
}
printf("Please Input start position:line,colume=?\n");
scanf("%d,%d",&i,&j);  /* 输入起跳点坐标 */
for(k=1; k<=63; k++) /* 求棋盘上63个落步点 */
{ object[i][j]=k;  /* 跳步路径存入数组object */
min=10;
for(p=1; p<=8; p++) /* 求从当前起跳点出发的8个可能落点 */
{ x=i+base[p][1];
y=j+base[p][2];
if(x>=1&&x<=8&&y>=1&&y<=8) /* 求出的可能落点在棋盘内 */
if(a[x][y]!=0) /* 此点没有落过棋子 */
{ a[x][y]--;  /* 由于[i、j]点落过棋子，此点度数减1 */
if(min>a[x][y]) /* 判断当前可能点度数是否最小 */
{ min=a[x][y];  /* 保存可能最小度数点的度数 */
m=x;  /* 保存可能最小度数点的坐标 */
n=y;
}
```

```
}
}
a[i][j]=0;  /* 落过棋子的[i、j]点度数为零 */
i=m;  /* 已求出的最小度数点为下次搜寻的起跳点 */
j=n;
}
object[i][j] = 64 ;
for(i=1; i<=8; ++i) /* 输出跳步结果路径 */
{ for(j=1; j<=8; j++)
if(j==8) printf("-",object[i][j]);
else printf("- ",object[i][j]);
printf("\n");
if(i!=8) printf(" \n");  /* 每行输出8个数据 */
}
rm0%=8;  /* 放在基础数组第一位的元素循环变化 */
rm0++;  /* 基础数组下一元素放在第一位 */
printf("continue?(1 or 0)");
scanf("%d",&cont);
}
}
```

【4.79】分析：采用试探法求解。

如图所示，用 I、J 表示行、列坐标。

开始棋盘为空，对于第1个皇后先占用第一行即 I=1，先试探它占用第一列 J=1位置，则它所在的行、列和斜线方向都不可在放其它皇后了，用线将它们划掉。第2个皇后不能放在 J=1、2的位置，试 J=3。第2个皇后占用[2、3]后，它的行列和斜线方向也不可再放其它皇后。第3个皇后不能放在 J=1、2、3、4的位置，试 J=5。第4个皇后可以试位置[4、2]，第5个皇后试位置[5、4]。第6个皇后已经没有可放的位置（棋盘上所有格子都已占满），说明前面所放位置不对。

退回到前一个皇后5，释放它原来占用的位置[5、4]，改试空位置[5、8]。然后再前进到第6个皇后，此时仍无位置可放，退回到第5个皇后，它已没有其它位置可选择。进一步退回到第4个皇后释放位置[4、2]改试位置[4、7]，再前进到第5个皇后进行试探，如此继续，直到所有8个皇后都选择一个合适的位置，即可打印一个方案。

然后从第8个皇后开始，改试其它空位置，若没有可改选的空位置，则退回到第7个皇后改试其它位置，若也没有空位置可改，继续退，直到有另外的空位置可选的皇后。将它原来占用的位置释放，改占其它新位置，然后前进到下一个皇后进行试探，直到所有8个皇后都找到合适位置，又求出一个解，打印输出新方案。按此方法可得到92个方案。

参考答案：

```
#define NUM 8

int a[NUM+1];

main()

{ int number,i,k,flag,nonfinish=1,count=0;

i=1;

a[1]=1;

while(nonfinish)

{ while(nonfinish && i<=NUM)

{ for(flag=1,k=1; flag && k

if(a[k]==a[i]) flag=0;

for(k=1; flag && k

if((a[i]==a[k]-(k-i)) || (a[i]==a[k]+(k-i))) flag=0;
```

```
if(! flag)
{ if(a[i]==a[i-1])
{ i--;
if(i>1 && a[i]==NUM) a[i]=1;
else if(i==1 && a[i]==NUM) nonfinish=0;
else a[i]++;
}
else if(a[i]==NUM) a[i]=1;
else a[i]++;
}
else if( ++i<=NUM )
if(a[i-1]==NUM) a[i]=1;
else a[i]=a[i-1]+1;
}
if(nonfinish)
{ printf("\n-:",++count);
for(k=1; k<=NUM; k++)
printf(" %d",a[k]);
if(a[NUM-1]else a[NUM-1]=1;
i=NUM-1;
}
}
```

```
}
```

【4.80】参考答案：

```
double findy(float x)
{ if(x>=0 && x<2)
retuen(2.5-x);
else if(x>=2 && x<4)
return(2-1.5*(x-3)*(x-3));  else if(x>=4 && x<6)
return(x/2.0-1.5);
}
main()
{ float x;
printf("Please enter x:");
scanf("%f",&x);
if(x>=0 && x<6)
printf("f(x)=%f\n",findy(x));
else
printf("x is out!\n");
}
```

【4.81】注释：此程序采用模拟手工方式，对分数进行通分后比较分子的大小。

参考答案：

```
main( )
{ int i, j, k, l, m, n;
```

```
printf("Input two FENSHU :\n");
scanf("%d/%d,%d/%d", &i, &j, &k, &l);  /* 输入两个分数 */
m = zxgb(j,l)/j * i;  /* 求出第一个分数通分后的分子 */
n = zxgb(j,l)/l * k;  /* 求出第二个分数通分后的分子 */
if(m>n )
printf("%d/%d > %d/%d\n",i,j,k,l);  /* 比较分子的大小 */
else if(m==n)
printf("%d/%d = %d/%d\n",i,j,k,l);  /* 输出比较的结果 */
else printf("%d/%d < %d/%d\n",i,j,k,l);
}
zxgb(a,b)
int a,b;
{ long int c;
int d;
if(a
for( c=a*b;  b!=0;  ) /* 用辗转相除法求 a 和 b 的最大公约数 */
{ d=b;  b=a%b;  a=d;
}
return((int) c/a);  /* 返回最小公倍数 */
}
```

【4.82】参考答案：

```
main()
```

```
{ int a[5],i,t,k;

for (i=100; i<1000; i++)

{ for(t=0,k=1000; k>=10; t++)

{ a[t]=(i%k)/(k/10);

k/=10;

}

if(f(a[0])+f(a[1])+f(a[2])==i)

printf("%d ",i);

}

}

f(m)

int m;

{ int i=0,t=1;

while(++i<=m) t*=i;

return(t);

}
```

【4.83】分析：任取两个平方三位数 n 和 n1，将 n 从高向低分解为 a、b、c，将 n1从高到低分解为 x、y、z。判断 ax、by、cz 是否均为完全平方数。

参考答案：

```
main( )

{ void f( );

int i,t,a[3],b[3];
```

```
printf("The possible perfect squares combinations are:\n");
for(i=11; i<=31; i++) /* 穷举平方三位数的取值范围 */
for(t=11; t<=31; t++)
{ f(i*i,a); /* 分解平方三位数的各位，每位数字分别存入数组中 */
f(t*t,b);
if(sqrt(a[0]*10+b[0])==(int)sqrt(a[0]*10+b[0])
&& sqrt(a[1]*10+b[1])==(int)sqrt(a[1]*10+b[1])
&& sqrt(a[2]*10+b[2])==(int)sqrt(a[2]*10+b[2]))
/* 若三个新的数均是完全平方数 */
printf(" %d and %d\n",i*i,t*t); /* 则输出 */
}
}
void f(n,s) /* 分解三位数n的各位数字，将各个数字 */
int n, *s; /* 从高到低依次存入指针s所指向的数组中 */
{ int k;
for(k=1000; k>=10; s++)
{ *s = (n%k)/(k/10);
k /= 10;
}
}
```

【4.84】参考答案：

```
main()
```

```
{ int i,j,l,n,m,k,a[20][20];

printf("Please enter n,m=");

scanf("%d,%d",&n,&m);

for(i=0; ifor(j=0; j{ printf("a[%d][%d]=",i,j);

scanf("%d",&a[i][j]);

}

for(i=0; i{ for(j=0; jprintf("m",a[i][j]);

printf("\n");

}

for(i=0; i{ for(j=0,k=0; jif(a[i][j]>a[i][k]) k=j;  /* 找出该行最大值 */

for(l=0; lif(a[l][k]

if(l>=n) /* 没有比a[i][k]小的数，循环变量l就超过最大值 */

printf("Point:a[%d][%d]==%d",i,k,a[i][k]);

}

}
```

【4.85】分析：按题目的要求进行分析，数字1一定是放在第一行第一列的格中，数字6一定是放在第二行第三列的格中。在实现时可用一个一维数组表示，前三个元素表示第一行，后三个元素表示第二行。先根据原题初始化数组，再根据题目中填写数字的要求进行试探。

参考答案：

```
#include int count;  /* 计数器 */

main( )

{ static int a[ ]={1,2,3,4,5,6};  /* 初始化数组 */

printf("The possible table satisfied above conditions are:\n");
```

```
for(a[1]=a[0]+1; a[1]<=5; ++a[1]) /* a[1]必须大于 a[0] */

for(a[2]=a[1]+1; a[2]<=5; ++a[2]) /* a[2]必须大于 a[1] */

for(a[3]=a[0]+1; a[3]<=5; ++a[3]) /* 第二行的 a[3]必须大于 a[0] */

for(a[4]=a[1]>a[3]?a[1]+1:a[3]+1; a[4]<=5; ++a[4])

/* 第二行的 a[4]必须大于左侧 a[3]和上边 a[1] */

if(jud1(a))

print(a); /* 如果满足题意，打印结果 */

}

jud1(s) /* 判断数组中的数字是否有重复的 */

int s[ ];

{ int i,l;

for(l=1; l<4; l++)

for(i=l+1; i<5; ++i)

if(s[l]==s[i])

return(0); /* 若数组中的数字有重复的，返回 0 */

return(1);          /* 若数组中的数字没有重复的，返回 1 */

}

print(u)

int u[ ];

{ int k;

printf("\nNo.:%d", ++count);

for(k=0; k<6; k++)
```

```
if(k%3==0) /* 输出数组的前三个元素作为第一行 */

printf("\n %d ",u[k]);

else /* 输出数组的后三个元素作为第二行 */

printf("%d ",u[k]);

}
```

【4.86】参考答案：

```
#include "string.h"

strcmbn(a,b,c) /* 数组合并函数：将数组 a、b 合并到 */

char a[],b[],c[];

{ char tmp;

int i,j,k,m,n;

m=strlen(a);

n=strlen(b);

for(i=0; i{ for(j=i+1,k=i; jif(a[j]

tmp=a[i];  a[i]=a[k];  a[k]=tmp;

}

for(i=0; i{ for(j=i+1,k=i; jif(b[j]

tmp=b[i];  b[i]=b[k];  b[k]=tmp;

}

i=0; j=0; k=0;

while(iif(a[i]>b[j])

c[k++]=b[j++];  /* 将 a[i]、b[j]中的小者存入 c[k] */
```

```
else
{ c[k++]=a[i++];
if(a[i-1]==b[j]) j++;  /* 如果 a、b 当前元素相等，删掉一个 */
}
while(iwhile(jc[k]='\0';
}
```

【4.87】参考答案：

```
pxn(x,n)
float x;
int n;
{ if(n==0) return(1);
else if(n==1) return(x);
else return(((2*n-1)*x*pxn(x,n-1)-(n-1)*pxn(x,n-2))/2);
}
```

【4.88】参考答案：

```
#include "stdio.h"
strout(s)
char *s;
{ if(*s!='\0')
{ strout(s+1);  /* 递归调用 strout 函数，字符串首地址前移一个字符 */
putch(*s);  /* 输出字符串首地址所指向的字符 */
}
```

```
else return;  /* 遇到字符串结束标志结束递归调用 */

}
```

【4.89】参考答案：杨辉三角形中的数，正是(x+y)的N次方幂展开式中各项的系数。本题作为程序设计中具有代表性的题目，求解的方法很多（可以使用一维数组，也可以使用二维数组），前面我们给出用数组的答案，这里给出一种使用递归求解的方法。

从杨辉三角形的特点出发，可以总结出：

⑴ 第N行有N+1个值（设起始行为第0行）；

⑵ 对于第N行的第J个值： (N>=2)

当J=1或J=N+1时： 其值为1

当J!=1且J!=N+1时： 其值为第N-1行的第J-1个值与第N-1行第J个值之和。

将这些特点提炼成数学公式可表示为：

c(x,y) = 1 x=1 或 x=N+1

c(x,y) = c(x-1,y-1) + c(x-1,y) 其它

下面给出的程序就是根据以上递归的数学表达式编制的。

参考答案：

```
#include main( )

{ int i,j,n=13;

printf("N=");

while( n>12 )

scanf("%d", &n);  /* 最大输入值不能大于12 */

for(i=0; i<=n; i++) /* 控制输出N行 */

{ for(j=0; j<12-i; j++)

printf(" ");  /* 控制输出第i行前面的空格 */
```

```
for(j=1; j

printf("m", c(i,j)); /* 输出第 i 行的第 j 个值 */

printf("\n");

}

}

int c(x,y) /* 求杨辉三角形中第 x 行第 y 列的值 */

int x, y;

{ int z;

if((y==1)||(y==x+1))

return(1); /* 若为 x 行的第1或第 x+1列，则输出1 */

else /* 否则；其值为前一行中第 y-1列与第 y 列值之和 */

z = c(x-1,y-1) + c(x-1,y);

return(z);

}
```

【4.90】分析：整型数在计算机中就是以二进制形式存储的，此题的目的仅是为了学习递归程序的编程。

参考答案：

```
turn(n,a,k)

int n,a[ ],k;

{ if(n>0)

{ a[k]=n%2;

turn(n/2,a,k-1);
```

```
}

else return;

}

main()

{ int i,n,a[16]={0};

printf("\nPlease enter n:");

scanf("%d",&n);

turn(n,a,15);

for(i=0; i<16; i++)

printf("%d",a[i]);

}
```

【4.91】分析：分析题目，我们可以将题目进行抽象：在有放回的前提下，求全部从 m 个不同的元素中任取 n 个元素的排列。根据题目的含义，我们可以用整数0～m-1表示这 m 个不同的元素，将要生成的 n 个元素分为两部分：第一个元素和其它 n-1个元素。如果 n=1，即要从 m 个元素中任取1种，这样有 m 种不同得取法，我们可以直接使用循环完成。若 n>1，则可以知道，第一个元素一定有 m 种不同的取法，可以针对第一个元素 m 种不同取法种的1种，对后面的 n-1个元素进行同样的（递归）操作即可产生一种新的不同的排列。具体算法描述如下：

fun（指向第一个元素的指针，从 m 个元素中，取 n 个元素）

{ for （ i=0； i{ 确定第一个元素的选取方法：=i；

if （n>1） fun（指向下一个元素的指针，从 m 中，取 n-1个）

else 完成一种排列

}

}

根据以上算法分析可以得出程序。

参考答案：

```
#include int a[10];

fun( int *p, int m, int n ) /* 从m个元素中取n个存入数组p中 */

{ int i;  /* 用数0～m-1表示m个不同的元素 */

for( i=0; i{ *p = i;

if( n > 1 ) fun( p+1, m, n-1);

else print(p);

}

}

print( int *p )

{ int *q;

for( q=a; q<=p; q++ ) /* 输出结果，将整数转换为字母a起始的序列 */

printf("%c", 'a'+ *q);

printf("\t");

}

main( )

{ int m, n;

printf("\nEnter m n:");

scanf("%d%d", &m, &n);

fun( a, m, n);

}
```

【4.92】参考答案：

```
smmt ( char s[ ] ) /* 指针 s 指向字符串的第一个字符 */
{ char *p;
p=s;
while(*p!='\0') p++;
p--;  /* 指针 p 指向字符串的最后一个字符 */
if(p==s) return(1);  /* 两个指针指向同一个字符表示字符串对称 */
else
{ if(*s!=*p)
return(0);  /* 两个指针指向字符不等表示字符串不对称 */
else
{ *p='\0';
smmt(s+1);  /* 取掉首尾比较过的字符继续比较 */
}
}
}
```

【4.93】参考答案:

```
#include int n, r, flag;  /* flag：标志，=0：表示要另起一行 */
main( )
{ int s;
printf("Enter N,R:");
scanf("%d%d", &n, &r);
```

```
printf("combinations:\n");

flag=1;

combination (1,r);

}

combination ( s, j )

int s, j;  /* 从 s 开始选 j 个元素 */

{ int i,k;

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

{ if( flag )

for( k=0; kmerge(a，b，c，m) /* 数组合并函数：将长度为 m 的*/

int a[]，b[]，c[]，m;  /* 数组 a、b 合并到 c */

{ int i=0，j=0，k=0;

while(iif(a[i]>b[j])

c[k++]=b[j++];  /* 将 a[i]、*b[j]中的小 */

else c[k++]=a[i++];  /* 者存入 c[k] */

while(iwhile(j}

mergesort(w，n) /* 数组排序函数: 对长度为 n */

int w[]，n;  /* 的数组 w 排序 */

{ int i，t，ra[N];

for(i=1; iif(i==n)

{ if(n>2) /* 递归调用结束条件 */

{ mergesort (w，n/2);  /* 将数组 w 一分为二，递归调 */
```

```
mergesort (w+n/2, n/2);  /* 用 mergesort */

merge( w, w+n/2, ra, n/2 );  /* 将排序后的两数组重新合并 */

for(i=0; iw[i]=ra[i];

} else if(*w>*(w+1))

{ t=*w;  *w=*(w+1);  *(w+1)=t;

} }

else printf("Error:size of array is not a power of 2/n");

}

main( )

{ int i;

static int key[N]={4,3,1,81,45,8,0,4,-9,26,7,4,2,9,1,-1};

mergesort(key, N);

for(i=0; iprintf("%d ", key[i]);

printf("\n");

}
```

【4.95】参考答案：

```
#include "stdio.h"

main( )

{ char s[21],*p,*q;

gets(s);

p=s;

q=s;
```

```
while(*q!='\0') q++;
q-=2;
while(p
if(*p++ != *q--) /* 指针 p、q 同时向中间移动，比较对称的两个字符 */
{ printf("NO\n");
break;
}
if(p>=q)
printf("YES\n");
}
```

【4.96】参考答案：

```
strcut(s,m,k)
char s[ ];
int m,k;
{ char *p;
int i;
p=s+m;  /* 指针 p 指向要被删除的字符 */
while((*p=*(p+k))!='\0') /* p+k 指向要前移的字符 */
p++;
}
```

【4.97】参考答案：

```
strchg(s)
```

```
char *s;
{ char c,*p;
p=s;
while(*p!='\0') p++;
p--;
while(s
{ c=*s;
*s++=*p;
*p--=c;
}
}
```

【4.98】参考答案：

```
insert(s1,s2,ch)
char s1[],s2[],ch;
{ char *p,*q;
p=s1;
while(*p++!=ch) ;
while(*s2!='\0')
{ q=p;
while(*q!='\0') q++;
while(q>=p)
*(q+1)=*q--;
```

```
*++q=*s2++;

p++;

}

}
```

【4.99】参考答案：

```
strcnb(s1,s2)

char s1[],s2[];

{ char *p;

int i=1;

p=s1;

while(*p!='\0') p++;

while((*p++=*s2++)!='\0') ;  /* 将 s2接于 s1后面 */

p=s1;

while(*p!='\0') /* 扫描整个字符串 */

{ if(*p==' ') /* 当前字符是空格进行移位 */

{ while(*(p+i)==' ') i++;  /* 寻找当前字符后面的第一个非空格 */

if(*(p+i)!='\0')

{ *p=*(p+i);  /* 将非空格移于当前字符处 */

*(p+i)=' ';  /* 被移字符处换为空格 */

}

else break;  /* 寻找非空格时到字符串尾，移位过程结束 */

}
```

```
p++;
}
}
```

【4.100】参考答案：

```
#include "stdio.h"
struct strnum
{ int i;
char ch;
}
main( )
{ char c;
int i=0,k=0;
struct strnum s[100]={0,NULL};
while((c=getchar())!='\n')
{ for(i=0; s[i].i!=0; i++)
{ if(c==s[i].ch)
{ s[i].i++;
break;
}
}
if(s[i].i==0)
{ s[k].ch=c;
```

```
s[k++].i=1;

}

}

i=0;

while(s[i].i>0)

{ printf("%c=%d ",s[i].ch,s[i].i);

i++;

}

}
```

【4.101】分析：程序中函数 cmult 的形式参数是结构类型，函数 cmult 的返回值也是结构类型。在运行时，实参 za 和 zb 为两个结构变量，实参与形参结合时，将实参结构的值传递给形参结构，在函数计算完毕之后，结果存在结构变量 w 中，main 函数中将 cmult 返回的结构变量 w 的值存入到结构变量 z 中。这样通过函数间结构变量的传递和函数返回结构型的计算结果完成了两个复数相乘的操作。

参考答案：

```
#include "stdio.h"

struct complx

{ int real;  /* real 为复数的实部 */

int im;  /* im 为复数的虚部 */

};

main( )

{ static struct complx za = {3,4} /* 说明结构静态变量并初始化 */

static struct complx zb = {5,6};
```

```
struct complx x, y, z;
struct complx cmult(); /* 说明函数 cmult 的返回值类型是结构 complx 型 */
void cpr( );
z=cmult(za, zb);  /* 以结构变量调用 cmult 函数，返回值赋给结构变量 z */
cpr (za, zb, z);  /* 以结构变量调用 cpr 函数，输出计算结果 */
x.real = 10;  x.im = 20;
y.real = 30;  y.im = 40;  /* 下一组数据 */
z = cmult (x, y);
cpr (x, y, z);
}
struct complx cmult(za, zb) /* 计算复数 za×zb，函数的返回值为结构类型 */
struct complx za, zb;  /* 形式参数为结构类型 */
{ struct complx w;
w.real = za.real * zb.real - za.im * zb.im;
w.im = za.real * zb.im + za.im * zb.real;
return (w);  /* 返回计算结果，返回值的类型为结构 */
}
void cpr (za,zb,z) /* 输出复数 za×zb=z */
struct complx za, zb, z;  /* 形式参数为结构类型 */
{ printf ("(%d+%di)*(%d+%di)=", za.real, za.im, zb.real, zb.im);
printf ("(%d+%di)\n", z.real, z.im);
}
```

【4.102】参考答案一:

```
#include "stdio.h"
struct student
{ int n;
int mk;
};
main()
{ int i,j,k,count=0,no;
struct student stu[100],*s[100],*p;
printf("\nPlease enter mark(if mark<0 is end)\n");
for(i=0; i<100; i++)
{ printf("No.d==",i+1);
scanf("%d",&stu[i].mk);
s[i]=&stu[i];
stu[i].n=i+1;
if(stu[i].mk<=0) break;
for(j=0; j
for(k=j+1; k<=i; k++)
if(s[j]->mk~~mk)~~
~~{ p=s[j]; s[j]=s[k]; s[k]=p;~~
~~}~~
~~}~~
```

~~for(no=1,count=1,j=0; j~~

~~{ if(s[j]->mk > s[j+1]->mk)~~

~~{ printf("\nNo.===MM : ",no,s[j]->mk,count);~~

~~for(k=j-count+1; k<=j; k++)~~

~~{ printf("d ",s[k]->n);~~

~~if((k=(j-count))==0&&k!=j)~~

~~printf("\n ");~~

~~}~~

~~count=1;~~

~~no++;~~

~~}~~

~~else count++;~~

~~}~~

~~}~~

~~参考答案二：~~

~~#include "stdio.h"~~

~~#define N 5~~

~~struct student~~

~~{ int number;~~

~~int score;~~

~~int rank;~~

~~int no;~~

```
}stu[N];

main( )

{ int i, j, k, count, rank, score;

struct student temp;

for( i=1; i<=N; i++ )

{ printf("Enter N.o %d=", i );

scanf("%d%d", &temp.number, &temp.score );

for( j=i-1; j>0; j-- )

if( stu[j-1].score < temp.score )

stu[j]=stu[j-1];

else break;

stu[j]=temp;

}

stu[0].rank=1;

count = 1;

k = 0;

for( i=0; i{ score = stu[i].score;

rank = stu[i].rank;

if( stu[i+1].score == stu[i].score )

{ stu[i+1].rank = stu[i].rank;

count++;

}
```

~~else~~

~~{ for(j=0; jstu[k+j].no = count-j;~~

~~stu[i+1].rank = stu[i].rank+1;~~

~~count = 1;~~

~~k = i+1;~~

~~}~~

~~if(i==N-2)~~

~~for(j=0; jstu[k+j].no = count-j;~~

~~}~~

~~for(i=0; i{ rank = stu[i].rank;~~

~~count = stu[i].no;~~

~~printf ("\n= (=)-%d: ", rank, stu[i].score, count);~~

~~for(k=1; k<=count; k++)~~

~~if((k-1)%3 !=0)~~

~~printf("%d ", stu[i+k-1].number);~~

~~else printf ("\n %d ", stu[i+k-1].number);~~

~~i+=count-1;~~

~~}~~

~~}~~

~~**【4.103】**参考答案:~~

~~#include "stdio.h"~~

~~struct time~~

```
{ int hour;
int minute;
int second;
};
main()
{ struct time now;
printf("Please enter now time(HH,MM,SS)=\n");
scanf("%d,%d,%d",&now.hour,&now.minute,&now.second);
now.second++;
if(now.second==60)
{ now.second=0;
now.minute++;
}
if(now.minute==60)
{ now.minute=0;
now.hour++;
}
if(now.hour==24)
now.hour=0;
printf("\nNow is d:d:d",now.hour,now.minute,now.second);
}
```

【4.104】参考答案：

```
#include #define SIZE 3

struct student /* 定义结构 */

{ long num;

char name[10];

int age;

char address[10];

} stu[SIZE], out;

void fsave ( )

{ FILE *fp;

int i;

if((fp=fopen("student", "wb"))== NULL) /* 以二进制写方式打开文件 */

{ printf("Cannot open file.\n"); /* 打开文件的出错处理 */

exit(1); /* 出错后返回，停止运行 */

}

for(i=0; iif(fwrite(&stu[i],sizeof(struct student),1,fp) != 1 )

printf("File write error.\n"); /* 写过程中的出错处理 */

fclose(fp); /* 关闭文件 */

}

main()

{ FILE *fp;

int i;

for(i=0; i{ printf("Input student %d:",i+1);
```

```
scanf("%ld%s%d%s",
&stu[i].num,stu[i].name,&stu[i].age,stu[i].address);
}
fsave();  /* 调用函数保存学生信息 */
fp = fopen("student","rb");  /* 以二进制读方式打开数据文件 */
printf(" No.  Name  Age  Address\n");
while(fread(&out,sizeof(out),1,fp)) /* 以读数据块方式读入信息 */
printf ("%8ld %-10s %4d %-10s\n",
out.num,out.name,out.age,out.address);
fclose(fp);  /* 关闭文件 */
}
```

【4.105】参考答案：

```
#include main( )
{ FILE *fp;
char str[100], filename[15];
int i;
if((fp=fopen("test", "w")) == NULL)
{ printf("Cannot open the file.\n");
exit(0);
}
printf("Input a string:");
gets(str);  /* 读入一行字符串 */
```

```
for(i=0; str[i]&&i<100; i++) /* 处理该行中的每一个字符 */
{ if(str[i] >= 'a' && str[i] <= 'z') /* 若是小写字母 */
str[i] -= 'a'-'A'; /* 将小写字母转换为大写字母 */
fputc(str[i],fp); /* 将转换后的字符写入文件 */
}
fclose(fp); /* 关闭文件 */
fp=fopen( "test", "r"); /* 以读方式打开文本文件 */
fgets(str,100,fp); /* 从文件中读入一行字符串 */
printf("%s\n", str);
fclose(fp);
}
【4.106】参考答案:
#include "stdio.h" FILE *fp;
main( )
{ int c, d;
if((fp = fopen("d:\\tc\\test8.c","r")) == NULL)
exit(0);
while((c=fgetc(fp)) != EOF)
if( c=='/' ) /* 如果是字符注释的起始字符'/' */
if((d=fgetc(fp)) == '*') /* 则判断下一个字符是否为'*' */
in_comment(); /* 调用函数处理(删除)注释 */
else /* 否则原样输出读入的两个字符 */
```

~~{ putchar(c);~~

~~putchar(d);~~

~~}~~

~~else~~

~~if(c=='\'' || c=='\"') /* 判断是否是字符'或" */ echo_quote(c); /* 调用函数处理字符'或"包含的字符 */ else putchar(c); } in_comment()~~

~~{ int c, d;~~

~~c=fgetc(fp);~~

~~d=fgetc(fp);~~

~~while(c!='*' || d!='/')~~

~~{ /* 连续的两个字符不是 * 和 / 则继续处理注释 */~~

~~c = d;~~

~~d = fgetc(fp);~~

~~}~~

~~}~~

~~echo_quote (c)~~

~~int c; /* c中存放的是定界符'或" */~~

~~{ int d;~~

~~putchar(c);~~

~~while((d=fgetc(fp))!=c) /* 读入下一个字符判断是否是定界符c */~~

~~{ putchar(c); /* 当不是定界符c时继续循环 */~~

~~if(d=='\\') /* 若出现转义字符\ */~~

```
putchar( fgetc(fp)); /* 则下一个字符不论是何均原样输出 */
}
putchar(d);
}
```