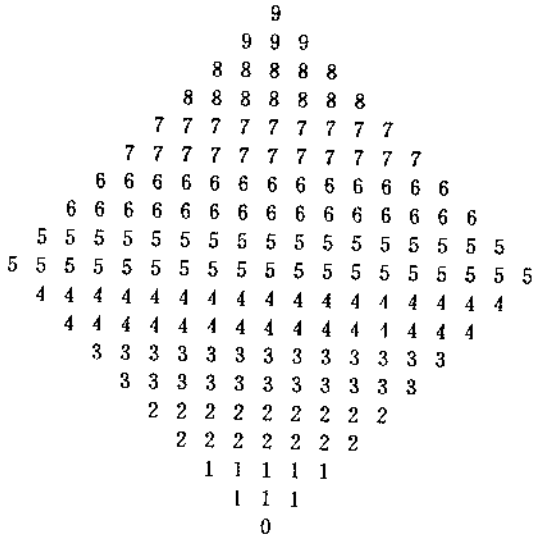
第一章　显示图案

计算机在正文方式下显示图案，题目变化多，具有趣味性。编程序时要灵活运用标准函数，使程序简洁、结构清楚，尽量使程序具有通用性。

【题1】 显示图案1

（1984年全国青少年程序设计竞赛高中组上机试题）



正文方式下显示图案用循环程序设计实现。通常要确定每行第一个字符的位置、每行显示字符的数目以及要显示的字符等。它们与图案的行号之间有一个函数关系。可列个表帮忙寻找函数关系。如果把图案显示在正中间，则函数关系如下：

行　　号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
第一个字符的位置	10	39	38	37	36	35	34	33	32	31	32	33	34	35	36	37	38	39	40
显示字符的数目	1	3	5	7	9	11	13	15	17	19	17	15	13	11	9	7	5	3	1
显示的字符	9	9	8	8	7	7	6	6	5	5	4	4	3	3	2	2	1	1	0

设行号用变量 i 表示，接下来的工作就是如何确定三个函数关系。

(1)确定第一个字符的位置与 i 的函数关系

变量 i 增大时，函数值先是减小，到过了 i=9 时，函数值增大，函数在 i=9 时有最小值 31，且关于 i=9 对称。绝对值函数也具有类似的性质，故用绝对值函数能方便地构造此函数关系，即 31+abs(i−9)，它用于表示第一个字符的位置。从而显示此字符前应留的空格数为 30+abs(i−9)。

(2)确定显示字符的数目与 i 的函数关系

变量 i 与显示字符的数目也是关于 i=9 的对称函数，i=9 时有最大值 19，每当 i 增大或减小 1 时，函数相应地增大或减小 2，所以函数关系可表示为 19−2 * abs(i−9)。它可转化为循环变量的初值和终值分别取作 0 和 18−2 * abs(i−9)。

(3)确定显示的字符与 i 的函数关系

每当变量 i 增大 2 时，函数值相应地减小 1，故函数关系可表示为 9−i div 2。

程序 1

```
program ex1_a;
var
  i, j : integer;
begin
  for i := 0 to 18 do
    begin
      write(' ':30 + abs(i - 9));
      for j := 0 to 18 - 2 * abs(i - 9) do
        write(9 - i div 2);
```

```
      writeln
    end
end.
```

运行程序 1 能显示题目中的图案。但要编写好程序，就不能满足于已编出了的程序，要不断地改进完善。由于图案上下对称，若将行号改为－9 至 9，则可简化程序。

程序 2

```
program ex1_b;
var
  i, j : integer;
begin
  for i := -9 to 9 do
    begin
      write(' ':30 + abs(i));
      for j := 0 to 18 - 2 * abs(i) do
        write(9 - (i + 9) div 2);
      writeln
    end
end.
```

一个好的程序应该有一定的通用性。引进常数标识符 n 用于表示菱形顶上的数字，可使程序更清楚，且具有通用性。把 n 改为小于 9 的正整数时，可显示类似的图案。比如，在 n＝4 时显示图案为：

```
          4
        4 4 4
      3 3 3 3 3
    3 3 3 3 3 3 3
  2 2 2 2 2 2 2 2 2
    2 2 2 2 2 2 2
      1 1 1 1 1
        1 1 1
          0
```

程序 3 中引进变量 k，用于表示第 i 行第一个字符离纵向中间对称轴的距离。

程序 3

```
program ex1_c;
const
  n = 9;
var
  i, j, k : integer;
begin
  for i := -n to n do
    begin
      k := n - abs(i);
      write(' ':39 - k);
      for j := 0 to 2 * k do
        write(n - (i + n) div 2);
      writeln
    end
end.
```

进一步增大题目的难度，改为显示空心菱形，空心菱形边的厚度为 m 个字符。可用“填空法”编程序，即在空的地方显示空格。比如 n=6，m=3 时显示图案为：

```
          6
        6 6 6
      5 5 5 5 5
    5 5 5   5 5 5
  4 4 4       4 4 4
4 4 4           4 4 4
3 3 3             3 3 3
4 4 4           4 4 4
  4 4 4       4 4 4
    5 5 5   5 5 5
      5 5 5 5 5
        6 6 6
          6
```

程序 4

```
program ex1_d;
```

```
const
  n = 9; m = 3;
var
  i, j, k : integer;
begin
  for i := -n to n do
    begin
      k := n - abs(i);
      write(' ':39 - k);
      for j := -k to k do
        if abs(j) >k - m
          then write(n - (i + n) div 2)
          else write(' ');
      writeln
    end
end.
```

空心图形也可用“跳跃法”编程序，即遇到空心部分，跳到后面有字符的地方。这时要用标准库单元 crt 中的预定义过程语句 gotoxy 来给光标定位。

程序 5

```
program ex1_e;
uses
  crt;
const
  n = 9; m = 3;
var
  i, j, k : integer;
begin
  clrscr;
  for i := -n to n do
    begin
      k := n - abs(i);
```

```
    for j := -k to k do
      begin
        gotoxy(40 + j, 12 + i);
        write(n - (i + n) div 2);
        if (k>= m) and (j = m - k - 1) then
          j := k    m
      end
  end
end.
```

【题 2】 显示图案 2

```
1   0   2   0   3   0   4   0   5
    1   0   2   0   3   0   4
        1   0   2   0   3
            1   0   2
                1
              -1   0  -2
          -1   0  -2   0  -3
      -1   0  -2   0  -3   0  -4
  -1   0  -2   0  -3   0  -4   0  -5
```

本题的难点是怎样确定要显示的数据，在同一行中数据是不同的，且有些数据是负的。可采用逐步深入的方法解决。先把有显示的数据简化为全是 1，编程序显示出形状相同的图案：

```
1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1
    1 1 1 1 1
      1 1 1
        1
      1 1 1
    1 1 1 1 1
  1 1 1 1 1 1 1 1 1
```

这个图案也是上下对称的，行号变化可从－4 到 4。可用上题类似的方法列出行号与第一个字符的位置以及行号与该行显示数据的数目之间的函数关系：

行　　号	−4	3	−2	−1	0	1	2	3	4
第一个字符的位置	27	30	33	36	39	36	33	30	27
显示数据的数目	9	7	5	3	1	3	5	7	9

行号用变量 i 表示，第一个字符的位置为 39－3＊abs(i)，显示数据的数目为 2＊abs(i)＋1. 引进常数标识符 n＝4，改变 n 的大小可显示大小不同的类似图案。

程序 1

```
program ex2_a;
const
  n = 4;
var
  i, j : integer;
begin
  for i := -n to n do
    begin
      write(' ':38 - 3 * abs(i));
      for j := 0 to 2 * abs(i) do
        write(1:3);
      writeln
    end
end.
```

运行程序 1 后显示如上所述的全 1 图案。

接下来研究怎样在同一行中显示不同的数据，目前暂且不考虑数据的负号，即欲显示下述图案：

```
      1  0  2  0  3  0  4  0  5
         1  0  2  0  3  0  4
            1  0  2  0  3
               1  0  2
                  1
               1  0  2
            1  0  2  0  3
         1  0  2  0  3  0  4
      1  0  2  0  3  0  4  0  5
```

由于每一行第一个数据都是1，且后面的数据变化情况相同，因此可列出相对序列号与数据的函数关系：

序列号	0	1	2	3	4	5	6	7	8
数　据	1	0	2	0	3	0	4	0	5

序列号用变量j表示，j为偶数时，数据为j/2+1，j为奇数时，数据为0.只要找到一个表达式，它在j为偶数时为1，奇数时为0，就能把上面的两种情况统一用一个表达式表示。(j+1) mod 2就是在j为偶数时为1，奇数时为0，从而，表达式((j+1) mod 2) * (j/2+1)就能正确表示序列号与数据的函数关系。

程序2

```
program ex2_b;
const
  n = 4;
var
  i, j : integer;
begin
  for i := -n to n do
    begin
      write(' ':38 - 3 * abs(i));
      for j := 0 to 2 * abs(i) do
        write(((j + 1) mod 2) * (j / 2 + 1):3:0);
      writeln
    end
end.
```

运行程序2后。将显示上述不带负号且数据不同的图案。

最后要解决数据的符号问题，即欲显示本题一开始给出的图案。从图案可见，当i<=0时数据非负，i>0时数据非正。故可构造一个函数，令其在i<=0时为1，在i>0时为-1，将它与前面程序中显示的数据相乘，就能得到题目中要显示的数据。此函数可用abs(0.5- i)/(0.5-i)来实现，而程序3即成为完成本题的最终形式。

程序 3

```
program ex2_c;
const
  n = 4;
var
  i, j : integer;
begin
  for i := -n to n do
    begin
      write(' ':38 - 3 * abs(i));
      for j := 0 to 2 * abs(i) do
        write(abs(0.5-i)/(0.5-i) * ((j+1)mod 2) * (j/2+1):3:0);
      writeln
    end
end.
```

【题 3】 显示图案 3

```
                0
              1   2
            3   4   5
          6   7   8   9
        A   B   C   D   E
      F   G   H   I   J   K
    L   M   N   O   P   Q   R
  S   T   U   V   W   X   Y   Z
```

本题要显示的图案中有数字和字母，可统一作为字符处理。使用与前两题类似的方法，可以用二重循环来实现。主要问题是确定要显示的字符。

字符“0”到“9”的 ASCII 码是 48 到 57，字符“A”到“Z”的 ASCII 码是 65 到 90，关键是确定要显示字符的 ASCII 码。有两种方法，一种方法是用变量来构造要显示字符的 ASCII 码，另一种方法是用数学的方法计算要显示字符的 ASCII 码。首先用一种普通的方法。变量 s 的值首

先是"0"的 ASCII 码 48，然后每次加 1，当 s 的值为"9"的 ASCII 码 57 后，用 if 语句来个"跳跃"，变为"A"的 ASCII 码 65。

程序 1

```
program ex3_a;
const
  n = 7;
var
  i, j, s : byte;
begin
  s := 47;
  for i := 0 to n do
    begin
      write(' ':39 - i);
      for j := 0 to i do
        begin
          s := s + 1;
          if s = 58 then
            s := 65;
          write(chr(s):2)
        end;
      writeln
    end
end.
```

程序可作一些小的改进。只要确定了每行第一个字符的 ASCII 码后，就能很方便地确定本行中各个字符的 ASCII 码。

程序 2

```
program ex3_b;
const
  n = 7;
var
  i, j, s : byte;
```

```
begin
  s := 48;
  for i := 0 to n do
    begin
      s := s + i;
      if s = 58 then
        s := 65;
      write(' ':39 - i);
      for j := s to s + i do
        write(chr(j):2);
      writeln
    end
end.
```

用数学方法能方便地计算出每行第一个字符的 ASCII 码。

程序 3

```
program ex3_c;
const
  n = 7;
var
  i, j, s : byte;
begin
  for i := 0 to n do
    begin
      write(' ':39 - i);
      s := i * (i + 1) div 2 + 48;
      if i >3 then
        s := s + 7;
      for j := s to s + i do
        write(chr(j):2);
      writeln
    end
end.
```

前面三个程序都是对行号和序列号进行二重循环，设法确定要显示字符的 ASCII 码。也可对要显示的字符进行一重循环，这时的关键是怎样确定换行。可采用构造的方法来确定行号和序列号，在行号与序列号相等时换行。

程序 4

```
program ex3_d;
var
  i, j : byte;
  c : char;
begin
  i := 0; j := 0;
  for c := '0' to 'Z' do
    begin
      if c = chr(ord('9') + 1) then
        c := 'A';
      if i = j then
        begin
          writeln;
          write(' ':39 - i);
          j := 0;
          i := i + 1;
        end;
      j := j + 1;
      write(c:2)
    end;
  writeln
end.
```

把没有规律的字符放在一个字符串中，就有规律了，从而能方便地进行各种处理。可以把要显示的 36 个字符放在一个字符串中，然后分段截取显示。程序 5 中用到了字符串预定义标准函数 copy。

程序 5

```
program ex3_e;
const
  n = 8;
var
  i : byte;
  c : char;
  s : string[72];
begin
  s := '';
  for c := '0' to '9' do
    s := s + c + ' ';
  for c := 'A' to 'Z' do
    s := s + c + ' ';
  for i := 1 to n do
    writeln(copy(s, (i - 1) * i + 1, 2 * i):40 + i)
end.
```

【题 4】 显示图案 4

不用数组显示图案：

```
1  2  4  7  1  6  2
   3  5  8  2  7  3
      6  9  3  8  4
         0  4  9  5
            5  0  6
               1  7
                  8
```

本题的程序也要考虑具有通用性，只要改变常数标识符 n 的值，便能显示类似下述形式的图案(n = 5)：

```
1  2  4  7  1
   3  5  8  2
      6  9  3
         0  4
            5
```

本题难点也是确定要显示的数。为了便于对图案中的数进行分析，把欲显示的图案改为：

```
 1  2  4  7 11 16 22
    3  5  8 12 17 23
       6  9 13 18 24
         10 14 19 25
            15 20 26
               21 27
                  28
```

而实际上要显示的数为上述图案中数的个位。从上述图案很容易发现，第 i+1 列与第 i 列同一行中数的差都是 i. 接下来要确定每一行的第一个数。第二行与第一行第一个数的差是 2，第三行与第二行第一个数的差是 3，依此类推，第 i 行与第 i−1 行第一个数的差应是 i. 下面程序中作为变量 s 初始值的变量 m 为每行第一个数。

程序 1

```
program ex4_a;
const
  n = 7;
var
  i, j, m, s : byte;
begin
  m := 0;
  for i := 1 to n do
    begin
      m := m + i;
      s := m;
      write(' ':2 * i);
      for j := i to n do
        begin
          write(s mod 10:2);
          s := s + j
```

```
        end;
      writeln
    end
end.
```

按第 i+1 列与第 i 列同一行中数的差是 i 的规律，把图案改为：

```
1 2 4 7 1 6 2
2 3 5 8 2 7 3
3 4 6 9 3 8 4
4 5 7 0 4 9 5
5 6 8 1 5 0 6
6 7 9 2 6 1 7
7 8 0 3 7 2 8
```

这时，每行第一个数就很容易确定了。显示图案中的数时，若是下三角形中的数，改为显示空格。

程序 2

```
program ex4_b;
const
  n = 7;
var
  i, j, s : byte;
begin
  for i := 1 to n do
    begin
      s := i;
      for j := 1 to n do
        begin
        if i > j
          then write(' ')
          else write(s:2);
        s := (s + j) mod 10
      end;
    writeln
```

```
  end
end.
```

前面两个程序都是采用计算机编程时常用的构造数据的方法。也可采用数学的方法来计算第 i 行、第 j 列的数。第 i 行中的数(不考虑求模运算)为 i,i+1,i+1+2,i+1+2+3,…,i+1+2+…+(n−1),所以第 j 列的数为 i+j*(j−1)/2。

程序 3

```
program ex4_c;
const
  n = 7;
var
  i, j : byte;
begin
  for i := 1 to n do
    begin
      for j := 1 to n do
        if i > j
          then write(' ')
          else write((i + j * (j - 1) div 2) mod 10:2);
    writeln
  end
end.
```

二重循环可方便地改为一重循环。

程序 4

```
program ex4_d;
const
  n = 7;
var
  i, j, m : byte;
begin
  for m := 0 to n * n-1 do
```

```
    begin
      i := m div n;
      j := m mod n;
      if i>j
        then write(' ')
        else write((1 + i + j * (j + 1) div 2) mod 10:2);
      if j = n - 1 then
        writeln
    end
end.
```