

C 语言 趣味程序百例精解

陈朔鹰 陈英 主编

吴鹤龄 审



北京理工大学出版社

北京理工大学出版社

TP312.49

12-48

C 语言趣味程序百例精解

陈翔鹰 陈 英 主编
贾 川 张 忠 编
查 礼 杜 桐
吴鹤龄 审



北京理工大学出版社

(京)新登字 149 号

内 容 简 介

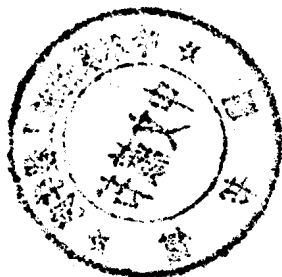
程序设计在某种意义上是一项复杂的工作,但同时又充满着探索 and 追求的乐趣。本书的作者在长期教学实践中将各种程序的设计技巧和方法融于趣味问题之中,通过对各种饶有趣味的问题的讨论和求解,使学生在轻松、愉快的气氛中理解和掌握程序设计的奥妙,从而达到事半功倍的学习效果,本书就是作者这一尝试的结晶。

书中全部例题均有问题分析与算法设计,所有程序均有详细的注释,并附运行结果,便于读者自学。

本书可作为大专院校计算机语言课程的辅助教学及习题参考用书,也可供初步掌握 C 语言的读者进一步学习提高编程技巧。

需要购买该书程序软盘的读者,请与北京理工大学出版软件室联系。

JS202 / 13



C 语言趣味程序百例精解

陈翔鹰 陈 英 主编

吴鹤龄 审

北京理工大学出版社出版发行

各地新华书店经售

北京理工大学印刷厂印刷

*

787×1092 毫米 16 开本 9.5 印张 226 千字

1994 年 3 月第一版 1994 年 3 月第一次印刷

ISBN 7-81013-573-2/TP·55

印数:1-10100 册 定价:7.50 元

前 言

程序设计在某种意义上来说是一项既简单又复杂的工作；它需要付出艰苦的劳动，同时又充满着探索和追求的乐趣。人们在不断地向各种问题挑战的活动中，增长了才干，开阔了眼界，锻炼了思维，同时也培养了克服困难的毅力和精神。

学习掌握程序设计的技术和方法，往往是枯燥乏味的。作为一种尝试，我们在教学中将各种程序设计的技术和方法融于趣味问题之中，通过对多种饶有趣味的问题的讨论和求解，使读者在轻松、愉快的气氛中理解和探索程序设计的奥妙，从而达到事半功倍的学习效果。本书就是这一尝试的结晶。

书中所选择的例题有不少是与数学有关的，完全可以用数学的方法求解，或者用数学的方法证明。但本书不是一本趣味数学方面的书，它的立足点是在程序设计上，因此，对书中绝大多数题目，仅从如何进行程序设计这一角度来进行讨论，以突出重点。感兴趣的读者可以从数学的角度对书中的问题进行讨论。

在本书中，我们将依据题目的性质和特点，大体上按以下次序展开讨论：

- 问题
- 问题分析与算法设计
- 程序与程序注释
- 程序运行结果
- 问题的进一步讨论
- 思考题

由于题目不同，对上述内容的取舍和侧重可能有所不同，但目的只有一个：通过例题剖析开阔思路。

在“问题分析与算法设计”部分，着重讨论算法的主要思路，而不涉及实现算法过程中的许多细节，以免分散读者对算法总体思想的注意力。算法的实现细节在“程序与程序注释”中进行说明。在这一部分中，对关键的语句、算法和使用的编程技巧，都在对应的程序语句后面给出尽可能简洁、清楚的注释说明。将程序与注释说明放在一起，可以帮助读者在阅读程序的同时十分方便地理解语句的含义，同时也是符合软件工程要求的一种好的编程习惯。

书中给出的算法未必是最优的，读者完全可以进一步改进和优化。

本书是在教学实践的基础上编写的。北京理工大学计算机科学工程系吴鹤龄教授认真审阅了全书，提出了许多宝贵意见和建议，同时还提供了书中使用的部分问题。特在此表示衷心的感谢。

本书由北京理工大学计算机科学工程系陈朔鹰与陈英共同主编，贾川、张忠、查礼和杜桐同志参加编写。由于我们水平有限，错误之处难免，敬请广大读者指正。

编著者

一九九三年十二月

目 录

第一章 最简单的问题与算法	1
1 绘制余弦曲线	1
2 绘制余弦曲线和直线	2
3 绘制圆	3
4 歌星大奖赛	4
5 求最大约数	5
6 高次方数的尾数	6
7 阶乘尾数零的个数	6
8 借书方案知多少	7
9 杨辉三角形	8
10 数制转换	10
第二章 生活中的数学问题	12
11 打鱼还是晒网	12
12 抓交通肇事犯	13
13 该存多少钱	14
14 怎样存钱利最大	15
15 捕鱼和分鱼	16
16 出售金鱼	17
17 平分七筐鱼	18
第三章 整数趣题	21
18 有限 5 位数	21
19 8 除不尽的自然数	21
20 一个奇异的三位数	22
21 4 位反序数	22
22 求车速	23
23 由两个平方三位数获得三个平方二位数	24
24 阿姆斯特朗数	25
25 完全数	25
26 亲密数	26
27 自守数	26
28 回文数	28
29 求具有 $abcd = (ab + cd)^2$ 性质的四位数	29
第四章 素数的家族	30
30 求素数	30
31 哥德巴赫猜想	31
32 可逆素数	32

33	回文素数	33
34	要发就发	34
35	素数幻方	36
第五章	不定方程求整数解	44
36	百钱百鸡问题	44
37	爱因斯坦的数学题	45
38	换分币	45
39	年龄几何	46
40	三色球问题	47
41	马克思手稿中的数学题	47
第六章	分数趣题	49
42	最大公约数和最小公倍数	49
43	分数比较	50
44	分数之和	51
45	将真分数分解为埃及分数	52
46	列出真分数序列	53
47	计算分数的精确值	54
第七章	逻辑推理与判断	56
48	新娘和新郎	56
49	委派任务	57
50	谁在说谎	58
51	谁是窃贼	59
52	黑与白	60
53	谜语博士的难题(1)	61
54	谜语博士的难题(2)	62
55	哪个大夫哪天值班	64
56	区分旅客国籍	66
57	谁家孩子跑最慢	68
第八章	数字 0 到 9 的奇妙变幻	70
58	拉丁方	70
59	填表格	71
60	1~9 分成 1:2:3 的三个 3 位数	72
61	1~9 组成三个 3 位的平方数	73
62	由 8 个整数形成奇特的立方体	75
63	减式还原	77
64	乘式还原(1)	78
65	乘式还原(2)	79
66	除式还原(1)	82
67	除式还原(2)	83
68	九位累进可除数	85
第九章	数的变幻	88
69	魔术师的猜牌术(1)	88
70	魔术师的猜牌术(2)	89

71	约瑟夫问题	90
72	邮票组合	91
73	和数能表示 1~23 的 5 个正整数	92
74	可称 1~40 磅的 4 块砝码	93
75	10 个小孩分糖果	94
76	小明买书	95
77	波瓦松的分酒趣题	97
第十章	定理与猜想	99
78	求 π 的近似值(1)	99
79	求 π 的近似值(2)	99
80	奇数平方的一个有趣性质	100
81	角谷猜想	101
82	四方定理	102
83	卡布列克常数	103
84	尼科彻斯定理	105
85	回文数的形成	106
第十一章	智力游戏	108
86	自动发牌	108
87	黑白子交换	110
88	常胜将军	113
89	抢 30	114
90	搬山游戏	117
91	人机猜数游戏(1)	119
92	人机猜数游戏(2)	121
93	汉诺塔	125
第十二章	其它趣味程序	128
94	兔子产子	128
95	将阿拉伯数翻译为罗马数字	129
96	选美比赛	130
97	满足特异条件的数列	131
98	八后问题	133
99	超长正整数的加法	135
100	数字移动	139

第一章 最简单的问题与算法

作为趣味程序设计的入门,我们先来涉猎一些简单的例子。通过这些例子,让读者体验一下程序设计的乐趣。

1. 绘制余弦曲线

在屏幕上用“*”显示 $0^\circ \sim 360^\circ$ 的余弦函数 $\cos(x)$ 曲线。

* 问题分析与算法设计

如果在程序中使用数组,这个问题十分简单。但若规定不能使用数组,问题就变得不容易了。

关键在于余弦曲线在 $0^\circ \sim 360^\circ$ 的区间内,一行中要显示两个点;而对一般的显示器来说,只能按行输出,即:输出第一行信息后,只能向下一行输出,不能再返回到上一行。为了获得本题要求的图形就必须在一行中一次顺序输出两个“*”。

为了同时得到余弦函数 $\cos(x)$ 图形在一行上的两个点,考虑利用 $\cos(x)$ 的左右对称性。将屏幕的行方向定义为 x ,列方向定义为 y ,则 $0^\circ \sim 180^\circ$ 的图形与 $180^\circ \sim 360^\circ$ 的图形是左右对称的。若定义图形的总宽度为 62 列,计算出 x 行 $0^\circ \sim 180^\circ$ 时 y 点的坐标 m ,那么在同一行与之对称的 $180^\circ \sim 360^\circ$ 的 y 点的坐标就应为 $62-m$ 。程序中利用反余弦函数 acos 计算坐标 (x,y) 的对应关系。

使用这种方法编出的程序短小精练,体现了一定的技巧。

* 程序说明与注释

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

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

```
main()
```

```
{ double y;
```

```
int x, m;
```

```
for (y=1; y>=-1; y-=0.1) { /* y 为列方向,值从 1 到 -1,步长为 0.1 */
```

```
    m=acos(y)*10; /* 计算出 y 对应的弧度 m,乘 10 为图形放大倍数 */
```

```
    for (x=1; x<m; x++) printf(" ");
```

```
    printf(" * ");
```

```
/* 控制打印左侧的 * 号 */
```

```
    for (; x<62-m; x++) printf(" ");
```

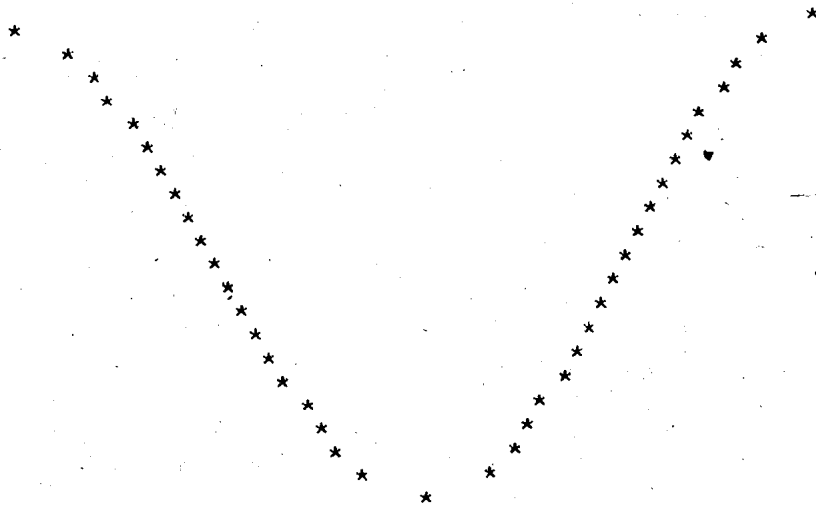
```
    printf(" * \n");
```

```
/* 控制打印同一行中对称的右侧 * 号 */
```

```
}
```

```
}
```

* 运行结果



* 思考题

如何实现用“*”显示 $0^\circ \sim 360^\circ$ 的 $\sin(x)$ 曲线。

2. 绘制余弦曲线和直线

在屏幕上显示 $0^\circ \sim 360^\circ$ 的 $\cos(x)$ 曲线与直线 $f(x) = 45 * (y-1) + 31$ 的迭加图形。其中 $\cos(x)$ 图形用“*”表示, $f(x)$ 用“+”表示, 在两个图形的交点处则用 $f(x)$ 图形的符号。

* 问题分析与算法设计

本题可以在上题的基础上进行修改。图形迭加的关键是要在分别计算出同一行中两个图形的列方向点坐标后, 正确判断相互的位置关系。为此, 可以先判定图形的交点, 再分别控制打印两个不同的图形。

* 程序说明与注释

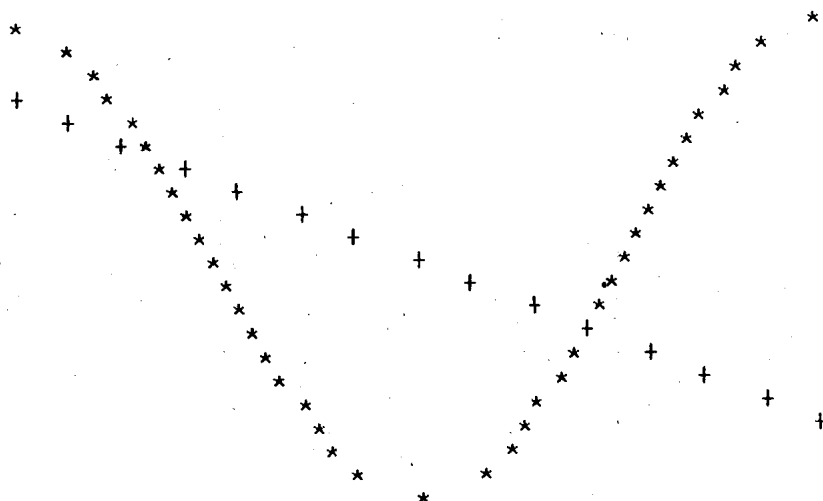
```
#include <stdio.h>
#include <math.h>
main()
{ double y;
  int x, m, n, yy;
  for (yy=0; yy<=20; yy++) {
      /* 对于每一个 y 坐标进行计算并在一行中打印图形 */
      y = 0.1 * yy;
      m = acos(1-y) * 10;
      n = 45 * (y-1) + 31;
      for (x=0; x<=62; x++)
          if (x==m && x==n)
              printf("+");
          else if (x==n)
              printf("+");
          /* y: 屏幕行方向坐标 */
          /* m: cos(x) 曲线上 y 点对应的屏幕列坐标 */
          /* n: 直线上 y 点对应的列坐标 */
          /* x: 屏幕列方向坐标 */
          /* 直线与 cos(x) 相交时打 '+' */
          /* 打印不相交时的直线图形 */
  }
```

```

else if (x==m || x==62-m) printf(" *"); /* 打印不相交时的 cos(x)图形 */
else
    printf(" "); /* 其它情况打印空格 */
    printf("\n");
}
}

```

* 运行结果



* 思考题

如何实现 $\sin(x)$ 曲线与 $\cos(x)$ 曲线图形的同时显示。

3. 绘制圆

在屏幕上用“*”画一个空心的圆。

* 问题分析与算法设计

打印圆可利用图形的左右对称性。根据圆的方程：

$$R \times R = X \times X + Y \times Y$$

可以计算出圆上每一点行和列的对应关系。

* 程序说明与注释

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

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

```
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。2.5 是屏幕纵横比调节系数,因为屏幕的
行距大于列距,不进行调节显示出来的将是椭圆 */
```

```
for (x=1; x<30-m; x++) printf(" ");
```

```
/* 图形左侧空白控制 */
```

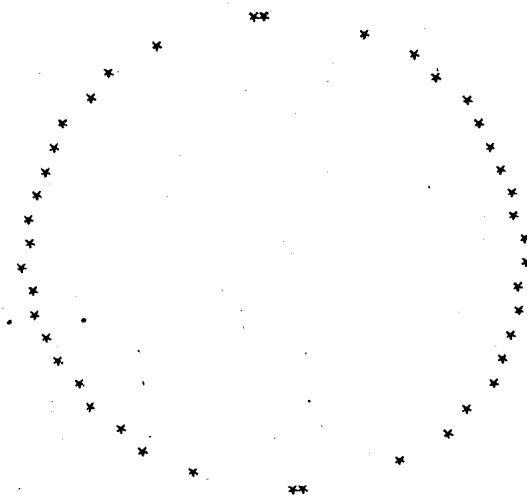
```

printf(" * ");
for (; x<30+m; x++) printf(" ");
printf(" * \n");
}
}

```

/* 圆的左侧 */
/* 图形的空心部分控制 */
/* 圆的右侧 */

* 运行结果



* 思考题

实现函数 $y=x^2$ 的图形与圆的图形的叠加显示。

4. 歌星大奖赛

在歌星大奖赛中,有 10 个评委为参赛的选手打分,分数为 1 到 100 分。选手最后得分为:去掉一个最高分和一个最低分后其余 8 个分数的平均值。请编写一个程序实现。

* 问题分析与算法设计

这个问题的算法十分简单,但要注意在程序中判断最大、最小值的变量是如何赋初值的。

* 程序说明与注释

```

main()
{
    int integer, i, max, min, sum;
    max=-32768; /* 先假设当前的最大值 max 为 C 语言整型数的最小值 */
    min=32767; /* 先假设当前的最小值 min 为 C 语言整型数的最大值 */
    sum=0; /* 将求累加和变量的初值置为 0 */
    for (i=1; i<=10; i++){
        printf("Input number %d=", i);
        scanf("%d", &integer); /* 输入评委的评分 */
        sum += integer; /* 计算总分 */
        if (integer>max) max=integer; /* 通过比较筛选出其中的最高分 */
        if (integer<min) min=integer; /* 通过比较筛选出其中的最低分 */
    }
    printf("Canceled max score: %d\nCanceled min score: %d\n", max, min);
}

```

```
printf(" Average score: %d\n", (sum-max-min)/8);
```

```
/* 输出结果 */
```

* 运行结果

```
Input number 1=90
Input number 2=91
Input number 3=93
Input number 4=94
Input number 5=90
Input number 6=99
Input number 7=97
Input number 8=92
Input number 9=91
Input number 10=95
Canceled max score: 99
Canceled min score: 90
Average score: 92
```

* 思考题

题目条件不变,但考虑同时对评委的评分进行裁判,即在 10 个评委中找出最公平(即评分最接近平均分)和最不公平(即与平均分的差距最大)的评委,程序应怎样实现?

5. 求最大约数

问 555555 的约数中最大的三位数是多少?

* 问题分析与算法设计

根据约数的定义,对于一个整数 N ,除去 1 和它自身之外,凡能整除 N 的数即为 N 的约数。因此,最简单的方法是用 2 到 $N-1$ 之间的所有数去除 N ,即可求出 N 的全部约数。本题只要求取约数中最大的三位数,则其取值范围可限制在 100 到 999 之间。

* 程序说明与注释

```
main()
{ long i;                                /* 使用长整型变量,以免超出整数的表示范围 */
  int j;
  printf("Please input number:");
  scanf("%ld", &i);
  for (j=999; j>=100; j--)
      /* 所求的约数的可能取值是从 999 到 100, j 从大到小 */
      if (i%j==0) {                      /* 若是约数,则输出结果 */
          printf("The max factor with 3 digits in %ld is: %d.\n", i, j);
          break;
      }
}
```

* 运行结果

输入:555555

输出:The max factor with 3 digits in 555555 is: 777.

6. 高次方数的尾数

求 13 的 13 次方的最后三位数。

* 问题分析与算法设计

解本题最直接的方法是:将 13 累乘 13 次后截取最后三位即可。

但是由于计算机所能表示的整数范围有限,用这种“正确”的算法不可能得到正确的结果。事实上,题目仅要求后三位的值,完全没有必要求 13 的 13 次方的完整结果。

研究乘法的规律会发现:乘积的最后三位的值只与乘数和被乘数的后三位有关,与乘数和被乘数的高位无关。利用这一规律,可以大大简化程序。

* 程序说明与注释

```
main()
{ int i, x, y, last=1;          /* 变量 last 保存求 X 的 Y 次方过程中的部分积的后三位 */
  printf("Input X and Y (X * * Y):");
  scanf ("%d * * %d", &x, &y);
  for (i=1; i<=y; i++)          /* X 自乘 Y 次 */
    last = last * x %1000;      /* 将 last 乘 X 后对 1000 取模,即求积的后三位 */
  printf("The last 3 digits of %d * * %d is: %d\n", x, y, last%1000);
}
```

* 运行结果

Input X and Y (X * * Y):13 * * 13

The last 3 digits of 13 * * 13 is: 253

Input X and Y (X * * Y):13 * * 20

The last 3 digits of 13 * * 20 is: 801

7. 阶乘尾数零的个数

100! 的末尾有多少个零?

* 问题分析与算法设计

可以设想:先求出 100! 的值,然后数一下末尾有多少个零。事实上,与上题一样,由于计算机所能表示的整数范围有限,这是不可能的。

为了解决这个问题,必须首先从数学上分析在 100! 结果值的末尾产生零的条件。不难看出:一个整数若含有一个因子 5 则必然会在求 100! 时产生一个零。因此问题转化为求 1 到 100 这 100 个整数中包含了多少个因子 5。若整数 N 能被 25 整除,则 N 包含 2 个因子 5;若整数 N 能被 5 整除,则 N 包含 1 个因子 5。

* 程序说明与注释

```
main()
{ int a, count=0;
```

```

for (a=5; a<=100; a+=5) {      /* 循环从 5 开始,以 5 的倍数为步长,考察整数 */
    count++;                    /* 若为 5 的倍数,计数器加 1 */
    if ( ! (a%25) ) count++;    /* 若为 25 的倍数,计数器再加 1 */
}
printf("The number of 0 in the end of 100! is: %d.\n", count); /* 打印结果 */
}

```

* 运行结果

The number of 0 in the end of 100! is: 24.

* 问题的进一步讨论

本题的求解程序是正确的,但存在明显的缺点。程序中判断整数 N 包含多少个因子 5 的方法是与题目中 100 有关的,若题目中的 100 改为 1000,则就要修改程序中求因子 5 的数目的算法了。

* 思考题

修改程序中求因子 5 的数目的算法,使程序可以求出任意 $N!$ 的末尾有多少个零。

8. 借书方案知多少

小明有五本新书,要借给 A、B、C 三位小朋友,若每人每次只能借一本,则可以有多少种不同的借法?

* 问题分析与算法设计

本问题实际上是一个排列问题,即求从 5 个中取 3 个进行排列的方法的总数。首先对五本书从 1 至 5 进行编号,然后使用穷举的方法,假设三个人分别借这五本书中的一本,当三个人所借的书的编号都不相同时,就是满足题意的一种借阅方法。

* 程序说明与注释

```

main()
{
    int a, b, c, count=0;
    printf("There are different methods for XM to distribute books to 3 readers:\n");
    for (a=1; a<=5; a++)          /* 穷举第一个人借 5 本书中的 1 本的全部情况 */
        for (b=1; b<=5; b++)      /* 穷举第二个人借 5 本书中的 1 本的全部情况 */
            for (c=1; a!=b && c<=5; c++)
                /* 当前两个人借不同的书时,穷举第三个人借 5 本书中的 1 本的全部情况 */
                if (c!=a && c!=b) /* 判断第三人与前两个人借的书是否不同 */
                    printf("count%8? \"%2d:%d,%d,%d\" : \"%2d:%d,%d,%d\n\" ,
                        ++count,a, b, c); /* 打印可能的借阅方法 */
}

```

* 运行结果

There are different methods for XM to distribute books to 3 readers:

1:1,2,3	2:1,2,4	3:1,2,5	4:1,3,2	5:1,3,4
6:1,3,5	7:1,4,2	8:1,4,3	9:1,4,5	10:1,5,2
11:1,5,3	12:1,5,4	13:2,1,3	14:2,1,4	15:2,1,5

16:2,3,1	17:2,3,4	18:2,3,5	19:2,4,1	20:2,4,3
21:2,4,5	22:2,5,1	23:2,5,3	24:2,5,4	25:3,1,2
26:3,1,4	27:3,1,5	28:3,2,1	29:3,2,4	30:3,2,5
31:3,4,1	32:3,4,2	33:3,4,5	34:3,5,1	35:3,5,2
36:3,5,4	37:4,1,2	38:4,1,3	39:4,1,5	40:4,2,1
41:4,2,3	42:4,2,5	43:4,3,1	44:4,3,2	45:4,3,5
46:4,5,1	47:4,5,2	48:4,5,3	49:5,1,2	50:5,1,3
51:5,1,4	52:5,2,1	53:5,2,3	54:5,2,4	55:5,3,1
56:5,3,2	57:5,3,4	58:5,4,1	59:5,4,2	60:5,4,3

9. 杨辉三角形

在屏幕上显示杨辉三角形:

```

          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
.....

```

* 问题分析与算法设计

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,这里仅给出一种。

从杨辉三角形的特点出发,可以总结出:

①第 N 行有 $N+1$ 个值(设起始行为第0行);

②对于第 N 行的第 J 个值: ($N \geq 2$)

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:

$$c(x,y) = \begin{cases} 1 & x=1 \text{ 或 } x=N+1 \\ c(x-1,y-1) + c(x-1,y) & \text{其它} \end{cases}$$

本程序就是根据以上递归的数学表达式编制的。

* 程序说明与注释

```
main()
```

```
{ int i,j,n=13;
```

```

printf("N=");
while (n>12)
    scanf("%d", &n);          /* 控制输入正确的值以保证屏幕显示的图形正确 */
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<i+2; j++) printf("%6d", 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 */
        z = c(x-1,y-1) + c(x-1,y);
        /* 否则, 其值为前一行中第 y-1 列与第 y 列值之和 */
        return(z);
    }
}

```

* 运行结果

输入: N=12

输出:

```

              1
            1  1
          1  2  1
        1  3  3  1
      1  4  6  4  1
    1  5 10 10  5  1
  1  6 15 20 15  6  1
1  7 21 35 35 21  7  1
  1  8 28 56 70 56 28  8  1
    1  9 36 84 126 126 84 36  9  1
      1 10 45 120 210 252 210 120 45 10  1
        1 11 55 165 330 462 462 330 165 55 11  1
          1 12 66 220 495 792 924 792 495 220 66 12  1

```

* 思考题

自行设计一种实现杨辉三角形的方法。

10. 数制转换

将任一整数转换为二进制形式。

* 问题分析与算法设计

将十进制整数转换为二进制形式的方法很多,这里介绍的实现方法利用了C语言能够对位进行操作的特点。对于C语言来说,一个整数在计算机内部就是以二进制形式存储的,所以没有必要再将一个整数经过一系列的运算转化为二进制形式,只要将整数在内存中的二进制表示输出即可。

* 程序说明与注释

```
#include <stdio.h>

main()
{int x;
 printf("Input number:");
 scanf("%d", &x);
 printf("number of decimal form: %d\n", x);
 printf("    it's binary form: ");
 printb(x, sizeof(int) * 8);
    /* x:整数    sizeof(int):int 型在内存中所占的字节数,
    sizeof(int) * 8:int 型对应的位数 */
 putchar('\n');
}

printb(x, n)
    int x, n;
{ if (n > 0) {
    putchar('0' + ((unsigned)(x & (1 << (n - 1))) >> (n - 1)));
    printb(x, n - 1);
}
}
```

* 运行结果

输入:8

输出:number of decimal form: 8

it's binary form: 0000000000001000

输入:-8

输出:number of decimal form: -8

it's binary form: 111111111111000

输入:32767

输出:number of decimal form: 32767