第3章 数式精彩纷呈

数式是由数字、运算符与等号构成的等式。如果说单纯的数或略显单调，那么数式则丰满得多！广阔得多！有趣得多！

本章探求并发掘各类新颖奇妙的数式，与诸位一道领略数学殿堂上数式的风采。

在整数素因数分解式的基础上，探索涉及分数的埃及分数式与桥本分数式；探讨涉及整数的优美和式、平方式、变序数和式与倍积式；探求优美隐序四则运算式与含乘方^的综合运算式。

重点探索与发掘从乘积、四则运算到综合运算的系列对称运算式，感受奇特而优雅的对称美；最后分段和幂式，是对卡普雷卡数广度的拓展与深层次的发掘，回味无穷。

数式精彩，精彩数式！

数式精彩，有待我们去欣赏，欣赏数式精彩所集聚的数学美。

精彩数式，有待我们去发掘，发掘人类尚未开发的数式瑰宝。

3.1 素因数分解式

整数分解素因数（又称分解质因数）是整数分解中最基本的案例。

分解一个整数的素因数并不轻松，甚至是一项艰难繁复的工作。但分解素因数是趣味数学的基础，众多趣味数学问题的求解往往离不开整数的素因数分解。

当整数的素因数不太大时，整数的素因数分解并不棘手。例如，分解 2016，其分解的素因数可写成以下乘积式与指数式：

$2016=2\times2\times2\times2\times2\times3\times3\times7$

$2016=2^5\times3^2\times7$

当分解整数的素因数比较大时，素因数分解如何实施？

本节在寻求分解基础上，编程拓展按指数形式的分解设计，把分解的结果表示为素因数的指数式。

【问题】 对整数 $n=201804$ 分解素因数，所分解的素因数按从小到大写为乘积式。

【思考】 从 2，3 开始，通过试商逐个找出素因数。

(1) 分解思路。

首先求出小于或等于$[\sqrt{n}]$（整数 n 开平方取整）的所有素数。

注意到$[\sqrt{201804}]=449$，小于或等于 449 的素数有 2，3，5，7，…，449，共计 87 个，理

论上这 87 个素数都可能成为 201804 的素因数。

首先分解出 201804 的所有 2 因数，直到变为奇数；然后对余下奇数分解出 201804 的所有 3 因数；再对余下奇数依次分解所有 5 因数、7 因数……直到分解 449 因数。

(2) 按以上思路分解操作。

① 分解 201804 的 2 因数：201804/2＝100902，100902/2＝50451，可得 2 个 2 因数。

② 分解奇数 50451 的 3 因数：50451/3＝16817，16817 不能被 3 整除，可得 1 个 3 因数。

③ 对余下数 16817 通过试商 5，7，11，…，61 等，没有这些因数。

④ 对余下数 16817 通过试商分解 67 因数：16817/67＝251，可得 1 个 67 因数。

⑤ 判断 251 为素数，即 251 为素因数。

(3) 写出素因数分解式。

于是得到 201804 素因数分解式：

$$201804=2\times2\times3\times67\times251$$

因$[\sqrt{201804}]=449$，原则上要试商小于或等于 449 的所有素数，可见试商分解的工作量是比较大的。本问题由于已得 251 为素因数，因此结束对 201804 的素因数分解。

【编程拓展】 对给定的正整数 n 分解素因数，表示为素因数从小到大顺序的指数形式。

如果被分解的整数本身是素数，则注明为素数。

1. 设计要点

为了扩展分解整数的范围，程序设计采用双精度型变量。

首先赋值“b＝n;”，分解操作只对 b 实施，以保持操作过程中整数 n 不变。

(1) 设置条件循环实施试商。

设置 k 条件循环($k=2,3,\cdots,\sqrt{n}$)实施试商，判别 k 是否为整数 n 的因数。

注意到整数 n 的最大因数可能为 $n/2$，用 $k(2\sim n/2)$试商是可行的，但并不是最简便的。事实上，用 $k(2\sim\sqrt{n})$试商可避免许多无效操作，其算法复杂度要低一些。

在 k 试商循环中，若 k 不能整除 b，说明数 k 不是 b 的因数，k 增 1 后继续试商。若 k 能整除 b，说明数 k 是 b 的因数，用“j＋＋;”统计 k 因数的个数；同时 b 除以 k 的商赋给 b (b＝floor(b/k))后继续用 k 试商(注意：可能有多个 k 因数)，直至 k 不能整除 b，k 增 1 后继续试商。

按上述 k 从小至大试商确定的因数显然为素因数。

(2) 检测大因数或素数。

如果整数 n 存在大于$\sqrt{n}$(即 pow(n,0.5))的因数(至多一个)，在试商循环结束后的检测试商后的 b 值。

若 $b=1$，素因数分解完成。

若 $b<n$，b 即为大于$\sqrt{n}$的一个因数，即行输出。

若 $b=n$，即整个试商后 b 的值没有任何缩减，仍为原待分解数 n，说明 n 是素数，做素数说明标记。

由此可见,对于某些难度较大的素因数分解问题,单凭人工推理计算是难以胜任的,在当今信息时代,必须考虑应用程序设计来解决。

(3) 按指数形式输出。

在素因数指数形式中,首先按素因数从小到大排列;如果存在相同的素因数,要求写成指数的形式输出。

在程序设计输出时,通常把指数上标用^标注,如 2^3 输出为 2^3。

例如,分解 123456789,按素因数乘积形式为 123456789=3×3×3607×3803,按素因数指数形式为 123456789=3^2×3607×3803。

为此,引入的变量 j 统计素因子的个数,$j=1$ 时不打印指数;$j>1$ 时需加打指数(^j)。这样要求程序设计进行必要的判别操作。

2. 素因数分解指数形式程序设计

```
//探求指定整数的素因数分解指数形式
#include <math.h>
#include <stdio.h>
void main()
{  double b,i,k,n; int j;
   printf("  请输入正整数 n:");scanf("%lf",&n);
   printf("  %.0f=",n);
   b=n;k=2;j=0;
   while(k<=pow(n,0.5))                          //枚举试商因数 k
     {  if(fmod(b,k)==0)
          { b=floor(b/k);j++;continue; }         //k 为素因数需返回再试
        if(j>=1)
          {  printf("%.0f",k);
             if(j>1) printf("^%d",j);            //打印素因数指数形式
             if(b>1) printf("×");
          }
        k++;j=0;
     }
   if(b>1 && b<n) printf("%.0f",b);              //输出大于 n 平方根的因数
   if(b==n) printf("(素数!)");                   //b=n,表示 n 无素因数
   printf("\n");
}
```

3. 程序运行示例与说明

```
请输入正整数 n:518666803200
518666803200=2^11×3^3×5^2×7^2×13×19×31
```

运行程序的整数 518666803200 是第 1 章探讨的 4-完全数,这样大的数进行素因数分解,应用双精度型处理是适宜的,若按整型数据处理则并不可行。可见,编程也要依据问题的具体实际情况来定,并没有千篇一律的固定模式。

顺便提到，当整数的素因数比较大时，其分解是艰难的。如果一个中学生想用一道题为难一个数学家，只要选择两个比较大的素数 a，b，算出 $a \times b = c$，请他在不使用程序设计的前提下分解 c 的素因数即可。

3.2　埃及分数式

金字塔的故乡埃及也是数学的发源地之一。古埃及数系中，记数常采用分子为 1 的分数，称为“埃及分数”。

人们研究较多且颇感兴趣的问题：把一个给定的整数或分数转化为若干不相同的埃及分数之和。转化的方法可能有很多种。常把分解式中埃及分数的个数最少，或在个数相同时埃及分数中最大分母为最小的分解式称为最优分解式。

把给定整数或分数分解为埃及分数之和，也是一个烦琐艰辛的过程。

例如，对 5/121 的分解，为尽可能减少分解项数，数学家布累策在《数学游览》中给出了以下优化的三项分解式：

5/121＝1/25＋1/759＋1/208725

同时布累策证明了 5/121 不可能分解为两个埃及分数之和。

从项数来说，上述三项分解式不可能再优化了。但对于最大分母，布累策的分解式不是最优的。我国两位青年数学爱好者于 1983 年发现以下 4 个分解式。

$$5/121 = 1/27 + 1/297 + 1/1089 \tag{3-2-1}$$

$$5/121 = 1/33 + 1/99 + 1/1089 \tag{3-2-2}$$

$$5/121 = 1/33 + 1/121 + 1/363 \tag{3-2-3}$$

$$5/121 = 1/33 + 1/91 + 1/33033 \tag{3-2-4}$$

这 4 个分解式都比布累策的结论要优。人们通常约定分解式中不得包含与待分解分数同分母的埃及分数。从这个意义上，显然应把分解式(3-2-3)排除在外。因此，现在所知把 5/121 分解为 3 个埃及分数的最小分母为 1089，即上述埃及分数分解式(3-2-1)和式(3-2-2)。

那么，分解 5/121 为 3 个埃及分数之和，其最大分母能否小于 1089 呢？可通过程序设计来探索拓展。

从简单地构建两个埃及分数分解式入手，先看一个分解埃及分数式的实例。

【问题】　试把分数 5/72 分解为分母分别是 a，b($a<b<200$)的埃及分数式。

$$5/72 = 1/a + 1/b$$

【探求】　拟先确定分母的取值范围，再通过具体实验调试确定。

(1) 明确两个分母关系。

若已知一个分母 a，则另一个分母 b 为

$$b = 72a/(5a - 72)$$

通过计算 72/5，可确定最小分母 a 的取值范围为 $14<a<28$。

(2) 通过 a 取值求 b。

取 $a=15$，代入得 $b=360>200$，不符合要求。

取 $a=16$，代入得 $b=144<200$，符合要求，得埃及分数的两个分母为 16，144。

取 $a=17$,代入得 b 为非整数,不符合要求。

取 $a=18$,代入得 $b=72$,与原分母相同,不符合要求。

取 $a=17\sim23$,代入得 b 为非整数,不符合要求。

取 $a=24$,代入得 $b=36<200$,符合要求,得埃及分数的两个分母为 24,36。

取 $a=25\sim27$,代入得 b 为非整数,不符合要求。

(3) 写出埃及分数式。

因而,得到满足要求 a,$b(a<b<200)$的两个埃及分数式为

$5/72=1/16+1/144$

$5/72=1/24+1/36$

根据埃及分数中最大分母为最小的分解式为最优,显然后者要优于前者。

从以上具体构建可以看出,规定埃及分数中最大分母的上限直接关系到埃及分数式的构建。

若缩小上限,把范围 $a<b<200$ 缩小为 $a<b<100$,则上面第一个分解式不符合要求。

若扩大上限,把范围 $a<b<200$ 扩大为 $a<b<400$,则增加了分解式

$$5/72=1/15+1/360$$

【编程拓展】

对给定的分数 m/d 分解为 3 个埃及分数的分解式,其分母为 a,b,c(a<b<c),最大分母不超过 z,输出所有埃及分数式。

(1) 设计要点。

通过三重循环实施枚举。

确定 a 循环的起始值 a1 与终止值 a2。

1/a1=m/d−2/z ⇔ a1=dz/(mz−2d) (即把 b,c 放大为 z)

3/a2=m/d ⇔ a2=3d/m+1 (即把 b,c 缩减为 a)

b 循环起始取 a+1,终止取 z−1。

c 循环起始取 b+1,终止取 z。

为方便判别,把分数式 m/d=1/a+1/b+1/c 转化为整数式

mabc=d(ab+bc+ca)

对于三重循环的每组 a,b,c,计算 x=mabc,y=d(ab+bc+ca):如果 x=y 且 b,c 不等于 d,即满足分解为 3 个埃及分数式的条件,则打印输出一个分解式。然后退出内循环,继续寻求。

(2) 构建埃及分数式程序设计。

```
//构建指定分数的 3 个埃及分数式和式
#include <stdio.h>
void main()
{  int a1,a2,a,b,c,d,m,n,z; long x,y;
   printf("  确定分数 m/d,请输入 m,d:"); scanf("%d,%d",&m,&d);
   printf("  请确定分母的上界: "); scanf("%d",&z);
   n=0;
```

```
    a1=d*z/(m*z-2*d); a2=d*3/m+1;
    for(a=a1;a<=a2;a++)                                    //建立三重循环枚举
    for(b=a+1;b<=z-1;b++)
    for(c=b+1;c<=z;c++)
    {  x=m*a*b*c; y=d*(a*b+b*c+c*a);                       //计算 x,y 值
       if(x==y && b!=d && c!=d)                            //满足条件时输出分解式
         {  printf("  NO%d: %d/%d=1/%d",++n,m,d,a);
            printf("+1/%d+1/%d \n",b,c);
            break;
         }
    }
    printf("  共以上%d个分解式。\n",n);
}
```

(3) 程序运行示例与说明。

```
确定分数 m/d,请输入 m,d: 5,121
请确定分母的上界: 1100
  NO1: 5/121=1/27+1/297+1/1089
  NO2: 5/121=1/33+1/99+1/1089
  NO3: 5/121=1/45+1/55+1/1089
共以上 3 个分解式。
```

通过程序设计探索得到：分解 5/121 为 3 个埃及分数之和，最大分母最小为 1089，即不可能有比上述 3 个分解式更优的分解。

结果中的最后一个分解式是程序设计得到的新的最优分解式。

注意：确定分母的上界大小直接关系所分解的埃及分数式解。例如，以上对 5/121 的分解，若输入上界为 1000，则没有分解式；若输入上界为 3000，则存在 7 个分解式。

3.3 桥本分数式

数学家桥本吉彦教授于 1993 年在我国山东举行的中日美三国数学教育研讨会上向与会者提出以下填数趣题：把 1,2,…,9 这 9 个数字不重复填入下式的 9 个方格□中，使下面的分数式成立。

$$\frac{\square}{\square\square}+\frac{\square}{\square\square}=\frac{\square}{\square\square}$$

桥本吉彦教授当即给出了一个解答。

【问题】 试探求这一分数式填数趣题的填数结果。

【思考】 以某一简单分数式为基础，每个分数分子、分母同时扩大以寻求数字不重复。

可采用在某一简单分数和式的基础上，试验把每一分数的分子、分母同时扩大或缩小，以寻求在式中出现 1～9 这 9 个数字而不重复的目标。

(1) 以简单分数式 1/14+1/14=1/7 为基础。

前一分数 1/14 的分子、分母同时扩大 4 倍得 4/56;

后一分数 1/14 的分子、分母同时扩大 7 倍得 7/98;

右边分数 1/7 的分子、分母同时扩大 3 倍得 3/21。

3 个分数实现数字 1～9 各出现一次不重复,因而得到一个解:4/56+7/98=3/21。

(2) 以简单分数式 9/51+8/51=1/3 为基础。

前一分数 9/51 的分子、分母同时乘以 2/3 倍得 6/34;

后一分数 8/51 保持不变;

右边分数 1/3 的分子、分母同时扩大 9 倍得 9/27。

3 个分数实现数字 1～9 各出现一次不重复,因而又得到一个解:6/34+8/51=9/27。

采用以上方法试探,可省略对分数式相等的检测,减少计算量。但带有一定的盲目性,要想试探出所有解是困难的。

要求解桥本分数式的所有解,有必要借助程序设计进行全面搜索。

【编程拓展】

桥本分数式这一 9 数字填数趣题究竟有多少个不同的解(约定式左边两个分数中分子小的在前)?

同时,原题并没有要求 3 个分数为最简分数(即分子、分母没有大于 1 的公因数),如果要求式中 3 个分数都为最简分数,又有多少个不同的解?

1. 设计要点

设分数式为 b1/b2+c1/c2=d1/d2,注意到等式左边两个分数交换次序只算一个解答,因而约定 b1<c1。

(1) 设置枚举循环。

对 6 个分数所涉及的六个整数设置六重循环枚举。其中,c1 从 b1+1 开始取值,确保 b1<c1。

(2) 通过三重检测。

若分数式不成立,即 b1 * c2 * d2+c1 * b2 * d2!=d1 * b2 * c2,则返回继续。

要求数字 1～9 在这 6 个变量中出现一次且只出现一次,分离出 9 个数字后用 f 数组统计各个数字的频数(如 f[3]=2,即数字 3 出现两次),存在重复数字时返回。

(3) 要求最简分数处理。

若在确定是否为最简分数时输入字符 y,即选择“要求 3 个分数为最简分数”,设置循环 j(2～9),如果存在 j 为某一分数分子、分母的公因数,则返回。

2. 桥本分数式程序设计

```
//搜索是否最简分数的桥本分数式 b1/b2+c1/c2=d1/d2
#include <stdio.h>
void main()
{  int b1,b2,c1,c2,d1,d2,j,n,t,x,f[11]; char ch;
   printf("  如果要求各分数都为最简分数,请输入字符 y:");
   ch=getchar();
```

```
    n=0;
    for(b1=1;b1<=8;b1++)
    for(c1=b1+1;c1<=9;c1++)                                  //确保 c1>b1
    for(d1=1;d1<=9;d1++)
    for(b2=12;b2<=97;b2++)                                   //枚举 3 个分数的分子、分母
    for(c2=12;c2<=98;c2++)
    for(d2=12;d2<=98;d2++)
    {  if(b1*c2*d2+c1*b2*d2!=d1*b2*c2) continue;  //若分数式不成立则返回
       for(x=0;x<=9;x++) f[x]=0;
       f[b1]++;f[c1]++;f[d1]++;
       f[b2/10]++;f[b2%10]++;f[c2/10]++;
       f[c2%10]++;f[d2/10]++;f[d2%10]++;                     //分离数字用 f 数组统计
       for(t=0,x=1;x<=9;x++)
         if(f[x]!=1) {t=1; break;}                           //检验数字 1~9 是否有重复
       if(t==0 && ch=='y')                                   //检验是否为 3 个最简分数
       {  for(j=2;j<=9;j++)
            if(b1%j==0 && b2%j==0){t=1;break;}
            else if(c1%j==0 && c2%j==0){t=1;break;}
            else if(d1%j==0 && d2%j==0){t=1;break;}
       }
       if(t==0)                                              //输出一个填数解
       {  printf("%4d: %1d/%2d+%1d/%2d",++n,b1,b2,c1,c2);
          printf("=%1d/%2d  ",d1,d2);
          if(n%2==0) printf("\n");
       }
    }
    printf("\n  共以上%d 个解。\n",n);
}
```

3. 程序运行结果与变通

若填数不要求 3 个分数为最简分数，程序运行结果如下。

```
如果要求各分数都为最简分数,请输入字符 y:n
   1: 1/78+4/39=6/52     2: 1/26+5/78=4/39
   3: 1/32+5/96=7/84     4: 1/32+7/96=5/48
   5: 1/96+7/48=5/32     6: 2/68+9/34=5/17
   7: 2/68+9/51=7/34     8: 4/56+7/98=3/21
   9: 5/26+9/78=4/13    10: 6/34+8/51=9/27
共以上 10 个解。
```

若填数要求 3 个分数为最简分数，则程序运行结果如下。

```
如果要求各分数都为最简分数,请输入字符 y:y
   1: 1/26+5/78=4/39     2: 1/32+7/96=5/48
   3: 1/96+7/48=5/32
共以上 3 个解。
```

程序能搜索不要求最简分数的桥本分数式,也能搜索附加条件“要求各分数都为最简分数”的分数式,这是程序设计的特色。

变通:把 0,1,2,…,9 这 10 个数字填入下式的 10 个方格中,要求:各数字不得重复;数字 0 不得填在各分数的分子或分母的首位。

$$\frac{\square}{\square\square}+\frac{\square}{\square\square\square}=\frac{\square}{\square\square}$$

这一分数式填数究竟共有多少个解?

3.4 优美和式与平方式

优美和式是一个创新填数趣题,其优美体现在所涉约定数字在和式中不重复,是和谐美的具体体现。

和式分为三项和式(左边两项求和)和引申的四项和式(左边三项求和)两类,益智训练,各具特色。

3.4.1 9 数字优美和式

因 9 数字优美和式涉及 1~9 这 9 个正整数,不涉及数字 0,变化较为简单。

把 1,2,…,9 这 9 个数字分别填入以下和式中的 9 个□中,要求 1~9 这 9 个数字在式中出现一次且只出现一次,使得和式成立。

$$\square\square\square+\square\square\square=\square\square\square \tag{3-4-1}$$

【问题 1】 试求和式(3-4-1)中的和(即式右 3 位数)的最大值。

【思考】 从进位次数确定式右的数字和入手。

注意到 1~9 这 9 个数字之和为 45。

(1) 确定进位次数。

式左求和时每次进位,所进的 1 相当于 10,求和结果的数字和会减少 9。

例如,数式 218+439=657 中,式右的数字和为 18,式左的数字和为 27,式右比式左的数字和要小 9,就是求和运算的一次进位 8+9=17 造成的。

首先确定:和式(1)左边求和过程中有一次且只有一次进位。

事实上,若左边求和时没有进位,则式左的 6 个数字之和与式右的 3 个数字之和相等,即都为 9 个数字之和 45 的一半(非整数),矛盾。

若左边求和有 2 次进位,使数字和减小了 2×9=18,因而式右的 3 个数字之和为 45−2×9=27 的一半(非整数),矛盾。

若左边求和有 3 次进位,则式右作为求和结果必为 4 位数,矛盾。

可见,和式(1)左边求和过程中有一次且只有一次进位。

(2) 确定式右 3 位数的数字和。

和式左边求和有一次进位,因而和式右边的 3 个数字之和为(45−9)/2=18。

(3) 探索最大值。

根据式右 3 位数的 3 个互不相同的数字之和为 18,其最大值的高位(即百位)数字尽

可能大，选择填 9；十位数字也尽可能大，可填 8；相应个位数字为 1。

对于所填和 981，需寻找相应的数字 1～9 没有重复数字的和式（可能存在多个），例如和式 235＋746＝981。

因而得和式(3-4-1)右边和的最大值为 981。

【编程拓展】

和式(3-4-1)右边和的最小值为多大？和式(3-4-1)共有多少种不同的填法？以下应用程序设计探求这些问题。

为避免式左两个加数交换个位数字，或交换十位数字，或交换百位数字造成多种重复，约定式左两个加数的前一个 3 位数的各位数字分别小于后一个 3 位数的相应数字。

同时，为了便于观察右边和的最小值与最大值，约定右边的和按从小到大排列，且要求每个和值只输出一个和式。

1. 设计要点

设 3 个 3 位数为 a，b，c，和式为 $a+b=c$。

(1) 设置循环。

建立和数 c 循环(315～987)，这里循环初值 315 是百位数为 3(因左边 a，b 都有较小的百位数字)且为 9 倍数的最小 3 位数；循环终值为没有重复数字的最大 3 位数 987；循环步长设为 9，以提高搜索效率。

同时建立加数 a 循环(123～$c/2$)，因约定 $a<b$，因而循环终止值为 $c/2$。另一个加数显然为 $b=c-a$。

(2) 分离数字排除数字重复。

应用整除/与求余%运算分解各数的各个数字。为了检测是否存在重复数字，设置 g 数组统计 a，b，c 所分解的 9 个数字的频数，应用二重循环实施比较，若存在重复数字，则标注 $t=1$ 返回。

(3) 条件输出。

在输出和式时为确保式左前一个 3 位数的各位数字分别小于后一个 3 位数的相应数字，在输出条件中增加条件限制

```
a/100<b/100 && a%10<b%10 && (a/10)%10<(b/10)%10
```

特别地，每输出一个和式，即行退出内循环，确保一个和值 c 只输出一个和式。

2. 程序设计

```
//探求 9 数字三项优美和式:□□□+□□□=□□□
#include <stdio.h>
void  main()
{  int a,b,c,d,t,m,n,x,g[10];
   n=0;
   for(c=315;c<=987;c+=9)                          //和 c 为 9 倍数
   for(a=123;a<c/2;a++)
   {  b=c-a;
```

```
        for(x=0;x<=9;x++) g[x]=0;
        d=a;g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++; //分解并统计 9 个数字的频数
        d=b; g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++;
        d=c; g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++;
        for(t=0,x=1;x<=9;x++)
          if(g[x]!=1) {t=1;break;}                    //检验数字 1~9 各出现一次
          if(t==0 && a/100<b/100 && a%10<b%10 && (a/10)%10<(b/10)%10)
          {  printf("%5d: %d+%d=%d",++n,a,b,c);      //统计并输出一个解
             if(n%3==0) printf("\n");
             a=c/2;break;                             //确保一个和值 c 只输出一个和式
          }
    }
    printf("\n  共以上%d 个 9 数字三项和式。\n",n);
}
```

3. 程序运行结果与说明

```
1: 173+286=459    2: 173+295=468    3: 127+359=486
4: 127+368=495    5: 162+387=549    6: 128+439=567
…
28: 216+738=954   29: 215+748=963   30: 314+658=972
31: 235+746=981
共以上 31 个 9 数字三项和式。
```

以上运行结果输出 31 个优美和式,这 31 个和式靠人工推算可能很难完成。

同时看出,和式(3-4-1)右边和的最小值为 459,最大值为 981,与上面所解相同。

【引申】 把和式(3-4-1)左边的两项分解为三项,即为以下的四项和式。

$$□+□□+□□□=□□□ \tag{3-4-2}$$

和式(3-4-2)涉及四项 $a+b+c=d$,在以上程序基础上,增加一重循环,各参数进行相应变通,具体修改自行完成。

优美和式(3-4-2)的解如下:

```
 1: 7+58+169=234   2: 6+58+179=243   3: 4+68+279=351
 4: 1+72+386=459   5: 1+72+395=468   6: 6+28+479=513
 7: 6+27+498=531   8: 1+82+493=576   9: 4+38+579=621
10: 1+43+685=729  11: 1+42+695=738  12: 2+53+764=819
13: 1+52+793=846
共以上 13 个 9 数字四项和式。
```

由以上运行结果输出的 13 个和式看出,和式(3-4-2)右边和的最大值为 846,最小值为 234。

3.4.2 10 数字优美和式

因 10 数字优美和式涉及 0～9 这 10 个数字,涉及数字 0,所以其变化较为复杂。

把0,1,2,…,9这10个数字分别填入以下和式中的10个□中,要求0～9这10个数字在式中出现一次且只出现一次(约定0不能在各个整数的首位),使得和式成立。

□□□＋□□□＝□□□□　　(3-4-3)

【问题2】 试求和式(3-4-3)中和(即式右4位数)的最大值。

【思考】 从进位次数确定式右4位数的数字和入手。

(1) 确定式(3-4-3)右边4位数的数字和。

左边求和至少有一次进位(否则右边和不可能为4位)。

同时,左边求和过程中不会出现2次进位。假设出现2次进位,注意到每次进位数字和会减少9,右边4位数的数字和为(45－2×9)/2,非整数,矛盾。

当左边求和有一次进位时,右边数字和为(45－9)/2＝18。

当左边求和有3次进位时,右边数字和为(45－3×9)/2＝9。

(2) 右边的4个数字中必有一个0。

当右边数字和为9,或为18时,数字0必出现在右边的4位数中。

(3) 右边数字和为9时探索最大值。

当右边数字和为9时,注意到右边4位数的高位只能为1,还有一个数字0,因而4个数字中的最大数字只可能为6(假设最大数字为7,造成两个数字1重复;假设最大数字为8,造成两个数字0重复)。

设4个数字中的最大数字为6,则4位数最大值可能为1620,1602。

注意到最大值若为1620,由于0出现在个位,左边没有适当数字匹配,不能成为和式。

设右边4位数最大值为1602,存在相应的和式:743＋859＝1602。

结论:和式右边和的最大值为1602。

【编程拓展】

和式(3-4-3)右边和的最小值为多大?和式(3-4-3)共有多少种不同的填法?以下借助程序设计探求这些问题。

为避免式中两个加数交换个位数字,或交换十位数字,或交换百位数字造成多种重复,约定这两个加数中前一个3位数的各位数字分别小于后一个3位数的相应数字。

同时,为了便于观察右边和的最小值与最大值,约定右边和按从小到大排列,且要求每个和值只输出一个和式。

(1) 设计要点。

设3个3位数为a,b,c,即和式为$a+b=c$。

建立和数c循环(1026～1987),这里1026是被9整除且没有重复数字的最小4位数,循环步长设为9,以提高搜索效率。

同时建立加数a循环(102～$c/2$),这里102是没有重复数字的最小3位数,约定$a<b$,因而循环终止值为$c/2$。另一个加数显然为$b=c-a$。

应用整除/与求余%运算分解各数的各个数字。为了检测是否存在重复数字,设置g数组统计a,b,c所分解的10个数字的频数,应用二重循环实施比较。

特别地,每输出一个和式,即行退出内循环,确保一个和值只输出一个和式。

(2) 程序设计。

```
//探求 10 数字三项优美和式:□□□+□□□=□□□□
#include <stdio.h>
void  main()
{  int a,b,c,d,t,m,n,x,g[10];
   n=0;
   for(c=1026;c<=1987;c+=9)                          //1026 是能被 9 整除的最小 4 位数
   for(a=102;a<c/2;a++)
   {  b=c-a;
      if(b>999) continue;
      for(x=0;x<=9;x++) g[x]=0;
      d=a;g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++;
      d=b;g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++;
      g[c/1000]++;d=c%1000;                          //统计式中各数字的频数
      g[d%10]++;g[d/100]++;m=(d/10)%10;g[m]++;
      for(t=0,x=0;x<=9;x++)
        if(g[x]!=1) {t=1;break;}                     //检验数字 0~9 各出现一次
      if(t==0)
      {  printf("%4d: %d+%d=%d",++n,a,b,c);         //统计并输出一个解
         if(n%3==0) printf("\n");
         a=c;break;
      }
   }
   printf("\n  共以上%d 个 10 数字三项和式。\n",n);
}
```

(3) 程序运行结果与说明。

```
 1: 437+589=1026   2: 246+789=1035   3: 264+789=1053
 4: 473+589=1062   5: 324+765=1089   6: 342+756=1098
 7: 347+859=1206   8: 426+879=1305   9: 624+879=1503
10: 743+859=1602
共以上 10 个 10 数字三项和式。
```

由以上运行结果输出的 10 个优美和式看出,和式(3-4-3)右边和的最小值为 1026,最大值为 1602,与上面所解结论相同。

【引申】 把和式(3-4-3)左边的两项分解为三项,即为以下的四项和式。

$$□+□□+□□□=□□□□ \tag{3-4-4}$$

和式(3-4-4)涉及四项 $a+b+c=d$,在以上程序基础上,增加一重循环,各参数进行相应变通,具体修改自行完成。

优美和式(3-4-4)的解如下:

```
1: 3+45+978=1026
2: 2+46+987=1035
3: 2+64+987=1053
4: 3+74+985=1062
共以上 4 个 10 数字四项和式。
```

由以上运行结果输出的 4 个和式看出，和式(3-4-4)右边和的最大值为 1062，最小值为 1026。

3.4.3 优美平方式

优美平方式：如果数字 1～9 不重复填入式(3-4-5)的 9 个□中使等式成立，则该等式称为优美平方式。

$$□□□□□□=(□□□)^2 \tag{3-4-5}$$

式(3-4-5)的左边为一个 6 位数，右边是一个 3 位数的平方。

编程构建所有优美平方式。

1. 设计要点

设置 f 数组存储式中各个数字的频数，便于比较是否存在重复数字。

(1) 枚举循环设置。

若整数 a 为 4 位，则 a^2 的位数达 7 位，超出范围。因而整数 a 只能为 3 位。

通过循环计算最小的 3 位整数 t，设置循环 a(t+1～10t−1)枚举 3 位整数。

计算 d=a*a，若 d 的位数不足 6 位，则 a 与 d 的位数之和小于 9，因而导致数字 1～9 填不下，则返回。

(2) 分离整数数字并统计频数。

应用求余与取整运算分离 a，d 的各个数字。

在数组元素 f[k](k：0～9)清零后，通过“f[k]++;”统计各数字的频数。

(3) 检测重复数字。

检测整数 a，d 的数字频数 f[k]，若 f[k]≠1(k：1～9)，则返回。

若 f[k]=1(k：1～9)，满足优美要求，则输出平方式。

2. 程序设计

```
//探求 9 数字优美平方式
#include <stdio.h>
void main()
{  int k,p,s,f[10]; long a,d,t,w;
   s=0;t=1;
   for(k=1;k<=2;k++) t=t*10;                    //t 为 3 位最小整数
   for(a=t+1;a<=t*10-1;a++)
   {  d=a*a; w=d;                               //确保 d 为平方数
      if(d<100000) continue;
      for(k=0;k<=9;k++) f[k]=0;
      while(w>0)                                //分离 d 的数字并统计频数
```

```
        { k=w%10;f[k]++;w=w/10;}
        w=a;
        while(w>0)                                    //分离 a 的数字并统计频数
        { k=w%10;f[k]++;w=w/10;}
        for(p=0,k=1;k<=9;k++)
          if(f[k]!=1) p=1;                            //平方式中是否有重复数字
        if(p==0)
          printf("%d: %ld=%ld^2\n",++s,d,a);          //统计个数并输出平方式
    }
    if(s>0)  printf("  共以上%d 个 9 数字优美平方式。\n",s);
    else  printf("\n  不存在满足要求的平方式。\n");
}
```

3. 程序运行示例与说明

```
1: 321489=567^2
2: 729316=854^2
共以上 2 个 9 数字优美平方式。
```

由以上运行结果可以看出,输出的 2 个优美平方式中的 9 个数字为 1～9 不重复出现。

能否加入数字 0 构建 10 数字优美平方式呢? 回答是否定的。因为 3 位整数 a 的平方数最多为 6 位,若 a 增加为 4 位,则 $d=a^2$ 至少为 7 位,超出 10 位的数字范围。

3.5 优美综合运算式

本节创新构建隐序四则运算式及综合运算数学式,这是一个有趣也有难度的填数游戏。

各数学式称为“优美”,是指各个数字在式中不重复,是和谐美的具体体现。

3.5.1 隐序四则运算式

本节所论述的数式除了含四则运算与数字不重复外,还必须符合指定隐序。这一新增要求是新颖的,也是有趣的。

把 0,1,2,…,9 这 10 个数字不重复填入以下含加、减、乘、除(乘除运算优先于加减运算)的四则运算式中的 10 个□中,使等式成立。

$$□□□+□□\div□-□□\times□=□ \tag{3-5-1}$$

约定式(3-5-1)填数字时,1,0 不出现在式左边的 1 位数中,且数字 0 不能为整数首位。

同时,要求式中的 10 个不重复的数字须符合指定隐序,指定隐序由输入的整数决定。例如,如果指定隐序为 4 位数 2019,则该运算式中数字的隐含顺序:数字 2 须在 0 的左边,数字 0 须在 1 的左边,而数字 1 须在 9 的左边(这就是隐序的含意)。

例如，267＋80÷5－31×9＝4 就是一个符合 2019 指定隐序的优美四则运算式。

输入指定隐序的整数，构建并输出所有符合指定隐序的优美四则运算式。

1. 设计要点

(1) 数据结构。

以上四则运算式中的各数依次设置为变量 a，b，c，d，e，f，即四则运算式为

$$a+b/c-d*e=f$$

同时设置 3 个数组：g 数组统计式中共 6 个整数的 10 个数字的频数，便于判别重复数字；h 数组标记式中 10 个数字的位置，便于判别是否符合指定隐序；w 数组存储指定隐序，为判别是否符合指定隐序提供数据。

(2) 设置枚举循环。

设置 a，c，d，e，f 循环，其中 c，e，f 都是 1 位数，f 循环 0～9 取值，c，e 循环 2～9 取值；数 a 为 3 位数，循环 102～987 取值；数 d 为 2 位数，循环 10～98 取值。

(3) 计算整数 b。

把以上四则运算式变形为以下的乘积式是简便的。

$$b=(d*e+f-a)c$$

对每组 a，c，d，e，f，计算 b。这样处理，可省略 b 循环，省略 b 是否能被 c 整除，也省略等式是否成立的检测。

计算 b 后，检测 b 是否为 2 位数。若计算所得 b 非 2 位数，则返回。

(4) 判别是否存在重复数字。

然后分别对 6 个整数进行数字分离，设置 g 数组对 6 个整数分离的共 10 个数字进行统计，g[x]即为数字 x(0～9)的频数。同时，应用 h 数组标记式中 10 个数字的位置。

例如，g[3]＝2，即为 2 个数字 3；h[5]＝4，即数字 5 在数式中第 4 个位置上。

若某一 g[x]≠1，不满足 10 个数字都出现一次且只出现一次，标记 t＝1。

若所有 g[x]全为 1，满足 10 个数字都出现一次且只出现一次，保持标记 t＝0，则优美四则运算式成立。

(5) 判别是否符合指定隐序。

式中 10 个数字的分布必须符合指定顺序的要求，这既是重点，也是难点。

输入的指定隐序的整数 m 要求没有重复数字，位数不限(当然不能超过 10)。因此，首先分离隐序 m 的各个数字(设为 n 个数字)，并从个位开始赋值给 w[1]～w[n]。

综合 h 与 w 数组来判别所得优美四则运算式的 10 个数字分布是否符合指定要求。

因指定隐序的整数 m 的 w[k]在 w[k－1]的前面，即 h[w[k]]＜h[w[k－1]](k：2～n)才是符合指定隐序。若出现在某个 k(2～n 中的一个)h[w[k]]＞＝h[w[k－1]](k：2～n)不符合指定隐序，即返回试下一个数式。

2. 程序设计

```
//探求指定隐序的优美四则运算式:□□□+□□/□-□□×□=□
//式中 10 个不重复的数字须符合指定隐序
#include <stdio.h>
```

```
void main()
{  int a,b,c,d,e,f,k,t,m,n,x,y,z,g[11],h[11],w[11];
   printf("  请输入指定隐序数 m(数字互不相同): "); scanf("%d",&m);
   n=0;y=m;
   while(y>0)
     {x=y%10;w[++n]=x;y=y/10;}                    //分离 m 的数字赋值 w 顺序数组
   m=0;
   for(f=0;f<=9;f++)
   for(a=102;a<=987;a++)
   for(c=2;c<=9;c++)
   for(d=10;d<=98;d++)                            //循环实施枚举 a,c,d,e,f
   for(e=2;e<=9;e++)
   {  b=(d*e+f-a)*c;                              //计算变量 b 省去 b 循环
      if(b<10 || b>99)  continue;
      for(x=0;x<=9;x++) g[x]=0;
      g[c]++;g[e]++;g[f]++;                       //3 个 1 位数给 g,h 数组赋值
      h[c]=6;h[e]=9;h[f]=10;
      y=a;
      for(k=1;k<=3;k++)
        { x=y%10;g[x]++;h[x]=4-k;y=y/10;}         //分离 a 的 3 个数字给 g,h 数组统计
      g[b/10]++;h[b/10]=4;g[b%10]++;h[b%10]=5;
      g[d/10]++;h[d/10]=7;g[d%10]++; h[d%10]=8; //分离 b,d 数字给 g,h 数组统计
      for(t=0,x=0;x<=9;x++)
        if(g[x]!=1) {t=1;break;}                  //检验数字 0~9 是否存在重复
      if(t==1) continue;
      for(t=0,k=n;k>=2;k--)
        if(h[w[k]]>=h[w[k-1]])    {t=1;break;} //检验是否符合 w 数组指定隐序
      if(t==1) continue;
      printf("  %2d:%d+%d÷%d",++m,a,b,c);
      printf("-%d×%d=%d",d,e,f);
      if(m%2==0) printf("\n");                    //以每行 2 个输出符合要求数式
   }
   printf("  共以上%d个符合指定隐序的优美四则运算式!\n",m);
}
```

3. 程序运行示例与说明

```
请输入指定隐序数 m(数字互不相同): 2019
 1:267+80÷5-31×9=4    2:204+18÷9-67×3=5
 3:207+18÷9-34×6=5    4:240+16÷8-79×3=5
 5:270+84÷6-31×9=5    6:720+45÷3-81×9=6
 7:250+81÷9-63×4=7    8:643+20÷5-71×9=8
 9:205+68÷4-71×3=9   10:208+56÷4-71×3=9
11:237+80÷5-61×4=9   12:250+81÷3-67×4=9
共以上 12 个符合指定隐序的优美四则运算式!
```

从运行结果中清楚可见，每个式中的 10 个数字无重复，且其分布隐含指定 2019 顺序。式中含加、减、乘、除的四则运算，运算结果使等式成立。

若输入的指定隐序数 m 为 20195，则只有以上第 2,3,4,5 个解满足要求。

若输入的指定隐序数 m 为 20197，则只有以上第 2,7 个解满足要求。

这里特别强调，输入隐序数 m 时，不能含有重复数字，否则，程序难以进行测试。

3.5.2 综合运算式

【问题】 在以下数式(3-5-2)中已填有数字 0,4,6，把另 7 个数字 1,2,3,5,7,8,9 不重复填入以下含加、减、乘、除与乘方的综合运算式中的 7 个□中，使得该式成立。

$$4^{6}+\square\square\div\square-\square\square\square\times\square=0 \tag{3-5-2}$$

约定数字 1 不出现在数式的 1 位数中。

【思考】 把式中乘积项设置在大于乘方数值附近探试。

式中 10 个数字已填有 3 个，是为了减少填数字的难度。

设式(3-5-2)中的 2 位数为 x。

注意到已有 $4^{6}=4096=512\times8$，因而拟把□□□×□设置在大于 512×8 附近，分以下情形讨论。

(1) 取 $4^{6}+\square\square\div\square-512\times9=0$，则□□÷□=512，不可能实现。

(2) 取 $4^{6}+\square\square\div\square-513\times8=0$，则□□÷□=8，即 2 位数 x 为 8 的倍数。

$x=16,24,32,40,48,56,64$ 与 80，分别导致数字 6,4,3,4,4,6,4,8 矛盾。

而对于 $x=72$，有 72÷9=8，则可得综合运算式

$$4^{6}+72\div9-513\times8=0 \tag{3-5-3}$$

即得对应式(3-5-2)的综合运算式(3-5-3)。

【编程拓展】 把 0,1,2,…,9 这 10 个数字不重复填入以下含加、减、乘、除与乘方的综合运算式中的 10 个□中，使得该式成立。

$$\square^{\square}+\square\square\div\square-\square\square\square\times\square=\square$$

约定数字 1,0 不出现在式左边的 1 位数中，且 0 不能为整数首位。试探索并输出所有综合运算式。

1. 设计要点

设综合运算式为

$$a\text{\^{}}b+z/c-d*e=f$$

把所有变量设置为整型，其中乘方 a^b 用 a 自乘 b 次实现。

(1) 设置枚举循环。

设置 a,b,c,d,e,f 循环，其中 a,b,c,e,f 都是 1 位数，f 循环为 0～9 取值，而 a,b,c,e 循环为 2～9 取值；数 d 为 3 位数，循环为 102～987。

(2) 计算数 z。

对每组 a,b,c,d,e,f，计算

$$z=(d*e+f-a\text{\^{}}b)*c$$

这样设计，可省略 z 循环，省略 z 是否能被 c 整除，省略等式是否成立的检测。

计算 z 后,检测 z 是否为 2 位数。若计算所得 z 非 2 位数,则返回。

(3) 判别是否存在重复数字。

然后分别对 7 个整数进行数字分离,设置 g 数组对 7 个整数分离的共 10 个数字进行统计,g[x]即为数字 x(0～9)的频数。

若某一 g[x]不为 1,不满足 10 个数字都出现一次且只出现一次,标记 t=1。

若所有 g[x]全为 1,满足 10 个数字都出现一次且只出现一次,保持标记 t=0,则输出所得的优美综合运算式。

2. 程序设计

```
//探求完美综合运算式:□^□+□□/□-□□□*□=□
//式左的 1 位数不能为 0 或 1, 式左的整数首位不能为 0
#include <stdio.h>
void main()
{  int a,b,c,d,e,f,k,t,n,x,y,z,g[11];
   n=0;
   for(f=0;f<=9;f++)
   for(a=2;a<=9;a++)
   for(b=2;b<=9;b++)
   for(c=2;c<=9;c++)
   for(d=102;d<=987;d++)                          //各数实施枚举
   for(e=2;e<=9;e++)
     {  for(t=1,k=1;k<=b;k++) t=t*a;              //计算乘方 a^b
        z=(d*e+f-t)*c;
        if(z<10 || z>98)  continue;               //计算 z,若 z 非 2 位数则返回
        for(x=0;x<=9;x++) g[x]=0;
        g[f]++;g[a]++;g[b]++;g[c]++;g[e]++;       //5 个 1 位数给 g 数组赋值
        y=d;
        for(k=1;k<=3;k++)
          { x=y%10;g[x]++;y=y/10;}                //分离 d 的 3 个数字给 g 数组统计
        g[z/10]++;g[z%10]++;                      //分离 z 的 2 个数字给 g 数组统计
        for(t=0,x=0;x<=9;x++)
          if(g[x]!=1) {t=1;break;}                //检验数字 0~9 是否各出现一次
        if(t==0)                                  //输出一个解
          {  printf("  %2d:%d^%d+%d÷%d",++n,a,b,z,c);
             printf("-%d×%d=%d  ",d,e,f);
             if(n%2==0) printf("\n");
          }
     }
     printf("  共以上%d 个完美综合运算式!\n",n);
}
```

3. 程序运行示例与变通

```
 1:4^6+72÷9-513×8=0    2:5^4+78÷6-319×2=0
 3:9^3+48÷6-105×7=2    4:9^3+64÷8-105×7=2
 5:2^9+78÷6-130×4=5    6:9^3+64÷2-108×7=5
 7:2^9+80÷5-174×3=6    8:5^4+18÷9-207×3=6
 9:9^3+50÷2-187×4=6   10:5^4+96÷8-210×3=7
11:6^3+54÷9-107×2=8   12:8^3+64÷2-107×5=9
共以上 12 个完美综合运算式!
```

由以上运行结果看出，输出 12 个完美综合运算式，式中不重复含有 0～9 这 10 个数字，设置有加、减、乘、除与乘方五则运算，优雅地展示出数式之美。

以上设计中应用 a 自乘 b 次实现 a^b，这样处理是简便的。同时，应用 g 数组进行数字统计检验是否存在有重复数字，检测手段颇为新颖。

变通：把 0,1,2,…,9 这 10 个数字分别填入以下含加、减、乘、除与乘方的综合运算式中的 10 个□中（约定 0,1 要求同前），修改以上程序，使得下式成立。

$$\square^{\square}+\square\square\square\div\square-\square\square\times\square=\square$$

$$\square^{\square}+\square\square\div\square-\square\square\times\square=\square\square$$

请问：上述综合运算式共有多少种不同的填入方法？

3.6　变序数和式

变序数是由同一组数字通过不同排列所得的位数相同的整数。

例如，由 1,0,2,2 这 4 个数字通过不同排列组成的 4 位整数 1022,1202,1220,2012,2021,2102,2120,2201,2210 都是变序数，也称同基因数。而 0122 实际上只是一个 3 位数，并不含 0，不属于上述诸数的变序数范畴。

本节探索构建涉及变序数的新颖数式——变序数和式。

例如，2385＋2853＝5238 就是一个 4 位变序数和式，式中 3 个 4 位数都是由数字 2,3,5,8 通过不同排列生成，是变序数关系。

先看一个简单的 3 位变序数和式填数题。

【问题】 在 1～9 这 9 个数字中选择 3 个不同的数字，把所选的 3 个数字以某种不同顺序分别填入以下和式中的 3 个 3 位数中，使和式成立。

$$\square\square\square+\square\square\square=\square\square\square$$

也就是说，式中 3 个 3 位数的组成数字是相同的，只是排列顺序不同，这 3 个 3 位数是变序数关系。

共有多少个不同的 3 位变序数和式（约定式中第 1 个 3 位数小于第 2 个 3 位数）。

【思考】 从进位次数确定式右 3 位数的数字和入手。

因为和式中的 3 个 3 位数是由同样数字经不同排列组成的，所以式左的数字之和为式右的数字和的 2 倍。

（1）确定进位次数与式右数字和。