# 第７章：　多次元超立体魔方陣の最新研究：摂田　寛二

多次元超立体魔方陣の研究に新たな展開があったので，以降数節にわたって報告する。

## 第１節：　超立体方陣研究の基礎

第６章で多次元超立体魔方陣の研究の最初の成果を報告したが，その後の研究でわかった事柄を補足説明する。結論の大筋に変更があるわけでは決してないが，書き換えて改訂するには規模が大きくなりそうなので，新たに章を起こすことにした。

今度の展開のそもそもの発端は，「超立体方陣の**展開見取り図**を何通りも描きたい。同じ種類の**基本図**をいったい何通り描くことができるものか」という素朴な疑問であった。

今までの研究でも，何種類もの展開見取り図を用意したが，同じ種類のものを何通りも描くことは無かった。通常三次元以下の魔方陣では，基本図は１個描けばそれで十分である。向きを変え視点を変えても，魔方陣が全然違った姿で現れることはない。我々はいつでも全体を見通すことができ，必要な情報をすべてその基本図から得ることができた。

ところが，**多次元超立体方陣**では，その常識的な習慣が通用しない。向きを変え視点を変えると，全然違った姿が現れるからだ。何しろ我々には，全体がいつでも見えない。神ならぬ人間は，誰も全体を直視することができないのである。陰に隠れた部分がどのようになっているかは，せいぜい推理することができるのみだ。超立体の超立体たる所以である。

ならば，我々には何通りもの違った基本図を描く必要があるのではないか。

だが，これを想像力だけで描くのはなかなか難しい。１通りくらいは描けても，何通りも描くのは難しい。まして全ての場合を尽くして具体的に描くのは，ひどく煩わしい。基本図の総数は理論的に予想して計算することができる。具体的な形を解の側から帰納して求めることもできるが，それでは順序が逆になってしまう。

何か良い工夫は無いか。直接に描かせる良いコンピュータ・プログラムは無いか。

探索を続けたら，それがあった。実に簡単な仕掛けでできるのである。最初は，信じ難い気がしたが，様々なテストを施して確かな手応えを得，次第に信じられるようになった。

**Ｎ進数発生プログラム**を直接に利用するのである。

その説明に入る前に少し既知の事柄を**復習**して，頭の準備体操をすることにしよう。

### １．多重ループのプログラムができること

まことに申し訳ない気がするが，ここでコンピュータ・プログラムについて若干触れなければならない。なるべく簡単なことしか書かないので，辛抱して読んでほしい。

```
//
//** Study of Extra-Cubic Object of Order 2^4 **
void pr41(){
 short n;
  printf("\n");
  printf("Study #41: One Dimensional Serial Numbers\n");
  for(n=1;n<17;n++){
    printf("%3d",n);
  }
  printf("\n");
}
```

```
P41：　一次元16数（連続する自然数）
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
```

```
//
void pr42(){
 short n;
  printf("\n");
  printf("Study #42: Two Dimensional Placement of Numbers:\n");
```

```
  printf(" [8 Pairs of Self-Complementary Numbers]\n");
  printf("   ");
  for(n=1;n<9;n++){printf("%3d",n);}
  printf("\n +)");
  for(n=1;n<9;n++){printf("%3d",17-n);}
  printf("\n  -------------------------\n   ");
  for(n=1;n<9;n++){printf("%3d",17);}
  printf("\n");
}
```

P42：　２列に折り返して置いて補数ペアを作る

```
     1  2  3  4  5  6  7  8
 +) 16 15 14 13 12 11 10  9
  ----------------------------
    17 17 17 17 17 17 17 17
```

最初の２つのプログラムは，ただ，連続する16個の自然数を単純に並べただけである。まだ for(.条件式.){...} 文は，一重のままで多重ループにはなっていない。

なのに何故この例から始めるかと言えば，これこそが全ての四方陣の出発点であり，全現象の背後にある唯一の「実体」だからである。

以降は，この連続する自然数16個を小分けにして，整然と並べる仕方の実例である。

```
//
void pr43(){
 short m,n;
  printf("\n");
  printf("Study #43: Two Dimensional Placement of Numbers:\n");
  printf(" [Two Dimensional Square of Order 4^2]\n");
  for(m=0;m<4;m++){
    printf("  ");
    for(n=0;n<4;n++){
      printf("%3d",m*4+n+1);
    }
    printf("\n");
  }
}
```

p43：16数を４数ずつ４行に分けて置く

```
   1  2  3  4
   5  6  7  8
   9 10 11 12
  13 14 15 16
```

このプログラムで初めて多重ループを用いている。for(.条件式.){...} 文の中にもう一つの for(.条件式.){...} 文が「入れ子」式に入っている形だ。

こうする目的は，４数ずつ４行に分けて表を書くだけだが，期せずして四方陣の「基本図」を描けている。４×４の「正方配列」を表現し得ている。

これにいろいろ付加プログラムを付けて，いかにも「基本図」らしく作図してみよう。

```
//
void pr44(){
 short d0,d1;
  printf("\n");
  printf("Study #44: Two Dimensional Square of Order 4^2\n");
  printf("     Classic/  Positional Parameters/\n");
  for(d0=0;d0<4;d0++){
    printf("  ");
    for(d1=0;d1<4;d1++){printf("%3d",d0*4+d1+1);}   //Change
```

```
    printf("  ");
   //
    for(d1=0;d1<4;d1++){printf(" (%d,%d)",d0,d1);}  //Change
    printf("\n");
  }
 //
  printf(" Mathematical/         N4i/        H/D4i/L\n");
  for(d0=0;d0<4;d0++){
    printf("  ");
    for(d1=0;d1<4;d1++){printf("%3d",d0*4+d1);}      //Change
    printf("  ");
   //
    for(d1=0;d1<4;d1++){
      printf(" %d%d",d0,d1);}                        //Change
    printf("   ");
   //
    printf("%2d%2d%2d%2d  ",d0,d0,d0,d0);            //Change
    for(d1=0;d1<4;d1++){
      printf("%2d",d1);}                             //Change
    printf("\n");
  }
}
```

p44： 二次元四方陣

```
   ／古典的表記    ／二次元座標による位置
    1  2  3  4    (0,0) (0,1) (0,2) (0,3)
    5  6  7  8    (1,0) (1,1) (1,2) (1,3)
    9 10 11 12    (2,0) (2,1) (2,2) (2,3)
   13 14 15 16    (3,0) (3,1) (3,2) (3,3)
   ／近代的表記    ／四進法表記     ／四進法分解図
    0  1  2  3    00 01 02 03     0 0 0 0    0 1 2 3
    4  5  6  7    10 11 12 13     1 1 1 1    0 1 2 3
    8  9 10 11    20 21 22 23     2 2 2 2    0 1 2 3
   12 13 14 15    30 31 32 33     3 3 3 3    0 1 2 3
```

「基本図」の意味は，これが実際の解とは違うということ。基本ポジションの名前を意味したり，変数名を表したり，原初方陣の第一代表解を指し示したりするのに使う。

このプログラムでわかるように，二重ループは，連続する自然数を４数ずつ４行に分けて書くばかりでなく，四進数まで発生させている。そのために割り算を用いずに直接に「四進法表記図」や「四進法分解図」を描けるのである。

元の自然数は，N=d0*4+d1+1 で合成している。四進法の計算式である。

この辺のマジックを，しっかり学習して身に付けてほしい。

//Change とあるのは，次のプログラムとの相違点を示す。

```
//
void pr45(){
 short d0,d1;
  printf("\n");
  printf("Study #45: Another View of 2 Dimensional Square of Order 4^2\n");
  printf("     Classic/   Positional Parameters/\n");
  for(d0=0;d0<4;d0++){
    printf("  ");
    for(d1=0;d1<4;d1++){printf("%3d",d1*4+d0+1);}   //Change
    printf("  ");
   //
    for(d1=0;d1<4;d1++){printf(" (%d,%d)",d1,d0);}  //Change
    printf("\n");
```

```
  }
 //
  printf(" Mathematical/         N4i/        H/D4i/L\n");
  for(d0=0;d0<4;d0++){
    printf("  ");
    for(d1=0;d1<4;d1++){printf("%3d",d1*4+d0);}      //Change
    printf("  ");
   //
    for(d1=0;d1<4;d1++){
      printf(" %d%d",d1,d0);}                        //Change
    printf("   ");
   //                                                //Change All
    for(d1=0;d1<4;d1++){
      printf("%2d",d1);}
    printf("  %2d%2d%2d%2d",d0,d0,d0,d0);
    printf("\n");
  }
}
```

P45：別角度から見た二次元四方陣

```
    ／古典的表記    ／二次元座標による位置
    1  5  9 13    (0,0) (1,0) (2,0) (3,0)
    2  6 10 14    (0,1) (1,1) (2,1) (3,1)
    3  7 11 15    (0,2) (1,2) (2,2) (3,2)
    4  8 12 16    (0,3) (1,3) (2,3) (3,3)
    ／近代的表記    ／四進法表記     ／四進法分解図
    0  4  8 12    00 10 20 30     0 1 2 3    0 0 0 0
    1  5  9 13    01 11 21 31     0 1 2 3    1 1 1 1
    2  6 10 14    02 12 22 32     0 1 2 3    2 2 2 2
    3  7 11 15    03 13 23 33     0 1 2 3    3 3 3 3
```

//Change のところをちょっと書き換えただけで，このようになる。何のいたずらをしたのか。d0，d1 の順序を交換しただけだと，気付いてほしい。結果は，主対角線の１本を軸に軸対称移動した形になった。またの名を「鏡映反転」とも言う。良く出て来る変換である。
面白いのは，これも「基本図」の１つだということ。全然別物を作ったわけではない。

次は，ループを４重にして**二進数**を発生させてみよう。

```
//
void pr46(){
 short d0,d1,d2,d3;
  printf("\n");
  printf("Study #46: Four Dimensional Extra-Cubic Object 2x2^3\n");
  printf(" [Simple View Diagram of 2x2^3 Extra-Cubic Object]\n");
  for(d0=0;d0<2;d0++){
    for(d1=0;d1<2;d1++){
      for(d2=0;d2<2;d2++){
        if(d2>0){printf("   ");}
        for(d3=0;d3<2;d3++){printf("%6d",((d0*2+d1)*2+d2)*2+d3+1);}
        printf("\n");
      }
    }
    printf("\n");
  }
}
```

P46：四次元超立体二方陣の展開見取り図（基本図）

```
    1-----2
    |   3--+--4
    5--|--6   |
       7-----8

    9-----10
    |  11--+-12
   13--|-14   |
      15-----16
```

連続する自然数16個を２個ずつ小分けにして８行に描いただけの簡単なプログラムだが，適宜スペースを入れて行くと，まるで２×２×２の立方体を２個並べた図のように見える。これで四次元超立体二方陣の展開見取り図の１つを立派に描けているのだ。

自然数は，N=((d0*2+d1)*2+d2)*2+d3+1 の式で計算して合成している。まさに**二進数**の計算法だ。４重のループが**二進数**を発生しているから，こんなことができるのである。

それが証拠に次のような見事な「基本図」も，簡単なプログラムの追加で描けてしまう。

```
//
void pr47(){
 short d0,d1,d2,d3;
  printf("Study #47: Four Dimensional Extra-Cubic Object 2x2^3\n");
  for(d0=0;d0<2;d0++){
    printf(" D%d/\n",d0);
    for(d1=0;d1<2;d1++){
      for(d2=0;d2<2;d2++){
        if(d2>0){printf("  ");}
        for(d3=0;d3<2;d3++){printf("%4d ",((d0*2+d1)*2+d2)*2+d3+1);}//Change
        printf("  ");
        for(d3=0;d3<2;d3++){printf("%4d ",((d0*2+d1)*2+d2)*2+d3);}  //Change
        printf("  ");
        for(d3=0;d3<2;d3++){printf("  %d%d%d%d",d0,d1,d2,d3);}       //Change
        printf("  ");
        printf("%3d%3d  %3d%3d  %3d%3d  ",d0,d0,d1,d1,d2,d2);         //Change
        for(d3=0;d3<2;d3++){
          printf("%3d",d3);}                                         //Change
        printf("\n");
      }
    }
  }
  printf("  /Classic   /Mathematical  /N2i        /D2i\n");
}
```

P47：四次元超立体二方陣の展開見取り図（基本図）

近代的表記，二進法表記，二進法分解図を添付

```
 D0/
   1----2      0----1     0000  0001    0--0    0--0    0  0    0  1
   | 3--+-4    | 2--+-3     0010  0011  | 0+-0  | 0+-0    1  1    0  1
   5-|--6 |    4-|--5 |   0100  0101    0-|0 |  1-|1 |  0  0    0  1
     7----8      6----7     0110  0111    0--0    1--1    1  1    0  1
 D1/
   9---10      8    9     1000  1001    1--1    0  0    0  0    0  1
   |11--+12     10   11     1010  1011  | 1+-1    0  0    1  1    0  1
  13-|-14 |   12   13     1100  1101    1-|1 |  1  1    0  0    0  1
    15---16     14   15     1110  1111    1--1    1  1    1  1    0  1
```

／古典的表記　／近代的表記　／二進法表記図　／二進法分解図

二進法で表現すると，原点0000の隣に来る数が 0001，0010，0100，1000 の４つに限られることが見て取れる。古典的表記で言えば，１の隣に来る数は，２，３，５，９になる。

このプログラムでも，d0, d1, d2, d3 の順序を入れ替えるイタズラをしてみよう。どこをどう書き換えたのか，上のプログラムと比較研究してみてほしい。

```
//
void pr48(){
 short d0,d1,d2,d3;
  printf("\n");
  printf("Study #48: Another View of 4 Dimensional Extra-Cubic Object 2x2^3\n");
  for(d0=0;d0<2;d0++){
    printf(" D%d/          /N2i          /D2i/\n",d0);
    for(d1=0;d1<2;d1++){
      for(d2=0;d2<2;d2++){
        if(d2>0){printf("  ");}
        for(d3=0;d3<2;d3++){printf("%4d ",((d3*2+d2)*2+d1)*2+d0+1);}  //Change
        printf("   ");
       //
        for(d3=0;d3<2;d3++){printf("  %d%d%d%d",d3,d2,d1,d0);}         //Change
        printf("   ");
       //
        for(d3=0;d3<2;d3++){
          printf("%3d",d3);}                                          //Change
        printf("  %3d%3d  %3d%3d  %3d%3d",d2,d2,d1,d1,d0,d0);         //Change
        printf("\n");
      }
    }
  }
}
```

P48：四次元超立体二方陣の別角度から見た展開見取り図（基本図-2）

```
 D0/           /N2i           /D2i/
   1----9      0000  1000     0--1    0--0    0  0    0  0
   | 5--+13      0100  1100   | 0+-1  | 1+-1    0  0    0  0
   3-|-11 |     0010  1010     0-|1 |  0-|0 |  1  1    0  0
     7---15      0110  1110     0--1    1--1    1  1    0  0
 D1/           /N2i           /D2i/
   2---10      0001  1001     0--1    0  0    0  0    1  1
   | 6--+14      0101  1101   | 0+-1    1  1    0  0    1  1
   4-|-12 |     0011  1011     0-|1 |  0  0    1  1    1  1
     8---16      0111  1111     0--1    1  1    1  1    1  1
```

前の図と全然違った姿だが，興味深い結果となった。原点0000の隣に 1000，0100，0010，0001 が来ている点から見ても，これが求める基本図の別の１個になり得ることがわかる。

要するに，d0, d1, d2, d3 の順序を入れ替えただけで，別個の基本図が得られるのである。こういう生産的なイタズラは是非実験してみるものだ。

次のプログラムは，別の種類のイタズラを実験したものだ。d0, d1, d2, d3 の順序は入れ替えてはいないが，一部 for(.条件式.){...} 文の条件式をカウント・ダウン型に変えている。そうすると，何を作り出すか。

```
//
void pr49(){
 short d0,d1,d2,d3;
  printf("\n");
  printf("Study #49: Another View of 4 Dimensional Extra-Cubic Object 2x2^3\n");
  for(d0=0;d0<2;d0++){
```

```
    printf(" D%d/          /N2i          /D2i/\n",d0);
    for(d1=0;d1<2;d1++){
      for(d2=0;d2<2;d2++){
        if(d2>0){printf("  ");}
        for(d3=1;d3>=0;d3--){printf("%5d",((d0*2+d1)*2+d2)*2+d3+1);}  //Change
        printf("   ");
       //
        for(d3=1;d3>=0;d3--){printf("  %d%d%d%d",d0,d1,d2,d3);}        //Change
        printf("   ");
       //
        printf("%3d%3d  %3d%3d  %3d%3d  ",d0,d0,d1,d1,d2,d2);          //Change
        for(d3=1;d3>=0;d3--){
          printf("%3d",d3);}                                          //Change
        printf("\n");
      }
    }
  }
}
```

P48：四次元超立体二方陣のさらに別角度から見た展開見取り図（基本図-3）

```
 D0/           /N2i           /D2i/
    2----1     0001  0000     0--0    0--0    0  0    1  0
    | 4--+-3     0011  0010   | 0+-0  | 0+-0    1  1    1  0
    6-|--5 |   0101  0100     0-|0 |  1-|1 |  0  0    1  0
      8----7     0111  0110     0--0    1--1    1  1    1  0
 D1/           /N2i           /D2i/
   10----9     1001  1000     1--1    0  0    0  0    1  0
    |12--+11     1011  1010   | 1+-1    0  0    1  1    1  0
   14-|-13 |   1101  1100     1-|1 |  1  1    0  0    1  0
     16---15     1111  1110     1--1    1  1    1  1    1  0
```

原点に２(0001)が来た。隣の数は 0000，0011，0101，1001 で，位取りを無視すれば，原点と１ビット違う数が来ている。これも別の基本図と見てよいようだ。
他の場所を変えてみても，別個の基本図が得られることがわかった。

次は，**六次元二方陣**に思い切って挑戦してみよう。自然数２$^6$乗個（＝64 個）の二進法表現を６重のループで発生させなければならない。神経を使って大変だが，考え方は同じだ。イタズラの実験の仕方も同じだ。何が描けるだろうか。

```
//
void pr66(){
 short d0,d1,d2,d3,d4,d5,m;
  printf("\n");
  printf("Study #66: Six Dimensional Extra-Cubic Object 2^6=8x2^3\n");
  printf(" Classic/   /N2i            /D2i\n");
  for(d0=0;d0<2;d0++){
   for(d1=0;d1<2;d1++){
     for(d2=0;d2<2;d2++){
       for(d3=0;d3<2;d3++){
         for(d4=0;d4<2;d4++){
           if(d4==1){printf("  ");}
           for(d5=0;d5<2;d5++){
             m=((((d0*2+d1)*2+d2)*2+d3)*2+d4)*2+d5; printf("%3d  ",m+1);}
           for(d5=0;d5<2;d5++){printf("  %d%d%d%d%d%d",d0,d1,d2,d3,d4,d5);}
           printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
                  d0,d0,d1,d1,d2,d2,d3,d3,d4,d4,0,1);
           printf("\n");
```

```
          }}
        printf("\n");
      }}
  }
 //
  printf(" Another View:\n");
  for(d0=0;d0<2;d0++){
   for(d1=0;d1<2;d1++){
     for(d2=0;d2<2;d2++){
       for(d3=0;d3<2;d3++){
         for(d4=0;d4<2;d4++){
           if(d4==1){printf("  ");}
           for(d5=0;d5<2;d5++){
             m=((((d0*2+d5)*2+d3)*2+d4)*2+d1)*2+d2;    //Changed a little!
             printf("%3d  ",m+1);}
           for(d5=0;d5<2;d5++){printf("  %d%d%d%d%d%d",d0,d5,d3,d4,d1,d2);}
           printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
                  d0,d0,0,1,d3,d3,d4,d4,d1,d1,d2,d2); //Changed a little!
           printf("\n");
         }}
       printf("\n");
     }}
  }
 //
  printf(" Another View:\n");
  for(d0=0;d0<2;d0++){
   for(d1=0;d1<2;d1++){
     for(d2=0;d2<2;d2++){
       for(d3=0;d3<2;d3++){
         for(d4=0;d4<2;d4++){
           if(d4==1){printf("  ");}
           for(d5=1;d5>=0;d5--){
             m=((((d0*2+d1)*2+d2)*2+d3)*2+d4)*2+d5;    //Changed a little!
             printf("%3d  ",m+1);}
           for(d5=1;d5>=0;d5--){printf("  %d%d%d%d%d%d",d0,d1,d2,d3,d4,d5);}
           printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
                  d0,d0,d1,d1,d2,d2,d3,d3,d4,d4,1,0); //Changed a little!
           printf("\n");
         }}
       printf("\n");
     }}
  }
}
```

P66： 六次元二方陣の展開見取り図（基本図-1）

```
  古典的／      二進法表記／    二進法分解図／
 1----2    000000  000001    0--0    0--0    0  0    0  0    0  0    0  1
 | 3--+-4    000010  000011  | 0+-0  | 0+-0    0  0    0  0    1  1    0  1
 5-|--6 |  000100  000101    0-|0 |  0-|0 |  0  0    1  1    0  0    0  1
   7----8    000110  000111    0--0    0--0    0  0    1  1    1  1    0  1

 9---10    001000  001001    0--0    0  0    1  1    0  0    0  0    0  1
 |11--+12    001010  001011  | 0+-0    0  0    1  1    0  0    1  1    0  1
13-|-14 |  001100  001101    0-|0 |  0  0    1  1    1  1    0  0    0  1
  15---16    001110  001111    0--0    0  0    1  1    1  1    1  1    0  1

17   18    010000  010001    0  0    1  1    0  0    0  0    0  0    0  1
  19   20    010010  010011    0  0    1  1    0  0    0  0    1  1    0  1
21   22    010100  010101    0  0    1  1    0  0    1  1    0  0    0  1
  23   24    010110  010111    0  0    1  1    0  0    1  1    1  1    0  1
```

```
25   26    011000  011001    0  0    1  1    1  1    0  0    0  0    0  1
  27   28    011010  011011    0  0    1  1    1  1    0  0    1  1    0  1
29   30    011100  011101    0  0    1  1    1  1    1  1    0  0    0  1
  31   32    011110  011111    0  0    1  1    1  1    1  1    1  1    0  1

33   34    100000  100001    1  1    0  0    0  0    0  0    0  0    0  1
  35   36    100010  100011    1  1    0  0    0  0    0  0    1  1    0  1
37   38    100100  100101    1  1    0  0    0  0    1  1    0  0    0  1
  39   40    100110  100111    1  1    0  0    0  0    1  1    1  1    0  1

41   42    101000  101001    1  1    0  0    1  1    0  0    0  0    0  1
  43   44    101010  101011    1  1    0  0    1  1    0  0    1  1    0  1
45   46    101100  101101    1  1    0  0    1  1    1  1    0  0    0  1
  47   48    101110  101111    1  1    0  0    1  1    1  1    1  1    0  1

49   50    110000  110001    1  1    1  1    0  0    0  0    0  0    0  1
  51   52    110010  110011    1  1    1  1    0  0    0  0    1  1    0  1
53   54    110100  110101    1  1    1  1    0  0    1  1    0  0    0  1
  55   56    110110  110111    1  1    1  1    0  0    1  1    1  1    0  1

57   58    111000  111001    1  1    1  1    1  1    0  0    0  0    0  1
  59   60    111010  111011    1  1    1  1    1  1    0  0    1  1    0  1
61   62    111100  111101    1  1    1  1    1  1    1  1    0  0    0  1
  63   64    111110  111111    1  1    1  1    1  1    1  1    1  1    0  1
```

／別角度からの展開見取り図（基本図-2）

```
 1---17    000000  010000    0  0    0  1    0  0    0  0    0  0    0  0
 | 5--+21    000100  010100    0  0    0  1    0  0    1  1    0  0    0  0
 9-|-25 |  001000  011000    0  0    0  1    1  1    0  0    0  0    0  0
  13---29    001100  011100    0  0    0  1    1  1    1  1    0  0    0  0

 2   18    000001  010001    0  0    0  1    0  0    0  0    0  0    1  1
   6   22    000101  010101    0  0    0  1    0  0    1  1    0  0    1  1
10   26    001001  011001    0  0    0  1    1  1    0  0    0  0    1  1
  14   30    001101  011101    0  0    0  1    1  1    1  1    0  0    1  1

 3   19    000010  010010    0  0    0  1    0  0    0  0    1  1    0  0
   7   23    000110  010110    0  0    0  1    0  0    1  1    1  1    0  0
11   27    001010  011010    0  0    0  1    1  1    0  0    1  1    0  0
  15   31    001110  011110    0  0    0  1    1  1    1  1    1  1    0  0

 4   20    000011  010011    0  0    0  1    0  0    0  0    1  1    1  1
   8   24    000111  010111    0  0    0  1    0  0    1  1    1  1    1  1
12   28    001011  011011    0  0    0  1    1  1    0  0    1  1    1  1
  16   32    001111  011111    0  0    0  1    1  1    1  1    1  1    1  1

33   49    100000  110000    1  1    0  1    0  0    0  0    0  0    0  0
  37   53    100100  110100    1  1    0  1    0  0    1  1    0  0    0  0
41   57    101000  111000    1  1    0  1    1  1    0  0    0  0    0  0
  45   61    101100  111100    1  1    0  1    1  1    1  1    0  0    0  0

34   50    100001  110001    1  1    0  1    0  0    0  0    0  0    1  1
  38   54    100101  110101    1  1    0  1    0  0    1  1    0  0    1  1
42   58    101001  111001    1  1    0  1    1  1    0  0    0  0    1  1
  46   62    101101  111101    1  1    0  1    1  1    1  1    0  0    1  1

35   51    100010  110010    1  1    0  1    0  0    0  0    1  1    0  0
  39   55    100110  110110    1  1    0  1    0  0    1  1    1  1    0  0
43   59    101010  111010    1  1    0  1    1  1    0  0    1  1    0  0
  47   63    101110  111110    1  1    0  1    1  1    1  1    1  1    0  0
```

```
 36   52    100011  110011    1  1    0  1    0  0    0  0    1  1    1  1
   40   56    100111  110111    1  1    0  1    0  0    1  1    1  1    1  1
 44   60    101011  111011    1  1    0  1    1  1    0  0    1  1    1  1
   48   64    101111  111111    1  1    0  1    1  1    1  1    1  1    1  1
```

／別角度からの展開見取り図（基本図-3）

```
  2----1    000001  000000    0  0    0  0    0  0    0  0    0  0    1  0
  | 4--+-3    000011  000010    0  0    0  0    0  0    0  0    1  1    1  0
  6-|--5 |  000101  000100    0  0    0  0    0  0    1  1    0  0    1  0
    8----7    000111  000110    0  0    0  0    0  0    1  1    1  1    1  0

 10    9    001001  001000    0  0    0  0    1  1    0  0    0  0    1  0
   12   11    001011  001010    0  0    0  0    1  1    0  0    1  1    1  0
 14   13    001101  001100    0  0    0  0    1  1    1  1    0  0    1  0
   16   15    001111  001110    0  0    0  0    1  1    1  1    1  1    1  0

 18   17    010001  010000    0  0    1  1    0  0    0  0    0  0    1  0
   20   19    010011  010010    0  0    1  1    0  0    0  0    1  1    1  0
 22   21    010101  010100    0  0    1  1    0  0    1  1    0  0    1  0
   24   23    010111  010110    0  0    1  1    0  0    1  1    1  1    1  0

 26   25    011001  011000    0  0    1  1    1  1    0  0    0  0    1  0
   28   27    011011  011010    0  0    1  1    1  1    0  0    1  1    1  0
 30   29    011101  011100    0  0    1  1    1  1    1  1    0  0    1  0
   32   31    011111  011110    0  0    1  1    1  1    1  1    1  1    1  0

 34   33    100001  100000    1  1    0  0    0  0    0  0    0  0    1  0
   36   35    100011  100010    1  1    0  0    0  0    0  0    1  1    1  0
 38   37    100101  100100    1  1    0  0    0  0    1  1    0  0    1  0
   40   39    100111  100110    1  1    0  0    0  0    1  1    1  1    1  0

 42   41    101001  101000    1  1    0  0    1  1    0  0    0  0    1  0
   44   43    101011  101010    1  1    0  0    1  1    0  0    1  1    1  0
 46   45    101101  101100    1  1    0  0    1  1    1  1    0  0    1  0
   48   47    101111  101110    1  1    0  0    1  1    1  1    1  1    1  0

 50   49    110001  110000    1  1    1  1    0  0    0  0    0  0    1  0
   52   51    110011  110010    1  1    1  1    0  0    0  0    1  1    1  0
 54   53    110101  110100    1  1    1  1    0  0    1  1    0  0    1  0
   56   55    110111  110110    1  1    1  1    0  0    1  1    1  1    1  0

 58   57    111001  111000    1  1    1  1    1  1    0  0    0  0    1  0
   60   59    111011  111010    1  1    1  1    1  1    0  0    1  1    1  0
 62   61    111101  111100    1  1    1  1    1  1    1  1    0  0    1  0
   64   63    111111  111110    1  1    1  1    1  1    1  1    1  1    1  0
```

六次元超立体二方陣の場合には，小立方体が８個も並ぶので，スケールが大きくなるが，どうやらめでたく違った基本図が３個描けたようだ。

次は，**四次元超立体三方陣**に挑戦してみよう。
今度は，自然数$3^4$(＝81)個の三進法表現を４重のループで発生させなければならない。

```
//** Sample of Basic View-Forms of Developed ECO3^4 **
//** File: 'StudyECO34.c' Dictated by K.Setsuda on Apr.17, 2005; *
//
#include <stdio.h>
//
int main(){
 short n,CC;
 short d0,d1,d2,d3;
 short e0,e1,e2,e3;
```

```
 printf("\n");
 printf("** Sample of Basic View-Forms of Developed Extra-Cubic Object 3^4 **\n");
 printf("#1\n");
 for(d0=0;d0<3;d0++){
  for(d1=0;d1<3;d1++){
    for(n=d1;n>=0;n--){printf("  ");}
    for(d2=0;d2<3;d2++){
      for(d3=0;d3<3;d3++){
        printf("%2d    ",
               ((d0*3+d1)*3+d2)*3+d3+1);} //Original
      printf("  ");}
    printf("\n");}}
//
 printf("#2\n");
 for(d0=0;d0<3;d0++){
  for(d1=0;d1<3;d1++){
    for(n=d1;n>=0;n--){printf("  ");}
    for(d2=0;d2<3;d2++){
      for(d3=0;d3<3;d3++){
        printf("%2d    ",
               ((d0*3+d1)*3+d3)*3+d2+1);} //Changed
      printf("  ");}
    printf("\n");}}
//
 printf("#3\n");
 for(d0=0;d0<3;d0++){
  for(d1=0;d1<3;d1++){
    for(n=d1;n>=0;n--){printf("  ");}
    for(d2=0;d2<3;d2++){
      for(d3=0;d3<3;d3++){
        printf("%2d    ",
               ((d2*3+d0)*3+d1)*3+d3+1);} //Changed
      printf("  ");}
    printf("\n");}}
//
 CC=2;
 printf("#4\n");
 for(d0=0;d0<3;d0++){e0=CC-d0;
  for(d1=0;d1<3;d1++){e1=CC-d1;
    for(n=d1;n>=0;n--){printf("  ");}
    for(d2=0;d2<3;d2++){e2=CC-d2;
      for(d3=0;d3<3;d3++){e3=CC-d3;
        printf("%2d    ",
               ((d0*3+d1)*3+d2)*3+e3+1);} //Changed
      printf("  ");}
    printf("\n");}}
//
 printf("#5\n");
 for(d0=0;d0<3;d0++){e0=CC-d0;
  for(d1=0;d1<3;d1++){e1=CC-d1;
    for(n=d1;n>=0;n--){printf("  ");}
    for(d2=0;d2<3;d2++){e2=CC-d2;
      for(d3=0;d3<3;d3++){e3=CC-d3;
        printf("%2d    ",
               ((e0*3+e1)*3+e2)*3+e3+1);} //Changed
      printf("  ");}
    printf("\n");}}
//
 printf(" [OK!]\n");
 printf("\n");
```

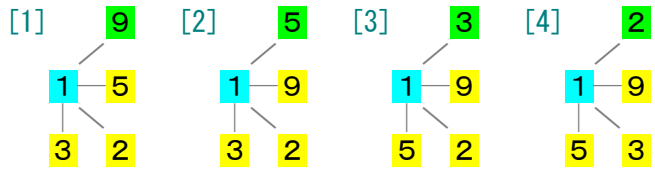
```
 return 0;
}
//

#1
  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
   |37  |   38     |39  |    |40  |   41     |42  |    43      44      45
   |   46      47   |   48    |   49      50   |   51      52      53      54
  55----|-56------57    |  58----|-59------60    |  61      62      63
    64  |    65      66  |    67  |    68      69  |    70      71      72
      73------74------75      76------77------78      79      80      81
#2
  1------ 4------ 7       2       5       8       3       6       9
   |10      13     |16     11      14      17      12      15      18
   |   19------22--+--25      20      23      26      21      24      27
  28    | 31      34    |  29      32      35      30      33      36
   |37  |   40     |43  |    38      41      44      39      42      45
   |   46      49   |   52      47      50      53      48      51      54
  55----|-58------61    |  56      59      62      57      60      63
    64  |    67      70  |    65      68      71      66      69      72
      73------76------79      74      77      80      75      78      81
#3
  1       2       3       28      29      30      55      56      57
    4       5       6       31      32      33      58      59      60
      7       8       9       34      35      36      61      62      63
  10      11      12      37      38      39      64      65      66
    13      14      15      40      41      42      67      68      69
      16      17      18      43      44      45      70      71      72
  19      20      21      46      47      48      73      74      75
    22      23      24      49      50      51      76      77      78
      25      26      27      52      53      54      79      80      81
#4
    3       2       1       6       5       4       9       8       7
     12      11      10      15      14      13      18      17      16
       21      20      19      24      23      22      27      26      25
   30      29      28      33      32      31      36      35      34
     39      38      37      42      41      40      45      44      43
       48      47      46      51      50      49      54      53      52
   57      56      55      60      59      58      63      62      61
     66      65      64      69      68      67      72      71      70
       75      74      73      78      77      76      81      80      79
#5
  81      80      79      78      77      76      75      74      73
    72      71      70      69      68      67      66      65      64
      63      62      61      60      59      58      57      56      55
  54      53      52      51      50      49      48      47      46
    45      44      43      42      41      40      39      38      37
      36      35      34      33      32      31      30      29      28
  27      26      25      24      23      22      21      20      19
    18      17      16      15      14      13      12      11      10
       9       8       7       6       5       4       3       2       1
  . . . . . .
```

最後の２つは，カウントダウン型の for ループを，以前とは別の仕方で実現している。
今回もどうやら成功したようだ。

この方法で，何個もの違った基本図が描けることがわかったのは，大きい。
基本図が増えると，目的方陣の「構造」が良くわかるようになる。構成のための条件式なども，正確に設計することができるようになる。各論に与える恩恵が大きい。

## ２．基本図は，何個あるのか。最低限何個必要なのか。

超立体二方陣の「基本図」は，相互に違ったものが全部でいったい幾つあるのだろうか。
４次元の場合，原点の隣に来れる数は，４軸上に４個ある。その並び方は，順列で計算して，${}_4P_4 = 4 \times 3 \times 2 \times 1 = 24$ 通りある。
一方，基本図の小立方体の二方陣は三次元であるから，実際には ${}_3P_3 = 3 \times 2 \times 1 = 6$ 通りしか表現できない。この６通りは，我々が直視できる姿なので，同一視できる。
従って，小立方体二方陣を基本図に用いる限り，$24 \div 6 = 4$ 通りの基本図を最小限用意する必要があることになる。例えば，以下のように始まる基本図を用意すればよい。




それを１で述べたプログラムで作る場合には，d0,d1,d2,d3 の順序を入れ替えながら，違った基本図を作る。その順列の数は，${}_4P_4 = 4 \times 3 \times 2 \times 1 = 24$ 通り。三次元で見て回転・反転形を省くと，４通りの違った基本図が残る。

### ＊　四次元超立体二方陣の展開見取り図（基本図４個）＊

```
 /古典的 /二進法表記
 1---5   0000 0100 | 1---9   0000 1000 | 1---9   0000 1000 | 1---9   0000 1000
  | 2-+-6   0001 0101 |  | 2-+10   0001 1001 |  | 2-+10   0001 1001 |  | 3-+11   0010 1010
  3-|-7 | 0010 0110 |  3-|11 | 0010 1010 |  5-|13 | 0100 1100 |  5-|13 | 0100 1100
    4---8   0011 0111 |    4--12   0011 1011 |    6--14   0101 1101 |    7--15   0110 1110
                      |                       |                       |
  9--13   1000 1100 |  5  13   0100 1100 |  3  11   0010 1010 |  2  10   0001 1001
  |10-+14   1001 1101 |    6  14   0101 1101 |    4  12   0011 1011 |    4  12   0011 1011
 11-|15 | 1010 1110 |  7  15   0110 1110 |  7  15   0110 1110 |  6  14   0101 1101
   12--16   1011 1111 |    8  16   0111 1111 |    8  16   0111 1111 |    8  16   0111 1111
```

その整理の結果を示した上図を見てほしい。
これらの基本図をもとにして，具体的な四次元超立体二方陣の構成に着手すればよい。

では，**六次元**の場合はどうか。同じ手順で考えてみよう。
原点１の隣には，６軸上に２，３，５，９，17，33 の６数が来る。その並び方は，${}_6P_6 = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$ 通り。従って，n1＝１の基本図も 720 通り描ける。
一方，小立方体二方陣は，${}_3P_3 = 3 \times 2 \times 1 = 6$ 通りの並び方を表現できるから，小立方体二方陣を基本図に用いる限り，$720 \div 6 = 120$ 通りの基本図を用意しなければならないことがわかる（６個から３個を取る順列で計算しても，${}_6P_3 = 6 \times 5 \times 4 = 120$）。

### ＊　六次元超立体二方陣の展開見取り図の例（基本図３個）

```
 /古典的   /二進法表記
 1--17    000000   010000 | 1--17    000000   010000 | 1--33    000000   100000
  | 2-+18    000001   010001 |  | 2-+18    000001   010001 |  | 9-+41    001000   101000
  3-|19 |  000010   010010 |  9-|25 |  001000   011000 | 17-|49 |  010000   110000
    4--20    000011   010011 |   10--26    001001   011001 |   25--57    011000   111000
                              |                              |
```

```
 5--21    000100   010100  |  3  19    000010   010010  |  2  34    000001   100001
 | 6-+22    000101   010101  |    4  20    000011   010011  |  10  42    001001   101001
 7-|23 |   000110   010110  | 11  27    001010   011010  | 18  50    010001   110001
   8--24    000111   010111  |  12  28    001011   011011  |  26  58    011001   111001
                              |                             |
 9  25    001000   011000  |  5  21    000100   010100  |  3  35    000010   100010
  10  26    001001   011001  |    6  22    000101   010101  |  11  43    001010   101010
11  27    001010   011010  | 13  29    001100   011100  | 19  51    010010   110010
  12  28    001011   011011  |  14  30    001101   011101  |  27  59    011010   111010
                              |                             |
13  29    001100   011100  |  7  23    000110   010110  |  4  36    000011   100011
  14  30    001101   011101  |    8  24    000111   010111  |  12  44    001011   101011
15  31    001110   011110  | 15  31    001110   011110  | 20  52    010011   110011
  16  32    001111   011111  |  16  32    001111   011111  |  28  60    011011   111011
                              |                             |
33  49    100000   110000  | 33  49    100000   110000  |  5  37    000100   100100
  34  50    100001   110001  |  34  50    100001   110001  |  13  45    001100   101100
35  51    100010   110010  | 41  57    101000   111000  | 21  53    010100   110100
  36  52    100011   110011  |  42  58    101001   111001  |  29  61    011100   111100
                              |                             |
37  53    100100   110100  | 35  51    100010   110010  |  6  38    000101   100101
  38  54    100101   110101  |  36  52    100011   110011  |  14  46    001101   101101
39  55    100110   110110  | 43  59    101010   111010  | 22  54    010101   110101
  40  56    100111   110111  |  44  60    101011   111011  |  30  62    011101   111101
                              |                             |
41  57    101000   111000  | 37  53    100100   110100  |  7  39    000110   100110
  42  58    101001   111001  |  38  54    100101   110101  |  15  47    001110   101110
43  59    101010   111010  | 45  61    101100   111100  | 23  55    010110   110110
  44  60    101011   111011  |  46  62    101101   111101  |  31  63    011110   111110
                              |                             |
45  61    101100   111100  | 39  55    100110   110110  |  8  40    000111   100111
  46  62    101101   111101  |  40  56    100111   110111  |  16  48    001111   101111
47  63    101110   111110  | 47  63    101110   111110  | 24  56    010111   110111
  48  64    101111   111111  |  48  64    101111   111111  |  32  64    011111   111111
                              |                             |
```

プログラムで描かせる場合でも，d0,d1,d2,d3,d4,d5 の並び替えは${}_6P_6$＝720 通りある。三次元で見て反転・回転形を省くと，720÷６＝120 通りの基本図が残る。

これは，ちょっと気の遠くなるような数字だ。120×８個もの小立方体を描き並べるのは，つい気が引けてしまう。今は，最初の数個を示すのみとしよう（上図）。

後は，各論の中で刈り込みと整理を試みる。条件式を立てる目的のためだけならば，実際には最大でも 20 通りの基本図を描けば用が足りるからだ。

**四次元超立体三方陣**の場合は，どうなのだろう。

原点１の隣に来る数は，２，４，10，28 の４個に限られる。その並び替えは，${}_4P_4$＝４×３×２×１＝24 通り。小立方体で基本図を描く限り，${}_3P_3$＝３×２×１＝６通り→

＊　四次元超立体三方陣の展開見取り図（基本図-1）＊

```
  1-----  2-----  3          4-----  5-----  6          7-----  8-----  9
   |10      11     |12        |13      14     |15        |16      17     |18
  28   19-29--20-30--21    31   22-32--23-33--24    34   25-35--26-36--27
   |37 |    38     |39 |     |40 |    41     |42 |     |43 |    44     |45 |
 55--46-56--47-57   48    58--49-59--50-60   51    61--52-62--53-63   54
    64 |    65      66 |      67 |    68      69 |      70 |    71      72 |
      73-----74-----75          76-----77-----78          79-----80-----81
```

/三進法表記

```
0000   0001   0002     0010   0011   0012     0020   0021   0022
  0100   0101   0102     0110   0111   0112     0120   0121   0122
    0200   0201   0202     0210   0211   0212     0220   0221   0222
1000   1001   1002     1010   1011   1012     1020   1021   1022
  1100   1101   1102     1110   1111   1112     1120   1121   1122
    1200   1201   1202     1210   1211   1212     1220   1221   1222
2000   2001   2002     2010   2011   2012     2020   2021   2022
  2100   2101   2102     2110   2111   2112     2120   2121   2122
    2200   2201   2202     2210   2211   2212     2220   2221   2222
 . . . . .
```

→を表現できるから，24÷６＝４通りの基本図を用意すればよいことになる。
四次元ならば，二方陣でも三方陣でも，この辺は変わりがないことがわかる。

では，**八次元超立体二方陣**の場合は，どうか。

＊　八次元超立体二方陣の展開見取り図（基本図-1）

```
  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 |  37  38    39  40    41  42    43  44    45  46    47  48
   49--50    51--52    53  54    55  56    57  58    59  60    61  62    63  64

 65--66    67  68    69  70    71  72    73  74    75  76    77  78    79  80
  |81--+82   83  84    85  86    87  88    89  90    91  92    93  94    95  96
 97-|98 |  99 100   101 102   103 104   105 106   107 108   109 110   111 112
  113-114   115 116   117 118   119 120   121 122   123 124   125 126   127 128

129 130   131 132   133 134   135 136   137 138   139 140   141 142   143 144
  145 146   147 148   149 150   151 152   153 154   155 156   157 158   159 160
161 162   163 164   165 166   167 168   169 170   171 172   173 174   175 176
  177 178   179 180   181 182   183 184   185 186   187 188   189 190   191 192

193 194   195 196   197 198   199 200   201 202   203 204   205 206   207 208
  209 210   211 212   213 214   215 216   217 218   219 220   221 222   223 224
225 226   227 228   229 230   231 232   233 234   235 236   237 238   239 240
  241 242   243 244   245 246   247 248   249 250   251 252   253 254   255 256
 . . . . .
```

自然数２$^{8}$（＝256）個の魔方陣だから，スケールが大きい。二進数を発生させるには，プログラムで８重のループが必要になる。
.....

あとは宿題にしよう。読者自身が自分で考えてみてほしい。
次節からは，各論を掘り下げて，厳密な議論を展開して行こう。

（著者：摂田 寛二(Kanji Setsuda)；jag12001@nifty.com;
初稿の執筆・計数・作表 with MacOSX & Xcode 1.5 on April 21, 2005;
最新清書稿 with MacOSX 10.10.2 and Xcode 6.1.1 on March 2, 2015）