JAPLA 研究会資料　2009/3/28

Ｊによるペントミノ・パズルーその１
データ構造とアルゴリズム

西川　利男

１．ポリオミノとペントミノ・パズル[1]

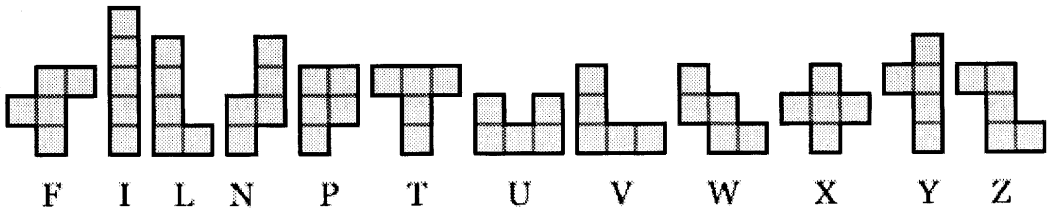
複数の正方形を辺と辺を接して接続させた図形をポリオミノ（polyomino）という。正方形の数に応じて、ドミノ（２個）、トロミノ（３個）、テトロミノ（４個）、ペントミノ（５個）、ヘキソミノ（６個）などと呼ぶ。

ポリオミノはハーバード大学の大学院生だったソロモン・ゴロム（S. W. Golomb）が創案したもので、彼はまた「ポリオミノ」を著わしている。

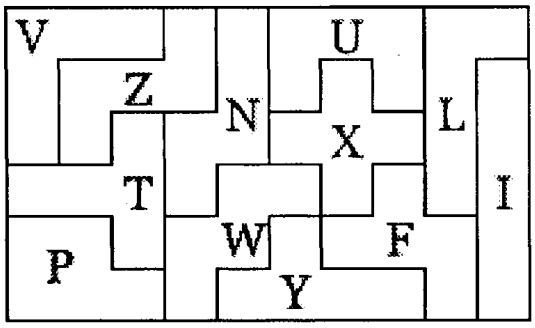
日本人にとって昔からなじみの四畳半、六畳など畳の敷き方はドミノである。このようなポリオミノである図形を敷き詰める問題は数が大きくなるとなかなか難しい。

ポリオミノ･パズルにはすべて同じピースで敷き詰める同形ポリオミノ･パズルと異形のポリオミノ･パズルがある。その中で最もよく知られているのが異形のペントミノ・パズルで、プラパズル No.5 として市販されていた。筆者が初台の東京工業試験所（筑波に移転後、その跡地は現在、新国立劇場オペラハウスとなる）に勤務していた頃、研究室の仲間で研究もそっちのけで、みんなでうつつを抜かしていた。当時の FACOM 270-20 で 2339 通りと計算されたとある。最近、また買い求めにいったら、東急ハンズで「食べられない明治チョコレート」という名前で売っていた。

そのペントミノ・パズルとはゴロムによって以下のように名付けられた１２種のピースを６×１０のワクの中にどうやってピッタリ詰めるかというものである。



一つの解は右のようになる。



今回は、Ｊでペントミノ・パズルを解くための基本のデータ構造、アルゴリズムなどについて述べる。

[1] 池野、高木、土橋、中村「数理パズル」中公新書(1976). p.151-160

２．ペントミノ・パズルをＪでどう表現するか

Ｊで「ペントミノ・パズルをどう解くか？」という問題は、「ペントミノ・パズルをＪでどう表現するか？」にかかっている。言い換えると「データ構造」と「アルゴリズム」をどう記述するかの問題となる。Ｊのプログラムは最後にまとめて示した。

２．１　単位ピースの表現と回転、鏡映

まず、各ピースの図形は0, 1の配列として表す。例えば「Ｐ」、「Ｆ」、「Ｖ」はつぎのような配列になる。

```
   P_0
1 1
1 1
1 0
   F_0
0 1 1
1 1 0
0 1 0
   V_0
1 0 0
1 0 0
1 1 1
```

しかし、これらを基本としてピースは回転させたり、裏返したりできるので、形はといろいろに変わる。そのための操作の関数をつぎのように定義した。

```
t0 =: ]               NB. 0
t1 =: |.⍉(1)@|:       NB. 90  deg. right turn
t2 =: |.⍉(1)@|.       NB. 180 deg. right turn
t3 =: |.@|:           NB. 270 deg. right turn
t4 =: |.⍉(1)          NB. mirror
t5 =: |.⍉(1)@(|.@|:)  NB. mirror & 90 deg. right turn
t6 =: |.              NB. mirror & 180 deg. right turn
t7 =: |:              NB. mirror & 270 deg. right turn
```

これを使って、例えば、つぎのようになる。

```
   t1 P_0
1 1 1
0 1 1
   t2 P_0
0 1
1 1
1 1
   t1 F_0
0 1 0
1 1 1
0 0 1
   t2 F_0
0 1 0
0 1 1
```

1 1 0

したがって、各ピースの回転鏡映可能な形は関数 pen を使って、つぎのようなる。

```
   pen =: ~.@(t0;t1;t2;t3;t4;t5;t6;t7)

   pen P_0
+---+-----+---+-----+---+-----+---+-----+
|1 1|1 1 1|0 1|1 1 0|1 1|0 1 1|1 0|1 1 1|
|1 1|0 1 1|1 1|1 1 1|1 1|1 1 1|1 1|1 1 0|
|1 0|      |1 1|      |0 1|      |1 1|     |
+---+-----+---+-----+---+-----+---+-----+
   pen F_0
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 1 1|0 1 0|0 1 0|1 0 0|1 1 0|0 0 1|0 1 0|0 1 0|
|1 1 0|1 1 1|0 1 1|1 1 1|0 1 1|1 1 1|1 1 0|1 1 1|
|0 1 0|0 0 1|1 1 0|0 1 0|0 1 0|0 1 0|0 1 1|1 0 0|
+-----+-----+-----+-----+-----+-----+-----+-----+
   pen V_0
+-----+-----+-----+-----+
|1 0 0|1 1 1|1 1 1|0 0 1|
|1 0 0|1 0 0|0 0 1|0 0 1|
|1 1 1|1 0 0|0 0 1|1 1 1|
+-----+-----+-----+-----+
```

ここで実際には、各ピースは6×10のワクの中に配置した配列でなければならない。

```
   p_data =: (6 10)&{. L:0 @pen
```

例えばピース「P」の可能な初期配列はつぎのようになる。

```
   PP =: p_data P_0
   PP
+------------------+------------------+
|1 1 0 0 0 0 0 0 0 0|1 1 1 0 0 0 0 0 0 0|
|1 1 0 0 0 0 0 0 0 0|0 1 1 0 0 0 0 0 0 0|
|1 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
+------------------+------------------+

（途中省略）

+------------------+
|1 1 1 0 0 0 0 0 0 0|
|1 1 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|
+------------------+
```

２．２　ピースの移動

つぎにピースの移動をどうやって行なうか、である。関数 shift を用いて、例えば、右に2、下に1への移動はつぎのようになる。

```
   2 1 shift > 0{PP
0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

さらに6×10のワクの中で、可能な移動のすべては関数membersにより示される。

```
   members > 0{PP
+------------------+------------------+------------------+
|1 1 0 0 0 0 0 0 0 0|0 1 1 0 0 0 0 0 0 0|0 0 1 1 0 0 0 0 0 0|
|1 1 0 0 0 0 0 0 0 0|0 1 1 0 0 0 0 0 0 0|0 0 1 1 0 0 0 0 0 0|
|1 0 0 0 0 0 0 0 0 0|0 1 0 0 0 0 0 0 0 0|0 0 1 0 0 0 0 0 0 0|   (途中省略)
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
+------------------+------------------+------------------+
```

```
                +------------------+------------------+------------------+
                |0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
                |0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
(途中省略)      |0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0|
                |0 0 0 0 0 0 1 1 0 0|0 0 0 0 0 0 0 1 1 0|0 0 0 0 0 0 0 0 1 1|
                |0 0 0 0 0 0 1 1 0 0|0 0 0 0 0 0 0 1 1 0|0 0 0 0 0 0 0 0 1 1|
                |0 0 0 0 0 0 1 0 0 0|0 0 0 0 0 0 0 1 0 0|0 0 0 0 0 0 0 0 1 0|
                +------------------+------------------+------------------+
```

この場合、可能な移動な配置は３６組ある。このようにしてピースの移動を処理する。

２．３　ピースの接触/非接触

ピースをピッタリと敷き詰める最も重要なポイントである。
簡単なデータで接触/非接触をどう検出するか、テストして見てみよう。

```
NB. test contact
tA =: 1 1 0 0 0
tB =: 0 0 1 1 0
tC =: 0 0 0 1 1
tD =: 0 1 1 0 0
NB. tA contact tB => 1: contact
NB. tA contact tC => 0: non contact(vacant)
NB. tA contact tD => 0: non contact(overlap)
```

データtAに対して、つぎの演算操作をして見る。

```
  AA =: 2 -~/¥  tA
tA   1  1  0  0  0
AA    0  _1  0  0
```

これは、２つずつの組に対して、等しいときは0、等しくないとき1→0 と下がれば_1、0→1と上がれば1を返す。つまり、変化の階差値を与える。
同じ、操作をデータtBについてもやってみる。

```
tB   0  0  1  1  0
```

```
BB    0  1  0  _1
```
階差値_1 と 1 の場合に接触し、この検出には AA と BB とをそれぞれ掛ければ良い。
```
   AA * BB
 0 _1 0 0
```
この結果を見ると、_1 となる位置で tA と tB とが接触している。
今度はデータ tC でやってみる。
```
tC   0  0   0  1  1
CC    0   0  1  0
   AA * CC
 0 0 0 0
```
すべて 0 なので tA と tC とは離れていて接触していない。
一方、データ tD は tA と重なっている。このときは
```
tA + tD   1  2  1  0  0
```
となる。つまり結果に 2 があれば重なっている。
関数 contact では、まず重なりのテストを行い、重なりがないときは接触/非接触のテストを行い、その接触の個数を返すようにした。さらに関数 tcontact で接触のインデックスを得るようにした。

ペントミノのデータでそのようすを見てみよう。
```
  P0 =: >0{PP
  NN =: p_data N_0
  N0 =: >0{NN
```
例えば「P」と「N」のそのままの位置ではつぎのようになる。
```
  P0
1 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
  N0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
  P0 contact N0
0
```
つまり、重なっている。
そこで、関数 shift により「N」を右に 1 だけ移動する。
```
  N1 =: 1 0 shift N0
  N1
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
```

```
0 1 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
   P0 contact N1
3
```
すると、「Ｐ」と「Ｎ」とは3箇所で接触していることがわかる。
　さらに、「Ｎ」を右に移動してみる。
```
   N2 =: 2 0 shift N0
   N2
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
   P0 contact N2
0
```
今度は、「Ｐ」と「Ｎ」とは離れて接触していない。このようにして、ピッタリ接触する箇所を見つけることができた。

３．いろいろなユーザ･インターフェースなど

　パズルとして楽しむためには、入出力のインターフェースがかかせない。

３．１　文字によるピースの表示

　各ピースの種類を表示して、配置を示すため0/1配列を文字で表示するための関数displayを作った。
```
  display =: 3 : 0
:
CHA =. '.',x.
y.{CHA
)
```
　これを使うと先の0/1配列は文字で示される。
```
   'P' display P0
PP........
PP........
P.........
..........
..........
..........
   'N' display N1
..N.......
..N.......
.NN.......
.N........
..........
```

```
..........
```
さらに、これらの配列を一つにまとめには、つぎの関数 rp を使う。
```
rp =: 3 : 0 "0
:
if. ('.' = x.) *. ('.' = y.) do. z =. '.' end.
if. '.' = x. do. z =. y. end.
if. '.' = y. do. z =. x. end.
z
)
   ('P' display P0) rp ('N' display N1)
PPN.......
PPN.......
PNN.......
.N........
..........
..........
```
このようにして、接触の状態が目に見える形で示された。

３．２　ピース・ワクの表示

上のままでも良いが、パズルの進行状況をはっきり示すため、各ピースの周囲をけい線で囲むようにして、さらに見やすくする関数 frame を作った。そのプログラムの詳細はやや複雑であり、末尾の定義を参照されたい。

```
   frame ('P' display P0) rp ('N' display N1)
+--------------------------------------+
| P   P | N | .   .   .   .   .   .   . |
|   .   .   .   .   .   .   .   .   .  |
| P   P | N | .   .   .   .   .   .   . |
|   +---+   .   .   .   .   .   .   .  |
| P | N   N | .   .   .   .   .   .   . |
|---+   +---+   .   .   .   .   .   .  |
| . | N | .   .   .   .   .   .   .   . |
|   +---+   .   .   .   .   .   .   .  |
| . . . . . . . . . . . . . . . . . . . |
|   .   .   .   .   .   .   .   .   .  |
| . . . . . . . . . . . . . . . . . . . |
+--------------------------------------+
```

３．３　パズルの実行プログラム

以上のいろいろな準備をした上で、パズルを実際に即した方法で入力して行くようにメインプログラムを作成する。

ピースの種類を選んでから、その回転・鏡映などの向きを決める。接触/非接触をテストしながら、左上から右下に向かって配置して行く。それぞれの途中のピースパターンを図示する。　続行、中断、やり直しの指示をしつつ、進めて行く。

パズルのプログラムは、数値計算などと異なり、データ構造とアルゴリズムなど、どうプログラム化するか、がおおきな鍵となっている。それと並んでユーザ・インターフェースが楽しむため面倒だが、大切である。

ウィンドウズ画面により、入力したり、ピースを色分け表示したりするのは、なおプログラム中であり、次回以降にしたい。
また、今回は自動的に解を求めるようにはなっていない。解の探索はそれだけで、一つの仕事であり、非常に興味ある問題だが、今後ゆっくりと検討したい。

以下には、最初にあげた例解の途中経過を含めた実行例を示す。

４．ペントミノ・パズルの実際

```
   pentomino 'V'
initial piece:
+---+---+---+---+
|0  |1  |2  |3  |
+---+---+---+---+
|V..|VVV|VVV|..V|
|V..|V..|..V|..V|
|VVV|V..|..V|VVV|
+---+---+---+---+
enter piece number ?
1
+-------------------------------------+
| V   V   V | .   .   .   .   .   .   . |
|   +---.---+   .   .   .   .   .   .   |
| V | .   .   .   .   .   .   .   .   . |
| . .   .   .   .   .   .   .   .   .   |
| V | .   .   .   .   .   .   .   .   . |
|---+   .   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
|   .   .   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
|   .   .   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
+-------------------------------------+
select pentomino piece ?
Z
+---+---+---+---+
|0  |1  |2  |3  |
+---+---+---+---+
|ZZ.|..Z|.ZZ|Z..|
|.Z.|ZZZ|.Z.|ZZZ|
|.ZZ|Z..|ZZ.|..Z|
+---+---+---+---+
enter piece number ?
1
i=0, Hit: Z(1)
Next:
+-------------------------------------+
| V   V   V | Z | .   .   .   .   .   . |
|   +---.---+   .   .   .   .   .   .   |
| V | Z   Z   Z | .   .   .   .   .   . |
| .   +---.---+   .   .   .   .   .   .   |
| V | Z | .   .   .   .   .   .   .   . |
|---.---+   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
|   .   .   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
|   .   .   .   .   .   .   .   .   .   |
| . . . . . . . . . . . . . . . . . . . |
+-------------------------------------+
continue(CR), undo(u), break(n) ?

select pentomino piece ?
N
```

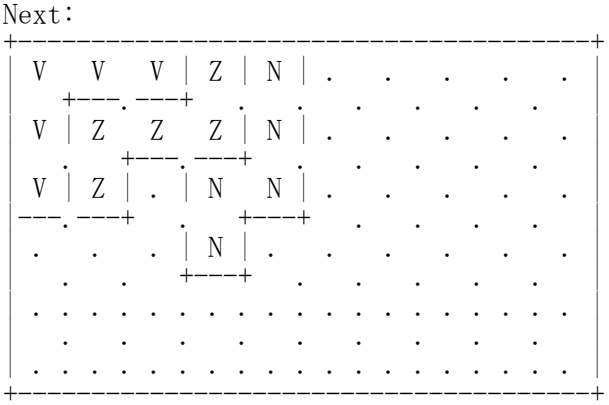
```
+--+----+--+----+--+----+--+----+
|0 |1   |2 |3   |4 |5   |6 |7   |
+--+----+--+----+--+----+--+----+
|.N|NN..|.N|NNN.|N.|.NNN|N.|..NN|
|.N|.NNN|NN|..NN|N.|NN..|NN|NNN.|
|NN|    |N.|    |NN|    |.N|    |
|N.|    |N.|    |.N|    |.N|    |
+--+----+--+----+--+----+--+----+
```

enter piece number ?

0

i=1, Hit: N(3)

Next:



continue(CR), undo(u), break(n) ?

select pentomino piece ?

U

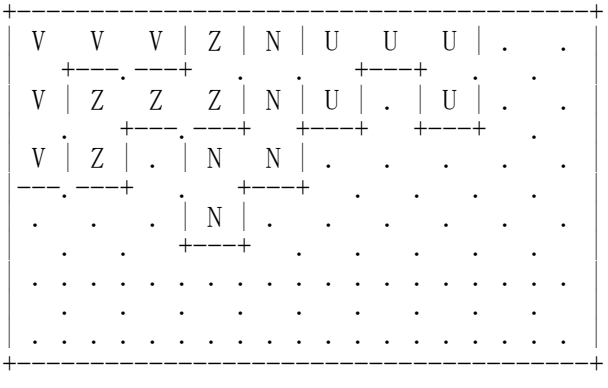
```
+---+--+---+--+
|0  |1 |2  |3 |
+---+--+---+--+
|U.U|UU|UUU|UU|
|UUU|U.|U.U|.U|
|   |UU|   |UU|
+---+--+---+--+
```

enter piece number ?

2

i=2, Hit: U(5)

Next:



continue(CR), undo(u), break(n) ?

（途中省略）

```
select pentomino piece ?
P
+--+---+--+---+--+---+--+---+
|0 |1  |2 |3  |4 |5  |6 |7  |
+--+---+--+---+--+---+--+---+
|PP|PPP|.P|PP.|PP|.PP|P.|PPP|
|PP|.PP|PP|PPP|PP|PPP|PP|PP.|
|P.|   |PP|   |.P|   |PP|   |
+--+---+--+---+--+---+--+---+
enter piece number ?
3
i=9, Hit: P(32)
Next:
+-----------------------------------+
| V   V   V | Z | N | U   U   U | L   L |
|   +---.---+   .   .   +---+   .   +---|
| V | Z   Z   Z | N | U | X | U | L | I |
|   .   +---.---+   +---+   +---+   .   |
| V | Z | T | N   N | X   X   X | L | I |
|---.---+   .   +---.---+   +---+   .   |
| T   T   T | N | W   W | X | F | L | I |
|---.---+   +---+   +---.---+   +---+   |
| P   P | T | W   W | . | F   F   F | I |
|   .   +---+   +---+   +---.---+   .   |
| P   P   P | W | .   .   .   . | F | I |
+-----------------------------------+
continue(CR), undo(u), break(n) ?

select pentomino piece ?
Y
+--+----+--+----+--+----+--+----+
|0 |1   |2 |3   |4 |5   |6 |7   |
+--+----+--+----+--+----+--+----+
|.Y|..Y.|Y.|YYYY|Y.|YYYY|.Y|.Y..|
|YY|YYYY|Y.|.Y..|YY|..Y.|.Y|YYYY|
|.Y|    |YY|    |Y.|    |YY|    |
|.Y|    |Y.|    |Y.|    |.Y|    |
+--+----+--+----+--+----+--+----+
enter piece number ?
7
i=10, Hit: Y(32)
Next:
+-----------------------------------+
| V   V   V | Z | N | U   U   U | L   L |
|   +---.---+   .   .   +---+   .   +---|
| V | Z   Z   Z | N | U | X | U | L | I |
|   .   +---.---+   +---+   +---+   .   |
| V | Z | T | N   N | X   X   X | L | I |
|---.---+   .   +---.---+   +---+   .   |
| T   T   T | N | W   W | X | F | L | I |
|---.---+   +---+   +---.---+   +---+   |
| P   P | T | W   W | Y | F   F   F | I |
|   .   +---+   +---+   +---.---+   .   |
| P   P   P | W | Y   Y   Y   Y | F | I |
+-----------------------------------+
```

プログラム・リスト

```
NB. Pentomino Program by Toshio Nishikawa
NB.
NB. Pentomino Data Make 2009/2/13
NB. turn and mirror operations
t0 =: ]                NB. 0
t1 =: |."(1)@|:        NB. 90  deg. right turn
t2 =: |."(1)@|.        NB. 180
t3 =: |.@|:            NB. 270
t4 =: |."(1)           NB. mirror
t5 =: |."(1)@(|.@|:)   NB. mirror 90
t6 =: |.               NB. mirror 180
t7 =: |:               NB. mirror 270

NB. make all of probable operations
NB.   eg. pen L_0
pen =: ~.@(t0;t1;t2;t3;t4;t5;t6;t7)
NB. make pentomino data
NB.   eg p_data L_0
p_data =: (6 10)&{. L:0 @pen

NB. pentomino elemnts
F_0 =: 3 3$0 1 1 1 1 0 0 1 0
I_0 =: 5 1$#5
L_0 =: 4 2$1 0 1 0 1 0 1 1
N_0 =: 4 2$0 1 0 1 1 1 1 0
P_0 =: 3 2$1 1 1 1 1 0
T_0 =: 3 3$1 1 1 0 1 0 0 1 0
U_0 =: 2 3$1 0 1 1 1 1
V_0 =: 3 3$1 0 0 1 0 0 1 1 1
W_0 =: 3 3$1 0 0 1 1 0 0 1 1
X_0 =: 3 3$0 1 0 1 1 1 0 1 0
Y_0 =: 4 2$0 1 1 1 0 1 0 1 0 1
Z_0 =: 3 3$1 1 0 0 1 0 0 1 1

NB. display 0,1 array using characters
NB.   'A' display A0
NB. AA......
NB. A.......
NB. A.......
display =: 3 : 0
:
CHA =. '.',x.
y.{CHA
)
```

```
shift =: 3 : 0
:
DA =. y.
DAXY =. $DA
DA =. DA, DAXY$0
'X Y' =. x.
DA =. (-X) |."(1) DA
DA =. (-Y) |. DA
DA =. ({.DAXY) {. DA
(}.DAXY) {."(1) DA
)

members =: 3 : 0
Y =. y.
'K L' =. $Y
N =. +/, Y
I =. 0
J =. 0
Z =. ''
while. J<K
  do.
    while. I<L
      do.
NB.          wr I, J
NB.          wr ZA =. (I, J) shift Y
         ZA =. (I, J) shift Y
         Z =. Z,<ZA
         if. 1 = +./{:"(1) ZA do. break. end.
         I =. I + 1
NB.          if. 'n' = rd 1 do. '*** break' return. end.
    end.
    if. 1 = +./{:"(2) ZA do. break. end.
NB.    if. 'n' = rd 1 do. '*** break' return. end.
    I =. 0
    J =. J + 1
end.
Z
)

tcontact =: 3 : 0
:
TC =. x. contact L:0 y.
(1&<:) L:0 TC
)
```

```
NB. TX contact L:0 L => return number of contacted case
NB.    2009/2/6
contact =: 3 : 0
:
NB. non overlap ovla = 1 -> OK , overlap ovla = 0 -> NO
ovla =. */ 2&>"(0) , x. +"(0) y.
if. 0 = ovla do. ovla return. end.
NB. wr 'OK'
XX =: 2 -~/¥ "(1) x.
YY =: 2 -~/¥ "(1) y.
XY =. XX * YY
XYF =. - +/ , XY
, XYF
)
NB. unite character and period
NB. 'A.' rp '.F' => 'AF'
rp =: 3 : 0 "0
:
if. ('.' = x.) *. ('.' = y.) do. z =. '.' end.
if. '.' = x. do. z =. y. end.
if. '.' = y. do. z =. x. end.
z
)

ch2flag =: 3 : 0"0
if. '.' = y. do. 0 else. 1 end.
)


wr =: 1!:2&2
rd =: 1!:1


NB. frame display version
pentomino =: 3 : 0
CH =. (-.&' ')@(-.&',') y. NB. remove comma & space
YA =. {.CH
wr 'initial piece:'
frame > DisIYA =. YA display L:0 pen ". YA,'_0'
wr ((<"(0)@(i.@#)) ,: ]) DisIYA
PN =. rd 1 [ wr 'enter piece number ?'
IYA =. (". PN) { p_data ". YA,'_0'
wr frame > DisA =. < YA display >IYA
```

```
i =. 0
while. i < 12
  do.
    NB. IY =. i{YY
    IY =. rd 1 [wr 'select pentomino piece ?'
  label_undo.
    DisIY =. IY display L:0 pen ". IY,'_0'
    wr ((<"(0)@(i.@#)) ,: ]) DisIY
    PN =. rd 1 [ wr 'enter piece number ?'
    MIY =. members > (".PN) { p_data ". IY,'_0'
    tF =. IYA tcontact MIY
    tIDX =. (, 1=>tF) i. 1
    GET =. tIDX { MIY
    DisGet =. IY display L:0 GET
    wr 'i=', (":i),', Hit: ', IY, '(', (": tIDX), ')'
    Next =. DisA (rp &. >) {. DisGet
    wr 'Next:'
    wr frame > Next
NB. select next operation
    yn =. rd 1 [wr 'continue(CR), undo(u), break(n) ?'
    if. 0 = #yn do. goto_next. end.
    if. 'u' = yn do. goto_undo. end.
    if. 'n' = yn
       do. wr '*** break'
           IYA
           return.
       end.
  label_next.
    DisA =. Next
    GETI =. ch2flag L:0 Next
    IYA =. GETI
    i =. i+1
end.
'*** end ***'
)


frame =: 3 : 0
F1 =. y.
F2 =. 2 < ¥"(2) F1
F3 =. =/ L:0 F2
F4 =. > F3
NB. F5 is horizontal lines
F5 =. (F4{'-'),"(2) '='
F6 =. F1,"(1) F5
```

```
F7 =. }: ((2*{.$F1),({:$F1))$,F6
FZ =: spmi L:0 <"(0) F7
NB. FZ is array with horizontal lines
NB. wr $FZ
NF =: {. $FZ NB. Global Value called from chalin
MF =: {: $FZ NB. Global Value called from chalin
NB. revised 1/31
DF =: (NF, 3*MF)$;FZ
DG =. (MF, NF, 3)$,DF
NB. CD is vertical lines
CD =: chalin FZ
CG =. (MF, NF, 1)$,CD,"(1)'&'
ZA =. DG,"(1) CG
ZZ =. (NF, (4*MF))$, ZA
< }:"(1) ZZ
)

NB. space or minus character called by frame1
spmi =: 3 : 0
sp =. ' '&,@ (,&' ')
mi =. '-'&,@ (,&'-')
if. '-' = y. do. mi y. else. sp y. end.
)

NB. Select according character or symbol line
NB. ex. chalin F8
chalin =: 3 : 0
PA =: 2 < ¥"(1) y.
PB =: -:/ "(1)> PA
PP =: <"(1)(NF, 3*MF) $;>y.
PF =: ((123&>) *.(>&64))@(a.&i.) L:0 PP
PG =: +/ L:0 PF
NB. PGB =. (,. ;PG),"(1) PB
PH =: ;PG
PDD =. ''
i =. 0
while. i < #PH
  do.
    if. 0 < i{PH
       do.   PDD =. PDD, (i{PB){"(1)'| ' NB. character
       else. PDD =. PDD, (i{PB){"(1)'+.' NB. line
    end.
    i =. i + 1
end.
(NF, ({:$PB))$PDD
```

)