bit Hoyle

計算譜

西村恕彦
Nisimura Hirohiko
電子技術総合研究所
田中裕一
Tanaka Yuichi
桜井彰人
Sakurai Akihito

カルキュレーションまたはコンピュータ。トランプの一人遊び。下手がやるとまったく成功しないが，上手がやれば9割成功するという。プログラム間の試合が計画されているが，ここでは戦略を含まない規則だけを示す。

規　則

目標　4つの台 *foundation* に，次のような列を昇順に作ること。

A，2，3，4，5，6，…，K
2，4，6，8，10，Q，…，K
3，6，9，Q，2，5，…，K
4，8，Q，3，7，J，…，K

開始　52枚の札をよく切って，裏返しにしたままで，山 *stock* におく。

山をめくる　山から1枚めくって手 *hand* にもつ。よく状況 *situation* を考えて，次の動作をする。

台につける　4つの台のどれかにつけることができるときには，つけてもよい（つけなくてもよい）。

屑におく　台につけ（られ）なかったときには，4つの屑 *talon* のどれかに表向きにおく。

これをすませたら，次には札を移すことができる（本によっては，手を見ながら札を移すことが許されるかのように説明してあるが，ここではそれを禁止する）。

札を移す　屑の天の札のうちに，台につけることのできるものがあれば，それを移してもよい。これは何回でも行なってよいし，まったく行なわなくてもよい。そして，また山をめくりに行く。

終了の判定　台の4つの列が完成し，表面にキングが4枚並べば成功。山が空っぽならば，屑から台に札を移せるだけ移す。どうしても移せず，屑が残り，台が完成しなければ，失敗。

COBOL 算譜の機能

算譜はこの規則の純粋な執行機構として作ったので，意思決定ははいっていない。二つの資源を利用するようになっている（図1参照）。

一つは山 *stock* であって，1枚ずつ手 *hand* をめくって計算譜に渡してくれる。計算譜は山を全面的に信頼しているので，検査は行なっていない。たとえば同じ数の札が5枚以上きたり，52枚未満で空っぽになったりしても，そのまま受け入れてしまう。ただし，受け取る情報は1から13までの数と99（山が尽きた印）だけであって，それ以外の情報は無視する。（実験するには，山は別にファイルしておくのがいちばん便利だった。しかし遊ぶためにはちゃんとした乱数，本当のトランプをしながらだと入力端末がよい。）

もう一つの資源は意思決定 *decision* である。計算譜から意思決定に，1枚の手を教える。意思決定はその手をどの屑 *talon* におくか，またはどの台 *foundation* につけるか *whereto* を決め，計算譜に返答する。屑の列は番号1～4，台の列は番号5～8で指定する。計算譜はこの番号を検査する。たとえばつけることのできない台

図1　構造

```
 TURN  OUT
?08
 1.  5  6
 2.  3  5
 3.  4 11  7  7 10  9  1 12 11  5  9 10
 4. 13 13 13
 5.  1
 6.  2  4  6
 7.  3  6  9
 8.  4  8 12
          TO 1
          TO 2
          TO 3
          TO 4
          TO 6
 WHERETO(08)
?6
 1.  5  6
 2.  3  5
 3.  4 11  7  7 10  9  1 12 11  5  9 10
 4. 13 13 13
 5.  1
 6.  2  4  6  8
 7.  3  6  9
 8.  4  8 12
    FROM 2        TO 8
  FROMTO
?  TURN  OUT
?08
```

図2　途中経過（COBOL）

表1　COBOL 計算譜の概要（数字はおよその行数）

データ部	30
準備 *starting*	20
開始 *play-game*	4
山をめくる *turn-up*	40
札を移す *move-card*	30
終了の判定 *test-result*	20
移せるか *inspection*	10
状況の説明 *display-situation*	40
誤りの注意 *warn-error*	5

を指定すると，それを無視し，再び番号を要求する。

また，札を移すときにも，意思決定を要求する。このときには返答は，どの屑 *from-where* から，どの台 *to-where* に移すかの，二つの番号となる。ゼロを返答すると，さしあたり札を移さない，という意味になる。山がまだあれば山をめくりに行く。山が空っぽで，移せる札があれば，再度意思決定を要求する。

なお，計算譜が意思決定に対して要求を出すときには，参考のために余分な情報も添える。それは，そのときの状況 *situation*，すなわち屑と台の様子と，手をどこにつける（おく）ことができるか，または札をどの屑からどの台に移すことができるかの番号である。意思決定はこれらの情報のなかから返答を選ぶべきである（図2）。

意思決定はブラウン管の端末に向かって座っている人間を想定したので，参考情報はしょっちゅう出る。ほかの環境だったら，参考情報は削減すべきだろう。

COBOL 作譜作法

手続きは，8つの副手続き *section* に分けて構成した。制御の移行がやたらにピョンピョン跳ねまわらないように心掛けた。つまり，ループや複雑な分岐による制御の移行は，それぞれの副手続きのなかに閉じ込めてしまい，各副手続きはただ一つの入り口とただ一つの出口をもつようにした。（そしてこの問題では，それが可能だった。）この結果，大きな制御の移行は，上から下へ順次に進む。ただし例外があって，終了の判定の副手続きは，大きなループの反復点が入れ子になっている。これを入れ子にするために，条件がはなはだ複雑になった。条件の判定を簡潔にしなかったのは，構造化プログラミングの風潮におもねったのである（図3）。

小さなループの形式をそろえることも考えてみたが，うまくゆかなかった。COBOL にはループのための文というものがない（!）ので，たとえば ADR 社ではいくつかの文を拡張しているくらいである。

```
115000
117000 IDENTIFICATION  DIVISION.
119000  PROGRAM-ID.  CALCULATION.
121000 ENVIRONMENT  DIVISION.
123000  CONFIGURATION  SECTION.
125000   SPECIAL-NAMES.
127000     REMOTE  IS  DECISION.
129000     REMOTE  IS  STOCK.
131000 DATA  DIVISION.
133000 WORKING-STORAGE  SECTION.
135000 01          WHERETO                          PIC  9(1).
137000 01  FROM-TO.
139000   20        FROM-WHERE                       PIC  9(1).
141000   20        TO-WHERE                         PIC  9(1).
143000 01          HAND                             PIC  9(2).
145000 01          CARD                             PIC  9(2).
147000 01          MOVABLE                          PIC  9(1).
149000 01  STACK-LENGTH-AREA.
151000   20        STACK-LENGTH   OCCURS  8          PIC  9(2).
153000 01  FILLER                 REDEFINES  STACK-LENGTH-AREA.
155000   10        T-LENGTH                         PIC  X(8).
157000   10        F-LENGTH                         PIC  X(8).
159000 01  TALON-AREA.
161000   10  TALON-SEQUENCE       OCCURS  8.
163000       20    TALON          OCCURS  53        PIC  9(2).
165000 01  FOUNDATION-AREA        REDEFINES  TALON-AREA.
167000   10  FOUNDATION-SEQUENCE  OCCURS  8.
169000       20    FOUNDATION     OCCURS  53        PIC  9(2).
171000 01          COUNT-STACK                      PIC  9(1).
173000 01  SUBSCRIPTS-AREA.
175000   20        I                                PIC  9(1).
177000   20        J                                PIC  9(2).
179000   20        M                                PIC  9(1).
181000   20        N                                PIC  9(2).
183000   20        J-SCREEN                         PIC  9(2).
185000 01  SCREEN-AREA.
187000   20        STACK-NUMBER                     PIC  Z(2),.
189000   20        SCREEN  OCCURS 18                PIC  ZZ9.
191000
193000 PROCEDURE  DIVISION.
195000 STARTING  SECTION.
197000  PREPARE-THE-FOUNDATION.
199000     MOVE      ZERO                 TO        TALON-AREA.
201000     MOVE      4                    TO        M.
203000  LOOP-FOUNDATIONS.
205000     ADD       1                    TO        M.
207000     IF        M  GREATER  8        GO  TO    END-LOOP-FOUNDATIONS.
209000     MOVE      ZERO                 TO        N.
211000     MOVE      ZERO                 TO        CARD.
213000  LOOP-N.
215000     ADD       1                    TO        N.
217000     IF        N  GREATER  13       GO  TO    END-LOOP-N.
219000     ADD       -4  M                TO        CARD.
221000     IF        CARD  GREATER  13
223000       ADD     -13                  TO        CARD.
225000     MOVE      CARD                 TO        FOUNDATION (M  N).
227000     GO  TO    LOOP-N.
229000  END-LOOP-N.
231000     MOVE      88                   TO        FOUNDATION (M  N).
233000     GO  TO    LOOP-FOUNDATIONS.
235000  END-LOOP-FOUNDATIONS.
237000     EXIT.
239000**************
241000 PLAY-GAME  SECTION.
243000  CLEAR-SEQUENCES.
245000     MOVE      "0101010100000000"   TO        STACK-LENGTH-AREA.
247000**************++
```

図3（a）COBOL 計算譜

二つの点について覚悟をはっきり決めずにコーディングに着手したので，途中で迷いが生じ，だいぶ時間と労力を浪費した。一つは屑と台であって，これを区別せずに8列の棚 *stack* とするつもりが最初あって，それをいったん別にし，またいっしょにしたので，現状のような不徹底なものになった。もう一点は添字であって，はじめは自由に添字式を使って書き，大型機ならたいてい通るだろうと多寡を括ったのが，案に相違してまったくだめで，あわてて指標に切り替え，やはり不自由なのと添字でも行数が増えないことがわかったのとで，けっきょく添字にもどった。結果はほとんど JIS どおりになったのだが，添字式や関数が使えないのは，この種の問題では痛かった。

手続きのうちで，山をめくる *turn-up* と札を移す *move-card* の主要部50行ほどをエンドレスループの形

でざっとコーディングして走らせ，あとは，トップダウン法でつけくわえていった。コーディング・打鍵・デバッグ・テストにちょうど2日かかった。そのあと，表示形式などの微調整に半日。感想。コーディングとデバッグの能力が老衰していることに悲哀を覚えた。200行という長さにもショックを感じたが，短縮する元気はもうなかった。算譜はすべて小文字で書いてあって読みやすいのだが，小文字の印字装置がなくて残念(表1)。

発　端

bit Hoyle の勧進元である島内剛一先生から，カルキュレーションの規則を書けとの仰せ付け。遊び方を知らないからと断ったら，じゃ教えるから，とただ1回だけ教えてもらい，あとはコーディングをしながら電話をかけて，規則の細部やら用語やらを確認した。そのほか参考にしたのは，藤崎哲之助：トランプの一人遊び（**bit** 1973年10月号）と野崎昭弘：とらんぷ（ダイヤモンド社)。下手っぴいの成功率は1～2割といわれているが　どうしてどうして，ぜんぜん成功しない。

計算譜が一通りできたところで，野崎先生の著者自訂本の例に従って追試。どうにか成功にこぎつけて，微調整。ついでに小さな誤植もみつけた。

カルキュレーションの(意思決定の)プログラムの試合が計画されている。挑戦される方は，本稿末尾のお知らせをみること。単純なアルゴリズムだけの問題ではないので，はたして実戦可能な意思決定のプログラムを作ることができるものかどうか。筆者には挑戦する気はさらさらない。

（以上：西村記）

```
249000 TURN-UP  SECTION.
251000 TAKE-A-STOCK-CARD.
253000     DISPLAY  "   TURN  OUT     "  UPON       STOCK.
255000     ACCEPT   HAND                 FROM       STOCK.
257000     IF       HAND  =  99          GO  TO     TURNED.
259000     IF       HAND  GREATER  13  OR  LESS  1
261000       GO TO  TAKE-A-STOCK-CARD.
263000 ASK-WHERE-TO.
265000     PERFORM  DISPLAY-SITUATION.
267000     MOVE     ZERO                 TO         M.
269000 DISPLAY-AVAILABLE.
271000     ADD      1                    TO         M.
273000     IF       M  GREATER  8        GO  TO     MAKE-QUESTION.
275000     ADD      1  STACK-LENGTH (M)  GIVING     N.
277000     IF       M  LESS  5  OR  FOUNDATION (M  N)  =  HAND
279000       DISPLAY  "          TO "  M  UPON      DECISION.
281000     GO  TO   DISPLAY-AVAILABLE.
283000 MAKE-QUESTION.
285000     DISPLAY  "  WHERETO(" HAND ")"  UPON     DECISION.
287000     ACCEPT   WHERETO               FROM       DECISION.
289000 IS-TALON.
291000     IF       WHERETO  GREATER  ZERO  AND  LESS  5
293000       ADD    1                    TO         STACK-LENGTH (WHERETO)
295000       MOVE   STACK-LENGTH (WHERETO)  TO      J
297000       MOVE   HAND                 TO         TALON (WHERETO  J)
299000       GO TO  TURNED.
301000 IS-FOUNDATION.
303000     IF       WHERETO  GREATER  4  AND  LESS  9
305000       ADD    1  STACK-LENGTH (WHERETO)  GIVING  N
307000       IF     FOUNDATION (WHERETO  N)  =  HAND
309000         ADD  1                    TO         STACK-LENGTH (WHERETO)
311000         GO TO  TURNED.
313000 ERROR-DECISION.
315000     PERFORM  WARN-ERROR.
317000     GO  TO   ASK-WHERE-TO.
319000 TURNED.
321000     EXIT.
323000*************
325000 MOVE-CARD  SECTION.
327000 REQUEST-DECISIONS.
329000     PERFORM  DISPLAY-SITUATION.
331000     MOVE     ZERO                 TO         MOVABLE.
333000     PERFORM  INSPECTION
335000              VARYING  I  FROM  1  BY  1  UNTIL  I GREATER 4
337000              AFTER    M  FROM  5  BY  1  UNTIL  M GREATER 8.
339000     IF       MOVABLE  =  ZERO     GO  TO     MOVED.
341000     DISPLAY  "   FROMTO        "  UPON       DECISION.
343000     ACCEPT   FROM-TO              FROM       DECISION.
345000     IF       FROM-TO  =  SPACE  OR  ZERO
347000       GO TO  MOVED.
349000     IF       FROM-WHERE  LESS  1  OR  GREATER  4
351000       OR     TO-WHERE    LESS  5  OR  GREATER  8
353000       PERFORM  WARN-ERROR
355000       GO TO  REQUEST-DECISIONS.
357000 MOVE-THE-CARD.
359000     MOVE     STACK-LENGTH (FROM-WHERE)   TO   J.
361000     ADD      1  STACK-LENGTH (TO-WHERE)  GIVING  N.
363000     IF       TALON (FROM-WHERE  J) = FOUNDATION (TO-WHERE  N)
365000*      MOVE   ZERO       TO         TALON (FROM-WHERE  J)
367000       ADD    -1         TO         STACK-LENGTH (FROM-WHERE)
369000       ADD    1          TO         STACK-LENGTH (TO-WHERE)
371000     ELSE
373000       PERFORM  WARN-ERROR.
375000     GO  TO   REQUEST-DECISIONS.
377000 MOVED.
379000     EXIT.
381000*************
```

図3 (b) COBOL 計算譜

bit

図6の算譜で審判譜は JUDGE(算程)であり，その下請けとして CONS, COPY(算程)を使っている。また，ここには戦略譜として TERMNL(算程)をつけた。これは TSS 端末を介して人間がゲームを行なうときのためのものである。さらに，主譜および SHUFFL(算程)，RANDOM(関数)を実際の使用例として掲げた。

以下に各単譜の概略を示す。

(1) JUDGE　戦略譜 PLAYER と札の列 STOCK とを入力とし，結果 WIN を出力とする。審判譜は，規則に従って戦略譜が取り得るすべての手を求め (0120～0410行)，戦略譜に提示する (0430～0550行)。戦略譜はそのうちの1つを選ぶことができるだけであって，台と屑の操作および札をめくる操作は審判譜が行なう (0560～0740行)。52枚の札が尽きれば終りである (0770～0810行)。配列 FOUND, TALON はそれぞれ4列ずつの台と屑の山であり，PFND, PTLN は，各山の一番上(天)の札を指示するポインタである。AVAIL は，すべての可能な着手（たかだか24通り）をリストアップしたものであり，各手について次の3種類の情報を与えている(図5)。

操作の種類　1……札を屑におく
　　2……札を台につける
　　3……屑の札を1枚，台に移す
　　4……次の札をめくる

受取り側の番号　操作が1，2，3の場合，札を受け取る側の山（屑または台）の番号を示す

送出し側の番号　操作が3の場合，屑の番号を示す

ここで番号は，条件に該当しない場合に0になっている。

(2) CONS, COPY　前者は配列 AVAIL に新しい着手を登録していくもの，後者は配列を定数倍してコピ


```
383000 TEST-RESULT  SECTION.
385000  IF-STOCK-EXAUSTED.
387000     IF       MOVABLE  =  1  AND  HAND  =  99
389000       DISPLAY  "  TRY MORE;  STOCK EXAUSTED  "
391000                                  UPON       DECISION
393000       GO TO  MOVE-CARD.
395000  NEXT-HAND-FROM-STOCK.
397000     IF       F-LENGTH  =  "13131313"  OR  HAND  =  99
399000       NEXT  SENTENCE
401000     ELSE
403000       GO TO  TURN-UP.
405000  IF-COMPLETE-OR-EXAUSTED.
407000     IF       F-LENGTH  =  "13131313"
409000       DISPLAY  "YOU WON THE GAME,  CONGRATULATIONS"
411000                                  UPON       DECISION
413000     ELSE
415000       DISPLAY  "  Y C U  L O S T  T H E  G A M E "
417000                                  UPON       DECISION.
419000  GAME-SET.
421000     PERFORM  DISPLAY-SITUATION.
423000     DISPLAY  "     N E W   G A M E   S T A R T E D
425000                                  UPON       DECISION.
427000     GO  TO   PLAY-GAME.
429000**************
431000 INSPECTION  SECTION.
433000  MAKE-A-COMBINATION.
435000     MOVE     STACK-LENGTH (I)    TO         J.
437000     ADD      1  STACK-LENGTH (M)  GIVING     N.
439000     IF       TALON (I  J)  =  FOUNDATION (M  N)
441000       MOVE   1                   TO         MOVABLE
443000       DISPLAY  "       FROM "  I  "       TO "  M
445000                                  UPON       DECISION.
447000 EXIT-INSPECTION.
449000     EXIT.
451000**************
453000 DISPLAY-SITUATION  SECTION.
455000  START-STACK-SCAN.
457000     MOVE     ZERO                TO         COUNT-STACK.
459000  LOOP-TALONS.
461000     ADD      1                   TO         COUNT-STACK.
463000     IF       COUNT-STACK  GREATER  4  GO TO  END-LOOP-TALONS.
465000     MOVE     STACK-LENGTH (COUNT-STACK)  TO  J.
467000     MOVE     SPACE               TO         SCREEN-AREA.
469000     MOVE     COUNT-STACK         TO         STACK-NUMBER.
471000     MOVE     ZERO                TO         J-SCREEN.
473000  LOOP-J.
475000     ADD      1                   TO         J-SCREEN.
477000     IF       J-SCREEN  GREATER  18
479000       OR     J  LESS  2          GO  TO     END-LOOP-J.
481000     MOVE     TALON (COUNT-STACK  J)  TO     SCREEN (J-SCREEN).
483000     ADD      1                   TO         J.
485000     GO  TO   LOOP-J.
487000  END-LOOP-J.
489000     DISPLAY  SCREEN-AREA         UPON       DECISION.
491000     GO  TO   LOOP-TALONS.
493000  END-LOOP-TALONS.
495000     MOVE     4                   TO         COUNT-STACK.
497000  LOOP-FOUNDATIONS.
499000     ADD      1                   TO         COUNT-STACK.
501000     IF       COUNT-STACK GREATER 8  GO TO  END-LOOP-FOUNDATIONS.
503000     MOVE     SPACE               TO         SCREEN-AREA.
505000     MOVE     COUNT-STACK         TO         STACK-NUMBER.
507000     MOVE     ZERO                TO         N.
509000  LOOP-N.
511000     ADD      1                   TO         N.
513000     IF       N  GREATER  STACK-LENGTH (COUNT-STACK)
515000       GO TO  END-LOOP-N.
517000     MOVE     FOUNDATION (COUNT-STACK  N)  TO  SCREEN (N).
519000     GO  TO   LOOP-N.
521000  END-LOOP-N.
523000     DISPLAY  SCREEN-AREA         UPON       DECISION.
525000     GO  TO   LOOP-FOUNDATIONS.
527000 END-LOOP-FOUNDATIONS.
529000     EXIT.
531000**************
533000 WARN-ERROR  SECTION.
535000  RELEASE-A-MESSAGE.
537000     DISPLAY  "...YOUR DECISION IS ILLEGAL..."
539000                                  UPON       DECISION.
541000**************
```

図3 (c) COBOL 計算譜

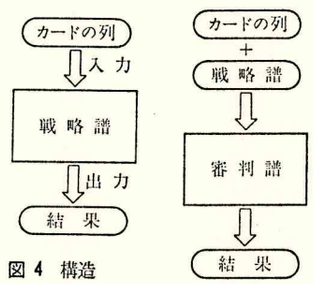

図4 構造

```
FOUNDATION
  [1] A 2 3
  [2] 2 4 6 8
  [3] 3 6 9 Q
  [4] 4 8 Q 3 7 J

TALON
  [1] 6 5 8 5 Q
  [2] 5 3 9 8
  [3] T 9 5 J Q A 9 T 7 7 J 4 6
  [4] K K K K

CARD  (44)
      ..... 2


( N)     --- OPERATION ---

( 1)   CARD      -> FOUND(3)
( 2)   CARD      -> FOUND(4)
( 3)   CARD      -> TALON(1)
( 4)   CARD      -> TALON(2)
( 5)   CARD      -> TALON(3)
( 6)   CARD      -> TALON(4)

INPUT N
=
```

図5 途中経過 (FORTRAN)

FORTRAN 計算譜

カルキュレーションの算譜の環境は，実際にゲームを行なう戦略譜（戦略を考え，手を与える算譜――人間でもよい）に対し，あらかじめ定められた52枚の札の列を順次入力すると，その結果が“成功”または“失敗”として得られるゲームと考えることができる(図4)。審判譜とは，このような戦略譜と札の列とを入力とし，戦略譜によるゲームの進行をシミュレートして，その結果を求めるような算譜である。審判譜はゲームの規則を全部知っていて，戦略譜が規則に違反する手を打つことを許さない。その意味で審判譜はゲームの規則を記述している。

ーするものである。

（3） 主譜（CALCULATION） 端末から初期値を読み込み，これに従って擬似乱数列を作り，札を混ぜる。そして戦略譜と札の列とを入力として審判譜に渡す。審判譜からの出力は，統計をとるのに便利なように1，0としてある。戦略譜が複数個あるときも，それらを順次審判譜に渡して結果の統計がとれるようになっている。

（4） RANDOM 区間（0, 1）の擬似一様乱数を発生する。

（5） SHUFFL 一様乱数を利用して52枚の札の乱列を作る。

（6） TERMNL 台および屑の内容を TSS 端末に表示して，取り得る手を示す。人間は手の番号を入力するだけである。 （以上：田中記）

LISP 計算譜

屑と台の列はそれぞれ T1, T2, T3, …, F3, F4 という名前のリストにした（図8）。列の天の札は CAR によって得られ，ポップアップは CDR によってなされる。

手の札をおく（つける）場所の指示は，上記の列の名前のどれか一つを引数として与える。屑から台へ札を移す操作の指示も一つの引数であるが，これは次のようなリストである。

NIL

を書くと，札を移さないで，次の札をめくりに行く。

((T3 F4) (T3 F4) (T1 F4))

というふうに書くと，これらの一連の操作をして，次の札をめくりに行く。これでは実は人間にとって具合が悪いので，たとえば次のようにも書ける。

((T2 F1) (T2 F1) (X))

Xのところは何でもよいから，移せる指示や NIL 以外のものにしておくと，その直前までの操作をしてから，再び次の指示を受けにくる（図7）。

```
      SUBROUTINE JUDGE(PLAYER,STOCK,WIN)
      IMPLICIT INTEGER (A-Z)
      INTEGER FOUND(13,4),PFND(4)
      INTEGER TALON(52,4),PTLN(4)
      INTEGER TOP(4)
      INTEGER STOCK(52)
      INTEGER AVAIL(24,3)
      INTEGER CFND(52),CTLN(208),CPFND(4),CPTLN(4),CAVAI(72)
      CALL COPY(0,PFND,PFND,4)
      CALL COPY(0,PTLN,PTLN,4)
      DO 400 COUNT=1,52
         K=0
         CARD=STOCK(COUNT)
         DO 20 I=1,4
            FND=1
            IF(PFND(I).GE. 1) FND=MOD(FOUND(PFND(I),I)+FND-1,13)+1
            IF(PFND(I).GE.13) FND=-1
            IF(CARD.EQ.FND) CALL CONS(AVAIL,K,2,+I,0)
   20       CONTINUE
         DO 30 I=1,4
            CALL CONS(AVAIL,K,1,+I,0)
   30       CONTINUE
         GOTO 70
   40    CONTINUE
         K=0
         DO 50 I=1,4
            FND=I
            IF(PFND(I).GE. 1) FND=MOD(FOUND(PFND(I),I)+FND-1,13)+1
            IF(PFND(I).GE.13) FND=-1
            TOP(I)=FND
   50       CONTINUE
         DO 60 I=1,4
            TLN=0
            IF(PTLN(I).GT.0) TLN=TALON(PTLN(I),I)
            IF(TLN.EQ.TOP(1)) CALL CONS(AVAIL,K,3,1,+I)
            IF(TLN.EQ.TOP(2)) CALL CONS(AVAIL,K,3,2,+I)
            IF(TLN.EQ.TOP(3)) CALL CONS(AVAIL,K,3,3,+I)
            IF(TLN.EQ.TOP(4)) CALL CONS(AVAIL,K,3,4,+I)
   60       CONTINUE
         IF(K.EQ.0) GOTO 400
         CALL CONS(AVAIL,K,4,0,0)
   70    CONTINUE
         CALL COPY(1,FOUND,CFND, 52)
         CALL COPY(1,TALON,CTLN,208)
         CALL COPY(1,PFND,CPFND,4)
         CALL COPY(1,PTLN,CPTLN,4)
         CALL COPY(1,AVAIL,CAVAI,72)
         ILLE=0
   80    CONTINUE
         IF(ILLE.GT.0) PRINT *, 'ILLEGAL OPERATION!
         IF(ILLE.GT.2) GOTO 500
         ILLE=ILLE+1
         CALL PLAYER
     &      (+COUNT,CFND,CPFND,CTLN,CPTLN,+CARD,CAVAI,+K,N)
         IF(N.LT.1.OR.N.GT.K) GOTO 80
         OPER=AVAIL(N,1)
         RECI=AVAIL(N,2)
         SERV=AVAIL(N,3)
         GOTO (100,200,300,400), OPER
  100    CONTINUE
            PTLN(RECI)=PTLN(RECI)+1
            TALON(PTLN(RECI),RECI)=CARD
            CARD=0
            GOTO 40
  200    CONTINUE
            PFND(RECI)=PFND(RECI)+1
            FOUND(PFND(RECI),RECI)=CARD
            CARD=0
            GOTO 40
  300    CONTINUE
            PFND(RECI)=PFND(RECI)+1
            FOUND(PFND(RECI),RECI)=TALON(PTLN(SERV),SERV)
            PTLN(SERV)=PTLN(SERV)-1
            GOTO 40
  400    CONTINUE
  500 CONTINUE
      WIN=0
      DO 510 I=1,4
         IF(PTLN(I).GT.0) RETURN
  510    CONTINUE
      WIN=1
      RETURN
      END
```

図6（a） FORTRAN 計算譜

(TURNUP)次の札（手）があれば PLACETHECARD を呼ぶ。なければ IFEND を呼んで，成功か失敗かを調べる。

(HAND) 山から次の札（手）をめくり，それを値とする。山が尽きたときには，NILを値とする。ここでは入力ファイルを用いている。

(IFEND) 成功ならば T，失敗ならば NIL。

(PLACETHECARD)いまめくった手を表示し，それをどの屑（台）におく（つける）かをたずねる（WHERETO）。それがおける（つけられる）場合にはおいて（LAY），MOVECARDS を呼ぶ。つけられない台を指定された場合には，PLACETHECARD を呼ぶ。

(WHERETO H S) Hは手，Sは8つの列のリスト。手をおく（つける）べき列の名前を値とす


```
      PROGRAM  CALCULATION
      INTEGER STOCK(52),WIN
      EXTERNAL TERMNL
      PRINT *, 'LUCKY NUMBER!
      READ *, INIT
      CALL RANINI(INIT)
      CALL SHUFFL(STOCK)
      CALL JUDGE(TERMNL,INIT,WIN)
      IF(WIN.EQ.1) PRINT *, 'SUCCESS!
      IF(WIN.EQ.0) PRINT *, 'FAILURE!
      STOP
      END
*
      SUBROUTINE CONS(AVAIL,K,OPER,RECI,SERV)
      INTEGER AVAIL(24,3),OPER,RECI,SERV
      K=K+1
      AVAIL(K,1)=OPER
      AVAIL(K,2)=RECI
      AVAIL(K,3)=SERV
      RETURN
      END
*
      SUBROUTINE COPY(K,A,B,N)
      INTEGER A(1),B(1)
      DO 1 I=1,N
         B(I)=K*A(I)
    1    CONTINUE
      RETURN
      END
*
      FUNCTION RANDOM()
      SAVE M
      M=M*3141592653
      RANDOM=ABS(M)*2.910383045676-11
      RETURN
      ENTRY RANINI(K)
      M=K
      RETURN
      END
*
      SUBROUTINE SHUFFL(STOCK)
      INTEGER STOCK(52)
      ABNORMAL RANDOM
      DO 10 I=1,52
         STOCK(I)=(I+3)/4
   10    CONTINUE
      DO 20 I=1,52
         N=RANDOM()*(52-I)+1+0.5
         K=STOCK(N)
         STOCK(N)=STOCK(I)
         STOCK(I)=K
   20    CONTINUE
      RETURN
      END
*
      SUBROUTINE TERMNL(COUNT,FOUND,PFND,TALON,PTLN,CARD,AVAIL,K,N)
      IMPLICIT INTEGER(A-Z)
      INTEGER FOUND(13,4),PFND(4)
      INTEGER TALON(52,4),PTLN(4)
      INTEGER AVAIL(24,3)
      CHARACTER DISP*1(13)/'A','2','3','4','5','6','7','8','9',
     &          'T','J','Q','K'/
    1 CONTINUE
      PRINT *, 'FOUNDATION!
      DO 10 I=1,4
         IF(PFND(I).LE.0) WRITE(6,61) I
         IF(PFND(I).GT.0) WRITE(6,61) I,(DISP(FOUND(J,I)),J=1,PFND(I))
   10    CONTINUE
      PRINT *, ' '
      PRINT *, 'TALON!
      DO 20 I=1,4
         IF(PTLN(I).LE.0) WRITE(6,61) I
         IF(PTLN(I).GT.0) WRITE(6,61) I,(DISP(TALON(J,I)),J=1,PTLN(I))
   20    CONTINUE
      IF(CARD.EQ.0) GOTO 22
         WRITE(6,62) COUNT
         PRINT *, '     ----- ',DISP(CARD)
   22    CONTINUE
      WRITE(6,63)
      DO 30 K1=1,K
         IF(AVAIL(K1,1).EQ.1) WRITE(6,64) K1,'TALON',AVAIL(K1,2)
         IF(AVAIL(K1,1).EQ.2) WRITE(6,64) K1,'FOUND',AVAIL(K1,2)
         IF(AVAIL(K1,1).EQ.3) WRITE(6,65) K1,AVAIL(K1,3),AVAIL(K1,2)
         IF(AVAIL(K1,1).EQ.4) WRITE(6,66) K1
   30    CONTINUE
      PRINT *, ' '
      PRINT *, 'INPUT N'
      READ *, N
      IF(N.GE.99) GOTO 1
      RETURN
   61 FORMAT(' ',I1,':',30A2)
   62 FORMAT('/' CARD ',I2,')')
   63 FORMAT('////' ( N)    --- OPERATION ---'/)
   64 FORMAT(' (',I2,')   CARD    => ',A5,'(',I1,')!)
   65 FORMAT(' (',I2,')   TALON(',I1,') -> FOUND(',I1,')')
   66 FORMAT(' (',I2,')   NEW CARD!)
      END
```

図6（b） FORTRAN 計算譜

```
120 (SETQ IBASE 10.)
130 (REWIND "NISIMURA/NOZAKI")
140 (LINENUMBER "NISIMURA/NOZAKI" NIL)
150 (SETQ *NOPOINT T)
160
170 (DE TURNUP ()
180    (COND ((SETQ H (HAND))
190                  (PLACETHECARD))
200          (T      (IFEND )) ))
210 (DE HAND ()
220   (READ "NISIMURA/NOZAKI"))
230 (DE IFEND ()
240    (AND (NULL T1)(NULL T2)(NULL T3)(NULL T4)
250         (EQUAL 13 (CAR F1))
260         (EQUAL 13 (CAR F2))
270         (EQUAL 13 (CAR F3))
280         (EQUAL 13 (CAR F4)) ))
290
300 (DE PLACETHECARD ()
310    (COND ((LAY (WHERETO H (LIST T1 T2 T3 T4 F1 F2 F3 F4)))
320                (MOVECARDS))
330          (T    (PLACETHECARD)) ))
340 (DE WHERETO (H S)
350    (PROG ()
360         (TERPRI)(TERPRI)
370         (PRINC "STACKS ARE")(TERPRI)
380         (TERPRI)
390         (PRINC "T1   ")(PRIN1 T1)(TERPRI)
400         (PRINC "T2   ")(PRIN1 T2)(TERPRI)
410         (PRINC "T3   ")(PRIN1 T3)(TERPRI)
420         (PRINC "T4   ")(PRIN1 T4)(TERPRI)
430         (TERPRI)
440         (PRINC "F1   ")(PRIN1 F1)(TERPRI)
450         (PRINC "F2   ")(PRIN1 F2)(TERPRI)
460         (PRINC "F3   ")(PRIN1 F3)(TERPRI)
470         (PRINC "F4   ")(PRIN1 F4)(TERPRI)
480         (TERPRI)
490         (PRINC "NEXT CARD IS  ")(PRIN1 H)(TERPRI)
500         (PRINC " WHERE  ")(TERPRI)
510         (RETURN (READ))) )
520 (DE LAY (N)
530    (COND ((ABLEP H N) (PUT H N))
540          (T           NIL) ))
550 (DE ABLEP (C N)
560    (COND ((OR (EQ N 'T1)(EQ N 'T2)(EQ N 'T3)(EQ N 'T4))
570               T)
580          (T      (ABLEM C N)) ))
590 (DE MAGIC (N)
600    (COND ((EQ N 'F1)  1)
610          ((EQ N 'F2)  7)
620          ((EQ N 'F3)  9)
630          ((EQ N 'F4) 10)
640          (T        NIL) ))
650 (DE PUT (C N)
660    (SET N (CONS C (EVAL N))) )
670
680 (DE MOVECARDS ()
690    (COND ((SHIFT (FROMTO (LIST T1 T2 T3 T4 F1 F2 F3 F4)))
700                  (TURNUP ))
710          (T      (MOVECARDS)) ))
720 (DE FROMTO (S)
730    (PROG ()
740         (PRINC " ((FROM TO) ... ) ")(TERPRI)
750         (RETURN (READ))) )
760 (DE SHIFT (M)
770    (COND ((NULL M)    T)
780          ((OR (EQ (CAAR M) 'T1)(EQ (CAAR M) 'T2)
790               (EQ (CAAR M) 'T3)(EQ (CAAR M) 'T4))
800                  (AND (EVAL (CAAR M))
810                       (ABLEM (CAR (EVAL (CAAR M))) (CADAR M))
820                       (MOVE (CAR M))
830                       (SHIFT (CDR M))))
840          (T      NIL) ))
850 (DE ABLEM (C N)
860    (COND ((OR (EQ N 'F1)(EQ N 'F2)(EQ N 'F3)(EQ N 'F4))
870                  (EQUAL (LENGTH (EVAL N))
880                         (REMAINDER (SUB1 (TIMES C (MAGIC N))) 13)))
890          (T      NIL) ))
900 (DE MOVE (M)
910    (AND (SET (CADR M) (CONS (CAR (EVAL (CAR M))) (EVAL (CADR M))))
920         (OR (SET (CAR M) (CDR (EVAL (CAR M)))) T) ))
930
940 (SETQ T1 (SETQ T2 (SETQ T3 (SETQ T4 NIL))))
950 (SETQ F1 (SETQ F2 (SETQ F3 (SETQ F4 NIL))))
960 (TURNUP)
```

図7 LISP 計算譜

る。ここでは値は，端末から人間が与えるようにしてある。

(LAY N)　札をおく。Nは札Hをおくべき列の名前。おいたらTを値とする。つけられない場合には，NILを値とする。

(ABLEP C N)　札Cを列Nにおけるかどうかを調べる。TまたはNILを値とする。

札 c が台 k につけられるかどうかは，c と k の値から，その札のつけられるべき位置を算出し，リストの現在の長さがちょうどその値の直前，すなわち

$$((c\cdot k^{\varphi(13)-1}-1) \bmod 13)$$

になっているかどうかで調べる。

ここで，$\varphi(13)=12$ はオイラーの関数であって，gcd$(k, m)=1$ ならば，

$$kx\equiv c \Longleftrightarrow x\equiv c\cdot k^{\varphi(m)-1} (\bmod m)$$

であることを用いる。

(MAGIC N)　Nは列の名前。Nが台 k の名前であるときに，$k^{11} \bmod 13$ を値とする。

(PUT C N)　札Cを列Nにおく。

(MOVECARDS)　屑から台への移替え操作の指示列を，リストとして受け取り (FROMTO)，これを順次に実行する (SHIFT)。すべて実行できれば，TURNUP を呼ぶ。途中で実行できなくなれば MOVECARDS を呼ぶ。

(FROMTO S)　Sは8つの列のリスト。移替え操作の指示列を値とする。ここでは値は，端末から人間が与えるようにしてある。

(SHIFT M)　札を移す。Mは移替え操作の指示列。操作が完了すれば，Tを値とする。完了しなければ，NILを値とする。

(ABLEM C N)　札Cを台Nにつけられるかどうか調べる。

(MAGIC N)——前出——

(MOVE M)　屑から台に札を1枚移す。Mは (T3 F2) というような形のリスト。

(以上：桜井記)


```
STACKS ARE

T1   (5 8 5 6)
T2   (3 5)
T3   (4 11 7 7 10 9 1 12 11 5 9 10)
T4   (13 13 13 13)

F1   (1)
F2   (8 6 4 2)
F3   (9 6 3)
F4   (12 8 4)

NEXT CARD IS  7
 WHERE ?
? T4
 ((FROM TO) ... ) ?
? ((T2 F4)(T4 F4))
```

図8　途中経過 (LISP)

計算譜競技会のお知らせ

あなたの知力を計算譜競技会で試してみませんか？　ただし参加資格のあるのは，あなたの知力を代行してくれる(はずの)プログラムに限ります。競技は，そのプログラムがあらかじめ与えられた数十個のシャッフル・データのうち何個について解けるかで優劣の判定をすることで行ないます（当然のことながら，山札の先読みをやってはいけません)。なお，厳正を期すため，このデータはプログラムが完成した時点で配布することにします。データの配布は今年九月ごろから始めたいと思っています。

この競技会は，毎年一月に箱根で行なわれているプログラミング・シンポジウムで，三年前に結成された GPCC (Game and Puzzle Competetion on Computer) の一環として行なわれているものです。参加される方は，下記のところにご連絡ください。

〒180　東京都武蔵野市緑町 3-9-11
電電公社武蔵野通研・池野特別研究室　竹内郁雄