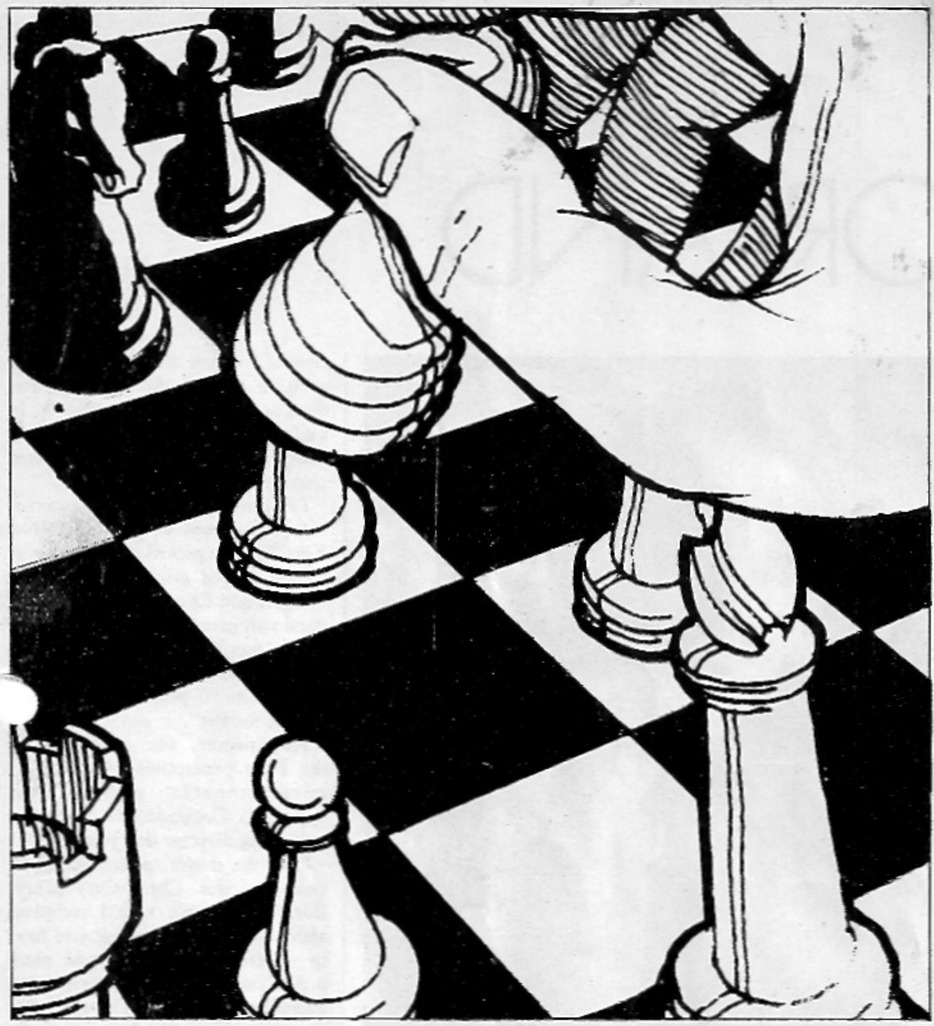


CHESS BOOK

The offered pawn in the Queen's Gambit may seem like easy pickings, but it will spell defeat time and time again for your micro if it is not primed to recognise this well-known snare. John White shows you how to program your machine [illegible] this and other book openings.

MANY COMMERCIAL games of strategy employ book-opening libraries in the early moves of the game. These are moves the manufacturer has pre-selected as particularly suitable for responding to moves by the opponent.

The advantage is that the book-opening move can be made almost at once, allowing more time for the program to consider its other moves. Secondly, it enables the program to make moves which long experience has taught are the best. Thirdly, it may enable the program to avoid opening traps.

Book openings are found in a variety of different games, but the most important example is that of chess, where literally hundreds of openings are known. A good example of an opening trap is the famous Queen's Gambit opening. After the moves d2-d4, d7-d5 and c2-c4, acceptance of the offered pawn, followed by a grim determination to cling on to it, nearly always spells disaster against accurate play by white. Yet it takes a deep search to see this, and all chess computers fall into the trap if they are not pre-programmed with a book opening which avoids it.

Have you ever wondered how book openings are added to a program of strategy? If you have, you probably assumed, as I did, that the book-opening library is constructed on the general principles of "If he does that, that I do this or this or this".

Let us consider what is required of a book-opening library. First, very few game positions are symmetrical in the sense that a book-opening library could be applied to play from both ends of the board. You need two libraries if you want to be able to play from both ends. For reasons of space, most commercial programs make do with a library which operates from one end of the board only.

Secondly, the program should be able to distinguish between several possible opponent responses which lead to different variations of the same opening, e.g., the Rauzer and Dragon variations of the Sicilian defence which diverge only at move 5 in the main lines.

Table 1.

Reference Number	Opening
0	Random selection by computer
1	Sicilian — Rauzer
2	Sicilian — Dragon
3	Ruy Lopez — Open
4	Giuoco Piano
5	King's Gambit
6	French — Winawer
7	Queen's Gambit Declined — Orthodox
8	Nimzo-Indian — Rubinstein
9	King's Indian — Classical
10	English — Symmetrical

Thirdly, it is desirable that the player should be able to select his own opening; failing this, an opening can be randomly selected by the program. Fourthly, the library should be able to prompt its opponent as to his continuation in the book.

The third and fourth features are unusual in games computers — again, because of the high cost of the extra memory needed.

Fifth, when a library is exhausted, or the opponent diverges from the library, the program must return to its own evaluation function. I soon found, by simple experiment, that construction of a simple tree of openings is tremendously wasteful of memory, and greatly retards the running of the program. The greater the tree, particularly for a deep opening of, say, eight moves, the slower the program runs. This becomes noticeable even at machine-code speeds. Yet commercial opening libraries give instant responses. Clearly, this was not the answer.

In the program presented here, all the machine moves are stored initially as four-figure strings in the two-dimensional array D$(MO,BO), where BO is the reference number of the opening and MO is the move number. The opponent responses are stored in the array C$(MO+1,BO) — MO+1 since one further move has been counted by the time the response is evaluated.

The principle is that the machine first chooses its opening number, BO, either randomly or by the opponent selecting it. It is then displayed. If no selection has been made, the response is matched against all similar responses and then the previous machine move, BQ, and the previous opponent move, BR$, are also checked to ensure that the matched moves have been derived from similar positions.

Duping the program

In theory, one could fool the program by arriving at the same consecutive three moves, on the same turns, from a different position. Yet this would be unlikely to arise by chance, and can be eliminated by careful construction of the library.

In this way, it is possible to switch from one opening to another, provided that both derive from the same original position. If the opponent has selected a book opening, then only the correct response for that opening is accepted. If no match is found at all, the counter BO is set to 0 and the program will henceforward ignore the library.

The advantages of this method is that, once

```
160  READ A(I,1)
170  A(I,2) = ~~80~~ 248 : A(I,7) = 158
180  NEXT          etc

211  FOR I = 1 TO 8
212  READ A(I,8)
213  NEXT
214  DATA 157,176,154,160,169,154,176,
                                     157
```

OPENINGS

the number of different openings has been fixed, the program always takes the same length of time to find each new move. The subroutine in lines 910-1170 contains the entire method.

The openings themselves are stored by reading a whole series of Data statements. The four-figure strings correspond to the array co-ordinates of the pieces on the board. For illustration, I have set up the data tables for 10 chess openings on a chessboard. The bottom-left square is labelled 1,1; the top-left is 1,8; the bottom-right is 8,1 and the top-right is 8,8. Thus a Data statement 7866 means: move from square 7,8 to square 6,6. This corresponds to a knight move from g8 to f6.

Data conversion

Castling is stored only as a king move. Extra routines recognise this and make the appropriate rook move. Various string-handling statements convert the raw numeric data into the usual algebraic notation.

Most of the rest of the program shows the results. The pieces of the board are represented most simply by ASCII codes corresponding to upper-case letters for black or lower-case for white. This is found in lines 150-210, and may need alteration according to your computer.

To maintain a static display — that is, one without scrolling — some cursor control is necessary, and I have used the standard codes:

[CLS] = CLEAR SCREEN
[HOME] = CURSOR HOME
[nCU] = CURSOR UP n times
[nCD] = CURSOR DOWN n times
[nSPC] = n SPACES

Equally, you could use screen Poking.

USR(62) sounds a beep on my Sharp MZ-80K and can be adapted or ignored. Other Sharp users will require an extended Basic for string inequalities and the logical operators And and Or. In addition, the ASCII codes given in the Data statements do not fit the Sharp, which has a non-standard set for lower-case letters.

When you Run the program, which takes about 5K, there will be a stage-wait while the moves are stored. You will then be asked if you wish to go first. Replying "Y" immediately makes the library inaccessible — this was put in to mimic the normal play from one end of the board only. After answering "N", you will be asked to select an opening — choose from the selection in table 1. For a random selection, type in 0.

The computer will now make its first move. You may then make your move by entering algebraic notation with commas, e.g., from

(continued on page 35)

```
100 REM** BOOK OPENING **
110 PRINT"[CLS]"
120 PRINT" BOOK OPENING by J.F.White."
130 PRINT" Storing positions."
140 DIMA (8,8),C$(10,10),D$(10,10),X(4),Y(4),X$(4),Y$(4)
150 FORI=1TO8
160 READA (I,8):A (I,1)= A (I,8)+ 32 } see above
170 A (I,7)=80:A (I,2)=112
180 NEXT
190 REM** SETTING UP BOARD **
200 REM** DATA IN 210 ARE ASCII VALUES FOR R,N,B,Q,K,B,N,R
210 DATA82,78,66,81,75,66,78,82 — See above
220 REM ** THE FOLLOWING DATA CAN BE INPUT FROM A DATA TAPE **
230 Z = 10: REM** Z = NO. OF OPENINGS STORED.
240 FORJ=1TOZ : FOR I= 1 TO 8
250 READ D$(I , J):READ C$(I +1,J )
260 NEXT:NEXT
1  270 DATA5254,3735,7163,4746,4244,3544,6344,7866,2133,2836,3175,3847,4142
   280 DATA1838,5131,2644
2  290 DATA5254,3735,7163,4746,4244,3544,6344,7866,2133,7776,3153,6877,6152
   300 DATA2836,5171,5878
3  310 DATA5254,5755,7163,2836,6125,1716,2514,7866,5171,6857,6151,2725,1423
   320 DATA4746,3233,5878
4  330 DATA5254,5755,7163,2836,6134,6835,3233,7866,4244,5544,3344,3524,3142
   340 DATA2442,2142,4745
5  350 DATA5254,5755,6264,5564,7163,7775,6134,4746,5171,8786,4244,6877,3233
   360 DATA2836,7273,7574
6  370 DATA5254,5756,4244,4745,2133,6824,5455,3735,1213,2433,2233,7857,4174
   380 DATA3544,7477,8878
7  390 DATA4244,4745,3234,5756,2133,7866,3175,6857,7163,5878,5253,2847,1131
   400 DATA3736,6143,4534
8  410 DATA4244,7866,3234,5756,2133,6824,5253,2726,6143,3827,7163,2433,2233
   420 DATA4745,3113,2847
9  430 DATA4244,7866,3234,7776,2133,6877,5254,4746,6152,5878,7163,5755,5171
   440 DATA2847,4445,4735
10 450 DATA3234,3735,2133,2836,7273,7776,6172,6877,7163,7866,5171,5878,4244
   460 DATA3544,6344,2644
470 REM ** END OF DATA **
480 INPUT"DO YOU WANT TO GO FIRST(Y\N)? ";A$
490 IFA$="Y"THENPRINT"[CLS]":GOTO600
500 INPUT"CHOOSE YOUR OPENING ";B$
510 BO=1:PRINT"[CLS]"
520 MO=MO+1
530 IFBO<>0THENGOSUB 910:IFBO<>0THEN600
540 REM ** MAIN PROGRAM HERE **
550 REM**
560 REM** NUMEROUS LINES **
570 PRINT"[CU]MAIN PROGRAM":USR(62):USR(62)
580 X$(1)="0":X$(2)="0":X$(3)="0":X$(4)="0"
590 REM**
600 PRINT"[HOME] FROM   ";X$(1) ;" ,";X$(2);"    TO    ";X$(3) ;" ,";X$(4)
610 USR(62)         15
620 IF A(X(3),X(4))= 107 THEN R=1:R1 = X(3): GOSUB 880
630 REM ** CASTLING
640 PRINT:PRINT
650 FORJ=8TO1STEP-1:FORI= 1 TO 8          : clearer
660 PRINTTAB(5*I-5);
670 IFA (I,J)=0 THENPRINT".";:GOTO690
680 PRINT CHR$(A (I,J));
690 NEXT:PRINT J:PRINT: NEXT
700 PRINT:PRINT"A    B    C    D    E    F    G    H"
710 PRINT"[HOME][22 CD]";
720 INPUT"YOUR MOVE.  FROM ?  " ;A1$,B1$  :USR(62)
730 IFA1$="Q"THEN PRINT"[HOME][24 CD]";:GOTO750
740 GOTO790
750 PRINT"BOOK OPENING REQUIRES FROM ";Y$(1);
760 PRINT" ,";Y$(2);" TO ";Y$(3);" ,";Y$(4);
770 FORI=1TO3000:NEXTI:PRINT"[HOME][24 CD]";
780 PRINT"[38 SPC]";:GOTO710
790 INPUT"                  TO  ?  ";A2$,B2$ :USR(62)
800 REM** ERROR CHECKING HERE **
810 A1=VAL(CHR$(ASC(A1$)-16))
820 B1= VAL(B1$): B2= VAL(B2$)
```

(listing continued on page 35)

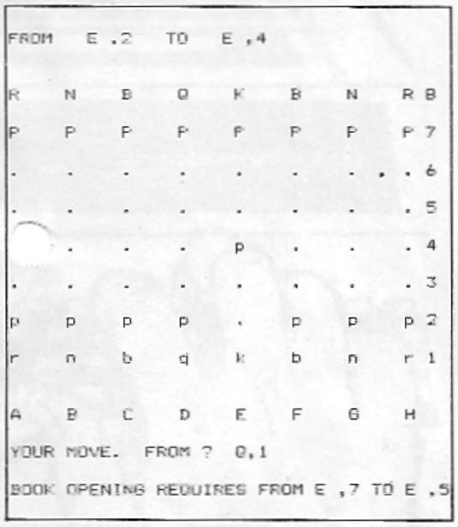
(continued from page 33)

d,7 to d,5, or enter q,1 when the computer will tell you what you ought to do.

When the library is exhausted, or if you enter a move not recognised by the program, the warning "Main Program" appears. The machine now enters a continuous loop, accepting only your responses without reply, and should be terminated with Break, CTRL/c etc., as appropriate.

The Data statements could easily be replaced by Data tapes or discs and it would be very easy to build up a library of tapes entitled Nimzowitsch, Ruy Lopez, and so on, as a graphic means of learning book openings.

The opening move.



```
FROM   E .2   TO   E ,4

R    N    B    Q    K    B    N    R B
P    P    P    P    P    P    P    P 7
.    .    .    .    .    .    .  . . 6
.    .    .    .    .    .    .    . 5
     .    .    .    P    .    .    . 4
.    .    .    .    .    .    .    . 3
p    p    p    p    .    p    p    p 2
r    n    b    q    k    b    n    r 1

A    B    C    D    E    F    G    H
YOUR MOVE.  FROM ?  Q,1
BOOK OPENING REQUIRES FROM E ,7 TO E .5
```

This could be far more fun than learning by rote from a book.

I wrote this program to find a way of storing book openings. I still do not know if this is the method used by commercial manufacturers — I suspect a yet simpler way may exist — but I hope that it will save some readers from reinventing the wheel. ■

(listing continued from page 33)

```
830 A2=VAL(CHR$(ASC(A2$)-16)): IF A(A1,B1)<90 OR A(A1,B1)>115THEN710
840 A1$=STR$(A1 ):A2$=STR$(A2 )
850 A(A2,B2)=A(A1,B1):A(A1,B1)=0
860 IF A(A2,B2) = 75 THEN R=8: R1= A2:GOSUB 880: REM ** CASTLING
870 GOTO520
880 IF R1 = 3 THEN A(4,R) = A(1,R): A(1,R) = 0
890 IF R1 = 7 THEN A(6,R) = A(8,R): A(8,R) = 0
900 RETURN
910 BO$="":BP$=A1$+B1$+A2$+B2$
920 IFVAL(B$) >0 THENBO=VAL(B$):IFBP$ = C$(MO,BO)THEN1010
930 FORI=1TO Z:REM ** Z = NO. OF BOOK OPENINGS
940 IF(BP$= C$(MO,I))AND(BQ$= D$(MO-1,I)) THEN 960
950 GOTO970
960 IF BR$ = C$(MO-1,I)THENBO$=BO$ + STR$(I)
970 NEXTI
980 IF BO$= "" THEN BO=0:RETURN
990 BS = INT(LEN(BO$)*RND(1) + 1)
1000 BO = VAL(MID$(BO$,BS,1))
1010 IF VAL(D$(MO,BO))=0 THEN BO=0 :RETURN
1020 X(1)  = VAL(LEFT$(D$(MO,BO),1))
1030 X(2)  = VAL(MID$(D$(MO,BO),2,1))
1040 X(3)  = VAL(MID$(D$(MO,BO),3,1))
1050 X(4)  = VAL(RIGHT$(D$(MO,BO),1))
1060 BQ$=STR$(X(1))+STR$(X(2))+STR$(X(3))+STR$(X(4)):BR$=BP$
1070 A(X(3),X(4))=A(X(1),X(2)):A(X(1),X(2))=0
1080 Y$(1) =  LEFT$(C$(MO+1,BO),1)
1090 Y$(2) =  MID$(C$(MO+1,BO),2,1)
1100 Y$(3) =  MID$(C$(MO+1,BO),3,1)
1110 Y$(4) =  RIGHT$(C$(MO+1,BO),1)
1120 FOR K=1TO3 STEP2
1130 X$(K) = CHR$(ASC(STR$(X(K)))+16)
1140 Y$(K) = CHR$(ASC(Y$(K))+16)
1150 X$(K+1) = STR$(X(K+1))
1160 NEXT
1170 RETURN
```