

Creating a Chess Player Part 3:

Larry R Atkin
Health Information Services
542 Michigan Av
Evanston IL 60202

Chess 0.5 (continued)

Peter W Frey
Dept of Psychology
Northwestern University
Evanston IL 60201

Listing 1: The second half of Chess 0.5, written in Pascal. This portion of the program covers evaluation of terminal nodes, the look-ahead procedure and user commands (listing 1 continued on page 146).

```

PROCEDURE EVALU8; (* EVALUATE CURRENT POSITION *)
VAR
  INTV : TV; (* SCORE *)

FUNCTION EVKING
  (A:RS;
   B:RS;TV:
  (* EVALUATE KING *)
  (* KING BIT BOARD *)
  (* FRIENDLY PAWN BIT BOARD *)

  VAR
    INTS : TS; (* SCRATCH *)
    INRS : RS; (* SCRATCH *)
    INTV : TV; (* SCRATCH *)

  BEGIN
    ANDRS(INRS,A,CORNR);
    IF MULRS(INRS) THEN (* KING NOT IN CORNER *)
      INTV := 0
    ELSE
      INTV := FKSAWD; (* KING SAFELY IN CORNER *)

    INRS := A;
    IF NEXTS(INRS,INTS) THEN
      BEGIN
        ANDRS(INRS,ATKFR(INTS),B); (* FIND PAWNS NEXT TO KING *)
        INTV := INTV + CNTRS(INRS)*FKPSMD; (* CREDIT EACH CLOSE PAWN *)
      END;

    EVKING := INTV; (* RETURN KING SCORE *)
  END; (* EVKING *)

FUNCTION EVMOBL
  (A,B:TP;TV:
  (* EVALUATE MOBILITY *)
  (* PIECE TYPES TO EVALUATE *)

  VAR
    INRS : RS; (* SCRATCH *)
    INTS : TS; (* SCRATCH *)
    INTV : TV; (* SCRATCH *)

  BEGIN
    IDRRS(INRS,TPLOC(A),TPLOC(B)); (* MERGE PIECE TYPES *)
    INTV := 0; (* INITIALIZE COUNT *)
    WHILE NEXTS(INRS,INTS) DO
      INTV := INTV + CNTRS(ATKFR(INTS)); (* COUNT ATTACKS *)
    EVMOBL := INTV; (* RETURN TOTAL ATTACKS *)
  END; (* EVMOBL *)

FUNCTION EYPAWN
  (A:RS;
   B:RS;
   C:TS;TV:
  (* EVALUATE PAWNS *)
  (* LOCATION OF PAWNS *)
  (* PAWN FORWARD DIRECTION *)
  (* PAWN HOME RANK *)

  VAR
    INRS : RS; (* SCRATCH *)
    INRS : RS; (* SCRATCH *)
    INTS : TS; (* SCRATCH *)
    INTV : TV; (* SCRATCH *)

  BEGIN
    SFTRS(INRS,A,S1);
    ANDRS(INRS,INRS,A1);
    INTV := CNTRS(INRS)*FPFLHX; (* BIT SET FOR SIDE BY SIDE *)
    (* SCORE PHALANX *)

    SFTRS(INRS,A,B1);
    ANDRS(INRS,INRS,A1);
    INTV := INTV + CNTRS(INRS)*FPCONN; (* BIT SET FOR PAWN DEFENSE *)
    (* CREDIT CONNECTED PAWNS *)

    SFTRS(INRS,A,B2);
    ANDRS(INRS,INRS,A1);
    INTV := INTV + CNTRS(INRS)*FPCONN; (* AND OTHER CONNECTED PAWNS *)

    SFTRS(INRS,A,B);
    NOTRS(INRS,TPLOC(INTS)); (* MOVE FORWARD *)
    ANDRS(INRS,INRS,INTS); (* OCCUPIED SQUARES *)
    INTV := INTV - CNTRS(INRS)*FPBLOK; (* BLOCKED PAWNS *)
    (* PENALIZE BLOCKED PAWNS *)

    CPYRS(INRS,A1);
    WHILE NEXTS(INRS,INTS) DO
      INTV := INTV + (ABS(ORD(C)-ORD(INTS))*FPADCR(INTS)); (* FOR EACH PAWN *)
      (* CREDIT PAWN ADVANCEMENT *)

    EYPAWN := INTV; (* RETURN PAWN SCORE *)
  END; (* EYPAWN *)

```

This month we conclude the listing and commentary of Chess 0.5 begun last issue. The program was written by Larry Atkin, who is coauthor with David State of the world championship chess program, Chess 4.6. The program is readily adaptable to personal computers having Pascal systems such as the UCSD Pascal project software. Part 4 concludes the series with a discussion of chess strategy and tactics.

Evaluating Terminal Positions

Another important aspect of any chess program is the function which provides a static evaluation of terminal positions in the look-ahead tree. In the present program, this routine also doubles as a preliminary scoring function for sorting moves at the first ply, at the beginning of the look-ahead search. Since the evaluation function is used repetitively in the search, efficiency demands that it be carefully engineered. We have left this task as an exercise for the reader. Our function presently includes only a few basic essentials.

The most important feature is material. We employ essentially the same function for this that is used by Chess 4.5. A trade-down bonus is also incorporated, ie: trade pieces but not pawns when ahead in material. A second feature which is considered is piece mobility. The mobility of Knights and Bishops is weighted more heavily than that for Rooks and Queens. Special credit is given to a King which is located in one of the four corner squares in each corner of the board, ie: 16 squares total. This encourages early castling. Pawn structure is considered by providing a bonus for advancing the pawns in the four center files, for having a pawn near the King, and for having a pawn adjacent to or defended by another pawn. This indirectly penalizes isolated or backward pawns. There is a direct penalty

if the square in front of a pawn is occupied. The position of the Rooks is considered by providing a bonus for placing a Rook on the seventh rank and for attacking another Rook of the same color (ie: doubled Rooks). The executive routine for these assessments is EVALU8.

The Look-Ahead Procedure

The look-ahead procedure is controlled by an executive routine called SEARCH. Several subprocedures are also defined which handle specific tasks. NEWBST keeps track of the move which is currently thought to be best, and dynamically re-orders the moves at the first ply level each time a new best-move is selected. M1NMAX determines whether the move under consideration will produce an a-b cutoff. SCOREM is called into action when the program can find no legal moves at a node. It determines whether the position should be scored as a checkmate or as a stalemate. SELECT is responsible for move ordering at each node. It determines whether there are any more moves to be searched and if so, makes sure that they are generated in the correct order (ie: captures, killers, castling moves, and then the remaining moves).

SEARCH incorporates a number of important features which make the look-ahead search more efficient. These include staged move generation, preliminary ordering scores, setting a narrow a-b window at the beginning of the search, conducting the search in an iterative fashion, and dynamically recording moves at the first ply as the search proceeds. Because of these features, the full-width search takes a long time instead of taking forever.

User Commands

For the user's convenience, the program should be able to respond to a few simple commands. Inputs to the program are processed by a lengthy routine, READER, which has many component subprocedures. The translation of the input string is handled by a group of routines: RDRERR, RDRGNT, RDRSFT, RDRCMP, RDLIN, RDRMOV and RDRNUM. Each of the commands is executed by a separate routine.

When the human player wishes to terminate the game before it has reached its conclusion (eg: when he is hopelessly lost and does not want to stay around to be crushed), he can simply type an END command and the ENDCMD routine will terminate the program. If the user simply wishes to start a new game, he can type INIT and the IN1CMD routine will set up for a new game.

TRS-80 USERS ; We've got the stuff that makes it tick !

SoftSide™

Your home computer is only as useful as what you put into it, and every month SoftSide™ publishes original debugged software that's bound to delight, entertain, educate and serve — free for the transcription in a form both you and your computer can understand.

Subscribe today, and ENTER a whole new world of software for personal use !

ham radio children's quizzes flying
science hobbies football religion
home accounting cribbage simulations music geography
machine boating word processing business education bio-feedback
language party games

Bored with
"zap the Klingons"
programming fare ?

The TRS-80 Software Exchange offers an alternative. At the back of each issue of SoftSide™, **The TRS-80 Software Exchange Market Basket** lists programs that serve a variety of interests. Are you a pilot ? How about a computerized flight plan ? If you're interested in using your TRS-80 for business, we've got programs to handle accounting, payroll, inventory management — even a cash register program ! Whether you're interested in statistical analysis, or Tarot cards, or even real estate, you're sure to find something in **The TRS-80 Software Exchange Market Basket !**

Where can you get it?

Only in SoftSide !

1 Year (12 issues) for \$15.00
Send check or money order to:



SoftSide Subscriptions
PO Box 68 Milford, NH 03055



VISA/Master Charge orders accepted.
Send signature, account #, exp. date of card and interbank # (M/C only) along with your order.

CBASIC Is Better

Take CBASIC—the best **Business BASIC**.
Make it faster, easier to use,
and more flexible, and you have

CBASIC-2

New features that enhance CBASIC's value as the best buy in Business BASIC:

CHAINING to pass control from program to program;
INTEGER VARIABLES allow fast computation in 16-bit binary arithmetic; **MULTIPLE LINE FUNCTIONS** allow easier, more structured coding; **XREF** produces a cross-reference dictionary of variables; **SAVEMEM** reserves memory space and loads subroutines; **SADDX** returns absolute address of strings; **CONSTATX** reports console status; **UCASEX** converts a string to upper case; **COMMANDX** returns contents of command line; plus **CONCHARX**, **XJECT**, **FILE BUFFERING**, **XPAGE**, and more.

Special Introductory Price: \$89.95

SSG CBASIC version one
owners can update for only: **\$49.95**
(send registration # when ordering)

Other vendors sell CBASIC.
But how many back up their customers when
software is updated? We do.

For information, or to order, contact:

Structured Systems Group

INCORPORATED

5208 CLAREMONT AVENUE
OAKLAND, CALIFORNIA 94618
(415) 547-1567

**GRAPHICS STOCK ANALYSIS
NAVIGATION ASTRONOMY
NUMBER THEORY GAMES**



Users of computers and programmable calculators enjoy this magazine! The articles are of interest to hobbyists and professionals. The programs are accompanied by explanations written in clear, logical language. Our authors are people who own and use machines like yours!

"The Recreational Programmer" is issued bi-monthly. It is available by subscription only for twelve dollars per year (domestic), or fifteen dollars per year (foreign).

Subscriptions and articles may be sent to:

THE RECREATIONAL PROGRAMMER
BOX 2571
KALAMAZOO MICHIGAN 49003

EXCELLENT CHOICE!

If the user would like to set up a specific position from the previous game or some other game, he can call the **BOACMD** routine, which will set up any position he desires. To use this instruction, the pieces are designated in the standard way (eg: K, Q, R, B, N and P) and the colors are designated by L for light and D for dark. The board is described by starting at the lower lefthand corner and listing, row by row, the 64 squares. Numbers are used to represent consecutive empty squares. The command to set up the position after 1. P-K4, P-K4, 2. N-KB3, N-QB3 is: **BOARD, LRNBQKB1 RPPPP1PPP5N24P34DP33N4PPPP1PPPR1B QKBNR**.

If the human player is lazy or simply wishes to test the program, he or she can type **GO** and the machine will select a move. By repeatedly typing **GO** the user can sit back and watch the machine play against itself. The routine that handles this is **GONCMD**. To specify a value for selected program parameter variables, the player can use **LETCMD**. For example, the amount of time the machine spends calculating a move can be controlled by specifying a limit for the number of nodes to be searched. The command **LET FNODEL = 1000** will cause the machine to set a target value of 1000 for the number of nodes to be searched. In this case it will not start another iteration if it has already searched 1000 nodes. If the user is confused about the current board configuration, the command **PRINT** will activate **PRICMD** which calls **PRINTB** for a representation (8 by 8 array) of the board. For diagnostic purposes the user can also ask for other information. The routine **PAMCMD** is activated by **PB** and provides an 8 by 8 attack map for each of the 64 squares. The routine **POPCMD** is activated by **PO** and gives information concerning the side to move (White or Black), the en passant status after the last move, the present castle status and the move number. If the user types **PM**, the routine **PMVCM** will provide a list of all moves which are legal for the side to move in the current position. The command **PL** activates **PLECMD** which prints the value of a designated variable; for example, the user can determine the present limit for the number of nodes to be searched by typing **PL FNODEL**.

The user also has control over several switches. He can ask the machine to repeat (echo) each entry, to pause after 20 lines of output, and to reply automatically each time the opponent enters a move. These switches are set by the switch commands (eg: **SW EC OFF**), and are processed by **SWICMD**. If the user wishes to manually alter one or more of the status conditions

(eg: side to move, move number, en passant, castling), this can be done by activating STACMD.

Notes on Notation

The program also processes standard chess notation. This is not strictly necessary. Many programs use their own convention for entering and reporting moves. A common procedure is to denote the squares using a number (1 through 8) for each row and a letter (A through H) for each column. A move is defined by listing the **present** square of the piece and then the destination square. For example, the common opening move, P-K4, would be E2E4. Moving the White Knight on the kingside from its original square to KB3 would be G1F3. This convention works nicely but it forces an experienced chess player to learn a new system. Most would prefer standard chess notation.

Because there are multiple ways to express the same move in standard notation, the translation routine needs to be fairly sophisticated. Consider a position in which the White Queen's Rook is on its original square and the neighboring Knight and Bishop have been moved. A move which

places the Rook on the Queen Bishop file can be designated as R-B1, R-QB1, R/1-B1, R/1-QB1, R/R1-B1, or R/R1-QB1. It is important that the program recognize that each of these character strings represents the same move. How is this done?

One way is to have the machine generate a list of all legal moves and then compare each of these with the move entered by the player. If his move matches one on the list, that move is noted. The rest of the list is then checked and if no more matches are found, the noted move is **assumed** to be the correct one. If no match is found, the machine prints "illegal move." If a second match is found (eg: P-B3 matches both P-KB3 and P-QB3), the machine prints "ambiguous move." The process of translating the opponent's move into machine compatible form **and** checking its legality or ambiguity is done by YRMOVE. The process of translating the machine's move into standard notation is handled by MYMOVE. Both of these procedures call MINENG, which is responsible for constructing the appropriating character strings.

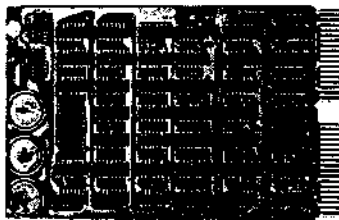
Final Thoughts

This completes our listing of our demonstration chess program. Despite the program's length, there are many desirable features which have been omitted. The reader with an interest in chess and programming should use this listing as a starting point for developing a program. The time required for move calculation can be reduced by writing machine dependent code for some of the frequently used routines. There are also features which can be added to improve the level of play.

One useful addition would be an opening library. An effective technique for this is described by Slate and Atkin in their chapter in *Chess Skill in Man and Machine* (P W Frey, editor, Springer-Verlag, New York, 1977). An opening library provides the user with a challenging set of opening moves and directs the game into situations which are familiar to the experienced chess player. By including various options at the early choice points and using a random selection procedure, the programmer can insure that the machine will not always select the same move sequence. The programmer can also give the user the option of specifying a particular opening against which he would like to practice. For important matches, the programmer can prepare surprise openings for the machine in order to gain a psychological edge on the opponent

Text continued on page 157

LSI-11 TIME



It's **TIME** you brought your LSI-11 up to **DATE**. **TIME** and **DATE**, two important parameters in the computer world, are available to your LSI-11 on one **DUAL SIZE BOARD**. When requested, the TCU-50D will present you with the date (month and day), time (hour and minutes), and seconds. Turn your computer off and forget about the time — your battery supported TCU-50D won't, not for 3 months anyway. The correct date and time will be there when you power up.

The TCU-50D is shipped preset to your local time, but can be set to any time you want by a simple software routine.

AT \$295

YOU CAN'T AFFORD TO IGNORE TIME

Time is only one way we can help you upgrade your LSI-11 or PDP-11 system. We'd also like to tell you about the others. So contact Digital Pathways if you're into -11's. **We are too.**



DIGITAL PATHWAYS INC.
4151 Middlefield Road • Palo Alto,
California 94306 • Telephone (415) 493-5544


Text continued from page 144

A second and somewhat more challenging project would be to develop a transposition table for the program. This requires the availability of unused memory (at least 8 K bytes and preferably 16 K or 32 K bytes), an efficient hashing scheme, and a set of decision rules to select among positions when a collision occurs (ie: two positions hash to the same address in the table). Another problem is that the use of a staged evaluation process and the a-b algorithm often provides an imprecise evaluation score (ie: the machine has determined that a position was not optimal but has not invested the time to find out exactly how bad it was). If the programmer succeeds with the transposition table, however, move calculation will take 30 to 50 per cent less time in most middle game positions and 60 to 90 per cent less time in many end game positions.

A third area for improvement is the evaluation function. Our program presently has only a rudimentary function. The reader should compare it with the one used by Chess 4.5 which is described in detail by Slate and Atkin. Their evaluation function provides an excellent starting point for revising our present function. In part 4 we will discuss the advantages of using a conditional evaluation function, ie: one that changes depending on the stage of the game and on the presence of special features. One implementation of this strategy is the special end game program described by Monroe Newborn in *Chess Skill In Man and Machine*.

It is appropriate for us to add two important disclaimers at this juncture. Although we have carefully tested each of the routines in the program and played several chess games, it is still possible that there are a few minor bugs in the program. If you find one, a letter to one of us or to BYTE would be appreciated. Secondly, our chess program was written primarily for pedagogical purposes. For this reason it is not a production program and does not run very efficiently. If you are the competitive type, our program should provide many useful ideas, but you should not expect it to compete successfully in tournament play unless you make extensive modifications and additions.


A chess program has a tendency to grow and change its personality as the programmer becomes more familiar with each of its many limitations. It provides a constant challenge for those of us who are too compulsive to tolerate obvious weaknesses. In fact one must be careful not to become totally obsessed with this project. We do not wish any of you to lose your job or your spouse because of a chess program.



Look to North Star And Computer Enterprises For Unbeatable Prices.

The North Star Horizon Computer System.
(With wooden cover and all standard items.)

Item Code	Description	List Price	Cash Price
NS-H-1	Horizon 1 kit	\$1599.	\$1299.
NS-H-1/A	Horizon 1 Assembled	\$1899.	\$1559.
NS-H-2	Horizon 2 kit	\$1999.	\$1639.
NS-H-2/A	Horizon 2 Assembled	\$2349.	\$1926.
North Star S-100 Boards			
NS-ZPB	Z-80A 4 mhz Processor kit	\$ 199.	\$ 163.
NS-ZPB/A	Z-80A 4 mhz Processor Assembled	\$ 259.	\$ 212.
NS-16K	16K 250ns RAM kit	\$ 399.	\$ 327.
NS-16K/A	16K 250ns RAM Assembled	\$ 459.	\$ 376.



P. O. Box 71 Fayetteville, N.Y. 13066

It's A Great Big Computer Worldbut you only need The Computer Corner



Don't know "bytes" from "bagels"? The courses given in our Seminar Room will get you "up and running". From home computerists to business people, our professionals have a course for you.

We offer classes with a hands-on approach using various micro-

computers. Courses range in scope from "Introduction to Home Computers" to "Professional Business Systems." Scheduled classes will include appropriate course materials and instruction by computer professionals.

- Processor Technology S0L
- Commodore PET & KIM
- North Star HORIZON • APPLE II

The computer corner

White Plains Mall, 200 Hamilton Avenue
White Plains, N.Y. 10601. (914) WHY-DATA
Hours: Daily & Sat. 10-6, Thurs. 10-9

Circle 65 on inquiry card.

December 1978 © BYTE Publications Inc. 157