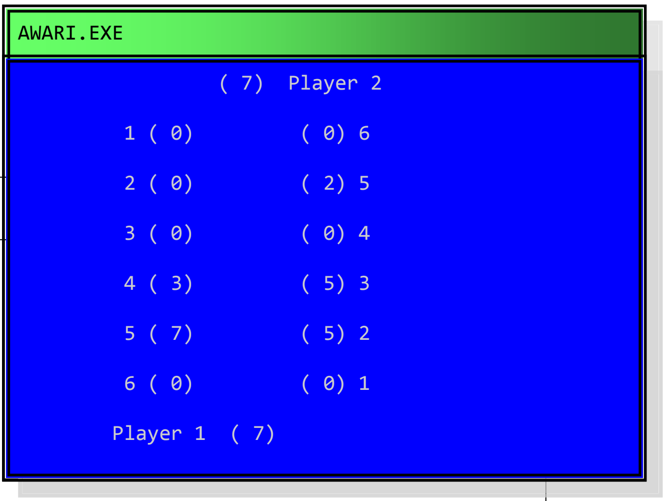
AWARI.C

Awari is a game that even a child could play. In the past it has been played with materials as simple as sticks and stones in the sand. The rules are simple. There are 6 places on each side of the board for stones and a place at either end for the players. You can play from your side and a move consists of taking all the stones in a place and dropping them, one-by-one into each place after the place you drew from going around in a circle, first towards your side, then drop a stone in your home, then back around the opponent's side, skipping the opponents home if you make it that far. If you end in your home you get another move. The twist comes in if the final stone to leave your hand lands in an empty spot. If the spot opposite it isn't empty you get to take that stone and all the stones in the spot opposite.

The computer's method for picking a move consists of looking ahead a little bit to rule out "bad" moves, but then following that up by comparing this game to previous games. If this game had been played before, and the computer lost, it would avoid making the same mistake again.

Awari was written by Joseph Larson based on a BASIC game by Geoff Wyvill as found in 'BASIC Computer Games' edited by David H. Ahl (c) 1978



```
                 ( 7)  Player 2

         1 ( 0)         ( 0) 6

         2 ( 0)         ( 2) 5

         3 ( 0)         ( 0) 4

         4 ( 3)         ( 5) 3

         5 ( 7)         ( 5) 2

         6 ( 0)         ( 0) 1

        Player 1  ( 7)
```

AWARI.C | You will need: a C/C++ complier .

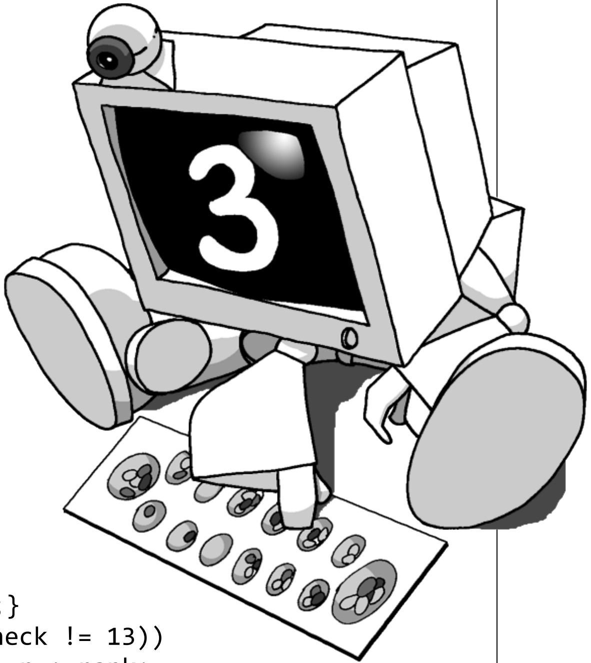
```
#include <stdio.h>
#include <time.h>
#include <math.h>
int memory[51], board[14];
int turn;
void intro (void) {
  char input;
  printf ("\nAwari\n");
  printf ("------\n");
  printf ("Do you need the rules? (y\\n) ");
  scanf ("%c", &input);
  if ((input == 'y') || (input == 'Y')) {
    printf ("\nAwari is an ancient African counting game. Each player has 6\n"
    "places with 3 markers or stones in each to start and an empty home. A\n"
    "move is made by taking all the markers from any non-empty place on your\n"
    "side, then \"sowing\" the markers one at a time in a counter-clockwise\n"
    "direction around the board. If the last marker sown lands in your home\n"
    "then you get a second move. (No more than two moves in one turn.)\n\n"
    "If the last marker is sown in an empty spot and the spot opposite is not\n"
    "empty then you capture the last marker and all the markers in the\n"
    "opposite spot. When either side is empty then the game is finished.\n\n"
    "Player one is at the left with home at the bottom, player two (or the\n"
    "computer in a one player game) is at the right with home at the top.\n"
    "By the way, the computer opponent is designed so that the more you play\n"
    "it, the better it gets.\n\n"
    "Good luck and have fun!\n");
  }
}
void drawboard (void) {
  int c;
  printf("\n\t\t (%2d)  Player 2", board[13]);
  for (c = 0; c < 6; c++)
    printf ("\n\n\t %d (%2d)\t\t(%2d) %d", c+1, board[c], board[12 - c], 6-c);
  printf("\n\n\tPlayer 1  (%2d)\n\n", board[6]);
}
```

Listing continued on page 2...

© 2008 Cymon's Games

Listing continued from page 1.

```
int domove (int move, int home) {
  int c;
  c = board[move]; board[move] = 0;
  do {
    if (++move > 13) move -= 14;
    board[move]++;
  } while (--c > 0);
  if ((board[move] == 1) && (move != 6) && (move !=13) && (board[12 - move])) {
    board[home] += board[12 - move] + 1;
    board[move] = board[12 - move] = 0;
  }
  if (move == home) return (1);
  else return (0);
}
void remember (int move) {
  turn++;
  if (move > 6) move -= 7;
  if (turn < 9) memory[memory[0]] = memory[memory[0]] * 6 + move;
}
int playermove (int pl) {
  int input;
  do {
    printf ("Player #%d, your move? (1 - 6) ", pl + 1);
    scanf ("%d", &input);
    if ((input < 1) || (input > 6) || !board[input + ((pl) ? 6 : - 1)]) {
      printf ("Illegal move.\n");
      input=0;
    }
  } while (!input);
  remember (--input);
  return (input + ((pl) ? 7 : 0));
}
int computermove (void) {
  int move, bestmove, rank, bestrank, c, r, check;
  int saveboard[14];
  for (c=0; c < 14; c++) saveboard[c] = board[c];
  bestrank = -99;
  for (move = 7; move <= 12; move++)
    if (board[move]) {
      rank = 0;
      domove(move, 13);
      for (c = 0; c < 5; c++)
        if (board[c]) {
          r = 0; check = board[c] + c;
          while (check > 14) { check -= 14; r = -1;}
          if ((!board[check]) && (check !=6) && (check != 13))
          r-=board[12 - check]; rank =(r < rank) ?  r : rank;
        }
      rank += board[13] - board[6];
      if (turn < 9) {
        check = memory[memory[0]] * 6 + move - 7;
        for (c = 1; c < memory[0]; c++)
          if (check == (int) ((float) memory[c] / pow (6, (7 - turn))))
            rank -= 2;
      }
      for (c=0; c < 14; c++) board[c] = saveboard[c];
      if (rank >= bestrank) {bestrank = rank; bestmove  = move;}
```



Listing continued on page 3...

© 2008 Cymon's Games

Listing continued from page 2...

```
    }
  remember (bestmove);
  printf("Computer moves %d", (bestmove - 6));
  return (bestmove);
}
int endgame (void) {
  int c, d, e;
  d = e = 0;
  for (c = 0; c < 6; c++) d += board[c];
  for (c = 7; c < 13; c++) e += board[c];
  if ((!d) || (!e)) return (1);
  else return (0);
}
void playgame (void) {
  int c, numpl;
  for (c = 0; c < 13; c++) board[c] = 3;
  board[6] = board [13] = memory[memory[0]] = numpl = 0;
  do {
    printf ("\nHow many players (1 - 2) ? ");
    scanf ("%d", &numpl);
    if (numpl < 1 || numpl > 2) {
      printf ("Please input either 1 or 2.\n");
      numpl = 0;
    }
  } while (!numpl);
  turn = 0;
  do {
    drawboard ();
    if (domove (playermove (0), 6))
      if (!endgame()) {
        drawboard ();
        printf ("Again.\n");
        domove (playermove (0), 6);
      }
    drawboard ();
    if  (!endgame ()) {
      if (domove ((numpl - 1) ? playermove (1) : computermove (), 13))
        if (!endgame()) {
          drawboard ();
          printf ("Again.\n");
          domove ((numpl - 1) ? playermove (1) : computermove (), 13);
        }
      printf("\n");
    }
  } while (!endgame ());
  drawboard ();
  printf ("\nGame Over.\n");
  c = board[6]-board[13];
  if (c < 0) printf ("%s by %d points.\n",
    (numpl - 1) ? "Player 2 wins" : "I win", (-c));
  else {
    if (!(numpl - 1) && memory[0] < 50) memory[0]++;
    if (c == 0) printf ("Drawn game.\n");
    else printf ("Player 1 wins by %d points.\n", c);
  }
}
```

Listing continued on page 4...

© 2008 Cymon's Games

AWARI.C	Listing continued from page 3...

```
int playagain (void) {
  char input[50];
  printf ("\nDo you want to play again? (y\\n) ");
  scanf ("%s", input);
  if ((input[0] == 'y') || (input[0] == 'Y')) return 1;
  else return 0;
}
void loadmemory (void) {
  FILE *fp;
  int c = 0;
  fp = fopen ("awari.dat", "r");
  if (fp == NULL) memory[0] = 1;
  else {
    while (!feof (fp)) fscanf (fp, "%d", &memory[++c]);
    fclose (fp);
    memory[0] = c;
  }
}
void savememory (void) {
  FILE *fp;
  int c;
  fp = fopen ("awari.dat", "w");
  if (fp != NULL) {
    for (c = 1; c < memory[0]; c++)
      fprintf(fp, "%d\n", memory[c]);
    fclose(fp);
  }
}
int main (void) {
  intro ();
  loadmemory ();
  do playgame (); while (playagain ());
  if (memory[0] > 1) savememory ();
  return 0;
}
```

Author's Notes:

This is a algorithmic conversion of a BASIC game, which is to say the BASIC game played exactly the same way as this one does. I think it is interesting to note how the computer remembers past moves. Since you can only make moves from 1 to 6 it stores all moves made as a base 6 number. In other words it takes the past game's moves, multiplies it by 6, and adds the current move to it. If it helps, imagine multiplying by 10 and adding the current move. Then to compare past moves all it needs to do is divide and compare the last digit.

This program improves on the basic game by remembering all past games in a file so it continues to learn game to game. My only problem with this is I hardly beat the game more than once. If you want to see it in action start with a win. Use these moves: 4, 5, 1, 3, 2, 4, 3, 6, 1, 5, 6. You should win by nine points. Then try to win the same way and watch the computer not let you get away with it twice.

© 2008 Cymon's Games