–

2022 年度 ソフトウェア技術

C 言語 [演習]

2023 年 10 月 7 日

S.Matoike

目次

第 1 章

三目並べ (Tic Tac Toe)

1.1　ゲームの概要

プログラムを実行すると、盤面が表示され、×の石を置く場所を指定するよう促されます。

画面上に示された番号を入力すると、その番号のスロットに×の石が置かれた盤面が表示され、次の手番の○に、石を置く場所を指定するように促されます。

手番を交互に変えながらゲームは進み、縦、横、斜めの何れかに、先に一列に自分の石を並べた方が勝ちとなります。

既に石の置かれているスロット番号を指定することはできません。また、スロット番号として 0 〜 8 以外の数値を指定することもできません。

```
スタート！ [Tic Tac Toe]
/---|---|---\
| 0 | 1 | 2 |
|---|---|---|
| 3 | 4 | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'X' さんの turn です
石を置く場所 0 〜 8 を指定して下さい : 4
/---|---|---\
| 0 | 1 | 2 |
|---|---|---|
| 3 | X | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'O' さんの turn です
石を置く場所 0 〜 8 を指定して下さい : 2
/---|---|---\
| 0 | 1 | O |
|---|---|---|
| 3 | X | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'X' さんの turn です
石を置く場所 0 〜 8 を指定して下さい :
```

1.2　主処理 (main)

```
int main(int argc, char *argv[]){
    /* 先手後手を決定 */
    int turn = BATSU,  winner, num;
    if (1 < argc){
        if (!strcmp(argv[1], "-r"))
            turn = MARU;
    }
    printf("スタート！␣[Tic␣Tac␣Toe]\n");
    do{
        printBoard();             /* ①盤面の表示 */
        num = slotNum(turn); /* ②手を入力 */
        board[num] = turn;        /* ③手を盤面に配置 */
        turn = switchTurn(turn); /* ④手番の交代 */
        winner = checkWinner();  /* ⑤勝敗の判定 */
    } while (winner == NEXT);
    /* 対戦結果の表示 */
    printBoard();
    result(winner);
    return 0;
}
```

1.3　盤面の表示 (printBoard)

```
void printBoard() {
    char bd[9];
    int i;
    for (i = 0; i < 9; i++) {
        if (board[i] == MARU) bd[i] = 'O';
        else if (board[i] == BATSU) bd[i] = 'X';
        else bd[i] = '0' + i;
    }
    printf("\n/---|---|---\\\n");
    printf("|␣%c␣|␣%c␣|␣%c␣|\n", bd[0], bd[1], bd[2]);
    printf("|---|---|---|\n");
    printf("|␣%c␣|␣%c␣|␣%c␣|\n", bd[3], bd[4], bd[5]);
    printf("|---|---|---|\n");
    printf("|␣%c␣|␣%c␣|␣%c␣|\n", bd[6], bd[7], bd[8]);
    printf("\\---|---|---/\n");
}
```

1.4　手番の交代 (switchTurn)

```
int switchTurn(int turn) {
    if (turn== BATSU)return MARU;
    return BATSU;
}
```

1.5　入力 (slotNum)

```
int slotNum(int turn) {
    int num;
    char *fig = "";
    if (turn==MARU) fig = "'O'";
    else if (turn==BATSU) fig = "'X'";
    do {
        printf("\n␣%s␣さんのturnです\n石を置く場所␣0␣〜␣8␣を指定して下さい：", fig);
        //while (getchar() != '\n');  /* 標準入力バッファのクリア */
        scanf("%d", &num);
        if (!(0 <= num && num < 9)) {
            printf("再指定：0␣〜␣8␣を指定して下さい");
            continue;
        }
        if (board[num] != num) {
            printf("再指定：そこには既に石が置かれています\n");
            continue;
        }
        break;
    } while (1);
    return num;
}
```

1.6　判定 (checkWinner)

```
int lineSum(int n1, int n2, int n3) {
    return board[n1] + board[n2] + board[n3];
}
int checkWinner() {
    int i, line = 0;
    for (i = 0; i < 8; i++) {
        switch (i) {
        case 0: line = lineSum(0, 1, 2); break;
        case 1: line = lineSum(3, 4, 5); break;
        case 2: line = lineSum(6, 7, 8); break;
        case 3: line = lineSum(0, 3, 6); break;
        case 4: line = lineSum(1, 4, 7); break;
        case 5: line = lineSum(2, 5, 8); break;
        case 6: line = lineSum(0, 4, 8); break;
        case 7: line = lineSum(2, 4, 6); break;
        }
        if (line == 3 * MARU) return MARU;
        else if (line == 3 * BATSU) return BATSU;
    }
    for (i = 0; i < 9; i++){
        if (0 <= board[i] && board[i] < 9) return NEXT;
    }
    return DRAW;
}
```

1.7　結果表示 (result)

```
void result(int winner) {
    printf("\n");
    switch (winner) {
    case DRAW: printf("引き分け\t"); break;
    case MARU: printf("'O'␣の 勝 ち\t"); break;
    case BATSU: printf("'X'␣の 勝 ち\t"); break;
    }
    printf("またね！\n");
}
```

1.8　各種宣言など

これは冒頭に記述する

```
#include <stdio.h>
#include <string.h>

// function prototypes
/*
int lineSum(int, int, int);
int switchTurn(int);
void printBoard();
int slotNum(int);
int checkWinner();
void result(int);
*/

static int board[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
#define MARU 10
#define BATSU -10
#define DRAW 100
#define NEXT 200
```

第 2 章

スライド・パズル (15Puzzle)

2.1　ゲームの概要

4 × 4 に区切った盤面上の各タイルに、0〜15 の番号が割り振られている。0 が割り振られたタイルは空欄になっていて、他のタイルはその空欄にスライドさせて移動することができる。最初は不規則に並べられている盤面ですが、空欄の上下、あるいは空欄の左右のタイルを選んでは、空欄の方向にスライドさせる事によって、最終的に 1 から 15 まで規則正しく並んだ盤面状態を目指すゲームである。

```
% ./slidetile

[11][ 3][ 6][10]
[ 9][ 1][ 8][14]
[ 2][ 4][ 7][12]
[13][15][  ][ 5]

Select number:6

[11][ 3][  ][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:11

[  ][11][ 3][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:11

[11][  ][ 3][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:
```

2.2　主処理 (main)

```
int main(void) {
    int sel;
    init();
    do {
        disp();
        do {
            printf("\nSelect number:");
            scanf("%d", &sel);
        } while( sel>ROW*COLUMN );
        moveTile(sel);
    } while( check() );

    return 0;
}
```

2.3　盤面の初期化 (init)

乱数を使ってタイルをシャフルするので、時刻を乱数の種に指定することによって、プログラムを起動する度に、異なるタイル配置になる様にしている

```
void init(){
    int i, j, n;
    srand((unsigned)time(NULL));
    for (i = 0; i < ROW; i++){
        for (j = 0; j < COLUMN; j++){
            n = i * COLUMN + j;
            tiles[n].num = n;
            tiles[n].row = i;
            tiles[n].clm = j;
        }
    }
    shuffle(1000);
}
```

2.4　シャッフル (shuffle)

乱数を使って、タイルを不規則に選択し移動させている

```
void shuffle(int n){
    int sel, min = 0, max = ROW*COLUMN;
    do{
        sel = (rand() % (max - min + 1)) + min;
        moveTile(sel);
        n--;
    } while (0 < n);
}
```

2.5　盤面の表示 (disp)

```
void disp(){
    int i, j, n;
    printf("\n");
    for (i = 0; i < ROW; i++){
        for (j = 0; j < COLUMN; j++){
            n = i * COLUMN + j;
            if (tiles[n].num == 0){
                printf("[␣␣]");
            } else {
                printf("[%2d]", tiles[i * COLUMN + j].num);
            }
        }
        printf("\n");
    }
}
```

2.6　タイルのスライド (moveTile)

```
int findTileNum(int num){
    int i;
    for (i = 0; i < ROW * COLUMN; i++){
        if (tiles[i].num == num){
            return i;
        }
    }
    return -1;
}
void swapTile(int n1, int n2){
    int tmp = tiles[n1].num;
    tiles[n1].num = tiles[n2].num;
    tiles[n2].num = tmp;
}
void swapTileR(int c, int r1, int r2){
    int n1 = r1 * COLUMN + c;
    int n2 = r2 * COLUMN + c;
    swapTile(n1, n2);
}
void swapTileC(int r, int c1, int c2){
    int n1 = r * COLUMN + c1;
    int n2 = r * COLUMN + c2;
    swapTile(n1, n2);
}
void moveTile(int sel){
    int j;
    int s = findTileNum(sel);
    int sr = tiles[s].row;
    int sc = tiles[s].clm;
    int z = findTileNum(0);
    int zr = tiles[z].row;
    int zc = tiles[z].clm;
    if (sr == zr){
        if (sc < zc){
            for (j = zc; j > sc; j--){
                if (0 <= j - 1){
                    swapTileC(sr, j, j - 1);
                }
```

```
            }
        } else if (zc < sc) {
            for (j = zc; j < sc; j++){
                if (j + 1 < COLUMN){
                    swapTileC(sr, j, j + 1);
                }
            }
        }
    }
    if (sc == zc){
        if (sr < zr){
            for (j = zr; j > sr; j--){
                if (0 <= j - 1){
                    swapTileR(sc, j, j - 1);
                }
            }
        } else if (zr < sr){
            for (j = zr; j < sr; j++){
                if (j + 1 < ROW){
                    swapTileR(sc, j, j + 1);
                }
            }
        }
    }
}
```

2.7　完成チェック (check)

```
enum BOOLEAN check(){
    int i, j, n;
    for (i = 0; i < ROW; i++){
        for (j = 0; j < COLUMN; j++){
            n = i * COLUMN + j;
            if (tiles[n].num != n){
                return true;
            }
        }
    }
    return false;
}
```

2.8　各種宣言など

これは冒頭に記述する

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// 定数
#define ROW 4
#define COLUMN 4
// 構造体
struct Tile{
      int num;
      int row;
      int clm;
};
```

```
14 enum BOOLEAN{
15     false, /* false = 0, true = 1 */
16     true
17 };
18 // function prototypes
19 /*
20 int findTileNum(int);
21 void swapTile(int,int);
22 void swapTileR(int,int,int);
23 void swapTileC(int,int,int);
24 void moveTile(int);
25 void shuffle(int);
26 enum BOOLEAN check();
27 void disp();
28 void init();
29 */
30
31 static struct Tile tiles[16];    /* 16 = ROW * COLUMN */
```

第 3 章

神経衰弱 (Flip Cards)

3.1　ゲームの概要

カードを 2 枚開いて、一致すれば得点となり、不一致なら再度伏せて相手の手番になる。

```
% ./flipcard
  [ a ][ b ][ c ][ d ][ e ]
  [ f ][ g ][ h ][ i ][ j ]
  [ k ][ l ][ m ][ n ][ o ]
  [ p ][ q ][ r ][ s ][ t ]

  A さん：0 点 B さん：0 点    = A さんの番です =
    Select 1st card : a

  [ 4 ][ b ][ c ][ d ][ e ]
  [ f ][ g ][ h ][ i ][ j ]
  [ k ][ l ][ m ][ n ][ o ]
  [ p ][ q ][ r ][ s ][ t ]

    Select 2nd card : p

  [ 4 ][ b ][ c ][ d ][ e ]
  [ f ][ g ][ h ][ i ][ j ]
  [ k ][ l ][ m ][ n ][ o ]
  [ 5 ][ q ][ r ][ s ][ t ]

  ハズレ

  [ a ][ b ][ c ][ d ][ e ]
  [ f ][ g ][ h ][ i ][ j ]
  [ k ][ l ][ m ][ n ][ o ]
  [ p ][ q ][ r ][ s ][ t ]

  A さん：0 点 B さん：0 点    = B さんの番です =
    Select 1st card : e

  [ a ][ b ][ c ][ d ][ 4 ]
  [ f ][ g ][ h ][ i ][ j ]
  [ k ][ l ][ m ][ n ][ o ]
  [ p ][ q ][ r ][ s ][ t ]
```

```
  Select 2nd card : a

[ 4 ][ b ][ c ][ d ][ 4 ]
[ f ][ g ][ h ][ i ][ j ]
[ k ][ l ][ m ][ n ][ o ]
[ p ][ q ][ r ][ s ][ t ]
```

当たり

```
[ 4 ][ b ][ c ][ d ][ 4 ]
[ f ][ g ][ h ][ i ][ j ]
[ k ][ l ][ m ][ n ][ o ]
[ p ][ q ][ r ][ s ][ t ]
```

A さん：0 点 B さん：1 点　= B さんの番です =

```
  Select 1st card : t

[ 4 ][ b ][ c ][ d ][ 4 ]
[ f ][ g ][ h ][ i ][ j ]
[ k ][ l ][ m ][ n ][ o ]
[ p ][ q ][ r ][ s ][ 0 ]

  Select 2nd card : q

[ 4 ][ b ][ c ][ d ][ 4 ]
[ f ][ g ][ h ][ i ][ j ]
[ k ][ l ][ m ][ n ][ o ]
[ p ][ 9 ][ r ][ s ][ 0 ]
```

ハズレ

```
[ 4 ][ b ][ c ][ d ][ 4 ]
[ f ][ g ][ h ][ i ][ j ]
[ k ][ l ][ m ][ n ][ o ]
[ p ][ q ][ r ][ s ][ t ]
```

A さん：0 点 B さん：1 点　= A さんの番です =

```
  Select 1st card :
```

3.2　主処理 (main)

```
int main( void ){
    int cnum1, cnum2, Closed=ROW*COLUMN;
    int pointA=0, pointB=0;
    init();
    disp();
    do{
        printf("Aさん：%d␣点\tBさん：%d␣点\n", pointA, pointB);
        if (Turn)
            printf("=␣Aさんの番です␣=\n");
        else
            printf("=␣Bさんの番です␣=\n");
        cnum1 = getNum("1st");
        cnum2 = getNum("2nd");
        if ( match(cnum1, cnum2) ){
            printf("当たり\n");
            Closed -= 2;
            if (Turn)
                pointA++;
            else
                pointB++;
        } else {
            printf("ハズレ\n");
            closeCard(cnum1);
            closeCard(cnum2);
            Turn = !Turn;   /* 手番の交代 */
        }
        sleep(4);   /* 開いたカードを見せておく時間 */
        disp();
    } while ( 0 < Closed );

    return 0;
}
```

3.3　盤面の初期化 (init)

プログラム中で乱数を使うので、時刻を乱数の種に指定して、プログラムを起動するたびに異なるカード配置になるようにしている

```
void init(){
    srand((unsigned)time(NULL));
    int i, j, num;
    for (i = 0; i < ROW/2; i++){
        for (j = 0; j < COLUMN; j++){
            num = i * COLUMN + j;
            cards[num].num = cards[ROW * COLUMN - num].num = num;
            cards[num].row = cards[ROW * COLUMN - num].row = i;
            cards[num].clm = cards[ROW * COLUMN - num].clm = j;
            cards[num].opn = cards[ROW * COLUMN - num].opn = false;
        }
    }
    shuffle();
}
```

3.4　カードのシャフル (shuffle, swapCards)

乱数を使ってカード配置を入れ替えている

```
1  void swapCards(int n1, int n2){
2      struct Card temp;
3      temp = cards[n2];
4      cards[n2] = cards[n1];
5      cards[n1] = temp;
6  }
7  void shuffle(){
8      int sel, min = 0, max = ROW*COLUMN;
9      while (0 < max){
10         sel = (rand() % (max - min + 1)) + min;
11         swapCards(sel, max--);
12     }
13 }
```

3.5　盤面の表示 (disp)

```
1  void disp(){
2      int i, j, n;
3      printf("\n");
4      for (i = 0; i < ROW; i++){
5          for (j = 0; j < COLUMN; j++){
6              n = i * COLUMN + j;
7              if ( !cards[n].opn )
8                  printf("[␣%c␣]", 'a' + i * COLUMN + j );
9              else
10                 printf("[␣%d␣]", cards[i * COLUMN + j].num );
11         }
12         printf("\n");
13     }
14     printf("\n");
15 }
```

3.6　入力 (getNum)

```
1  int getNum(char* s){
2      int cnum;
3      char str[24], work[24];
4      do {
5          sprintf(work, "%s%s%s", "\tSelect␣", s, "␣card␣:␣");
6          printf("%s", work);
7          scanf("%s", str);
8          cnum = str[0] - 'a';
9      } while (cnum > ROW * COLUMN);
10     openCard(cnum);
11     disp();
12     return cnum;
13 }
```

3.7　カードの開閉 (openCard, closeCard)

```
void openCard(int n){
    cards[n].opn = true;
}
void closeCard(int n){
    cards[n].opn = false;
}
```

3.8　一致不一致の判定 (match)

```
enum BOOLEAN match(int n1, int n2){
    if ( cards[n1].num == cards[n2].num )
        return true;
    return false;
}
```

3.9　各種宣言など

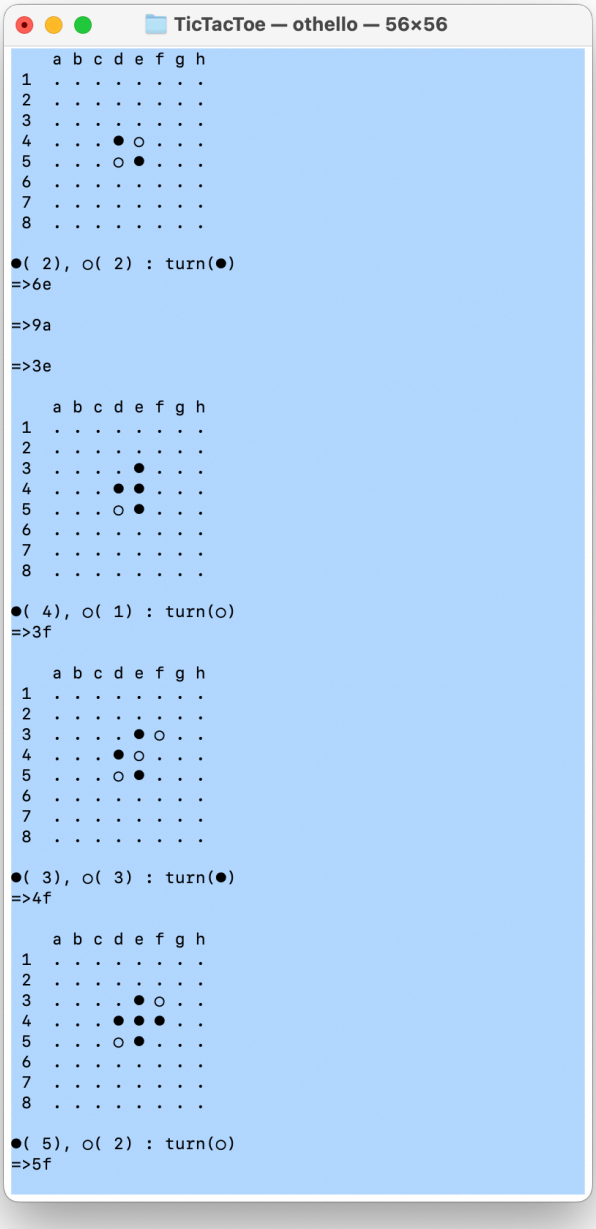
これは冒頭に記述する

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
// 定数
#define ROW 4
#define COLUMN 5
// 構造体
enum BOOLEAN {
    false, /* false = 0, true = 1 */
    true
};
struct Card {
    int num;
    int row;
    int clm;
    enum BOOLEAN opn;
};
// function prototypes
/*
void openCard(int);
void closeCard(int);
void switchTurn(enum BOOLEAN);
void disp();
int getNum(char*);
enum BOOLEAN match(int, int);
void wasteTime(int);
void swapCards(int,int);
void shuffle();
void init();
*/
static struct Card cards[20]; /* 20 = ROW * COLUMN */
enum BOOLEAN Turn = true;
```

第 4 章

オセロ（リバーシ）

4.1　ゲームの概要



4.2　主処理

ゲーム盤のサイズは、$4\times4, 6\times6, 8\times8$ など偶数の幅を $\#define$ 文で設定する。行番号（1,2,3,...）と列記号文字（a,b,c,...）の連続する半角 2 文字で石の位置を指定する。手番の石を置けないところを指定しても無視される。

```
1   POS decode(char* str){
2       POS pos;
3       pos.x = atoi(str)-1;
4       char* alpha="abcdefghijklmnopqrstuvwxyz";
5       int i;
6       for(i=0; i<BOARDW; i++){
7           if(*(str+1)==alpha[i])
8               break;
9       }
10      pos.y = i;
11      return pos;
12  }
13
14  void event(POS pos){
15      if(endflag){
16          initboard();
17          return;
18      }
19      if(0<passcount){
20          nextturn();
21          drawboard();
22          return;
23      }
24      if(!isinside(pos))
25          return;
26      if(0==flippable(pos, turn))
27          return;
28      for(int i=0; i<8; i++){
29          int loop = search(pos, i, turn);
30          POS temp = pos;
31          for(int j=0; j<loop; j++){
32              temp = movepos(temp, i);
33              setstone(temp, turn);
34          }
35      }
36      setstone(pos, turn);
37      nextturn();
38      drawboard();
39  }
40
41  int main(void){
42      if(!initboard())
43          return 8;
44      char inpt[]="␣␣␣";
45      while(!endflag){
46          printf("=>");
47          scanf("%s", inpt);
48          POS pos = decode(inpt);
49          printf("\n");
50          event(pos);
51      }
52      return 0;
53  }
```

4.3　盤面の表示

盤面を表示するたびに、石の数を数えて表示している。

```
    int count(int color){
        int num=0;
        for(int i=0; i<(BOARDW * BOARDW); i++)
            if(board[i]==color)
                num++;
        return num;
    }

    void drawboard(){
        for(int x=0; x<BOARDW; x++){
            if(x==0){
                printf("␣␣␣␣");
                char a='a';
                for(int i=0; i<BOARDW; i++)
                printf("%c␣", a+i);
                printf("\n");
            }
            printf("%2d␣␣", x+1);
            for(int y=0; y<BOARDW; y++){
                int index = y * BOARDW + x;
                switch(board[index]){
                    case BLACK:printf("%s␣", TILE[BLACK]);break;
                    case WHITE:printf("%s␣", TILE[WHITE]);break;
                    case NONE: printf("%s␣", TILE[NONE]); break;
                }
            }
            printf("\n");
        }
        printf("\n%s(%2d),␣%s(%2d)",TILE[BLACK],count(BLACK),TILE[WHITE],count(WHITE
          ));
        if(!endflag)
            printf("␣:␣turn(%s)\n", TILE[turn]);
        else
            printf("\n");
    }
```

4.4　初期化

ボードの幅が偶数でないとゲームを始められない。

```
    enum BOOLEAN initboard(){
        if(BOARDW%2)
            return false;
        int x, y;
        POS pos;
        for(y=0; y<BOARDW; y++)
            for(x=0; x<BOARDW; x++){
                pos.x = x; pos.y = y;
                setstone(pos, NONE);
            }
        pos.x = pos.y = BOARDW/2-1;
        setstone(pos, BLACK);
        pos.x = BOARDW/2; pos.y = pos.x-1;
        setstone(pos, WHITE);
        pos.y = BOARDW/2; pos.x = pos.y-1;
```

```
16          setstone(pos, WHITE);
17          pos.x = pos.y = BOARDW/2;
18          setstone(pos, BLACK);
19          turn = BLACK;
20          passcount = 0;
21          endflag = false;
22          drawboard();
23          return true;
24      }
```

4.5　手番の交代

指定された場所が、盤の内部の位置かどうか、相手の石を反転させられるかどうかをチェックしている。

```
1       POS movepos(POS pos, int v){
2           POS p;
3           p.x = pos.x + UNITV[v][0];
4           p.y = pos.y + UNITV[v][1];
5           return p;
6       }
7
8       enum BOOLEAN isinside(POS pos){
9           if( (pos.x<0) || (BOARDW<=pos.x) )
10              return false;
11          if( (pos.y<0) || (BOARDW<=pos.y) )
12              return false;
13          return true;
14      }
15
16      int search(POS pos, int v, int num){
17          int piece = 0;
18          while(true){
19              pos = movepos(pos, v);
20              if(!isinside(pos))
21                  return 0;
22              if(getstone(pos)==NONE)
23                  return 0;
24              if(getstone(pos)==num)
25                  break;
26              piece ++;
27          }
28          return piece;
29      }
30
31      int flippable(POS pos, int num){
32          if(getstone(pos)!=NONE)
33              return 0;
34          int total = 0;
35          int vec[]={0,0};
36          for(int i=0; i<8; i++)
37              total += search(pos, i, num);
38          return total;
39      }
40
41      void nextturn(){
42          turn ^= 1;
43          int empty = 0;
44          for(int y=0; y<BOARDW; y++)
45              for(int x=0; x<BOARDW; x++){
46                  POS pos; pos.x=x; pos.y=y;
47                  if(getstone(pos)==NONE)
48                      empty++;
```

```
49                  if(0<flippable(pos,turn)){
50                      passcount = 0;
51                      return;
52                  }
53              }
54          if(empty==0){
55              endflag = true;
56              return;
57          }
58          passcount++;
59          if(2<=passcount)
60              endflag = true;
61      }
```

4.6　各種宣言など

これは冒頭に記述する。

BOOLEAN 形を定義している。ボードの幅は *#define* 文で指定する。

```
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       enum BOOLEAN{
5           false,    /* false=0, true=1 */
6           true
7       };
8
9       #define BOARDW (8) // 4, 6, 8, ....
10      #define BLACK (0)
11      #define WHITE (1)
12      #define NONE (2)
13
14      int UNITV[][2] = {{0,-1},{1,-1},{1,0},{1,1},{0,1},{-1,1},{-1,0},{-1,-1}};
15      int turn = BLACK;
16      int passcount = 0;
17      int endflag = false;
18      int board[BOARDW * BOARDW];
19
20      const char* TILE[] = {
21          "●",  //TILE_BLACK
22          "○",  //TILE_WHITE
23          "."   //TILE_NONE
24      };
25
26      typedef struct {
27          int x, y;
28      }POS;
29
30      void setstone(POS pos, int num){
31          int index = (pos.y * BOARDW) + pos.x;
32          board[index] = num;
33      }
34
35      int getstone(POS pos){
36          int index = (pos.y * BOARDW) + pos.x;
37          return board[index];
38      }
```

付録 A

全体プログラム

A.1　三目並べ

ソースコード A.1　Tic Tac Toe(三目並べ)

```c
#include <stdio.h>
#include <string.h>

// function prototypes
/*
int lineSum(int, int, int);
int switchTurn(int);
void printBoard();
int slotNum(int);
int checkWinner();
void result(int);
*/

static int board[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
#define MARU 10
#define BATSU -10
#define DRAW 100
#define NEXT 200
/* ============================================================ */
int switchTurn(int turn) {
    if ( turn == BATSU) return MARU;
    return BATSU;
}
/* ============================================================ */
void printBoard() {
    char bd[9];
    int i;
    for (i = 0; i < 9; i++) {
        if ( board[i] == MARU ) bd[i] = 'O';
        else if ( board[i] == BATSU ) bd[i] = 'X';
        else bd[i] = '0' + i;
    }
    printf("\n/---|---|---\\\n");
    printf("| %c | %c | %c |\n", bd[0], bd[1], bd[2]);
    printf("|---|---|---|\n");
    printf("| %c | %c | %c |\n", bd[3], bd[4], bd[5]);
    printf("|---|---|---|\n");
    printf("| %c | %c | %c |\n", bd[6], bd[7], bd[8]);
    printf("\\---|---|---/\n");
}
/* ============================================================ */
void result(int winner) {
```

```
43      printf("\n");
44      switch (winner) {
45      case DRAW: printf("引き分け\t"); break;
46      case MARU: printf("'O'␣の勝ち\t"); break;
47      case BATSU: printf("'X'␣の勝ち\t"); break;
48      }
49      printf("またね！\n");
50  }
51  /* ============================================================ */
52  int slotNum(int turn) {
53      int num;
54      char *fig = "";
55      if ( turn == MARU ) fig = "'O'";
56      else if ( turn == BATSU ) fig = "'X'";
57      do {
58          printf("\n␣%s␣さんのturnです\n石を置く場所␣0␣〜␣8␣を指定して下さい：", fig);
59          //while (getchar() != '\n');  /* 標準入力バッファのクリア */
60          scanf("%d", &num);
61          if (!(0 <= num && num < 9)) {
62              printf("再指定：0␣〜␣8␣を指定して下さい");
63              continue;
64          }
65          if (board[num] != num) {
66              printf("再指定：そこには既に石が置かれています\n");
67              continue;
68          }
69          break;
70      } while (1);
71      return num;
72  }
73  /* ============================================================ */
74  int lineSum(int n1, int n2, int n3) {
75      return board[n1] + board[n2] + board[n3];
76  }
77  /* ============================================================ */
78  int checkWinner() {
79      int i, line = 0;
80      for (i = 0; i < 8; i++) {
81          switch (i) {
82          case 0: line = lineSum(0, 1, 2); break;
83          case 1: line = lineSum(3, 4, 5); break;
84          case 2: line = lineSum(6, 7, 8); break;
85          case 3: line = lineSum(0, 3, 6); break;
86          case 4: line = lineSum(1, 4, 7); break;
87          case 5: line = lineSum(2, 5, 8); break;
88          case 6: line = lineSum(0, 4, 8); break;
89          case 7: line = lineSum(2, 4, 6); break;
90          }
91          if (line == 3 * MARU) return MARU;
92          else if (line == 3 * BATSU) return BATSU;
93      }
94      for (i = 0; i < 9; i++){
95          if (0 <= board[i] && board[i] < 9) return NEXT;
96      }
97      return DRAW;
98  }
99  /* ============================================================ */
100 int main(int argc, char *argv[]){
101     /* 先手後手を決定 */
102   int turn = BATSU, winner, num;
103     if (1 < argc){
104         if (!strcmp(argv[1], "-r"))
105             turn = MARU;
106     }
107     printf("スタート！␣[Tic␣Tac␣Toe]\n");
```

```
108     do{
109         printBoard();              /* ①盤面の表示 */
110         num = slotNum(turn);       /* ②手を入力 */
111         board[num] = turn;         /* ③手を盤面に配置 */
112         turn = switchTurn(turn); /* ④手番の交代 */
113         winner = checkWinner();  /* ⑤勝敗の判定 */
114     } while (winner == NEXT);
115     /* 対戦結果の表示 */
116     printBoard();
117     result(winner);
118     return 0;
119 }
```

A.2　スライド・パズル

ソースコード A.2　スライド・パズル

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// 定数
#define ROW 4
#define COLUMN 4
// 構造体
struct Tile{
    int num;
    int row;
    int clm;
};
enum BOOLEAN{
  false, /* false = 0, true = 1 */
  true
};
// function prototypes
/*
int findTileNum(int);
void swapTile(int,int);
void swapTileR(int,int,int);
void swapTileC(int,int,int);
void moveTile(int);
void shuffle(int);
enum BOOLEAN check();
void disp();
void init();
*/

static struct Tile tiles[16];   /* 16 = ROW * COLUMN */

// サブプログラム
int findTileNum(int num){
    int i;
    for (i = 0; i < ROW * COLUMN; i++){
        if (tiles[i].num == num){
            return i;
        }
    }
    return -1;
}
/* ============================================================ */
void swapTile(int n1, int n2){
    int tmp = tiles[n1].num;
    tiles[n1].num = tiles[n2].num;
    tiles[n2].num = tmp;
}
/* ============================================================ */
void swapTileR(int c, int r1, int r2){
    int n1 = r1 * COLUMN + c;
    int n2 = r2 * COLUMN + c;
    swapTile(n1, n2);
}
/* ============================================================ */
void swapTileC(int r, int c1, int c2){
    int n1 = r * COLUMN + c1;
    int n2 = r * COLUMN + c2;
```

```
59      swapTile(n1, n2);
60  }
61  /* ============================================================ */
62  void moveTile(int sel){
63      int j;
64      int s = findTileNum(sel);
65      int sr = tiles[s].row;
66      int sc = tiles[s].clm;
67      int z = findTileNum(0);
68      int zr = tiles[z].row;
69      int zc = tiles[z].clm;
70      if (sr == zr){
71          if (sc < zc){
72              for (j = zc; j > sc; j--){
73                  if (0 <= j - 1){
74                      swapTileC(sr, j, j - 1);
75                  }
76              }
77          } else if (zc < sc) {
78              for (j = zc; j < sc; j++){
79                  if (j + 1 < COLUMN){
80                      swapTileC(sr, j, j + 1);
81                  }
82              }
83          }
84      }
85      if (sc == zc){
86          if (sr < zr){
87              for (j = zr; j > sr; j--){
88                  if (0 <= j - 1){
89                      swapTileR(sc, j, j - 1);
90                  }
91              }
92          } else if (zr < sr){
93              for (j = zr; j < sr; j++){
94                  if (j + 1 < ROW){
95                      swapTileR(sc, j, j + 1);
96                  }
97              }
98          }
99      }
100 }
101 /* ============================================================ */
102 void shuffle(int n){
103     int sel, min = 0, max = ROW*COLUMN;
104     do{
105         sel = (rand() % (max - min + 1)) + min;
106         moveTile(sel);
107         n--;
108     } while (0 < n);
109 }
110 /* ============================================================ */
111 enum BOOLEAN check(){
112   int i, j, n;
113     for (i = 0; i < ROW; i++){
114         for (j = 0; j < COLUMN; j++){
115             n = i * COLUMN + j;
116             if (tiles[n].num != n){
117                 return true;
118             }
119         }
120     }
121     return false;
122 }
123 /* ============================================================ */
```

```
124 void disp(){
125   int i, j, n;
126     printf("\n");
127     for (i = 0; i < ROW; i++){
128         for (j = 0; j < COLUMN; j++){
129             n = i * COLUMN + j;
130             if (tiles[n].num == 0){
131                 printf("[␣␣]");
132             } else {
133                 printf("[%2d]", tiles[i * COLUMN + j].num);
134             }
135         }
136         printf("\n");
137     }
138 }
139 /* ============================================================ */
140 void init(){
141     srand((unsigned)time(NULL));
142     int i, j, n;
143     for (i = 0; i < ROW; i++){
144         for (j = 0; j < COLUMN; j++){
145             n = i * COLUMN + j;
146             tiles[n].num = n;
147             tiles[n].row = i;
148             tiles[n].clm = j;
149         }
150     }
151     shuffle(1000);
152 }
153 // メイン
154 int main(void) {
155     int sel;
156     init();
157     do {
158         disp();
159         do {
160             printf("\nSelect␣number:");
161             scanf("%d", &sel);
162         } while( sel>ROW*COLUMN );
163         moveTile(sel);
164     } while( check() );
165 
166     return 0;
167 }
```

A.3　神経衰弱

ソースコード A.3　神経衰弱

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

// 定数
#define ROW 4
#define COLUMN 5
// 構造体
enum BOOLEAN {
  false, /* false = 0, true = 1 */
  true
};
struct Card {
    int num;
    int row;
    int clm;
    enum BOOLEAN opn;
};
// function prototypes
/*
void openCard(int);
void closeCard(int);
void switchTurn(enum BOOLEAN);
void disp();
int getNum(char*);
enum BOOLEAN match(int, int);
void wasteTime(int);
void swapCards(int,int);
void shuffle();
void init();
*/

static struct Card cards[20]; /* 20 = ROW * COLUMN */

enum BOOLEAN Turn = true;

// サブプログラム
void openCard(int n){
    cards[n].opn = true;
}
/* ============================================================ */
void closeCard(int n){
    cards[n].opn = false;
}
/* ============================================================ */
void disp(){
    int i, j;
    printf("\n");
    for (i = 0; i < ROW; i++){
        for (j = 0; j < COLUMN; j++){
            int n = i * COLUMN + j;
            if ( !cards[n].opn )
                printf("[␣%c␣]", 'a' + i * COLUMN + j );
            else
                printf("[␣%d␣]", cards[i * COLUMN + j].num );
        }
        printf("\n");
```

```
59      }
60      printf("\n");
61  }
62  /* ============================================================ */
63  int getNum(char* s){
64      int cnum;
65      char str[24], work[24];
66      do {
67          sprintf(work, "%s%s%s", "\tSelect␣", s, "␣card␣:␣");
68          printf("%s", work);
69          scanf("%s", str);
70          cnum = str[0] - 'a';
71      } while (cnum > ROW * COLUMN);
72      openCard(cnum);
73      disp();
74      return cnum;
75  }
76  /* ============================================================ */
77  enum BOOLEAN match(int n1, int n2){
78      if ( cards[n1].num == cards[n2].num )
79          return true;
80      return false;
81  }
82  /* ============================================================ */
83  void swapCards(int n1, int n2){
84      struct Card temp;
85      temp = cards[n2];
86      cards[n2] = cards[n1];
87      cards[n1] = temp;
88  }
89  /* ============================================================ */
90  void shuffle(){
91      int min = 0;
92      int max = ROW*COLUMN;
93      while (0 < max){
94          int sel = (rand() % (max - min + 1)) + min;
95          swapCards(sel, max--);
96      }
97  }
98  /* ============================================================ */
99  void init(){
100     srand((unsigned)time(NULL));
101     int i, j;
102     for (i = 0; i < ROW/2; i++){
103         for (j = 0; j < COLUMN; j++){
104             int num = i * COLUMN + j;
105             cards[num].num = cards[ROW * COLUMN - num].num = num;
106             cards[num].row = cards[ROW * COLUMN - num].row = i;
107             cards[num].clm = cards[ROW * COLUMN - num].clm = j;
108             cards[num].opn = cards[ROW * COLUMN - num].opn = false;
109         }
110     }
111     shuffle();
112 }
113 // メイン
114 int main( void ){
115     int cnum1, cnum2, Closed=ROW*COLUMN;
116     int pointA=0, pointB=0;
117     init();
118     disp();
119     do{
120         printf("Aさん：%d␣点\tBさん：%d␣点\n", pointA, pointB);
121         if (Turn)
122             printf("=␣Aさんの番です␣=\n");
123         else
```

```
124             printf("= B さんの番です =\n");
125         cnum1 = getNum("1st");
126         cnum2 = getNum("2nd");
127         if ( match(cnum1, cnum2) ){
128             printf("当たり\n");
129             Closed -= 2;
130             if (Turn)
131                 pointA++;
132             else
133                 pointB++;
134         } else {
135             printf("ハズレ\n");
136             closeCard(cnum1);
137             closeCard(cnum2);
138             Turn = !Turn;
139         }
140         sleep(4);
141         disp();
142     } while ( 0 < Closed );
143     return 0;
144 }
```

A.4　オセロ（リバーシ）

ソースコード A.4　オセロ

```
#include <stdio.h>
#include <stdlib.h>

enum BOOLEAN{
  false,    /* false=0, true=1 */
  true
};

#define BOARDW (8) // 4, 6, 8, ....
#define BLACK (0)
#define WHITE (1)
#define NONE (2)

int UNITV[][2] = {{0,-1},{1,-1},{1,0},{1,1},{0,1},{-1,1},{-1,0},{-1,-1}};
int turn = BLACK;
int passcount = 0;
int endflag = false;
int board[BOARDW * BOARDW];

const char* TILE[] = {
  "●",  //TILE_BLACK
  "○",  //TILE_WHITE
  "."   //TILE_NONE
};

typedef struct {
  int x, y;
}POS;

void setstone(POS pos, int num){
  int index = (pos.y * BOARDW) + pos.x;
  board[index] = num;
}

int getstone(POS pos){
  int index = (pos.y * BOARDW) + pos.x;
  return board[index];
}

int count(int color){
  int num=0;
  for(int i=0; i<(BOARDW * BOARDW); i++)
    if(board[i]==color)
      num++;
  return num;
}

void drawboard(){
  for(int x=0; x<BOARDW; x++){
    if(x==0){
      printf("␣␣␣␣");
      char a='a';
      for(int i=0; i<BOARDW; i++)
        printf("%c␣", a+i);
      printf("\n");
    }
    printf("%2d␣␣", x+1);
    for(int y=0; y<BOARDW; y++){
```

```
59        int index = y * BOARDW + x;
60        switch(board[index]){
61        case BLACK:printf("%s␣", TILE[BLACK]);break;
62        case WHITE:printf("%s␣", TILE[WHITE]);break;
63        case NONE: printf("%s␣", TILE[NONE]); break;
64        }
65      }
66      printf("\n");
67    }
68    printf("\n%s(%2d),␣%s(%2d)", TILE[BLACK],count(BLACK),TILE[WHITE],count(WHITE));
69    if(!endflag)
70      printf("␣:␣turn(%s)\n", TILE[turn]);
71    else
72      printf("\n");
73  }
74
75  enum BOOLEAN initboard(){
76    if(BOARDW%2)
77      return false;
78    int x, y;
79    POS pos;
80    for(y=0; y<BOARDW; y++)
81      for(x=0; x<BOARDW; x++){
82        pos.x = x; pos.y = y;
83        setstone(pos, NONE);
84      }
85    pos.x = pos.y = BOARDW/2-1;
86    setstone(pos, BLACK);
87    pos.x = BOARDW/2; pos.y = pos.x-1;
88    setstone(pos, WHITE);
89    pos.y = BOARDW/2; pos.x = pos.y-1;
90    setstone(pos, WHITE);
91    pos.x = pos.y = BOARDW/2;
92    setstone(pos, BLACK);
93    turn = BLACK;
94    passcount = 0;
95    endflag = false;
96    drawboard();
97    return true;
98  }
99
100 POS movepos(POS pos, int v){
101   POS p;
102   p.x = pos.x + UNITV[v][0];
103   p.y = pos.y + UNITV[v][1];
104   return p;
105 }
106
107 enum BOOLEAN isinside(POS pos){
108   if( (pos.x<0) || (BOARDW<=pos.x) )
109     return false;
110   if( (pos.y<0) || (BOARDW<=pos.y) )
111     return false;
112   return true;
113 }
114
115 int search(POS pos, int v, int num){
116   int piece = 0;
117   while(true){
118     pos = movepos(pos, v);
119     if(!isinside(pos))
120       return 0;
121     if(getstone(pos)==NONE)
122       return 0;
123     if(getstone(pos)==num)
```

```
124       break;
125     piece ++;
126   }
127   return piece;
128 }
129
130 int flippable(POS pos, int num){
131   if(getstone(pos)!=NONE)
132     return 0;
133   int total = 0;
134   int vec[]={0,0};
135   for(int i=0; i<8; i++)
136     total += search(pos, i, num);
137   return total;
138 }
139
140 void nextturn(){
141   turn ^= 1;
142   int empty = 0;
143   for(int y=0; y<BOARDW; y++)
144     for(int x=0; x<BOARDW; x++){
145       POS pos; pos.x=x; pos.y=y;
146       if(getstone(pos)==NONE)
147         empty++;
148       if(0<flippable(pos,turn)){
149         passcount = 0;
150         return;
151       }
152     }
153   if(empty==0){
154     endflag = true;
155     return;
156   }
157   passcount++;
158   if(2<=passcount)
159     endflag = true;
160 }
161
162 POS decode(char* str){
163   POS pos;
164   pos.x = atoi(str)-1;
165   char* alpha="abcdefghijklmnopqrstuvwxyz";
166   int i;
167   for(i=0; i<BOARDW; i++){
168     if(*(str+1)==alpha[i])
169       break;
170   }
171   pos.y = i;
172   return pos;
173 }
174
175 void event(POS pos){
176   if(endflag){
177     initboard();
178     return;
179   }
180   if(0<passcount){
181     nextturn();
182     drawboard();
183     return;
184   }
185   if(!isinside(pos))
186     return;
187   if(0==flippable(pos, turn))
188     return;
```

```
189   for(int i=0; i<8; i++){
190     int loop = search(pos, i, turn);
191     POS temp = pos;
192     for(int j=0; j<loop; j++){
193       temp = movepos(temp, i);
194       setstone(temp, turn);
195     }
196   }
197   setstone(pos, turn);
198   nextturn();
199   drawboard();
200 }
201
202 int main(void){
203   if(!initboard())
204     return 8;
205   char inpt[]="␣␣␣";
206   while(!endflag){
207     printf("=>");
208     scanf("%s", inpt);
209     POS pos = decode(inpt);
210     printf("\n");
211     event(pos);
212   }
213   return 0;
214 }
```

参考文献

[1] 田中賢一郎、「ゲームで学ぶ Java Script 入門」、インプレス