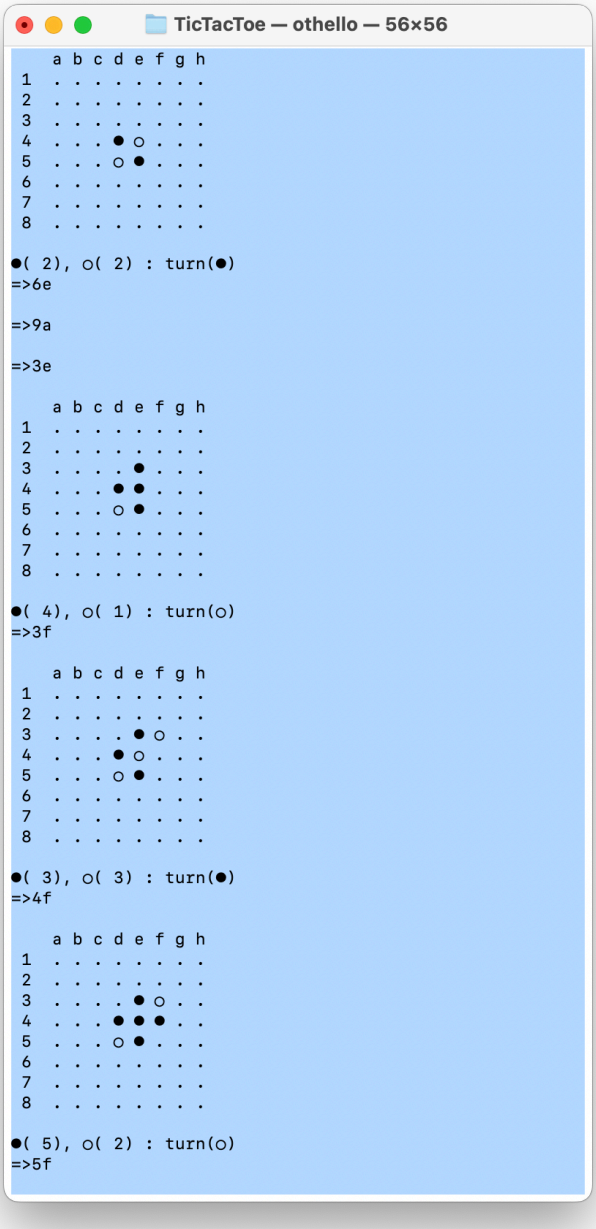
# C 言語：オセロ（リバーシ）

◯×工業高校 機械工学科 2 年

2023 年 10 月 8 日

# 1　ゲームの概要


TicTacToe — othello — 56×56
a b c d e f g h
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . . ● ○ . . .
5 . . . ○ ● . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .
●( 2), ○( 2) : turn(●)
=>6e
=>9a
=>3e
a b c d e f g h
1 . . . . . . . .
2 . . . . . . . .
3 . . . . ● . . .
4 . . . ● ● . . .
5 . . . ○ ● . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .
●( 4), ○( 1) : turn(○)
=>3f
a b c d e f g h
1 . . . . . . . .
2 . . . . . . . .
3 . . . . ● ○ . .
4 . . . ● ○ . . .
5 . . . ○ ● . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .
●( 3), ○( 3) : turn(●)
=>4f
a b c d e f g h
1 . . . . . . . .
2 . . . . . . . .
3 . . . . ● ○ . .
4 . . . ● ● ● . .
5 . . . ○ ● . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .
●( 5), ○( 2) : turn(○)
=>5f

## 2　各種宣言など

1. これは冒頭に記述する
2. BOOLEAN 型を enum で定義する
3. 盤面の幅 BOARDW は #define 文で指定する
4. 盤面に黒の石が置かれている場所には BLACK 、白の石が置かれている場所は WHITE、何も置かれていない場所は NONE で区別する
5. 現在注目している石の位置 (x,y) から見て、下の方向には (0,-1)、右下には (1,-1)、右には (0,1)、右上には (1,1)、上には (1,0)、左上には (-1,1)、左には (-1,0)、左下には (-1,-1) を加えることで、上下、左右、斜め右上下、斜め左上下の、全部で 8 つの方向の場所に着目できる
6. 盤面上の場所 (x,y) を指定するための構造体 POS を定義する
7. 盤面状態を保持している 1 次元配列 board 上の位置は、POS 型変数 (x,y) から $y \times BOARDW + x$ によって求める
8. 手番は変数 turn に保持する
9. パスの回数 passcount が 2 回になるとゲームは終了する
10. 終了フラグ endflag が偽である間ゲームは続く
11. setstone() 関数は盤面の指定位置に石を置く操作
12. getstone() 関数は盤面の指定位置の状態を知る操作

---

```
#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;
```

```
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      }
```

## 3 主処理

1. 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
2. decode() 関数を使って、入力された位置の文字列を POS 型変数に変換する（ここに 2 桁の行番号は想定していない）
3. 手番の石を置けない場所（盤面の外：!isinside()、又は反転できる石がない：flippable() がゼロ）を指定しても無視する
4. 現在位置 pos の周辺 8 方向に、それぞれ search() で反転できる石の数を数えて、それらを反転する

```
1       POS decode(char* str){
2           POS pos;
3           pos.x = atoi(str) - 1;
4           pos.y = *(str+1) - 'a';
5           return pos;
6       }
7
8       void event(POS pos){
9           if(endflag){
10              initboard();
11              return;
12          }
13          if(0<passcount){
14              nextturn();
15              drawboard();
16              return;
17          }
18          if(!isinside(pos))
19              return;
20          if(0==flippable(pos, turn))
21              return;
22          for(int i=0; i<8; i++){
23              int loop = search(pos, i, turn);
24              POS temp = pos;
25              for(int j=0; j<loop; j++){
26                  temp = movepos(temp, i);
27                  setstone(temp, turn);
28              }
29          }
```

```
        setstone(pos, turn);
        nextturn();
        drawboard();
    }

    int main(void){
        if(!initboard())
            return 8;
        char inpt[]="␣␣␣";
        while(!endflag){
            printf("=>");
            scanf("%s", inpt);
            POS pos = decode(inpt);
            printf("\n");
            event(pos);
        }
        return 0;
    }
```

# 4 初期化

1. ゲームの盤面の幅は、4 以上 8 以下の偶数である

```
    enum BOOLEAN initboard(){
        if((BOARDW%2)||(BOARDW<4)||(8<BOARDW))
            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;
        setstone(pos, WHITE);
        pos.x = pos.y = BOARDW/2;
        setstone(pos, BLACK);
        turn = BLACK;
        passcount = 0;
        endflag = false;
        drawboard();
        return true;
    }
```

# 5 盤面の表示

1. 盤面を表示するたびに、count() 関数で石の数を数えて表示する

```
    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");
    }
```

## 6 手番の交代

1. isinside() 関数は、指定された場所が盤の内部の位置かどうかを調べる
2. flippable() 関数は、相手の石を何枚反転させられるかを調べる
3. movepos() 関数は、着目点を現在位置 pos から上下左右斜めの 8 方向に移動させる
4. search() 関数は、v で指示された方向に何個の石を反転させられるかを数えている
5. nextturn() 関数では、
   石を置ける空いてる場所があるかどうかを調べて無かったら終了フラグを立てる
   指定位置 pos が相手の石を反転できる場所ならパスの回数をリセットする
   パスの回数が 2 回続いたら終了フラグを立てる

```
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      }
```

# 7　プログラムの全体

ソースコード 1　オセロ

```
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 /* ================================================================== */
31 void setstone(POS pos, int num){
32   int index = (pos.y * BOARDW) + pos.x;
33   board[index] = num;
34 }
35 /* ================================================================== */
36 int getstone(POS pos){
37   int index = (pos.y * BOARDW) + pos.x;
38   return board[index];
39 }
40 /* ================================================================== */
41 int count(int color){
42   int num=0;
43   for(int i=0; i<(BOARDW * BOARDW); i++)
44     if(board[i]==color)
45       num++;
46   return num;
47 }
48 /* ================================================================== */
49 void drawboard(){
50   for(int x=0; x<BOARDW; x++){
51     if(x==0){
52       printf("␣␣␣␣");
```

```
53        char a='a';
54        for(int i=0; i<BOARDW; i++)
55          printf("%c␣", a+i);
56        printf("\n");
57      }
58      printf("%2d␣␣", x+1);
59      for(int y=0; y<BOARDW; y++){
60        int index = y * BOARDW + x;
61        switch(board[index]){
62        case BLACK:printf("%s␣", TILE[BLACK]);break;
63        case WHITE:printf("%s␣", TILE[WHITE]);break;
64        case NONE: printf("%s␣", TILE[NONE]); break;
65        }
66      }
67      printf("\n");
68    }
69    printf("\n%s(%2d),␣%s(%2d)", TILE[BLACK],count(BLACK),TILE[WHITE],count(WHITE
        ));
70    if(!endflag)
71      printf("␣:␣turn(%s)\n", TILE[turn]);
72    else
73      printf("\n");
74  }
75  /* ================================================================== */
76  enum BOOLEAN initboard(){
77    if((BOARDW%2)||(BOARDW<4)||(8<BOARDW))
78      return false;
79    int x, y;
80    POS pos;
81    for(y=0; y<BOARDW; y++)
82      for(x=0; x<BOARDW; x++){
83        pos.x = x; pos.y = y;
84        setstone(pos, NONE);
85      }
86    pos.x = pos.y = BOARDW/2-1;
87    setstone(pos, BLACK);
88    pos.x = BOARDW/2; pos.y = pos.x-1;
89    setstone(pos, WHITE);
90    pos.y = BOARDW/2; pos.x = pos.y-1;
91    setstone(pos, WHITE);
92    pos.x = pos.y = BOARDW/2;
93    setstone(pos, BLACK);
94    turn = BLACK;
95    passcount = 0;
96    endflag = false;
97    drawboard();
98    return true;
99  }
100 /* ================================================================== */
101 POS movepos(POS pos, int v){
102   POS p;
103   p.x = pos.x + UNITV[v][0];
104   p.y = pos.y + UNITV[v][1];
105   return p;
106 }
107 /* ================================================================== */
108 enum BOOLEAN isinside(POS pos){
109   if( (pos.x<0) || (BOARDW<=pos.x) )
```

```
110       return false;
111     if( (pos.y<0) || (BOARDW<=pos.y) )
112       return false;
113     return true;
114   }
115   /* ================================================================== */
116   int search(POS pos, int v, int num){
117     int piece = 0;
118     while(true){
119       pos = movepos(pos, v);
120       if(!isinside(pos))
121         return 0;
122       if(getstone(pos)==NONE)
123         return 0;
124       if(getstone(pos)==num)
125         break;
126       piece ++;
127     }
128     return piece;
129   }
130   /* ================================================================== */
131   int flippable(POS pos, int num){
132     if(getstone(pos)!=NONE)
133       return 0;
134     int total = 0;
135     int vec[]={0,0};
136     for(int i=0; i<8; i++)
137       total += search(pos, i, num);
138     return total;
139   }
140   /* ================================================================== */
141   void nextturn(){
142     turn ^= 1;
143     int empty = 0;
144     for(int y=0; y<BOARDW; y++)
145       for(int x=0; x<BOARDW; x++){
146         POS pos; pos.x=x; pos.y=y;
147         if(getstone(pos)==NONE)
148           empty++;
149         if(0<flippable(pos,turn)){
150           passcount = 0;
151           return;
152         }
153       }
154     if(empty==0){
155       endflag = true;
156       return;
157     }
158     passcount++;
159     if(2<=passcount)
160       endflag = true;
161   }
162   /* ================================================================== */
163   POS decode(char* str){
164     POS pos;
165     pos.x = atoi(str)-1;
166     pos.y = *(str+1) - 'a';
167     return pos;
```

```
}
/* ========================================================================== */
void event(POS pos){
  if(endflag){
    initboard();
    return;
  }
  if(0<passcount){
    nextturn();
    drawboard();
    return;
  }
  if(!isinside(pos))
    return;
  if(0==flippable(pos, turn))
    return;
  for(int i=0; i<8; i++){
    int loop = search(pos, i, turn);
    POS temp = pos;
    for(int j=0; j<loop; j++){
      temp = movepos(temp, i);
      setstone(temp, turn);
    }
  }
  setstone(pos, turn);
  nextturn();
  drawboard();
}
/* ========================================================================== */
int main(void){
  if(!initboard())
    return 8;
  char inpt[]="␣␣␣";
  while(!endflag){
    printf("=>");
    scanf("%s", inpt);
    POS pos = decode(inpt);
    printf("\n");
    event(pos);
  }
  return 0;
}
```

## 参考文献

[1] 松原拓也（有限会社ニコ）、「Python でリバーシを作ろう」、日経ソフトウェア 2023 年 5 月号