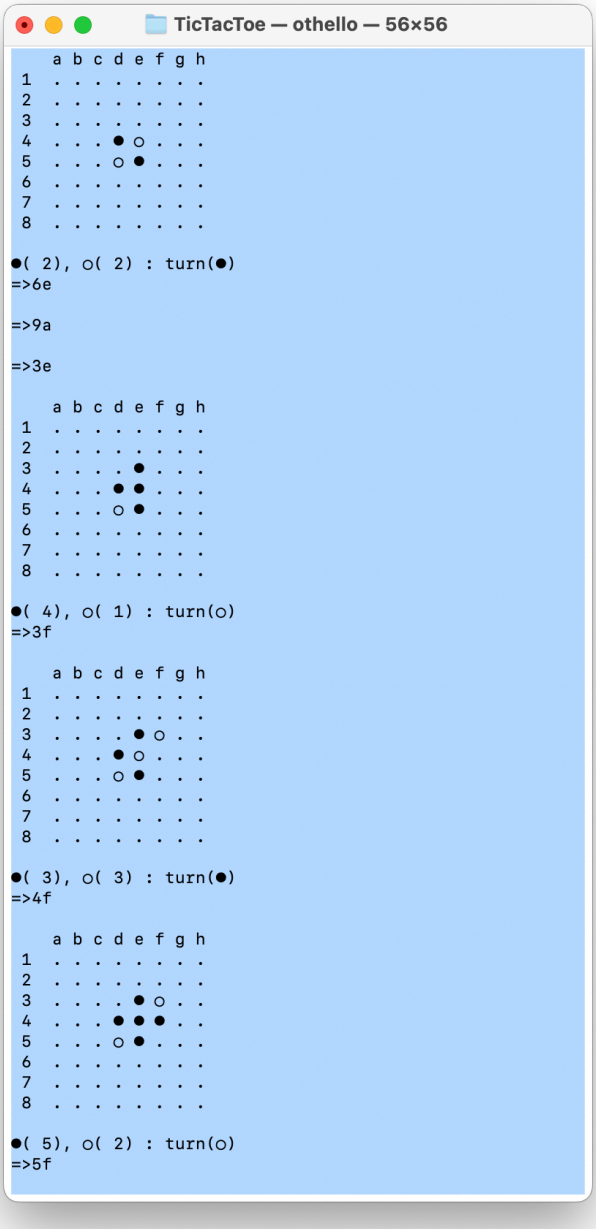
C 言語：オセロ（リバーシ）

◯×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 つの方向の場所を確認していくことができる
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 に保持している
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 endglag 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
};

```

```
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      }
```

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反転していく

```
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);
```

```
            }
        }
        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

```
1   enum BOOLEAN initboard(){
2       if((BOARDW%2)||(BOARDW<4)||(8<BOARDW))
3           return false;
4       int x, y;
5       POS pos;
6       for(y=0; y<BOARDW; y++)
7           for(x=0; x<BOARDW; x++){
8               pos.x = x; pos.y = y;
9               setstone(pos, NONE);
10          }
11      pos.x = pos.y = BOARDW/2-1;
12      setstone(pos, BLACK);
13      pos.x = BOARDW/2; pos.y = pos.x-1;
14      setstone(pos, WHITE);
15      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  }
```

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");
    }
```

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

```
POS movepos(POS pos, int v){
    POS p;
    p.x = pos.x + UNITV[v][0];
    p.y = pos.y + UNITV[v][1];
    return p;
}

enum BOOLEAN isinside(POS pos){
    if( (pos.x<0) || (BOARDW<=pos.x) )
        return false;
    if( (pos.y<0) || (BOARDW<=pos.y) )
        return false;
    return true;
}

int search(POS pos, int v, int num){
    int piece = 0;
    while(true){
        pos = movepos(pos, v);
        if(!isinside(pos))
            return 0;
        if(getstone(pos)==NONE)
            return 0;
        if(getstone(pos)==num)
            break;
        piece ++;
    }
    return piece;
}

int flippable(POS pos, int num){
    if(getstone(pos)!=NONE)
        return 0;
    int total = 0;
    int vec[]={0,0};
    for(int i=0; i<8; i++)
        total += search(pos, i, num);
    return total;
}

void nextturn(){
    turn ^= 1;
    int empty = 0;
    for(int y=0; y<BOARDW; y++)
        for(int x=0; x<BOARDW; x++){
            POS pos; pos.x=x; pos.y=y;
            if(getstone(pos)==NONE)
                empty++;
            if(0<flippable(pos,turn)){
                passcount = 0;
                return;
            }
        }
    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

```
#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;
}
```

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

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

```
168         if(0<flippable(pos,turn)){
169           passcount = 0;
170           return;
171         }
172       }
173     if(empty==0){
174       endflag = true;
175       return;
176     }
177     passcount++;
178     if(2<=passcount)
179       endflag = true;
180   }
181
182   /* ================================================================== */
183
184   POS decode(char* str){
185     POS pos;
186     pos.x = atoi(str)-1;
187     pos.y = *(str+1) - 'a';
188     return pos;
189   }
190
191   /* ================================================================== */
192
193   void event(POS pos){
194     if(endflag){
195       initboard();
196       return;
197     }
198     if(0<passcount){
199       nextturn();
200       drawboard();
201       return;
202     }
203     if(!isinside(pos))
204       return;
205     if(0==flippable(pos, turn))
206       return;
207     for(int i=0; i<8; i++){
208       int loop = search(pos, i, turn);
209       POS temp = pos;
210       for(int j=0; j<loop; j++){
211         temp = movepos(temp, i);
212         setstone(temp, turn);
213       }
214     }
215     setstone(pos, turn);
216     nextturn();
217     drawboard();
218   }
219
220   /* ================================================================== */
221
222   int main(void){
223     if(!initboard())
224       return 8;
225     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