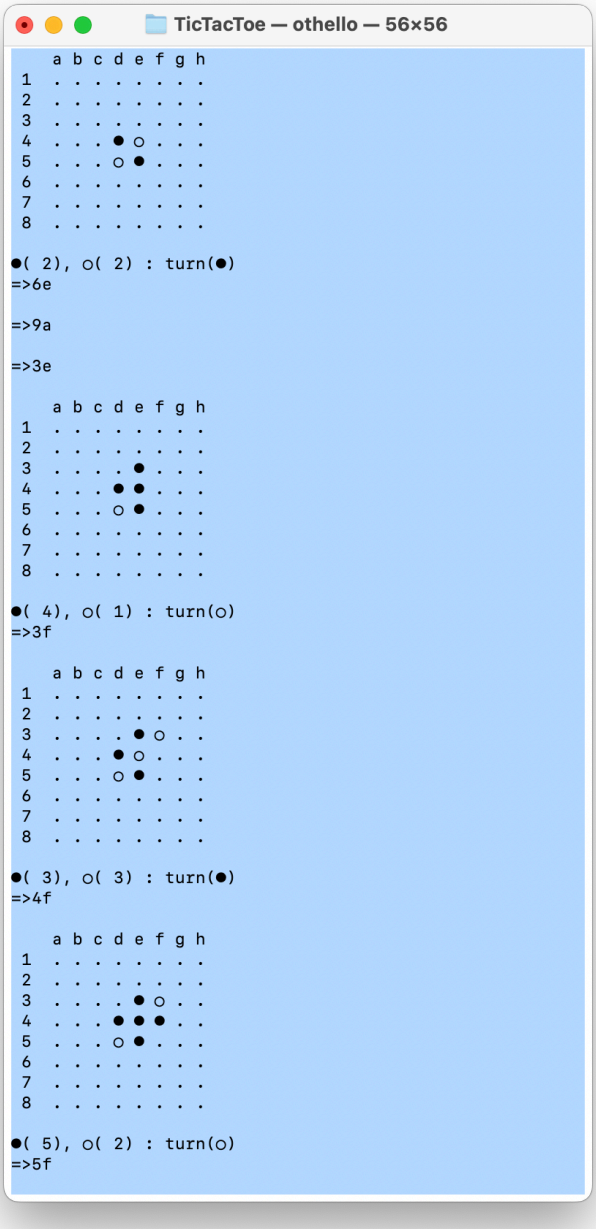
C 言語：オセロ（リバーシ）

◯×工業高校 機械工学科 2 年

2023 年 10 月 7 日

1　ゲームの概要



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。

```
    POS decode(char* str){
        POS pos;
        pos.x = atoi(str)-1;
        char* alpha="abcdefghijklmnopqrstuvwxyz";
        int i;
        for(i=0; i<BOARDW; i++){
            if(*(str+1)==alpha[i])
            break;
        }
        pos.y = i;
        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;
    }
```

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 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;
        setstone(pos, WHITE);
        pos.x = pos.y = BOARDW/2;
        setstone(pos, BLACK);
        turn = BLACK;
        passcount = 0;
        endflag = false;
        drawboard();
        return true;
    }
```

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。

```
    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;
```

```
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      }
```

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
```

```
        "."   //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];
    }
```

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;
}

```

```
35 int getstone(POS pos){
36   int index = (pos.y * BOARDW) + pos.x;
37   return board[index];
38 }
39
40 int count(int color){
41   int num=0;
42   for(int i=0; i<(BOARDW * BOARDW); i++)
43     if(board[i]==color)
44       num++;
45   return num;
46 }
47
48 void drawboard(){
49   for(int x=0; x<BOARDW; x++){
50     if(x==0){
51       printf("␣␣␣␣");
52       char a='a';
53       for(int i=0; i<BOARDW; i++)
54         printf("%c␣", a+i);
55       printf("\n");
56     }
57     printf("%2d␣␣", x+1);
58     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 }
```