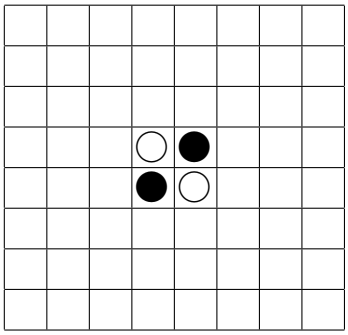
プログラミングおよび実習 I・第 2 回

2003 年 4 月 16 日

1　オセロゲームのルール

オセロゲームは 2 人のプレーヤーで行うゲームで、 8×8 のマスからなる格子状の盤と、白色と黒色の 2 つの面を持つ 64 個の小さな円盤 (石) を使用します。ゲームの開始時には、次の図のように盤の中央には白黒 2 つずつの石 (その色を表にした円盤) が置かれます。



2 人のプレーヤーは、それぞれ黒と白の石を、空いているマスに交互に置いていきます。ただし、

- 黒石を置くプレーヤーが先手です
- 置いた石と自分の石で (隙間なく) 挟まれた相手の石はすべて自分の石となります (円盤が反転して色が変ります)
- 相手の石を少なくとも 1 つは挟むように石を置かなければなりません
- そのようなマスがない場合は自分の手番をパスします
- 両方のプレーヤーが石を置けなくなったらゲームが終了します
- ゲームが終了したときに盤面に多くの石があった方のプレーヤーの勝ちとなります

2　盤面の状態の表現とその表示

2.1　2 次元配列

オセロゲームの盤面の状態をプログラム中で記憶しておくためには **2 次元配列**を使うのが便利です。これまでに使った配列は同じデータ型の変数を 1 次元的に並べたものでしたが、 2 次元配列は縦横に 2 次元的に並べたものとなります。 2 次元配列の宣言は次のように行います。

```
int a[10][50];
```

この宣言によって、 10 × 50 個の `int` 型のデータを格納できるような 2 次元配列 `a` の記憶領域が確保されます。 2 次元配列の要素は、配列名に続いて [ ] で囲んだ添字を 2 つ並べて書き、`a[0][0]` や `a[8][23]`、`a[9][49]` のようにして識別します。それぞれの要素をあたかも `int` 型の変数であるかのように利用することが可能です。 2 つの添字はどちらも 0 から始まることに注意してください。

2.2 初期状態の設定とその表示

次のプログラム (othello01.c) では、8×8 の int 型の要素からなる 2 次元配列 board を使って、オセロゲームの盤面の状態を表現しようとしています。オセロゲームの盤の y 行 x 列のマスの状態 (石がない、黒石がある、白石がある) を、2 次元配列の要素 board[$x-1$][$y-1$] に int 型の値として保持しています。

このプログラムでは、まず、すべてのマスを石がない状態にしてから盤の中央に黒石と白石を 2 つずつ置き、ゲーム盤の初期状態を配列 board に作っています。その後、この配列 board に保持されている盤の状態に従って盤面の表示を行っています。

othello01.c

```
 1 #include <stdio.h>
 2
 3 #define BLACK           (1)     /* 黒石を表す定数 */
 4 #define WHITE           (-1)    /* 白石を表す定数 */
 5 #define NONE            (0)     /* 石がないことを表す */
 6
 7 #define BOARD_SIZE      (8)     /* オセロボードの縦横のマス数 */
 8
 9 int main()
10 {
11     /* 盤面の状態 (BLACK/WHITE/NONE) */
12     int board[BOARD_SIZE][BOARD_SIZE];
13     int x, y;
14
15     /* すべてのマスをクリアする */
16     for (x = 0; x < BOARD_SIZE; x++) {
17         for (y = 0; y < BOARD_SIZE; y++) {
18             board[x][y] = NONE;
19         }
20     }
21
22     /* 中央に4つの石を置く */
23     board[BOARD_SIZE/2-1][BOARD_SIZE/2-1] = WHITE;
24     board[BOARD_SIZE/2-1][BOARD_SIZE/2] = BLACK;
25     board[BOARD_SIZE/2][BOARD_SIZE/2-1] = BLACK;
26     board[BOARD_SIZE/2][BOARD_SIZE/2] = WHITE;
27
28     /* 盤面を表示する */
29     for (y = 0; y < BOARD_SIZE; y++) {
30         for (x = 0; x < BOARD_SIZE; x++) {
31             if (board[x][y] == BLACK)
32                 printf("●");
33             else if (board[x][y] == WHITE)
34                 printf("○");
35             else
36                 printf("□");
37         }
38         printf("\n");
39     }
40
41     /* プログラムを終了する */
42     return 0;
```

[1] 1-609 実習室の Linux 環境では、「○」や「●」、「□」などの記号は、日本語入力の状態で「Ctrl」キーを押しながら「^」キーを押すと一覧から選んで入力することができます。

```
43 }
```

このプログラムをコンパイルして実行すると次のようになります。

```
s1609h017% cc othello01.c -o othello01
s1609h017% ./othello01
□□□□□□□□
□□□□□□□□
□□□□□□□□
□□□○●□□□
□□□●○□□□
□□□□□□□□
□□□□□□□□
□□□□□□□□
s1609h017%
```

3 演習問題

3.1 例題プログラムを試す

講義で解説した othello01.c というプログラムを自分で作り、コンパイル、実行してみなさい。完成したら、「mprog1 othello01.c」のようにして mprog1 コマンドを実行し、正しくできているかどうかチェックしてください。正しくできている場合は、そのプログラムが提出されたことになります。プログラムは Prog1 というディレクトリに作ることを忘れないようにしましょう。以下の 3 つの演習問題も同様です。

3.2 列や行の番号の表示

まず、othello01.c を othello02.c にコピーしなさい。その後 othello02.c を変更して、列や行の番号を (次の例のように) 表示するようにしなさい。ただし、BOARD_SIZE のマクロ定義や配列 board の大きさは変えないでプログラムを作りなさい。

```
s1609h017% cc othello02.c -o othello02
s1609h017% ./othello02
 1 2 3 4 5 6 7 8
1□□□□□□□□
2□□□□□□□□
3□□□□□□□□
4□□□○●□□□
5□□□●○□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
s1609h017%
```

3.3 石を置く位置の入力

まず、othello02.c を othello03.c にコピーしなさい。その後 othello03.c を変更して、次の実行例[1]のように、最初に石を置く位置 (列番号と行番号) を入力し、入力で指定された位置[2]に黒石を置いた状態を表示するようにしなさい。

ただし、すでに石が置かれているマスにも石を置いてもよいものとし、石が挟まれても反転しないものとします。盤外の列番号や行番号が入力されたときは、 ユーザーにもう一度入力を促すようにしなさい。

```
s1609h017% cc othello03.c -o othello03
s1609h017% ./othello03
列 = 4
行 = 3
 1 2 3 4 5 6 7 8
1□□□□□□□□
2□□□□□□□□
3□□□●□□□□
4□□□○●□□□
5□□□●○□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
s1609h017% ./othello03
列 = 0
列 = 5
行 = 1
 1 2 3 4 5 6 7 8
1□□□□●□□□
2□□□□□□□□
3□□□□□□□□
4□□□○●□□□
5□□□●●□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
s1609h017%
```

3.4 繰り返し石を置く

まず、othello03.c を othello04.c にコピーしなさい。その後 othello04.c を変更して、次の実行例のように、まず初期状態を表示した後、位置の入力とそこに黒石を置いた状態の表示を永遠に繰り返すようにしなさい[3]。othello03.c と同様に、盤外の列番号や行番号が入力されたときは、ユーザーにもう一度入力を促すようにします。また、すでに石が置かれているマスにも気にしないで石を置くようにし、石が挟まれても反転させません。

```
s1609h017% cc othello04.c -o othello04
s1609h017% ./othello04
 1 2 3 4 5 6 7 8
```

[1]実行例の中の「列 = 」や「行 = 」はこのプログラム (othello03) が出力したもので、それに続く整数はユーザーが入力したものです。

[2]列や行の番号は 1 〜 8 で、配列 board の添字は 0 〜 7 であることに注意しましょう。

[3]プログラムは割り込みキー (Ctrl+C) で終了させるものとします。

```
1□□□□□□□□
2□□□□□□□□
3□□□□□□□□
4□□□○●□□□
5□□□●○□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
列 = 4
行 = 3
 1 2 3 4 5 6 7 8
1□□□□□□□□
2□□□□□□□□
3□□□●□□□□
4□□□○●□□□
5□□□●○□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
列 = 0
列 = 5
行 = 2
 1 2 3 4 5 6 7 8
1□□□□□□□□
2□□□□●□□□
3□□□●□□□□
4□□□○●□□□
5□□□●●□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
列 = 7
行 = 9
行 = 2
 1 2 3 4 5 6 7 8
1□□□□□□□□
2□□□□●□●□
3□□□●□□□□
4□□□○●□□□
5□□□●●□□□
6□□□□□□□□
7□□□□□□□□
8□□□□□□□□
列 = ^C
s1609h017%
```

次回の予定

次回は `othello04.c` をさらに発展させて、つぎのようなことを行う予定です。

- すでに石のあるマスには石を置けないようにします
- 黒石と白石を交互に置くようにします
- 盤面の表示を行う部分を関数として独立させます

プログラミングおよび実習 I・第 2 回・終り