19　オセロゲーム１

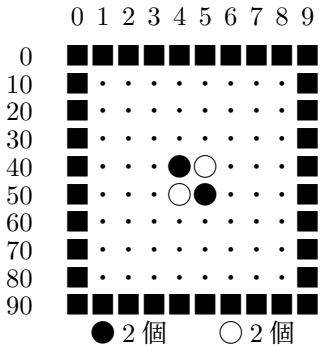
今回から本格的なプログラムとして，オセロゲームを作っていきます。

大きなプログラムを作る時は，一度に完成品を作ろうとするのではなく，少しずつ積み重ねて作っていきます。一度に多くのものを作ると，思い通りに動作しない時，どこが間違っているか見つけるのが大変です。少しずつ機能を追加して，その都度完成させて行くと，苦労が半分で済みます。

19.1　盤の内部表現と外部表現

オセロゲームは，8×8 のマス目をもつ盤を使いますが，プログラムでは外側に “外” を表すマスをつけ加えた 10×10 のマス目をもった盤を使います。“外” を表すマスがあるとプログラムが簡単になるのです。

19.1.1　外部表現



●の番です

図のように，10×10 のマスがあり，それぞれが，外（■）と空（・），黒（●）と白（○）のいずれかの状態になっています。盤の情報の他に，●と○の個数，手番（今度コマを置く番の人）が誰か（●か○）の情報もあります。画面にはありませんが，手番の相手が誰か（手番と黒白が反対）という情報ももっています。

19.1.2　内部表現

外部表現にある情報を内部のデータとして記憶していないといけません。

盤は外部表現では 10×10 ですが，内部では 1×100 の 1 次元配列とします。その方が，斜め方向のコマを裏返すなどの操作が簡単になるからです。マスには左上から順に $00, 01, 02, \ldots, 99$ の通し番号がついています。外部表現の盤の上と左に数が書いてありますが，１つのマスの番号は，そのマスがある行の左の数（10 の倍数）と列の上の数（一の位）を足した数になります。

例 19.1

(1) 外マスの番号は，$0\cdot, 9\cdot, \cdot0, \cdot9$ です。

(2) 開始状態（上の図）で，●があるマスは 44 番と 55 番，○があるマスは 45 番と 54 番です。

プログラムの先頭で，盤の内部表現を記憶するためのデータ構造を定義します。

```
program Othello1; // 学生証番号　氏名
{$APPTYPE CONSOLE}
uses SysUtils;

type
  TMasuNo = 0..99;                          // 盤のマスの番号
  TYouso  = (Soto,Kara,Kuro,Siro);          // 盤の要素（外，空，●，○）
  TBan    = array [TMasuNo] of TYouso;      // 盤（10 × 10）
  TPlayer = Kuro..Siro;                     // プレイヤー（要素名で表す）
  TKosuu  = array [TPlayer] of Integer;     // 各プレイヤーのコマの個数

var
  Ban    : TBan;                            // 盤
  Teban  : TPlayer;                         // 手番（今度コマを置く番の人）
  Aite   : TPlayer;                         // 手番でない方の人
  Kosuu  : TKosuu;                          // コマの個数
```

19.1.3　初期設定と画面表示

まずは，ゲームの開始状態を設定し，画面に表示します。

```
procedure SyokiSettei;
          { 初期設定                          }
          { 盤を開始状態にして，手番を●にする }
  var
    No  : TMasuNo;      // マスの番号
    Ten : TMasuNo;      // マスの番号の十の位
    One : TMasuNo;      //            一の位
  begin
    // Ban[00]～Ban[99] をすべて Kara にする
    // Ban[0.], Ban[9.], Ban[.0], Ban[.9] をすべて Soto に変える
    // Ban[44], Ban[55] を Kuro に変える
    // Ban[45], Ban[54] を Siro に変える
    // Kosuu[Kuro], Kosuu[Siro] を 2 にする
    // Teban を Kuro にする
    // Aite  を Siro にする
  end; {SyokiSettei}

procedure WriteBan;
          { 盤を書く }
  var
    No : TMasuNo;        // マスの番号
  begin
    WriteLn;
    Write('':22);
    for No := 0 to 9 do
      begin
        Write(No:2);
      end;
    WriteLn;
    for No := 0 to 99 do
```

```
      begin
        if No mod 10 = 0
          then Write(No:20, '  ');
        case Ban[No] of
          Soto : Write('■');
          Kara : Write('・');
          Kuro : Write('●');
          Siro : Write('○');
        end;
        if No mod 10 = 9
          then WriteLn;
      end;
    Write(''':22);
    Write('  ●', Kosuu[Kuro]:2, '個  ');
    Write('  ○', Kosuu[Siro]:2, '個  ');
    WriteLn;
    if Teban = Kuro
      then WriteLn('●の番です')
      else WriteLn('○の番です');
  end; {WriteBan}

procedure Game;
          { １ゲーム行う }
  begin
    SyokiSettei;
    WriteBan;
  end; {Game}

begin {Main}
  Game;
  ReadLn;
end.
```

問 19.1 プロシージャ SyokiSettei の実行部をコメントに従って作成しなさい。

19.2　コマを置く

どこに置くか問い合わせて，そこに置きます。置けない場合は再度置く場所を問い合わせます。とりあえず，相手のコマを挟まなくても，空のマスならどこにも置けることにします。

```
  end; {WriteBan}      // これより上は変更なし

function  ReadMasuNo : TMasuNo;
          { どこに置くか問い合わせて，マスの番号読む     }
          { マスの番号でない数を入れたら再度問い合わせる }
  var
    No  : Integer;     // 読んだ数
  begin
    // どこに置くか問い合わせて，No を読む
    // No が 0〜99 になるまで繰り返す
    // No を関数の戻り値とする
  end; {ReadMasuNo}

function  Okeru(No : TMasuNo) : Boolean;
```

```
          { Ban[No] にコマを置けるか          }
          { とりあえず，空なら置けることにする }
  var
    Ok : Boolean;
  begin
    Ok := False;
    if Ban[No] = Kara
      then Ok := True;
    Okeru := Ok;
  end; {Okeru}

procedure Oku(No : TMasuNo);
          { Ban[No] に手番のコマを置く }
          { とりあえず，置くだけ       }
  begin
    Ban[No] := Teban;
    Kosuu[Teban] := Kosuu[Teban]+1;
  end; {Oku}

procedure OneMove;
          { １手行う }
  var
    No : TMasuNo;
  begin
    repeat
      No := ReadMasuNo;
    until Okeru(No);
    Oku(No);
  end; {OneMove}

procedure Hanten(var Player : TPlayer);
          { 白黒反転する }
  begin
    case Player of
      Kuro : Player := Siro;
      Siro : Player := Kuro;
    end;
  end; {Hanten}

procedure OneGame;
          { １ゲーム行う }
  begin
    SyokiSettei;
    WriteBan;
    repeat
      OneMove;
      Hanten(Teban);
      Hanten(Aite);
      WriteBan;
    until False;    // とりあえず，終了条件はなし
  end; {OneGame}

begin {Main}  // これより下は変更なし
```

問 19.2 ファンクション ReadMasuNo の実行部を完成させなさい。

19.3　コマを置けるか判定する

コマを置けるのは，相手のコマを 1 つでも裏返せるときに限ります。したがって，マスの番号を指定された時，そこが空であることはもちろん，さらに 8 方向について裏返せるコマがあるかどうか調べないといけません。

マスは (行, 列) でなく通し番号で識別するようにしたので，方向は番号がいくつずつ変化するか指定すればわかります。たとえば上方向は −10 ずつ変化していくので，`Muki = -10` となります。

まだ，置けるかどうか判定して置くだけで，相手のコマを裏返すことはしません。

```
  end; {ReadMasuNo} // これより上は変更なし

function Uragaeseru(No : TMasuNo; Muki : Integer) : Boolean;
         { Ban[No] にコマを置いたとして，Muki の方向の相手のコマを裏返せるか}
  var
    Ok   : Boolean;
    N    : TMasuNo;
  begin
    N := No+Muki;                                  // Muki 方向に 1 マス進む
    if Ban[N] <> Aite                              // 相手のコマ以外か
      then Ok := False                             // 相手のコマでなければダメ
      else begin                                   // 相手のコマのとき
             // 相手のコマ以外のマスに到達するまで Muki 方向に進んでいく
             // （「1 マス進む」を繰り返す）
             // そこが自分のコマならよし，それ以外ならダメ
             // （Ok を設定する）
           end;
    Uragaeseru := Ok;
  end; {Uragaeseru}

function  Okeru(No : TMasuNo) : Boolean;
          { Ban[No] にコマを置けるか            }
  var
    Ok : Boolean;
  begin
    Ok := False;
    if Ban[No] = Kara
      then Ok := True;
    Okeru := Ok and (Uragaeseru(No,-10) or
                     Uragaeseru(No,-1)  or
                     Uragaeseru(No,+1)  or
                     Uragaeseru(No,+10) );
  end; {Okeru}

procedure Oku(No : TMasuNo);  // これより下は変更なし
```

問 19.3

(1) プロシージャ `Uragaeseru` を完成させなさい。

(2) このままでは上下左右の方向しか調べていません。斜め方向も調べるように追加しなさい。

19.4　相手のコマを裏返す

それでは，コマを置く前に相手のコマを裏返しましょう。

```
  end; {Okeru}     // これより上は変更なし

procedure UragaesetaraUragaesu(No : TMasuNo; Muki : Integer);
          { Muki 方向の相手のコマを裏返せたら裏返す }
  var
    N  : TMasuNo;
  begin
    if Uragaeseru(No,Muki)              // この方向の相手のコマを裏返せるか
      then begin                        // 裏返せるとき
             // 隣のマスから始めて相手のコマを裏返すことを繰り返す
             // コマの数 Koma[] も変更する
           end;
  end; {UragaesetaraUragaesu}

procedure Oku(No : TMasuNo);
          { Ban[No] に手番のコマを置く }
  begin
    Ban[No] := Teban;
    // 8 方向について，裏返せたら裏返す
    Kosuu[Teban] := Kosuu[Teban]+1;
  end; {Oku}

procedure OneMove; // これより下は変更なし
```

問 19.4

(1) プロシージャ UragaesetaraUragaesu を完成させなさい。

(2) プロシージャ Oku を完成させなさい。

つづく