情報処理概論

第１４回　演習２

情報基盤研究開発センター　谷本 輝夫

今回の内容

- 四目並べ：Drop ルーチン（例）
- 四目並べ：勝敗判定

先週の演習の解答例

- 四目並べの手の入力
  - 入力のチェック
  - 列が一杯かどうか？
  - 盤の更新

```
  1  2  3  4  5  6  7
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][*][ ][ ][ ][ ]
 [ ][ ][o][*][ ][ ][ ]
 [*][*][o][o][o][*][ ]
o : Drop where? (0 = Exit) 2

  1  2  3  4  5  6  7
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][ ][ ][ ][ ][ ]
 [ ][ ][*][ ][ ][ ][ ]
 [ ][o][o][*][ ][ ][ ]
 [*][*][o][o][o][*][ ]
* : Drop where? (0 = Exit)
```

入力のチェック

- 入力された値が
  - ０の場合　→　終了
  - １～７以外　→　再入力
  - １～７の場合
    - 玉が落とせるか？
      - 落とせなかった → 再入力
      - 落とせた → 盤面の更新
- 再入力ってことは？
  - ループが必要
  - ループを抜ける条件は？
    - ０の場合？
    - 玉が落とせる場合

入力のチェック

- 入力された値が
  - 0の場合　→　終了
  - 1～7以外　→　再入力
  - 1～7の場合
    - 玉が落とせるか？
      - 落とせなかった → 再入力
      - 落とせた → 盤面の更新
- 再入力ってことは？
  - ループが必要
  - ループを抜ける条件は？
    - 0の場合？
    - 玉が落とせる場合

```
do while (再入力なら)
  read x

  if (x == 0) then
    終了
  end if

  if (x < 1 .or. x > 7) then
    範囲外の表示
    再入力
  else if (玉が落とせない？) then
    落とせない旨の表示
    再入力
  else
    盤面の更新
    ループを抜ける
  end if

end do
```

入力のチェック

1. フラグ（センチネル、番兵）を用いる
   - 事前に done = 0
   - 玉を落とせたら done=1
   - done==1なら、ループを抜ける

```
done = 0
do while (done == 0)
  read x

  if (x == 0) then
    stop
  end if

  if (x < 1 .or. x > 7) then
    範囲外の表示
    再入力
  else if (玉が落とせない？) then
    落とせない旨の表示
    再入力
  else
    盤面の更新
    done = 1
  end if

end do
```

入力のチェック

1. フラグ（センチネル、番兵）を用いる
   - 事前に done = 0
   - 玉を落とせたら done=1
   - done==1なら、ループを抜ける
2. exit, cycleなどを用いる
   - exitはループを抜ける
   - cycleはループの頭に戻る

```
do
  read x

  if (x == 0) then
    stop
  end if

  if (x < 1 .or. x > 7) then
    範囲外の表示
    cycle
  else if (玉が落とせない？) then
    落とせない旨の表示
    cycle
  else
    盤面の更新

  end if

end do
```

入力のチェック

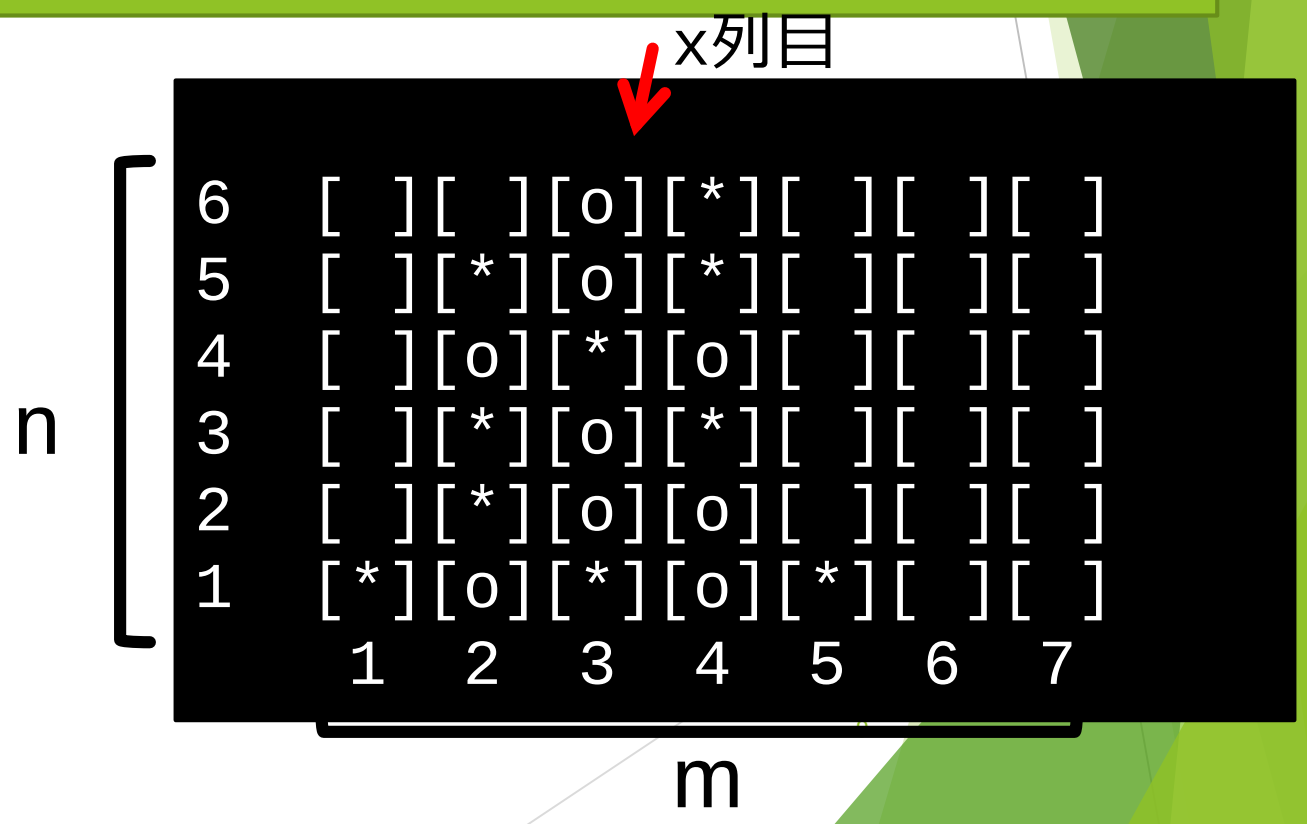
1. フラグ（センチネル、番兵）を用いる
   - 事前に done = 0
   - 玉を落とせたら done=1
   - done==1なら、ループを抜ける
2. exit, cycleなどを用いる
   - exitはループを抜ける
   - cycleはループの頭に戻る
3. goto文を用いる（非推奨）
   - 任意の場所にジャンプ
   - プログラムが構造化しにくい

```
done = 0
do while (done == 0)
  read x

  if (x == 0) then
    stop
  end if

  if (x < 1 .or. x > 7) then
    範囲外の表示
    goto 100
  else if (玉が落とせない？) then
    落とせない旨の表示
    goto 100
  else
    盤面の更新
    done = 1
  end if

end do

100 write…
```

入力のチェック：球が落とせない？

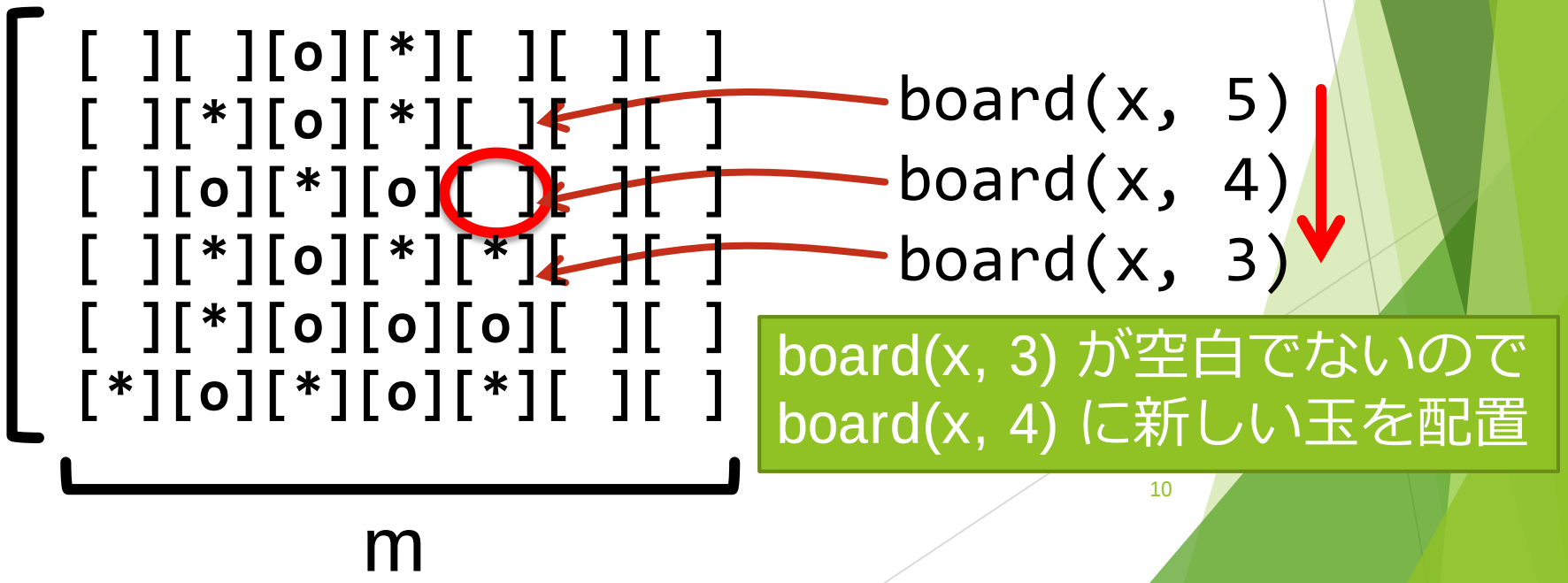
- 指定された列の一番上が空かどうか？

```
if (board(x, n) /= ' ') then
  write(*, *) '  This column is full!!'
```



盤面の更新

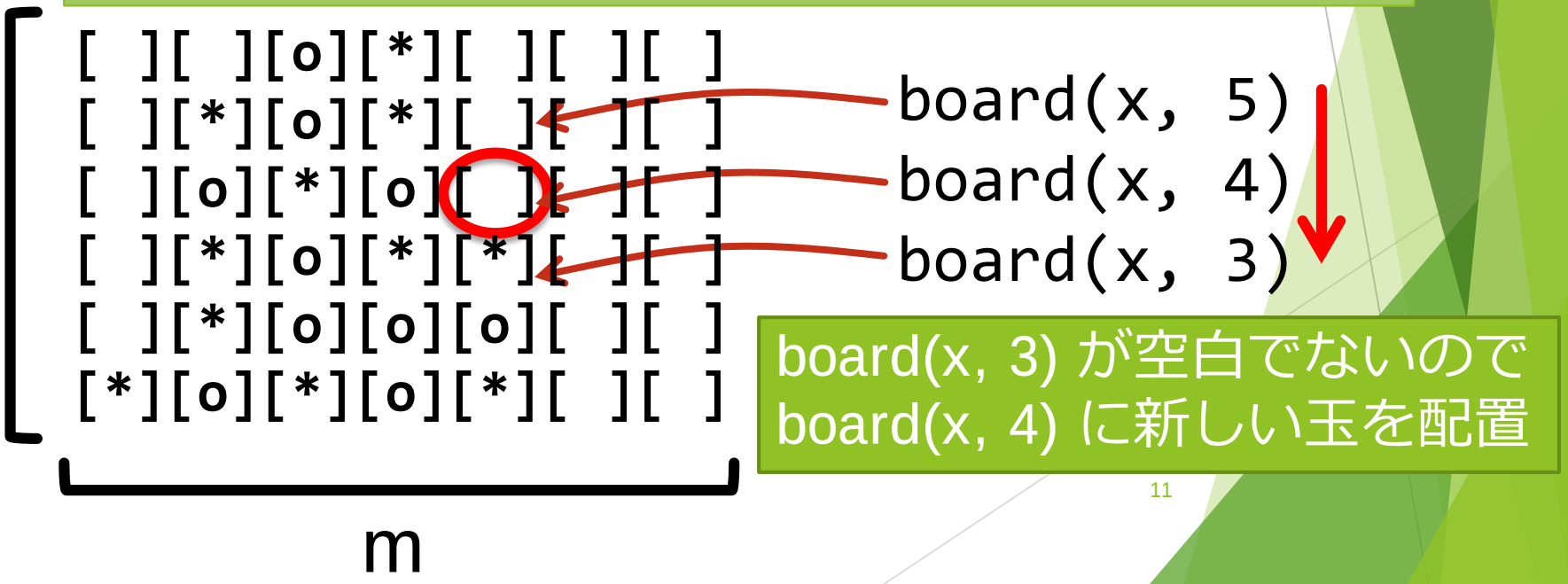
- 指定された列の空白な要素のうち一番下の要素に新しい玉を追加
  - 例）x列目を上から下へ調べてゆき、「一つ下の要素が空白以外」である要素が見つかったら、そこに新しい玉を配置



盤面の更新

- 指定された列の空白な要素のうち一番下の要素に新しい玉を追加

```
y = n
do while ((y  >  1)  .and.  (board(x, y-1)  ==  ' '))
  y = y - 1
end do
board(x, y) = mark
```



board(x, 3) が空白でないので
board(x, 4) に新しい玉を配置

サブルーチン drop

```
subroutine drop(m, n, board, side)
implicit none
integer, intent(IN) :: m, n, side
character(len=1), dimension(m, n), intent(INOUT) :: board
integer :: x, y, done
character(len=1) :: mark

  select case (side)
  case (1)
    mark = '*'
  case (2)
    mark = 'o'
  end select

  done = 0

  do while (done == 0)
    write(*, '(a, a)', advance='NO') &
          mark, ' : Drop where? (0 = Exit) '
    read(*, *) x
```

サブルーチン drop

```
    if (x == 0) then
       stop
    else if (x > m .or. x < 1) then
      write(*, *) '  Out of range!!'
    else if (board(x, n) /= ' ') then
      write(*, *) '  This column is full!!'
    else
      y = n
      do while ((y > 1) .and. (board(x, y-1) == ' '))
        y = y - 1
      end do
      board(x, y) = mark
      done = 1
    end if
  end do
end subroutine drop
```

サブルーチン drop (別の方法)

```
    if (x == 0) then
       stop
    else if (x > m .or. x < 1) then
      write(*, *) '  Out of range!!'
    else
      y = 1
      do while ((y <= n) .and. (board(x, y) /= ' '))
        y = y + 1
      end do
      if (y <= n) then
        board(x, y) = mark
        done = 1
      else
        write(*, *) '  This column is full!!'
      end if
    end if
  end do
end subroutine
```

指定された列を下から上に調べてゆき、空白の要素が見つかったらそこに新しい玉を配置する

今回の内容

- 四目並べ：Drop ルーチン（例）
- 四目並べ：勝敗判定

勝敗判定

- 縦、横、斜めに、同じ玉が4つ並ぶとその玉のプレーヤの勝ち
- 盤面の全ての縦、横、斜めをチェック！？
  - それでもO.K.　（基本）

- 新しく追加された玉を含む列だけでもO.K.　（発展）
  - 例えば、(3, 2) に玉が追加された
  - 横：(i, 2) iを１からmまで変化させて、並びを数える
  - 縦：(3, j) jを１からnまで変化させて、並びを数える
  - 斜めは？

メインプログラム

```
program four
  implicit none
  integer, parameter :: m=7, n=6
  character(len=1), dimension(m, n) :: board
  integer :: step, side

  board = ' '
  call show(m, n, board)

  do step = 1, m*n/2
     do side =1, 2
        call drop(m, n, board, side)
        call show(m, n, board)
        call check(m, n, board, side)
     end do
  end do
stop
end program
```

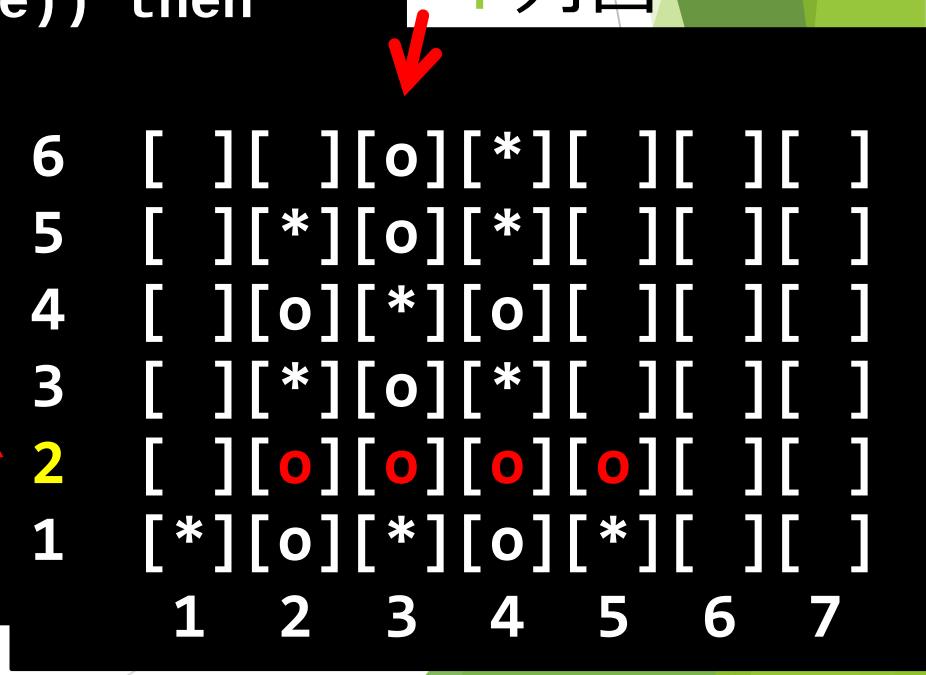
勝敗判定サブルーチン（枠組み）

```
subroutine check(m, n, board, side)
  implicit none
  character(1), dimension(m, n), intent(INOUT) :: board
  integer, intent(IN) :: m, n, side
  integer :: i, j, count
  character(1), dimension(2) :: ball = (/'*', 'o'/)

!横
!縦
!右上がり
!右下がり



endsubroutine check
```

ボールが４つ並んでいるか？

- 連続している数を数えれば良い
  - 横に並んでいる場合を考える
  - まずは２行目だけ

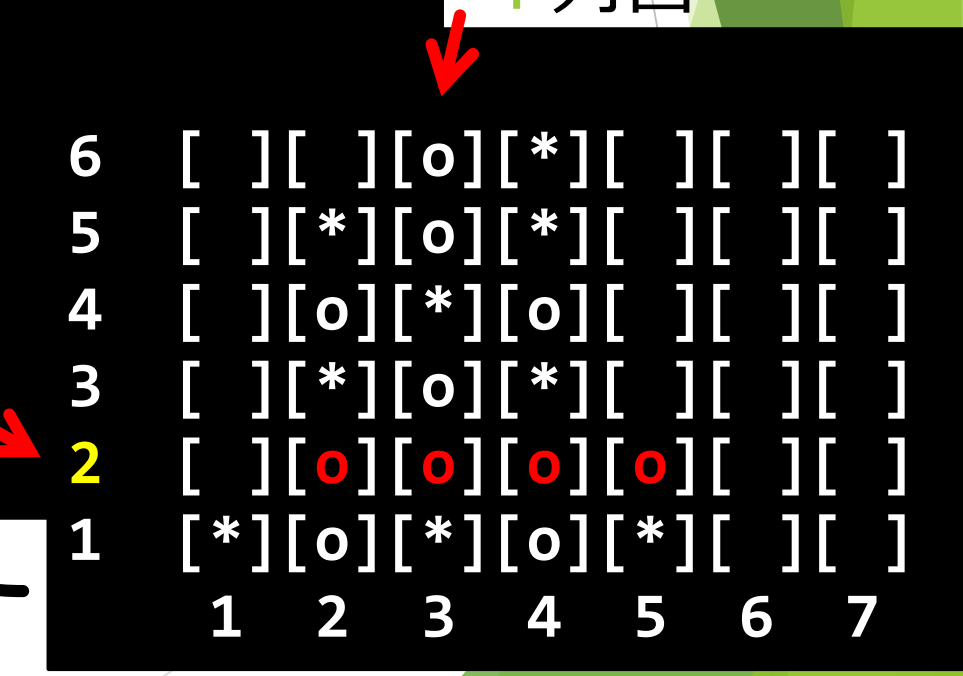
```
count = 0
do i =1, m
  if (board(2, i)  == ball(side)) then
    count = count + 1
    if (count == 4) then
      write (*,*) "won!"
      stop
    end if
  else
    count = 0
  end if
end do
```



ボールが４つ並んでいるか？

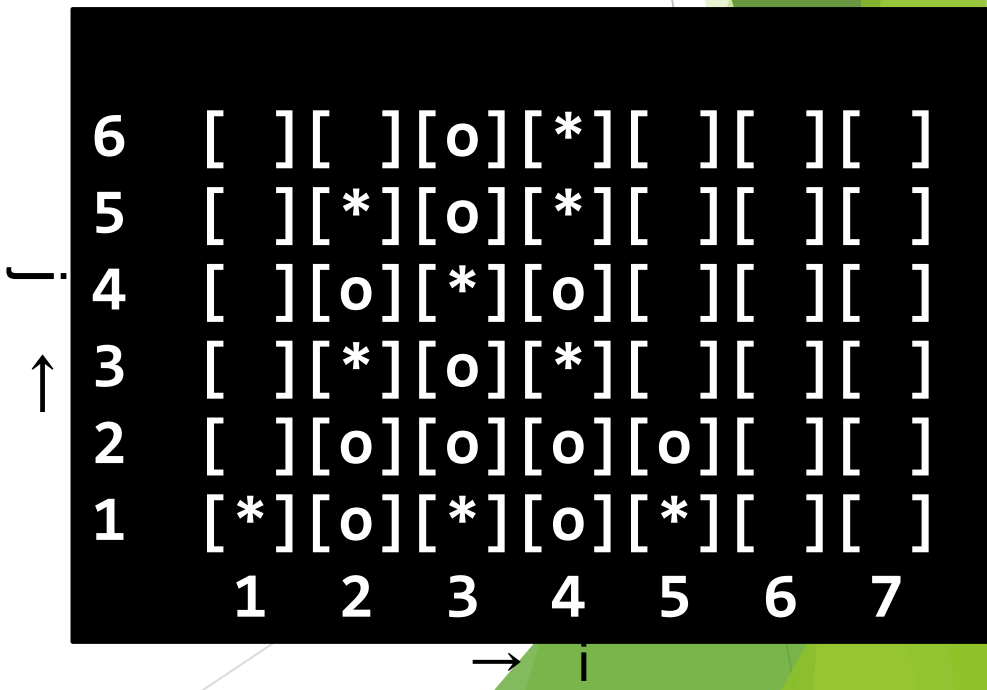
- 次に１行目から６行目まで全部チェックするには？

```
count = 0
do i =1, m
  if (board(2, i)  == ball(side)) then
    count = count + 1
    if (count == 4) then
      write (*,*) "won!"
      stop
    end if
  else
    count = 0
  end if
end do
```



縦はどう考える？

- まずは３列目を考える…　横とほぼ同じプログラム
- 次にそれを、1列目から７列目まで順番にチェック

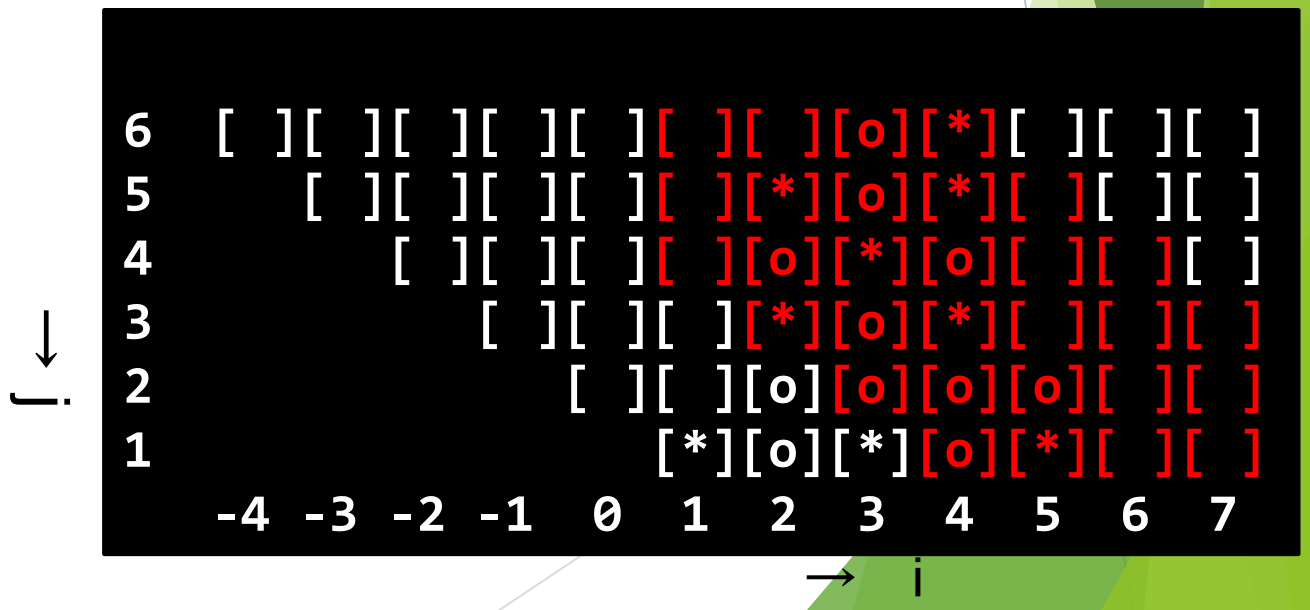


斜め（右上がり）はどう考える？

- かなりメンドクサイ…
  - 斜めに４マス以上ある部分だけチェックする
  - iを１増やす時に、jも１増やす
  - 配列範囲外を見ようとするとエラー

斜め（右下がり）はどう考える？

- こちらも、かなりメンドクサイ…
  - 同じく斜めに４マス以上ある部分だけチェックする
  - iを１増やす時に、jは１減らす
  - 配列範囲外を見ようとするとエラー



勝敗判定サブルーチン（別タイプ）

- 並びをチェックするのは、dropしたボールの周辺だけで良い！
  - → dropしたボールの情報を受け取る！
  - → dropサブルーチンから？　メインルーチンから？

```
subroutine check(x, y, m, n, board, side)
  implicit none
  character(1), dimension(m, n), intent(INOUT) :: board
  integer, intent(IN) :: x, y, m, n, side
  integer :: i, j, count
  character(1), dimension(2) :: ball = (/'*', 'o'/)

!横　　（縦の位置は　y　に固定して良い）
!縦　　（横の位置は　x　に固定して良い）
!右上がり
!右下がり

endsubroutine check
```