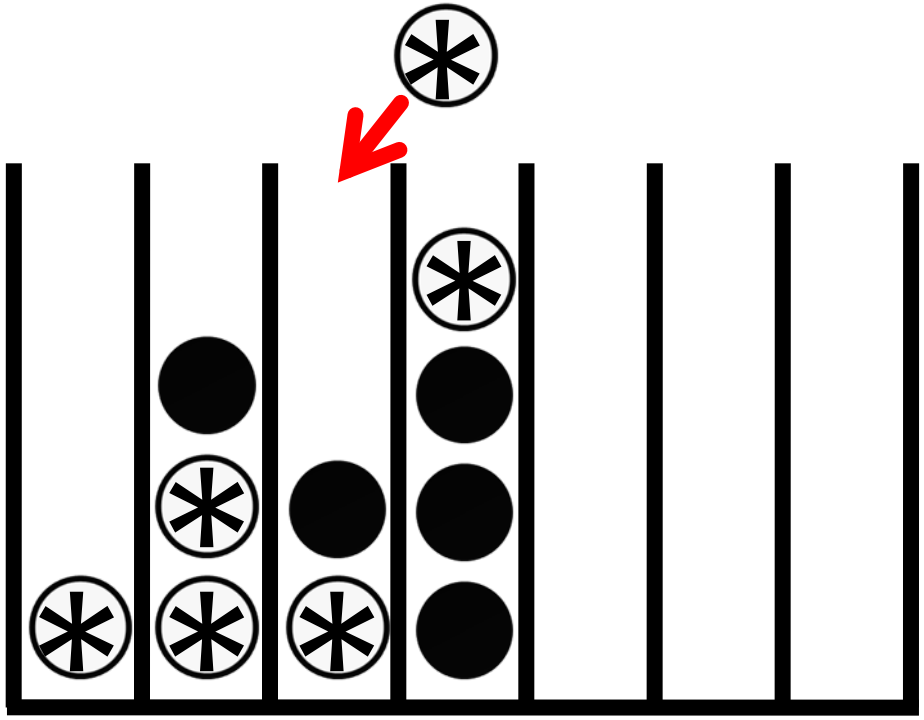
情報処理概論

第１２回　演習１

情報基盤研究開発センター　谷本 輝夫

演習

- 次ページの立体四目並べの主プログラムから
  呼び出す、入力サブルーチン drop を作成する



立体四目並べの主プログラム

```
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) ! Initial status

  do step = 1, m*n/2
    do side = 1, 2
      ! Drop ball to the specified column
      call drop(m, n, board, side)
      call show(m, n, board)
    end do
  end do
stop
end program
```

show サブルーチンの例

```
subroutine show(m, n, board)
implicit none
integer, intent(IN) :: m, n
character(len=1), dimension(m, n), intent(IN) :: board
integer :: x, y

  write(*, '(1x)', advance='NO')
  do x = 1, m
    write(*, '(1x,i2)', advance='NO') x
  end do
  write(*, *)

  do y = n, 1, -1
    write(*, '(1x)', advance='NO')
    do x = 1, m
      write(*, '(a1,a1,a1)', advance='NO') '[',board(x, y),
']'
    end do
    write(*, *)
  end do

end subroutine
```

先週、自分で作ったものを使う

dropサブルーチンの仕様

- 引数：　４個
  - 盤の幅（＝列数）　… 整数
  - 盤の高さ（＝行数）　… 整数
  - 盤の配列　… 文字列（len=1)の２次元配列
  - 手番　… 整数（１：先攻 (*) 、 ２：後攻 (o)）
- 玉を落とす列の番号を入力させ、
  それに応じて盤の配列を更新する
- 間違った入力がされた場合、
  メッセージを表示して再度入力
  - 盤の範囲から外れた列番号
    - ただし、0 が入力されると終了することにする
  - もう玉を入れることのできない列の番号

表示例

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

どちらの手番かを表示する

エラーの表示例

```
  1  2  3  4  5  6  7
[ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ]
[*][ ][*][o][o][ ][ ]
* : Drop where? (0 = Exit)8
  Out of range!!
* : Drop where? (0 = Exit)
```

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

間違った場所に置こうとするとメッセージを表示して再入力

出来た人は

- 次回作成予定の、
  勝敗判定サブルーチンの設計を始める
- 縦、横、斜めのいずれかの方向で四目揃ったかどうかを判定
- 主プログラムや　dropサブルーチンは、
  必要に応じて適宜変更する

進め方がわからない人は

- まず、主プログラムを入力
  - 中身をじっくり読んで、理解する
- 次に show サブルーチンや drop サブルーチンの枠組みを入力
  - 枠組み： subroutine, end subroutineの行、および各引数の宣言
  - 例）showサブルーチンの枠組み

```
subroutine show(m, n, board)
implicit none
integer, intent(IN) :: m, n
character(len=1), dimension(m, n), intent(IN) :: board

end subroutine
```

ここまでで、とりあえずコンパイル、実行してみる

サブルーチンの中身の作成

- まず、サブルーチンでやるべきことを列挙し、
  分かる部分から段階的に実装して、
  それぞれが正しく動く事を確認していく。
  例えば…

  第１段階　手番の番号（１or２）に応じて記号（＊ or ｏ）
  を表示する

  第２段階　列番号の入力を促すメッセージ表示の write
  および入力のための read 文を追加

  第３段階　入力された列番号が
  0 なら終了
  盤の範囲外なら 'Out of range!!' の表示

  第４段階　正しい入力がされるまで繰り返す

  ・・・

センチネル（番兵）の利用

第４段階　正しい入力がされるまで繰り返す

- 繰り返す → ループ文
- 終了条件 → 正しい入力
- 正しい入力とは何だろうか？
  - 0ではない
  - 盤の範囲内（１～７）
  - その列が一杯ではない

→ 判定条件が沢山あって大変

- その他のループ制御
  - exit
  - cycle
  - stop

センチネルを利用

```
flag = 0
do while (flag == 0)
   :
   if ( . . .)
      flag = 1
   endif
   :
end do
```