Práctica 4
Funciones y repetición condicional

Introducción a la Programación
2do Semestre de 2016

Los ejercicios que corresponden a los **contenidos mínimos** recomendados se encuentran marcados con el simbolo ⊛.

1. Alternativa Condicional

Ejercicio 1

⊛ Evaluar las siguientes expresiones en los Tableros de la Figura **??**.

1. `not hayBolitas(Rojo)`
2. `puedeMover(Norte) && puedeMover(Este)`
3. `puedeMover(Norte) || puedeMover(Este)`
4. `not puedeMover(Norte) && puedeMover(Este)`
5. `nroBolitas(Negro)+nroBolitas(Azul)`
6. `nroBolitas(Negro)==nroBolitas(Azul) && nroBolitas(Negro)==nroBolitas(Verde)`
7. `puedeMover(opuesto(opuesto(Este)))`

Ejercicio 2

⊛ Escribir expresiones que denoten

1. `True` cuando la celda actual tiene más de 5 bolitas en total;
2. `True` cuando la celda actual tiene al menos 5 bolitas en total;
3. `True` cuando la celda actual tiene al menos 5 bolitas en total y el borde se encuentra justo al este de la misma;
4. `True` cuando la celda actual tiene una celda vecina al Norte o al Este;
5. `True` cuando la celda actual tiene bolitas de todos colores;
6. `True` cuando en la celda actual faltan bolitas de al menos un color (dar una solución sin usar la función del ítem anterior y otra usándola);
7. `True` cuando hay una celda al Este con bolitas negras o rojas (no hay precondición);

Ejercicio 3

⊛ Corresponder las siguientes oraciones que describen propósitos y precondiciones con sus respectivos procedimientos.

- No tiene precondición.
- Pone una bolita verde si la celda actual contiene bolitas rojas; caso contrario, pone una bolita negra.
- Saca una bolita de color `c1` si hay más que de color `c2`; caso contrario, saca una bolita de color `c2`.
- Si hay al menos dos bolitas de color `c`, entonces saca dos bolitas de color `c`; caso contrario, saca todas las bolitas de color `c` y pone una negra.
- Pone una bolita verde si la celda actual contiene bolitas rojas.
- Hay al menos una bolita de color `c1` o `c2`.
- No hace nada.

```
procedure A() {
    if (False) {
        Poner(Azul)
    }
}
```

```
procedure B() {
    if (hayBolitas(Rojo)) {
        Poner(Verde)
    }
}
```

```
procedure C() {
    if (hayBolitas(Rojo)) {
        Poner(Verde)
    }
    if (not hayBolitas(Rojo)) {
        Poner(Negro)
    }
}
```

```
procedure D() {
    if (hayBolitas(Rojo)) {
        Poner(Verde)
    } else {
        Poner(Negro)
    }
}
```

```
procedure E(c1, c2) {
    if (nroBolitas(c1) > nroBolitas(c2)) {
        Sacar(c1)
    } else {
        Sacar(c2)
    }
}
```

```
procedure Anidado() {
    if (hayBolitas(c)) {
        Sacar(c)
        if (hayBolitas(c)) {
            Sacar(c)
        } else {
            Poner(Negro)
        }
    } else {
        Poner(Negro)
    }
}
```

Ejercicio 4

⊛ Escribir un procedimiento `SacarSiHay(c)` que, dado un color `c`, saque una bolita de color `c` de la celda actual. Si no hubiera bolitas de color `c` el procedimiento no hace nada.

Ejercicio 5

⊛ Escribir el procedimiento `PonerCSiHayXeY(c, x, y)` que agregue una bolita de color `c` a la celda actual si hay una bolita de color `x` y otra de color `y` en la misma.

Ejercicio 6

Escribir un procedimiento `DesempatarDos(c1,c2)` que, dados dos colores `c1` y `c2`, ponga una bolita de color `c1` si la celda actual contiene la misma cantidad de bolitas de color `c1` y `c2`.

Ejercicio 7

Escribir el procedimiento `PonerCSiXNorteSur(c,x)` que agregue una bolita de color `c` en la celda actual, si en la celdas del Norte y del Sur hay bolitas de color `x`. Si no se cumple la condición dada, agregar una bolita negra en la celda actual. ¿Qué problema surge de la utilización de los condicionales anidados?

Ejercicio 8

⊛ Escribir un procedimiento `Desempatar` que ponga una bolita azul si la celda actual contiene la misma cantidad de bolitas de cada color. **Sugerencia:** evitar la anidación de alternativas condicionales; si no se le ocurre cómo, trate de escribir una expresión que indique que la celda actual tiene una bolita de cada color.

Ejercicio 9

Escribir un procedimiento `PonerOSacarAcorde(c, acorde)` que, dados dos colores `c` y `acorde`, se comporte de la siguiente forma: si hay una bolita de color `acorde`, entonces el procedimiento pone una bolita de color `c`; caso contrario, el procedimiento saca una bolita de color `c`. Discutir la precondición del procedimiento.

Ejercicio 10

⊛ Escribir un procedimiento `ParidadLindantes(test, pone)` que, dados dos colores `test` y `pone`, ponga una bolita de color `pone` en la celda actual si y sólo si cero, dos o cuatro de las celdas lindantes (al sur, norte, este y oeste) tienen bolitas de color `test`. El procedimiento debe funcionar **independientemente** de la posición en que se encuentre el cabezal. Puede suponer que no hay bolitas de color `pone` en la celda actual. **Sugerencia:** estructure bien su solución evitando la anidación de alternativas condicionales.

Ejercicio 11

⊛ Escribir el procedimiento `RotarColoresAdy` que modifique el contenido que tienen las celdas adyacentes a la celda actual como se describe a continuación. Las celdas adyacentes a una dada son las que se encuentran al Norte, Noreste, Este, Sureste, Sur, Suroeste, Oeste y Noroeste de la misma. Para este ejercicio vamos a suponer que las celdas a procesar tienen a lo sumo una bolita (o sea, o no hay bolita, o hay sólo una). El procedimiento debe cambiar el color de cada bolita según el siguiente esquema:

- si al comienzo hay una bolita roja, la celda procesada debe terminar con una bolita verde.
- si al comienzo hay una bolita verde, la celda procesada debe terminar con una bolita negra.
- si al comienzo hay una bolita negra, la celda procesada debe terminar con una bolita azul.
- si al comienzo hay una bolita azul, la celda procesada debe terminar con una bolita roja.

Ejercicio 12

Escribir un procedimiento `PonerSi` que dado un color `c` y un booleano `b`, ponga una bolita de color `c` en la celda actual si `b` es `True`, y no haga nada si este es `False`.

Ejercicio 13

Utilizando el ejercicio anterior, escriba los procedimientos `PonerCSiHayXeY` y `DesempatarDos`. ¿Que beneficios trae tener un procedimiento `PonerSi` contra utilizar `if` en cada caso?

Ejercicio 14

Escriba los procedimientos `SacarSi(c, b)` y `MoverSi(d, b)` que actúan de forma similar a PonerSi.

2. Ejercicios combinados

Ejercicio 15

⊛ Considere el problema de sacar 8 bolitas de color Azul. La precondición, como es de esperar, es que haya al menos 8 bolitas azules en la celda actual. Sólo uno de los siguientes procedimientos resuelve correctamente dicho problema. ¿Cuál es? **Justifique**.

```
procedure DibujarLineaA() {
    repeat 8 {
        Sacar(Azul)
    }
}
```

```
procedure DibujarLineaB() {
    repeat 8 {
        SacarSiHay(Azul)
    }
}
```

Ejercicio 16

⊛ Escribir un procedimiento `AlMenosUnaDeCada` que deje al menos una bolita de cada color en la celda actual. Es decir, para cada color c, `AlMenosUnaDeCada` pone una bolita de color c cuando no haya bolitas de color c. En caso de haber anidado una alternativa condicional dentro de la repetición indexada, modifique su código para evitar esto usando procedimientos.

Ejercicio 17

Escribir un procedimiento `DejarColorMinimo()` que, para cada color c, saque todas las bolitas de color c si la celda actual tiene bolitas de un color más chico que c. En otras palabras, 1. si la celda actual tiene bolitas azules, entonces se deben sacar todas las bolitas negras, rojas y verdes, 2. si la celda actual no tiene bolitas azules pero tiene negras, entonces hay que sacar todas las bolitas rojas y verdes y, por último, 3. si la celda actual no tiene bolitas azules ni negras pero sí tiene bolitas rojas, entonces hay que sacar todas las bolitas verdes. ¿El código propuesto funcionaría si a Gobstones se le agregaran colores? En caso negativo, modifique su código para que no dependa de que Gobstones tenga sólo 4 colores.

Ejercicio 18

Escribir un procedimiento `DegradarCuadrado(n,deg,c)` que, dados dos números `n`, `deg` y un color `c`, haga lo siguiente para cada celda del cuadrado de lado `n` cuyo vértice inferior izquierdo es la celda actual (ver Figura 1). Si hay más de `deg` bolitas de color `c` en una celda, entonces el procedimiento saca `deg` bolitas de la celda; caso contrario, saca todas las bolitas de color `c`. El cabezal debe quedar en la celda inicial. **Sugerencia:** dividir el problema en subproblemas. En particular, se sugiere tener un procedimiento que degrade una celda y otro que degrade una línea horizontal.

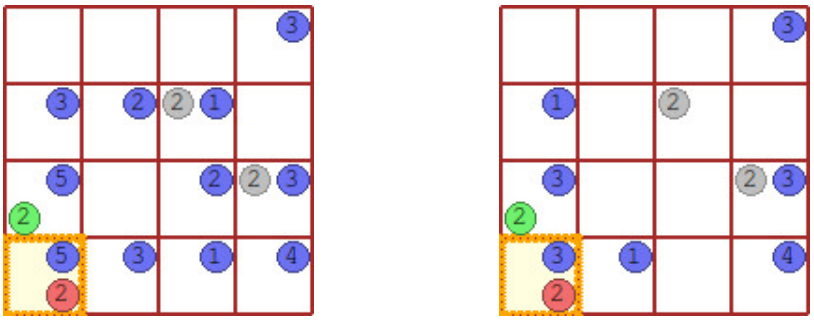


Figura 1: A la derecha se muestra el tablero correspondiente a ejecutar DegradarCuadrado(3,2,Azul) cuando el tablero inicial es el de la izquierda.

3. Funciones

Ejercicio 19

⊛ Corresponder las siguientes oraciones que describen propósitos y precondiciones con sus respectivas funciones.

- Denota la cantidad de bolitas de la celda en dirección `d`.
- El cabezal se puede mover en dirección `d`.
- Indica si la celda actual tiene más bolitas que todas las celdas lindantes.
- Denota true si la celda actual no tiene bolitas.
- El cabezal no se encuentra en un extremo del tablero.
- No tiene precondición.
- Denota la cantidad de bolitas de la celda actual.

```
function a() {
    return(nroBolitas(Azul) + nroBolitas(Rojo) +
           nroBolitas(Verde) + nroBolitas(Negro))
}
function b(d) {
    Mover(d)
    return(a())
}
function c() {
    return(a() == 0)
}
function d() {
    return(a() > b(Norte) && a() > b(Este) && a() > b(Sur) && a() > b(Oeste))
}
```

¿Qué efectos se producen en el tablero y/o el cabezal cuando se invoca `b(Norte)`?

Ejercicio 20

Para cada una de las siguientes oraciones, determine si la misma describe el propósito de un procedimiento, el propósito de una función, o no describe ningún propósito. **Justifique**.

- Demora menos de cuatro segundos.

- Deben haber al menos 7 bolitas azules en la celda al este.
- Indica si la celda al este tiene al menos 7 bolitas azules.
- Pone 7 bolitas azules en la celda al este.
- Pone 7 bolitas azules en la celda al este, e indica si la celda al este queda con mas bolitas que la celda actual.
- Pone 7 bolitas azules en la celda actual en menos de cuatro segundos.
- Denota la cantidad de bolitas de un color `c`, recibido como parámetro, que tiene la celda al origen.
- Pone tantas bolitas de color `c` en la celda actual como aquellas que haya en la celda del origen.
- Debe haber tantas bolitas de color `c` en la celda actual como aquellas que haya en la celda del origen.
- Denota la cantidad de bolitas totales que aparecen en el tablero.

Ejercicio 21

⊛ Indique todos los efectos que tiene el siguiente programa.

```
function sinEfectoMariposa() {
    Poner(Azul); Poner(Azul);
    Poner(Azul); Poner(Verde)
    return(Rojo)
}
```

```
program() {
    Poner(sinEfectoMariposa())
}
```

Ejecute el programa en PyGobstones para verificar su respuesta.

Ejercicio 22

⊛ Complete la siguiente tabla poniendo SI o NO en cada celda

	Comando Básico	Operación Básica	Procedimiento	Función
La invocación puede modificar el tablero				
La invocación es una expresión				
Puede invocarse como argumento				
Puede invocarse sin ser argumento				
Puede tener parámetros				
Puede invocar comandos	n/a	n/a		
Puede invocar opearciones	n/a	n/a		
Puede invocar procedimientos	n/a	n/a		
Puede invocar funciones	n/a	n/a		
Puede tener alternativas condicionales	n/a	n/a		
Puede tener repeticiones indexadas	n/a	n/a		
Tienen propósito y precondición				

Por básica nos referimos a los comandos y las operaciones que vienen en Gobstones.

En vista de esta tabla, ¿podemos afirmar que los procedimientos son comandos definidos por el usuario? ¿Y que las funciones son operaciones definidas por el usuario?

Ejercicio 23

⊛¿Cuál es la precondición de las siguientes funciones? Ejecute las mismas en PyGobstones y compare sus resultados

```
function hayBolitasAl(c, d) {
//Determina si hay bolitas de
//color c en dirección d
    Mover(d)
    return(hayBolitas(c))
}
```

```
function lindanteConColor(c, d) {
//Determina si hay una celda lindante en
//dirección d con bolitas de color c
    return(puedeMover(d) && hayBolitasAl(c,d))
}
```

Explique qué es el *cortocircuito* y cómo funciona dentro de la funcion `lindanteConColor`.

Ejercicio 24

⊛ Escribir las siguientes funciones. Cada función debe hacer uso de las funciones previas.

1. `vacia`, que determina si la celda actual no tiene bolitas de ningún color.
2. `conBolitas`, que determina si la celda actual tiene bolitas (no importa el color).
3. `conBolitasAl(d)`, que determina si la celda en dirección `d` tiene bolitas (no importa el color); esta función hace BOOM si el cabezal no se puede mover en dirección `d`.
4. `lindanteConBolitas(d)`, que determina si hay una celda en dirección `d` que tenga bolitas (no importa el color); notar que esta función es total y denota `False` cuando no se puede mover en dirección `d`.
5. `lindantesConBolitas()`, que indica si en todas las direcciones hay celdas lindantes con bolitas (no importa el color).
6. `enCruz()`, que determina si la celda actual tiene forma de cruz, i.e., determina si la celda actual tiene bolitas y a la vez en todas las direcciones hay celdas lindantes con bolitas (no importa el color).
7. `enCruzAl(d)`, que determina si la celda en dirección `d` tiene forma de cruz; esta función hace BOOM si el cabezal no se puede mover en dirección `d`.
8. `enCruzDiag(d)`, que determina si la celda en la diagonal `d` + `siguiente(d)` tiene forma de cruz; está funcion hace BOOM si el cabezal no se puede mover en dirección `d` o en dirección `siguiente(d)`.
9. `lindanteEnCruz(d)`, que indica si hay una celda en dirección `d` que tenga forma de cruz; esta función es total y denota `False` cuando no se puede mover en dirección `d`.
10. `diagonalEnCruz(d)`, que indica si hay una celda en diagonal `d` + `siguiente(d)` que tenga forma de cruz; esta función es total y denota `False` cuando no se puede mover en dirección `d` o en dirección `siguiente(d)`.
11. `explosion()`, que indica si en TODAS las direcciones N, NE, E, SE, S, SO, O, NO, hay celdas con forma de cruz. En caso contrario devuelve `False`.

Para reflexionar. ¿Le gustaría poder utilizar una repetición indexada para resolver la función `explosion`?

Ejercicio 25

⊛ Reescribir el procedimiento `PonerCSiXNorteSur(c,x)` de la Práctica anterior (parte alternativa condicional) utilizando la función `hayBolitasAl` del Ejercicio 23. **Nota:** esta prohibido anidar alternativas condicionales para resolver este ejercicio. ¿Qué ventajas tiene el uso de expresiones? **Recordar:** `PonerCSiXNorteSur(c,x)` agregua una bolita de color `c` en la celda actual, si en la celdas del Norte y del Sur hay bolitas de color `x`. Si no se cumple la condición dada, agrega una bolita negra en la celda actual.

Ejercicio 26

⊛ Escribir la función `puedeMoverN(n,d)` que determine si el cabezal puede moverse `n` posiciones en dirección `d`. **Ayuda.** Primero use una alternativa condicional para escribir un procedimiento **total** `MoverSiPuede(d)` que mueva el cabezal en dirección `d` de ser posible. Luego, aproveche la repetición indexada para escribir un procedimiento **total** `MoverALoSumo(n,d)` que mueva el cabezal `n` posiciones en dirección `d` de ser posible. En el caso en que no haya tantas celdas en dirección `d`, el cabezal debe quedar en la última celda. Por último, observe que `puedeMoverN(n,d)` denota `True` cuando `puedeMover(d)` denota `True` luego de `MoverALoSumo(n-1,d)`.

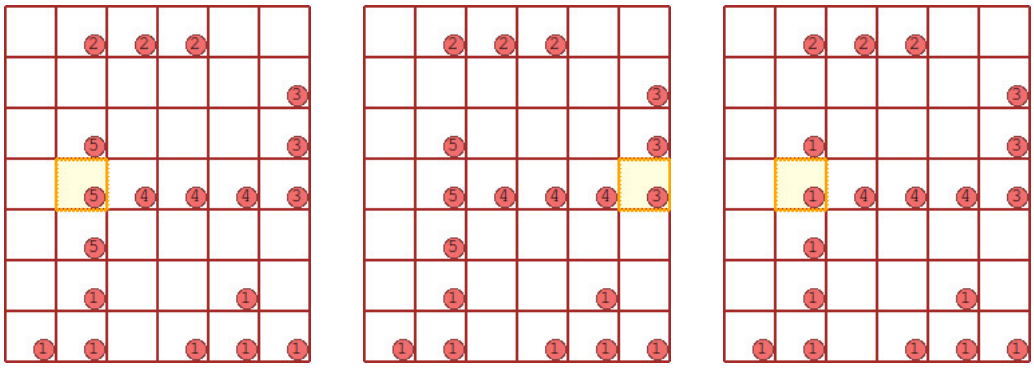
Ejercicio 27

⊛ Supongamos que estamos programando una variante del juego de tetris[1] en GOBSTONES donde las únicas piezas posibles son los bastones de longitud 3 (ver Figura 2 (a)). Cada celda que compone un pieza se llama una *sección*. Para representar las secciones, vamos a usar bolitas rojas. Aquellas secciones que correspondan a piezas que ya tocaron el piso se representan usando una bolita roja. Las secciones de estas piezas se consideran parte del piso. En cambio, las secciones de las piezas que aún están flotando se identifican con al menos dos bolitas rojas. Las tres secciones de una pieza flotante tienen la misma cantidad de bolitas rojas; dicha cantidad es el *número* de la pieza. No puede haber dos piezas flotando con el mismo número. De esta forma, es fácil distinguir dónde termina una pieza y empieza otro. Programe las siguientes funciones y procedimientos en GOBSTONES.

1. `colorPieza()`, que denota Rojo.
2. `numeroPiso()`, que denota 1.
3. `numeroSeccion()` que, suponiendo que el cabezal se encuentra sobre una sección, retorne el número de dicha sección (i.e., la cantidad de bolitas de color `colorPieza()`). Ejemplo: `numeroSeccion()` evaluado en la Figura 2 (a) denota 5, mientras que en la Figura 2 (c) denota 1.
4. `estaFlotando()` que, suponiendo que el cabezal se encuentra sobre una sección, denote `True` si la sección corresponde a una pieza flotante (i.e., denota si el numero de sección es mayor a 1). Ejemplo: `estaFlotando()` evaluado en la Figura 2 (a) denota `True`, mientras que en la Figura 2 (c) denota `False`.
5. `haySeccionAl(d)` que indica si hay una sección en la celda lindante en dirección `d`. Esta función debe ser total; si el cabezal no puede moverse, la función retorna `False`. Ejemplo: en la Figura 2 (b) `haySeccionAl(Norte)` denota `True`, mientras que `haySeccionAl (Este)` y `haySeccionAl(Sur)` denotan `False`.

[1] `http://es.wikipedia.org/wiki/Tetris`

6. `numeroSeccionAl(d)` que indica el número de la sección en la celda lindante en dirección `d`. Puede suponer que `haySeccionAl(d)` denota `True`. Ejemplo: `numeroSeccionAl (Norte)` evaluado en la Figura 2 (a) denota 5, mientras que en la Figura 2 (b) denota 3.
7. `continuaPiezaHacia(d)` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, retorna `True` si la pieza continua hacia la dirección `d`. Notar que esta función es total. Ejemplo: `continuaPiezaHacia(Sur)` evaluado en la Figura 2 (a) denota `True`, mientras que en la Figura 2 (b) denota `False`.
8. `centradoEnPieza()` que denote `True` cuando el cabezal se encuentra en la celda central de un pieza. Ejemplo: `centradoEnPieza()` evaluado en la Figura 2 (a) denota `True`, mientras que en la Figura 2 (b) denota `False`.
9. `CentrarEnPieza()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, mueva el cabezal para centrarlo en dicha pieza. Notar que `CentrarPieza` no tiene efectos si el cabezal ya está centrado. Ayuda: para los ejercicios que siguen, conviene utilizar `CentrarPieza` para reducir la cantidad de casos a considerar.
10. `esHorizontal()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, denote `True` si la pieza está en posición horizontal. Ejemplo: `esHorizontal()` evaluado en la Figura 2 (a) denota `False`; si el cabezal se encontrara sobre la pieza 4 denotaría `True`.
11. `BorrarPieza()` que, suponiendo que el cabezal se encuentra en una pieza flotante, elimine dicha pieza del tablero. El cabezal debe quedar en la posición central de la pieza.
12. `puedeBajarPieza()` que, suponiendo que el cabezal se encuentra en una pieza flotante, denote `True` si la pieza puede descender una posición manteniendo su orientación. Para ello, no pueden haber secciones de otras piezas en las celdas lindantes al sur de las secciones de la pieza bajo el cabezal (como ocurre en el tetris). Ejemplo: en la Figura 2 (a) `puedeBajarPieza()` denota `False`, mientras que en la Figura 2 (b) denota `True`.
13. `DibujarPiezaHorizontal(n)` que dibuje la pieza horizontal número `n` centrada en la celda actual. Puede suponer que las celdas lindantes al este y al oeste están vacías.
14. `DibujarPiezaVertical(n)` que dibuje la pieza vertical número `n` centrada en la celda actual. Puede suponer que las celdas lindantes al norte y al sur están vacías.
15. `BajarPieza()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, lo baje de fila en caso de ser posible. Si la pieza no puede bajar, entonces no debe modificarse la posición del mismo.
16. `RotarPieza()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, lo rote de acuerdo a su celda central. Si la pieza no puede entrar rotada (porque hay otras secciones), entonces el procedimiento no tiene efectos.
17. `tocaPiso()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante, denote `True` si alguna de sus secciones a) es lindante hacia el sur con alguna sección del piso, o b) se encuentra en el extremo sur del tablero. Ejemplo: `tocaPiso()` evaluado en la Figura 2 (a) denota `True`, mientras que en la Figura 2 (b) denota `False`.
18. `PasarAPiso()` que, suponiendo que el cabezal se encuentra sobre una pieza flotante que toca el piso, transforme dicha pieza en parte del piso, dejando exactamente una bolita



(a) (b) (c)

Figura 2:

roja por cada una de sus secciones. Ejemplo: la Figura 2 (c) es el resultado de invocar `PasarAPiso()` en la Figura 2 (a).

¿Se le ocurren más funciones y procedimientos útiles?

Ejercicio 28

⊛ Modifique el ejercicio anterior para que los bastones sean de color Azul. ¿Para qué sirve la función `colorPieza()`?

4. Repetición condicional

Ejercicio 29

⊛ Corresponder las siguientes oraciones que describen propósitos y precondiciones con sus respectivas funciones y procedimientos.

- Mueve el cabezal al extremo en dirección `d`.
- No tiene precondición.
- Determina si hay una celda sin bolitas en la fila actual.
- Posiciona el cabezal en alguna celda de la fila actual que no tenga bolitas.
- Hay alguna celda con bolitas en la fila actual
- Dibuja una diagonal de color `c` en el tablero.
- Hay mas filas que columnas en el tablero.
- Hay mas columnas que filas en el tablero.

```
procedure A(d) {
    while(puedeMover(d)) {
        Mover(d)
    }
}
```

```
function b() {
    A(Oeste)
    while(puedeMover(Este) && conBolitas()) {
        Mover(Este)
    }
    return(conBolitas())
}
```

```
procedure C() {
    A(Oeste)
    while(conBolitas()) {
        Mover(Este)
    }
}
```

```
procedure D(c) {
    IrAlOrigen()
    while(puedeMover(Este)) {
        Poner(c)
        Mover(Este)
        Mover(Norte)
    }
    Poner(c)
}
```

```
procedure E(c) {
    IrAlOrigen()
    while(puedeMover(Norte)) {
        Poner(c)
        Mover(Este)
        Mover(Norte)
    }
    Poner(c)
}
```

```
procedure F(c) {
    IrAlOrigen()
    while(puedeMover(Este) && puedeMover(Norte)) {
        Poner(c)
        Mover(Este)
        Mover(Norte)
    }
    Poner(c)
}
```

Ejercicio 30

⊛ Determine las precondiciones de los siguientes procedimientos.

```
procedure MayorarHasta(c1, c2) {
//Pone bolitas de color c1 para igualar la cantidad de bolitas de color c2
    while(nroBolitas(c1) != nroBolitas(c2)) {
        Poner(c1)
    }
}
```

```
procedure RobinHoodear(c1, c2) {
//Saca del que mas color hay para poner del que menos color hay hasta igualar
    while(nroBolitas(c1) != nroBolitas(c2)) {
        if(nroBolitas(c1) >nroBolitas(c2)) {
            Sacar(c1); Poner(c2);
        } else {
            Sacar(c2); Poner(c1);
        }
    }
}
```

```
procedure PingPonguear(c1, c2) {
//Mueve el cabezal en forma de ping pong de acuerdo a las bolitas de color c1 y c2
    while(hayBolitas(c1) || hayBolitas(c2)) {
        if(hayBolitas(c1)) {
            MoverN(nroBolitas(c1), Este)
        } else {
            MoverN(nroBolitas(c2), Oeste);
        }
    }
}
```

Ejercicio 31

Escriba un procedimiento `PonerVerdeAlNorteConRojas` que deposite una bolita verde en la primera celda que esté al Norte de la celda actual y que tenga bolitas rojas.

Ejercicio 32

⊛ Escriba el procedimiento `IrHastaColor(c, d)` que posicione el cabezal en la primer celda en dirección `d` que contenga bolitas de color `c`. Reescriba el procedimiento anterior reutilizando `IrHastaColor`.

Ejercicio 33

Escriba un procedimiento `PonerVerdesAlNorte` que coloque una bolita verde en todas las celdas al Norte de la celda actual.

Ejercicio 34

⊛ Escriba un procedimiento `PintarConColorHacia(c, d)` que coloque una bolita de color `c` en todas las celdas que aparecen en dirección `d` desde la celda actual. Reescriba el procedimiento anterior usando `PintarConColorHacia`

Ejercicio 35

⊛ Escriba un procedimiento `IrAlExtremo(d)` que posicione el cabezal en la última celda en un recorrido en dirección `d`.

Ejercicio 36

⊛ Usando `IrAlExtremo`, escriba el procedimiento `IrAlOrigen()` que posicione el cabezal en el origen del tablero.

Ejercicio 37

⊛ Combine `IrAlExtremo` y `PintarConColorHacia` para obtener el procedimiento `PintarFila (c)` que coloque una bolita de color `c` en cada una de las celdas de la fila actual.

Ejercicio 38

⊛ Escribir una función `hayBolitasEnFila(c)` que indique si la fila actual tiene una bolita de color `c`.

Ejercicio 39

⊛ Escribir una función `hayExplosion` que indique si la fila actual tiene una celda con forma de explosión (ver Ejercicio 24).

Ejercicio 40

Indique cuál es el propósito del procedimiento `LimpiarTableroBis`, donde `LimpiarCelda` es el procedimiento definido en la Práctica 3.

```
procedure LimpiarTableroBis() {
//Proposito: a determinar.
//Precondición: no tiene.
    while(not esUltimaCelda()) {
        LimpiarCelda()
        IrASiguienteCelda()
    }
    LimpiarCelda()
}
```

```
procedure IrASiguienteCelda() {
//Va a la sig. celda en un recorrido del tablero
//``primero al Norte y luego al Este''.
//Precondicion: not esUltimaCelda()
    if(puedeMover(Norte) {
        Mover(Norte)
    } else {
        IrAlExtremo(Sur)
        Mover(Este)
    }
}
```

```
function esUltimaCelda() {
//Indica si la celda actual es la ultima en un recorrido del tablero
//que primero va hacia el norte y luego hacia el este.
//Precondición: no tiene.
    return(not puedeMover(Norte) && not puedeMover(Este))
}
```

Para pensar: ¿Qué habría que modificar para sacar una bolita azul de cada celda del tablero? ¿Y si quisiéramos poner una bolita de cada color en cada celda del tablero?