Práctica 3
Repetición indexada y alternativa condicional

Introducción a la Programación
1er Semestre de 2015

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

1. Valores, expresiones y tipos

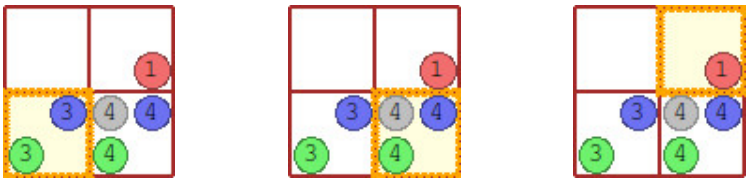
Ejercicio 1

⊛ Indicar cuáles de las siguientes tiras de símbolos son expresiones de Gobstones (es decir, denotan un valor). Además, para aquellas que sean expresiones, indicar si son literales o no y cuál es su tipo (i.e., si denota un número, una dirección, un color o un booleano). **Justifique**.

1. `7`
2. `1+6`
3. `+`
4. `-1+6`
5. `4 div 5`
6. `4 mod 5`
7. `Verde`
8. `Rojo+Azul`
9. `Siguiente(Rojo)`
10. `siguiente(Rojo)`
11. `siguiente(Este)`
12. `maxColor(Rojo)`
13. `siguiente(maxColor())`
14. `Mover(Este)`
15. `opuesto(previo(minDir()))`
16. `puedeMover(Este)`
17. `5<8`
18. `8<5`
19. `False`
20. `5 <= 8`
21. `5<8 || 5==8`
22. `5<8 && 5==8`
23. `x+5`, *con* `x` *parámetro*
24. `5<x`, *con* `x` *parámetro*
25. `x<y`, *con* `x` *e* `y` *parámetros*
26. `MoverN(2 * n)`, *con* `n` *parámetro*
27. `Poner(c)`, *con* `c` *parámetro*
28. `siguiente(c)`, *con* `c` *parámetro*
29. `hayBolitas(siguiente(c))`, *con* `c` *parámetro*
30. `nroBolitas(Azul)<0`
31. `not hayBolitas(Verde)`

Ejercicio 2

⊛ Agrupar las siguientes expresiones de forma tal que las expresiones que denotan el mismo



(a) (b) (c)

Figura 1: Tableros de ejemplo

valor pertenezcan al mismo grupo, suponiendo que se evalúan en el tablero de la Figura 1 (a). ¿Cómo habría que agruparlas si el tablero fuera el de la Figura 1 (b) y (c)?

```
minColor()              Azul              nroBolitas(Azul)      --3
hayBolitas(Azul)        Este              siguiente(minDir())   3
nroBolitas(Azul) == 0   puedeMover(Este)  nroBolitas(Verde)     Verde
```

Observar que expresiones diferentes pueden denotar el mismo valor, y que una expresión puede denotar distintos valores en distintos tableros.

Ejercicio 3

⊛ Evaluar las siguientes expresiones en los Tableros de la Figura 1.

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 4

⊛ Detectar el error que tiene el siguiente programa. Justificar.

```
procedure A(c){Poner(c)}
program{A(9)}
```

Discutir por qué es importante documentar la naturaleza (tipo) de un parámetro junto con el propósito y la precondición.

Ejercicio 5

⊛ Suponiendo que `c` es un color y `n` es un número, indicar cuáles expresiones denotan números, cuáles colores, cuáles direcciones y cuáles booleanos.

- `previo(siguiente(siguiente(previo(minColor()))))`
- `hayBolitas(siguiente(c))`
- `nroBolitas(c) div 7 + 13 <= 15`

- `2 * nroBolitas(siguiente(c)) >n || n == nroBolitas(Azul)`

Ejercicio 6

⊛ Escribir expresiones que denoten

1. la cantidad de bolitas total (de los 4 colores) en la celda actual;
2. el doble de la cantidad de bolitas rojas en la celda actual;
3. `True` cuando la celda actual tiene más de 5 bolitas en total;
4. `True` cuando la celda actual tiene al menos 5 bolitas en total;
5. `True` cuando la celda actual tiene al menos 5 bolitas en total y el borde se encuentra justo al este de la misma;
6. `True` cuando la celda actual tiene una celda vecina al Norte o al Este;
7. `True` cuando la celda actual tiene bolitas de todos colores;
8. `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);
9. `True` cuando hay una celda al Este con bolitas negras o rojas (no hay precondición);

Ejercicio 7

Indicar cuáles de las siguientes invocaciones están bien escritas en Gobstones, teniendo en cuenta que `Pepe(c,d,n)` es un procedimiento que recibe un color `c`, una dirección `d` y un número `n`. **Justificar.**

- `Pepe(Azul, Este, 5)`
- `Poner(Poner)`
- `Poner(Mover(Este), Azul)`
- `Poner(Poner(Azul))`
- `Pepe(Poner(Azul), Mover(Este), 4)`
- `Pepe(Azul, Este, 5 == 7)`
- `Pepe(minColor(), minDir(), 7 div 5)`
- `Pepe(siguiente(siguiente(minColor())), Este, nroBolitas(Azul))`

Ejercicio 8

⊛ Indicar cuáles de los siguientes procedimientos están bien escritos en Gobstones, donde `Quique(n, c)` es un procedimiento que recibe un numero `n` y un color `c`. **Justificar** todas sus afirmaciones.

```
procedure Pepe(n, c) {
    // n: número, c: color
    Poner(c); Mover(Este)
    Quique(n, c)
}
```

```
procedure Francisco(c) {
    // c: color
    Sacar(nroBolitas(c))
    nroBolitas(c) < 2
}
```

```
procedure Papa() {
    Poner(Azul)
    Quique(nroBolitas(Azul), minColor())
}
```

```
procedure Benedicto() {
    Poner(Poner(Poner(Azul)))
    nroBolitas(Azul) == 3
}
```

2. Repetición

Ejercicio 9

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

- Mueve el cabezal tantas posiciones al Este como indica el parámetro `n`
- Hay tantas celdas al este como bolitas rojas hay en la celda actual.
- Pone una bolita negra y una roja
- Si `n` > 0, entonces pone `n` bolitas de color `c`; caso contrario, saca `-n` bolitas de color `c`.
- Mueve el cabezal 100 posiciones al Este.
- No tiene precondición.
- Hay al menos `n` celdas al Este de la actual.
- Dibuja un cuadrado de color `c` de lado 2 hacia el Noreste.
- Hay al menos `-n` bolitas de color `c`.
- El cabezal no se encuentra en el extremo Este ni en el extremo Norte.
- Mueve el cabezal al este en tantas posiciones como bolitas rojas hay en la celda actual.
- Dibuja un cuadrado de lado 2 en las que cada celda tiene una bolita de cada color.
- Hay al menos 100 celdas al este de la actual.

```
procedure A() {
    repeat 100 {
        Mover(Este)
    }
}
```

```
procedure B(n) {
    repeat n {
        Mover(Este)
    }
}
```

```
procedure C() {
    repeat nroBolitas(Rojo) {
        Mover(Este)
    }
}
```

```
procedure D() {
    foreach c in [Negro..Rojo] {
        Poner(c)
    }
}
```

```
procedure E(c) {
    foreach d in [minDir()..maxDir()] {
        Poner(c); Mover(d)
    }
}
```

```
    procedure F(c, n) {
        repeat n {
            Poner(c)
        }

        repeat -n {
            Sacar(c)
        }
    }
```

```
procedure G(n, c) {
    foreach c in [minColor()..maxColor()] {
        E(c)
    }
}
```

Ejercicio 10

Escribir un procedimiento `PonerCincuenta` usando `repeat` que ponga 50 bolitas azules en la celda actual.

Ejercicio 11

Escribir el procedimiento `PonerTantasVerdesComoRojas` que deposita, en la celda actual, tantas bolitas verdes como bolitas rojas hay en la misma.

Ejercicio 12

Usando como base el procedimiento anterior, escribir un procedimiento `PonerTantasXComoY (x,y)` que, dados dos colores `x` e `y`, ponga en la celda actual tantas bolitas de color `x` como bolitas de color `y` haya en la celda actual. ¿Cómo se debería invocar `PonerTantasXComoY` para resolver el ejercicio anterior?

Ejercicio 13

⊛ Escribir un procedimiento `PonerN(n, c)` que tenga como parámetros una cantidad `n` y un color `c` cuyo propósito sea colocar `n` bolitas de color `c` en la celda actual. ¿Qué ocurre cuando se invoca `PonerN(0, Azul)`? ¿Y si invocamos `PonerN(-4, Verde)`? ¿Cuál sería, entonces, la precondición de `PonerN`? Resolver el ejercicio anterior utilizando `PonerN` en conjunto con la operación `numeroBolitas`. ¿Cuál es la ventaja de utilizar `PonerN`?

Ejercicio 14

⊛ Escribir el procedimiento `SacarN(n,c)` que, dado un número `n` y un color `c`, saque `n` bolitas de color `c` de la celda actual. Suponiendo que no hay bolitas en la celda actual ¿Qué ocurre cuando se invoca `SacarN(0, Azul)`? ¿Y si invocamos `SacarN(-4, Verde)`? ¿Cuál sería, entonces, la precondición de `SacarN`?

Ejercicio 15

⊛ Utilizando `SacarN`, escribir el procedimiento `SacarTodas(c)` que, dado un color `c`, saque **todas** las bolitas de color `c` de la celda actual. ¿Este procedimiento funciona siempre?

Ejercicio 16

⊛ Escribir un procedimiento `PonerEnLindantes(c)` usando `foreach`, `minDir` y `maxDir`, que reciba un color `c` y deposite una bolita de color `c` en cada una de las celdas lindantes (al N, E, S, O) de la celda actual. Se recomienda utilizar un procedimiento auxiliar que reciba un color `c` y una dirección `d` y ponga una bolita de color `c` en la celda lindante en dirección `d` sin mover el cabezal.

Ejercicio 17

⊛ Escribir un procedimiento `PonerTodosLosColores` que deposite una bolita de cada color en

la celda actual. Haga uso de `foreach`, `minColor` y `maxColor`. Este procedimiento es equivalente al procedimiento `PonerUnaDeCada` de la Práctica 2 ¿Qué procedimiento le parece mejor escrito? ¿Y cuál preferiría si GOBSTONES tuviera 32000 colores?

Ejercicio 18

⊛ Utilizando `SacarTodas`, escribir el procedimiento `LimpiarCelda` que saque todas las bolitas de la celda actual. ¿Este procedimiento funciona siempre? ¿Cómo se comporta su procedimiento si los diseñadores de GOBSTONES le agregan los colores `Amarillo`, `Celeste`, y `Violeta` al lenguaje (notar que en este caso, `minColor()` y `maxColor()` dejarían de denotar `Azul` y `Verde`)? En caso en que la celda no quede limpia, modifique su programa para que funcione **independientemente** de cuáles sean los colores disponibles en Gobstones.

Ejercicio 19

Escribir el procedimiento `EscaleraSeisVerdeAlEste` que ponga 1 bolita verde en la celda actual, 2 bolitas verdes en la celda lindante al este, 3 bolitas verdes en la celda que le sigue al este, y así siguiendo hasta poner 6 bolitas en la celda que se encuentra a 5 celdas de distancia de la celda actual.

Ejercicio 20

⊛ Usando como base el ejercicio anterior, escribir el procedimiento `Escalera(n,c,d)` que, dado un número `n`, un color `c` y una dirección `d`, recorra las primeras `n` celdas en dirección `d`, poniendo 1 bolita de color `c` en la celda actual, 2 bolitas de color `c` en la siguiente celda, 3 bolitas de color `c` en la siguiente, etc. Ejecutar `Escalera(3,Verde,Este)` en un tablero que tenga 3 columnas en el que el cabezal se encuentre en el extremo oeste. Si el procedimiento hace BOOM, modificarlo para que funcione en este caso.

Ejercicio 21

Escribir un procedimiento `CopiarCeldaAlNorte` que copie todas las bolitas de la celda actual a la celda lindante al Norte.

Ejercicio 22

⊛ Escribir un procedimiento `CopiarCeldaAl(d)` que reciba por parámetro una dirección `d` y copie las bolitas a la celda lindante en dirección `d`. Para ello, se recomienda escribir el procedimiento `PonerNAl(c,d, n)` que reciba un color `c`, una dirección `d` y un número `n` y ponga `n` bolitas de color `c` en la celda lindante en dirección `d`, dejando el cabezal en la celda inicial.

Ejercicio 23

⊛ Utilizando el procedimiento anterior escribir otro, llamado `CopiarCeldaALindantes`, que copie las bolitas de la celda actual a cada una de sus celdas lindantes (al Norte, Este, Sur y Oeste).

Ejercicio 24

⊛ Utilizando nuevamente el procedimiento del ejercicio 22, escribir un procedimiento que copie las bolitas de la celda actual a sus celdas lindantes incluyendo las diagonales (al Norte, Noreste, Este, Sureste, Sur, Suroeste, Oeste y Noroeste). El nuevo procedimiento debe llamarse `CopiarCeldaAAdyacentes`.

3. Alternativa Condicional

Ejercicio 25

⊛ 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 26

⊛ 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 27

⊛ 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 28

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 29

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 30

⊛ 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 31

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 32

⊛ 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. **Sugerencia:** estructure bien su solución evitando la anidación de alternativas condicionales.

Ejercicio 33

⊛ 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 34

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 35

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

Ejercicio 36

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

4. Ejercicios combinados

Ejercicio 37

⊛ 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 38

⊛ 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 39

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 40

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 2). 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.

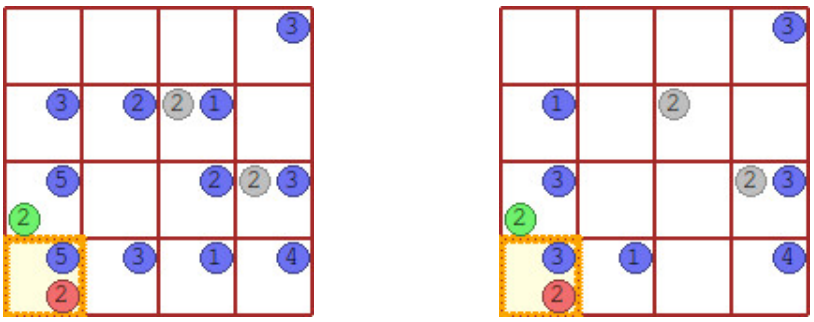


Figura 2: A la derecha se muestra el tablero correspondiente a ejecutar DegradarCuadrado(3,2,Azul) cuando el tablero inicial es el de la izquierda.