Práctica 5
Repetición condicional y Recorridos

Introducción a la Programación
2do Semestre de 2016

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

1. Repetición condicional

Ejercicio 1

⊛ 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)
    }
}

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

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

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 2

⊛ 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 3

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 4

⊛ 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 5

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

Ejercicio 6

⊛ 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 7

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

Ejercicio 8

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

Ejercicio 9

⊛ 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 10

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

Ejercicio 11

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

Ejercicio 12

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?

2. Repaso :Funciones y repetición condicional

Ejercicio 13

Escribir el procedimiento `MoverNRotativo(n,d)` que mueva el cabezal `n` posiciones en dirección `d` con la siguiente salvedad: en el momento en el que el cabezal no se pueda mover en dirección `d`, el mismo tiene que desplazarse hacia el extremo `opuesto(d)`. Estructure su solución utilizando un `repeat` que utilice el procedimiento `IrAlBorde`.

Ejercicio 14

Escribir el procedimiento `PonerSalteado(n, c, d)` que, dado un número `n`, un color `c` y una dirección `d`, ponga una bolita de color `c` en: la celda actual, la celda a distancia `n` en dirección `d`, la celda a distancia `2n` en dirección `d`, y así siguiendo mientras pueda. **Ayuda**: recuerde la función `puedeMoverN` de la práctica anterior. ¿Se le ocurre alguna forma de resolver el ejercicio actual sin utilizar `puedeMoverN`? ¿Qué problemas tendría su solución si los efectos de `puedeMoverN` no desaparecieran? Discutir las ventajas de que las funciones no produzcan efectos.

3. Recorridos estándar

Ejercicio 15

⊛ Escribir las siguientes funciones y procedimientos que sirven para recorrer el tablero. El recorrido se hace avanzando internamente en dirección `d1` y externamente en dirección `d2`. Por ejemplo, la Figura 1 muestra el recorrido cuando `d1` denota `Sur` y `d2` denota `Oeste`; el recorrido empieza en el extremo Noreste, y se va avanzando por columnas hacia el oeste, recorriendo cada columna de norte a sur.

1. `IrAlInicioT(d1, d2)`, que recibe dos direcciones `d1` y `d2` y mueve el cabezal al inicio del recorrido. En otras palabras, el cabezal se mueve al extremo `opuesto(d1) + opuesto(d2)`.
2. `puedeMoverT(d1, d2)` que denota `True` cuando el cabezal se puede avanzar a la siguiente celda. En otras palabras, denota `True` cuando el cabezal no se encuentra en el extremo `d1+d2`.
3. `MoverT(d1, d2)` que mueve el cabezal a la siguiente celda del recorrido. En otras palabras, el cabezal se ubica en la celda lindante en dirección `d1` de ser posible. Si no es posible, entonces el cabezal se mueve en dirección `d2` y hacia el extremo `opuesto(d1)`. ¿Cuál es la precondición de este procedimiento?

Ejercicio 16

⊛ Identifique el esquema de recorrido de las siguientes funciones y procedimientos, indicando qué elementos se recorren (e.g., las celdas del tablero, las celdas de una fila, las celdas de una

13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

Figura 1: Ejemplo del recorrido del tablero con direccion `d1` = Sur y `d2` = Oeste.

columna, las celdas que tienen tales bolitas, etc.), los procedimientos que se usan para ir a la primer celda, avanzar a la siguiente celda y procesar cada celda, y la función que se utiliza para determinar si hay más celdas por procesar dentro del recorrido. Escriba, asimismo, las precondiciones de las funciones y procedimientos.

```
procedure LimpiarTablero() {
    IrAlInicioT(Norte, Este)
    while(puedeMoverT(Norte, Este)) {
        LimpiarCelda()
        MoverT(Norte, Este)
    }
    LimpiarCelda()
}
```

```
procedure LimpiarFila() {
    IrAlExtremo(Oeste)
    while(puedeMover(Este)) {
        LimpiarCelda()
        Mover(Este)
    }
    LimpiarCelda()
}
```

```
procedure hayVaciaHacia(d) {
    while(puedeMover(d) && not vacia()) {
        Mover(d)
    }
    return(vacia())
}
```

```
procedure IrASiguienteColor(c) {
    while(not hayBolitas(c)) {
        MoverT(Norte, Este)
    }
}
```

```
procedure haySiguienteColor(c) {
    while(puedeMoverT(Norte, Este) &&
          not hayBolitas(c)) {
        MoverT(Norte, Este)
    }
    return(hayBolitas(c))
}
```

```
procedure IrAUltimaConColor(c) {
    while(haySiguienteColor(c)) {
        IrASiguienteColor(c)
    }
}
```

Ejercicio 17

⊛ Escribir un procedimiento `PonerEnVacias(c)` que ponga una bolita de color `c` en todas las celdas vacías del tablero. El procedimiento debe utilizar un esquema de recorrido que recorra todas las celdas del tablero.

Ejercicio 18

Escribir un procedimiento `DuplicarBolitas()` que duplique la cantidad de bolitas de cada celda del tablero. Es decir, si una celda contiene 8 bolitas azules, 4 rojas y una verde, entonces deberá contener 16 bolitas azules, 8 rojas y 2 verdes al finalizar el procedimiento. Se sugiere escribir un procedimiento que duplique la cantidad de bolitas de una celda.

Ejercicio 19

⊛ Escribir la función `hayCeldaCromatica()` que denote `True` si alguna celda del tablero contiene una bolita de cada color. Para ello, escribir una función `esCromatica` que indique si la celda actual tiene una bolita de cada color.

Ejercicio 20

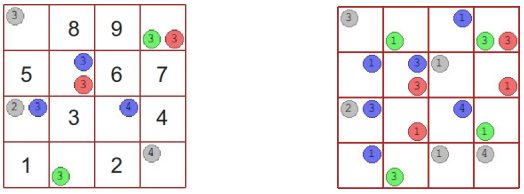
⊛ Escribir un procedimiento `DegradarTablero(c,deg)` que, dado un color `c` y un número `deg`, haga lo siguiente para cada celda del tablero. Si hay más de `deg` bolitas de color `c` en una celda, entonces el procedimiento saca `deg` bolitas de la celda; caso contrario, saca todas las bolitas de color `c`. **Ayuda:** este ejercicio es similar al procedimiento `DegradarCuadrado` de la Práctica 3 (parte: repetición indexada); trate de reutilizar procedimientos del mismo.

Ejercicio 21

⊛ Escribir el procedimiento `BuscarSiguienteVacia(d1,d2)` que posicione el cabezal en la siguiente celda vacía del tablero en un recorrido en direcciones `d1 + d2` (ver Ejercicio 15). Puede suponer que dicha celda siempre existe.

Ejercicio 22

⊛ Escribir el procedimiento `DistribuirVacias` que ponga una bolita en cada celda vacía del tablero, de forma tal que los colores de las bolitas nuevas alternen entre `Azul`, `Negro`, `Rojo` y `Verde` (en ese orden) en un recorrido del tablero en direccion `Este + Norte` (ver Figura 2). Recordar que no está permitida la anidación de repeticiones. **Ayuda:** escribir un procedimiento `PonerUnaDeCadaEnCeldasVacias` que recorra los colores y, por cada color `c`, busque una celda vacía y ponga una bolita de color `c`. Luego, repita este procedimiento mientras queden celdas vacías.



(a) (b)

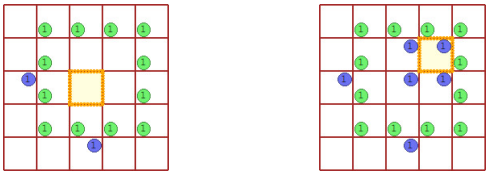
Figura 2: Ejemplo de la aplicación de `DistribuirVacias`. El Tablero (a) tiene 9 celdas vacías, que se recorrerían en el orden indicado. El Tablero (b) muestra el resultado esperado en este caso.

Ejercicio 23

⊛ Escribir un procedimiento `RecorrerComoReloj(c)` que realice la siguiente acción: comienza moviéndose hacia el `Norte` tantas celdas como bolitas de color `c` haya en la celda actual, una vez ahí repite la acción pero moviéndose el `Este` (tomando la cantidad de bolitas de color `c` de la nueva celda actual). Continuar moviéndose cambiando la dirección en el sentido de las agujas del reloj, hasta caer en una celda que no tenga bolitas de color `c`. `Sugerencia`: escribir un procedimiento que recorra una vez cada dirección, y luego ponga este procedimiento dentro de un recorrido. ¿Qué precondición tiene el procedimiento `RecorrerComoReloj(c)`?.

Ejercicio 24

Escribir un procedimiento `Rellenar(c, r)` que, suponiendo que la celda actual se encuentra dentro de un rectángulo vacío (de dimensión desconocida) demarcado por bolitas de color `c`, rellene su interior con bolitas de color `r` (ver Figura 3). Notar que no hay bolitas de color `c` dentro del rectángulo. `Sugerencia`: escriba los procedimientos `IrAInicioRect`, `MoverRect`, y la función `puedeMoverRect` que implementen un recorrido por el rectángulo.



(a) (b)

Figura 3: El Tablero (b) muestra el resultado de aplicar `Rellenar(Verde, Azul)` al Tablero (a).

Ejercicio 25

Suponga la siguiente codificación de movimientos usando colores:

1 azul	moverse en dirección Norte
2 azules	moverse en dirección Este
3 azules	moverse en dirección Sur
4 azules	moverse en dirección Oeste
verdes	distancia
0 azules	finalizar

1. Escribir un procedimiento `MoverPorAzul` que lea la codificación de la celda actual y pase a la celda siguiente según la misma. Por ejemplo, si la celda tiene una bolita azul y dos verdes, debe moverse dos celdas al Norte. Si no tiene bolitas azules, el cabezal no debe moverse. Puede suponer que ninguna celda tiene más de 4 bolitas azules. ¿Cuál es la precondición de este procedimiento?
2. Escribir un procedimiento `RecorridoPorAzul`, que recorra las celdas del tablero según la codificación, colocando una bolita negra en cada celda visitada. El recorrido termina cuando se llega a una celda con bolitas rojas. ¿Cuál es la precondición del procedimiento?
3. Escriba la función `recorridoAzulValido` que indique si el recorrido es posible y termina. Para ello, considere (a) escribir la función `puedeMoverAzul` que denote si el movimiento siguiente es valido, y (b) marcar cada celda recorrida agregando 5 bolitas azules para saber si el camino pasa dos veces por el mismo lugar (en ese caso, el camino no termina).