Práctica 6
Repetición condicional y recorridos

Introducción a la Programación
1er Semestre de 2017

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 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 4

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 5

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

Ejercicio 6

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

Ejercicio 7

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 8

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

Ejercicio 9

Escribir una función `hayExplosion` que indique si la fila actual tiene una celda con forma de explosión (ver Ejercicio 15.11 del TP5).

2. Recorridos

Ejercicio 10

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 11

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 columna, las celdas que tienen tales bolitas, etc.), los procedimientos que se usan para ir a la

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.

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 12

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 13

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 14

Escribir un procedimiento `DegradarTablero(c,deg)` que, dado un color `c` y un número `deg`,

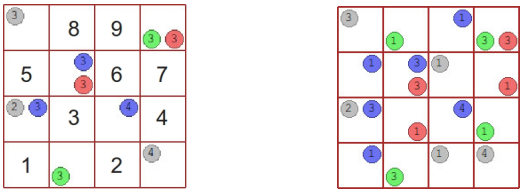
haga lo siguiente para cada celda del tablero. Si hay más de `deg` bolitas de color `c` en una celda, entonces el procedimiento saca `deg` bolitas de la celda; caso contrario, saca todas las bolitas de color `c`. **Ayuda:** este ejercicio es similar al procedimiento `DegradarCuadrado` de la Práctica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

Ejercicio 15

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 10). Puede suponer que dicha celda siempre existe.

Ejercicio 16

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.

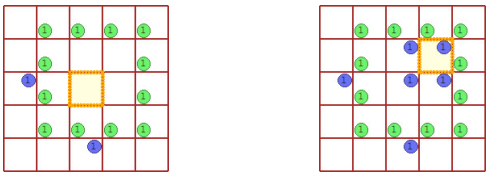


(a) (b)

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 17

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)`?.



(a) (b)

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