Gobstones

Introducción a la Programación - Práctica 7

Variables y funciones con procesamiento

CONSEJOS:

- Leer el enunciado en su totalidad y pensar en la forma de resolverlo **ANTES** de empezar a escribir código.
- Si un ejercicio no sale, se puede dejar para después y continuar con los ejercicios que siguen.
- Los ejercicios están pensados para ser hechos después de haber mirado la teórica correspondiente, y esta después de las actividades de indagación.
- Los ejercicios están tomados de las guías prácticas utilizadas en la materia de Introducción a la Programación de la Universidad Nacional de Quilmes por Pablo Ernesto “Fidel” Martínez López y su equipo, Introducción a la Programación de la Universidad Nacional de Hurlingham de Alan Rodas Bonjour y su equipo, de ejercicios y actividades realizadas por Federico Aloi y Miguel Miloro, a su vez basada en las guías Ejercicios de Introducción a la Programación del CIU General Belgrano, elaboradas por Carlos Lombardi y Alfredo Sanzo, y Fundamentos de la Programación del Proyecto Mumuki. Agradecemos a todos los que nos ayudaron con su inspiración.
- **Realizar EN PAPEL los ejercicios que así lo indiquen.**
- **Si un ejercicio indica BIBLIOTECA significa que será útil para la realización de futuros ejercicios tanto en esta guía como en las siguientes, pudiendo ser utilizado sin tener que volver a definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

EJERCICIOS:

1. Escribir la función **`hayBolitas_EnElBorde_(color, dirección)`**, que indica si en la celda que se encuentra en el borde dado por la dirección, hay bolitas del color indicado.

2. **BIBLIOTECA** Escribir la función **`tieneBolitas_Al_(color, dirección)`**, que, suponiendo que existe una celda lindante en la dirección dada, indica si la misma tiene o no bolitas del color indicado.

3. **BIBLIOTECA** Escribir la función **`hayBolitas_Al_(color, dirección)`**, que indica si hay una celda lindante en la dirección indicada y la misma tiene bolitas del color dado.

4. Escribir las funciones **`máximoEntre_Y_(valor1, valor2)`**, **`mínimoEntre_Y_(valor1,valor2)`** que describen el valor más grande o más chico, respectivamente, de los valores dados. ¿Qué herramienta hace falta? Si el código resultante utiliza una función con procesamiento, ofrecer otra versión que no utilice solamente funciones sin procesamiento.

5. Escribir la función **`nroBolitas_EnLaFilaActual(color)`** que describa la cantidad de bolitas del color dado en la fila actual.

    a. Escribir la solución con un recorrido de la fila actual que utilice una variable **`cantidadDeBolitasYaVistas`** cuyo propósito sea describir la cantidad de bolitas del color correspondiente que se contaron en cada momento. ¿Cuántas se vieron antes de empezar a contar? ¿Cómo estar seguro que se consideraron todas las celdas para contarlas?

    b. ¿En qué celda queda el cabezal al utilizar la función desde cualquier punto del programa? ¿Por qué?

6. Escribir las funciones **`nroFilas()`** y **`nroColumnas()`** que describan la cantidad de filas y columnas del tablero. ¿Se podría conocer la cantidad de filas o columnas del tablero sin que el cabezal se mueva *realmente* de la celda actual? ¿Qué habría que usar si hubiese que hacerlo exclusivamente con procedimientos?

7. Escribir la función **`distanciaAlBorde_(dirección)`**, que describe la cantidad de celdas que hay entre la celda actual y el borde indicado. **Observación:** si la celda actual se encuentra en el borde, la distancia es 0.

8. Escribir las funciones **`coordenadaX()`** y **`coordenadaY()`** que retornen la coordenada columna y la coordenada fila de la celda actual, respectivamente. Suponer que 0 es la coordenada de la primer fila y columna. ¿Es necesario escribir un recorrido para estas funciones, o puede reutilizarse alguna otra función ya hecha?

9. El bosque, parte 3

a. Escribir las funciones **`árbol()`**, **`semilla()`**, **`bomba()`**, **`nutriente()`** que describen las representaciones de los elementos del ejercicios “El bosque”, de las prácticas 4 y 5.

b. Escribir la función **`nroTotalDeÁrbolesEnElTerreno()`** que describa la cantidad de árboles que hay en el bosque. Organizar el código como un recorrido genérico sobre las parcelas instanciado para el sentido Sur-Oeste.

10. Escribir una función **`nroVacías()`** que describa la cantidad de celdas vacías del tablero. Estructurar el código como recorrido de celdas. Con respecto al código:recordar que no es buena práctica anidar herramientas en el mismo procedimiento o función. ¿Cómo hacer, entonces, para aumentar la cantidad de vacías en cada momento, dado que la variable solo puede modificarse de forma local?
    Ayuda: considerar una función que devuelva la *cantidad de celdas vacías* que hay considerando solamente la celda actual… (Observar que esa cantidad solamente puede ser 1 o 0.)

11. Escribir una función **`colorMínimoConBolitas()`** que denote el color más chico (en el orden Azul, Negro, Rojo, Verde) del cual hay bolitas en la celda actual.

    a. Considerar primero el contrato. ¿Qué ocurre si en la celda actual no hay bolitas de ningún color?

    b. Escribir la solución organizada como un recorrido de búsqueda sobre los colores que use una variable **`colorActual`** cuyo propósito sea denotar el color que se está recorriendo.

    c. ¿La solución funciona si se agregan más colores a Gobstones? En caso negativo, modificar la solución para que funcione con cualquier cantidad de colores.
    Ayuda: considerar utilizar las expresiones primitivas **`minColor()`** y **`maxColor()`** que describen al color más chico y al color más grande posibles.

12. Nova enfrentó su primer revisión de pares de su código, y no le fue muy bien. Los nombres de la mayoría de las funciones que escribió son letras sueltas, que no aportan legibilidad, ¡y encima no escribió los contratos de ninguna de las funciones! (El caradura puso los comentarios con puntitos. Se vé que vio en el código de algunos colegas algo parecido, y no se dio cuenta que en realidad el editor permite colapsar secciones y las muestra con puntitos. Por suerte indenta bien, porque si no, sería mucho más catastrófico…)
    El revisor estuvo escribiendo los propósitos para cada función, pero lo hizo en una serie de post-its (los cuadraditos de papel sueltos) y se mezclaron todas.

    a. Corresponder las siguientes propuestas de propósito hechas por el revisor, con las funciones correspondientes.
        A. Describir el color del cual hay más bolitas en la celda actual.
        B. Describir la cantidad de bolitas de la que hay más entre las bolitas de los colores dados.

C. Describir la cantidad de bolitas del color dado que tiene el tablero.
D. Describir la cantidad total de bolitas de todos los colores de la celda actual.
E. Describir el color de entre los dados que tiene más bolitas en la celda actual.
F. Describir cuál es la mayor cantidad de bolitas del color dado que tiene alguna celda del tablero.
G. Describir el mayor de dos valores dados.

```
function a(c1, c2){ /*..*/
  return(choose c2 when (nroBolitas(c2) > nroBolitas(c1))
                c1 otherwise)
}

function b(n1, n2) { /*..*/
  return(choose n1 when (n1 > n2)
                n2 otherwise)
}

function c() { /*..*/
  mayor := Azul
  colorActual:= siguiente(Azul)
  while (colorActual /= Azul) {
    mayor := a(mayor, colorActual)
    colorActual := siguiente(colorActual)
  }
  return(mayor)
}

function d() { /*..*/
  cant := 0
  colorActual := Azul
  while (colorActual /= Verde) {
    cant := cant + nroBolitas(colorActual)
    colorActual := siguiente(colorActual)
  }
  return(cant + nroBolitas(colorActual))
}

function e(c1, c2) { /*..*/
  return(b(nroBolitas(c1), nroBolitas(c2)))
}

function f(c1, c2) { /*..*/
  return(nroBolitas(a(c1, c2)))
}

function g(c) { /*..*/
  IrAlInicioDeUnRecorrido__(Norte, Este)
  cant := 0
  while(not estoyElFinalDeUnRecorrido__(Norte, Este)) {
```

```
    cant := cant + nroBolitas(c)
    PasarASiguienteCelda__(Norte, Este)
  }
  return(cant + nroBolitas(c))
}

function h(c) { /*..*/
  IrAlInicioDeUnRecorrido__(Norte, Este)
  mayor := nroBolitas(c)
  while(not estoyElFinalDeUnRecorrido__(Norte, Este)) {
    mayor := b(mayor, nroBolitas(c))
    PasarASiguienteCelda__(Norte, Este)
  }
  return(b(mayor, nroBolitas(c)))
}
```

b. Una vez hecha la correspondencia, renombrar las funciones, parámetros (y sus usos) de manera adecuada, para que el programa resulte legible.

13. La revisión de código sigue mal para Nova. Esta vez se trata de unos procedimientos que además de no tener contratos ni buenos nombres, algunos no andan y otros usan las variables de formas inadecuadas. Encontrar los errores en los procedimientos que escribió Nova. Justificar por qué son errores. Luego, renombrar los procedimientos, variables y parámetros (y sus usos) de manera adecuada, para que el programa resulte legible, y escribir los contratos.

```
procedure P(p)
  { p := p + 1; Poner__Veces(Azul, p) }

procedure Q(p) {
  if(p /= 3) {v := 3}
  Poner__Veces(Azul, v)
}

procedure R()
   { Poner__Veces(Azul, v) Mover__Veces(Este, v) }

procedure S(p) {
  v := Este
  Mover__Veces(v, 5)
  v := 5
  Mover__Veces(Este, v)
}
```

¡Alguien debe enseñarle pronto a Nova lo difícil que es entender y corregir el código si no se escriben buenos nombres y buenos contratos!

14. ¡Alguien tendría que sentarse con Nova y darle un par de lecciones! En su código aparecieron dos funciones muy mal indentadas, y donde algunos parámetros y variables tienen nombres que no ayudan a entender su propósito, o los argumentos

que eligió no están bien. ¡Y es tan difícil entender para qué puso una variable si no la nombra adecuadamente!

a. Asociar las oraciones A y B que describen propósitos de variables con sus respectivas variables.

   A. Denota la cantidad de bolitas del color color que ya se recorrieron.
   B. Denota la mayor cantidad de bolitas del color dado que tenía alguna de las celdas recorridas.

```
function total(x) {
// nro bolitas x en el tablero
IrAPrimeraCeldaEnUnRecorridoAl_Y_(Sur,
Oeste)
var := 0
while(haySiguienteCeldaEnUnRecorridoAl_Y_(Norte, Este))
{ var := var + nroBolitas(x)
IrASiguienteCeldaEnUnRecorridoAl_Y_( (Norte, Este) }
return(var + nroBolitas(x))
}

function bolitasDeColor(n) {
// nro máximo bolitas n en una celda
IrAPrimeraCeldaEnUnRecorridoAl_Y_(Sur, Oeste)
var := nroBolitas(n)
IrASiguienteCeldaEnUnRecorridoAl_Y_(Norte, Este)
while(haySiguienteCeldaEnUnRecorridoAl_Y_(Norte, Este))
{ var := máximoEntre_Y_(var, nroBolitas(n))
IrASiguienteCeldaEnUnRecorridoAl_Y_(Norte, Este)}
return(máximoEntre_Y_(var, nroBolitas(n)))
}
```

b. Una vez asociado, renombrar las variables y parámetros de forma adecuada, corregir los argumentos incorrectos y reescribir la indentación para que se entiendan de manera adecuada.

15. Escribir un procedimiento **`IrANésimaVacía_(n)`** que posicione el cabezal en la celda vacía número **`n`** que se encuentra en un recorrido del tablero por celdas en las direcciones Este y Norte. Si no hay suficientes celdas vacías, deja el cabezal en la esquina NorEste.
    Por ejemplo, **`IrANésimaVacía_(1)`** posiciona el cabezal en la primer celda vacía, **`IrANésimaVacía_(2)`** posiciona el cabezal en la segunda celda vacía, etc. Organizar la solución como un recorrido de búsqueda por celdas que utilice una variable **`celdasVacíasYaVistas`**, cuyo propósito sea denotar la cantidad de celdas vacías que ya se recorrieron.

16. La torre de control

    En este ejercicio las columnas del tablero representan canales aéreos por los que despegan y aterrizan aviones. Se considera que un canal está libre para aterrizar si

no hay avión en ninguna de las posiciones de ese el canal. Los aviones se representan con tantas bolitas azules como el número de su vuelo, y con una bolita Roja en la misma celda si está aterrizando o con una bolita Verde si está despegando. El tablero representa a todo el aeropuerto; en cada celda hay a lo sumo un avión. Implementar las siguientes operaciones:

a. La función **`cantidadDeCanalesLibres()`**, que devuelva la cantidad de canales sin aviones.
b. La función **`cantidadDeAvionesDespegando()`**, que devuelva la cantidad total de aviones que están despegando en todo el aeropuerto.
c. El procedimiento **`IrACanalLibreParaDespegar()`**, que deje el cabezal en el primer canal libre más cercano al Sureste del aeropuerto.
d. La función **`cantidadDeAvionesTotales()`**, que devuelva la cantidad total de aviones que están despegando/o aterrizando en todo el aeropuerto.
e. El procedimiento **`IrACanalLibreParaAterrizar()`**, que debe dejar al cabezal en el primer canal libre más cercano al Noroeste del aeropuerto.
f. La función **`cantidadDeCanalesConColisiónInminente()`**, que devuelva la cantidad de carriles con posibles colisiones en todo el aeropuerto. Se considera que hay una colisión posible si en el mismo carril hay un avión que está despegando debajo de un avión que está aterrizando.
g. El procedimiento **`IrACanalConColisiónInminente()`**, que deje el cabezal en el primer Carril con una colisión inminente (desde el extremo Suroeste del aeropuerto).
h. La función **`mayorNroDeVueloDespegando()`**, que retorne el número de vuelo más grande que esté despegando en el aeropuerto.

17. El casino, parte 2

En el marco del ejercicio del casino de la práctica 6, escribir las siguientes operaciones. En cada caso, recordar escribir primero el contrato, y expresar la estrategia usando subtareas (preferentemente con la metodología *top-down*).

a. **`estáElJugador_(nroJugador)`**, que denota verdadero si el jugador indicado tiene registrada, al menos, una apuesta.

b. **`alguienJugóA_MasDe_(nroApostado, cantidadMínima)`**, que denota verdadero si algún jugador apostó al número indicado por un monto mayor a **`cantidadMínima`**.