Gobstones

Introducción a la Programación - Práctica 9

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.**
- **Sí 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.**

Última actualización: 5 de Noviembre 2021

FUNCIONES CON PROCESAMIENTO:

1. Mirando la celda vecina

   Escribir la función **`hayBolitas_EnCeldaAl_`**, que, suponiendo que existe una celda lindante en la dirección dada, indica si la misma tiene o no bolitas del color indicado. Sí no hay una celda lindante, hace BOOM.

2. Mirando la celda vecina, incluso si no hay vecina

   BIBLIOTECA Escribir la función **`hayBolitas_Al_`**, que indica si hay una celda lindante en la dirección indicada y la misma tiene bolitas del color dado. Sí no hay celda lindante describe Falso.

3. Mirando en la celda al borde

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

4. Mirando en la fila o columna

   Escribir la función **`hayBolitas_Hacia_`** que indica si en alguna de las celdas hacia la dirección dada (sin incluir la celda actual) hay bolitas del color dado.

5. Y volviendo a mirar en la fila o columna

   Escribir la función **`hayCeldaVacíaHacia_`**, que indica si en alguna de las celdas hacia la dirección dada (sin incluir la celda actual) hay una que esté vacía.

6. Y si miramos el tablero

   Escribir la función **`hayAlgunaBolita`**, que indica si en alguna de las celdas del tablero existe una bolita del color dado.

7. Y volvemos a mirar el tablero

   Escribir la función **`hayAlgunaCeldaVacía`**, que indica si alguna de las celdas del tablero está vacía.

VARIABLES:

8. Copiamos una celda

   BIBLIOTECA Escribir el procedimiento **`CopiarCeldaAl_`**, que copia los contenidos de la celda actual a la celda lindante en la dirección dada. Note que la celda de destino debe quedar tal cual la celda actual, independientemente de los contenidos que tuviera la celda de destino previamente.

9. Copiamos las esquinas

   Escribir el procedimiento **`CopiarOrigenEnEsquinas`** que copia en cada esquina los contenidos que hay en la celda actual (las 4 esquinas deben terminar con exactamente las mismas bolitas de cada color que había en la celda donde estaba originalmente el cabezal en el tablero inicial. La posición final del cabezal no es relevante).

10. No era taaaaaan necesario...

Los ejercicios 8 y 9 de esta guía pueden resolverse sin variables. Vuelva a pensar cómo resolver los ejercicios anteriores sin el uso de las mismas.

11. ¡Y Dale!, ¡Y dale!, ¡Y dale Nova dale!!

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. Se pide entonces, encontrar los errores en los procedimientos que escribió Nova y 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!

ALTERNATIVA CONDICIONAL EN EXPRESIONES

12. El más chico

BIBLIOTECA Escribir la función **`mínimoEntre_Y_`**, que dados dos valores describe aquel que sea más chico. Por ejemplo, **`mínimoEntre_Y_(3, 7)`** describe **3**, mientras que **`mínimoEntre_Y_(9, 4)`** describe **4**.

a. ¿De qué tipo son los parámetros?

b. ¿Es válida la expresión **`mínimoEntre_Y_(Rojo, Azul)`**? ¿Qué describe?

c. ¿Qué se describe si son iguales? ¿Es relevante si es uno o el otro?

13. El más grande

BIBLIOTECA Escribir ahora la función **`máximoEntre_Y_`** que dados dos valores describe aquel que sea el más grande.

14. Mi caminante se mueve

La primitiva del ejercicio **"14. El Caminante"** de la **Práctica 8**, **`direcciónDelCódigo_(código)`**, puede implementarse con alternativa condicional de expresiones. Se pide que la implemente, y que pruebe ahora su código del caminante para verificar su correcto funcionamiento.

15. Piedra, Papel o Tijeras

EN PAPEL Escribir **`jugadaGanadoraDePiedraPapelOTijerasEntre_Y_`**, que dadas dos jugadas, describe la jugada ganadora entre ambas. Para olvidarnos de cómo está codificada la jugada, tenemos las funciones **`piedra()`**, **`papel()`** y **`tijeras()`**, que representan a cada una de las jugadas. En piedra papel o tijeras, el jugador puede elegir una de tres opciones, y cada opción pierde contra alguna otra y le gana a alguna otra.

Jugada	Pierde contra	Gana contra
piedra()	papel()	tijeras()
papel()	tijeras()	piedra()
tijeras()	piedra()	papel()

16. Piedra, Papel o Tijeras... Lagarto, Spock

EN PAPEL La popular variante del juego piedra, papel o tijeras, lagarto, spock, popularizada por Sheldon Cooper, es un juego en esencia idéntico al clásico, pero con mayor número de resultados posibles. El jugador puede elegir entre 5 posibles jugadas, y cada una pierde y/o gana ante dos jugadas, según se muestra en la siguiente tabla:

Jugada	Pierde contra	Gana contra
piedra()	papel(), spock()	tijeras(), lagarto()
papel()	tijeras(), lagarto()	piedra(), spock()
tijeras()	piedra(), spock()	papel(), lagarto()
lagarto()	tijeras(), piedra()	spock(), papel()
spock()	papel(), lagarto()	tijeras(), piedra()

Escribir **`jugadaGanadoraDePiedraPapelOTijerasLagartoSpockEntre_Y_`**, que dadas dos jugadas, describe la jugada ganadora entre ambas. Para olvidarnos de cómo está codificada la jugada, tenemos las funciones **`piedra()`**, **`papel()`**, **`tijeras()`**, **`lagarto()`** y **`spock()`**que representan a cada una de las jugadas.

VARIABLES Y ACUMULACIONES:

17. Contando bolitas

Escribir la función **`nroBolitas_EnLaFilaActual`** 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?

18. Contando celdas hacia un lado

BIBLIOTECA Escribir la función **`distanciaAlBorde_`**, 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.

19. Mis coordenadas son...

BIBLIOTECA Escribir las funciones **`coordenadaX`** y **`coordenadaY`** que retornen la coordenada de la columna y la coordenada de la fila de la celda actual, respectivamente. Suponer que 0 es la coordenada de la primera fila y columna.

a. ¿Es necesario escribir un recorrido para estas funciones, o puede reutilizarse alguna otra función ya hecha?

20. Contando filas y columnas

BIBLIOTECA Escribir las funciones **`nroFilas`** y **`nroColumnas`** que describan la cantidad de filas y columnas del tablero respectivamente.

a. ¿Se podría conocer la cantidad de filas o columnas del tablero sin que el cabezal se mueva *realmente* de la celda actual?

b. ¿Qué habría que usar si hubiese que hacerlo exclusivamente con procedimientos?

c. ¿Es necesario escribir un recorrido para estas funciones, o puede reutilizarse alguna otra función ya hecha?

21. Contando celdas vacías

BIBLIOTECA Escribir una función **`nroVacías`** que describa la cantidad de celdas vacías del tablero. Estructurar el código como recorrido por las celdas del tablero.

22. Contando celdas con bolitas.

BIBLIOTECA Escribir la función **`cantidadDeCeldasConBolitasDeColor_`** que describe la cantidad de celdas que contienen al menos una bolita del color dado.

23. Contando bolitas de un color

BIBLIOTECA Escribir una función **nroBolitasTotalDeColor_** que describa la cantidad de bolitas del color dado que hay en total en todo el tablero. Estructurar el código como recorrido por las celdas del tablero.

24. Y volvemos a mirar el tablero

EN PAPEL Dado que en el tablero está representada una carretera, y en cada celda puede haber hasta un auto, se pide que realice la función **`cantidadDeAutosEnLaCarretera`**. Para realizar esto se puede hacer uso de las siguientes:

```
function hayUnAuto()
  /*
    PROPÓSITO: Indica si hay un auto en la celda actual.
    TIPO: Booleano.
    PRECONDICIONES: Ninguna.
  */
```

25. El bosque, parte 5

a. Escribir la función **`cantidadTotalDeÁrbolesEnElTerreno`** 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.

b. Escribir **`cantidadTotalDeÁrbolesEnElTerrenoLuegoDeExplociones`**, una función que indica la cantidad total de árboles que quedarán en el terreno luego de explotar todas las bombas que hayan en este.

26. Contar se vuelve más fácil

BIBLIOTECA Una subtarea de mucha utilidad es aquella que describe 1 cuando se cumple una condición o cero en caso contrario. Se pide entonces escriba la función **`unoSi_ceroSiNo`** que realiza precisamente esto.

27. Y volviendo a contar

Reescriba sus recorridos de acumulación anteriores para utilizar la función **`unoSi_CeroSiNo`** en los casos en los que sea posible.

RECORRIDOS SOBRE ENUMERATIVOS:

28. Otra vez una de cada

Volver a escribir el procedimiento **`PonerUnaDeCadaColor`** que pone una bolita de cada color, estructurando la solución como un recorrido sobre colores.

29. Limpiando la cruz

Escribir el procedimiento **`LimpiarCruzDeColor_`** que dado un color limpia el dibujo de una cruz realizado con bolitas de dicho color, bajo la suposición de que el cabezal se encuentra en el centro de dicha cruz.

30. Hacia la cual hay bolitas

Escribir la función **`direcciónHaciaLaCualHayBolitasDe_`** que dado un color describe la dirección hacia la cual hay bolitas de dicho color, bajo la suposición de que existe una celda vecina con bolitas de dicho color en alguna de las dirección, y es única (no hay más de una vecina con bolitas de ese color).

31. Vecinas con bolitas

Escribir la función **`cantidadDeVecinasConBolitas`** que describe la cantidad de celdas vecinas que contienen bolitas (de cualquier color). En este caso el concepto de vecindad implica tanto las celdas ortogonales como las diagonales, es decir, las celdas hacia el N, E, S y O y también las diagonales hacia el NE, SE, SO y NO. La función realizada debe ser total.

32. Incrementando las cantidades

Escribir el procedimiento **`Poner_EnLineaHacia_De_IncrementandoDeA_ComenzandoEn_`** que dado un número que representa una cantidad de celdas a abarcar, una dirección hacia donde dibujar la línea, un color que indica en color de bolitas a poner, un número que indica el factor de incremento, y un número inicial, pone una línea de bolitas en donde, en la primer celda pone tantas bolitas del color dado como el número inicial, en la celda siguiente hacia la dirección dada, pone tantas bolitas como el número inicial sumado en el factor de incremento, en la dos lugares hacia la dirección tantas como el número inicial sumado en el doble del factor del incremento, y así siguiendo tantos lugares como el primer argumento. Ej. sí se invoca al procedimiento de la siguiente forma **`Poner_EnLineaHacia_De_IncrementandoDeA_ComenzandoEn_(5, Norte, Rojo, 3, 3)`** entonces se dibujará una línea de 5 celdas hacia el norte de bolitas de color rojo, comenzando en la celda actual, en donde se tendrán en cada celda (contando de la actual) las siguientes cantidades de bolitas: 3, 6, 9, 12, 15.

33. El número de Fibonacci

Escribir la función **`fibonacciNro_`**, que dado un número que representa una posición en la secuencia de fibonacci (debe ser mayor o igual a cero) describe el número de fibonacci correspondiente a dicha posición.

La sucesión de fibonacci es una sucesión infinita de número en donde se comienza con el número 1 como elemento en la primera y segunda posición de la sucesión, y luego, cada elemento de la sucesión se calcula como la suma de los dos elementos anteriores. A continuación se deja una pequeña tabla de la sucesión de fibonacci para los primeros números:

Posición	0	1	2	3	4	5	6	7	8	9	10
Elemento	1	1	2	3	5	8	13	21	34	55	89
Cálculo	-	-	1+1	2+1	3+2	5+3	8+5	13+8	21+13	34+21	55+34

RECORRIDOS SOBRE ENUMERATIVOS:

34. La celda con más

Escribir las funciones **coordenadaXConMásBolitas** y **coordenadaYConMásBolitas** que describen las coordenadas X e Y de aquella celda que tiene más bolitas (en total) que el resto. Se garantiza por precondición que hay alguna celda que tiene más bolitas que el resto.

35. El borde más cercano

Escribir la función **bordeMásCercano** que describe la dirección hacia la cual se encuentra el borde que está más cerca (a menor cantidad de celdas). Si hubiera dos bordes a la misma distancia describe la dirección más chica entre ellas.

36. Color más chico del cual hay bolitas

Escribir la función **colorMásChicoDelCualHayBolitas** que describe el color más chico para el cual haya bolitas en la celda actual. Por ej. sí en la celda hay bolitas Negras, Rojas y Verdes, el color más chico del cual hay bolitas es Negro. Sí solo hay bolitas de color Rojo y Verde, el más chico es Rojo.

a. ¿Qué pasa si no hay bolitas en la celda actual?
b. ¿Qué tipo de recorrido se está aplicando?

37. Color más grande del cual hay bolitas

Escribir la función **colorMásChicoDelCualHayBolitas** que describe el color más grande del cual hay bolitas. Por ej. sí en la celda hay bolitas Negras, Rojas y Verdes, el color más grande del cual ay bolitas es Verde. Sí solo hay bolitas de color Rojo y Negro, el más grande es Rojo.

EJERCICIOS INTEGRADORES:

38. Nova y sus variables misteriosas

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

```
function total(x) {
// nro bolitas x en el tablero
IrAlInicioDeUnRecorrido__(Sur,
Oeste)
var := 0
while(not estoyElFinalDeUnRecorrido__(Norte, Este))
{ var := var + nroBolitas(x)
PasarASiguienteCelda__( (Norte, Este) }
return(var + nroBolitas(x))
}
function bolitasDeColor(n) {
// nro máximo bolitas n en una celda
IrAlInicioDeUnRecorrido__(Sur, Oeste)
var := nroBolitas(n)
PasarASiguienteCelda__(Norte, Este)
while(not estoyElFinalDeUnRecorrido__(Norte, Este))
{ var := máximoEntre_Y_(var, nroBolitas(n))
PasarASiguienteCelda__(Norte, Este)}
return(máximoEntre_Y_(var, nroBolitas(n)))
}
```

a. Asociar las siguientes oraciones A y B que describen propósitos de variables con sus respectivas variables en el código de Nova.

A. Denota la cantidad de bolitas del color dado que ya se contaron.

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

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

39. Posicionandose en una celda vacía puntual

Escribir un procedimiento **`IrAVacíaNúmero_(númeroDeVacía)`** que posicione el cabezal en la celda vacía número **`númeroDeVacía`** 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, **`IrAVacíaNúmero_(1)`** posiciona el cabezal en la primer celda vacía, **`IrAVacíaNúmero_(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.

40. La torre de control

En este ejercicio las columnas del tablero representan cada una una pista de aterrizaje en la que despegan y aterrizan aviones. Se considera que una pista está libre para aterrizar si no hay avión en ninguna de las posiciones de esa pista. 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. Además sabemos que en cada celda hay a lo sumo un avión, y que en una misma pista pueden haber varios aviones.

a. La función **cantidadDePistasLibres**, que devuelva la cantidad de pistas de aterrizaje sin aviones en ella.

b. La función **cantidadDeAvionesDespegando**, que devuelva la cantidad total de aviones que están despegando en todo el aeropuerto.

c. El procedimiento **IrAPistaLibreParaDespegar**, que deje el cabezal en la primera pista libre más cerca al Este del aeropuerto, en la posición más al Sur de la pista.

d. La función **cantidadDeAvionesTotales**, que devuelva la cantidad total de aviones que están despegando o aterrizando en todo el aeropuerto.

e. El procedimiento **IrAPistaLibreParaAterrizar**, que debe dejar al cabezal en la primera pista libre más al Oeste del aeropuerto, en la celda más al Norte de la misma.

f. La función **cantidadDePistasConColisiónInminente**, que devuelva la cantidad de pistas con posibles colisiones en todo el aeropuerto. Se considera que hay una colisión posible si en la misma pista hay un avión que está despegando debajo (más al Sur) de un avión que está aterrizando.

g. El procedimiento **IrAPistaConColisiónInminente**, que deje el cabezal en la primera pista con una colisión inminente contando desde el Oeste, y en la celda más al Sur de la pista.