UNIVERSIDAD NACIONAL DE HURLINGHAM

Instituto de Tecnologia e Intenieria / Programacion Estructurada

ALTERNATIVA CONDICIONAL EN EXPRESIONES Y ESQUEMAS DE RECORRIDO

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.*
- *Algunos de 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. También 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 podrá ser utilizado en el parcial sin definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.*

ALTERNATIVA CONDICIONAL EN EXPRESIONES

11. 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.

- ¿De qué tipo son los parámetros?
- ¿Es válida la expresión mínimoEntre_Y_(Rojo, Azul)? ¿Qué describe?
- ¿Qué se describe si son iguales? ¿Es relevante si es uno o el otro?

12. El más grande

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

13. Mi caminante se mueve

La primitiva del ejercicio "El Caminante", práctica integradora, **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.

14. 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()

15. 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

16. 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?

17. 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.

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

19. 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?

20. 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.

21. Contando celdas con bolitas

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

22. 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.

23. 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.
*/
```

24. 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.

25. 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.

26. 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

27. 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.

28. 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.

29. 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 hay una única celda con bolitas de dicho color en la celda. La función realizada debe ser total.

30. 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.

31. 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.

Poner_EnLineaHacia_De_IncrementandoDeA_ComenzandoEn_(5, Norte, Rojo, 3, 3) 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.

32. 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 DE MÁXIMO Y MÍNIMO

33. 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.

34. 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). Sí hubiera dos bordes a la misma distancia describe la dirección más chica entre ellas.

35. 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?

36. Color más grande del cual hay bolitas

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