

EJERCICIOS CLIPS – RELACIÓN REGLAS

CLIPS

EJERCICIO 1 Y 2

1. En una granja hay animales de los siguientes tipos: perros, gatos, patos, vacas, ovejas y gallos. Escribe una serie de reglas de manera que en base a un hecho que indique el sonido que hace un animal identifique qué clase de animal es. Por ejemplo, si se afirma un hecho como (sonido kikiriki), el programa debe imprimir por pantalla el mensaje "Se trata de un gallo".
2. Repite el ejercicio anterior pero utilizando ahora la siguiente plantilla y una sola regla que resuelva el ejercicio:

```
(deftemplate animal
(slot nombre)
(slot sonido))
```

```
(defacts sonido-animal
(sonido-animal gato miau)
(sonido-animal perro guau)
(sonido-animal gallo kikiriki)
(sonido-animal pajarito pio pio)
(sonido-animal pato cuack)
(sonido-animal vaca mu)
(sonido-animal oveja bee))
```

```
(defrule busca-animal-sonido
(sonido-animal ?animal ?sonido)
(or (sonido ?sonido)(animal ?animal))
=>
(printout t "El "?animal " hace "?sonido crlf)
)
```

EJERCICIO 3

Haz un programa que dado un conjunto de hechos (datos <val1> val2>)(hechos de tipo datos con dos valores numéricos), detecte qué hechos cumplen que el segundo valor sea mayor que el primero y los imprima.

```
(defrule regla-ejercicio
(datos ?x ?y)
(test(< ?x ?y))
=>
```

```
(printout t "El dato " ?y " es mayor que " ?x crlf)
)
```

EJERCICIO 4

Haz un programa que dado un conjunto de hechos de tipo datos con un número indefinido de valores, detecte e imprima aquellos tal que el primer valor sea par y menor o igual al último.

```
(defrule datos
(datos ?nombre ?x $? ?y)
(test (evenp ?x))
(test (<= ?x ?y))
=>
(printout t "Los datos " ?nombre " cumplen la condicion" crlf)
)
```

EJERCICIO5

Haz un programa que dado un único hecho datos con un número ndefinido de valores (ejemplo: (datos hola 1 3 nuevo 1 adios)), elimine todas las apariciones del valor 1.

```
(defrule borrar-1s
?h<-(datos $?valores1 ?x&:(integerp ?x)&:(= ?x 1) ?valores2)
=>
(retract ?h)
(assert(datos $?valores1 $?valores2))
```

```
(deffacts f-datos
(datos 3 asdfa 7 as 8 1 1 7)
(datos 1 1 1 1 1 5)
)
```

EJERCICIO 6

Haz un programa que detecte e imprima los hechos (vector nombreVector> ...) que contengan los valores 3 y 4 en alguna posición, y que entre éstos haya un número impar de valores. Se debe utilizar la función length\$ (ver documentación).

```
(defrule regla-ejercicio
  (vector ?nombre $?inicio ?tres &:(integerp ?tres)&:(= ?tres 3) $?medio ?cuatro
    &:(integerp ?cuatro)&:(= ?cuatro 4) $?final)
  (test (oddp (length $?medio))))
=>
(printout t "El vector " ?nombre " cumple la regla" crlf)
)

(deffacts valores

)
```

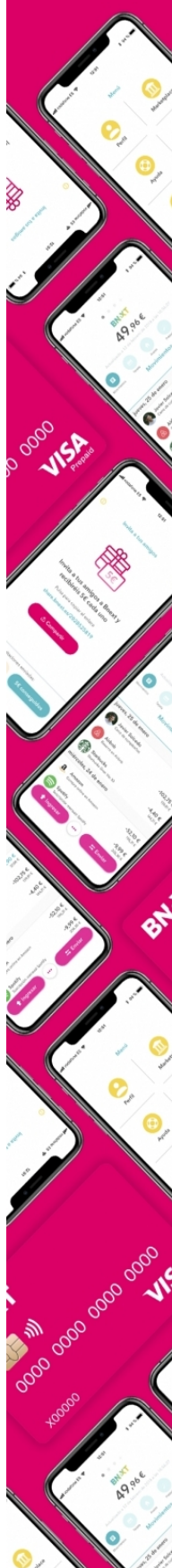
EJERCICIO 7

Haz un programa que dado un único hecho datos con un número indefinido de valores, elimine los que no sean numéricos.

```
(defrule regla-ejercicio
  ?h<-(datos $?inicio ?nonumerico $?final)
  (not (test (numberp ?nonumerico))))
=>
(retract ?h)
(assert (datos $?inicio $?final))
)
```

EJERCICIO 8

Haz un programa que dado un conjunto de hechos (vector <nombreVector> <val1> ... <valN>), detecte aquellos hechos cuyos valores no están ordenados de menor a mayor e imprima el mensaje "El vector <nombreVector> no está ordenado". Se entenderá que en la base de hechos no habrá dos hechos vector con mismo nombre de vector.

BNEXT10€
GRATISAL ACTIVAR TU
TARJETA BNEXT

```
(defrule comprobar-vector
  (vector ?vname $?)
  (exists
    (and
      (vector ?vname $? ?y ?x $?)
      (test (> ?y ?x))
    )
  )
  =>
  (printout t "+ El vector " ?vname " esta mal ordenado." crlf)
)
```

```
(deffacts los-datos
  (vector v1 2 2 3 8 9)
  (vector v2 2 3 2 9 8)
  (vector v3 2 5 3 8 9)
)
```

EJERCICIO 9

Haz un programa que dado un conjunto de hechos de la forma (dato 1), (dato 5), (dato verde)..., cree un único hecho (todos-los-datos ...) con todos los valores de los hechos anteriores.

```
(defrule regla1-ejercicio
  ?h<-(dato ?valor)
  ?w<-(todos-los-datos $?datos)
  =>
  (retract ?h)
  (retract ?w)
  (assert(todos-los-datos $?datos ?valor))
)
```

```

(defrule regla2-ejercicio
(not (todos-los-datos $?))
=>
(assert (todos-los-datos ))
)

```

EJERCICIO 10

Haz un programa que dado un conjunto hechos (vector nombreVector <val1> ... <valN>) con valores numéricos, ordene sus valores de menor a mayor.

```

(defrule ordenar
?h <- (datos ?nombre $?ini ?x ?y &:(> ?x ?y)
$?fin)
=>
(retract ?h)
(assert (datos ?nombre $?ini ?y ?x $?fin))
)

```

```

(deffacts datosss
(datos x 9 7 5 1 3)
(datos v -1 0 8 4 5 3)
)

```

EJERCICIO 11

Haz un programa que resuelva el ejercicio 9 pero sin eliminar los hechos (dato 1), (dato 5), (dato verde)....

```

(defrule regla1-ejercicio
(dato ?valor)
?w<-(todos-los-datos $?datos)
(not (todos-los-datos $? ?valor $?))
=>
(retract ?w)

```

```
(assert(todos-los-datos $?datos ?valor))  
)
```

```
(defrule regla2-ejercicio  
(not (todos-los-datos $?))  
=>  
(assert (todos-los-datos ))  
)
```

EJERCICIO 12

Haz un programa que dado un conjunto de hechos de tipo dato y un único valor numérico, imprima los valores numéricos por pantalla de menor a mayor. Vigila que el programa funciona correctamente incluso con la estrategia de ejecución de reglas Random.

```
(defrule r-ordenar-datos  
?v <- (dato ?x)  
(not (dato ?y&:(< ?y ?x)))  
=>  
(retract ?v)  
(printout t ?x crlf)  
)
```

```
(deffacts los-datos  
(dato 6)  
(dato 1.3)  
(dato 2)  
(dato -1)  
(dato 5.3)  
)
```

EJERCICIO 13

Haz un programa que resuelva el ejercicio 9, pero que sólo utilice los hechos con valores numéricos y los inserte de forma ordenada (de menor a mayor) en el hecho todos-los-datos.

```
(defrule regla1-ejercicio
  ?h<-(dato ?valor)
  ?w<-(todos-los-datos $?datos)
  (not (dato ?y&:(< ?y ?valor)))
  =>
  (retract ?h)
  (retract ?w)
  (assert(todos-los-datos $?datos ?valor))
)
```

```
(defrule regla2-ejercicio
  (not (todos-los-datos $?))
  =>
  (assert (todos-los-datos ))
)
```

```
(defrule regla3-ejercicio
  (declare (salience 50))
  ?h<-(dato $?inicio ?valor $?final)
  (not(test (numberp ?valor)))
  =>
  (retract ?h)
  (assert (dato ?inicio ?final))
)
```

BNEXT

10€ GRATIS

AL ACTIVAR TU TARJETA BNEXT

EJERCICIO 14

Una planta industrial tiene diez sensores identificados por un código numérico entre 1 y 10. Cada sensor puede encontrarse en un estado Correcto o incorrecto. Escribe una plantilla que permita representar la información relativa a los sensores y un conjunto de reglas que imprima un mensaje de advertencia si tres o más sensores se encuentran en un estado incorrecto. Sólo debe mostrarse un mensaje de error aunque haya más de tres sensores en estado incorrecto.

FORMA 1

```
(deftemplate sensor
  (slot codigo (type INTEGER))
  (slot estado (type SYMBOL))
  )

(defrule regla-ejercicio
  (exists
    (sensor (codigo ?codigo1)(estado incorrecto))
    (exists
      (and (sensor (codigo ?codigo2)(estado incorrecto))
        (sensor (codigo ?codigo3)(estado incorrecto))

        (test (<> ?codigo1 ?codigo2))
        (test (<> ?codigo1 ?codigo3))
        (test (<> ?codigo2 ?codigo3))

      )
    )
  )
=>
  (printout t "Hay 3 o mas de 3 sensores incorrectos" crlf)
  )

(deffacts sensores
```



```
(sensor (codigo 1) (estado incorrecto))
(sensor (codigo 2) (estado incorrecto))
(sensor (codigo 3) (estado incorrecto))
(sensor (codigo 4) (estado incorrecto))
(sensor (codigo 5) (estado incorrecto))
(sensor (codigo 6) (estado incorrecto))
(sensor (codigo 7) (estado incorrecto))
(sensor (codigo 8) (estado incorrecto))
(sensor (codigo 9) (estado incorrecto))
(sensor (codigo 10) (estado incorrecto))
)
```

FORMA 2

```
(deftemplate sensor
(slot codigo (type INTEGER)(range 1 10))
(slot estado (type SYMBOL)(allowed-values correcto incorrecto))
)
```

```
(defrule regla-ejercicio
(forall (sensor (codigo ?nombre1) (estado incorrecto))
        (sensor (codigo ?nombre2) (estado incorrecto))
        (sensor (codigo ?nombre3) (estado incorrecto))
        (test (<> ?nombre1 ?nombre2))
        (test (<> ?nombre1 ?nombre3))
        (test (<> ?nombre2 ?nombre3))
)
=>
(printout t "Hay 3 o mas de 3 sensores incorrectos." crlf)
)
```

```
(def facts sensores
(sensor (codigo 1) (estado correcto))
```

```

(sensor (codigo 2) (estado correcto))
(sensor (codigo 3) (estado correcto))
(sensor (codigo 4) (estado correcto))
(sensor (codigo 5) (estado correcto))
(sensor (codigo 6) (estado correcto))
(sensor (codigo 7) (estado incorrecto))
(sensor (codigo 8) (estado incorrecto))
(sensor (codigo 9) (estado incorrecto))
(sensor (codigo 10) (estado incorrecto))
)

```

EJERCICIO 15

Escribe un programa para ayudar a una persona a decidir qué plantas podría plantar. La siguiente tabla indica las características de una serie de plantas (tolerancia al frío, tolerancia a la sombra, tolerancia al clima seco...). La entrada al programa debe consistir en un conjunto de hechos (caracteristica-deseada <característica>) que indiquen características que se desee que tenga una planta. El programa debe mostrar por pantalla el nombre de las plantas que cuenten con las características indicadas, aunque aparte de esas cuenten con otras características.

```

(deftemplate planta
  (slot nombre)
  (multislot caracteristicas)
)

(defrule regla-ejercicio
  (caracteristica-deseada ?caracteristica)
  (planta (nombre ?nombre)(caracteristicas $? ?caracteristica $?))
  =>
  (printout t "La planta " ?nombre " cumple la caracteristica" crlf)
)

(deffacts plantas
  (planta (nombre "Hortensia")(caracteristicas sombra ciudad maceta rapido))
)

```

```

(planta (nombre "Adelfa")(caracteristicas ciudad maceta facil rapido))
(planta (nombre "Laurel")(caracteristicas frio sombra sequedad humedo ciudad facil rapido))
(planta (nombre "Madreselva")(caracteristicas ciudad maceta facil rapido))
(planta (nombre "Gardenia")(caracteristicas sombra acido maceta))
(planta (nombre "Enebro")(caracteristicas frio sequedad acido ciudad facil))
(planta (nombre "Pimentero")(caracteristicas frio sombra humedo acido facil))
(planta (nombre "Escaramujo")(caracteristicas frio sombra humedo ciudad facil))
(planta (nombre "Aucuba")(caracteristicas sombra sequedad maceta facil))
(planta (nombre "Azalea")(caracteristicas sombra humedo acido maceta))
)

```

EJERCICIO 16

Haz un programa en base a la información del ejercicio anterior, pero ahora se imprimirán sólo aquellas plantas que cumplan exactamente con todas las características deseadas (no puede contener otras características)

```

(deftemplate planta
  (slot nombre)
  (multislot caracteristicas)
)

(defrule regla-ejercicio
  (planta (nombre ?nombre))
    (forall(caracteristica-deseada $? ?caracteristica1 $?)
      (planta (nombre ?nombre)(caracteristicas $? ?caracteristica1 $?))
    )
    (forall(planta (nombre ?nombre)(caracteristicas $? ?caracteristica2 $?))
      (caracteristica-deseada $? ?caracteristica2 $?)
    )
)

=>

(printout t "Planta: " ?nombre crlf)
)

```

BNXT

10€ GRATIS

AL ACTIVAR TU TARJETA BNEXT

(deffacts plantas

(planta (nombre "Hortensia")(caracteristicas sombra ciudad maceta rapido))

(planta (nombre "Adelfa")(caracteristicas ciudad maceta facil rapido))

(planta (nombre "Laurel")(caracteristicas frio sombra sequedad humedo ciudad facil rapido))

(planta (nombre "Madreselva")(caracteristicas ciudad maceta facil rapido))

(planta (nombre "Gardenia")(caracteristicas sombra acido maceta))

(planta (nombre "Enebro")(caracteristicas frio sequedad acido ciudad facil))

(planta (nombre "Pimentero")(caracteristicas frio sombra humedo acido facil))

(planta (nombre "Escaramujo")(caracteristicas frio sombra humedo ciudad facil))

(planta (nombre "Aucuba")(caracteristicas sombra sequedad maceta facil))

(planta (nombre "Azalea")(caracteristicas sombra humedo acido maceta))

)

EJERCICIO 19

Haz un programa que dado un único hecho vector con un número indefinido de valores numéricos, imprima el valor que se sitúa justo en medio, o la media de los dos valores de en medio.

(deffacts datos

(vector v1 1 2 3 4 5 6 7 8 9)

(vector v2 2 2 2 6 4 2 2)

)

(defrule regla1-ejercicio

(vector ?nombre \$?ant ?valor \$?desp)

(test (= (length\$ \$?ant)

(length\$ \$?desp)

)

)

=>

(printout t "VECTOR: " ?nombre " VALOR: " ?valor crlf)

)

```

(defrule reglaz-ejercicio
(vector ?nombre $?ant ?valor1 ?valor2 $?desp)
(test (= (length$ (create$ $?ant ?valor1))
          (length$ (create$ $?desp ?valor2))
        )
      )
)

=>

(printout t "VECTOR " ?nombre " MEDIA: " (/ (+ ?valor1 ?valor2) 2) crlf)
)

```

EJERCICIO 20

Haz un programa que, dado un único hecho vector, detecte si sus valores se repiten de forma simétrica. Siempre que la salida sea correcta, puedes modificar el vector o utilizar hechos auxiliares.

```

(defrule r-no-es-simetrico
(vector ?nombre ?x $? ?y&:(<> ?x ?y))

=>

(printout t "El vector " ?nombre " NO ES SIMETRICO" crlf)
)

```

```

(defrule r-es-simetrico
(or
  (vector ?nombre ?)
  (vector ?nombre ))
)

=>

(printout t "El vector " ?nombre " es simetrico" crlf)
)

```

```

(defrule r-simetrico-mod
?h <- (vector ?nombre ?x $?vals ?x)

=>

(retract ?h)

```

```
(assert(vector ?nombre $?vals))  
)
```

```
(deffacts datos  
  (vector v1 9 4 4 9)  
  (vector v2 1 2 3 4 5)  
  (vector v3 2 5 1 5 2)  
)
```