Prácticas de CLIPS. Ejercicios curso 2021-2022

Ismael Da Palma Fernández

ÍNDICE

Ejercicio 1.- Encontrar elementos repetidos en una lista de números.

Código

El **primer ejercicio** nos pide diseñar un programa que dado un vector numérico, lo analice y devuelva los números que estén repetidos en dicho vector.

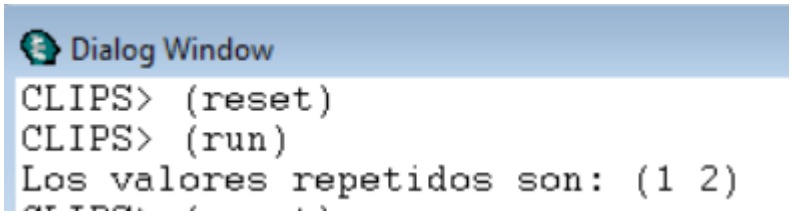
```
1  (defrule inicio
2  =>
3  (assert (hecho 1 2 3 1 4 2))
4  (assert (unico)))
5
6  (defrule salida
7  ?b <- (hecho $?a)
8  =>
9  (printout t "Los valores repetidos son: " $?a crlf))
10 (retract ?b)
11 (assert (hecho)))
12
13 (defrule unique
14 ?a <- (hecho $?q ?x $?w)
15 ?b <- (unico $?m)
16 (not(unico $? ?x $?))
17 =>
18 (retract ?a)
19 (retract ?b)
20 (assert (hecho $?q $?w))
21 (assert (unico $?m ?x)))
```

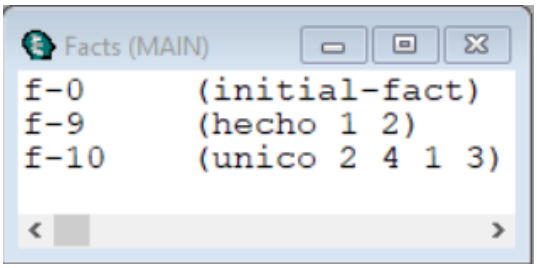
Guardará inicialmente todos los elementos del vector, al finalizar sólo contendrá los elementos repetidos del vector

Vector que guardará los elementos vistos por primera vez

Una vez visitados todos los elementos del vector, se mostrará por pantalla los elementos contenidos en el vector "hecho"

Ejemplo de ejecución





Con el vector de ejemplo definido en el código, los números repetidos son el **1** y el **2**

Ejercicio 2.- Sumar los elementos de una lista de números

Código

El **segundo ejercicio** nos pide diseñar un programa que dado un vector numérico, realice la suma de todos sus elementos y muestre el resultado de dicha suma.

```
1  (defrule inicio
2  =>
3  (assert (hecho 1 3 5 2 3 9))
4  (assert (suma 0)))
5
6  (defrule suma-int
7  (hecho $?)
8  ?x <- (hecho ?a $?b)
9  ?s <- (suma ?total)
10 =>
11 (retract ?x)
12 (retract ?s)
13 (assert (suma (+ ?total ?a)))
14 (assert (hecho $?b)))
15
16 (defrule salida
17 (not(hecho $? ?a $?))
18 (suma ?total)
19 =>
20 (printout t "La suma es :" ?total crlf))
```

Guardará inicialmente todos los elementos del vector, en cada iteración se le quitará un elemento hasta que se vacíe

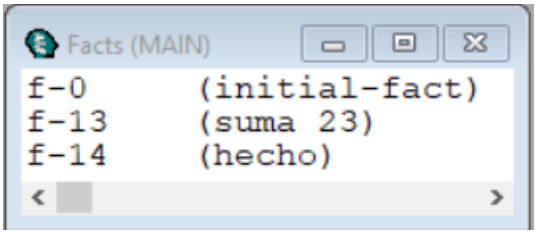
Guardaremos en "suma" la suma total de los elementos del vector, inicialmente vale 0

Insertamos en "suma" su contenido actual más el contenido de total y viceversa

Una vez se vacíe el vector "hecho", mostramos por pantalla la suma total del vector

Ejemplo de ejecución

```
CLIPS> (reset)
CLIPS> (run)
La suma es :23
```



El vector de ejemplo contiene los valores: **1-3-5-2-3-9** que al sumarlos obtenemos **23.**

Ejercicio 3.- Intersección de dos conjuntos numéricos

Código

El **tercer ejercicio** nos pide diseñar un programa que dado dos vectores, obtengamos los elementos que tienen en común ambos vectores, es decir, su intersección. Para resolver este ejercicio iremos comprobando cada elemento del primer vector con todos los del segundo vector, en el caso de que el elemento esté en ambos conjuntos, se añadirá al vector "hecho3".

```
1  (defrule inicio
2  =>
3  (assert (hecho1 1 2 3 4 5 6))
4  (assert (hecho2 2 4 7 5))
5  (assert (hecho3)))
6
7  (defrule inter
8  ?a <- (hecho1 $?q ?x $?w)
9  ?b <- (hecho3 $?m)
10 (hecho2 $? ?x $?)
11 =>
12 (retract ?a)
13 (retract ?b)
14 (assert (hecho1 $?q $?w))
15 (assert (hecho3 $?m ?x)))
16
17 (defrule salida
18 (hecho3 $?)
19 ?b <- (hecho3 $?a)
20 =>
21 (printout t "La interseccion es: " $?a crlf))
22 (retract ?b))
```

Guardará el contenido de los elementos del primer vector, con este vector vamos a ir comprobando los diferentes elementos con respecto al vector "hecho2"

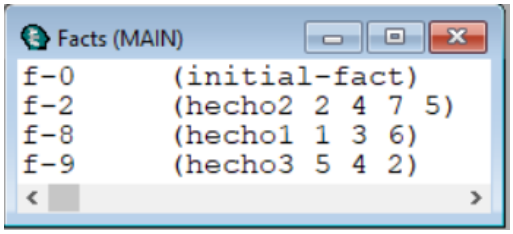
Segundo vector con el que se va a comparar el vector1 (hecho1)

Guardaremos en "hecho3" los elementos que tengan en común "hecho1" y "hecho2"

Una vez recorrido por completo el vector1, mostraremos por pantalla cuál es la intersección de los vectores

Ejemplo de ejecución

```
CLIPS> (reset)
CLIPS> (run)
La interseccion es: (5 4 2)
```



Dado los vectores **hecho1(1-2-3-4-5-6)** y **hecho2(2-4-7-5)**. Sus elementos comunes son: **5-4-2.**

Ejercicio 4.- Resta de dos conjuntos numéricos

Código

El **cuarto ejercicio** nos pide diseñar un programa que dado dos vectores, obtengamos los elementos que existan en uno de los vectores, pero que no existan en el otro. Cogeremos uno de los vectores y por cada elemento que tenga lo iremos comparando con los del otro vector, si el elemento existe en ambos vectores no se inserta en "hecho3", en caso contrario si se inserta.

```
1  (defrule inicio
2  =>
3  (assert (hecho1 1 2 3 4 5 6))
4  (assert (hecho2 2 4 7 5))
5  (assert (hecho3)))
6
7  (defrule dife
8  ?a <- (hecho1 $?q ?x $?w)
9  ?b <- (hecho3 $?m)
10 (hecho2 $? ?x $?)
11 =>
12 (retract ?a)
13 (retract ?b)
14 (assert (hecho1 $?q $?w))
15 (assert (hecho3 $?m ?x)))
16
17 (defrule salida
18 (hecho1 $?)
19 ?b <- (hecho1 $?a)
20 =>
21 (printout t "La diferencia es: " $?a crlf))
22 (retract ?b))
```

Guardaremos en "hecho1" el primer conjunto de elementos, cada iteración compararemos un elemento de "hecho1" con "hecho2", si tienen el mismo elemento, se eliminará de "hecho1", en caso contrario no

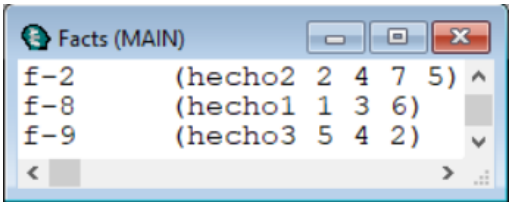
Segundo vector con el que se va a comparar el vector1 (hecho1)

Vector en el que se guardarán los elementos comunes a ambos vectores

Una vez recorrido por completo el vector1, mostraremos por pantalla los elementos que han quedado en el vector "hecho1", es decir, la diferencia entre los vectores

Ejemplo de ejecución

```
CLIPS> (reset)
CLIPS> (run)
La diferencia es: (1 3 6)
```



Dado los vectores **hecho1(1-2-3-4-5-6)** y **hecho2(2-4-7-5)**. Los elementos que no tienen en común son: **1-3-6.**

Ejercicio 5.- Diferencia entre máximo y mínimo de una lista de números

Código

El **quinto ejercicio** nos pide diseñar un programa que dado un vector, encuentre el elemento máximo y mínimo y calcule la diferencia entre estos.

```
1  (defrule inicio
2  =>
3  (assert (hecho 1 5 3 4 9)))
4
5  (defrule salida
6  (max ?a)
7  (min ?b)
8  =>
9  (printout t "Maximo(" ?a ") menos minimo(" ?b ") es :" (- ?a ?b) crlf))
10
11 (defrule maximo
12 (hecho $? ?x $?)
13 (not (hecho $? ?y&:(> ?y ?x) $?))
14 =>
15 (assert (max ?x)))
16
17 (defrule minimo
18 (hecho $? ?x $?)
19 (not (hecho $? ?y&:(< ?y ?x) $?))
20 =>
21 (assert (min ?x)))
```

Inicialmente, guardaremos todos los elementos en el vector "hecho"

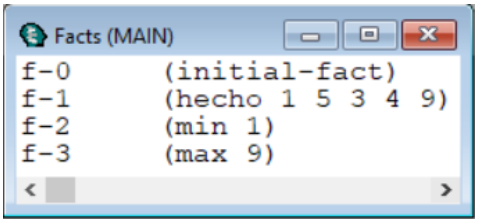
Una vez recorremos el vector por completo, mostramos por pantalla la diferencia entre el máximo y el mínimo

Creamos una regla para definir el valor máximo del vector y lo almacenaremos en "a"

Creamos otra regla para definir el valor mínimo del vector y lo almacenaremos en "b"

Ejemplo de ejecución

```
CLIPS> (reset)
CLIPS> (run)
Maximo(9) menos minimo(1) es : 8
```



Dado el vector **hecho (1-5-3-4-9)**, el **máximo es 9** y el **mínimo es 1**, la diferencia de ambos es 9-1 = **8**