Ejercicios de CLIPS

Patricia Maria Bohórquez Pérez L1

1_personas.clp

```
(defrule Regla1
        (EsPadre Pedro)
        =>
        (assert (QuiereASusHijos Pedro)))
```

1_personas.datos.clp

```
(deftemplate Persona
            (field Nombre)
            (field Edad)
            (field Sexo)
            (field EstadoCivil))

(deffacts VariosHechos
            (Persona
                (Nombre JuanCarlos)
                (Edad 33))
            (Persona
                (Nombre Maria)
                (Sexo M)) )

(deffacts OtrosHechos
            (NumeroDeReactores 4))
```

2_alarma.clp

```
(deftemplate Emergencia
        (field tipo)
        (field sector)
        (field campo))

(deftemplate SistemaExtincion
        (field tipo)
        (field status)
        (field UltimaRevision))

(defrule Emergencia-Fuego-ClaseB
        (Emergencia
            (tipo ClaseB))
        (SistemaExtincion
            (tipo DioxidoCarbono)
             (status operativo))
        =>
           (printout t "Usar extintor CO2" crlf))
```

2_alarma.datos.clp

```
(deffacts HechosSistemaExtincion
        (SistemaExtincion
           (tipo DioxidoCarbono)
           (status operativo)
           (UltimaRevision diciembre)))
```

3_eccema.clp

```
(deftemplate FichaPaciente
        (field Nombre)
        (field Casado)
        (field Direccion))
(deftemplate DatosExploracion
(field Nombre)
        (multifield Sintomas)
        (field GravedadAfeccion))
(defrule DiagnosticoEccema
        (DatosExploracion
                (Nombre ?N)
                (Sintomas $? picor $? vesiculas $?))
        (FichaPaciente
                (Nombre ?N))
        =>
(printout t "Posible diagnóstico para el paciente " ?N ": Eccema " crlf))
```

3_eccema.datos.clp

```
(deffacts Hecho2
            (FichaPaciente
               (Nombre Pedro))
            (FichaPaciente
               (Nombre Juan))
            (FichaPaciente
               (Nombre Alejandro)))
```

```
(deffacts Hechos
        (DatosExploracion
          (Nombre Pedro)
          (Sintomas rojo picor amarillo vesiculas verde))
        (DatosExploracion
          (Nombre Juan)
          (Sintomas rojo picor amarillo vesiculas verde))
        (DatosExploracion
          (Nombre Alejandro)
          (Sintomas amarillo vesiculas)))
```

4_IRPF.clp

```
(deftemplate Persona
   (field Nombre)
   (field Edad)
   (field NombreConyuge)
   (field PosicionEconomica)
   (field Salario))

;Hechos

(deffacts VariosHechos
      (Persona
            (Nombre Ricardo)
            (Edad 23)
            (NombreConyuge Maria)
            (PosicionEconomica Desahogada)
            (Salario 1000)))
```

```
(deffacts VariosHechos2
        (Persona
                (Nombre Ana)
                (Edad 40)
                (NombreConyuge Jose)
                (PosicionEconomica Desahogada)
                (Salario 12000)))

(deffacts VariosHechos3
        (Persona
                (Nombre Maria)
                (Edad 39)
                (NombreConyuge Marcos)
                (PosicionEconomica Dificultad)
                (Salario 1000)))

(deffacts VariosHechos4
        (Persona
                (Nombre Alberto)
                (Edad 60)
                (NombreConyuge Miriam)
                (PosicionEconomica Desahogada)
                (Salario 10400)))

(deffacts VariosHechos5
        (Persona
                (Nombre Paco)
                (Edad 60)
                (NombreConyuge Maria)
                (PosicionEconomica Dificultad)
                (Salario 1000)))
```

```
(deffacts VariosHechos6
	(Persona
		(Nombre Manolo)
		(Edad 80)
		(NombreConyuge Josefina)
		(PosicionEconomica Desahogada)
		(Salario 10060)))

;REGLAS

;APARTADO A

(defrule PersonaEdad
	(Persona
		(Nombre ?Nombre)
		(Edad 60))
		=>
		(printout t ?Nombre " tiene 60 años" crlf ))


;APARTADO B

(defrule PersonaEdadSalario
	(Persona
		(Nombre ?Nombre)
		(Edad 40)
		(Salario ?Salario))
		=>
		(printout t ?Nombre " tiene 40 años y un salario de " ?Salario crlf ))
```

```
;APARTADO C


(defrule PersonaDatos
	(Persona
		(Nombre ?Nombre)
		(Edad ?Edad)
		(NombreConyuge ?NombreConyuge)
		(PosicionEconomica ?PosicionEconomica)
		(Salario ?Salario))
		=>
		(printout t ?Nombre " tiene " ?Edad ", esta casado con " ?NombreConyuge " con una posicion economica " ?PosicionEconomica " y salario de " ?Salario  crlf ))



;APARTADO D


(defrule PersonaEconomia
	(Persona
		(Nombre ?Nombre)
		(NombreConyuge ?NombreConyuge)
		(PosicionEconomica Desahogada))
		=>
		(printout t ?Nombre " está casado con " ?NombreConyuge " y tiene una posicion economica desahogada " crlf ))



;APARTADO E


(defrule PersonaConyuge
	(Persona
		(Nombre ?Nombre)
		(NombreConyuge ?NombreConyuge)
```

```
                (PosicionEconomica Desahogada))


        =>
                (assert(DatosFiscales ?Nombre ConyugeDesahogado)))

;APARTADO F

(defrule BorrarDatos
   (Persona
      (Nombre ?Nombre)
      (PosicionEconomica Desahogada))
   ?PE <- (Persona (PosicionEconomica Desahogada))
   =>
   (retract ?PE)
  (printout t ?Nombre " ha sido eliminado" crlf))


;APARTADO G

(defrule BorrarDatos2
   (Persona
      (Nombre ?Nombre)
      (PosicionEconomica Desahogada))
   ?PE <- (Persona (PosicionEconomica Desahogada))
   =>
   (retract ?PE)
   (printout t ?Nombre " eliminado" crlf))
```

5_eccema.clp

```
(deftemplate FichaPaciente
        (field Nombre)
        (field Casado)
        (field Direccion))

(deftemplate DatosExploracion
        (field Nombre)
        (multifield Sintomas)
        (field GravedadAfeccion))

(deftemplate Diagnostico
        (field Nombre)
        (field Resultado)
        (field ProximaRevision))

(deftemplate DiagnosticoPersonas
        (field Nombre)
        (field Edad)
        (field Tipo))

(deftemplate Terapia
        (field Nombre)
        (field PrincipioActivo)
        (field Posologia))

(deftemplate Terapia2
        (field Nombre)
        (field Posologia))
```

```
;REGLAS

;APARTADO A

(defrule DiagnosticoEccema
	(DatosExploracion
		(Nombre ?N)
		(Sintomas $? picor $? vesiculas $?))
	(FichaPaciente
		(Nombre ?N))
	=>
(printout t "Posible diagnóstico para el paciente " ?N ": Eccema " crlf))



;APARTADO B

(defrule DiagnosticoGeneral
	(Diagnostico
		(Nombre ?Nombre)
		(Resultado ?Resultado)
		(ProximaRevision ?ProximaRevision))
		=>
		(printout t ?Nombre " tiene un resultado " ?Resultado " y su proxima revision es
en " ?ProximaRevision crlf))



;APARTADO C

(defrule DiagnosticoEdad
	(DiagnosticoPersonas
		(Nombre ?Nombre)
```

```
                (Edad ?Edad)
                (Tipo ?Tipo))
                (test (< ?Edad 2))
                =>
                (printout t ?Nombre " tiene menos de dos año " ?Tipo crlf))


;APARTADO D

(defrule DiagnosticoTerapia
        (Terapia
                (Nombre ?Nombre)
                (PrincipioActivo ?PrincipioActivo)
                (Posologia ?Posologia))
                =>
                (printout t ?Nombre " es " ?PrincipioActivo " se le administrara " ?Posologia
crlf))


;APARTADO E

(defrule TerapiaRecomendable
        (Terapia2
                (Nombre ?Nombre)
                (Posologia ?Posologia))
                =>
                (printout t "Es aconsejable para " ?Nombre " usar " ?Posologia crlf))
```

5_eccema.datos.clp

```
(deffacts Hecho2
        (FichaPaciente
          (Nombre Pedro))
        (FichaPaciente
          (Nombre Juan))
        (FichaPaciente
          (Nombre Erica))
        (FichaPaciente
          (Nombre Marta)))

(deffacts Hechos
        (DatosExploracion
          (Nombre Pedro)
          (Sintomas picor rojo vesiculas inflamada))
        (DatosExploracion
          (Nombre Juan)
          (Sintomas picor rojo vesiculas normal))
        (DatosExploracion
          (Nombre Erica)
          (Sintomas picor rojo))
        (DatosExploracion
          (Nombre Marta)
          (Sintomas vesiculas inflamada)))

(deffacts Hecho3
      (Diagnostico
            (Nombre Pedro)
            (Resultado positivo)
            (ProximaRevision Marzo))
```

```
(Diagnostico
        (Nombre Juan)
        (Resultado negativo)
        (ProximaRevision Junio))
    (Diagnostico
        (Nombre Erica)
        (Resultado positivo)
        (ProximaRevision Junio))
    (Diagnostico
        (Nombre Marta)
        (Resultado negativo)
        (ProximaRevision Abril)))

(deffacts Hecho4
    (DiagnosticoPersonas
        (Nombre Pedro)
        (Edad 30)
        (Tipo Es_adulto))
    (DiagnosticoPersonas
        (Nombre Juan)
        (Edad 10)
        (Tipo Es_adulto))
    (DiagnosticoPersonas
        (Nombre Erica)
        (Edad 0)
        (Tipo Es_un_bebe))
    (DiagnosticoPersonas
        (Nombre Marta)
        (Edad 1)
        (Tipo Es_un_bebe)))
```

```
(deffacts Hecho5
        (Terapia
                (Nombre Pedro)
                (PrincipioActivo grave)
                (Posologia corticoide))
        (Terapia
                (Nombre Juan)
                (PrincipioActivo leve)
                (Posologia corticoide))
        (Terapia
                (Nombre Erica)
                (PrincipioActivo grave)
                (Posologia corticoide))

        (Terapia
                (Nombre Marta)
                (PrincipioActivo leve)
                (Posologia crema_hidratante)))

(deffacts Hecho6
        (Terapia2
                (Nombre Pedro)
                (Posologia corticoide))
        (Terapia2
                (Nombre Juan)
                (Posologia corticoide))
        (Terapia2
                (Nombre Erica)
                (Posologia corticoide))
        (Terapia2
                (Nombre Marta)
                (Posologia crema_hidratante)))
```

6_minimo.clp

```
(deffacts Minimo
        (elemento 3)
        (elemento 6)
        (elemento 26)
        (elemento 36)
        (elemento 60)
        (elemento 66))


(defrule ElementoMinimo
(elemento $? ?x $?)
(not (elemento $? ?y&:(< ?y ?x) $?))
=>
(printout t "El minimo es "?x crlf))
```

7_suma.clp

```
(deftemplate Suma
        (field elem1)
        (field elem2)
        (field elem3))

(deffacts SumaElementos
        (Suma
                (elem1 3)
```

```
                (elem2 6)
                (elem3 26)))

(defrule SumaVector
        (Suma
                (elem1 ?elem1)
                (elem2 ?elem2)
                (elem3 ?elem3))
                =>
(bind ?Total (+ ?elem1 ?elem2 ?elem3))
(printout t "La suma de los elementos es = " ?Total crlf))
```

8_sustituciones.clp

```
(deftemplate Palabra
        (multifield Caracter1)
        (multifield Caracter2)
        (multifield Caracter3)
        (multifield Caracter4))


(deffacts Sustituir
         (Palabra
                (Caracter1 B)
                (Caracter2 C)
                (Caracter3 D)
                (Caracter4 Z)))


;Reglas

(defrule Cambiar1
```

```
  ?indice <- (Palabra (Caracter2 ?Caracter2))

  =>

  (modify ?indice (Caracter2 D L))

  (printout t " Con la regla 1 se ha modificado " ?Caracter2 " por D L quedando el vector como
B D L D Z"  crlf))



(defrule Cambiar2

  ?indice <- (Palabra (Caracter2 ?Caracter2))

  =>

  (modify ?indice (Caracter2 B M))

  (printout t " Con la regla 2 se ha modificado " ?Caracter2 " por B M quedando el vector como
B B M D Z"  crlf))


(defrule Cambiar3

  ?indice <- (Palabra (Caracter1 ?Caracter1))

  =>

  (modify ?indice (Caracter1 M M))

  (printout t " Con la regla 3 se ha modificado " ?Caracter1 " por M M quedando el vector como
M M C D Z"  crlf))



(defrule Cambiar4

  ?indice <- (Palabra (Caracter4 ?Caracter4))

  =>

  (modify ?indice (Caracter4 B B M))

  (printout t " Con la regla 2 se ha modificado " ?Caracter4 " por B B M quedando el vector
como B C D B B M"  crlf))
```

9_union.clp

```
(deffacts datos-iniciales
        (cadena1 B C A D E E B C E)
        (cadena2 E E B F D E))

(defrule calcula-union
        =>
        (assert (union)))

(defrule union-base
        ?union <- (union $?u)
        ?cadena1 <- (cadena1 $?e-1)
        ?cadena2 <- (cadena2)
        =>
        (retract ?cadena1 ?cadena2 ?union)

        (assert (union ?e-1 ?u))
        (assert (escribe-solucion)))

(defrule escribe-solucion
        (escribe-solucion)
        (union $?u)
        =>
        (printout t "La union es " ?u crlf))

(defrule union-con-primero-compartido
        ?cadena2 <- (cadena2 ?e $?r-2)
        (cadena1 $? ?e $?)
        =>
        (retract ?cadena2)
        (assert (cadena2 ?r-2)))
```

```
(defrule union-con-primero-no-compartido
        ?union <- (union $?u)
        ?cadena2 <- (cadena2 ?e $?r-2)
        (not (conjunto-1 $? ?e $?))
        =>
        (retract ?cadena2 ?union)
        (assert (cadena2 ?r-2)
        (union ?u ?e)))
```

10_dependencia.clp

```
(deftemplate Producto
        (field CodigoVendedor)
        (field CodigoProducto)
        (field PVPProducto)
)
(defrule comprobacion_positiva
        (Producto
                (CodigoVendedor ?cv1)
                (CodigoProducto ?cp1)
                (PVPProducto ?pvp1)
        )
        (Producto
                (CodigoVendedor ?cv2)
                (CodigoProducto ?cp2)
                (PVPProducto ?pvp2)
        )
        (test(<> ?cv1 ?cv2))
        (test(= ?cp1 ?cp2))
```

```
        (test(= ?pvp1 ?pvp2))
        =>
        (assert (confirmado ?cv1 ?cv2 ?cp1 ?cp2 ?pvp1 ?pvp2))
)
(defrule comprobacion_negativa
        (Producto
                (CodigoVendedor ?cv1)
                (CodigoProducto ?cp1)
                (PVPProducto ?pvp1)
        )
        (Producto
                (CodigoVendedor ?cv2)
                (CodigoProducto ?cp2)
                (PVPProducto ?pvp2)
        )
        (test(<> ?cv1 ?cv2))
        (test(= ?cp1 ?cp2))
        (test(<> ?pvp1 ?pvp2))
        =>
        (assert (negativo ?cv1 ?cv2 ?cp1 ?cp2 ?pvp1 ?pvp2))
)
(defrule mostrar_positivos
        (confirmado ?c_cv1 ?c_cv2 ?c_cp1 ?c_cp2 ?c_pvp1 ?c_pvp2)
        ?caso <-(confirmado ?c_cv2 ?c_cv1 ?c_cp1 ?c_cp2 ?c_pvp1 ?c_pvp2)
        =>
        (retract ?caso)
        (printout t "Los vendedores " ?c_cv1 " y " ?c_cv2 " cumplen la regla con
CodigoProducto(" ?c_cp1 ")" crlf)
)
(defrule mostrar_negativos
        (negativo ?n_cv1 ?n_cv2 ?n_cp1 ?n_cp2 ?n_pvp1 ?n_pvp2)
        ?caso <-(negativo ?n_cv2 ?n_cv1 ?n_cp1 ?n_cp2 ?n_pvp2 ?n_pvp1)
        =>
```

```
        (retract ?caso)
        (printout t "Los vendedores " ?n_cv1 " y " ?n_cv2 " no cumplen la regla con
CodigoProducto(" ?n_cp1 ")" crlf)
)
```

10_dependencia.datos.clp

```
(deffacts Productos_tipo_1
        (Producto
                (CodigoVendedor 1)
                (CodigoProducto 1000)
                (PVPProducto 1.00)
        )
)
(deffacts Productos_tipo_2
        (Producto
                (CodigoVendedor 2)
                (CodigoProducto 2000)
                (PVPProducto 2.00)
        )
)
(deffacts Productos_tipo_3
        (Producto
                (CodigoVendedor 3)
                (CodigoProducto 4000)
                (PVPProducto 2.00)
        )
)
(deffacts Productos_tipo_combo_1
        (Producto
                (CodigoVendedor 12)
```

```
            (CodigoProducto 1000)
            (PVPProducto 1.00)
        )
)
(deffacts Productos_tipo_combo_2
        (Producto
            (CodigoVendedor 22)
            (CodigoProducto 2000)
            (PVPProducto 1.00)
        )
)
(deffacts Producto5
        (Producto
            (CodigoVendedor 5)
            (CodigoProducto 2000)
            (PVPProducto 2.00)
        )
)
```

11_empaquetado.clp

```
(defrule Forrado
        (declare (salience 100))
        (Articulo
            (Nombre ?Nm)
            (Tipo ?T)
            (Forrado No)
            (Empaquetado No)
```

```
            (Dimension ?D)
        )
        ?art <- (Articulo (Nombre ?Nm)(Tipo ?T) (Forrado No) (Empaquetado No)(Dimension
?D))
        =>
        (retract ?art)
        (printout t "Se forra el articulo cuyo nombre es : " ?Nm " y su tipo: " ?T crlf)
        (assert (Articulo (Nombre ?Nm)(Tipo ?T) (Forrado Si) (Empaquetado No)(Dimension
?D)))
)

(defrule Empaquetado
        (declare (salience 99))
        (Articulo
            (Nombre ?Nm)
            (Tipo ?T)
            (Forrado Si)
            (Empaquetado No)
            (Dimension ?D)
        )
        ?art <- (Articulo (Nombre ?Nm)(Tipo ?T) (Forrado Si) (Empaquetado No)(Dimension
?D))
        (Caja
            (IdCaja ?I)
            (Abierta Si)
            (Empezada ?E)
            (TipoContenido ?T)
            (EspacioLibre ?L)
        )
        ?caj <- (Caja (IdCaja ?I)(Abierta Si)(Empezada ?E)(TipoContenido ?T)(EspacioLibre
?L))
        (test (< ?D ?L))
        =>
        (retract ?art ?caj)
```

```
        (printout t "Se empaqueta el articulo cuyo nombre es : " ?Nm " y su tipo: " ?T  " en la
caja " ?I crlf)
        (assert (Articulo (Nombre ?Nm)(Tipo ?T) (Forrado Si) (Empaquetado Si)(Dimension
?D)))
        (bind ?L (- ?L ?D))
        (assert(Caja (IdCaja ?I)(Abierta Si)(Empezada Si)(TipoContenido ?T)(EspacioLibre
?L)))
        if (= ?L 0)
        then (printout t "La caja " ?I " esta llena" crlf)
)
```

11_empaquetado.datos.clp

```
(deftemplate Articulo
 (field Nombre)
 (field Tipo)   ; Valores permitidos: fragil, pesado
 (field Forrado)    ; Valores permitidos Si - No
 (field Empaquetado)  ; Valores permitidos Si - No
 (field Dimension)  ; Valor numérico de 0 a 200
)

(deftemplate Caja
 (field IdCaja)
 (field Abierta)   ; Valores permitidos Si - No
 (field Empezada)   ; Valores permitidos Si - No
 (field TipoContenido)  ; Valores permitidos: fragil, pesado
 (field EspacioLibre)  ; Valor numérico que indica
    ; el espacio que todavía queda libre.
    ; Al principio, contiene la dimensión
    ; de la caja. Viene en las mismas
```

```
    ; unidades que el field dimensión
; del template Articulo.
)


(deffacts Inventario
        (Articulo
                (Nombre camara)
                (Tipo fragil)
                (Forrado No)
                (Empaquetado No)
                (Dimension 10))

        (Articulo
                (Nombre spinner)
                (Tipo pesado)
                (Forrado No)
                (Empaquetado No)
                (Dimension 5))

        (Articulo
                (Nombre movil)
                (Tipo fragil)
                (Forrado No)
                (Empaquetado No)
                (Dimension 10)))

(deffacts MisCajas
        (Caja
                (IdCaja 1)
                (Abierta Si)
                (Empezada No)
```

```
        (TipoContenido fragil)
        (EspacioLibre 6))

    (Caja
        (IdCaja 2)
        (Abierta Si)
        (Empezada No)
        (TipoContenido pesado)
        (EspacioLibre 11))

    (Caja
        (IdCaja 3)
        (Abierta No)
        (Empezada No)
        (TipoContenido fragil)
        (EspacioLibre 5)))
```

12_bandera.clp

```
(deftemplate bandera
    (field Pais)
    (multifield Bandera))

(defrule pedir_color
    ?fase <- (fase pedir_color)
    =>
    (retract ?fase)
```

```
        (printout t "Introduzca un color: ")
        (assert (color (read)))
        (assert (fase pedir_respuesta)))

(defrule pedir_respuesta
        ?fase <- (fase pedir_respuesta)
        =>
        (retract ?fase)
        (printout t "¿Quiere introducir otro color? (s/n) ")
        (assert (respuesta (read))
        (fase comprobar_respuesta)))

(defrule respuesta_no
        ?fase <- (fase comprobar_respuesta)
        ?respuesta <- (respuesta n)
        =>
        (retract ?fase ?respuesta)
        (assert (fase calcular)))

(defrule respuesta_si
        ?fase <- (fase comprobar_respuesta)
        ?respuesta <- (respuesta s)
        =>
        (retract ?fase ?respuesta)
        (assert (fase pedir_color)))

(defrule respuesta_incorrecta
        ?fase <- (fase comprobar_respuesta)
        ?respuesta <- (respuesta ~s&~n)
        =>
        (retract ?fase ?respuesta)
        (assert (fase pedir_respuesta))
```

```
        (printout t "Responda 's' o 'n'" crlf))


(defrule banderas_colores
        (fase calcular)
        (bandera (Pais ?Pais))
        (forall (color ?color)
                (bandera (Pais ?Pais) (Bandera $? ?color $?)))
        =>
        (printout t ?Pais crlf))
```

12_bandera.datos.clp

```
(deffacts banderas
        (bandera (Pais Espanna)(Bandera rojo amarillo))
        (bandera (Pais Belgica)(Bandera negro amarillo rojo))
        (bandera (Pais Polonia)(Bandera blanco rojo))
        (bandera (Pais Monaco)(Bandera blanco rojo))
        (bandera (Pais Suiza)(Bandera amarillo azul))
        (bandera (Pais Panama)(Bandera blanco azul))
        (fase pedir_color))
```