# Clips: Acciones procedimentales

**José A. Alonso y Francisco J. Martín**

**Ciencias de la Computación e Inteligencia Artificial**

Universidad de Sevilla

# Nim con if

- **Ejemplo**

```
(deffacts INICIO::indeterminados
  (numero-de-piezas-elegidas indeterminado)
  (turno indeterminado))

(defrule INICIO::elige-jugador
  ?h <- (turno indeterminado)
  =>
  (retract ?h)
  (printout t "Elige quien empieza: "
              "computadora o Humano (c/h) ")
  (bind ?jugador (read))
  (if (or (eq ?jugador c) (eq ?jugador h))
      then (assert (turno ?jugador))
      else (printout t ?jugador
                       " es distinto de c y h" crlf)
           (assert (turno indeterminado))))

(defrule INICIO::elige-numero-de-piezas
  ?h <- (numero-de-piezas-elegidas indeterminado)
  =>
  (retract ?h)
  (printout t "Escribe el numero de piezas: ")
  (bind ?n (read))
  (if (and (integerp ?n) (> ?n 0))
      then (assert (numero-de-piezas ?n))
      else (printout t ?n
                     " no es un numero entero mayor que 0"
                     crlf)
           (assert (numero-de-piezas-elegidas
                     indeterminado))))
```

# Nim con if

```
(defrule HUMANO::eleccion-humana
  ?pila <- (numero-de-piezas ?n&:(> ?n 1))
  =>
  (retract ?pila)
  (printout t "Escribe el numero de piezas que coges: ")
  (bind ?m (read))
  (if (and (integerp ?m) (>= ?m 1) (<= ?m 3) (< ?m ?n))
      then (bind ?nuevo-numero-de-piezas (- ?n ?m))
           (assert (numero-de-piezas
                     ?nuevo-numero-de-piezas))
           (printout t "Quedan " ?nuevo-numero-de-piezas
                     " pieza(s)" crlf)
           (return)
      else (printout t "Tiene que elegir "
                       "un numero entre 1 y 3" crlf)
           (assert (numero-de-piezas ?n))))
```

- **Sintaxis**

```
(if <expresion>
    then <accion>*
    [else <accion>*])
```

# Nim con `while`

- **Ejemplo 1**

```
(defrule INICIO::elige-jugador
  =>
  (printout t "Elige quien empieza: "
              "computadora o Humano (c/h) ")
  (bind ?jugador (read))
  (while (not (or (eq ?jugador c) (eq ?jugador h))) do
         (printout t ?jugador " es distinto de c y h"
                     crlf)
         (printout t "Elige quien empieza: "
                     "computadora o humano (c/h) ")
         (bind ?jugador (read)))
  (assert (turno ?jugador)))

(defrule INICIO::elige-numero-de-piezas
  =>
  (printout t "Escribe el numero de piezas: ")
  (bind ?n (read))
  (while (not (and (integerp ?n) (> ?n 0))) do
         (printout t ?n " no es un numero entero "
                        "mayor que 0" crlf)
         (printout t "Escribe el numero de piezas: ")
         (bind ?n (read)))
  (assert (numero-de-piezas ?n)))
```

# Nim con `while`

```
(defrule HUMANO::eleccion-humana
  ?pila <- (numero-de-piezas ?n&:(> ?n 1))
  =>
  (printout t "Escribe el numero de piezas "
              "que coges: ")
  (bind ?m (read))
  (while (not (and (integerp ?m)
                   (>= ?m 1)
                   (<= ?m 3)
                   (< ?m ?n))) do
         (printout t "Tiene que elegir un numero "
                     "entre 1 y 3" crlf)
         (printout t "Escribe el numero de piezas "
                     "que coges: ")
         (bind ?m (read)))
  (retract ?pila)
  (bind ?nuevo-numero-de-piezas (- ?n ?m))
  (assert (numero-de-piezas ?nuevo-numero-de-piezas))
  (printout t "Quedan " ?nuevo-numero-de-piezas
              " pieza(s)" crlf)
  (return))
```

- **Sintaxis**

```
(while <expresion> [do]
       <action>*)
```

# Nim con funciones definidas

- **Ejemplo 2**

```
(deffunction INICIO::jugador-elegido ()
  (printout t "Elige quien empieza: "
              "computadora o humano (c/h) ")
  (bind ?jugador (read))
  (while (not (or (eq ?jugador c) (eq ?jugador h))) do
         (printout t ?jugador
                     " es distinto de c y h" crlf)
         (printout t "Elige quien empieza: "
                     "computadora o humano (c/h) ")
         (bind ?jugador (read)))
  ?jugador)

(defrule INICIO::elige-jugador
  =>
  (assert (turno (jugador-elegido))))

(deffunction INICIO::piezas-elegidas ()
  (printout t "Escribe el numero de piezas: ")
  (bind ?n (read))
  (while (not (and (integerp ?n) (> ?n 0))) do
         (printout t ?n " no es un numero entero "
                        "mayor que 0" crlf)
         (printout t "Escribe el numero de piezas: ")
         (bind ?n (read)))
  ?n)

(defrule INICIO::elige-numero-de-piezas
  =>
  (assert (numero-de-piezas (piezas-elegidas))))
```

# Nim con funciones definidas

```
(deffunction HUMANO::piezas-cogidas-de (?n)
  (printout t "Escribe el numero de piezas que coges: ")
  (bind ?m (read))
  (while (not (and (integerp ?m)
                   (>= ?m 1)
                   (<= ?m 3)
                   (< ?m ?n))) do
         (printout t "Tiene que elegir un numero "
                     "entre 1 y 3" crlf)
         (printout t "Escribe el numero de piezas "
                     "que coges: ")
         (bind ?m (read)))
  ?m)

(defrule HUMANO::eleccion-humana
  ?pila <- (numero-de-piezas ?n&:(> ?n 1))
  =>
  (retract ?pila)
  (bind ?m (piezas-cogidas-de ?n))
  (bind ?nuevo-numero-de-piezas (- ?n ?m))
  (assert (numero-de-piezas ?nuevo-numero-de-piezas))
  (printout t "Quedan " ?nuevo-numero-de-piezas
              " pieza(s)" crlf)
  (return))
```

# Nim con acciones definidas

```
(deffunction INICIO::turno-de-jugador-elegido ()
  (printout t "Elige quien empieza: "
              "computadora o humano (c/h) ")
  (bind ?jugador (read))
  (while (not (or (eq ?jugador c) (eq ?jugador h))) do
         (printout t ?jugador " es distinto de c y h"
                     crlf)
         (printout t "Elige quien empieza: "
                     "computadora o humano (c/h) ")
         (bind ?jugador (read)))
  (assert (turno ?jugador)))

(defrule INICIO::elige-jugador
  =>
  (turno-de-jugador-elegido))

(deffunction INICIO::numero-de-piezas-elegidas ()
  (printout t "Escribe el numero de piezas: ")
  (bind ?n (read))
  (while (not (and (integerp ?n) (> ?n 0))) do
         (printout t ?n " no es un numero entero "
                        "mayor que 0" crlf)
         (printout t "Escribe el numero de piezas: ")
         (bind ?n (read)))
  (assert (numero-de-piezas ?n)))

(defrule INICIO::elige-numero-de-piezas
  =>
  (numero-de-piezas-elegidas))
```

# Nim con acciones definidas

```
(deffunction HUMANO::coge-piezas (?n)
  (printout t "Escribe el numero de piezas que coges: ")
  (bind ?m (read))
  (while (not (and (integerp ?m)
                   (>= ?m 1)
                   (<= ?m 3)
                   (< ?m ?n))) do
         (printout t "Tiene que elegir un numero "
                     "entre 1 y 3" crlf)
         (printout t "Escribe el numero de piezas "
                     "que coges: ")
         (bind ?m (read)))
  (bind ?nuevo-numero-de-piezas (- ?n ?m))
  (assert (numero-de-piezas ?nuevo-numero-de-piezas))
  (printout t "Quedan " ?nuevo-numero-de-piezas
              " pieza(s)" crlf))

(defrule HUMANO::eleccion-humana
  ?pila <- (numero-de-piezas ?n&:(> ?n 1))
  =>
  (retract ?pila)
  (coge-piezas ?n)
  (return))
```

# Observaciones

- **Ayuda sobre las funciones procedimentales**

  `(help) // FUNCTION_SUMMARY // PROCEDURAL_FUNCTIONS`

- **Relación de funciones procedimentales:**
  - `(bind <variable> <expresion>*)`
  - `(if <expresion> then <accion>* [else <accion>*])`
  - `(while <expresion> [do] <accion>*)`
  - `(loop-for-count <rango> [do] <accion>*)`
  - `(progn <expresion>*)`
  - `(return [<expresion>])`
  - `(break)`
- **Recomendaciones del uso de funciones procedimentales:**
  - **Uso juicioso**
  - **No escribir acciones con anidamientos de `if` y `while`**