Licenciatura en Obras Públicas

Ingeniería del Conocimiento

Curso 08/09

Ingeniería del Conocimiento

Curso 08/09

- **La asignatura**
  - La Ingeniería del Conocimiento es el proceso de diseñar y hacer operativos sistemas que resuelven problemas usando conocimiento.

- **El profesor:**
  - Nombre: Fernando Pérez Nava
  - Despacho: Edificio de la ETSII 2ª Planta
  - Correo electrónico: fdoperez@ull.es
  - Tutorías: Jueves de 9.30 a 1.30

- **Las clases:**
  - Miercoles 12.30-14.30
    Viernes 12.30-14.30

Ingeniería del Conocimiento

Curso 08/09

- **¿Cómo aprobar la asignatura?**
  - La evaluación de la asignatura se realiza mediante un trabajo relacionado con lo visto en clase.
  - La asistencia a clase es obligatoria (salvo causa justificada)
- **Objetivos**
  - El objetivo de esta asignatura es dar una visión general introductoria de la Ingeniería del Conocimiento y de algunos de los aspectos más importantes incluidos en esta disciplina.
  - El alumno conocerá técnicas y herramientas para la adquisición de conocimientos y para su representación adecuada, de tal manera que pueda ser tratada, manejada y obtener nuevos conocimientos.
  - Se pretende que el alumno adquiera capacidades básicas para modelar, diseñar e implementar sistemas expertos. Conocimientos de sus características, estructura y de los procesos de desarrollo, así como de habilidades y destrezas para su realización metodológica práctica

Ingeniería del Conocimiento

Curso 08/09

- **Temario**
  - TEMA 1. Introducción a la Ingeniería del Conocimiento
    - Inteligencia Artificial
    - Sistemas basados en conocimiento
    - Sistemas expertos
    - Ingeniería del Conocimiento.
  - TEMA 2. Sistemas Basados en Conocimiento: Sistemas Expertos
    - Introducción. Características. Estructura y funcionalidades
    - Dominios de aplicación de los SBC.
    - Fases de Desarrollo de los SBC.

Ingeniería del Conocimiento

Curso 08/09

- **Temario**
  - TEMA 3. Introducción a la programación en CLIPS
    - Introducción a CLIPS
    - Comandos básicos
    - Introducción a los Hechos y Reglas
    - Variables y Emparejamiento de Patrones
    - Operadores Matemáticos y Lógicos
    - Plantillas y Casillas
    - Entrada y Salida

Ingeniería del Conocimiento

Curso 08/09

- **Material de la Asignatura**
  - Transparencias de clase
    - Están disponibles para los alumnos
  - Bibliografía recomendada
    - Gómez A., Juristo N., Montes V. y Pazos J. *Ingeniería del Conocimiento.* Ed. Centro de Estudios Ramón Areces. 1997
    - Botía, J. A. Cadenas, J.M. Hernández, Moreno, L.D. y Paniagua, E. (2001) *Inteligencia Artificial*. Colección Texto Guía Universidad de Murcia.
    - Giarratano, J. y Riley, G. (2001) *Sistemas Expertos. Principios y programación*, Thomson Learning.
    - Mira J., Delgado A.E., Boticario J.G. y Díez F.J. *Aspectos Básicos de La Inteligencia Artificial.* Ed. Sanz y Torres. 1995
    - Russell S. y Norvig P. (1997) *Inteligencia Artificial: Un enfoque moderno.* Prentice Hall.
    - Todos están disponibles en la biblioteca
  - Se darán más referencias a lo largo del curso

Introducción a la Ingeniería del Conocimiento

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **Contenidos**
  - Inteligencia Artificial (IA)
  - Sistemas basados en Conocimiento (SBC)
  - Sistemas Expertos (SE)
  - Ingeniería del Conocimiento (IC)

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **¿Qué es la Inteligencia Artificial?**
  - No existe una definición unificada debido a la ambigüedad del término inteligencia
  - Una clasificación de varias definiciones:

<table>
<tr><td rowspan="2">Pensamiento-Actuación</td><td>Pensamiento</td><td>“La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades tales como toma de decisiones, resolución de problemas, aprendizaje,…” (Bellman, 1978)</td><td>“El estudio de las facultades mentales mediante el uso de modelos computacionales” (Charniack y McDermott, 1985)</td></tr>
<tr><td>Actuación</td><td>“La IA es la parte de la ciencia de los computado-res, concerniente con el diseño de sistemas de computadores inteligentes, esto es, sistemas que exhiben las características que asociamos con la inteligencia en la conducta humana” (Barr y otros, 1982)</td><td>“La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente” (Luger y Stubblefield, 1997)</td></tr>
<tr><td rowspan="2"></td><td colspan="2">Humana</td><td>Racional (ideal)</td></tr>
<tr><td colspan="3">Eficiencia</td></tr>
</table>

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **Objetivos de la Inteligencia Artificial**
  - La Inteligencia Artificial se puede considerar como una disciplina con una doble vertiente de estudio científico y construcción ingenieril. Sus objetivos son:
  - Objetivo científico:
    - Comprende el estudio empírico de los sistemas inteligentes naturales existentes (IA cognitiva) y la comprensión de los principios y mecanismos generales necesarios para el comportamiento inteligente (IA básica).
  - Objetivo ingenieril:
    - Diseñar sistemas capaces de efectuar tareas de forma inteligente independientemente de si contribuyen o no a explicar algún aspecto de la inteligencia (IA aplicada). Este objetivo es de naturaleza tecnológica.

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **Paradigmas en la IA**
  - Simbólico
    - Se basa en la metáfora del cerebro como ordenador. Consiste en la representación simbólica por parte del sistema inteligente de eventos y situaciones externas e internas y en los procesos de manipulación explícita de dichas representaciones mediante reglas.
  - Neuronal
    - Asume que los componentes y estructura del cerebro (neuronas y redes neuronales) son esenciales para el comportamiento inteligente. Se considera por tanto que el nivel adecuado para estudiar la inteligencia es subsimbólico, de bajo nivel y cercano al sustrato neuronal.
  - Basado en Comportamientos
    - Se basa en la idea de que el comportamiento indicativo de la inteligencia emerge de la forma en que las percepciones están ligadas a las acciones. Así se requiere que todo el conocimiento del agente se extraiga de sensores físicos y que sus objetivos se expresen a través de acciones físicas.

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **Sistemas basados en Conocimiento (SBC)**
  - Se encuentran dentro del Paradigma Simbólico
  - Son programas que resuelven problemas usando un
  - determinado dominio de conocimiento
- **Sistemas expertos (SE)**
  - Emulan la capacidad de razonamiento y de decisión de un experto humano
- **Relación entre ambos**
  - Los SBC utilizan conocimiento no necesariamente experto mientras que los SE se caracterizan por usar conocimiento extraído de un experto.
  - Los SE son parte de los SBC.

Ingeniería del Conocimiento

Introducción a la Ingeniería del Conocimiento

- **Ingeniería del Conocimiento**
  - La Ingeniería del Conocimiento es el proceso de diseñar y hacer operativos los Sistemas Basados en Conocimiento.
  - Desde el punto de vista científico:
    - La Ingeniería del Conocimiento se define como el subcampo de la Inteligencia Artificial concerniente a la adquisición, representación y aplicación de conocimientos.
  - Desde el punto de vista ingenieril:
    - Es un disciplina de la Ingeniería por la cual es conocimiento se integra dentro de un sistema de computador para resolver problemas complejos que normalmente requieren un alto nivel de experiencia humana.

Ingeniería del Conocimiento

Sistemas Expertos (1)

- **Definiciones de SE:**
  - Del libro "Sistemas Expertos: Principios y Programación" de J. Giarratano y G. Riley:
    - "Un sistema computacional que emula la capacidad de toma de decisiones de un experto humano en un dominio restringido"
  - De Edward Feigenbaum (Creador del primer SE)
    - "Un programa de ordenador inteligente que utiliza el conocimiento y los procesos de inferencia para resolver problemas que son lo suficientemente complicados como para requerir de forma significativa las aptitudes de un experto humano para su solución"

Ingeniería del Conocimiento

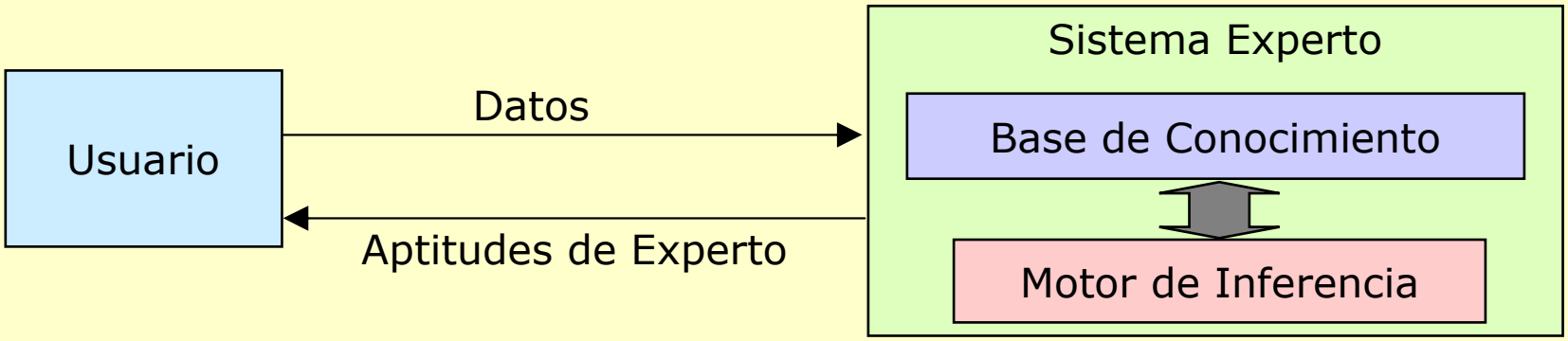
Sistemas Expertos (2)

- **Características generales de los SE**
  - Se basan en el conocimiento representado internamente para la realización de tareas.
  - Utilizan métodos de razonamiento para obtener nuevo conocimiento.
  - Generalmente están restringidos a resolver problemas en un dominio específico
  - Algunos sistemas intentan capturar el conocimiento general de "sentido común":
    - General Problem Solver (Newell, Shaw, Simon)
    - Cyc (Lenat)

Ingeniería del Conocimiento

Sistemas Expertos (3)

- **Esquema básico del funcionamiento de un SE**
  - El usuario:
    - Aporta los datos que describen el problema.
    - Recibe la solución que proporcionaría un experto.



- **Componentes principales de un SE**
  - Base de conocimiento
    - Contiene la información esencial acerca del dominio del problema
    - Representada generalmente como hechos y reglas
  - Motor de inferencia
    - Mecanismo para obtener nuevo conocimiento a partir de la base de conocimiento y la información proporcionada por el usuario.
    - Generalmente basado en la utilización de reglas

Ingeniería del Conocimiento

Sistemas Expertos (4)

- **Etapas principales en el desarrollo de un SE**
  - Identificación del Problema
    - Requerimientos, factibilidad
  - Adquisición del Conocimiento
    - Obtener el conocimiento de un experto
  - Representación del Conocimiento
    - Elegir un formalismo de representación del conocimiento
  - Implementación del SE
    - Seleccionar las herramientas y construir el sistema
  - Verificación y validación
    - Comprobación de que el sistema funciona correctamente y cumple los requerimientos
  - Estas etapas se suelen repetir de forma iterativa.

Ingeniería del Conocimiento

Sistemas Expertos (5)

- **Aspectos tecnológicos de la construcción de SE**
  - El desarrollo de los SE se ha visto influenciado fuertemente por las ciencias cognitivas y las matemáticas.
    - Al estudiar la forma en que los humanos resuelven los problemas
    - Al basarse en modelos formales como la lógica
  - Uno de los mecanismos de representación del conocimiento más extendido se basa en reglas de producción (Reglas del tipo `SI` … `ENTONCES` …)
    - Proporcionan un modelo de razonamiento similar al humano
    - Puede manipularse por los ordenadores
    - Nivel de descripción adecuado
      - Los "pedazos" de conocimiento son manejables tanto para el ordenador como para los humanos.

Ingeniería del Conocimiento

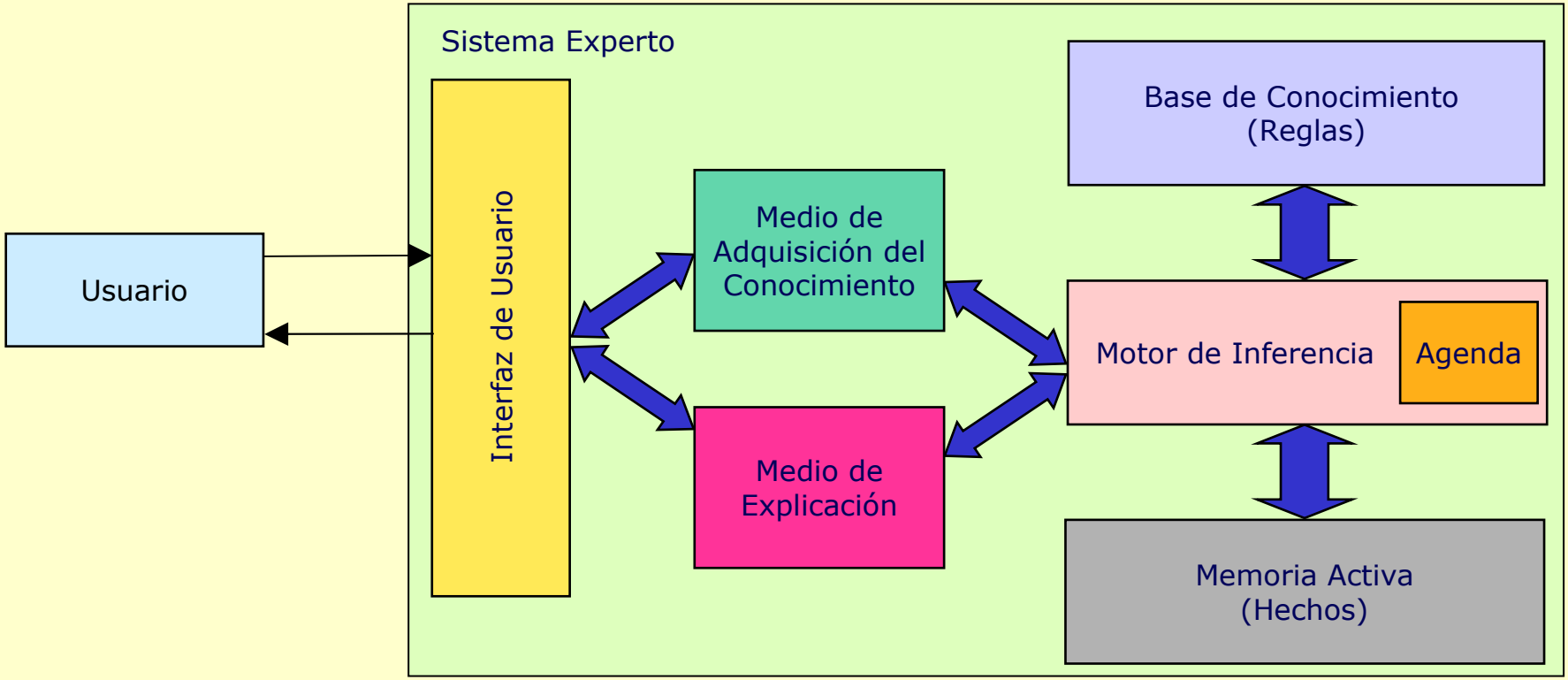
Sistemas Expertos (6)

- **Herramientas para la construcción de SE**
  - Lenguajes de SE
    - Son lenguajes de alto nivel diseñados específicamente para la representación del conocimiento y razonamiento
    - Ejemplos: SAIL, KRL, KQML
  - Shells
    - Un entorno o herramienta de desarrollo de SE donde el usuario proporciona la base de conocimiento
    - Ejemplos: CLIPS, SOAR, OPS5

Ingeniería del Conocimiento

Sistemas Expertos (7)

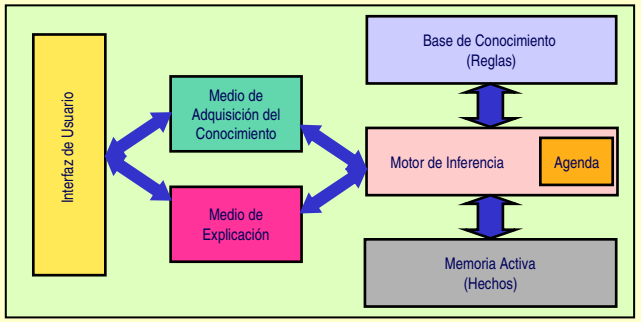
- **Estructura de un SE basado en reglas**



Ingeniería del Conocimiento

Sistemas Expertos (8)

- **Componentes de un SE**



  - Interfaz de usuario:
    - Mecanismo que permite la comunicación entre el usuario y el SE
  - Medio de explicación:
    - Explica al usuario el razonamiento del sistema
  - Memoria activa:
    - Base de datos global con los hechos usados por las reglas
  - Mecanismo de inferencia:
    - Hace inferencias al decidir que reglas satisfacen los hechos, da prioridad a las reglas satisfechas y ejecuta al regla de mayor prioridad
  - Agenda:
    - Lista de reglas con prioridades asignadas que satisfacen los hechos
  - Medio de adquisición de conocimiento
    - Vía para que el usuario introduzca conocimientos en el sistema.

Ingeniería del Conocimiento

Sistemas Expertos (9)

- **SE basado en Reglas**
  - Las reglas son del tipo `SI … ENTONCES …`
    - La parte del `SI` recibe el nombre de antecedente, parte condicional, patrón o lado izquierdo.
    - La parte del `ENTONCES` recibe el nombre de consecuente, o lado derecho.
- **Ejemplos de Reglas**

Reglas `SI … ENTONCES`

```
Regla: Luz_Roja
  SI          la luz está en rojo
  ENTONCES    parar
Regla: Luz_Verde
  SI          la luz está en verde
  ENTONCES    seguir
```

Antecente (parte derecha)

Consecuente (parte izquierda)

Ingeniería del Conocimiento

Sistemas Expertos (10)

- **Ejemplo de regla de un SE real (MYCIN)**

Formato legible por un humano

```
SI          la cepa del organismo es gram negativo
Y           la morfología del organismo es de bastón
Y           the aerobiología del organismo es gram anaeróbico
ENTONCES    existe una evidencia fuertemente sugestiva (0.8)
            de que la clase del organismo es enterobacteriana
```

Formato MYCIN

```
IF      (AND (SAME CNTEXT GRAM GRAMNEG)
             (SAME CNTEXT MORPH ROD)
             (SAME CNTEXT AIR AEROBIC)
THEN (CONCLUDE CNTEXT CLASS ENTEROBACTERIACEAE
        TALLY .8)
```

Ingeniería del Conocimiento

Sistemas Expertos (11)

- **Funcionamiento de un SE basado en Reglas**
  - El Motor de Inferencia controla el comportamiento del sistema
  - Funciona en un ciclo de reconocimiento y actuación:
    - **Reconocimiento**:
      - Determina aquellas reglas para las que se cumplen sus antecedentes.
        - » Para ello la parte izquierda debe "emparejarse" con un hecho en la memoria activa.
      - Coloca las reglas que se cumplen en la agenda.
      - Elige la regla de mayor prioridad para activarla
    - **Actuación:**
      - Ejecuta las acciones asociadas con la regla que se activa
        - » Una regla que se activa puede generar nuevos hechos mediante su lado derecho.
        - » Por tanto, la activación de una regla puede generar la activación de otras reglas.
      - Elimina la regla que se activa de la agenda
  - El ciclo termina cuando no hay más reglas en la agenda o cuando se encuentra una orden de parar.

Ingeniería del Conocimiento

Sistemas Expertos (12)

- **Encadenamiento hacia delante y hacia atrás**
  - Son métodos diferentes de activación de reglas
    - Encadenamiento hacia delante (guiado por los datos)
      - Se razona desde los hechos a la conclusión
      - En cuanto se dispone de hechos se les utiliza para emparejarlos con los antecedentes de las reglas
      - Se utilizan generalmente en sistemas expertos de tiempo real para seguimiento y control de procesos
      - Ejemplos: CLIPS, OPS5.
    - Encadenamiento hacia atrás (guiado por conclusiones)
      - Se parte de una conclusión (que actúa como hipótesis) y se busca si hay hechos que la soporten. Se utilizan generalmente en sistemas de diagnóstico.
      - Ejemplos: EMYCIN

Introducción a la Programación en CLIPS

Ingeniería del Conocimiento

Introducción a la Programación en CLIPS

- **Contenidos**
  - Motivación
  - Objetivos
  - Repaso de Sistemas Expertos
  - Introducción a CLIPS
  - Comandos básicos
  - Introducción a los Hechos y Reglas
  - Variables y Emparejamiento de Patrones
  - Operadores Matemáticos y Lógicos
  - Plantillas y Casillas
  - Entrada y Salida
  - Resumen

Introducción a la Programación en CLIPS

Motivación

- **CLIPS es una herramienta para el desarrollo de Sistemas Expertos basados en reglas.**
- **Ilustra muchos de los conceptos y métodos generales usados en otras herramientas para el desarrollo de SE**
- **Permite la representación del conocimiento y su uso para la resolución de problemas**
- **Otras ventajas de CLIPS:**
  - Puede ejecutarse en muchas plataformas (UNIX, Linux, Windows, MacOS)
  - Es de dominio público
  - Está bien documentado

Introducción a la Programación en CLIPS

Objetivos

- **Familiarizarse con los conceptos y métodos más importantes de las herramientas de desarrollo de SE basados en reglas.**
  - Hechos, reglas, variables, emparejamiento de patrones, motor de inferencia, agenda, memoria activa.
- **Comprender los elementos básicos en el desarrollo de un sistema experto.**
  - Representación del conocimiento.
  - Razonamiento
- **Ser capaz de aplicar técnicas basadas en reglas para ejemplos simples**
- **Poder evaluar la adecuación de un SE para tareas específicas que requieran conocimiento.**

Introducción a la Programación en CLIPS

Material

- **Software**
  - CLIPS se descarga de la dirección de Internet
    `http://www.ghg.net/clips/CLIPS.html`
- **Libros**
  - J. C. Giarretano y G. Riley, *"Sistemas Expertos. Principios y Programación",* (3ª edición). International Thompson Editores (2001).
- **Apuntes**
  - Español:
    - L.D. Hernández Molinero, *Tutorial de CLIPS*
      `http://intelec.dif.um.es/~gdia/Asignaturas/V33/Material-adicional/Tutorial-CLIPS.pdf`
  - Inglés:
    - J. C. Giarretano, *Clips User Guide*
      `http://www.ghg.net/clips/download/documentation/usrguide.pdf`

Introducción a la Programación en CLIPS

Introducción (1)

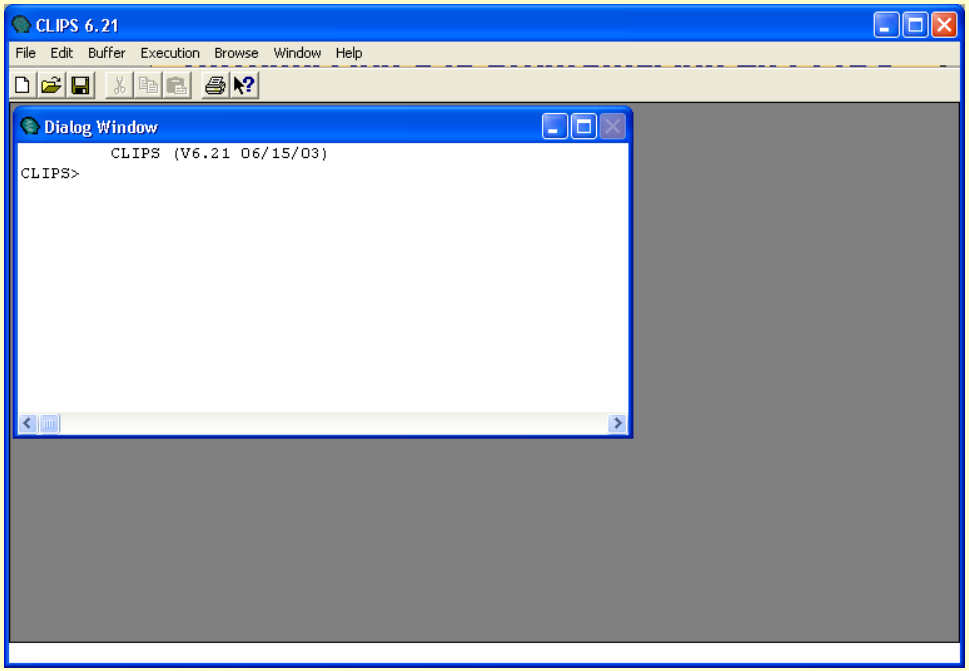
- **CLIPS es la abreviatura de:**
  - C Language Implementation Production System
- **Desarrollado por la NASA a mediados de los 80.**
- **Componentes de CLIPS:**
  - Lenguaje basado en reglas que:
    - Permite definir una lista de hechos
    - Permite crear un conjunto de reglas
    - Proporciona un motor de inferencia de encadenamiento hacia delante que obtiene una solución aplicando las reglas a los hechos iniciales.
  - Lenguaje basado en objetos (COOL)

Introducción a la Programación en CLIPS

Introducción (2)

- **Para comenzar CLIPS en Windows**
  - Hacer click en el icono del programa 

  - Se abrirá una ventana con una ventana de diálogo que actúa como interfaz de usuario



Introducción a la Programación en CLIPS

Comandos básicos de CLIPS

- **Todos los comandos se escriben entre paréntesis.**
- **Algunos comandos pueden ejecutarse desde los menús**
  - `(exit)` salir de CLIPS
  - `(clear)` borra de CLIPS todos los hechos, reglas y definiciones. Equivalente a cerrar CLIPS y abrirlo de nuevo
  - `(reset)` pone el sistema en su estado inicial. (Borra todos los hechos; coloca un hecho inicial `(initial_fact)` así como todos los que el usuario defina por defecto. Debe de efectuarse antes de ejecutar cualquier programa
  - `(run)` ejecuta el programa cargado en CLIPS
  - `(load "nombrefichero.clp")` Carga un programa CLIPS a partir del fichero `nombrefichero.clp`

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (1)

- **Hecho**
  - Es un elemento de información elemental
  - Se almacena en la llamada lista de hechos.
  - Cada hecho en la lista de hechos tiene un identificador que indica su índice en la lista
  - Hay dos tipos de hechos:
    - ordenados
    - no ordenados.
- **Regla**
  - Sirven para representar el conocimiento del experto
  - En general, una regla se expresa de la forma `SI` las condiciones son ciertas `ENTONCES` haz alguna acción.

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (2)

- **Hechos ordenados**
  - Son los formados por varios símbolos entre paréntesis separados por espacios.
  - Se pueden utilizar para representar:
    - Un valor: `(lunes)`
      - "hoy es lunes"
    - Una pareja atributo-valor: `(color verde)`
      - "el color es verde"
    - Una tripleta atributo-objeto-valor: `(hijo_de Luis Daniel)`
      - "Luis es hijo de Daniel"
  - En los hechos ordenados la posición es importante. No es lo mismo
    - `(hijo_de Luis Daniel) ; Luis es hijo de Daniel`
    - `(hijo_de Daniel Luis) ; Daniel es hijo de Luis`
  - Los comentarios en CLIPS se escriben con punto y coma

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (3)

- **Operaciones con Hechos**
  - Agregar un hecho
    - Los hechos se agregan en CLIPS con el comando `(assert)`
    - Ejemplo:

      ```
      CLIPS> (assert (color verde))
      <Fact-0>
      ```

    - CLIPS responde a la agregación de un hecho con `<Fact-xx>` donde `xx` es el índice numérico asignado a ese hecho
    - Se pueden agregar varios hechos simultáneamente haciendo por ejemplo:

      ```
      (assert (sabor dulce)(sabor salado)(sabor amargo) )
      ```

  - Examinar los hechos
    - Para examinar todos los hechos en memoria se utiliza el comando `(facts)`
    - Ejemplo:

      ```
      CLIPS> (facts)
      f-0 (color verde)
      For a total of 1 fact
      ```

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (4)

- **Operaciones con Hechos**
  - Eliminar un hecho
    - Los hechos se borran en CLIPS con el comando `(retract)`
    - Ejemplo:

      ```
      CLIPS> (retract 0)
      ```

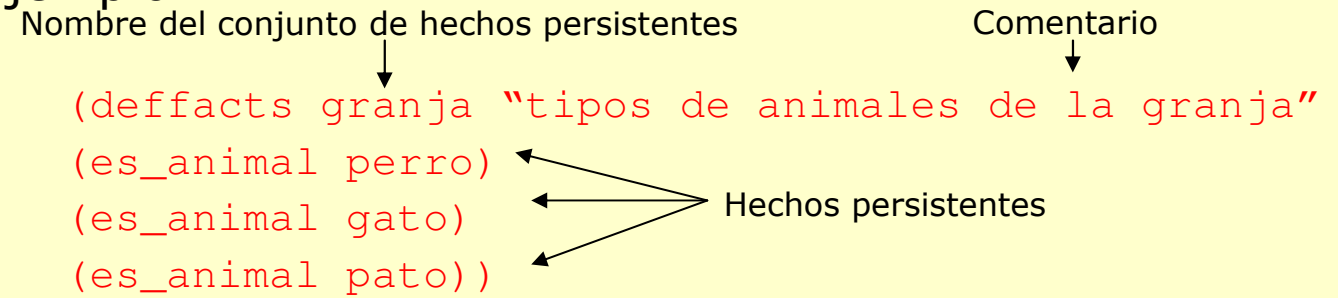
    - Nótese que para borrar un hecho es necesario conocer su índice
    - Puede eliminarse todos los hechos mediante `(retract *)`
- **Cuestiones prácticas**
  - Cuando se pretenden insertar varios hechos es mejor crear una ventana separada (con File, New) y salvar el resultado como fichero de tipo “batch”.
  - Para cargar el fichero se utiliza la orden:

    ```
    (batch “nombrefichero.bat”)
    ```

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (5)

- **Definición de hechos persistentes**
  - Se puede especificar una lista de hechos que no se pierden al reiniciar CLIPS con el comando `(reset)` para ello se utiliza el constructor `(deffacts)`
  - Ejemplo:

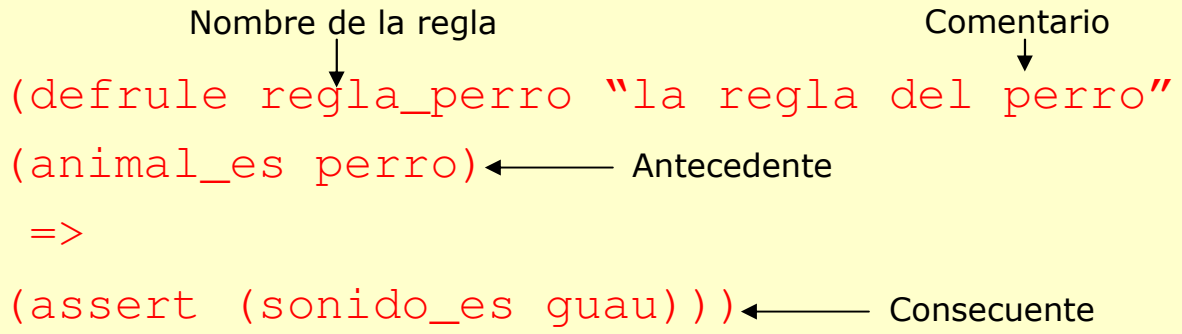


  - Para listar todos los deffacts se utiliza `(list-deffacts)` y para visualizar un deffact específico `(pp-deffacts nombre)` donde `nombre` el nombre del deffact
  - Para eliminar un deffact se utiliza `(undeffacts nombre)` donde `nombre` el nombre del deffact
- **Seguimiento de los hechos en la memoria activa**
  - Para comprobar las inserciones y eliminaciones de hechos de la memoria activa se utiliza el comando `(watch facts).`

Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (6)

- **Reglas**
  - En general, una regla se expresa de la forma:
    - `SI` las condiciones son ciertas `ENTONCES` haz alguna acción.
  - Una regla se compone de antecedente (o parte izquierda de la regla) y consecuente (o parte derecha).
    - Antecedente: es el conjunto de condiciones que deben cumplirse para que la regla se active.
    - Consecuente: es el conjunto de acciones a realizar cuando se activa la regla.
  - Ejemplo:

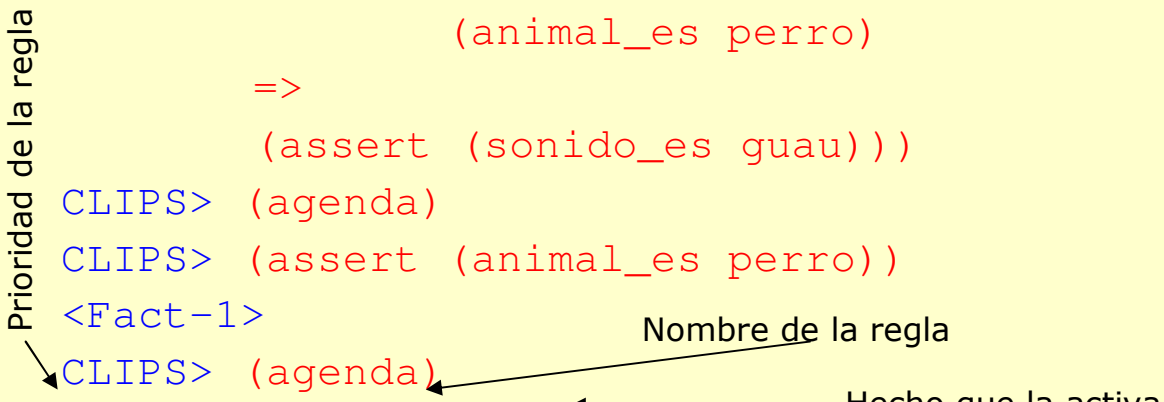


Introducción a la Programación en CLIPS

Introducción a los Hechos y Reglas (7)

- **Examinar las reglas**
  - Para examinar todos las reglas en memoria se utiliza el comando `(rules)`
  - Para visualizar una regla específica se utiliza `(ppdefrule nombre)` donde `nombre` el nombre de la regla
- **La Agenda**
  - Es el conjunto de reglas que pueden activarse (se cumplen sus antecedentes)
  - Ejemplo:

```
CLIPS> (defrule regla_perro "la regla del perro"
                (animal_es perro)
        =>
        (assert (sonido_es guau)))
CLIPS> (agenda)
CLIPS> (assert (animal_es perro))
<Fact-1>
CLIPS> (agenda)
0  regla_perro :f-1
For a total of 1 activation
```



Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (1)

- **Variables**
  - Una variable almacena información dinámica, esto es, los valores que toma pueden cambiar (a diferencia de un hecho que siempre es estático).
  - Permiten la escritura de reglas más generales y complejas.
- **Nombre de una variable**
  - Se compone del símbolo ? y de uno o más caracteres.
  - Ejemplos:
    - ?nombre ?color
- **Uso de las variables**
  - Uno de los usos más comunes de las variables es el de tomar un valor en el antecedente de una regla y usarlo en el consecuente.
  - El proceso de tomar valores en el antecedente se llama emparejamiento de patrones
  - Las reglas se activan para todos los valores de las variables que cumplen el antecedente

Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (2)

- **Ejemplo simple del uso de variables y reglas**

```
CLIPS> (reset)
CLIPS> (defrule abuelo "la regla del abuelo"
                (es_abuelo ?nombre)
        =>
         (assert (es_hombre ?nombre)))
CLIPS> (assert (es_abuelo juan))
CLIPS> <Fact-1>
CLIPS> (assert (es_abuelo luis))
CLIPS> <Fact-2>
CLIPS> (run)
CLIPS> (facts)
f-0 (initial-fact)
f-1 (es_abuelo juan)
f-2 (es_abuelo luis)
f-3 (es_hombre luis)
f-4 (es_hombre juan)
For a total of 5 facts
```

Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (3)

- **Ejemplos complejos del uso de variables y reglas**
  - Cuando hay varias condiciones en los antecedentes deben cumplirse todas las condiciones para poder aplicar la regla
    - ```
      (defrule hacer-tortilla
      (hay aceite)(hay huevos)(hay sal)
      =>
      (assert (hacer tortilla)))
      ```
  - Cuando hay varias apariciones de la misma variable en el antecedente, para activar la regla la variable debe tomar un valor único que haga cumplir todos los hechos.
    - ```
      (defrule abuelo "la regla del parking"
       (es_alumno ?nombre ?facultad) (tiene ?nombre coche)
        =>
       (assert (puede_aparcar_en ?nombre ?facultad)))
      ```
    - ```
      (defrule ave
      (animal ?nombre) (sangre-caliente ?nombre) (pone-huevos
      ?nombre)
       =>
       (assert (es_ave ?nombre)))
      ```
      ```

Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (3)

- **Eliminación de Hechos con Reglas y Variables**
  - Recordamos:
    - Para eliminar un hecho se necesita conocer su índice en la lista de hechos
    - Una vez conocido el `indice` se hace `(retract indice)`
  - Es posible conocer el índice de un hecho es necesario asignarlo a una variable con <-
  - Ejemplos:

```
(defrule elimina-huevos
?indice_hecho<-(hay huevos)
=>
(retract ?indice_hecho))
```

```
(defrule elimina-ave
?indice_hecho<-(es_ave ?nombre)
=>
(retract ?indice_hecho))
```

Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (4)

- **Comodines**
  - Complementan las variables y permiten la escritura de reglas más generales y complejas
    - Comodín `?` :Toma el valor de un campo dentro de un hecho
    - Comodín `$?` Toma el valor de cero o varios campos en un hecho
- **Ejemplo**

```
(miembros-de beatles lennon mccartney harrison starr)
(defrule musico
(miembro-de ?grupo $? ?miembro $?) => assert(musico ?miembro))
```

  - Emparejamientos:
    - `?grupo=beatles $?=vacio ?miembro=lennon $?= mccartney harrison starr`
    - `?grupo=beatles $?=lennon ?miembro=mccartney $?= harrison starr`
    - `?grupo=beatles $?=lennon mccartney ?miembro= harrison $?=starr`
    - `?grupo=beatles $?=lennon mccartney harrison ?miembro=starr $?=vacio`
  - Resultado:
    - `(musico lennon) (musico mccartney) (musico harrison) (musico starr)`

Introducción a la Programación en CLIPS

Variables y Emparejamiento de Patrones (5)

- **Comodines y Variables**
  - Los comodines pueden combinarse con variables para almacenar los resultados

- **Ejemplo**

```
(miembros-de beatles lennon mccartney harrison starr)
(defrule musico
(miembro-de ?grupo $?miembros)
⇒
assert(ricos $?miembros))
```

  - Emparejamientos:

```
?grupo=beatles $?miembros=lennon mccartney harrison starr
```

  - Resultado:

```
(ricos lennon mccartney harrison starr)
```

Introducción a la Programación en CLIPS

Operadores Matemáticos y Lógicos (1)

- **Los operadores y funciones matemáticas permiten realizar los cálculos necesarios en CLIPS**
- **Se representan con notación "prefija"**
  - Ejemplo:
    - 2+3 se escribe como (+ 2 3)
    - 2*(3+4) se escribe como (* 2 (+ 3 4))
- **Operaciones estándar**
  - + (suma), - (resta), *(producto), / (división), div (división entera), max (máximo), min (mínimo), abs (v. Absoluto)
- **Otras operaciones**
  - sqrt (raíz cuadrada), ** potencial, exp (exponencial), log (logaritmo neperiano), sin (seno), cos (coseno).
- **Para consultar el conjunto completo de operaciones consultar el manual de CLIPS**

Introducción a la Programación en CLIPS

Operadores Matemáticos y Lógicos (2)

- **En el consecuente de una regla una operación se calcula automáticamente.**
  - Ejemplo

```
CLIPS>(defrule suma (numeros ?x ?y)
       =>
       (printout t "la suma es" (+ ?x ?y) crlf))
CLIPS> (assert (numeros 2 3))
CLIPS> <Fact-0>
CLIPS> (run)
CLIPS> la suma es 5
```

- **En el antecedente de una regla las operaciones matemáticas sólo se calculan cuando siguen el símbolo =**
  - Ejemplo

```
CLIPS>(defrule propuesta-suma
      (numeros ?x ?y)
      (propuesta = (+ ?x ?y))
       =>
      (printout t "suma correcta" crlf))
CLIPS> (assert (numeros 2 3))
CLIPS> <Fact-0>
CLIPS> (assert (propuesta 5))
CLIPS> <Fact-1>
CLIPS> (run)
CLIPS> suma correcta
```

Introducción a la Programación en CLIPS

Operadores Matemáticos y Lógicos (3)

- **La función bind (fijar)**
  - Permite darle un valor a una variable en el consecuente de una regla
  - Resulta útil cuando el mismo valor va a usarse más de una vez
  - Ejemplo:

```
(defrule suma
(numeros ?x ?y)
 =>
(bind ?respuesta (+ ?x ?y))
(assert respuesta ?respuesta)
(printout t "la suma es" ? Respuesta crlf))
```

Introducción a la Programación en CLIPS

Operadores Matemáticos y Lógicos (4)

- **Operadores Lógicos y de Comparación**
  - Se utilizan generalmente en el antecedente de una regla para comprobar si determinadas condiciones son ciertas
- **Operadores principales:**
  - Lógicos
    - and (y), or (o), not (no)
    - Ejemplo:

```
(defrule paraguas
(or  (tiempo llueve)  (tiempo nieva))
 =>
(printout t "llevar paraguas" crlf))
```

  - Comparación
    - eq (igual), neq (no igual), =(igual numérico), !=(no igual numérico), > (mayor), >= (mayor o igual), < (menor), <= (menor o igual)
    - Para realizar comparaciones en el antecedente de una regla debe utilizarse el elemento condicional test
    - Ejemplo:

```
(defrule padre
(es_hombre ?nombre) (tiene_hijos ?nombre ?cantidad) (test (>
   ?cantidad 0))
 =>
(printout t "es padre" ?nombre crlf))
```

Introducción a la Programación en CLIPS

Hechos no ordenados (1)

- **Hasta ahora se han utilizado los hechos ordenados.**
  - Generalmente se han empleado hechos del tipo (atributo valor) o (atributo objeto valor)
  - El orden de los campos dentro del hecho es importante.
- **En aplicaciones complejas esta representación tiene problemas**
  - ¿Como representar objetos con varios atributos?: Juan tiene 20 años, pesa 80 kg, mide 188 cm y su presión es de 130 y 80 y está enfermo
    - Opción 1
      - (nombre juan) (edad juan 20) (peso juan 80) (altura juan 188) (presion juan 130 80) (enfermo juan si)
        - » El número de hechos es muy grande
    - Opción 2
      - (juan 20 80 188 130 80 si)
        - » Programas difíciles de leer, escribir y entender.
- **Con los hechos no ordenados cada campo recibe un nombre, lo único que se debe hacer es recordar el nombre del campo**

Introducción a la Programación en CLIPS

Hechos no ordenados (2)

- **Plantillas (templates) y casillas (slots)**
  - Sirven para trabajar con objetos que tienen varios atributos
  - Se crean utilizando el constructor deftemplate que especifica la forma que toman los hechos
    - Ejemplo:

```
(deftemplate datos_personales
(slot nombre)
(slot edad)
(slot peso)
(slot altura)
(multislot presion_arterial)
(slot enfermo)
```

  - Los hechos se siguen introduciendo con assert pero ahora no importa el orden ni es necesario especificar todos los atributos
    - Ejemplos:

```
(assert (datos personales (nombre juan) (edad 20) (peso 80) (altura 188) (presion_arterial 130 80) (enfermo si))
(assert (datos personales (peso 80) (nombre luis)))
```

Introducción a la Programación en CLIPS

Hechos no ordenados (3)

- **Hechos no ordenados y reglas**
  - Cuando de utiliza un hecho no ordenado en una regla solo es necesario especificar los atributos que se quieren emparejar.
  - Ejemplo:
    - ```
      (defrule persona_mayor "personas de mas de 60 años"
      (datos_personales (nombre ?n) (edad ?e))
      (test (> ?e 60))
      =>
      (printout t "la persona " ?n " es mayor"))
      ```
  - Cuando se utilizan multicasillas es necesario utilizar el comodín $?
    - ```
      (defrule muestra_presion
      (datos_personales (nombre ?n)
      (presion_arterial $?p))
       =>
      (printout t "la presión de " ?n " es " $?p))
      ```

Introducción a la Programación en CLIPS

Hechos no ordenados (4)

- **Tipos y valores por defecto**
  - El constructor deftemplate permite restringir los valores de los atributos utilizando (type).
  - Los tipos más usados son:
    - SYMBOL: un símbolo
    - STRING: un texto
    - INTEGER: un número entero
    - FLOAT: un número real
  - También se pueden utilizar valores por defecto con (default) especificando sus valores o utilizando los valores por defecto de CLIPS con ?DERIVE
  - Ejemplo:
    -
```
(deftemplate datos_personales
(slot nombre (type STRING) (default ?DERIVE))
(slot edad (type INTEGER) (default 30))
(slot peso (type FLOAT) (default 70))
(slot altura (type FLOAT) (default 70))
(multislot presion_arterial)
(slot enfermo (type SYMBOL) (default si)))
```

Introducción a la Programación en CLIPS

Hechos no ordenados (5)

- **Modificación de casillas**
  - Además de la utilización de retract y assert para la modificación de hechos no ordenados se puede emplear una forma sencilla para la modificación de casillas con el comando modify
  - Ejemplo:
    - ```
      (defrule  "los enfermos se curaron"
      ?indice<-(datos_personales (enfermo si))
       =>
      (modify ?indice (enfermo no)))
      ```
- **Hechos no ordenados por defecto**
  - Se utiliza deffacts (igual que para los hechos ordenados)
    - ```
      (deffacts  mis_datos
      (datos_personales (nombre juan) (edad 20)))
      ```

Introducción a la Programación en CLIPS

Entrada y Salida

- **Imprimir información**
  - Se utiliza: `(printout dispositivo elementos_a_imprimir)`
    - El valor de `dispositivo` para el monitor es `t`
    - Los elementos a imprimir se separan por espacios.
    - Para separar lineas se pone `crlf`
    - Ejemplo:

```
(defrule busca_persona
(persona ?nombre1 ?ojos1 ?pelo1) =>
(printout t ?nombre1 " tiene los ojos " ?ojos1 "
 y el pelo " ?pelo1 crlf)
```

- **Leer información**
  - Se utiliza: `(read dispositivo)`
    - Si el dispositivo es el teclado, la lectura se hace con `(read)`
    - Ejemplo:

```
(defrule aguas_residuales
(tipo agua residual) =>
(printout t "¿Está el agua turbia? (si/no)")
(assert(turbia agua (read))
```

- **Para Lectura/Escritura de ficheros consultar el manual**