



# PRÁCTICAS DE MATEMÁTICAS 1

## 2018-2019



- !5 octubre-comienzo Fase 1.
- CONSEJOS DE EJECUCIÓN.

- Link a Youtube de las clases del Prof. Francisco Gallego (2017-18).  
<https://www.youtube.com/watch?v=7I6X5NZd2kw&list=PLmxqg54iaXrgiF20LM2JfetrnZF87UMBD>





## FASE 1

[logica.i3a.ua.es](http://logica.i3a.ua.es)

- FASE 1: desde **15** hasta **28 octubre**
- Nº mapas: **3** de diferente dificultad.
- Mapa resuelto  $\geq 75\%$  >> permite descargar siguiente mapa.
- Hasta 100% se permiten **3 intentos** más >> la nota **nunca baja**.

FASE	DIFICULTAD				
	D1	D2	D3	D4	D5
0	0,100				
1	0,350	0,450	0,500	0,550	0,650
2	0,525	0,675	0,750	0,825	0,975
3	1,225	1,575	1,750	1,925	2,228
4		2,025	2,250	2,475	

- En Materiales : MapasExtrasFase 1.zip
- mapa12.pl mapa22.pl mapa 32.pl mapa42.pl





## Rutina para resolver mapas

**Editar** fichero solución (extensión .pl):

```
$ gedit solucion.pl &
```

**Ejecutar** solución de un mapa

```
$ ./plman mapa.pl solucion.pl
```

**Escribir** en la 1ª línea del fichero solución.pl

```
:- use_module('pl-man-game/main').
```

**Resto** de líneas

**Código Prolog con acciones para Plman**





## Acciones de Plman

**Acciones:** **do(ACCION).**

ACCION = { move(D1), get(D1), drop(D1), use(D1) }

D1 = {up, down, left, right}

Plman No se mueve: do(move(none)).

**Sensor de visión:** **see(normal, D2, OBJ)**

D2 = { right, left, down, up, here, down-right, down-left, up-right, up-left }

OBJ: Cualquier caracter en el mapa

Las letras mayúsculas y los símbolos especiales entre comillas simples

Ej. Regla simple

**do(ACCION(D1)) :- see(normal, D2, OBJ).**





## Consejos para resolver mapas

>> Los mapas de Fase 1 se resuelven con **reglas simples**.

>> Regla simple: sólo **un sensor** de visión en la cláusula

Ej Reglas simples.

`do(move(up)) :- see(normal, up, '.').`





## Consejos para resolver mapas

>> Consigue la mejor solución :

- . Que tenga menor número de movimientos.
- . Menor nº de inferencias (parámetro  $-n$ ).

Para ello al comenzar a escribir el código especifica la estrategia con la que “arrancas” el movimiento que debe hacer Plman para no dejar cocos al pasar.

### **REGLA DE ORO:**

**No te dejes cocos** al pasar por un sitio,  
puede ser **imposible regresar**.





## Consejos para resolver mapas

**>> Elige dirección prioritaria para move: usa la dirección contraria al movimiento.**

Ej mapa\_22.pl:

Solución inicial:

```
do(move(right))    :- see(normal, right, '.').
do(move(up))       :- see(normal, up,   '.').
do(move(down))     :- see(normal, down, '.').
do(move(left))     :- see(normal, left, '.').
```

**>> Para recorrer espacios: escribe reglas que sigan la lógica del movimiento de Plman**

Ej mapa\_22.pl:

Reglas para espacios en blanco:

```
do(move(left))     :- see(normal, left, ' ').
do(move(up))       :- see(normal, up,   ' ').
do(move(down))     :- see(normal, down, ' ').
do(move(right)).
```

## Consejos para resolver mapas

>> Comprueba y controla la regla que entra en acción y escribe “justo” encima de ella la que creas que debe ejecutarse antes.

Ej mapa\_22.pl

*Se debe coger la llave ‘a’ >> al ser esto más prioritario que el coco de la derecha lo pongo como primera regla.*

```
do(get(up))      :- see(normal, up, a).  
do(move(right))  :- see(normal, right, '.').  
.....
```

>> **Cuidado!** con quitar prioridad a otras reglas.

>> Ir a la **prioridad mínima**.

>> El **orden** en que se escriben las reglas es muy importante.





## Consejos para resolver mapas

>> Puedes usar “ver el objeto en diagonal” para moverte

Ej mapa\_22.pl

```
do(move(up))      :- see(normal, up-left, a).
```

>> Puedes usar “ver el objeto” para quedarte quieto

Ej mapa\_22.pl

*Cuando veas la llave ‘a’ no te muevas...:*

```
do(move(none))    :- see(normal, left, a).
```

*Decide dónde añades estas dos reglas :*

```
do(move(up))      :- see(normal, up-left, a).
```

```
do(move(none))    :- see(normal, left, a).
```





Si quieres añadir más predicados en el cuerpo de la regla usa la conjunción (,)

>> **varios predicados see/3**

Ej. `do(move(down)) :- see(normal, down, '.'), see(normal, up, ' ')`.

>> **predicado predefinido para escribir mensajes: writeln/1**

Ej. `do(move(down)) :- see(normal, down, '.'), writeln('veo coco abajo')`.

>> **otros predicados** que ya veremos en fases sucesivas.

Puedes tomar decisiones cuando no veas “algo”

Ej. `do(move(down)) :- not(see(normal,down,'E'))`.





## Siempre debes tener en cuenta cómo “recorre” Prolog tu código solución

- >> De **arriba/abajo** se ejecuta la primera cláusula que sea cierta. (hecho / regla)
- >> **Cuerpo de la regla** : conjunción de condiciones que Plman debe cumplir para que se realice la acción correspondiente indicada en la cabeza.
- >> Se comprueban las condiciones de **izda a dcha** mientras no fracase ninguna.
- >> Si **todas** las condiciones son **ciertas** se ejecuta la acción indicada en la cabeza.





## !Cuidado con el “descuento” de puntos! ESTADÍSTICAS NEGATIVAS

El auto-corrector **reduce la nota** de la solución de un mapa si se incurre en:

- **Colisiones** con paredes u objetos sólidos : - **0,25/u** ptos.
- **Acciones erróneas** / intento de hacer una acción en una regla: - **0,50/u** ptos.
- **Fallos de regla**/ no se puede ejecutar ninguna regla : - **0,75/u** ptos.
- **Intento de acción**: - **0,30/u** ptos.

### MÁXIMO NÚMERO DE MOVIMIENTOS (TIC-TAC)

Fase 0:	<b>50</b>
Fase 1:	<b>350</b>
Fase 2:	<b>500</b>
Fase 3:	<b>750</b>
Fase 4:	<b>1.250</b>
Examinador:	<b>750</b>

Penalización: 0,5 puntos

