**Practica Planificacion**

**Carlos Casar Morejon**

**Victor Gonzalez Prieto**

**Gerard Otero Martín**

**Indice**

[Dominio](#_msyh132m24r)

[Operadores](#_uqvsfmc9b9ql)

[Juegos de pruebas](#_9pa0a36rdm1o)

[Generador de juegos de pruebas](#_s06l27tietsm)

# Dominio

Hemos representado el dominio como una serie de 10 niveles, 15 días y 5 ejercicios, ya que es lo único que necesitamos para representar adecuadamente el dominio de nuestro problema.

Hemos definido 4 tipos de datos: exercise, order, date y level. exercise y order son object, y date y level son order. En exercise tenemos los 5 ejercicios, los 10 niveles son tipo level y los 15 días son tipo date, y estos últimos dos son a la vez tipo order.

Como predicados, tenemos los siguientes:

● (sucesor ?pre ?pos): significa que ?pre va seguido de ?pos en un cierto orden. Es necesario para ordenar los niveles y los días.

● (achieved ?ex ?lvl): significa que el usuario ha conseguido llegar al nivel ?lvl del ejercicio ?ex en algún día anterior al actual.

● (preparador ?pre ?ex): indica que ?pre es el ejercicio preparador de ?ex.

● (predecesor ?pre ?ex): indica que ?pre es el ejercicio predecesor de ?ex.

● (current-day ?day): indica que estamos en el día ?day.

● (done-today ?ex ?lvl): indica que el ejercicio ?ex de nivel ?lvl se ha hecho en el día actual, almacenado en (current-day ?day). Necesario para saber si se puede realizar un ejercicio del cual ?ex es el preparador.

● (just-done ?ex ?lvl): indica que el ejercicio ?ex de nivel ?lvl es el último que se ha hecho en el día actual, almacenado en (current-day ?day). Necesario para saber si se puede realizar un ejercicio del cual ?ex es el predecesor.

● (kicked-off): indica cuando empieza la ejecución del programa, para inicializar el día actual al día 1 y poner el tiempo usado a 0.

● (progress ?ex ?lvl): indica que en el día actual se ha llegado al nivel ?lvl del ejercicio ?ex.

Como funciones, tenemos las siguientes:

● (time-spent): indica el tiempo que se ha usado en el día actual del programa. Siempre deberá ser menor a 90.

● (time ?ex): indica el tiempo que el ejercicio ?ex consume.

# Operadores

Hemos implementado 2 operadores:

● do-exercise (parameters ?ex ?lvl): primero comprueba si el ejercicio ?ex de nivel ?lvl cumple las restricciones impuestas por los ejercicios predecesores y preparadores, a parte de si el nivel actual que tiene el usuario de ese ejercicio ?ex ya ha sido conseguido, y en consecuencia define que ?ex ha sido realizado en dia actual, y es el ejercicio que se acaba de realizar, y niega el ejercicio que se había acabado de realizar justo antes de este. Es necesario ya que sin él no podríamos definir si se ha realizado un ejercicio o no.

● Change-day (parameters ?today ?tomorrow): primero comprueba que ?today es el día actual, y comprueba que ?tomorrow es el sucesor de ?today, y en consecuencia cambia el día actual a ?tomorrow, pone como achieved todos los ejercicios realizados el día anterior y niega todos los ejercicios realizados en ?today, incluido el último de ellos. Es necesario ya que sin él no podríamos cambiar de día y todos los ejercicios estarían forzados a ser realizados el mismo día, y tampoco podríamos marcar como achieved los ejercicios ya realizados, con lo que no podríamos llegar al estado objetivo.

● kick-off (parameters ?first): “inicia” la ejecución del programa, es decir, comprueba que estemos en el principio de la ejecución mirando si estamos en el día uno y kicked-off está en falso, y pone kicked-off a cierto, el current-day al día 1 y time-spent a 0.

# Juegos de pruebas

En este apartado explicaremos los diferentes juegos de prueba que hemos hecho para testear nuestra práctica y el generador de juego de pruebas. En los juegos de pruebas manuales, el primer dígito es la extensión que se quiere testear y el segundo dígito el número de prueba.

#### **Problema 01**

Esta es una prueba sencilla cumpliendo las reglas del nivel básico. El ejercicio 7 y 4 tiene un preparador cada uno. El resultado es una solución satisfacible como se espera.

#### **Problema 02**

En este caso se prueba un bucle en los ejercicios para comprobar que el resultado sea insatisfacible.

#### **Problema 11**

Igual que la prueba 01 pero esta vez para la extensión 1, añadimos más de un preparador a más de un ejercicio para comprobar que es satisfactible.

#### **Problema 12**

Como la prueba 02, pero con más de un preparador para cada ejercicio y creando el bucle. El resultado es insatisfacible.

#### **Problema 21**

Primera prueba para testear la extensión 2, en el que añadimos los predecesores. Resultado satisfactible.

#### **Problema 22**

En esta prueba creamos un pequeño bucle, como hicimos con anterioridad, pero en esta ocasión intercalando un predecesor. Resultado insatisfacible.

#### **Problema 23**

Caso más complicado que 21, en que un ejercicio preparador del ejercicio goal tiene un predecesor. Resultado satisfactible.

#### **Problema 24**

En este caso, tenemos un preparador del ejercicio goal, que es predecesor de otro ejercicio preparador del ejercicio goal. Resultado satisfactible.

#### **Problema 31**

Ahora pasamos a la extensión 3, y en esta ocasión forzamos a que los dos ejercicios a cumplir el goal, no quepan en un mismo dia. En este caso es satisfactible ya que entre ambos ocupan los 15 días.

#### **Problema 32**

Igual que el problema 31, pero en esta ocasión insatisfacible, ya que hemos bajado el nivel inicial, para que en 15 dias no tengan tiempo suficiente para alcanzar el objetivo.

#### **Problema 33**

En este problema probamos como en los anteriores con 2 ejercicios que tienen un goal, pero en esta ocasión sí que caben juntos en el mismo día sin superar los 6 ejercicios diarios. Resultado satisfactible.

#### **Problema 34**

Prueba crítica de un ejercicio con 6 preparadores, con lo cual no puede realizarse en ningun dia ya que necesita 7 ejercicios. Resultado insatisfacible.

#### **Problema 41**

En esta prueba determinamos una prueba sencilla en que los predecesores y preparadores del goal + tiempo de goal superen los 90 min. Resultado satisfactible.

#### **Problema 42**

Como el anterior, pero esta vez superan el límite diario temporal, por tanto insatisfacible.

# Generador de juegos de pruebas

El generador de juegos de prueba automático, le hace 3 sencillas preguntas, para tener una pequeña guía, y a partir de esos datos crea 10 juegos de pruebas con aleatoriedad. Las preguntas que le hace son las siguientes:

● Nombre del juego de pruebas(para crear el nombre de la carpeta).

● Número máximo de preparadores por ejercicio(puede ser 0).

● Si se quiere que los ejercicios puedan tener predecesores.

● SI se desea que los ejercicios tengan tiempo.

A partir de estos datos genera los 10 juegos de pruebas correspondientes en la carpeta con el nombre indicado.

***nota:*** *debe compilarse con c++11: # g++ -std=c++11 generador.cpp -o generador*