Locker Room: Una Herramienta Para El Aprendizaje de Punteros Basada en La Metáfora de Las Taquillas

Carlos Martín Villanova, Tonghong Li,
Claudio Soriente, Ricardo Jiménez Peris and Marta Patiño Martínez

1. Introducción

Algoritmos y estructuras de datos son las bases de la educación universitaria en informática. Los programas de informática intentan asegurarse que los estudiantes obtengan un conocimiento profundo de los fundamentos de la materia como ordenación y búsqueda así como de los avanzados.

El objetivo de las universidades es encontrar nuevos métodos que ayuden a los estudiantes a convertirse en profesionales. Estos métodos permiten probar nuevas técnicas y experimentar distintas estrategias de impartir la clase. Una de esas técnicas es incluir visualización y animación de algoritmos y estructuras de datos (que serán nombrados como "visualización de algoritmos" o VA de aquí en adelante) en los programas. Estas VAs permiten ver los algoritmos al representar gráficamente los distintos estados y haciendo animaciones de la transición entre los estados. Esto permite experimentar a los estudiantes distintos escenarios y les libera de tener que hacer sus propias representaciones. Las mejores VAs ilustran las estructuras de datos de una forma natural, haciendo una abstracción de las direcciones de memoria y las llamadas a funciones.

Siguiendo estos fundamentos de la excelencia en VAs se ha desarrollado una animación de algoritmos para ilustrar las metáfora de la taquilla.

2. Metáfora de La Taquilla

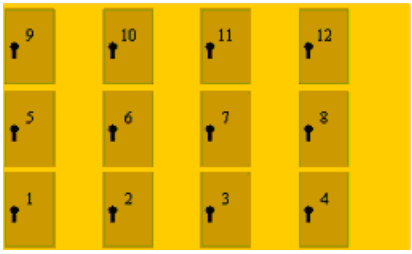
Un método para la enseñanza de la memoria dinámica.

La memoria dinámica es un aspecto crucial en los lenguajes imperativos y por ello se estudia en muchos cursos. A la par, resulta uno de los temas más difíciles de comprender, debido a que los conceptos que se manejan al hablar de memoria dinámica son muy abstractos.

Existen muchas maneras de explicar temas de programación que mejoran la compresión, por ejemplo, la animación de algoritmos. Aparte de esto es aconsejable utilizar un método que es muy común en informática para manejar conceptos abstractos y definidos arbitrariamente: las metáforas. Ha habido muchas metáforas que han servido para explicar conceptos en este campo: el escritorio, la memoria de los ordenadores,...Se ha de tener en cuenta que la palabra “metáfora” en este contexto se usa como cualquier imagen que pueda representar a una cosa y que se utiliza en lugar de algo para explicarlo mejor. El uso de analogías al enseñar conceptos abstractos es a menudo un elemento clave para facilitar el aprendizaje, como en el caso de la metáfora de los esquimales e iglúes de Ben-Ari para explicar la programación concurrente.

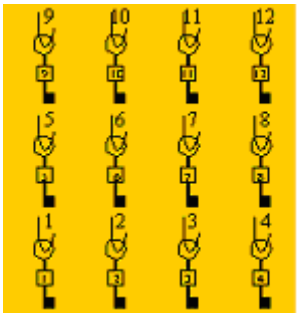
Siguiendo este esquema se ha desarrollado una metáfora para enseñar los conceptos básicos de la memoria dinámica, una vez que éstos están bien entendidos se puede pasar al método tradicional de enseñanza. El escenario escogido para la metáfora tiene que ser uno que todo el mundo comprenda fácilmente, en este caso es una consigna de equipaje. Una consigna es una habitación donde existen numerosas taquillas y cada una de las cuales se puede abrir con su llave correspondiente.

Cada llave tiene un número que corresponde con el número de taquilla que abre. La memoria dinámica es muy parecida a una consigna, las celdas de memoria son las taquillas. De la misma forma que las taquillas se pueden abrir con la llave, se puede acceder a una celda con su dirección de memoria correspondiente. Nótese que ni las llaves ni las direcciones de memoria se pueden modificar.



En la metáfora, los llaveros se usan para sujetar las llaves. Cada llavero puede tener como mucho una llave, es decir cero o una. Los llaveros y las llaves representan las variables de tipo puntero y los valores de direcciones de memoria en los lenguajes de programación imperativa. Los llaveros representan las variables de tipo puntero que pueden tener una llave de taquilla, una dirección de memoria. Un llavero que no tiene ninguna llave representa el valor NIL de una variable. Dos llaveros pueden tener una copia de la misma llave, en este caso los dos pueden abrir la misma taquilla. Esta situación se corresponde al hecho de que dos variables pueden apuntar a la misma celda de memoria, esto es, pueden tener la misma dirección de memoria.

Las operaciones simples de memoria pueden explicarse gráficamente con la metáfora. La comparación de punteros (llaveros) es clara, solo será verdad si los dos llaveros están vacíos o las llaves que contienen abren la misma taquilla. La asignación de punteros se corresponde con el siguiente procedimiento. Si el llavero origen está vacío el llavero destino se deja vacío, si no, se hace una copia de la llave del llavero origen, la llave (si es que existe) del llavero destino se desecha y finalmente se pone la copia de la llave en el llavero destino. En la metáfora se puede observar una propiedad muy importante de la asignación, cuando el contenido de un llavero se copia en otro, la llave del destino se pierde sin posibilidad de recuperarla después. Cuando alguien necesita un llavero va a la oficina del taquillero (persona que gestiona la entrega de llaves) y le pide una taquilla. El taquillero entonces coge cualquier llave que esté disponible y se la pone en el llavero, si había antes una llave en él, se desecha. Aunque la manera en la que el taquillero almacena las llaves pueda parecer irrelevante para el usuario, puede ayudar a entender completamente cierta clase de errores. Podemos imaginar que el taquillero tiene una tabla con ganchos numerados [figura de abajo] uno para cada casilla. Si la llave está colgada en el gancho, la taquilla está libre, si no, está asignada.



En los lenguajes de programación es el gestor de memoria dinámica el que se encarga de saber el estado de las celdas. Cuando se necesita una celda se invoca la primitiva *New* (se utilizaran las primitivas de PASCAL para representar lo que se haría en un lenguaje de programación) que devuelve la dirección de una celda de memoria libre. Esta gestión es similar a la que haría el taquillero pero es irrelevante.

Cuando una taquilla no se va a necesitar más, el usuario se lo notifica al taquillero y le da el llavero con la llave. El taquillero hace una copia de la llave y la pone en la tabla colgada del gancho correspondiente para indicar que está libre. Después le da el llavero de vuelta al usuario con la llave todavía en él. Se tiene que tener en cuenta que sorprendentemente la llave sigue en el llavero después de haberse liberado. Esta parte de la metáfora corresponde a la siguiente situación. Cuando una celda de memoria no se necesita debe ser liberada con la primitiva *Dispose* pero esta operación puede dejar el puntero con su valor anterior.

Un resumen de la equivalencia entre la metáfora y la memoria dinámica se puede ver en la tabla siguiente:

Memoria	Consigna
Gestor de memoria	Taquillero
Puntero	Llavero
Celda de memoria	Taquilla
Dirección	Llave de una taquilla
Lista de celdas disponibles	Tabla con los ganchos
k1, k2: Type	k1 y k2 son llaveros
New (k1)	Se pide una taquilla, se obtiene la llave y se pone en el llavero k1
k1^	Se abre la taquilla cuya llave está en el llavero k1 para ver su contenido.
k1 = NIL	Se comprueba si el llavero k1 está vacío
k1 = k2	Se comprueba si k1 y k2 están los dos vacíos o tienen llaves que abren la misma taquilla.
k1 := NIL	Se vacía el llavero k1.
k1 := k2	Si k2 está vacío entonces se vacía k1, y si no, se hace una copia de la llave de k2 y se pone en k1.
Dispose (k1)	Se libera la taquilla cuya llave está en el llavero k1 dejando una copia en la tabla.

Existen muchas situaciones en las que la metáfora supone una ayuda al ilustrar gráficamente los conceptos abstractos de la programación dinámica. Las más interesantes son algunos de los errores más comunes a la hora de manejar punteros. A la hora de representarlos, se incluye el código en PASCAL para que la ilustración sea más completa.

2.1 Copia de Punteros y Celdas

Un error muy frecuente de los estudiantes cuando intentan duplicar el contenido de una celda es hacerlo asignando punteros. El origen de este error es que se piensa que copiar punteros

implica la duplicación de la celda a la cual apuntan. Siguiendo la metáfora copiar una llave es obvio, el resultado es que después se tienen dos llaves de la misma taquilla. Evidentemente, ni se ha hecho una nueva taquilla ni se han duplicado los contenidos de ésta. Para aclararlo aún más, utilizaremos el siguiente ejemplo:

> *Tenemos dos llaveros k1 y k2. Pedimos al taquillero una llave para el llavero k1. El taquillero nos asigna una taquilla. Ahora podemos abrir la taquilla con la llave del llavero k1 y ponemos 100.000 pesetas en su interior. Duplicamos la llave de k1 y ponemos la copia en k2. Con la llave de k2 abrimos la taquilla y cogemos el dinero. Luego abrimos otra vez con la llave que está en el llavero k1 y no hay nada.*

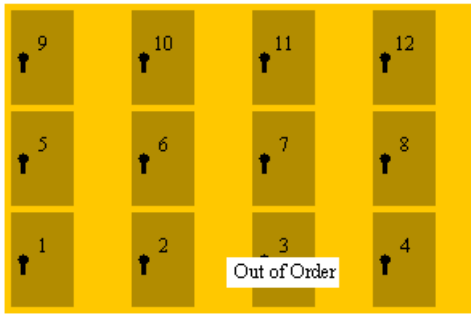
En este ejemplo está claro que copiar llaveros no es el método de hacer más dinero, porque se copian las llaves, no el contenido. El ejemplo es equivalente al siguiente código PASCAL:

```
VAR
k1, k2 : ^INTEGER;
new(k1);
k1^:=1000;
k2:=k1;
k2^:=0;
write(k1^);
```

Este ejemplo sacará por pantalla el número 0.

2.2 Desperdicio de Memoria

Otro error consiste en olvidar liberar las celdas que no se utilizan. En la metáfora es evidente que una taquilla sólo se puede volver a reutilizar si su llave es devuelta al taquillero. Pero si la llave se pierde la taquilla no se volverá a usar ni se podrá asignar a otro.



Para clarificar la situación se utiliza el siguiente ejemplo:

> *Tenemos el llavero k1. Pedimos al taquillero una llave que nos pone en el llavero. Abrimos la taquilla con la llave del llavero k1 y ponemos 1000$ dentro. Después tiramos la llave a un río dejando el llavero vacío. Cuando intentamos abrir otra vez la taquilla para recuperar el dinero, nos damos cuenta que no podemos porque no tenemos la llave.*

En este ejemplo la taquilla nunca podrá volver a ser usada hasta que el taquillero tenga la llave otra vez. El código correspondiente en PASCAL es:

```
VAR
k1 : ^INTEGER;
new(k1);
k1^:=1000;
k1:=NIL;

write(k1^);
```

que provoca un error de ejecución.

2.3 Mal Uso de las Celdas Liberadas

Otro error común es usar celdas que ya han sido liberadas. En la metáfora la situación es como sigue. Una vez que la taquilla ha sido liberada, ésta se puede asignar a quien sea. Sin embargo, el antiguo propietario sigue manteniendo una copia de la llave, por lo tanto dos personas diferentes pueden acceder a la misma taquilla. Aunque sólo una de ellas esté autorizada. Como consecuencia, el buen funcionamiento de la consigna depende de la buena voluntad de los usuarios. Esto se puede ilustrar de la siguiente manera.

> *Nosotros tenemos el llavero k1 e Iñigo tiene el llavero k2. Pedimos al taquillero una llave para ponerla en el llavero k1. Nos dan por ejemplo la taquilla 6. Después liberamos la taquilla, teniendo en cuenta que nos quedamos con una copia de la llave. Después Iñigo pide una llave para su llavero k2, suponemos que se le asigna la misma taquilla (la número 6). Iñigo pone 100.000 pesetas en la taquilla. Entonces nosotros podemos coger el dinero de allí (tenemos una llave). Cuando Iñigo va a recuperar el dinero no hay nada.*

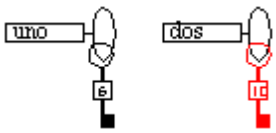
En PASCAL:

```
VAR
k1, k2 : ^INTEGER;
new(k1);
...
Uso de k1
...
dispose(k1);
...
new(k2);//Se supone que le da la misma dirección de memoria
k2^:=1000;
k1^:=0;
write(k2^);
```

Hay que tener en cuenta que este programa es útil a la hora de enseñar el mal uso de los punteros, pero es más didáctico leerlo que ejecutarlo, porque sólo en los casos en que las dos veces se asigne la misma dirección de memoria, saldrá el resultado esperado.

2.4 Punteros sin Inicializar

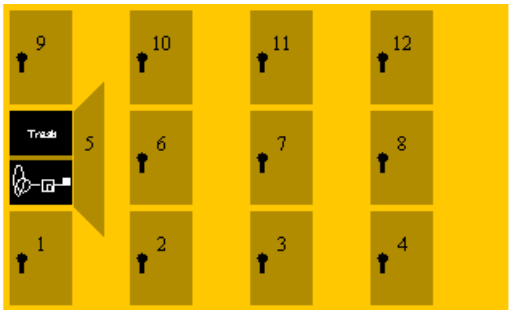
Otro error consiste en usar los punteros sin haberlos inicializado como si estuvieran apuntando a una variable dinámica. En la metáfora la situación puede modelizarse de la siguiente forma. Al principio los llaveros pueden tener llaves de usos antiguos. Por esta razón los llaveros no deben usarse ya que podemos abrir taquillas que no estamos autorizados a abrir. Para usar un llavero primero tenemos que vaciarlo o pedir una nueva llave para él (hacer lo mismo para éste). El llavero "uno" está inicializado mientras que el llavero "dos" tiene una llave de antigua.



La forma de utilizar esta metáfora es explicando primero el escenario de la consigna, pasando luego a la programación a través de los errores y malos usos para luego seguir con la forma tradicional. Haciendo mención a la metáfora cuando sea necesario para que los alumnos tengan los conceptos básicos claros. La metáfora se puede adaptar para que pueda manejar distintos tipos,

haciendo distintos tipos de taquillas. Pero entonces la metáfora se complica, por lo que es preferible hacerlo solo con un tipo simple, como INTEGER.

La metáfora también se puede extender a listas enlazadas, simplemente considerando que la taquilla tiene dos estanterías. En una se almacena el objeto y en la otra un llavero. El llavero de la segunda estantería se puede usar para tener la llave de otra taquilla, por lo tanto las taquillas se pueden enlazar. Esta extensión es interesante para explicar las estructuras de datos lineales. De todas formas parece más aconsejable utilizar la metáfora en la explicación de los punteros. Los estudiantes entenderán rápidamente el código en PASCAL y luego podrán referirse a la metáfora si tienen alguna duda.



La utilización de las metáforas para la enseñanza de informática tiene tres aproximaciones. Operacional, enfocada a medir su efecto en la enseñanza. Estructural, que desarrolla una representación formal de las relaciones entre el dominio origen y el destino. Pragmática, que dice que el conocimiento de las metáforas es incompleto pero que su potencia se basa en esas divergencias. Esta metáfora se ha diseñado siguiendo el segundo método porque se quería una biyección entre los elementos de la metáfora y los de la memoria dinámica. Las características importantes de esta metáfora son:

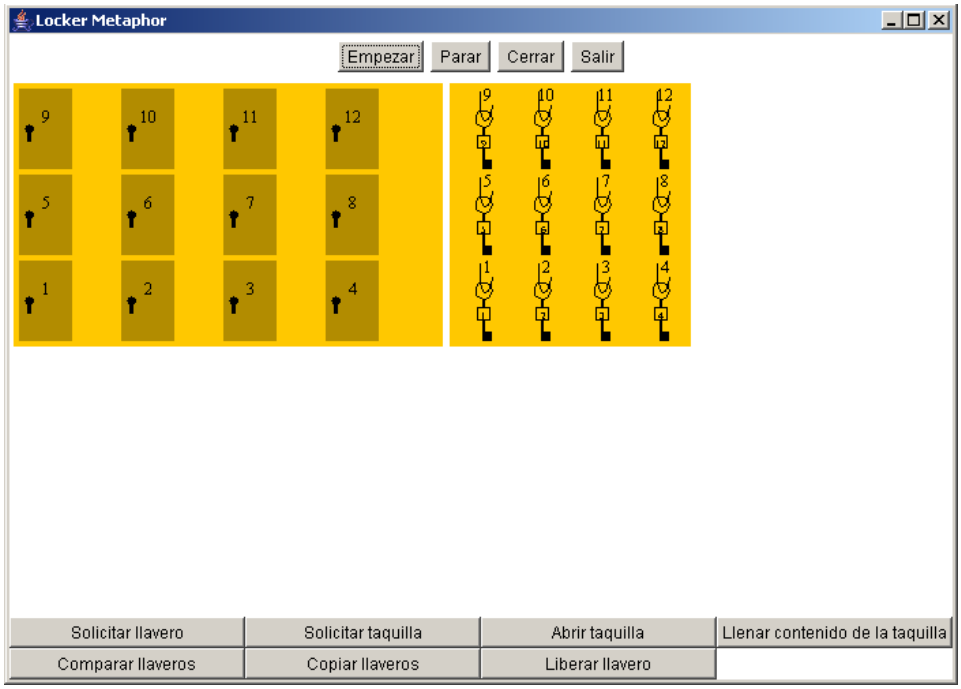
- Es simple y hace referencia a una situación común.
- Tiene una estructura rica que permite una representación clara de todos los conceptos y actividades que están involucrados en la memoria dinámica.
- Su expresividad permite explicar la memoria dinámica utilizando sólo esta metáfora. Esto es importante porque la mezcla de metáforas puede confundir a los principiantes.

3. Locker Room

LockerRoom es una herramienta interactiva para la animación de la metáfora de la taquilla. En ella se pueden visualizar todos los elementos de la metáfora gráficamente. El conjunto de taquillas, la tabla con las llaves de las taquillas y los llaveros en uso. Esta aplicación permite la interacción con los elementos de la metáfora por medio de un interfaz gráfico.

Las opciones de la línea de comandos permiten escoger entre las distintas opciones de lenguaje, dado que la aplicación soporta multilinguismo. En este momento también hay que decidir que representación se quiere mostrar porque la aplicación aparte de representar los elementos de la metáfora puede representar los diagramas estándar de usados para representar los punteros y el resto de elementos programación de memoria dinámica. También es posible elegir el lenguaje de la aplicación, puede ser desde el punto de vista de la metáfora o desde el del lenguaje de programación.

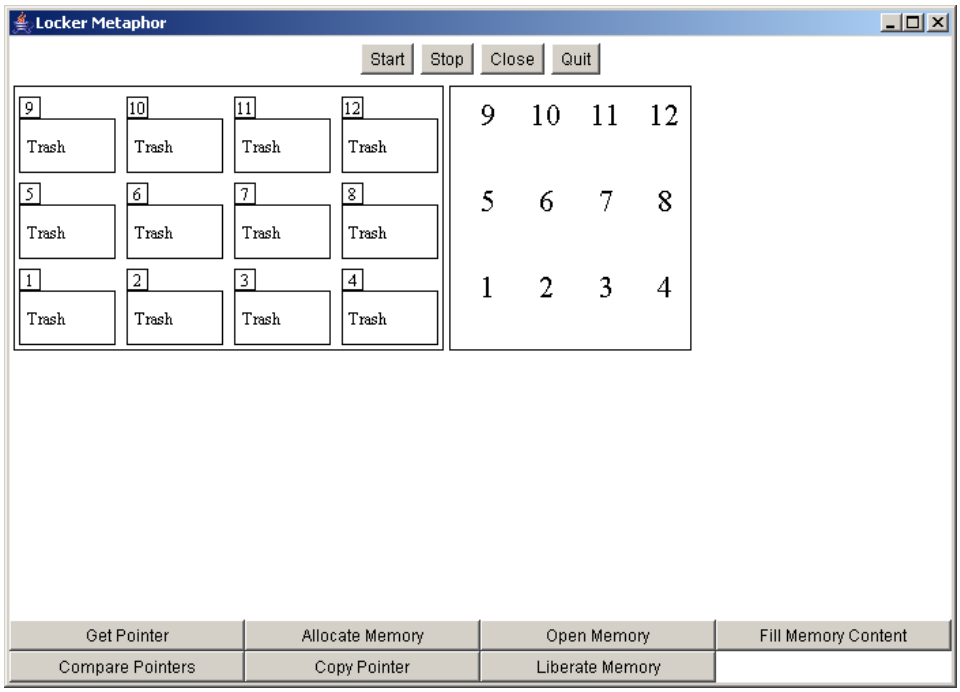
Una vez arrancada la aplicación tiene una serie de botones cada uno de los cuales soporta una acción de la metáfora. En la figura se muestra la aplicación con la opción de idioma español y la de representación y lenguaje de la metáfora.



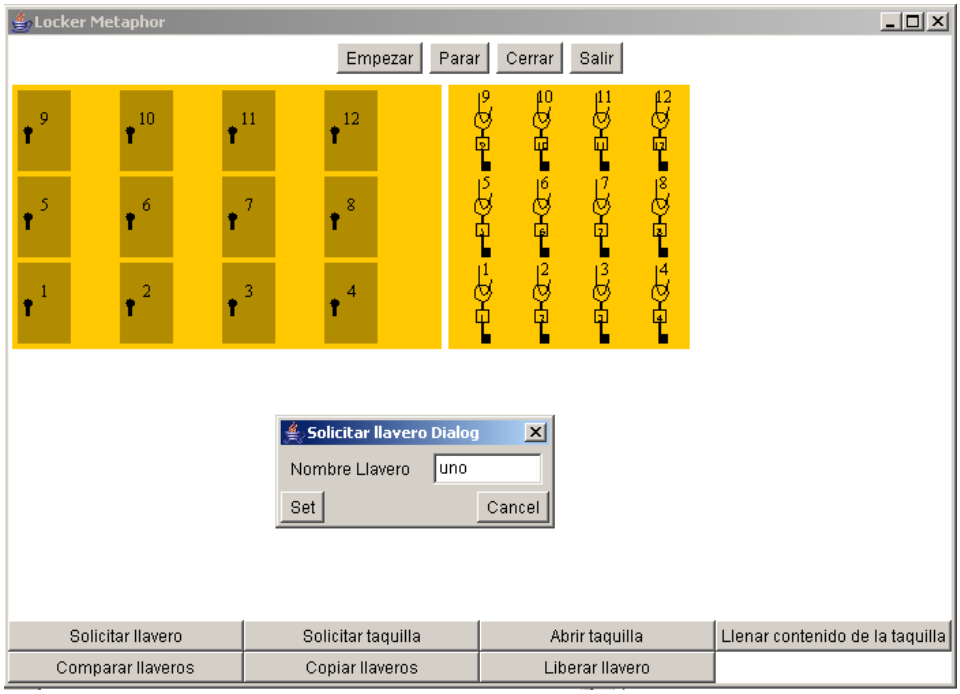
Las operaciones disponibles son:

- Solicitar llavero
- Solicitar taquilla
- Abrir taquilla
- Llenar contenido de la taquilla
- Llenar contenido del llavero
- Comparar llaveros
- Copiar llaveros
- Liberar llavero

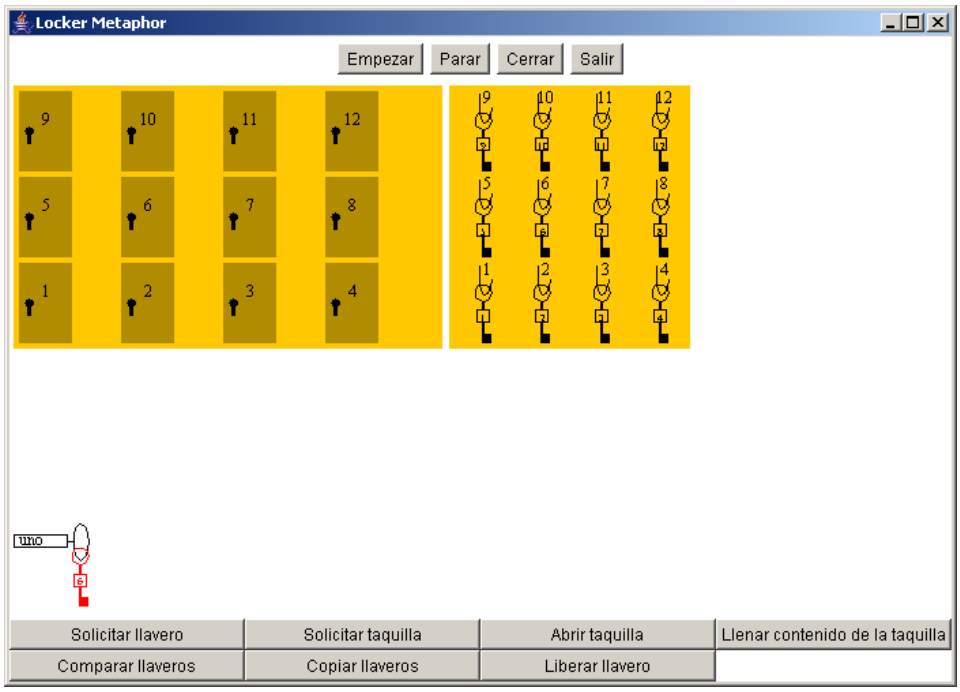
En cambio cuando se selecciona la opción de la representación de programación en vez de los elementos de la metáfora se muestran los esquemas de la representación técnica. A su vez se puede seleccionar el lenguaje de programación para los botones y mensajes de la aplicación. La figura representa la aplicación con el idioma inglés y la representación y lenguaje de programación.



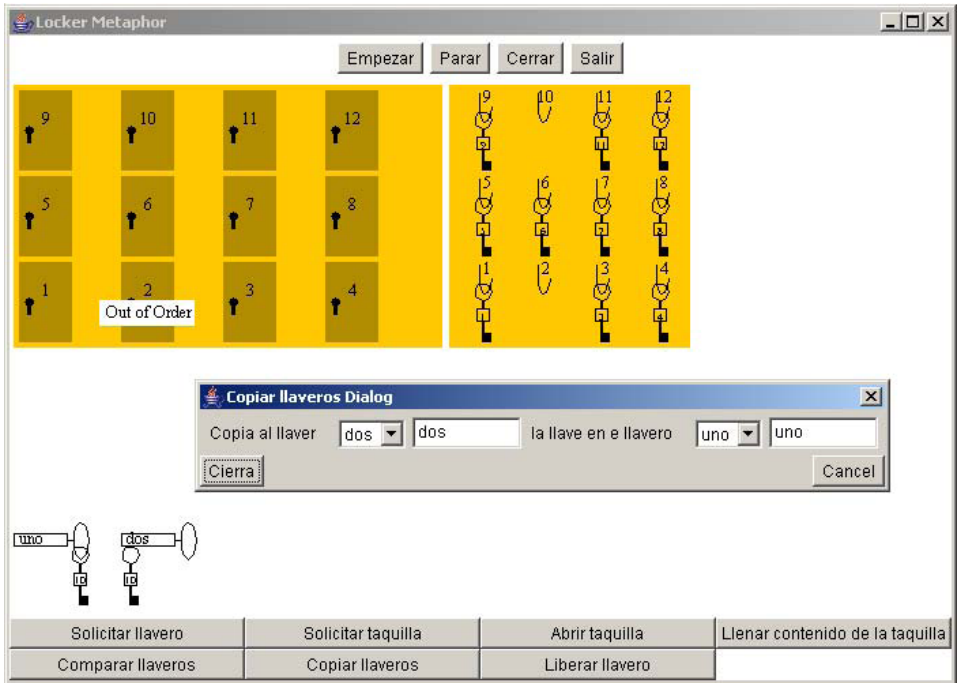
Cuando el intérprete se inicia se muestra la situación inicial de la metáfora. Una interacción típica con el entorno puede ser como sigue. El usuario presiona el botón específico de la acción que quiere realizar y aparece el diálogo correspondiente.



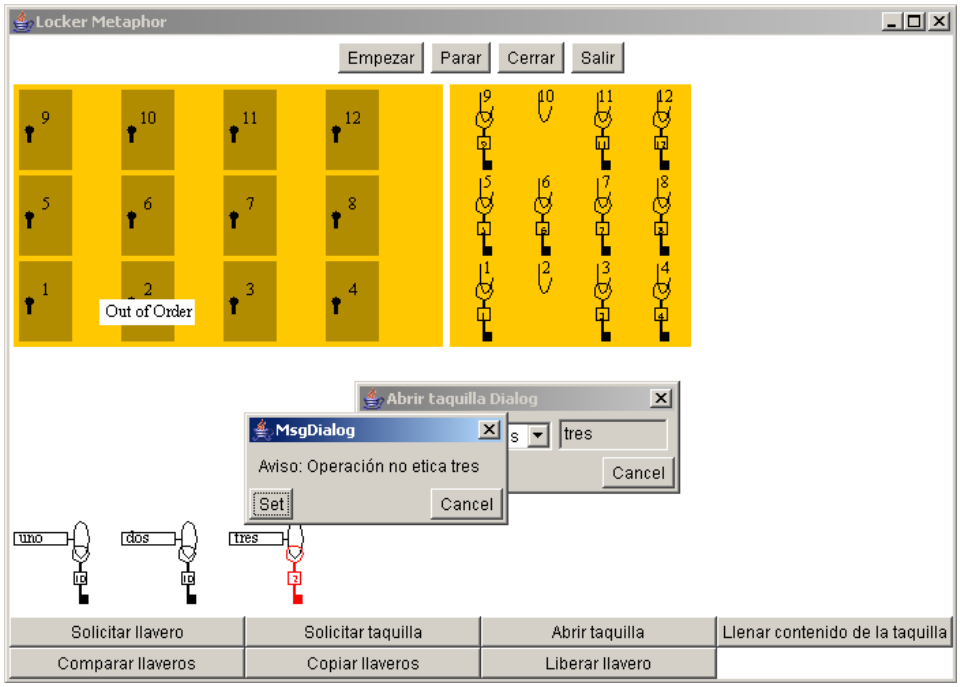
Cuando los datos están completos y validados, la acción se ejecuta y visualiza.



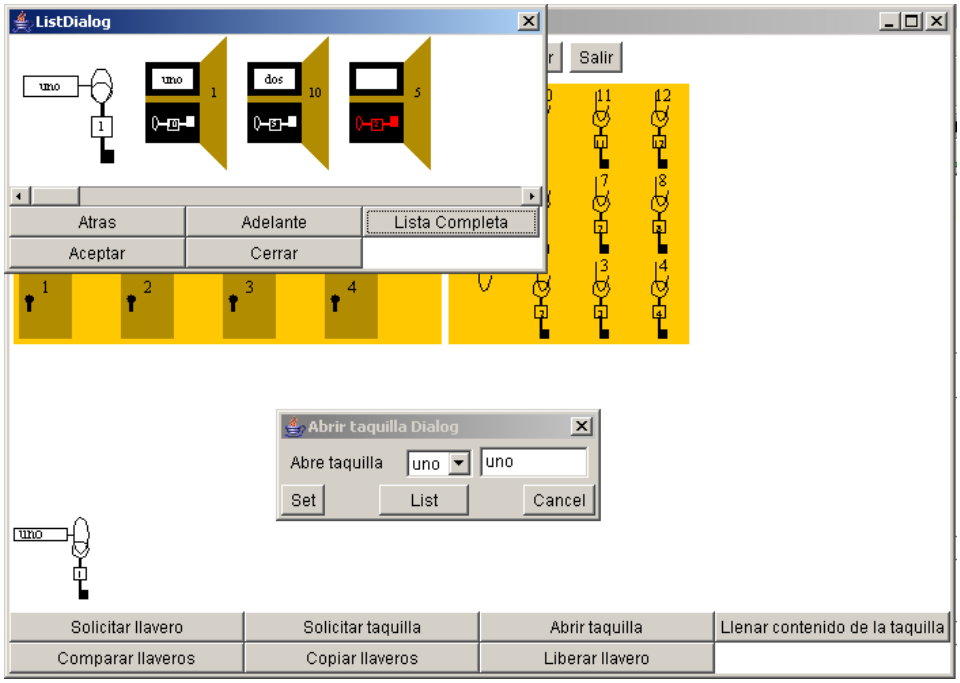
En la aplicación se pueden reproducir las situaciones más comunes en la programación con memoria dinámica como la copia de punteros y el desperdicio de memoria.



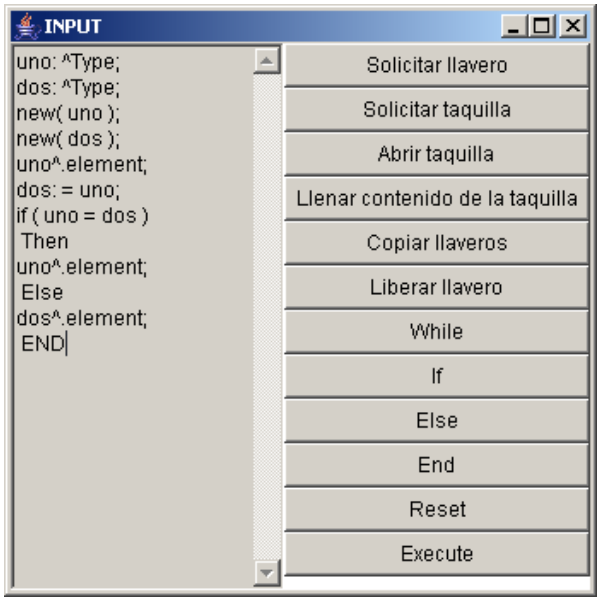
O mal uso de celdas liberadas y punteros sin inicializar.



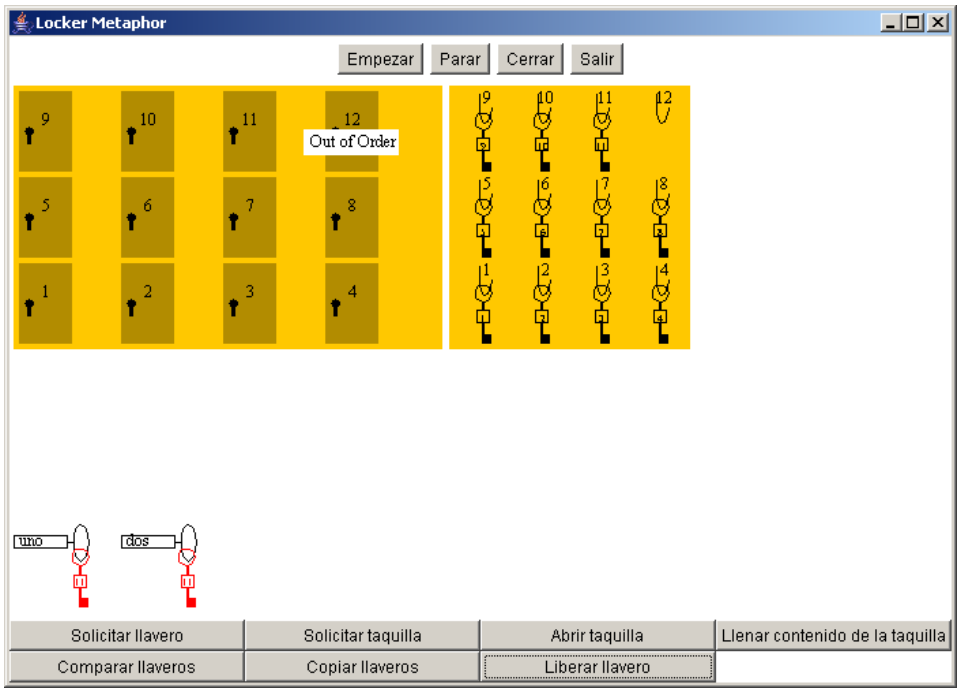
Otra representación implementada en la aplicación es la posibilidad de dividir las taquillas en dos secciones. Una se corresponde con el espacio que permite almacenar algo y la otra seria otro llavero, permitiendo así que se generen cadenas de taquillas y llaveros tal y como se hace en la programación con los punteros.

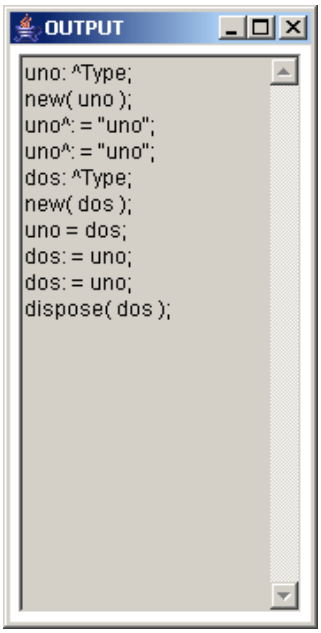


La aplicación Lockerroom tiene un interfaz que permite escribir un programa en pseudocódigo que luego después de validarlo realiza la animación como si ésta fuera hecha paso a paso con el interfaz de usuario.



También esta implementado el sentido contrario, se puede tener una ventana que graba los movimientos del usuario en un fichero para luego reproducirlos.





La aplicación es fácilmente configurable ya que solo tiene las opciones de línea de comando que se explicadas en el Manual de usuario y los ficheros con los mensajes de los distintos idiomas y lenguajes.

LockerRoom esta totalmente animado y todas la acciones generan que suponen movimiento de elementos están implementados con una animación que hacen que sea más fácil imaginar la metáfora ayudando de esta manera a entender los conceptos de programación de memoria dinámica. Por ejemplo la solicitud de una nueva taquilla provoca que la llave correspondiente se mueva hasta el llavero.