



Universidad Simón Bolívar  
Departamento de Computación  
y Tecnología de la Información  
Algoritmos y Estructuras I  
CI-2615 Ene-Mar2005

## Algoritmos y Estructuras I (CI-2691) Proyecto

Este proyecto tiene como objetivo consolidar los conocimientos adquiridos en el curso CI-2691, a través del diseño e implantación del juego conocido como "Arenas Movedizas".

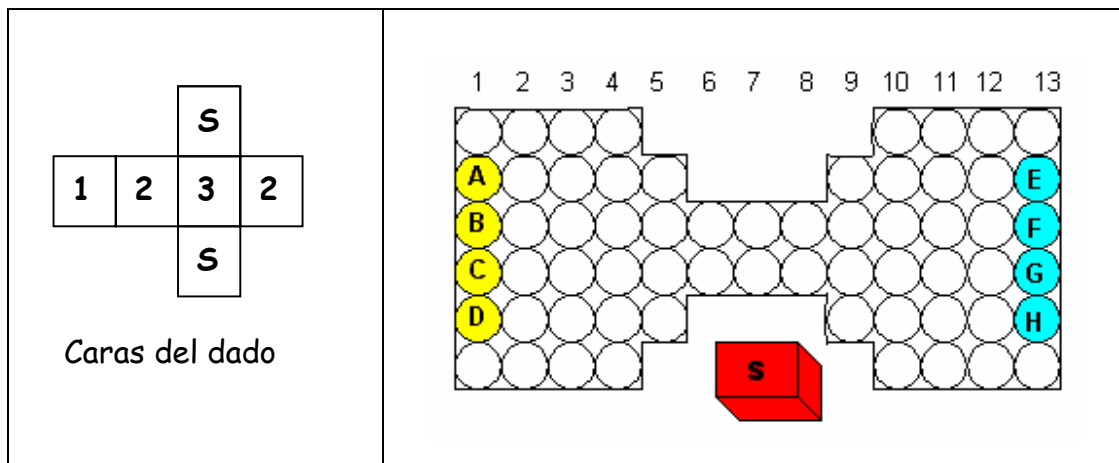
Fechas:

**1ra entrega (intermedia): 31/03/2005**

**2da entrega (final): 07/04/2005**

### EQUIPO DEL JUEGO

- Ocho (8) relojes de arena (4 para cada jugador) identificados por el color del equipo y una letra (A, B, C, D, E, F, G, H).
- Un tablero con forma de reloj de arena como se muestra en la figura 1.
- Un dado cuyas caras se especifican en la figura 1.



**Figura 1.** Caras del Dado y Tablero del juego con los relojes de arena de los jugadores y resultado del dado.

## OBJETIVO DEL JUEGO

Ser el primer jugador en colocar todos sus relojes de arena en la columna inicial de su oponente.

## El juego

A continuación se describen características principales del juego:

1. Todos los relojes tienen una capacidad de 30 unidades.
2. Cada uno de los jugadores ubica sus cuatro relojes de arena con toda la arena en la parte inferior del reloj en su columna inicial (columna 1 para uno y columna 13 para el otro).
3. La única forma de poder arrancar los relojes es sacando **S** en el dado, en ese momento se voltea el reloj (de tal manera que comience a fluir la arena hacia el vaso inferior del reloj) y se avanza un espacio.
4. Una vez dejado la columna inicial, los relojes se mueven la cantidad de espacios indicados en el dado.
5. Todos los relojes trasiegan una unidad de arena a su vaso inferior cada vez que se lanza el dado.
6. Los relojes se pueden mover hacia delante, hacia atrás, hacia la derecha, hacia la izquierda y en forma diagonal y se pueden mover en mas de una dirección en el mismo turno, por ejemplo si saca tres, se puede mover dos espacios hacia delante y uno hacia la derecha (o la izquierda). **NO se puede mover más de un reloj en el mismo turno.**
7. Al mover un reloj, se tienen que usar todos los espacios indicados en el dado.
8. Si a un reloj se le acaba la arena, es decir es trasegada completamente al vaso inferior del reloj el mismo tiene que ser colocado en su columna inicial.
9. Para que un reloj ingrese en la columna inicial del oponente lo tiene que hacer utilizando **todos** los espacios indicados en el dado.
10. Una vez que un reloj llega a la columna inicial del oponente, desaparece del juego.
11. **Para ganar el juego los cuatro relojes deben llegar a la columna inicial del oponente.**
12. Los cuatro relojes se pueden ubicar en cualquiera de los seis espacios de la columna inicial del oponente

13. Los relojes no pueden “brincar” a otro reloj
14. **Cuando un jugador saca S en el dado tiene tres posibles acciones:**
  - 14.1. Puede voltear uno de los relojes propios que están en su columna inicial y moverlo un espacio.
  - 14.2. Puede voltear uno de sus relojes que NO están en su columna inicial y dejarlo en el mismo sitio donde estaba.
  - 14.3. Puede voltear uno de los relojes del oponente y dejarlo en misma posición.
15. **Todas las jugadas donde se saque S se tienen que ejecutar (no se puede “pasar”)**

## PLANTEAMIENTO DEL PROBLEMA

Se desea que se escriba un programa que permita jugar “arenas movedizas” entre dos personas hasta que una de ellas gane o se rinda o se hallan realizado un número máximo de jugadas.

Utilizando la máquina de trazado de GaCeLa, deberá crear una ventana con dimensión nxm. En esta ventana dibujará los elementos del juego: el tablero, los relojes y el dado. Adicionalmente, debe tomar en cuenta un área para mostrar mensajes de interacción con los jugadores.

El programa debe tener las siguientes funcionalidades:

1. Al iniciar el juego se debe solicitar el nombre de cada jugador para mostrarlo cuando sea su turno de jugar. Al primer jugador se le informa que sus relojes son los azules marcados con los nombres A, B, C y D; a la segunda persona se le informa que sus relojes son los amarillos marcados con los nombres E, F, G y H. Adicionalmente se pide el número máximo de jugadas o movimientos que se realizaran en el juego.
2. Se debe seleccionar de forma aleatoria el jugador que inicia el juego (utilice la acción *random* de GaCeLa <http://gacela.ac.labf.usb.ve/GCLWiki/Wiki.jsp?page=LangRandom>).
3. El jugador en turno lanza el dado o se rinde.
4. Si un jugador se rinde, automáticamente se declara ganador al otro jugador y termina el juego.
5. Si el jugador lanza el dado se activa un procedimiento que le resta una unidad a todos los relojes de arena, si alguno llega a cero lo informa “Se

le acabó la arena al reloj X del jugador YY (donde X = A, B, C, D, E, F, G O H; y YY ES 1 ó 2)", se coloca el reloj X en su columna inicial. Luego se activa un procedimiento que genera un número aleatorio en el rango entero [1, 6], asociado a las seis opciones del dado. Muestra en pantalla la opción obtenida y el jugador ejecutará alguna de las siguientes acciones:

- 5.1. Si obtuvo una S se le solicita al jugador el identificador (la letra) del reloj que va a usar y la computadora ejecuta la jugada con ese reloj de acuerdo a la regla dada arriba.
- 5.2. Si obtuvo un número, se solicita al jugador el identificador del reloj que va a mover, el programa verifica que el reloj especificado se puede mover, si no puede moverse se solicita otro identificador hasta que indique uno que si se pueda mover
  - 5.2.1. Una vez especificado el identificador del reloj que puede moverse, el usuario selecciona la dirección de movimiento del reloj, indicando un número según se muestra en la figura 2. El esquema de movimiento mostrado en la Figura 2, puede estar dibujado en el tablero o aparecer cada vez que se requiera. El reloj se mueve un espacio en la dirección indicada.

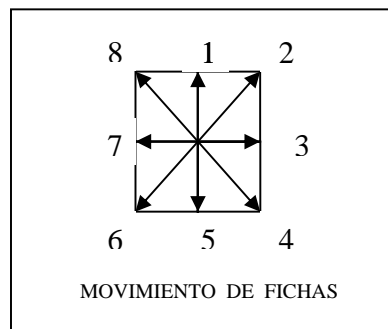


Figura 2. Movimientos posibles de un reloj sobre el tablero.

- 5.2.2. La especificación de la dirección de movimiento se debe realizar tantas veces como lo indique el número obtenido al lanzar el dado. También es posible que todos los movimientos se realicen en una sola dirección. Si el movimiento no se puede efectuar, se debe informar al usuario, quien selecciona otra dirección de movimiento.
6. Si al terminar los movimientos el reloj llega a la columna inicial del oponente, se debe informar "El reloj X del jugador YY (donde X = A, B, C, D, E, F, G O H; y YY ES 1 ó 2) llegó a la columna inicial del oponente" y se elimina el reloj del juego. Si se han eliminado todos los relojes de un jugador, se declara ganador a ese jugador y se termina el juego.

7. Si se han ejecutado el número máximo de jugadas, se termina el juego declarándolo "empate". De otra forma, se asigna el turno de juego al jugador contrario y se repite las acciones anteriores desde el paso numero 3.
8. Cuando se termine un juego, se debe preguntar si se desea iniciar otro juego.

## **ESQUEMA PARA EL PROGRAMA**

Una versión inicial del programa ofrece una visión de la lógica del mismo y permite detectar las partes que se deben desarrollar en detalle para obtener una solución más concreta. Esto es precisamente el objetivo del análisis descendente: comenzar con un problema grande y plantear su solución en función de las soluciones de sub-problemas más sencillos.

A partir del análisis descendente, se pueden identificar los sub-problemas más importantes:

- Determinar si se desea jugar
- Si llegaron todos los relojes de algún jugador a la columna inicial del oponente
- Si una jugada dada es válida
- Realizar una jugada
- Evaluar el tablero y mostrar el resultado final.

El proyecto de programación involucra dos componentes básicas:

- una parte de cálculo, con la lógica del juego y
- una parte de interfaz, con los elementos a través de los cuales se va a interactuar con la componente de cálculo.

En la componente de cálculo, lo primero que se debe hacer es especificar una visión general de la solución.

Aplicando análisis descendente se obtiene una primera versión:

```
[[
  Inicializar juego
  Selección aleatoria del jugador inicial
  do (se desea jugar) -->
    do (no_llegaron_todos_relojes  $\wedge$  nro.jugadas < Máximo ) -->
      Jugador de turno realiza su jugada
      Determinar próximo jugador
      Incrementar el nro.jugadas
    od
  Resultado
od
]]
```

donde Jugador de turno realiza su jugada sería:

```
  Lanzar dado y determinar numero de movimientos M a realizar
  Actualiza relojes
  k:= 0
  Intentos = 0
  do k < M  $\wedge$  Intentos < MAXIMO_INTENTOS -->
    Decidir movimiento
    if movimiento es válido -->
      Realizar movimiento
      k := k + numero de pasos realizados
    [] movimiento no válido -->
      Mensaje de Alerta
    fi
  od
```

(MAXIMO\_INTENTOS = M \* 8,

donde M es el número de movimientos y 8 representa el número máximo de intentos erróneos que puede seleccionar el jugador en turno).

## ESTRUCTURA DE DATOS SUGERIDA

Antes de proceder a la especificación de los sub-problemas identificados en la sección *Estructura general del programa*, resulta conveniente decidir el tipo de datos que se usará para representar las componentes principales del juego: el tablero, los relojes del jugador 1, los relojes del jugador 2 y el dado

### Tablero:

Puede considerarlo como una matriz  $N \times M$  de dimensión 2 (matriz) de enteros. Los elementos de esta matriz representan una casilla del tablero y podrá tomar alguno de los valores siguientes:

- 0: para indicar que la casilla está vacía.
- X:  $1 \leq X \leq 4$ , para indicar que la casilla está ocupada por el reloj X del jugador 1.
- Y:  $5 \leq Y \leq 8$ , para indicar que la casilla está ocupada por el reloj Y del jugador 2.
- -1 para indicar que la casilla es inválida para todas las jugadas.

### Relojes de jugador

Para los relojes es necesario saber cuánta arena le falta por trasegar y su ubicación en la pantalla (fila y columna). Los relojes de un jugador pueden representarse como una matriz de números enteros, de cuatro (4) filas y tres (3) columnas, cada fila de la matriz representa un reloj y las columnas de la matriz representan: cantidad de arena por trasegar, la fila de la posición del reloj en pantalla y la columna de la posición del reloj en pantalla.

La cantidad de arena de cada reloj debe ser inicializada en la capacidad máxima permitida (30) y disminuida en una unidad en cada lanzamiento del dado. Cuando el reloj llegue a la columna del oponente se le podría asignar un valor negativo a la cantidad de arena de ese reloj.

Note que una cosa es como se almacena y maneja la información dentro del computador y otra es lo que se muestra al usuario a través de la interfaz del programa.

Una vez decidido el tipo de datos que conviene para representar el tablero, los relojes del jugador 1, los relojes del jugador 2 y la jugada, se puede proceder a las especificaciones de los sub-problemas.

### Dado

Variable entera cuyos valores posibles son valores enteros asociados a las caras del dado especificadas arriba.

## SUB-PROBLEMAS

Los sub-problemas determinados en base a la estructura general del programa son:

### Lanzar dado:

Función entera que ejecuta el lanzamiento aleatorio del dado y devuelve un valor entero asociado a la cara del dado que salió. Para obtener valores aleatorios utilice la acción random de GaCeLa

### Actualizar relojes:

Función o procedimiento que disminuye en uno los contadores de todos los relojes de cada jugador, verifica los estados de estos e informa si alguno llegó a cero, y ejecuta las acciones indicadas en las funcionalidades del juego descritas arriba.

### Decidir Movimiento:

Función o procedimiento que determina el movimiento a realizar, solicitando al jugador en turno la dirección de la jugada y la cantidad de pasos que desea avanzar en esa dirección.

### MovimientoVálido:

Es una función booleana que debe determinar si un movimiento es válido o no, basándose en la información indicada por el jugador. Deberá verificar si la ficha (reloj) puede moverse en la dirección indicada y en el número de pasos indicados, en caso afirmativo devuelve true y en caso negativo devuelve false. Esta función basa su decisión en valores obtenidos por la función DecidirMovimiento.

### RealizarMovimiento:

Este procedimiento ejecuta la jugada determinada en DecidirMovimiento, actualizando los valores en la matriz Tablero, la(s) posición(es) de lo(s) reloj(es) y actualizando el dibujo mostrado en pantalla.

### Actividades a reportar:

Para cada uno de las funciones y procedimientos que defina en base a los sub-problemas, Ud deberá reportar:

1. Proponga una especificación de una función o de un procedimiento, describiéndola como se muestra en el ejemplo siguiente:  
booleano func JugadasValida (entrada T : Tablero de enteros, Fila, Columna : entero)  
{ Pre: ... PreJugadasValida...}  
{ Post: ...PostJugadasValida...}
2. Escriba las instrucciones de la función o procedimiento.

## INTERFAZ CON EL USUARIO

La segunda parte en la elaboración del proyecto consiste en introducir los elementos de interfaz necesarios para que el programa intercambie información con el usuario.

Para facilitar la programación en este proyecto, se limitarán las operaciones de entrada de información por teclado en modo texto, vía la consola donde se ejecuta el programa, tal como se ha hecho en los programas del laboratorio.

Adicionalmente, el programa se comunicara a través de la pantalla de dibujo, la cual se define con la máquina de trazado. En esta pantalla se dibujaran los elementos gráficos del juego: el tablero, los relojes y el dado.

### Elementos de la interfaz

La información que introducirán los jugadores (entrada del programa) es:

- Nombre
- Número máximo de jugadas .
- Se desea lanzar el dado o rendirse.
- La identificación del reloj a mover.
- La dirección en la que va a mover el dado y número de espacios a mover.
- Deseo de jugar otra partida.

La información que dará el computador (salida del programa) es:

- Dibujar el tablero actualizado.
- Informar las jugadas invalidas.
- Mostrar que un reloj se regresa a su columna inicial.
- Mostrar que un reloj llega a la columna inicial del oponente y por ende sale del juego.
- Mostrar cuando se ha alcanzado el número máximo de jugadas.
- Mostrar cuando uno de los jugadores gana o se rinde.

## ENTREGAS

### Primera entrega (31/03/2005):

En esta entrega su programa debe:

- Dibujar el tablero
- Solicitar los nombres de los jugadores
- Seleccionar en forma aleatoria al jugador que comienza y

Debe entregar junto con el programa, un informe con las especificaciones de las funciones y procedimientos, indicando claramente precondition y poscondición de cada uno de ellas.

Reporte en este informe todo lo que se indica en "Actividades a reportar" (no es el código fuente del programa).

### Entrega final (07/04/2005)

Para el día de la revisión usted deberá llevar una versión operativa del programa implementado en GaCeLa que permita jugar "Arenas movedizas" según las especificaciones indicadas en el planteamiento del problema.

Es importante que el equipo trabaje de manera integrada. Durante la revisión se verificará el funcionamiento del programa y se hará un interrogatorio corto a cada miembro del equipo.

La versión operativa del programa se deberá entregar en un disquete debidamente identificado. Además deberá entregar un informe que describa en detalle todo el proceso de diseño y desarrollo de la solución

El informe debe contener, como mínimo, las siguientes secciones

- ✓ Portada
- ✓ Introducción:
  - Breve descripción del problema atacado en el proyecto
  - Contenido del informe
- ✓ Diseño:
  - Resultado del análisis descendente del problema
  - Optimizaciones e innovaciones incorporadas en el proyecto
- ✓ Detalles de implantación:
  - Estructuras de datos.
  - Lista de procedimientos y funciones indicando la relación de invocación (quien invoca a quien)
  - Código fuente **documentado**
- ✓ Conclusiones:
  - Resultados obtenidos
  - Dificultades presentadas
  - Recomendaciones

## MATERIAL DE APOYO

1. Para la generación de números aleatorios revise algunos de los ejemplo del GCLWiki. (<http://gacela.ac.labf.usb.ve/GCLWiki>)  
Uso de la acción "random".  
Después de usar "random r" la variable r de tipo double toma un valor entre 0 y 1 ( $0 \leq r \leq 1$ )  
Si se desea obtener un entero entre 0 y n hay que multiplicar por n a la variable r y truncar el valor (usar  $\text{toInt}(r * n) + 1$ ).
2. Para aprender a usar la máquina de trazados o incrementar sus destrezas en el uso de la misma revise ejemplos en el GCLWiki. Préstele particular atención al uso de las aserciones de corrección.
3. Consulte la guía del Profesor Oscar Meza para ejemplos de análisis descendente.
4. También puede encontrar en el GCLWiki información sobre el uso de los procedimientos y funciones en GCL.