Trabajo Práctico AEDD

Fecha de entrega: 16 de diciembre 2024

Objetivos:

En el trabajo práctico integrador de este año, cada grupo deberá desarrollar una aplicación que permita a los usuarios registrados acceder a una serie de algoritmos numéricos.

El grupo deberá especificar la estructura de menús de la interfaz de usuario, y también definir y codificar algoritmos numéricos, implementados mediante funciones. La aplicación se va a completar con el desarrollo de la gestión de usuarios y una simulación del juego Super Mario Bros.

La aplicación deberá ser implementada utilizando los conceptos desarrollados en la asignatura, tales como:

- Estrategia de descomposición modular descendente.
- Funciones y recursividad.
- Tipos de datos simples, compuestos y abstractos.
- Archivos de texto y binarios para el almacenamiento persistente de los datos.

Además, se espera que los alumnos desarrollen una aplicación amigable, que minimice la posibilidad de ingreso de datos erróneos, con mensajes adecuados que faciliten la experiencia del usuario. La aplicación deberá proporcionar una interfaz de usuario intuitiva y funcional.

I. APLICACIÓN INTEGRADA

Al ingresar a la aplicación, se presenta un ***Menú de acceso***. A continuación se muestra el esquema de opciones que debe respetar:

Menú de acceso

```
Menú de acceso
==========================
1.- Registro
2.- Iniciar sesión
3.- Algoritmos Numéricos
4.- Juego Super Mario Bros
9.- Salir de la aplicación

Ingrese una opción:
```

En este menú, el funcionamiento solicitado es el siguiente:

- **Opción 1:** Al ingresar a esta opción el usuario tendrá la posibilidad de crear una nueva cuenta. Para esto el programa le solicitará nombre de usuario y contraseña válidos. En la sección II. GESTIÓN Y AUTENTICACIÓN DE USUARIOS se especifican las reglas a seguir para la conformación de cuentas válidas.

  Una vez ingresado un nombre de usuario correcto y una contraseña válida, se le solicitará al usuario que confirme la contraseña (reingresándola). Si la cadena reingresada coincide con la original, se informará el éxito del registro de usuario y se preguntará si desea iniciar sesión.

  En caso afirmativo, se ejecutará la secuencia de operaciones asociada a la opción **Iniciar sesión**. En caso contrario, el programa volverá al **Menú de acceso**. Por otro lado, si el reingreso de la contraseña no coincide con la cadena original, se informará el fracaso de la operación de registro y el programa volverá al **Menú de acceso**.

- **Opción 2:** La aplicación solicita al usuario que ingrese su nombre de usuario y clave. Si los datos ingresados no corresponden a una cuenta de usuario válida, se informa el error. El usuario tendrá 3 intentos consecutivos para ingresar los datos correctos. Si en alguno de estos intentos los datos son correctos, se continúa con la ejecución normal. Vencidos los tres intentos, el programa mostrará un mensaje de error por posible violación de seguridad y luego finalizará su ejecución.

  Si los datos ingresados coinciden con la información de alguna de las cuentas registradas, se procede a mostrar un mensaje de bienvenida e información del último acceso:

```
Bienvenido mARtin12

=======================

Último acceso a la aplicación: 16/11/2024
```

- **Opción 3:** Se deberán implementar 2 (dos) algoritmos numéricos, a los cuales se accederá desde el siguiente “Menú Algoritmos Numéricos”.

Menú Algoritmos Numéricos

```
Menú Algoritmos Numéricos

==========================

1.- Algoritmo A (Solución iterativa)
2.- Algoritmo A (Solución recursiva)
3.- Algoritmo B (Solución iterativa)
4.- Algoritmo B (Solución recursiva)
9.- Salir de la aplicación

Ingrese una opción:
```

Se deberá reemplazar cada opción con el nombre de los 2 algoritmos detallados a continuación, los cuales deberán resolverse de manera iterativa y/o recursiva, según las definiciones que se brindan a continuación:

1) Conversión Binario a Decimal:

Leer un número entero en sistema de representación Binario, y mostrar su representación en Decimal.
Codificar una versión iterativa y una función recursiva.

Problema relacionado en OmegaUp: https://omegaup.com/arena/problem/Binario-a-decimal/

Recursos:

- https://youtu.be/f9b0wwhTmeU
- https://www.youtube.com/watch?v=g9-MRBBcvdg

2) ThreeBonacci:

Leer un número N positivo, menor que 60, e informar los números mayores ó iguales que 1 que pertenezcan a la secuencia ThreeBonacci, según se define en el problema relacionado en Spoj.com.

Problema relacionado en www.spoj.com:

- https://www.spoj.com/problems/TREEBONACCI/

Recursos:

- https://youtu.be/yDyMSliKsxI

Diseñar la solución usando funciones, las funciones auxiliares pueden ser iterativas ó recursivas.

Para cada uno de los problemas asignados se deben seguir los siguientes pasos:

1) Diseñar una o más funciones para codificar un algoritmo que lo resuelva.
2) Verificar para cada algoritmo, que la solución codificada sea aceptada para el problema correspondiente.
3) Incorporarlo al menú del TP, en el punto **Menú Algoritmo Numéricos.**

Si el grupo lo desea, puede incorporar más algoritmos numéricos al menú. Deberán ser del set de problemas que figuran en la tarea Trabajo Práctico, además de los 2 asignados.

Cada vez que se accede a un algoritmo numérico, debe aparecer un nuevo menú que deberá presentar las siguientes opciones:

SUBMENÚ Algoritmo XXXX

```
Algoritmo XXXX
==========================
1.- Ver definición
2.- Ejecutar

9.- Volver al menú anterior

Ingrese una opción:
```

**Donde XXXX representa el nombre del algoritmo elegido*

Opción 1.- Ver definición

Se visualiza una descripción del algoritmo en forma de string.

Opción 2.- Ejecutar

Se ejecuta el algoritmo en base a sus lineamientos.

Opción 9.- Volver al menú anterior

Retorna al menú principal.

Nota: Ver documento "Errores comunes en la implementación de menús" en el que se explican los errores que suelen cometerse al realizar este tipo de implementaciones. Se recomienda realizar una lectura de este documento, previo a comenzar con el desarrollo del trabajo, a fin de entender los requerimientos planteados antes de comenzar a diseñar/codificar su solución.

- **Opción 4: Juego Super Mario Bros**

Tu grupo debe implementar la Simulación A (opcionalmente, pueden implementar también la simulación B).

Se presentará el **Menú de Juego Super Mario Bros**:

```
Menú de Juego Super Mario Bros
==============================

A.- Simulación de encuentro entre Mario y Luigi
B.- Simulación de búsqueda de Princesa Peach

9.- Volver al menú anterior

Ingrese una opción:
```

Cada vez que se accede a una opción de SMB, se muestra un nuevo menú que deberá presentar las siguientes opciones:

```
Juego Super Mario Bros - Simulación X
=====================================

1.- Ver instrucciones
2.- Jugar y simular

9.- Volver al menú anterior

Ingrese una opción:
```

**Donde X representa el ítem de simulación (A o B).*

Opción Ver instrucciones

Descripción del juego y su simulación en forma de string.

Opción Jugar y simular

Se inicia una partida y al terminarla se deberá visualizar el puntaje obtenido.

Opción Volver al menú anterior

Retorna al menú principal.

La especificación del juego y las simulaciones a implementar se encuentran en el **Anexo I - Super Mario Bros: Juegos y simulaciones** de este documento.

II. GESTIÓN Y AUTENTICACIÓN DE USUARIOS

Al ingresar por la opción de registración, el usuario tendrá la posibilidad de crear una nueva cuenta. Para esto el programa le solicitará nombre de usuario y contraseña. Podrán existir como máximo 100 cuentas.

Nombre de usuario: Quedará definido por una cantidad mínima de 6 caracteres y máxima de 10, los cuales podrán ser letras, dígitos y/o símbolos del conjunto {+,-,/,*}. Deberá cumplir con los siguientes requisitos:

a. Ser único para cada usuario registrado.
b. Comenzar con una letra minúscula.
c. Tener al menos 2 letras mayúsculas.
d. Tener como máximo 3 dígitos.

Ejemplos de nombres de usuario incorrectos: “AbC123” (no cumple con b), ”pTS*1234” (no cumple con d), ”g178Mci” (no cumple con c), ”mARtin123gomez” (tiene más de 10 caracteres).

Ejemplos de nombres de usuario correctos: “mARtin12”, “jo97+AR” .

Contraseña: Su conformación no podrá darse al azar, sino que deberá respetar las pautas establecidas en el problema 2253: ‘Passwords Validator’ del juez en línea Beecrowd.

Nota: Tanto en el nombre de usuario como en la contraseña deben distinguirse mayúsculas y minúsculas.

III. ESTRUCTURAS DE DATOS Y PERSISTENCIA

Se deberán diseñar las estructuras de datos necesarias para gestionar tanto las cuentas de usuario como las partidas realizadas.

A su vez, para no perder la información entre ejecuciones, es necesario persistir los datos de las cuentas registradas.

La estrategia sugerida para cumplir con este objetivo es:

- Cuando se inicia la aplicación, se carga en estructuras de datos en memoria la información almacenada en los archivos.
- Durante la ejecución de la aplicación se actualizan las estructuras mencionadas.
- Al salir de la aplicación se sobrescriben los archivos con los datos que están en memoria.

Como sugerencia de diseño, podría definirse un **arreglo de hasta 100 usuarios** para registrar los datos de usuario. Cada componente del arreglo contiene la siguiente información:

Estructura Usuario:

Tipo	*Campo*	*Observaciones*
char[11]	nombre_usuario	Identificación de usuario.
char[37]	clave	Clave o contraseña de acceso.
Fecha	ultimo_acceso	Fecha de último acceso del usuario a la aplicación.

Las cuentas de usuario se almacenarán en un **archivo binario** llamado **Usuarios.dat**, que almacenará la siguiente información:

Archivo Usuarios.dat

Tipo	*Campo*	*Observaciones*
char[11]	nombre_usuario	Identificación usuario cifrada
char[37]	clave	Clave de acceso cifrada
Fecha	ultimo_acceso	Fecha de último acceso del usuario a la aplicación.

Las cuentas de los usuarios (nombre de usuario y contraseña) deberán almacenarse cifradas siguiendo las pautas establecidas en el problema 1024 del juez en línea Beecrowd.

IV. PAUTAS DE ENTREGA

Conformación de grupos

Los grupos serán conformados por 3 (tres) estudiantes. Si algún grupo tuviera problemas para respetar esta pauta, DEBE CONSULTAR LA SITUACIÓN CON LOS PROFESORES. No se aceptarán entregas de grupos que no respeten esta pauta y no hayan sido debidamente autorizados por los profesores.

Calificación

La solución entregada por cada grupo pasará una revisión de código, como así también una prueba de funcionamiento. El código entregado no solamente debe compilar y ejecutar sin errores, sino también ajustarse a las buenas prácticas de codificación aprendidas a lo largo del cursado.

Las entregas deberán realizarse por grupo, es decir que sólo uno de los integrantes tiene que subir los archivos a la tarea correspondiente del Campus de la materia y a Beecrowd, el mismo alumno para ambos casos. Los programas serán analizados con herramientas de software específicas para detectar plagios de código.

Los trabajos no entregados en tiempo y forma y/o que no cumplan con las pautas de autoría establecidas, NO SERÁN EVALUADOS y se considerarán NO APROBADOS.

Consultas

Las consultas de los grupos serán atendidas por el docente asignado a la corrección, mediante mensajes privados de TEAMS ó email.

Interacción con el usuario y validaciones

La aplicación debe solicitar en todo momento el ingreso de información de forma clara y consistente.

La interacción con el usuario debe ser lo más amigable posible: mensajes de información y/o de error específicos, limpieza de pantalla, etc. Se deben implementar todas las validaciones que se consideren necesarias (además de las enunciadas en este documento).

Documentación a entregar

SÓLO UNO de los integrantes deberá entregar, a través de la tarea **Trabajo Práctico** del Campus, un archivo llamado **TPAEDD-GrupoX.zip**, donde **X** corresponderá al número del grupo. El .zip deberá contener:

- *integrantes.txt:* archivo de texto que contendrá el apellido, nombre y correo electrónico de cada uno de los integrantes del grupo (en orden alfabético por apellido) y el nombre de usuario de URI del responsable de subir las soluciones a la tarea correspondiente.
- El **proyecto** fuente completo: con los archivos del TDA Personaje, librerías y el programa cliente que contiene el desarrollo de la aplicación C++.

SÓLO UNO de los integrantes deberá entregar a través de la tarea **Trabajo Práctico** de Beecrowd las soluciones a los problemas 1024: Encription y 2253: Passwords Validation.

Anexo I - Super Mario Bros: Juegos y simulaciones

***Super Mario Bros**, o **Super Mario Brothers** es un videojuego de plataformas, diseñado por Shigeru Miyamoto, lanzado el 13 de septiembre de 1985 y producido por la compañía Nintendo, para la consola Nintendo Entertainment System (NES).*

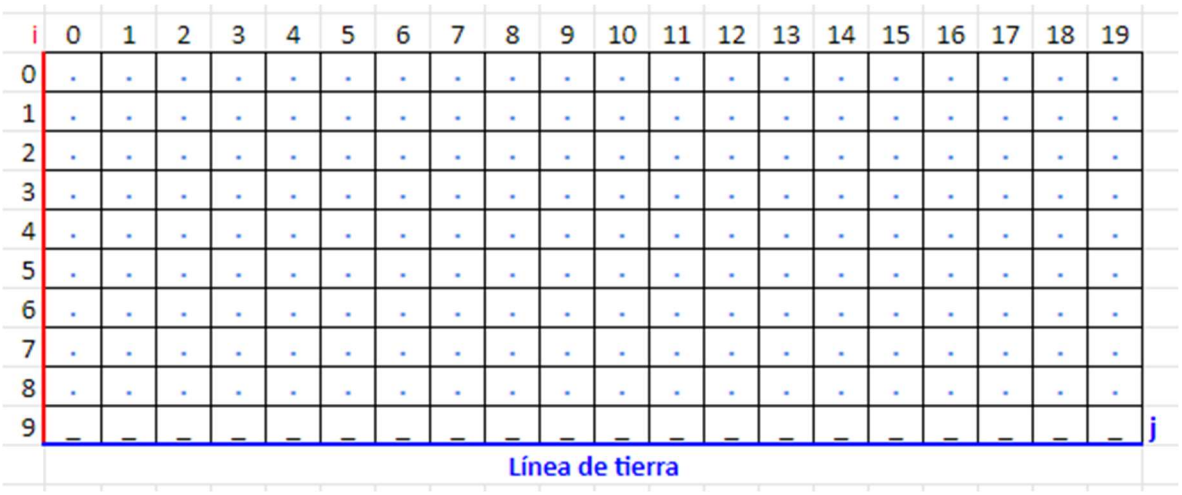
El juego describe las aventuras de los hermanos Mario y Luigi, personajes que ya protagonizaron el arcade Mario Bros de 1983. En esta ocasión ambos deben rescatar a la Princesa Peach del Reino Champiñón que fue secuestrada por el rey de los Koopas, Bowser.

Luigi - Mario

Consideraciones del juego

Para modelar el juego se deberán respetar las siguientes especificaciones:

- La pantalla del juego, el Reino Champiñón, se considera como el primer cuadrante de un sistema de coordenadas cartesianas, La pantalla estará representada por una matriz de 10 filas y 20 columnas, siendo la línea de tierra la última fila de matriz. A continuación se presenta un ejemplo de representación gráfica inicial:



Donde las celdas con '.' (punto) representan el cielo y las celdas con '_' (guión bajo) representan la línea de tierra.

- Tanto Luigi como Mario son personajes que se caracterizan por tener un nombre, usan camisa, gorro y pantalón de determinados colores (ver imagen), y en el gorro tienen la inicial de su nombre.
- La ubicación inicial de los personajes es una celda **[i,j]** de la pantalla, en la línea de tierra.
- Los personajes tienen una orientación, que puede ser derecha o izquierda y que puede cambiar cuando giran, siempre en 180 grados.
- Los personajes tienen la posibilidad de:
  - Saludar, diciendo "Hola, soy <nombre_personaje>!"
  - Caminar un paso en la dirección en la que se encuentran orientados.
  - Girar 180 grados.
  - Saltar, para subirse a una piedra.
  - Bajar, para descender de una piedra.
  - Trepar, para subir a una piedra de altura mayor a uno.
  - Indicar la ubicación en la que se encuentran.
- Para crear los personajes hay que indicar todas sus características, incluyendo la orientación y la ubicación.

Menú de Simulaciones - SMB

Como ya se explicó, cuando se ingresa al juego se presentará un menú con 3 opciones:

```
A.- Simulación de encuentro entre Mario y Luigi
B.- Simulación de búsqueda de Princesa Peach
9.- Volver al menú anterior
```

Las dos simulaciones arrancan con el armado del Reino Champiñón según se especifica.

A.- Simulación de encuentro entre Mario y Luigi

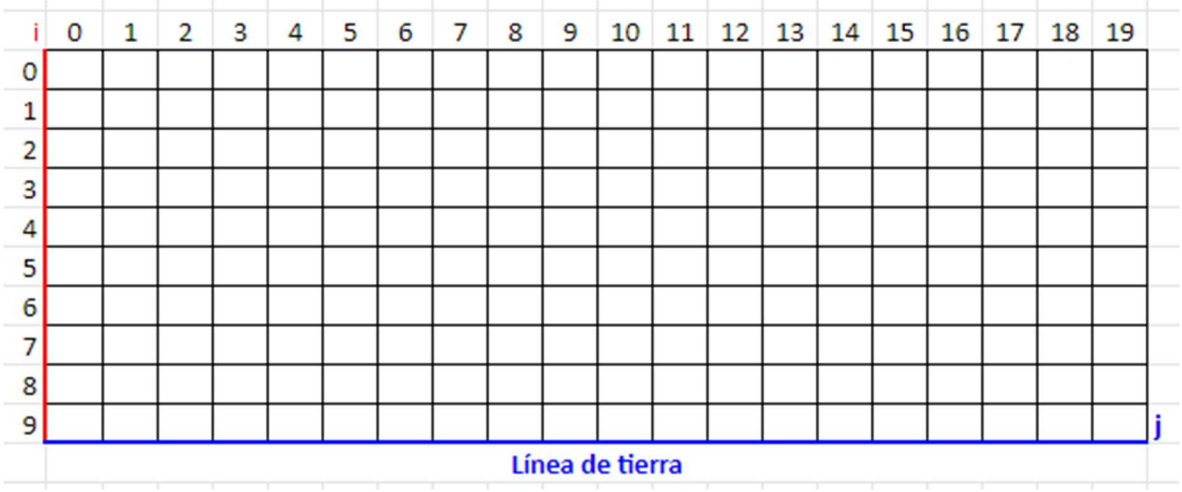
Armado del Reino Champiñón

Al ingresar se deberá generar automáticamente el tablero que representa al Reino Champiñón, haciendo uso de la función cargarTablero de la librería *stdpanel* (descargar).

La función cargarTablero ubica en forma aleatoria y oculta al usuario, aves y obstáculos (piedras) según se indica a continuación:

- 10 aves: cada una ocupará una celda [i,j] que cumpla con las siguientes condiciones:
  - i <= 2, es decir, en la parte más alta del cielo del Reino Champiñón
  - j, cualquier valor válido en la matriz para este índice
- 5 piedras, dos de las cuales tendrán altura 3, otras dos tendrán altura 2 y una tendrá altura uno. La base deberá estar en la línea de tierra, ocupando celdas hacia arriba según la altura de la piedra.
- No puede haber dos piedras en una misma posición.
- No puede haber dos piedras en ubicaciones consecutivas.
- No puede haber una piedra ni en la celda inicial ni en la celda final de la línea de tierra.

Al comenzar el juego el usuario ve por pantalla al Reino Champiñón vacío, incluso sin marcas de cielo ni de línea de tierra:



Luego, el usuario deberá descubrir celdas (es decir, pares de valores i,j) hasta identificar todas las aves y piedras en base al tablero que se generó con cargarTablero; o hasta que el usuario ingrese el número 99 para el valor de i. Las posibles respuestas son:

- **Aire o Tierra**, si en la casilla no hay ni un ave ni una piedra (ni una parte de ella).
- **Acierto**, si la casilla corresponde a un ave, o a alguna piedra (o una parte de ella).

Si se produce Aire o Tierra, se deberá refrescar la pantalla reemplazando la celda vacía por el caracter correspondiente a cielo o tierra según ya se indicó anteriormente..

Si se produce un Acierto se deberá refrescar el tablero reemplazando el valor de la celda por el caracter correspondiente, a saber:

- '^' (acento circunflejo): Ave.
- 'X' (equis mayúscula): Bloque de piedra.

Una vez que el usuario haya descubierto dónde estaban todas las aves y las piedras o cuando el usuario ingresa 99 para el valor de i, se mostrará el puntaje obtenido que resulta de sumar el total de aciertos de aves y piedras.

De ahora en adelante el Reino Champiñón siempre estará visible en pantalla incluyendo todas las aves y piedras que existan (hayan sido halladas o no en la etapa anterior) junto con las zonas de cielo y línea de tierra. Ejemplo:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	^	.	.	.	.	^	.	.	.	.	.	.	.	^	.	.	.	.	^	.
1	.	.	.	.	.	.	.	.	^	.	.	.	.	.	.	^	.	.	.	.
2	.	^	.	^	.	.	.	.	.	.	.	.	^	.	.	.	.	.	^	.
3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.
8	.	.	.	.	X	.	X	.	.	.	.	.	.	.	X	.	.	X	.	.
9	M	_	_	_	X	_	X	_	_	_	X	_	_	_	X	_	_	X	L	_

j

Línea de tierra

Simulación:

A partir de ahora comienza la simulación, en la que se pide:

- Crear a **Mario** sabiendo que comienza el juego en una celda aleatoria de la línea de tierra, y con una orientación, "Derecha" o "Izquierda", que será también aleatoria. Crear a **Luigi** con los mismos criterios que a Mario. Considerar que:
    - Luigi y Mario no pueden iniciar la simulación ubicados en la misma celda.
    - La ubicación de los personajes debe ser en celdas libres, es decir, sin piedras.
- En la pantalla deberá aparecer el Reino Champiñón y los dos personajes, L y M, cada uno en su celda inicial.
- Ambos tienen que moverse en forma alternada hasta encontrarse, mostrando su ubicación por pantalla a cada paso. Los personajes se encuentran cuando coinciden en la misma celda. Cada vez que uno de los personajes avanza, se debe refrescar la pantalla mostrando nuevamente al Reino Champiñón con los personajes en su nueva ubicación.
- Si un personaje llega a algún límite del Reino Champiñón debe girar para cambiar su orientación y así poder avanzar en su siguiente turno.

- Debe haber una espera entre cada movimiento para que se pueda visualizar la simulación.
- En el camino pueden encontrarse con piedras, y deberán moverse en consecuencia:
    - Si la piedra es de altura 1, salta una vez sobre ella, y luego deberá bajar cayendo a la línea de tierra.
    - Si es de altura mayor a 1, la trepará tantas veces según sea su altura-1. Al llegar a tope deberá saltar sobre el último nivel de la piedra, para luego bajar cayendo a la línea de tierra.
- Se debe contar cada movimiento que haga cada personaje: caminar, girar, saltar, trepar, bajar. Los movimientos se van alternando por turno entre personajes, pero si alguno se encuentra con una piedra seguirá moviéndose hasta sortearla por completo y llegar a la línea de tierra. Luego se volverá a mover el otro personaje.
- Cuando se encuentran, debe aparecer por pantalla un mensaje que indique que se encontraron, y luego ambos se tienen que saludar. En la celda del encuentro deberá aparecer una **B** (Brothers en inglés).
- Al finalizar la simulación se muestra por pantalla cuántos movimientos tuvo que hacer cada uno: gana el que haya hecho la menor cantidad! Si la cantidad de movimientos es igual, debe aparecer un mensaje de empate.

B.- Simulación de búsqueda de Princesa Peach

Armado Reino Champiñón:

Al ingresar se deberá armar automáticamente el tablero que representa al Reino Champiñón, solo con los caracteres que representan al cielo y a la línea de tierra.

Simulación

- Crear a **Luigi** y a **Mario**, sabiendo que ambos comienzan el juego en la línea de tierra, con un mismo valor de j generado en forma aleatoria, y orientación “Izquierda”.
- Luego, en la pantalla deberá aparecer el Reino Champiñón y los dos personajes, L y M, en la celda correspondiente a la que arrancan jugando. Como en una celda no puede haber dos caracteres, se podrán representar ambos con la **B** (Brothers, hermanos en inglés).
- Crear a **Princesa Peach**, que es un personaje que tiene las mismas características que Mario y Luigi, con la diferencia de que su gorro y camisa son color violeta, y el pantalón es color amarillo. Bowser secuestró a Princesa Peach y la tiene atrapada en una torre invisible. Su ubicación, que es en la línea de tierra también con **j** aleatorio, coincidirá con la ubicación de la torre.
- La ubicación de Princesa Peach en su torre no podrá coincidir con la de los hermanos **B**.
- Como Mario y Luigi no pueden tener comunicación directa con Princesa Peach debido a que Bowser les rompió el celular, los hermanos deberán acudir a Whalum, un ánima siniestra del

grupo de Oráculos que sólo puede darles pistas para ubicar a Princesa Peach. Whalum tiene visión ultravioleta y es capaz de detectar las torres invisibles, pero no todo es tan fácil: los va a orientar en base a preguntas que ellos vayan haciendo.

En esta instancia, el juego pasa a modo primera persona: el jugador se convierte en protagonista, ya que podrá tomar el rol de Mario o de Luigi según elija a través de un menú.

- Una vez elegido el personaje, comienza el intercambio con Whalum, quien simplemente aparece en la escena sin necesidad de buscarlo ni de representarlo en la pantalla. Al jugador se le presentará un nuevo submenú interactivo que podría ser similar al siguiente:

```
Oráculo Whalum, te tengo que hacer una pregunta sobre la ubicación de
Princesa Peach:
    1.- Posición de Princesa Peach mayor que ...
    2.- Posición de Princesa Peach menor que ...
    3.- Te arriesgas al valor ...
```

  Así, el usuario podrá consultar a Whalum si Princesa Peach se encuentra en una posición mayor, menor o igual a un valor **j** que también deberá ingresar. Whalum responderá en consecuencia, siempre con la verdad. No es necesario que Whalum y el personaje elegido tengan que encontrarse.

  Si el jugador elige la opción 1 o la 2, luego de que Whalum responda se vuelve a presentar el mismo menú para seguir investigando sobre la ubicación de Princesa Peach.

  Cuando el jugador elige la opción 3, se arriesga a un resultado definitivo: Si acierta el juego continúa volviendo al modo tercera persona pero si pierde la partida finaliza con Game Over.

- Si acertaron la ubicación de la torre, ésta se visualizará en pantalla donde corresponda con una T. Luego, Mario y Luigi deberán orientarse en el sentido que les permita llegar hasta donde está Princesa Peach y comenzar a avanzar hasta llegar a la torre.

- Debe haber una espera entre cada movimiento para que se pueda visualizar la simulación.

- En el camino del Reino Champiñón hay flores que Mario y Luigi irán recogiendo. Mario sólo juntará aquellas que están en las casillas de **j** con valor absoluto múltiplo de 2 y 7, mientras que Luigi solo recogerá aquellas que están en casillas de **j** con valor absoluto múltiplo de 3 y 5. No es obligación ver las flores en la pantalla pero sí debe ir refrescándose a medida que los hermanos B se van moviendo hacia la torre.

- Una vez que Mario y Luigi llegan al rescate de Princesa Peach ella evaluará quién le regaló más flores y se irá con él a recorrer el Reino Champiñón. Si hay empate, elegirá en forma aleatoria lanzando una moneda al azar: Si cae cruz gana Mario, si cae cara gana Luigi.

- El programa deberá indicar con quién se queda Princesa Peach.

Requisitos:

- El Reino Champiñón debe modelarse con una matriz.
- Personaje deberá implementarse como TDA.
- El juego deberá estar criteriosamente modularizado.
- Se deberán realizar las validaciones pertinentes.