LABORATORIO IV

GONZALO DE VARONA

A00358687

ALGORTIMOS Y PROGRAMACIÓN 2

UNIVERSIDAD ICESI

OCTUBRE 19 DE 2019

**Objetivos**

**Unidad 4: Estructuras y Algoritmos Recursivos**

OE4.1 Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.

OE4.2 Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.

OE4.3 Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.

OE4.4 Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

**Enunciado**

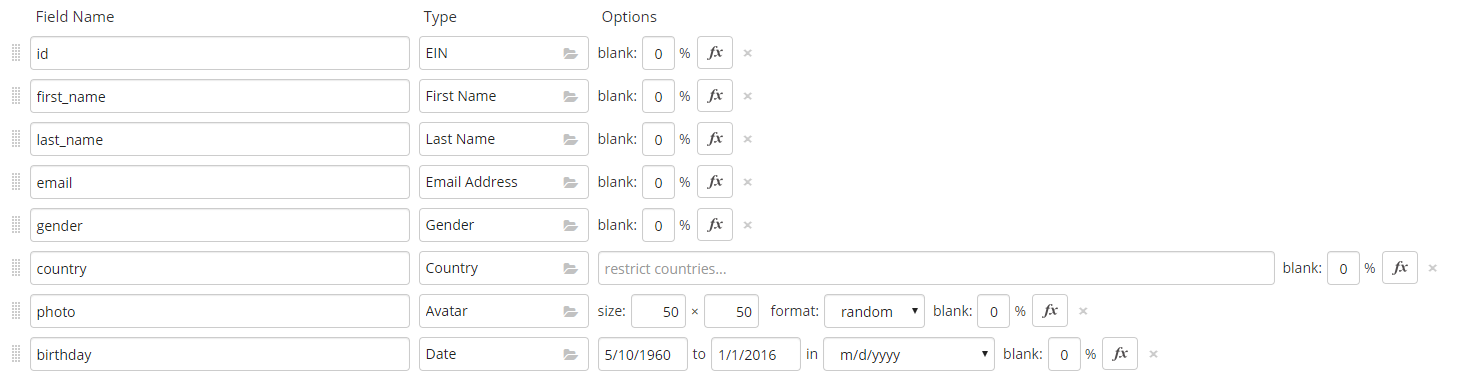
La [IV Copa Panamericana de Voleibol Masculino Sub-21](https://es.wikipedia.org/wiki/Copa_Panamericana_de_Voleibol_Masculino_Sub-21_de_2019) se celebrará del 5 al 11 de mayo de 2019 en Lima, (Perú) y la federación tiene la necesidad de registrar algunos datos de los espectadores que desean asistir al evento y por motivo de algunas labores de logística durante el evento, el personal de registro y control estará permanentemente realizando búsquedas de espectadores y participantes en el software, utilizando como criterio el id de registro del evento.

Usted ha sido elegido para desarrollar un programa que permita leer la información de los posibles asistentes al evento desde un archivo de texto delimitado por comas y cargar la información de manera que se puedan hacer búsquedas a través del identificador del registro del evento.

El programa debe permitir leer el archivo de texto con la información de todos los posibles participantes. Esta información se cargará en una estructura de datos de tipo árbol binario de búsqueda (ABB) cuya relación de orden estará dada por el id de registro del evento. Una vez cargada esta información en el árbol de espectadores, se seleccionará aleatoriamente el 50% de esos espectadores para agregarlos a una lista doblemente enlazada, que será la estructura de datos que almacenará los inscritos al evento o participantes al evento.

Luego de la lectura del archivo y la creación de las dos estructuras de datos: una para los posibles participantes y otra para los inscritos al evento, se debe permitir hacer búsquedas por id, para cada una de las estructuras creadas. Su programa debe indicar, en cualquier caso, si el espectador se encuentra o no entre los buscados así como el tiempo que tardó la búsqueda. Si el espectador fue encontrado, deben visualizarse sus datos en un panel a la izquierda.

Los datos de entrada al programa pueden ser generados utilizando la herramienta web [**https://www.mockaroo.com**](https://www.mockaroo.com/)que permite generar datos aleatorios de forma personalizada. La configuración y orden de los datos generados debe ser la que se muestra a continuación:



Se debe generar el máximo de filas posible para hacer buenas pruebas y el formato en que se debe exportar debe ser CSV el cual delimita los archivos de texto con coma por defecto.

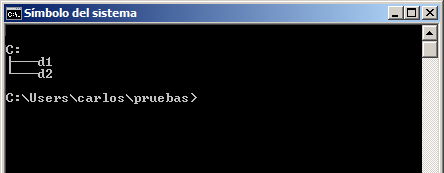


Ya que el máximo es 1000, realice pruebas del tiempo de la consulta para archivos de entrada mas grande como 100000, por ejemplo (generando 100 veces 1000 y uniéndolos). Compare los tiempos de las búsquedas cuando se realizan sobre todos los programadores y los participantes.

La interfaz del programa debe cumplir con los lineamientos:

* Permite cargar los datos de un archivo plano (ejemplo: archivo.csv) y confirmar la correcta carga, la ruta que se entra por teclado puede ser relativa o absoluta, debe especificarle al usuario final cual debe ingresar.
* Buscar espectador por id, debe cuantificar cuánto tiempo tarda la búsqueda y mostrar los datos del espectador (la foto en esta versión no será visualizada).
* Buscar participantes por id, debe cuantificar cuánto tiempo tarda la búsqueda y mostrar los datos del participante (la foto en esta versión no será visualizada).

El programa también contará con una opción que podrá visualizar los datos de las dos estructuras de un país determinado, donde la estructura del árbol binario de búsqueda se verá como un árbol y la lista se visualizará en forma consecutiva. La estructura árbol puede ser vista de la siguiente manera:



Donde c: es la raíz del árbol y d1 y d2 son hijos de c:,para cada opción se visualizará sólo los datos del país seleccionado (puede ser el componente JOptionPane de Java), eso quiere decir, que para la estructura de espectadores el árbol pintado solo debe tener espectadores de la ciudad seleccionada e igual para la estructura de los participantes.

**Entregables.** **1.** Requerimientos. **2.** Diagrama de clases de modelo e interfaz (no generado automáticamente). **3.** Implementación completa de todos los requerimientos en Java. **4.** Diseño de casos de pruebas unitarias de todas las funcionalidades relacionadas con los árboles. **5.** Implementación de todas las pruebas unitarias diseñadas y corriendo perfectamente. **6**. Todos los métodos para administrar las estructuras deben ser implementados usando recursividad. **7**. La estructura de los espectadores deben ser un árboles binarios de búsqueda y los participantes una lista doblemente enlazada, tal como se explicaron en el tema visto en clase y no podrán usar otro tipo de contenedor como array, ArrayList, etc.

**Fecha de Entrega:** viernes 19 de octubre de 2019 a las 11:59 p.m. a través de Moodle.

**Integrantes**: el laboratorio debe realizarse individualmente.

|  |  |
| --- | --- |
| **Nombre** | **R.F. 1. Buscar un espectador** |
| **Resumen** | El usuario debe ingresar el nombre del espectador, para buscarlo en el sistema. |
| **Entradas** | |
| -ID del espectador | |
| **Resultados** | |
| Espectador encontrado. | |

|  |  |
| --- | --- |
| **Nombre** | **R.F. 2. Buscar un participante** |
| **Resumen** | El usuario debe ingresar el nombre del participante, para buscarlo en el sistema. |
| **Entradas** | |
| -ID del participante | |
| **Resultados** | |
| Participante encontrado. | |

|  |  |
| --- | --- |
| **Nombre** | **R.F. 3. Cargar archivos para el sistema** |
| **Resumen** | El usuario debe ingresar la ruta donde se encuentra el archivo csv, para cargar los datos en el sistema. |
| **Entradas** | |
| -Ruta del archivo | |
| **Resultados** | |
| Archivo cargado. | |

|  |  |
| --- | --- |
| **Nombre** | **R.F. 4. Mostrar tiempos de busqueda** |
| **Resumen** | El programa debe de mostrar los tiempos de busqueda de un espectador o de un participante |
| **Entradas** | |
| <ninguna> | |
| **Resultados** | |
| Tiempos de búsqueda mostrados. | |

|  |  |
| --- | --- |
| **Nombre** | **R.F. 5. Mostrar espectadores de un país en forma de árbol binario** |
| **Resumen** | El usuario debe ingresar el país de los espectadores, los cuales se mostraran en pantalla en forma de árbol binario. |
| **Entradas** | |
| -Nombre del país de los espectadores | |
| **Resultados** | |
| Espectadores de un país mostrados como un árbol binario. | |

|  |  |
| --- | --- |
| **Nombre** | **R.F. 5. Mostrar participantes de un país** |
| **Resumen** | El usuario debe ingresar el país de los participantes, los cuales se mostraran en pantalla en forma de una lista. |
| **Entradas** | |
| -Nombre del país de los participantes | |
| **Resultados** | |
| Participantes de un país mostrados como una lista. | |

**Diseño de casos de pruebas unitarias**

**Manager:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valor Entrada** | **Resultado** |
| Manager | inicializarCasillasLibres() | El escenario inicializa un buscaminas de nivel principiante que es una matriz 8x8 con diez minas ubicadas aleatoriamente en ella | - | Verdadero, porque al contar todas las casillas libres del la matriz son 54=8x8-10 |
| Manager | inicializarCasillasLibres() | El escenario inicializa una buscaminas de nivel intermedio que es una matriz 16x16 con 40 minas ubicadas aleatoriamente en ella r | - | Verdadero, porque al contar todas las casillas libres del la matriz son 216=16x16-40 |
| Manager | inicializarCasillasLibres() | El escenario inicializa un buscaminas de nivel experto que es una matriz de 16x30 con 99 minas ubicadas aleatoriamente en ella |  | Verdadero, porque al contar todas las casillas libres del la matriz son 381=16x30-99 |

**Objetivo:** Probar si el método de inicializar las casillas libres genera la cantidad de casillas libres para cada nivel de manera correcta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valor Entrada** | **Resultado** |
| Manager | inicializarPartida() | El escenario inicializa un buscaminas de nivel principiante con sus valores por defecto | - | Verdadero, puesto que el método solo puede trabajar con 3 niveles, principiante(1), intermedio(2), y experto(3) |
| Manager | inicializarPartida() | El escenario inicializa un buscaminas de nivel intermedio con sus valores por defecto | - | Verdadero , puesto que el método solo puede trabajar con 3 niveles, principiante(1), intermedio(2), y experto(3) |
| Manager | inicializarPartida() | El escenario inicializa un buscaminas de nivel experto con sus valores por defecto | - | Verdadero, puesto que el método solo puede trabajar con 3 niveles, principiante(1), intermedio(2), y experto(3) |

**Objetivo:** Probar si la partida se inicializa según alguna escala de dificultad dada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valor Entrada** | **Resultado** |
| Buscaminas | generarMinas() | El escenario inicializa un buscaminas de nivel principiante con sus valores por defecto, ubicando sus minas correspondientes de manera aleatoria | - | Verdadero, puesto que el contador de minas es el mismo con el numero de minas que debe haber en el nivel principiante |
| Buscaminas | generarMinas() | El escenario inicializa un buscaminas de nivel intermedio con sus valores por defecto, ubicando sus minas correspondientes de manera aleatoria | - | Verdadero, puesto que el contador de minas es el mismo con el numero de minas que debe haber en el nivel intermedio |
| Buscaminas | generarMinas() | El escenario inicializa un buscaminas de nivel experto con sus valores por defecto, ubicando sus minas correspondientes de manera aleatoria | - | Verdadero, puesto que el contador de minas es el mismo con el numero de minas que debe haber en el nivel experto |

**Objetivo:** Probar si el método de generar minas, genera la cantidad de minas de manera aleatoria correspondiente a cada nivel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Valor Entrada | Resultado |
| Buscaminas | resolver(); | El escenario inicializa un buscaminas con nivel principiante con los valores por defecto | - | Verdadero, puesto que marca todas las casillas del buscaminas como seleccionadas y las muestra por pantalla |
| Buscaminas | resolver(); | El escenario inicializa un buscaminas con nivel intermedio con los valores por defecto | - | Verdadero, puesto que marca todas las casillas del buscaminas como seleccionadas y las muestra por pantalla |
| Buscaminas | Resolver(); | El escenario inicializa un buscaminas con nivel experto con los valores por defecto | - | Verdadero, puesto que marca todas las casillas del buscaminas como seleccionadas y las muestra por pantalla |

**Objetivo:** Probar si el método resuelve la matriz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valor Entrada** | **Resultado** |
| Buscaminas | abrirCasilla(); | El escenario inicializa un buscaminas de nivel principiante/intermedio/experto con sus valores por defecto | La casilla -1,-1 | Falso, puesto que esas coordenadas para una casilla no existen |
| Buscaminas | abrirCasilla(); | El escenario inicializa un buscaminas de nivel principiante/intermedio/experto con sus valores por defecto | La casilla 300,200 | Falso, puesto que esas coordenadas para una casilla exceden los límites de la matriz, por tanto no existen |
| Buscaminas | abrirCasilla(); | El escenario inicializa un buscaminas de nivel principiante/intermedio/experto con sus valores por defecto | La casilla  -1,200 | Falso, puesto que esas coordenadas para una casilla no existen |
| Buscaminas | abrirCasilla(); | El escenario inicializa un buscaminas de nivel principiante/intermedio/experto con sus valores por defecto | La casilla 1,1 | Verdadero, puesto que la casilla siempre existe y es posible destaparla |

**Objetivo:** Probar si el método abre las casillas dentro de una matriz especifica a una dificultad.

**Casilla:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores entrada** | **Resultado** |
| Casilla | Casilla(int) | El escenario inicializa una casilla como mina y los valores por defecto del constructor | 100 | Verdadero, se crea la casilla de tipo mina junto con sus valores por defecto |
| Casilla | Casilla(int) | El escenario inicializa una casilla como libre y los valores por defecto del constructor | 50 | Verdadero, se crea la casilla de tipo mina junto con sus valores por defecto |

**Objetivo:** Probar si el método está creando correctamente las minas y las casillas libres.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores entrada** | **Resultado** |
| Casilla | esMina() | El escenario inicializa una casilla como mina y los valores por defecto del constructor | - | Verdadero puesto que la casilla es de tipo mina |
| Casilla | esMina() | El escenario inicializa una casilla como libre y los valores por defecto del constructor | - | Falso, puesto que la casilla es de tipo libre. |

**Objetivo:** Probar si el método reconoce cuando una casilla es mina está libre.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores entrada** | **Resultado** |
| Casilla | modificarValor(int) | El método modifica el valor de minas que una casilla libre tiene a su alrededor | Valor de minas que una casilla libre tiene a su alrededor | Verdadero, ya que se hace una comparación se hace con el mismo numero que se modifico |
| Casilla | modificarValor(int) | El método modifica el atributo valor de una casilla | Cualquier n>0 | Falso, ya que las minas siempre deben tener como valor -1 |

**Objetivo:** Probar si el método modifica el atributo valor para cualquier casilla

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores Entrada** | **Resultado** |
| Casilla | mostrarCasillaValor() | El método inicializa una casilla como Libre y sus valores por defecto | - | Verdadero, ya que la casilla es de tipo mina y muestra la cadena “ - ” |
| Casilla | mostarCasillaValor() | El método inicializa una casilla como Mina y sus valores por defecto | - | Verdadero, ya que la casilla es de tipo mina y muestra la cadena “ \* ” |

**Objetivo:** Probar si el método genera la cadena de String esperada según el atributo tipo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores entrada** | **Resultado** |
| Casilla | destapar() | El método destapa una casilla mina | - | Verdadero, puesto que la casilla puede destaparse |
| Casilla | destapar(); | El método destapa una casilla libre | - | Verdadero, puesto que la casilla puede destaparse |
| Casilla | destapar(); | El método destapa una casilla donde seleccionado = true | - | Falso, puesto que la casilla ya estaba destapada |

**Objetivo:** Probar si el método resuelve la partida