TÉCNICAS DE SIMULACIÓN EN COMPUTADORAS **CICLO 01/2022**

DESCRIPCIÓN DE PROYECTO #1: ESTRUCTURAS DE DATOS

Objetivo:

Fortalecer las habilidades de manipulación de estructuras de datos estáticas y dinámicas utilizando el lenguaje de programación C++.

Características:

- El proyecto debe ser desarrollado en equipos de 3-4 estudiantes.
- El proyecto debe ser desarrollado exclusivamente en el lenguaje C++.
- No se permite el uso de librerías especializadas (e.g. *vector*).
- No se permite uso de interfaz gráfica, ejecución exclusiva a nivel de consola.

Arquitectura del Proyecto:

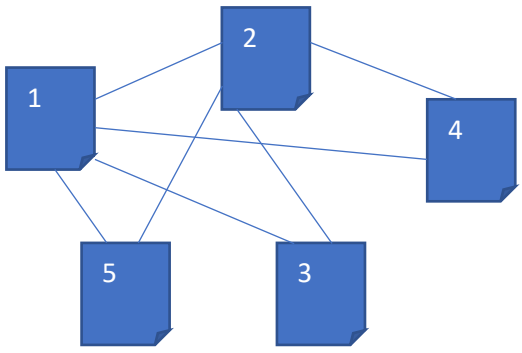
Nivel 1:

La aplicación manejará un Grafo G, formado por N_G nodos ($5 \leq N_G \leq 25$), donde cada nodo tendrá C_G conexiones ($1 \leq C_G \leq 24$), es decir, puede llegar a estar conectado con cada uno de los otros nodos, pero debe estar conectado con al menos un nodo.

Las conexiones de un nodo deberán definirse de forma aleatoria:

- Se define aleatoriamente cuántas conexiones tendrá.
- Por cada conexión, se define aleatoriamente con cuál de los nodos existentes será la conexión.

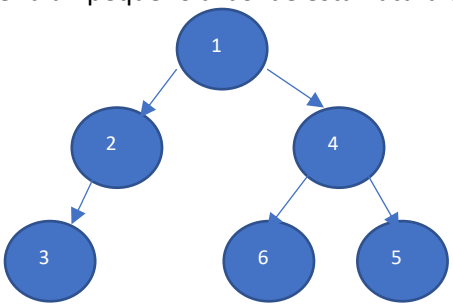
A continuación se muestra un posible grafo de 5 nodos:



Como puede apreciarse en el diagrama, cada nodo tendrá un número identificador que lo diferencia de todos los demás nodos del grafo.

Nivel 2:

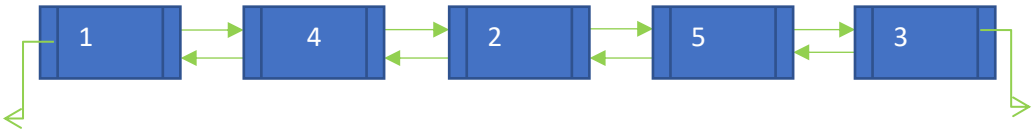
Cada nodo del grafo G contendrá un Árbol Binario de Búsqueda T, formado por N_T nodos ($5 \leq N_T \leq 50$), donde el criterio de ordenamiento será explicado en el siguiente apartado. La siguiente imagen es una muestra de cómo se vería un pequeño árbol de esta naturaleza:



Similar al diagrama del grafo, en la figura anterior el número representa un identificador para cada nodo que debe ser único en todo el árbol.

Nivel 3:

Cada nodo de un árbol T contendrá una Lista enlazada doble L, formada por N_L nodos ($5 \leq N_L \leq 75$), similar a la siguiente figura:

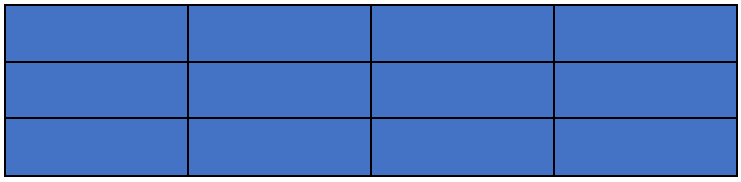


Una vez más, los números representan identificadores, y no necesariamente representarán una secuencia o un orden establecido.

Cada lista L tendrá una longitud inicial, ya que posteriormente podrá ir adquiriendo o perdiendo nodos, y es **esta longitud inicial la que se utilizará como criterio de ordenamiento para el árbol del Nivel 2**.

Nivel 4:

Cada nodo de una lista doble L contendrá una Matriz M, un arreglo estático bidimensional, de dimensiones F x C ($10 \leq F, C \leq 15$):



En este caso no se utilizará un identificador para las celdas, el sistema de índices *(fila, columna)* será suficiente para diferenciarlas.

Nivel 5:

Cada celda de una matriz M contendrá un Arreglo estático unidimensional A, de longitud N_A ($7 \leq N_A \leq 100$):

Similar al nivel anterior, no es necesario el uso de identificadores, ya que el sistema de indexado (las posiciones del arreglo) será suficiente para diferenciar las celdas.

Nivel 6:

Cada celda de un arreglo A contendrá un Objeto O, que puede ser creado ya sea a partir de una clase o de un struct, que debe contener al menos 5 datos de interés, lo que lleva a la siguiente sección…

Información a almacenar:

Es hora de brindar tributo a los Nerds/Geeks/Otaku, por lo que cada equipo de trabajo debe seleccionar una de las siguientes franquicias:

- Pokémon.
- Digimon.
- Touhou.
- Gundam.
- Yu-Gi-Oh!
- Magic: The Gathering.
- Flesh and Blood.
- Hololive.
- DC Comics.
- Marvel Comics.
- Doctor Who.
- Star Wars.
- Star Trek.
- The Lord of the Rings.
- Harry Potter.
- Game of Thrones.
- The Name of the Wind (The Kingkiller Chronicle).
- The Withcer.
- Dune.
- Ender's Game.
- Stephen King's Universe.
- Malazan: Book of The Fallen.
- Norse Mythology.
- Greek Mythology.
- The Legend of Zelda.
- FromSoftware Inc. (Dark Souls, Bloodborne & Elden Ring).
- Bleach.
- Naruto & Boruto.
- One Piece.
- Dragon Ball.
- Hunter x Hunter.
- Persona.
- JoJo's Bizarre Adventure.
- League of Legends (Riot universe).
- Final Fantasy.
- World of Warcraft & Hearthstone.
- Street Fighter.
- Mortal Kombat.
- Attack on Titan.
- Sword Art Online.
- Fate Grand/Order (Type-Moon).
- Monster Hunter.

Cada equipo debe utilizar una franquicia diferente, y una vez seleccionada debe ser reportada al profesor de la materia para reservarla, por lo que se recomienda que eso sea lo más pronto posible.

Una vez seleccionada, aprobada, y asignada oficialmente la franquicia del equipo, se deberá recopilar la mayor cantidad posible de información al respecto, el objetivo es: **tener suficiente información para poder definir una jerarquía de 5 niveles**.

Una vez identificadas, hacer un "mapeo" con los niveles de la arquitectura del proyecto, por ejemplo: si la temática fuera "El Salvador" se tendría un esquema como el siguiente:

```
Nivel 1 (Grafo) -> Regiones

Nivel 2 (Árboles) -> Departamentos

Nivel 3 (Listas) -> Municipios

Nivel 4 (Matrices) -> Ciudades

Nivel 5 (Arreglos) -> Colonias
```

En donde los Objetos internos de todo el sistema serían instancia de una Class/Struct "Colonia".

Funcionamiento:

La aplicación debe comenzar con un menú con las siguientes opciones:

- Visualizar información almacenada.
- Agregar información al sistema.
- Modificar información almacenada.
- Remover información del sistema.
- Salir.

La opción "Salir" simplemente muestra un mensaje de despedida y cierra la aplicación. En cuanto a las otras opciones, se detalla a continuación lo requerido.

Visualizar información almacenada:

Lo primero que debe hacerse en este caso es mostrar el nodo inicial del grafo G de una manera similar a la siguiente:

```
Grafo G:

Nodo { 1 }

Conexiones: { 2 }, { 4 }, { 3 }, { 5 }
```

Esta etapa debe permitir tres operaciones:

- Moverse a uno de los nodos conectados al nodo actual.
- Explorar el contenido del nodo actual.
- Regresar a la etapa anterior.

De ingresar a un nodo del grafo, se muestra el nodo raíz de su correspondiente árbol de una forma similar a la siguiente:

```
Grafo G -> Nodo { 1 } -> Árbol T:
```

```
Nodo { 1 }

Izquierda: { 2 }        Derecha: { 4 }
```

Esta etapa debe permitir 4 operaciones:

- Subir en el árbol.
- Bajar en él árbol, ya sea por la izquierda o por la derecha.
- Explorar el contenido del nodo actual.
- Regresar a la etapa anterior.

Y así sucesivamente… Es decir, cada vez que se explora un nodo, debe mostrar su contenido, sus conexiones, y permitir al usuario moverse en la estructura actual o volver a la etapa anterior.

Para el caso de las estructuras de datos estáticas, se mostrarán en su totalidad, y se le permitirá al usuario ingresar los índices correspondientes a la celda/posición de su interés.

En el nivel más profundo, simplemente se mostrará el contenido del objeto actual (su información).

Agregar información al sistema:

En esta opción se le pedirá al usuario una cadena de caracteres con el siguiente formato:

GGTTLLFFCCAA

- **GG**: Primeros dos caracteres de la cadena. Corresponderá al ID del nodo en el grafo en el que se quiere ingresar información, e.g. “01” es nodo 1, “25” es nodo 25.
- **TT**: Tercer y cuarto caracter de la cadena. Corresponderá al ID del nodo en el árbol del nodo GG del grafo en el que se quiere ingresar información.
- **LL**: Quinto y sexto caracter en la cadena. Corresponderá al ID del nodo en la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere ingresar información.
- **FF** y **CC**: Séptimo y octavo, y noveno y décimo caracter de la cadena, respectivamente. Corresponderá a la fila FF y columna CC que identifican la celda de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere ingresar información.
- **AA**: Undécimo y duodécimo caracter de la cadena. Corresponden a la posición AA del arreglo de la celda (FF, CC) de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere ingresar información.

Si al llegar al fondo en la posición AA ya hay un Objeto almacenado, se sobreescribe el existente.

Luego, el usuario puede detenerse en cualquier nivel de la estructura colocando ‘NN’ en las etapas posteriores a la de su interés, ejemplos:

- **011207NNNNNN**: En este caso el usuario se detiene en el nodo 07 de la lista del nodo 12 del árbol del nodo 01 del grafo. En este caso se ha colocado ‘NN’ para FF, para CC, y para AA. Esto se interpreta como que el usuario quiere sobreescribir toda la matriz que se encuentra en esta posición, por lo que se le deberá solicitar toda la información pertinente.
- **01NNNNNNNNNN**: En este caso el usuario se detiene en el nodo 01 del grafo, queriendo sobreescribir todo lo que se encuentra ahí.

Modificar información almacenada:

En esta opción se le pedirá al usuario una cadena de caracteres similar a la de la sección anterior:

GGTTLLFFCCAA

- **GG**: Primeros dos caracteres de la cadena. Corresponderá al ID de un nodo en el grafo.
- **TT**: Tercer y cuarto caracter de la cadena. Corresponderá al ID de un nodo en el árbol del nodo GG del grafo.
- **LL**: Quinto y sexto caracter en la cadena. Corresponderá al ID de un nodo en la lista del nodo TT del árbol del nodo GG del grafo.
- **FF** y **CC**: Séptimo y octavo, y noveno y décimo caracter de la cadena, respectivamente. Corresponderá a la fila FF y columna CC que identifican la celda de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo.
- **AA**: Undécimo y duodécimo caracter de la cadena. Corresponden a la posición AA del arreglo de la celda (FF, CC) de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo.

En esta caso no se permite detenerse en niveles previos, solo se permite llegar hasta el fondo.

Una vez se llega al fondo, la aplicación debe permitir al usuario cambiar los datos del Objeto ahí almacenado, esto deberá corresponder con la Class/Struct diseñada por el equipo de trabajo.

Remover información del sistema:

En esta opción se le pedirá al usuario una cadena de caracteres con el siguiente formato:

GGTTLLFFCCAA

- **GG**: Primeros dos caracteres de la cadena. Corresponderá al ID del nodo en el grafo en el que se quiere eliminar información.
- **TT**: Tercer y cuarto caracter de la cadena. Corresponderá al ID del nodo en el árbol del nodo GG del grafo en el que se quiere eliminar información.
- **LL**: Quinto y sexto caracter en la cadena. Corresponderá al ID del nodo en la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere eliminar información.
- **FF** y **CC**: Séptimo y octavo, y noveno y décimo caracter de la cadena, respectivamente. Corresponderá a la fila FF y columna CC que identifican la celda de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere eliminar información.
- **AA**: Undécimo y duodécimo caracter de la cadena. Corresponden a la posición AA del arreglo de la celda (FF, CC) de la matriz del nodo LL de la lista del nodo TT del árbol del nodo GG del grafo en el que se quiere eliminar información.

Al llegar al fondo en la posición AA, se destruye el Objeto existente, si hubiese.

Una vez más, el usuario puede detenerse en cualquier nivel de la estructura colocando ‘NN’ en las etapas posteriores a la de su interés, ejemplos:

- **0503100602NN**: En este caso el usuario se detiene en la celda (06,02) de la matriz del nodo 10 de la lista del nodo 03 del árbol del nodo 05 del grafo. En este caso se ha colocado ‘NN’ para AA. Esto se interpreta como que el usuario quiere eliminar todo el arreglo que se encuentra en esta posición.
- **2109NNNNNNNN**: En este caso el usuario se detiene en el nodo 09 del árbol del nodo 21 del grafo, queriendo eliminar toda la lista que se encuentra en esta posición.

Aspectos a considerar en **todas** las funcionalidades:

- Ser creativos con la presentación, procurar ser lo más llamativo y estético posible.
- Asegurarse de validar todo lo que necesite.
- Efectuarse un correcto manejo y limpieza de la memoria del sistema.

Entrega del proyecto:

- El proyecto se entregará a más tardar el día **sábado 30 de abril a las 23:59**.
- La entrega está configurada como un **Group Assignment**.
- Todos los miembros de todos los equipos deben enlazar su cuenta de GitHub.
- Un miembro del equipo debe crear el grupo en Github Classroom.
- El resto de miembros del equipo deberán unirse al grupo creado.
- La entrega deberá ser **por medio de un release**.

El repositorio de la entrega debe contener:

- README con los integrantes del equipo de trabajo y notas o consideraciones que consideren relevantes.
- Documento Word (o equivalente) que explique brevemente la estructura de la franquicia asignada y describa los niveles identificados, así como el contenido de la Class/Struct diseñada.
    - Mínimo 1 página y máximo 2 páginas.
- Documento con instrucciones especiales para la compilación (si hubiese).
- Código fuente, debidamente estructurado en carpetas. Adicionalmente, cada archivo debe estar debidamente comentado.
- Un “script de exploración” para su aplicación, es decir:
    - Pasos de llenado sugeridos.
    - Pasos de exhibición de las funcionalidades.