DIIT Departamento de Ingeniería e Investigaciones Tecnológicas	**ALGORITMOS Y ESTRUCTURAS DE DATOS**	Universidad Nacional de La Matanza

Guía Complementaria de Actividades Prácticas No Obligatorias

Carrera: Ingeniería Informática

Plan: 2023

Materia: Algoritmos y Estructuras de Datos (3640)

Año Lectivo: 2023

Contenido:
Descripción de la materia
Programa
Bibliografía
Cronograma
Guía de Trabajos Prácticos

Docentes:
Álvarez, Erik
Cacho Mendoza, Ariel
Calaz, Ezequiel
Cuesta, Cristian
Ghigo, Paola
González, Giselle
Guatelli, Renata
Jordi, Brian
Martínez, Pablo
Mendoza, Matías
Pan, Néstor
Pezzola, Federico
Soligo, Pablo
Uran Acevedo Jónathan

Descripción de la asignatura

Esta asignatura introduce al estudiante en la implementación y desarrollo de Tipos de Datos Abstractos (TDAs). Presenta las estructuras de datos lineales (pilas, colas, listas) y arborescentes (árboles binarios); se profundiza en el diseño iterativo y en el diseño recursivo, tanto en el razonamiento sobre la corrección de un diseño dado como en la detección y mejora de soluciones ineficientes. Se presentan diversas implementaciones de estructuras de datos lineales y arborescentes. Se realizan implementaciones realistas de las mismas, usando primitivas independientes del tipo de dato, tanto estos sean tipos de datos simples como recursivos, para versiones estáticas como dinámicas. Se trabaja de forma tal que el estudiante comprenda su representación en memoria.

Metodología de enseñanza

Se utilizan metodologías activas de enseñanza, especialmente aprendizaje basado en problemas (ABP). Se presenta cada unidad temática introduciendo los conceptos fundamentales realizando analogías con ejemplos reales, que permite relacionar los contenidos de la materia con las herramientas habituales de trabajo.

Los contenidos de la asignatura se presentan de forma iterativa e incremental, de forma que le permitan al estudiante, construir sus propios procedimientos para resolver una situación problemática, lo que implica que sus ideas puedan verse modificadas y de esta forma siga construyendo nuevos conocimientos.

La materia tiene una fuerte carga práctica. En la misma se resuelven problemas, desarrollando pequeños sistemas, aplicando lo visto en las clases, trabajando de forma colaborativa, simulando un entorno de trabajo real.

Se motiva a los estudiantes en el uso de los foros de la plataforma MIeL y recursos de Teams, para la resolución de dudas, tanto de conceptos teóricos como prácticos, permitiendo desarrollar las capacidades de comunicación y afianzar el uso del lenguaje técnico. Además, la cátedra cuenta con soporte digital de los contenidos, que los estudiantes pueden consultar luego de haber asistido a la clase.

Objetivos de aprendizaje

A través de esta asignatura, el estudiante habrá adquirido los conocimientos necesarios y suficientes para estar en condiciones de:

Objetivos Generales:
La materia se desarrolla teniendo en cuenta los siguientes objetivos generales:

- analizar, plantear y resolver situaciones problemáticas.
- organizar y planificar su trabajo.
- hacer transferencia de los conocimientos teóricos a la práctica.
- adquirir la capacidad de trabajar en equipo.
- identificar un problema a partir de una situación problemática presentada.
- diseñar un algoritmo eficiente para la resolución de una situación problemática analizada.
- desarrollar un algoritmo utilizando un lenguaje de programación.
- plantear casos de prueba de forma tal de ver los casos generales y particulares de cada situación problemática planteada.
- expresar los contenidos teóricos de la materia y su vinculación con situaciones de la vida real.
- aplicar los principios de la Programación Estructurada.
- detectar la fuerte vinculación de esta asignatura con materias de años anteriores y posteriores del plan de carrera.

- integrar grupos de trabajo, potenciando su propio aprendizaje a través de la interacción y cooperación con sus pares.
- utilizar con fluidez el lenguaje técnico relacionado con la materia.

Objetivos Específicos:

- Diseñar e implementar como TDAs, estructuras de datos lineales y árboles binarios, utilizando diferentes implementaciones independientes del dato que se almacene en dicha estructura.
- Desarrollar e implementar la solución que mejor se adapte de acuerdo con los requisitos establecidos.
- Seleccionar las estructuras de datos más adecuadas para la resolución de un problema dado.
- Detectar las ventajas y desventajas de la programación recursiva.

Contenidos mínimos

Estructuras de datos genéricas. Tipo de Dato Abstracto. Recursivos. Representación en memoria. Recursividad. Estructuras. Pila, Cola, Lista y Árbol. Estrategias de implementación. Diseño e implementación de variantes de usos y aplicaciones.

Competencias a desarrollar

Genéricas

- **Competencias tecnológicas**
  - Identificar, formular y resolver problemas de ingeniería.
  - Concebir, diseñar y desarrollar proyectos de ingeniería.
  - Gestionar, planificar, ejecutar y controlar proyectos de ingeniería.
  - Utilizar de manera efectiva las técnicas y herramientas de aplicación en la ingeniería.

- **Competencias sociales, políticas y actitudinales**
  - Desempeñarse de manera efectiva en equipos de trabajo.
  - Comunicarse con efectividad.
  - Actuar con ética, responsabilidad profesional y compromiso social, considerando el impacto económico, social y ambiental de su actividad en el contexto local y global.
  - Aprender en forma continua y autónoma.
  - Actuar con espíritu emprendedor.
  - Capacidad de análisis y síntesis.
  - Toma de decisiones.
  - Razonamiento crítico.

Específicas

- Capacidad de resolución de problemas aplicando conocimientos de matemáticas, ciencias e ingeniería.
- Tener capacidad para realizar la formalización y especificación de problemas reales cuya solución requiere el uso de la informática.
- Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo las estructuras de datos más adecuadas, administrando de forma eficiente los recursos disponibles.

Programa analítico	
Unidad 1	**Tipo de dato abstracto (TDA) - Tipo recursivos de datos** Concepto de Tipo abstracto de datos. Introducción al uso de tipo recursivos de datos. Gestión de memoria dinámica. Implementación de estructuras de datos enlazadas mediante tipos recursivos de datos. Algoritmos iterativos y recursivos para resolver problemas de búsqueda y recorrido en estructuras de datos enlazadas accediendo directamente a la representación basada en nodos y punteros a nodos.
Unidad 2	**Estructura de datos Pila como un TDA** Asignación dinámica de memoria vs. asignación estática de memoria. Primitivas para el manejo de Pilas, compatibilidad de primitivas entre la implementación estática y dinámica de Pilas. Su relación con la recursividad.
Unidad 3	**Estructura de datos Cola como un TDA** Asignación dinámica de memoria vs. asignación estática de memoria. Primitivas para el manejo de Colas, compatibilidad entre la implementación estática y dinámica de Colas. Su relación con el "buffer" de teclado.
Unidad 4	**Estructura de datos Lista como un TDA** Primitivas para el manejo de Listas. Creación, inserción, ordenamiento, búsqueda, eliminación, etc., con asignación dinámica de memoria. Listas circulares, su importancia en la implementación de colas y su parecido y diferencia con la implementación de pilas. Listas doblemente enlazadas. Diferentes tipos de implementación. Funciones map, filter y reduce.
Unidad 5	**Recursividad** Concepto de recursión. Estructura de funciones recursivas. Ejecución de funciones recursivas. Envoltura para funciones recursivas. Tipos de recursividad. Eficiencia de la recursividad. Recursión versus Iteración.
Unidad 6	**Estructura de datos Árbol** Recursividad. Árbol binario, creación, recorridas EnOrden, PreOrden y PosOrden. Árbol binario de búsqueda, su relación con la búsqueda binaria en arrays. Determinación de altura, y otras funciones. Árboles AVL y balanceados, determinación. Diferentes tipos de implementación. Concepto de archivos de índice (.idx) Implementación de índices sobre un árbol binario de búsqueda. Ventajas de implementar índices sobre árboles binarios de búsqueda.

Planificación de actividades

Semana	Clase	Actividad	Tipo	Duración estimada	Unidad /des
Semana 1 14/08/2023	1	Presentación de los docentes del curso, breve explicación de las pautas generales de la materia. Breve repaso de conceptos adquiridos en “Tópicos de Programación”, TDA, Memoria dinámica, funciones genéricas. Recursividad. Concepto. Elementos básicos de las funciones recursivas. Envoltorios para funciones recursivas. Tipos de recursividad. Desarrollar funciones recursivas. Tipo recursivo de datos. Concepto de nodo.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	0 - 1 -5

Semana 2 21/08/2023	2	Estructura de datos Pila. Implementación como TDA sobre una estructura dinámica. Aplicaciones de Pilas, ejemplos sobre la práctica. **Presentación del trabajo práctico.**	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	2
Semana 3 28/08/2023	3	Estructura de datos Pila. Implementación como TDA sobre una estructura estática. Primitivas coherentes entre una implementación estática y dinámica. Aplicaciones de Pilas, ejemplos sobre la práctica.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	2
Semana 4 04/09/2023	4	**Aula:** Estructura de datos cola. Implementación como TDA sobre una estructura dinámica. Aplicaciones de Colas, ejemplos sobre la práctica.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	3
Semana 5 11/09/2023	5	**Aula:** Estructura de datos cola. Implementación como TDA sobre una estructura estática. Primitivas coherentes entre implementación estática y dinámica. Aplicaciones de Colas, ejemplos sobre la práctica.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	3
Semana 6 18/09/2023	6	Estructura de datos Lista. Implementación dinámica como TDA. Primitivas de inserción: al inicio, al final, en orden, por posición, etc.. Mención de su implementación estática. Aplicaciones de Listas, ejemplos sobre la práctica.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	4
Semana 7 25/09/2023	7	Estructura de datos Lista. Implementación dinámica como TDA. Primitivas de eliminación: al inicio, al final, en orden, por posición, etc.. Primitivas de ordenamiento. Aplicaciones de Listas, ejemplos sobre la práctica.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	4
Semana 8 02/10/2023	8	Explicación conceptual de listas circulares, implementación de colas y pilas con asignación dinámica de memoria en listas circulares, similitud y diferencia entre las primitivas de pila y cola dinámica en listas circulares. Primitivas coherentes con las ya vistas. Funciones map, filter y reduce.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	4
Semana 9 09/10/2023	9	Explicación conceptual de listas doblemente enlazadas. Desarrollo de primitivas para insertar, eliminar y ordenar listas doblemente enlazadas.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	4
Semana 10 16/10/2023	10	**Aula:** Desarrollar funciones recursivas de búsqueda, ordenamiento, de cadenas, de listas, etc. Comparar la implementación de	Teoría - Práctica en laboratorio	4 hs	5

		funciones recursivas e iterativas. Ventajas y desventajas de implementaciones recursivas vs. iterativas.	o virtual sincrónico		
Semana 11 23/10/2023	11	Estructura de datos Árbol. Árbol binario. Árbol binario de búsqueda. Primitivas. Recorridas. Mención de su compatibilidad con la implementación estática. Funciones para verificar árbol (AVL) semi balanceados, balanceados, completos. Similitud con búsquedas dicotómicas. Comparar implementaciones recursivas vs. iterativas. Presentación de funciones variadas (búsqueda de la clave de ordenamiento, altura, contar hojas, contar no hojas, contar nodos que cumplen una determinada condición, eliminar árbol, eliminar nodos, podar ramas, etc.).	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	6
Semana 12 30/10/2023	12	Concepto de índice. Archivo de índice (idx). Implementación de índices sobre un árbol binario de búsqueda. Crear archivo de índice. Regenerar archivo de índice.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	6
Semana 13 06/11/2023	13	Parcial 1 Entrega trabajo práctico	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	1 - 2 - 3 - 4 - 5 - 6
Semana 14 13/11/2023	14	Devolución parcial 1. Defensa trabajo práctico Ejercitación sobre todos los temas vistos hasta el momento. Consultas sobre el trabajo práctico.	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	1 - 2 - 3 - 4 - 5 - 6
Semana 15 20/11/2023	15	Recuperatorio Parcial 1 Entrega trabajo práctico	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	1 - 2 - 3 - 4 - 5 - 6
Semana 16 27/11/2023	16	Defensa trabajo práctico Entrega de notas finales	Teoría - Práctica en laboratorio o virtual sincrónico	4 hs	1 - 2 - 3 - 4 - 5 - 6

Evaluación

El proceso de evaluación consta de:

- un examen parcial presencial
- un trabajo práctico grupal. La aprobación del trabajo práctico conlleva la aprobación del código entregado y una defensa oral individual del mismo.
- un examen recuperatorio presencial.

Las evaluaciones podrán ser teórico - prácticas. o únicamente prácticas o teóricas.

Las evaluaciones teóricas constan de preguntas para desarrollar y/o preguntas del tipo opción múltiple y/o verdadero / falso, justificando la respuesta o no, detectar errores de un código, etc.

Las evaluaciones prácticas constan de la resolución de ejercicios en máquina. Desarrollando el código necesario para resolver el problema solicitado y el lote de pruebas correspondiente

Primera evaluación	Semana 13	teórico - práctica	4 hs
Aprobación TP Grupal	A partir de la semana 13	teórico - práctica	4 hs
Recuperatorio	Semana 15	teórico - práctica	4 hs

Bibliografía obligatoria [Disponibles en la Biblioteca Leopoldo Marechal, o con acceso digital]

Título	Autor	Editorial	Edición	Año
El Lenguaje de Programación C	Kernighan y Ritchie	Prentice Hall	2.ed.	1991
Cómo Programar en C / C++	Deitel y Deitel	Prentice Hall	2a. ed.	1994
Apuntes de cátedra	Docentes de la cátedra	Disponibles en MIeL		

Bibliografía complementaria recomendada [disponible en la Biblioteca Leopoldo Marechal, o con acceso digital]

Título	Autor	Editorial	Edición	Año
Data Structures and Program Design in C	Kruse, Leung y Tondo	Prentice Hall		
Estructuras de Datos con C y C++	Langsam, Augenstein y Tenembaum	Prentice Hall		
Código Limpio	Robert Cecil Martin	Anaya Multimedia		

Otros recursos obligatorios

Nombre	

Otros recursos complementarios

Nombre	
https://pythontutor.com/c.html#mode=edit	Permite visualizar en el navegador lo que la computadora está haciendo paso a paso mientras ejecuta un programa.
https://learngitbranching.js.org/	Una herramienta de visualización Git interactiva para educar y desafiar

Tema: TDA, Memoria dinámica, funciones genéricas. Recursividad. Concepto. Elementos básicos de las funciones recursivas. Envoltorios para funciones recursivas. Tipos de recursividad. Desarrollar funciones recursivas. Tipo recursivo de datos. Concepto de nodo.

Unidad: Ejercicios de repaso.

Objetivo: Revisión de los principales conceptos introducidos en la materia Tópicos de Programación.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:
1- Tiempo estimado de resolución: 7 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 1
Implemente un TDA Vector. Debe desarrollar una versión con memoria estática y otra con memoria dinámica. Debe implementar las primitivas:

- crear vector
- vector lleno
- vector vacío
- insertar elemento en orden
- ver elemento de una posición dada
- eliminar elemento
- destruir vector

Tenga en cuenta que la primitiva 'destruir' debe, según sea el caso, liberar la memoria reservada, o poner la cantidad de elementos en cero.
Contemple los casos frontera.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas.

Ejercicio 2
Implemente una función que se denomine ***calcular***. Esta función debe recibir al menos tres argumentos:

- Operando 1.
- Operando 2.
- Puntero a función. Por ahí recibe la función que resuelve la operación (sumar, restar, multiplicar y dividir).

Contemple los casos frontera.
Determine cómo se hará la devolución del resultado.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas.

Ejercicio 3
Diseñe e implemente un programa que cargue un TDA Vector a partir del contenido de un archivo de texto cuyo formato es el siguiente:

```
3
23
13
18
```

En donde la primera línea del archivo contiene un número que corresponde a la cantidad de registros que contiene el archivo. Para el ejemplo anterior, el archivo va a contener 3 (tres) números; y esos números van a ser 23, 13 y 18.
Contemple los casos frontera.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas.

Ejercicio 4

Modifique el Ejercicio 3 para que funcione con un archivo de texto cuyos registros correspondan a un tipo persona (tPersona) diseñado por usted.
Contemple los casos frontera.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas

Ejercicio 5

Modifique el Ejercicio 4 para que funcione con un archivo de texto cuyos registros correspondan a cualquier tipo de dato. ¿Necesita algún dato adicional?
Contemple los casos frontera.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas

Ejercicio 6

Modifique el Ejercicio 5 para que funcione con un archivo binario cuyos registros correspondan a cualquier tipo de dato. ¿Necesita algún dato adicional?
Contemple los casos frontera.
Defina y prepare el entorno para que se puedan probar las funcionalidades solicitadas

Ejercicio 7

Escriba una función recursiva que:

- Calcule el factorial de un número entero.
- Muestre el contenido de un array de char de a un carácter a la vez. Ej.: “Hola”
    - ‘H’ ‘o’ ‘l’ ‘a’
- Ídem anterior, mostrando en orden inverso.
- Muestre el contenido de un array de char de la siguiente forma. Ej.: “Hola”
    - Hola
    - ola
    - la
    - a
- Ídem anterior, mostrando en orden inverso.
    - a
    - la
    - ola
    - Hola
- Muestre el contenido de un array de char de la siguiente forma. Ej.: “Hola”
    - H
    - Ho
    - Hol
    - Hola
- Ídem anterior, mostrando en orden inverso.
    - Hola
    - Hol
    - Ho
    - H

- Dado un número entero lo muestre descompuesto en los dígitos que lo forman. Ej.: 1234
    - 4
    - 3
    - 2
    - 1
- Ídem anterior, mostrando en orden inverso.
- Dado un número entero lo muestre de la siguiente forma. Ej.: 1234
    - 1234
    - 123
    - 12
    - 1
- Ídem anterior, mostrando en orden inverso.
- Dado un número entero lo muestre de la siguiente forma. Ej.: 1234
    - 4
    - 34
    - 234
    - 1234
- Ídem anterior, mostrando en orden inverso.
- Dado un número entero, retorne la suma de sus dígitos.
- Dada una cadena que solo contiene dígitos, retorne la suma de los caracteres que representan dígitos.
- Muestre el contenido de un array de enteros en orden inverso, devolviendo la suma de todos los elementos.
- Ídem anterior, devolviendo la suma de los pares.
- Ídem anterior, devolviendo la suma de los que están en posiciones pares.
- Escriba versiones recursivas de las funciones de biblioteca <strlen>, <strchr> y <strrchr>.

Tema: Concepto de Tipo abstracto de datos. Introducción al uso de tipo recursivos de datos. Gestión de memoria dinámica. Implementación de estructuras de datos enlazadas mediante tipos recursivos de datos. Algoritmos iterativos y recursivos para resolver problemas de búsqueda y recorrido en estructuras de datos enlazadas accediendo directamente a la representación basada en nodos y punteros a nodos.

Unidad 1: **Tipo de dato abstracto (TDA) - Tipo recursivos de datos**.

Objetivo: Comprender el concepto y uso de un TDA. Comprender el concepto y uso de un Tipo de dato recursivo, en que casos se utilizan, sus ventajas y sus desventajas.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 7 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 1.1

Ya conoce al menos dos TDA, por ejemplo Vector y Fecha. Sugiera al menos dos nuevos TDA's teniendo en cuenta para qué se utilizarían y cuáles serían sus primitivas básicas.

Ejercicio 1.2

Elija una estructura de datos recursiva. Plantee su implementación. Tenga en cuenta cómo es su representación en memoria. Sugiera al menos dos nuevos TDA's con estructura recursiva teniendo en cuenta para qué se utilizarían y cuáles serían sus primitivas básicas.

Ejercicio 1.3

Grafique un Nodo y luego impleméntelo. Asegúrese de comprender cuales son cada uno de sus componentes y cómo deben gestionarse los punteros para acceder a cada una de sus partes.

Ejercicio 1.4

Grafique una *Lista Simplemente Enlazada* (cuando avancemos en la materia las trataremos en detalle en la unidad 4) utilizando dos o más Nodos como los que se planteó en el Ejercicio 1.3. Qué estrategia utilizaría para recorrer dicha lista?

Teniendo en cuenta que la lista deberá ser implementada en lenguaje C:

- ¿Cómo se crearía un nodo?
- ¿Cómo se enlazarían los nodos entre sí?
- ¿Cómo sería el manejo de los punteros para recorrer la lista y acceder a sus diferentes miembros (lista, nodos, información, etc) ?

Ejercicio 1.5

Grafique un *Árbol binario de Búsqueda* (cuando avancemos en la materia las trataremos en detalle en la unidad 6) utilizando dos o más Nodos como los que se planteó en el Ejercicio 1.3. Esos nodos servirían para implementar el árbol ? Qué estrategia utilizaría para recorrer el árbol ?

Teniendo en cuenta que la lista deberá ser implementada en lenguaje C:

- ¿Cómo se crearía un nodo?
- ¿Cómo se enlazarían los nodos entre sí?
- ¿Cómo sería el manejo de los punteros para recorrer el árbol y acceder a sus diferentes miembros (árbol, nodos, información, etc) ?

Tema: Asignación dinámica de memoria vs. asignación estática de memoria. Primitivas para el manejo de Pilas, compatibilidad de primitivas entre la implementación estática y dinámica de Pilas. Su relación con la recursividad.

Unidad 2: Estructura de datos Pila como un TDA

Objetivo: Comprender el concepto, funcionamiento y usos de un TDA Pila. Comprender sus diferentes implementaciones.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 14 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 2.1

Acorde con lo visto en clases, desarrolle la implementación estática de Pila, creando sus archivos "pila.h" y "pila.c", deje estos archivos en un subdirectorio "./estatica/". Para que resulte de uso más general el tipo de dato para la información declárelo en otro archivo junto con los prototipos de las funciones que permiten ingresar, mostrar, etc. esa información.

Ejercicio 2.2

Ídem para la implementación dinámica de Pila, creando sus archivos "pila.h" y "pila.c", en un subdirectorio "./dinamica/". Para la información, utilice lo mismo del punto anterior.

Ejercicio 2.3

La información de los puntos anteriores deberá constar de un código de producto, descripción, proveedor (alfanuméricos de 7, 15 y 15 respectivamente), fechas de compra y de vencimiento, cantidad, precios de compra y de venta. Los desarrollos de las funciones para el manejo de la información deben estar en sus propios archivos fuente.

Ejercicio 2.4

Escriba un programa que al comenzar lea (si lo puede abrir) un archivo de binario <"datos">, y lo cargue en una pila con implementación estática de memoria. El archivo debe ser cerrado al terminar la carga de la pila. A continuación, y valiéndose de una función de menú, que permita cargar más información en la pila, ver la información del tope de la pila, sacar de la pila, salir del menú. Al salir del menú, se terminará de cargar el archivo con la información que aún quede en la pila. Si el archivo resultara vacío, deberá ser eliminado. Pruebe repetidamente el programa, hasta que logre que el archivo quede con información.

Ejercicio 2.5

Ídem anterior, pero con implementación dinámica de memoria.

Ejercicio 2.6

Dado un archivo binario (tal como el <”datos”> antes indicado), proceda a ordenarlo valiéndose de dos pilas.

Ejercicio 2.7

Resuelva el cálculo de la suma de dos números enteros de muchos dígitos (30 o muchos más) haciendo uso de dos pilas en las que almacena sólo los dígitos. Tenga en cuenta que debe utilizar una tercera pila en la que irá cargando los resultados parciales. Compruebe que obtiene idénticos resultados con ambas implementaciones de Pila (estática y dinámica).

Tema: Asignación dinámica de memoria vs. asignación estática de memoria. Primitivas para el manejo de Colas, compatibilidad entre la implementación estática y dinámica de Colas. Su relación con el "buffer" de teclado.

Unidad 3: Estructura de datos Cola como un TDA

Objetivo: Comprender el concepto, funcionamiento y usos de un TDA Cola. Comprender sus diferentes implementaciones.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 14 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 3.1
Ídem [Ejercicio 2.1] para la estructura Cola.

Ejercicio 3.2
Ídem [Ejercicio 2.2] para la estructura Cola.

Ejercicio 3.3
Ídem [Ejercicio 2.4] pero cargando en una Cola con implementación estática.

Ejercicio 3.4
Ídem [Ejercicio 2.4] pero cargando en una Cola con implementación dinámica.

Ejercicio 3.5
Resuelva la simulación de la cola de espera en un cajero automático. Suponga que cada cliente demora en el mismo un tiempo aleatorio de 1, 3 ó, 5 minutos con igual probabilidad, y que los clientes llegan al mismo de a uno, con intervalo de arribo aleatorio de 1, 5 ó 9 minutos, con igual probabilidad. La simulación termina después que la cola queda vacía cinco veces.

Ejercicio 3.6
Se dispone de un archivo como el generado en el [Ejercicio 2.4]. Se desea generar dos nuevos archivos: en <"datos1"> con los registros cuya clave comience o termine con un carácter representativo de un dígito, pero en orden contrario al del archivo original, valiéndose de una Pila; y en <"datos2"> en el mismo orden en que estaban grabados.
El proceso debe ser resuelto con una única lectura del archivo de entrada, mostrando por pantalla cada registro leído, para luego a la vez que se genera cada archivo de salida, mostrar qué se graba en cada uno.

Tema: Primitivas para el manejo de Listas. Creación, inserción, ordenamiento, búsqueda, eliminación, etc., con asignación dinámica de memoria. Listas circulares, su importancia en la implementación de colas y su parecido y diferencia con la implementación de pilas. Listas doblemente enlazadas. Diferentes tipos de implementación. Funciones map, filter y reduce.

Unidad 4: Estructura de datos Lista como un TDA

Objetivo: Comprender el concepto, funcionamiento y usos de un TDA Cola. Comprender sus diferentes implementaciones.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 28 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 4.1
De acuerdo con lo visto en clases, desarrolle la implementación dinámica de Lista, creando sus archivos “lista.h” y “lista.c”, deje estos archivos en un subdirectorio “./dinamica/”. Para que resulte de uso más general el tipo de dato para la información declárelo en otro archivo junto con los prototipos de las funciones que permiten ingresar, mostrar, etc. esa información.

Ejercicio 4.2
Utilizando una lista, ordene el archivo <”datos1”> valiéndose de una **inserción en orden,** y el archivo <“datos2”> valiéndose de una función que inserte al comienzo de la lista para luego **ordenar la lista**. Los archivos deben resultar ordenados por fecha de vencimiento, y a igualdad del mismo por proveedor y clave.

Ejercicio 4.3
Implemente un programa que le permita cargar el archivo <”datos”> en una Lista dinámica. En la información a cargar no se debe incluir el nombre del proveedor.
Pruebe las siguientes alternativas:

a. **Insertar los nuevos nodos al final** de la lista, salvo que la clave ya estuviera cargada y la descripción coincide, con lo cual se acumula la cantidad, se retiene la última fecha de compra y la fecha de vencimiento más vieja, el mayor precio de compra y de venta; si la descripción no coincidiera, se genera un nuevo nodo. Eliminar todos los nodos cuya clave está más de una vez, mostrando su información por pantalla y grabándolos en un archivo de texto <“errores2”> (tienen distinta descripción). **Ordenar la lista** resultante, y luego grabar esta información en un nuevo archivo binario <”depurado”>.

b. **Insertar los nuevos nodos ordenados** por clave y a igualdad de clave por descripción, salvo que esta clave compuesta coincida con lo que se procede a **acumular igual** que antes. **Eliminar** todos los nodos cuya clave está más de una vez, mostrando su información por pantalla y grabándolos en un archivo de texto <”errores2”> (tienen distinta descripción). Grabar esta información en un nuevo archivo <”depurado2”>.

Ejercicio 4.4
Implemente y pruebe las siguientes funciones de listas en versiones para listas ordenadas/no ordenadas:

c. Función iterativa (buscar_cla) que **busque por una clave** en una lista y recupere la información del nodo con esa clave, devolviendo un indicador de éxito/fracaso en su cometido, y eliminando o no el nodo encontrado, según lo indique un argumento extra.
d. Ídem, recursiva.
e. Función iterativa (contar_cla) que busque y devuelva la cantidad de veces que encontró la clave, y eliminando o no los nodos encontrados, según lo indique un argumento extra.
f. Ídem, recursiva.
g. Función iterativa (buscar_cla_n) que busque por una clave en una lista y recupere la información de la ocurrencia n de la clave indicada.
h. Ídem, recursiva.

Ejercicio 4.5

Genere una matriz poco densa de FIL filas por COL columnas (con muchos ceros). A partir de la información cargada en la matriz, genere una lista simplemente enlazada con miembros de información para la fila, columna y valor (sólo si este es distinto de cero). Valiéndose de un menú que permita:

a- el ingreso de fila y columna e informe el valor correspondiente, buscándolo en la lista.
b- el ingreso de una fila, y muestre los elementos de la fila.
c- el ingreso de una columna, y muestre los elementos de la columna.
d- que muestre a partir de la lista los elementos de la matriz.

Tenga en cuenta para los puntos [a-], [b-] y [d-] que debe disponer de la lista ordenada de modo que la búsqueda sea óptima.

Ejercicio 4.6

Se dispone de un archivo de texto con el número y el nombre de cada agrupación que se presenta en las elecciones para elegir los congresales de una asociación civil sin fines de lucro; y además de un archivo binario en el que se almacenó el número de agrupación, de distrito y de región (un registro por cada voto electrónico emitido).

Se requiere un proceso que a partir de la lectura de ambos archivos almacene en arrays bidimensionales los nombres de las agrupaciones y el total de votos obtenidos por distrito.

A partir de los arrays, genere una lista (con inserción en orden), a fin de poder mostrar, al final del proceso, los nombres de las tres agrupaciones que obtienen mayor cantidad de votos para cada distrito.

En todo momento en la lista sólo deben quedar a lo sumo las tres agrupaciones ganadoras para cada distrito, con nombre de agrupación (sólo los primeros 25 caracteres para el ordenamiento alfabético) y los votos obtenidos por la agrupación en qué distrito y el total de votos obtenidos en el país.

Ejemplo del archivo de texto (sus campos son de longitud variable y NO hay un carácter especial de separación de campo). El número de agrupación no tiene ninguna relación de orden ni de correlatividad. El nombre puede tener más de los 25 caracteres indicados.

```
1028Celeste y Blanca
4Verde
125Unión por Todos y Para Todos
... ... ...
```

El archivo binario responde a la siguiente información:

nagrup	número de agrupación	entero	cualquiera hasta cuatro dígitos
region	número de región	entero	de 1 a 9, no utilizado en este proceso
distri	número de distrito	entero	suponga de 1 a 20

Contemple los distintos errores que se pueden producir al leer el archivo de texto (que no exista, que esté vacío, que el número o el nombre de la agrupación no existan, que el mismo número o nombre

se repitan, etc.) e infórmelos. Suponga que hay un máximo de 25 agrupaciones, y controle no excederlas.
Contemple los distintos errores que se pueden producir al leer el archivo de votos (que no exista, que esté vacío, que el número de agrupación no coincida con alguna de las agrupaciones leídas del archivo de texto, que el número de distrito no esté en el rango posible, es decir la cantidad de columnas con que diseñó el array de votos, que la región no esté en el rango fijado, etc.), e infórmelos.
Tenga en cuenta que para un distrito puede haber los siguientes resultados, con $v_p > v_s > v_t > ...$ (restante cantidad de votos), salvo que hubiera sólo una o dos agrupaciones para ese distrito.

v_p	una	una	una	dos	tres o más
v_s	una	una	dos o más	una o más	
v_t	una	una o más			

Ejercicio 4.7
Se ha hecho una encuesta de aceptación del público de una serie de productos y se desea generar un informe en que se muestren los tres productos con más aceptación por cada una de las zonas. Al efecto se dispone de un archivo de texto en el que se ha almacenado un código de identificación del producto y la denominación del mismo.
Con los resultados de la encuesta, por cada producto elegido por cada encuestado se ha generado un archivo binario en que se almacenó el código de identificación del producto, el código del encuestador y la zona donde habita el encuestado.
Ejemplo del archivo de texto (sus campos son de longitud variable y el carácter de separación de campo es ~). El primer campo es el código de identificación del producto (alfanumérico de hasta 8 caracteres) y el segundo es la denominación de los productos encuestados.

```
K1028~Balizas Verde Agua
A5~Cereales Rellenos de Dulce con Chispas de Chocolate
F125~Cable de Yeso
... ... ...
```

El archivo binario responde a la siguiente información:

prod	código de identificación del producto	alfanumérico cualquiera, de hasta 8 caracteres
encu	código del encuestador	entero de 1 a 197
zona	zona	entero suponga de 1 a 20

El jefe de programadores ha decidido que se deben resolver las siguientes funciones:

- Una función que lea el archivo de texto y almacene en arrays bidimensionales los códigos de identificación de los productos y sus denominaciones (prod y deno). Haga uso de una función que se encargue de separar y validar estos campos (que no estén vacíos, que el prod no exceda los 8 caracteres, etc.).
- Una función que se encargue de leer (del archivo binario) y acumular las preferencias del público.
- Una función que a partir de los arrays invoque a una función que se encargue de insertar en orden en una lista de modo de poder hacer un informe con los cinco productos que cuentan con mayor aceptación en cada zona, de modo que quede ordenado por zona, cantidad de encuestados que lo eligen, y denominación del producto. Haga las consideraciones correspondientes al ejercicio anterior para determinar los cinco (o más) productos preferidos.

Ejercicio 4.8
Resuelva las siete primitivas de Pila y Cola implementadas en lista circular con asignación dinámica de memoria.

Ejercicio 4.8
Resuelva las primitivas de lista doblemente enlazada, teniendo en cuenta que para la inserción en la lista doblemente enlazada hay tres variantes: tener la dirección del primer nodo de la lista, del último de la lista, y del último insertado (cuando se hace una carga ordenada).

- Resuelva la inserción al comienzo, al final y en orden por una clave.
- Resuelva el ordenamiento de la lista.
- Resuelva la búsqueda por la clave, con eliminación o no del nodo, recuperando la información en caso de encontrarse la clave y teniendo en cuenta que hay tres variantes (se tiene la dirección del primer nodo de la lista, del último, o del último tratado).

Escriba un programa que le permita comprobar el correcto funcionamiento de sus funciones.

Ejercicio 4.9
La función map tiene como objetivo transformar una lista. Es decir aplicará una función a cada uno de sus elementos.
Se le solicita que diseñe y desarrolle una función denominada Map, la cual recibirá como argumento un puntero a una función y una lista. Map deberá aplicar la función recibida como puntero a cada uno de los elementos de la lista y retornar una nueva lista con los resultados de la transformación realizada.

Ejercicio 4.10
La función filter tiene como objetivo filtrar elementos de una lista que cumplan con un determinado criterio y retornar una nueva lista con los elementos que cumplan dicho criterio.
Se le solicita que diseñe y desarrolle una función denominada filter, la cual recibirá como argumento un puntero a una función y una lista. La función pasada como argumento será la encargada de producir el filtrado. Filter deberá retornar una nueva lista con los elementos que cumplieron el criterio de filtrado.

Ejercicio 4.9

La función reduce recibe como argumento un puntero a función y una lista. Su objetivo es “reducir” la lista a un único valor. La forma en que es reducida dependerá de la función que se pasa como argumento y tendrá dos parámetros. Es decir aplicará la función a los elementos de de a pares de izquierda a derecha, y ese resultado se aplicará al siguiente valor de la lista.

Por ejemplo si se tuviera una lista con los valores [1, 2, 3, 4] la función debería retornar 10, sin embargo el cálculo se realizaría de la siguiente manera: ((1 + 2) + 3) + 4 cuyo resultado es 10.

Se le solicita que diseñe y desarrolle una función denominada reduce, la cual recibirá como argumento un puntero a una función y una lista. Reduce deberá aplicar la función recibida como puntero y deberá retornar una nueva lista con el resultado.

Tema: Concepto de recursión. Estructura de funciones recursivas. Ejecución de funciones recursivas. Envoltura para funciones recursivas. Tipos de recursividad. Eficiencia de la recursividad. Recursión versus Iteración.

Unidad 5: Recursividad

Objetivo: Comprender el concepto de recursión. Conocer y comprender la conveniencia de su uso en función del tipo de requerimiento a resolver.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 7 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

NOTA: Para los ejercicios de esta sección se le solicita:

- Determine la condición de corte.
- Contemple las condiciones de borde.

Ejercicio 5.1
Escriba una función que calcule la potencia de un número de manera recursiva. La función debe recibir como argumentos la base y el exponente.

Ejercicio 5.2
Implemente una función recursiva que calcule el producto de dos números enteros.

Ejercicio 5.3
Escriba una función recursiva que permita calcular la el término i de la serie de Fibonacci. La serie se construye:

$$\text{fibo}(i) = \begin{cases} 1 & i = 1 \\ 1 & i = 2 \\ \text{fibo}(i-1) + \text{fibo}(i-2) & i > 2 \end{cases}$$

Ejercicio 5.4
Implementa una función recursiva que pase un número en base 10(decimal) a base 2 (binario).

Ejercicio 5.5
Diseña un algoritmo recursivo que permita calcular la función de Ackermann de dos números enteros la cual se define:

$$A(m,n) = \begin{cases} n+1, & \text{si } m = 0; \\ A(m-1,1), & \text{si } m > 0 \text{ y } n = 0; \\ A(m-1, A(m,n-1)), & \text{si } m > 0 \text{ y } n > 0 \end{cases}$$

Ejercicio 5.6
Escriba una función recursiva para determinar si una cadena es un palíndromo, o sea se igual al derecho y al revés. Ej.: “Anita, lava la tina”, “Arriba la BIRRA!!!”

Ejercicio 5.7
Escriba una función recursiva para simular bsearc:

```
void *bsearch(const void *key, const void *base, size_t nitems, size_t size, int (*compar)(const void *, const void *))
```

Ejercicio 5.8
Escriba una función recursiva para mostrar una lista simplemente enlazada en orden inverso.

Ejercicio 5.9
Escriba una función recursiva para insertar un elemento en una lista simplemente enlazada.

Tema: Recursividad. Árbol binario, creación, recorridas EnOrden, PreOrden y PosOrden. Árbol binario de búsqueda, su relación con la búsqueda binaria en arrays. Determinación de altura, y otras funciones. Árboles AVL y balanceados, determinación. Diferentes tipos de implementación.
Concepto de archivos de índice (.idx) Implementación de índices sobre un árbol binario de búsqueda. Ventajas de implementar índices sobre árboles binarios de búsqueda.

Unidad 6: Estructura de datos Árbol.

Objetivo: Comprender el concepto, funcionamiento y usos de un TDA Árbol.

Competencia/s a desarrollar:

- **Competencias tecnológicas**
  - Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
  - Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
  - Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
  - Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**
  - Desempeño en equipos de trabajo.
  - Comunicación efectiva.
  - Actuación profesional ética y responsable.
  - Aprendizaje continuo.
  - Desarrollo de una actitud profesional emprendedora.

Se espera que el estudiante logre:
Comprensión del requerimiento. Diseñar el algoritmo que resuelva el requerimiento. Implementar el algoritmo. Establecer las condiciones de borde para su funcionamiento. Establecer los lotes de pruebas y sus correspondientes salidas esperadas.

Descripción de la Actividad:

1- Tiempo estimado de resolución: 14 días.
2- Metodología: Planteo de requerimiento. Comprensión de requisitos y diseño de solución. Diseño de casos de prueba. Determinación de la salida esperada para cada caso de prueba. Codificación de la solución en lenguaje C. Prueba y validación de resultados.
3- Forma de entrega: se indicará qué ejercicios deberán ser resueltos y entregados sobre la plataforma MIeL.
4- Metodología de corrección y feedback al alumno: serán planteadas posibles estrategias de resolución. Serán evacuadas dudas puntuales respecto a la interpretación del ejercicio a resolver y su resolución.

Bibliografía: obligatoria sugerida por la cátedra (ver al inicio de este documento).

Ejercicio 6.1
Valiéndose de las primitivas de Árbol vistas en clases, escriba un programa que permita cargar información en un árbol binario de búsqueda. Esta información estará compuesta de: legajo, apellido-y-nombre y cargo (alfanuméricos de 10, 35 y 15 caracteres respectivamente), fecha de alta y fecha de baja (la fecha de alta no puede faltar a diferencia de la de baja).
Valiéndose de un menú:

- Pruebe las primitivas de cargarArbol en ambas versiones (recursiva e iterativa).
- Implemente las funciones que recorren el árbol, mostrando la información de sus nodos en las tres formas de recorrido (inOrden, preOrden y posOrden).
- Implemente la función que determina la altura del árbol.
- Implemente las funciones que:
    - Muestre los nodos hoja.
    - Muestre los nodos no-hoja.
    - Muestre los nodos que sólo tienen hijo por izquierda.
    - Muestre los nodos que tienen hijo por izquierda.
    - Elimine todos los nodos de un árbol.
    - 'pode' las ramas de un árbol de modo que no supere una altura determinada.
    - 'pode' las ramas de un árbol de una altura determinada o inferior.
    -

Al terminar el programa, genere un archivo con la información del árbol, haciendo uso de una función que lo recorre en pre orden.

Ejercicio 6.2
Valiéndose de un árbol en el que sólo almacena la clave y el número de registro que le corresponde en el archivo, escriba un programa que intente abrir un archivo, y si lo puede abrir, haga la carga del árbol. La información a tratar es la del ejercicio anterior. A continuación, haciendo uso de un menú, implemente las funciones necesarias que permitan:

- Agregar nuevos registros de información (siempre que la clave no exista en el árbol), agregando el registro al final del archivo y su clave y número de registro en el árbol.
- Ingresar la clave, para buscarla en el árbol y con el número de registro muestre la información del archivo.
- Recorriendo el árbol, muestre la información de los registros del archivo en orden.
- Ídem en preorden.
- Ídem en posorden.
- Ingresar la clave, para buscarla en el árbol y con el número de registro asignar la fecha de baja.

Al terminar el programa, debe generar un nuevo archivo ordenado por la clave.

Ejercicio 6.3
Escriba un programa que le permita verificar que el archivo ha quedado ordenado.

Ejercicio 6.4
A partir de un árbol binario de búsqueda cargado en memoria implemente las siguientes funciones que determinen si el árbol es:

- Completo
- Balanceado
- AVL.

Las funciones reciben el árbol y retornan 1 o 0 según se cumpla o no la condición.

Ejercicio 6.5
Implemente una función denominada *determinarTipoDeArbol* que reciba como argumento un árbol binario de búsqueda y retorne el tipo de árbol (Completo, Balanceado, AVL, Otro) que se le ha pasado como argumento.
Defina y utilice macro reemplazos para retornar el tipo de árbol detectado.
Genere el entorno para realizar la prueba de la función teniendo en cuenta las variantes.

Ejercicio 6.6
Dispone de un archivo binario con registros del tipo tPersona, cuya clave es el DNI de la persona. El archivo se encuentra ordenado por DNI y no posee registros con DNI duplicado.
Diseñe e implemente una función que reciba como parámetros un puntero del tipo *FILE y un puntero a un tipo Árbol Binario de Búsqueda. La función debe crear un árbol lo más balanceado posible a partir del archivo ordenado.
Contemple las condiciones de borde.
Arme un proyecto en el cual divida el código en en diferentes archivos fuente según la funcionalidad que aporten.
Debe producir una aplicación completamente funcional.

Ejercicio 6.7
Dispone de un archivo binario con registros del tipo tPersona, cuya clave es el DNI de la persona. El archivo no posee registros con DNI duplicados y se encuentra desordenado.
Diseñe e implemente una aplicación que permita acceder al archivo mediante un **índice**. El tipo índice contendrá el DNI de la persona y el número de registro.
Contemple las condiciones de borde.
Arme un proyecto en el cual divida el código en en diferentes archivos fuente según la funcionalidad que aporten.
Debe producir una aplicación completamente funcional.