UNIVERSIDAD DE GUADALAJARA

1. DATOS GENERALES DE LA UNIDAD DE APRENDIZAJE (UA) O ASIGNATURA			
Nombre de la Unidad de Aprendizaje (UA) o Asignatura			**Clave de la UA**
Seminario de solución de problemas de estructuras de datos II			IB131
Modalidad de la UA	**Tipo de UA**	**Área de formación**	**Valor en créditos**
Escolarizada	Seminario	Básica común	5

UA de pre-requisito	UA simultaneo	UA posteriores
Estructuras de datos I	Estructuras de datos II	
Horas totales de teoría	**Horas totales de práctica**	**Horas totales del curso**
0	80	80

Licenciatura(s) en que se imparte	Módulo al que pertenece
Ingeniería Fotónica	Sistemas Electro-ópticos y de Comunicaciones
Departamento	**Academia a la que pertenece**
Ciencias Computacionales	Estructuras de datos
Elaboró	**Fecha de elaboración o revisión**
M en C. Luis Alberto Muñoz Gómez	20/07/2018

2. DESCRIPCIÓN DE LA UA O ASIGNATURA

Presentación

En la presente unidad de aprendizaje, el estudiante elaborará soluciones de software, las cuales demuestren sus conocimientos acerca de la gestión de archivos (organizar y clasificar la información durante la escritura para la posterior recuperación de información) con el máximo de eficiencia, aplicado tanto en archivos de texto como en archivos binarios, conociendo los por menores acerca de la fragmentación interna y externa de archivos, y además, el modelado de información en memoria principal en estructuras de datos avanzadas, para modelar la realidad en la computadora de forma óptima y de rápido acceso. Para dicha gestión de archivos, el estudiante implementará el traslado de datos ubicados en memoria principal (el cual ya debe estar organizado en estructuras de datos como listas, pilas, colas, etc.), llevando los datos hacia la memoria secundaria (escribiendo en uno o más archivos), para luego, viceversa, cargar datos pertinentes desde un archivo hacia la memoria principal. En algunas soluciones se accede al archivo en modo secuencial (mediante las técnicas de delimitadores con archivos de texto, o bien, con campos de dimensión con archivos binarios y, finalmente, serialización de objetos) y en otras en modo aleatorio (con acceso directo en archivos binarios). Además, el estudiante aplica soluciones de grafos para modelar datos de la realidad y sus diversas interconexiones, se organiza la búsqueda de información en disco mediante la implementación de índices y se acelera el acceso a la información en archivos mediante tablas de dispersión y árboles binarios paginados. El estudiante implementará dichas soluciones de software, algunas en el paradigma estructurado/modular usando el lenguaje de programación C, y otras. aplicando el paradigma orientado a objetos en C++. Algunas soluciones son desarrolladas de forma aislada y otras como un proyecto evolutivo entregado por etapas. Para lograr lo anteriormente expuesto, el estudiante aplicará conocimientos previamente adquiridos ya sea, en la unidad de aprendizaje denominada “Estructuras de datos II” (que ya hubiese sido acreditada en un semestre anterior), o bien, aquellos conocimientos obtenidos de forma autónoma, acordes a la bibliografía citada en la sección 6 de este documento.

Relación con el perfil

Modular	De egreso
En el módulo de “Arquitectura y programación de sistemas” se conocen los recursos que ofrece la computadora al programador para que finalmente el programador aproveche sus recursos de manera óptima mediante un programa de cómputo. En la presente unidad de aprendizaje se abona a la arquitectura, porque el estudiante conoce la importancia de cuando y cómo organizar y clasificar la información considerando cómo trabajan en general, tanto la memoria principal como el medio de almacenamiento secundario (disco duro, memoria USB, disco óptico, cinta) para el almacenamiento persistente, la diferencia en velocidad de la memoria principal contra la memoria secundaria, y finalmente los por menores respecto a la fragmentación de archivos; en cuanto a la programación de sistemas se abona en que el estudiante desarrolla soluciones de software para la implementación de sistemas eficientes tanto para el uso óptimo de la memoria principal, que requieren interactuar activamente con la memoria secundaria.	Se forma al estudiante conociendo el hardware de la computadora, específicamente el cómo trabajan la memoria RAM y el almacenamiento secundario (disco duro, memoria USB, disco óptico, cinta), para lograr el almacenamiento persistente mediante técnicas adecuadas, implementadas en software.

Competencias a desarrollar en la UA o Asignatura

Transversales	Genéricas	Profesionales
1. Abstrae, analiza y sintetiza problemas de la realidad; 2. Identifica y resuelve problemas que sean representables en la computadora; 3. Crítica las soluciones de sus pares y las compara con las propias; 4. Trabaja en equipo en la resolución de situaciones triviales; 5. Aplica en la práctica los conocimientos adquiridos; 6. Organiza su tiempo para entregar sus trabajos en tiempo y forma;	1. Abstrae, analiza y sintetiza los componentes de un problema de la realidad, para representarlos en la computadora mediante estructuras de datos modeladas en memoria principal y almacenables en archivos; 2. Identifica y resuelve problemas resolubles mediante la computadora (computables), eligiendo la estructura de datos adecuada para dicho problema; 3. Crítica las soluciones software de sus pares y las compara con las propias tanto en cuanto a utilización de recursos como en el desempeño de la solución software;	1. Propone y desarrolla soluciones a problemas de la realidad, que sean representables mediante la computadora; 2. Identifica y resuelve problemas computables; 3. Evalúa soluciones software en cuanto a recursos utilizados y desempeño; 4. Trabaja en equipo en el diseño de un proyecto; 5. Organiza y planifica sus entregables de un proyecto entregable por etapas; 6. Aplica su capacidad creativa en la solución de problemas.

7. Efectúa un trabajo autónomo sobre los mismos problemas que sus compañeros; 8. Desarrolla su capacidad creativa en la solución de problemas.	4. Trabaja en equipo en la resolución de situaciones triviales en cuanto a la escritura de programas de computadora; 5. Aplica en la práctica los conocimientos previamente adquiridos de cursos anteriores de Estructuras de Datos, o bien conocimientos producto de su aprendizaje autogestivo; 6. Organiza su tiempo para entregar sus trabajos; 7. Efectúa un trabajo autónomo sobre los mismos problemas que resuelven sus compañeros; 8. Desarrolla su capacidad creativa en la solución de problemas, para elegir la estructura de datos adecuada al problema en cuestión.	
Saberes involucrados en la UA o Asignatura		
Saber (conocimientos)	**Saber hacer (habilidades)**	**Saber ser (actitudes y valores)**
1. Registros de Longitud Variable y Registros de Longitud Fija 2. Serialización 3. Índices 4. Tabla de Dispersión 5. Árbol Binario Paginado 6. Grafos	1. Abstrae un problema de la realidad y lo modela gráficamente 2. Analiza los componentes del problema a resolver para determinar la estructura de datos adecuada y modelar gráficamente la solución mediante lenguaje unificado de modelado (UML); 3. Diseña el cómo organizar los datos almacenables en archivos; 4. Acuerda con su equipo el diseño preliminar del proyecto del curso; 5. Organiza su tiempo para entregar tanto programas aislados como sus avances de proyecto en tiempo y forma; 6. Desarrolla su capacidad creativa en la solución de problemas, para determinar el diseño o rediseño de un sistema y elegir las estructuras de datos adecuados al problema en cuestión; 7. Implementa soluciones tanto en el paradigma estructurado modular, como en el paradigma orientado a objetos.	1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Liderazgo para la toma de decisiones en el diseño de una solución; 3. Negociación en cuanto a las ventajas y desventajas de las propuestas de los compañeros de su equipo; 4. Manejo de tiempo respecto a la planificación de entrega de avances de proyecto; 5. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 6.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 7. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 8. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 9. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.
Producto Integrador Final de la UA o Asignatura		

Título del producto #1: Aplicación software para gestión de información en disco.

Objetivo: Diseñar e implementar una aplicación software que resuelva un problema de la realidad, analizar la serie de requisitos a cumplir acorde al levantamiento de requerimientos, aplicar su capacidad creativa en modelar gráficamente la solución, implementar el producto software utilizando las técnicas adecuadas acordes a la naturaleza de los datos a almacenar, con el fin de evaluar la solución propuesta, en contraste con las soluciones de otros compañeros.

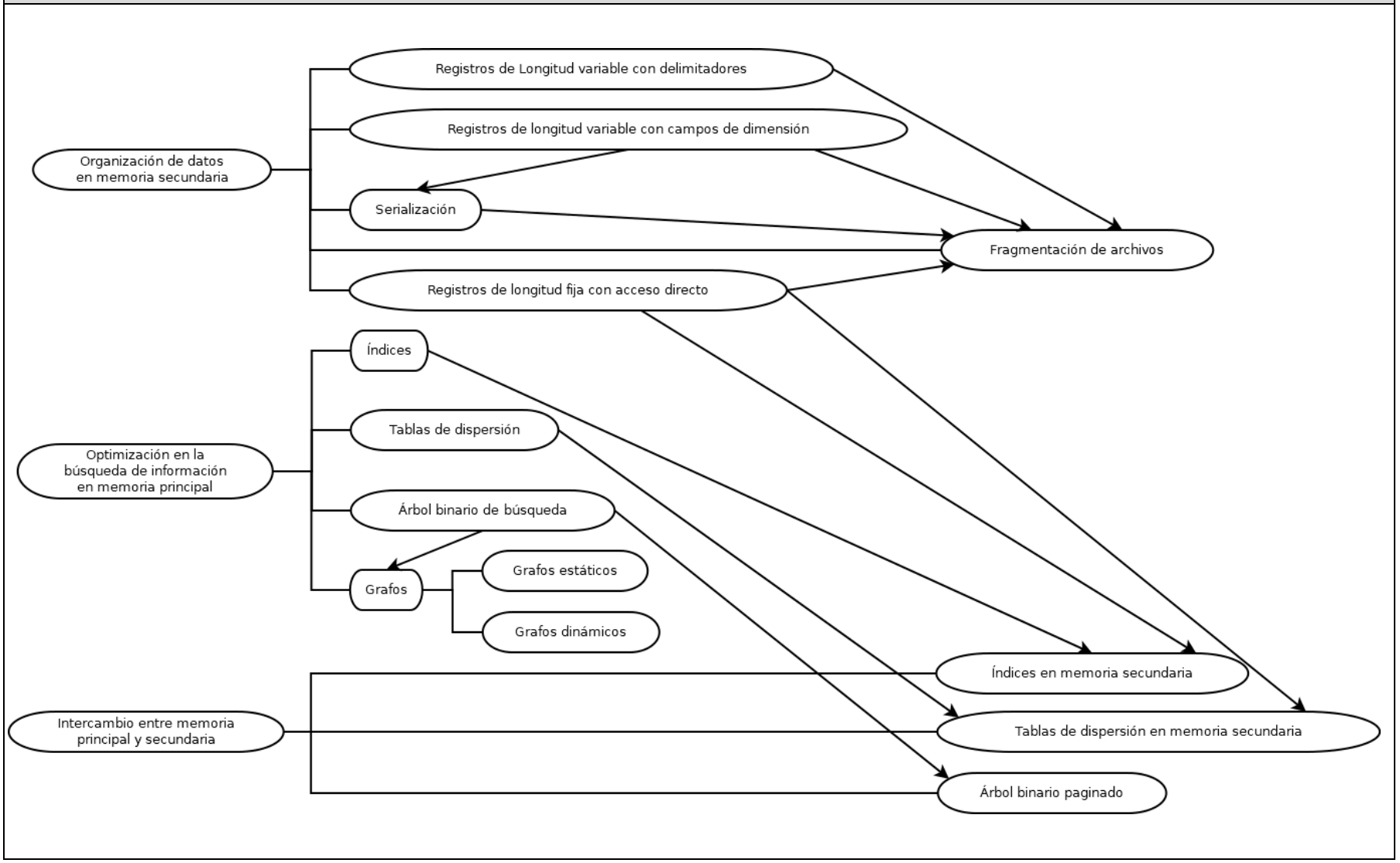
Descripción: La aplicación de software consiste en una solución, diseñada (mediante lenguaje unificado de modelado, UML solo para el diagrama de clases y sus relaciones) e implementada aplicando el paradigma orientado a objetos (con el lenguaje de programación C++), la cual represente al menos cinco tipos de entidades (clases) de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que permita operaciones ABC (al menos incluyendo altas, bajas, y consultas de información para listados de registros/objetos); que mantenga su información en memoria principal usando estructuras de datos (por ejemplo y sin limitar: arreglos, listas, pilas, colas, índices, tablas de dispersión, grafos), ya sea estáticas y/o dinámicas; que almacene los datos de forma persistente en archivos (mediante las técnicas de: registros de longitud variable con delimitadores, registros de longitud variable con campo de dimensión, registros de longitud fija con acceso directo, serialización); y que finalmente, incluya estructuras de datos para búsqueda de información (por ejemplo: índices y/o tablas de dispersión, gestionados en memoria principal y en disco duro, y/o árboles binarios paginados). Dicha aplicación será propuesta y especificada a detalle por el profesor del curso, tanto para los requerimientos a cumplir como en el alcance, con la finalidad de que los estudiantes contrasten la solución propia con la desarrollada por otros estudiantes de su mismo grupo.

Título del producto #2: Portafolio de soluciones independientes.

Objetivo: Diseñar e implementar soluciones de software que resuelvan problemas pequeños y aislados, demostrar los conocimientos adquiridos para aquellos saberes (conocimientos) que no hayan sido integrados en el Producto #1 (debido solo al diseño y alcances de dicho producto)con el fin de evaluar las soluciones propuestas, en contraste con las soluciones de otros compañeros.

Descripción: Elaborar pequeños productos de software, diseñados e implementados aplicando el paradigma orientado a objetos (en el lenguaje de programación C++), los cuales modelen cada uno al menos un tipo de entidad de la realidad, que mantengan su información ya sea solo en estructuras de datos en memoria principal y/o sean almacenados los datos de forma persistente en archivos. Dichos productos serán propuestos y especificados a detalle por el profesor del curso, tanto para los requerimientos a cumplir como en el alcance de cada uno, con la finalidad de que los estudiantes contrasten la solución propia con la desarrollada por otros estudiantes de su mismo grupo.

3. ORGANIZADOR GRÁFICO DE LOS CONTENIDOS DE LA UA O ASIGNATURA



4. SECUENCIA DEL CURSO POR UNIDADES TEMÁTICAS

Unidad temática 1: Organización de datos en Memoria Secundaria

Objetivo de la unidad temática: Reconocer y practicar las diferentes técnicas para leer y escribir archivos en la memoria secundaria en base al algoritmo para escritura en disco, con la finalidad de contrastar las ventajas y desventajas de organizar los datos mediante cada una de las diferentes técnicas.

Introducción: En el caso de un programa en ejecución que requiera del almacenamiento persistente de datos, la organizados de dichos datos en memoria secundaria determina la eficiencia del programa, así como los requerimientos de espacio en memoria secundaria a largo plazo. La técnica de organización elegida para un archivo, determina la posibilidad de optimización en la búsqueda de información almacenada.

Contenido temático	Saberes involucrados	Producto de la unidad temática
1.1. Clases administradoras de objetos **1.2 Registros de longitud variable** **1.2.1 Técnica con delimitadores** **1.2.2 Técnica con campos de dimensión** **1.3. Registros de longitud fija con acceso directo** **1.4. Serialización** **1.5. Fragmentación de archivos** **1.5.1 Fragmentación interna** **1.5.2 Fragmentación externa** **1.6. Objetos de acceso a datos (DAOs)**	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos de texto; 7. Registros de longitud variable con delimitadores; 8. Escritura y lectura de archivos binarios; 9. Registros de longitud variable con campos de dimensión; 10. Registros de longitud fija con acceso directo; 11. Serialización; 12. Fragmentación interna; 13. Fragmentación externa; 14. Objetos de acceso a datos. Saber hacer: 1. Abstrae un problema de la realidad y lo modela gráficamente; 2. Analiza los componentes del problema a resolver para determinar la estructura de datos adecuada y modelar gráficamente la solución mediante lenguaje unificado de modelado (UML); 3. Diseña el cómo organizar los datos almacenables en archivos; 4. Acuerda con su equipo el diseño preliminar del proyecto del curso; 5. Organiza su tiempo para entregar tanto programas aislados como sus avances de proyecto en tiempo y forma; 6. Desarrolla su capacidad creativa en la solución de problemas, para determinar el diseño o rediseño de un sistema y elegir las estructuras de datos adecuados al problema en cuestión; 7. Implementa soluciones tanto en el paradigma estructurado modular, como en el paradigma orientado a objetos.	Actividades de aprendizaje (entregables) del proyecto integrador (Producto integrador #1 de la UA): 1.1. Modelado de una aplicación software (implementable en un lenguaje de programación orientado a objetos) mediante lenguaje UML (solo para el diagrama de clases, con atributos, métodos y relaciones), la cual represente al menos cinco tipos de entidades (clases) de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que provea operaciones ABC (al menos altas, bajas y consultas de información para todos los tipos de entidad); y que contemple tipos de datos primitivos tanto enteros como reales y caracteres, y al menos un arreglo de enteros o reales. 1.2. Implementación de una aplicación software ABC, que modele tipos de datos primitivos (tanto enteros como reales y caracteres), uno o más arreglos de primitivos y un arreglo de registros; que toda variable de estado del programa sea almacenada dentro de un objeto; que mantenga su información en memoria principal usando una o más estructuras de datos estáticas; que almacene los datos de forma persistente en archivo de texto (cuyo formato elija el estudiante); que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo; y que además se incluya el diseño UML actualizado (con todas las correcciones

	Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Liderazgo para la toma de decisiones en el diseño de una solución; 3. Negociación en cuanto a las ventajas y desventajas de las propuestas de los compañeros de su equipo; 4. Manejo de tiempo respecto a la planificación de entrega de avances de proyecto; 5. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 6.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 7. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 8. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 9. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.	detectadas durante la retroalimentación de la actividad anterior). 1.3. Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad), donde esta última entidad sea acorde al diseño UML actualizado; que mantenga su información en memoria principal usando una estructura de datos estática, la cual guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo mediante la técnica de registros de longitud variable con delimitadores; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo. 1.4. Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad diferente a la de la actividad anterior, donde esta última entidad sea acorde al diseño UML actualizado; que mantenga su información en memoria principal usando una estructura de datos dinámica, la cual guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo mediante la técnica de registros de longitud variable con campos de dimensión; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo. 1.5. Implementación de una aplicación software ABC orientada a objetos, cuyo

		código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad diferente a la de actividades anteriores, donde esta última entidad sea acorde al actual diseño UML; que mantenga su información en memoria principal usando una estructura de datos dinámica (diferente a la estructura de datos usada en la actividad anterior), que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo mediante la técnica de registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo. 1.6 Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, reorganice todos los módulos del programa encargados de escribir y leer archivos a su forma orientada a objetos en la forma de objetos de acceso a datos (DAOs), aplicando la actualización correspondiente al diseño UML. Actividades de aprendizaje (entregables) del proyecto integrador (Producto integrador #2 de la UA): 1.7. Implementación de una aplicación software que modele 3 entidades relacionadas, sin relaciones recursivas entre una y la otra, que almacenen cada una diversos atributos primitivos y también, un apuntador hacia otro objeto; se lleve a cabo la serialización del objeto principal del modelo y posteriormente la deserialización de dicho objeto.

		1.8. Examen de programación, consistente en completar la implementación de una aplicación software ABC orientada a objetos (diseñada e implementada parcialmente por el profesor, previo al examen); que incluya la implementación de un tipo de entidad de la realidad; que mantenga su información en memoria principal usando diversas estructuras de datos (dinámicas y/o estáticas), que guarden apuntadores a objetos; que almacene los datos de forma persistente en archivos mediante todas y cada una de las técnicas de: registros de longitud variable con delimitadores, registros de longitud variable con campos de dimensión y registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.

Actividades del docente	Actividades del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
1. Organizar a los estudiantes en grupos de 3 a 5 integrantes; 2. Planteamiento del problema a resolver presentando la descripción del proyecto ante el grupo; 3. Aclarar dudas sobre el proyecto; 4. Establecer fechas para cumplir con cada etapa del proyecto; 5. Proveer asesoría durante el diseño de la aplicación software mediante UML y sus eventuales correcciones; 6. Proveer los medios para resolver dudas en cuanto a codificación en C++, para las diferentes etapas del proyecto, apoyándose también de otros estudiantes del grupo; 7. Retroalimentar el diseño UML propuesto por el grupo; 8. Retroalimentar el código fuente de cada actividad de aprendizaje; 9. Solicitar correcciones al diseño del proyecto y/o al código fuente entregado; 10. Revisar las correcciones solicitadas en el diseño y/o código fuente.	1. Agruparse con compañeros del grupo, preferentemente conocidos o de la misma carera; 2. Lectura de bibliografía previo a cada sesión de clase futura; 3. Solicitar aclaraciones iniciales sobre el proyecto planteado por el profesor; 4. Diseñar la aplicación mediante UML; 5. Contrastar su diseño UML con sus compañeros de equipo y hacer consenso de un diseño por parte de su equipo; 6. Contrastar el diseño UML de su equipo, con los de otros equipos y hacer consenso de un diseño por parte del grupo; 7. Corregir el diseño del proyecto, a nivel individual, luego el producido por el equipo, y finalmente el generado por el grupo; 8. Contrastar el código fuente producido para cada actividad de aprendizaje, con los producidos por otros compañeros de su equipo o del grupo; 9. Corregir el código fuente producido para cada actividad de aprendizaje, acorde a las observaciones detectadas en su trabajo, previo a la entrega de la siguiente actividad de aprendizaje.	Para cada una de las actividades de aprendizaje (producto de la unidad temática), entregar un video (y no varios videos), en formato MP4 (cuyo nombre de archivo sea el especificado por el profesor), el cual muestre toda la pantalla de la computadora, e incluya una explicación oral por parte del alumno durante todo el video, ilustrando: 1. El diseño UML actual del proyecto; 2. El código fuente, mostrando de este exclusivamente los nombres de clases, sus atributos y los prototipos de métodos (sin mostrar	1. Herramienta CASE (Computer Assisted Software Engineering) Dia (u otra propuesta por el profesor); 2. IDE Code::Blocks con Mingw (u otro propuesto por el profesor, pero que continúe dicho IDE siendo actualizado); 3. Editor de documentos LibreOffice Writer (u otro propuesto por el profesor); 4. Herramienta de captura de video aTube Catcher (u otra propuesta por el profesor), que genere videos en formato MP4; 5. Bloc de notas básico del sistema operativo;	28

		la implementación de las subrutinas); 3. El programa en ejecución, mostrando todas las características del programa visibles al usuario de la aplicación; 4. El contenido de los archivos generados por la aplicación, resultado de aplicar la técnica de escritura, propia de cada actividad de aprendizaje, haciendo énfasis en mostrar la ubicación de los datos de tipo cadena y de tipo entero, usando para lo anterior un bloc de notas para abrir el archivo.	6. Aplicación lectora de documentos PDF (como Acrobat Reader); 7. Aplicación para imprimir documentos hacia el formato PDF (como el Microsoft Print to PDF de Windows).	

Unidad temática 2: Eficiencia en la Búsqueda de Información

Objetivo de la unidad temática: Seleccionar y emplear estructuras de datos para clasificar la información tanto en memoria principal como en memoria secundaria, con la finalidad de proponer la estructura adecuada acorde a la naturaleza de la información a buscar por parte del usuario a través de su aplicación software.

Introducción: La técnica de diseño de algoritmos “divide y vencerás” se aplica en gran variedad de problemas donde el tamaño del conjunto de datos a explorar es importante. La eficiencia de las estructuras de datos utilizadas para un problema dado contrasta una búsqueda secuencial de orden lineal “O(n)”, contra una búsqueda más rápida, de orden “$O(\log_2(n))$”; así mismo, existen situaciones en las que es deseable que la eficiencia sea de orden constante “O(1)”, o cercano a ello. Hacer más eficiente la búsqueda de información es especialmente necesaria cuando los datos se encuentran en memoria secundaria y no es posible, por su volumen, almacenarlos en memoria principal.

Contenido temático	Saberes involucrados	Producto de la unidad temática
2.1. Indización **2.1.1 Índices primarios** **2.1.2 Índices secundarios** **2.1.3 TDA Lista basada en cursores** **2.1.4 Listas invertidas** **2.1.5 Intercambio entre memoria principal y secundaria** **2.2. Tablas de dispersión** **2.3.1 Manejo de colisiones** **2.3.2 Saturación progresiva** **2.3.3 Compartimentos** **2.3.4 Intercambio entre memoria principal y secundaria** **2.3. Árbol binario paginado** **2.3.1 Árbol binario de búsqueda** **2.3.2 Paginación en disco duro**	Saber: 1. Búsqueda lineal; 2. Búsqueda binaria; 3. Índices primarios en memoria principal; 4. Índices secundarios en memoria principal; 5. Índices en memoria secundaria; 6. TDA Lista basada en cursores; 7. Listas invertidas; 8. Tabla de dispersión en memoria principal; 9. Manejo de colisiones en tabla de dispersión; 10. Saturación progresiva en tabla de dispersión; 11. Compartimientos en tabla de dispersión; 12. Árbol binario de búsqueda; 13. Árbol binario paginado.	Actividades de aprendizaje (entregables) del proyecto integrador (Producto integrador #2 de la UA): 2.1. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando estructuras de datos dinámicas, tanto para un índice como para un listado de objetos, ambas estructuras guardando apuntadores a objetos; que almacene los

	Saber hacer: 1. . Organiza su tiempo para entregar programas en tiempo y forma; 2. Desarrolla su capacidad creativa en la solución de problemas, para elegir la estructura de datos adecuada al problema en cuestión; 3. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Manejo de tiempo respecto a la planificación de entrega de productos de software; 3. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 4.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 5. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 6. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 7. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.	datos de forma persistente en archivos, tanto el índice como el listado de objetos, lo anterior mediante la técnica de registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos. 2.2. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando una estructura de datos dinámica de tipo tabla de dispersión, que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivos, tanto la tabla de dispersión como el listado de objetos; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos. 2.3. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando una estructura de datos dinámica de tipo árbol binario de búsqueda, que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivos, tanto el árbol binario paginado como el listado de objetos; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.

Actividades del docente	**Actividades del estudiante**	**Evidencia de la actividad**	**Recursos y materiales**	**Tiempo destinado**
1. Planteamiento de la especificación de cada actividad a resolver; 2. Aclarar dudas sobre cada actividad; 3. Establecer fechas para cumplir con cada actividad; 4. Proveer los medios para resolver dudas en cuanto a codificación en C++ para cada actividad, apoyándose de también de otros estudiantes del grupo;	1. Lectura de bibliografía previo a cada sesión de clase futura; 2. Solicitar aclaraciones iniciales sobre la actividad planteado por el profesor; 3. Contrastar el código fuente producido para cada actividad de aprendizaje, con los producidos por otros compañeros de su equipo o del grupo; 4. Corregir el código fuente producido para cada actividad de aprendizaje, acorde a las observaciones	Para cada una de las actividades de aprendizaje (producto de la unidad temática), entregar un video (y no varios videos), en formato MP4 (cuyo nombre de archivo sea el especificado por el	1. IDE Code::Blocks con Mingw (u otro propuesto por el profesor, pero que continúe dicho IDE siendo actualizado); 2. Herramienta de captura de video aTube	28

5. Retroalimentar el código fuente de cada actividad de aprendizaje; 6. Solicitar correcciones al diseño de código fuente entregado; 7. Revisar las correcciones solicitadas en el código fuente.	detectadas en su trabajo, previo a la entrega de la siguiente actividad de aprendizaje.	profesor), el cual muestre toda la pantalla de la computadora, e incluya una explicación oral por parte del alumno durante todo el video, ilustrando: 1. El código fuente, mostrando de este exclusivamente los nombres de clases, sus atributos y los prototipos de métodos (sin mostrar la implementación de las subrutinas); 2. El programa en ejecución, mostrando todas las características del programa visibles al usuario de la aplicación; 3. El contenido de los archivos generados por la aplicación, resultado de aplicar la técnica de escritura, propia de cada actividad de aprendizaje, haciendo énfasis en mostrar la ubicación de los datos de tipo cadena y de tipo entero, usando para lo anterior un bloc de notas para abrir el archivo.	Catcher (u otra propuesta por el profesor), que genere videos en formato MP4; 3. Bloc de notas básico del sistema operativo.	

Unidad temática 3: Grafos

Objetivo de la unidad temática: Reconocer aquellas situaciones de la realidad que se modelan mediante la teoría matemática de grafos, conectar los elementos de la realidad a través de sus nodos y arcos, y manipular las diversas estructuras de datos conocidas hasta ahora como parte integral de una estructura de datos más general (el TDA Grafo), con la finalidad de formular representaciones de datos, flexibles y listas para un análisis matemático a llevar a cabo en la unidad de aprendizaje posterior a esta.

Introducción: Los grafos han sido objeto de estudio en las Ciencias Computacionales por su amplia y flexible capacidad de representación de datos de la realidad, tanto concreta como abstracta; cada nodo y cada arco del grafo puede representar tanto un dato simple como un objeto complejo en cuanto a sus atributos y métodos; conocer los pormenores para representar el TDA (tipo de datos abstracto) Grafo de tamaño estático o dinámico presenta ventajas y desventajas a conocer en esta unidad temática.

Contenido temático	Saberes involucrados	Producto de la unidad temática

3.1. TDA Grafo Estático **3.1.1 Grafos dirigidos / no dirigidos** **3.1.2 Grafos ponderados / no ponderados** **3.2. TDA Grafo Dinámico** **3.2.1 Grafos dirigidos / no dirigidos** **3.2.2 Grafos ponderados / no ponderados**	Saber: 1. Representación gráfica de un grafo; 2. Propiedades de los grafos; 3. Grafos no dirigidos; 4. Grafos dirigidos; 5. Grafos no ponderados; 6. Grafos ponderados; 7. TDA Grafo estático; 8. TDA Grafo dinámico. Saber hacer: 1. . Organiza su tiempo para entregar programas en tiempo y forma; 2. Desarrolla su capacidad creativa en la solución de problemas, para elegir la estructura de datos adecuada al problema en cuestión; 3. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Manejo de tiempo respecto a la planificación de entrega de productos de software; 3. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 4.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 5. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 6. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 7. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.	Actividades de aprendizaje (entregables) del proyecto integrador (Producto integrador #2 de la UA): 3.1. Implementación de una aplicación software ABC orientada a objetos, que gestione al menos un tipo de entidad de la realidad (por ejemplo: Aeropuerto, Avión); que mantenga su información en memoria principal usando una estructura de datos grafo estático. 3.2. Implementación de una aplicación software que cumpla en código fuente con todos los requisitos de la actividad anterior (con todas las correcciones en implementación, detectadas durante la retroalimentación de tal actividad); pero que mantenga su información en memoria principal usando una estructura de datos grafo dinámico, requiriendo este entregable únicamente el cambio de la sentencia “#include” que hace referencia a un TDA Grafo estático, para en su lugar incluir un TDA Grafo dinámico.

Actividades del docente	**Actividades del estudiante**	**Evidencia o de la actividad**	**Recursos y materiales**	**Tiempo destinado**
1. Planteamiento de la especificación de cada actividad a resolver; 2. Aclarar dudas sobre cada actividad; 3. Establecer fechas para cumplir con cada actividad; 4. Proveer los medios para resolver dudas en cuanto a codificación en C++ para cada actividad, apoyándose de también de otros estudiantes del grupo;	1. Lectura de bibliografía previo a cada sesión de clase futura; 2. Solicitar aclaraciones iniciales sobre la actividad planteado por el profesor; 3. Contrastar el código fuente producido para cada actividad de aprendizaje, con los producidos por otros compañeros de su equipo o del grupo; 4. Corregir el código fuente producido para cada actividad de aprendizaje, acorde a las observaciones	Para cada una de las actividades de aprendizaje (producto de la unidad temática), entregar un video (y no varios videos), en formato MP4 (cuyo nombre de archivo sea el especificado por el	1. IDE Code::Blocks con Mingw (u otro propuesto por el profesor, pero que continúe dicho IDE siendo actualizado); 2. Herramienta de captura de video aTube	12

5. Retroalimentar el código fuente de cada actividad de aprendizaje; 6. Solicitar correcciones al diseño de código fuente entregado; 7. Revisar las correcciones solicitadas en el código fuente.	detectadas en su trabajo, previo a la entrega de la siguiente actividad de aprendizaje.	profesor), el cual muestre toda la pantalla de la computadora, e incluya una explicación oral por parte del alumno durante todo el video, ilustrando: 1. El código fuente, mostrando de este exclusivamente los nombres de clases, sus atributos y los prototipos de métodos (sin mostrar la implementación de las subrutinas); 2. El programa en ejecución, mostrando todas las características del programa visibles al usuario de la aplicación.	Catcher (u otra propuesta por el profesor), que genere videos en formato MP4.	

5. EVALUACIÓN Y CALIFICACIÓN

Requerimientos de acreditación:

De acuerdo al “REGLAMENTO GENERAL DE EVALUACIÓN Y PROMOCIÓN DE ALUMNOS DE LA UNIVERSIDAD DE GUADALAJARA”:

Artículo 20. “Para que el alumno tenga derecho al registro del resultado final de la evaluación en el periodo ordinario, establecido en el calendario escolar aprobado por el H. Consejo General Universitario, se requiere:
I. Estar inscrito en el plan de estudios y curso correspondiente, y
II. Tener un mínimo de asistencia del 80% a clases y actividades registradas durante el curso.”

Artículo 27. “Para que el alumno tenga derecho al registro de la calificación en el periodo extraordinario, se requiere:
I. Estar inscrito en el plan de estudios y curso correspondiente.
II. Haber pagado el arancel y presentar el comprobante correspondiente.
III. Tener un mínimo de asistencia del 65% a clases y actividades registradas durante el curso.”

Criterios generales de evaluación:

Evaluación continua de las actividades de aprendizaje citadas en cada una de las unidades temáticas, acorde a la especificación y ponderación, que para cada una de dichas actividades establezca el profesor, por escrito mediante un documento de requerimientos; para aquellas actividades de aprendizaje que forman parte del producto integrador #1, entregar las correcciones solicitadas en dicha actividad (en la fecha de entrega de la siguiente actividad de aprendizaje), acorde a la retroalimentación recibida; cumplir con las reglas de operación y evaluación establecidas por escrito por el profesor.

Evidencias o Productos

Evidencia o producto	Competencias y saberes involucrados	Contenidos temáticos	Ponderación
1.1. Modelado de una aplicación software (implementable en un lenguaje de programación orientado a objetos) mediante lenguaje UML (solo para el diagrama de clases, con atributos, métodos y relaciones), la cual represente al menos cinco tipos de entidades (clases) de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que provea operaciones ABC (al menos altas, bajas y consultas de información para todos los tipos de entidad); y que contemple tipos de datos primitivos tanto enteros como reales y caracteres, y al menos un arreglo de enteros o reales.	Saber: 1. Diseño de software mediante diagramas de clases en UML; Saber hacer: 1. Abstrae un problema de la realidad y lo modela gráficamente; 2. Analiza los componentes del problema a resolver para determinar la estructura de datos adecuada y modelar gráficamente la solución mediante lenguaje unificado de modelado (UML); 3. Acuerda con su equipo el diseño preliminar del proyecto del curso; 4. Organiza su tiempo para entregar sus avances de proyecto en tiempo y forma; 5. Desarrolla su capacidad creativa en la solución de problemas, para determinar el diseño de un sistema adecuado al problema en cuestión; Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros;	Conocimientos sobre modelado de clases en UML, adquiridos de la unidad de aprendizaje I5886 Estructuras de datos I	**5%**

	2. Liderazgo para la toma de decisiones en el diseño de una solución; 3. Negociación en cuanto a las ventajas y desventajas de las propuestas de los compañeros de su equipo; 4. Manejo de tiempo respecto a la planificación de entrega de avances de proyecto; 5.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 6. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones.		
1.2. Implementación de una aplicación software ABC, que modele tipos de datos primitivos (tanto enteros como reales y caracteres), uno o más arreglos de primitivos y un arreglo de registros; que toda variable de estado del programa sea almacenada dentro de un objeto; que mantenga su información en memoria principal usando una o más estructuras de datos estáticas; que almacene los datos de forma persistente en archivo de texto (cuyo formato elija el estudiante); que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo; y que además se incluya el diseño UML actualizado (con todas las correcciones detectadas durante la retroalimentación de la actividad anterior).	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos de texto; Saber hacer: 1. Abstrae un problema de la realidad y lo modela gráficamente; 2. Analiza los componentes del problema a resolver para determinar la estructura de datos adecuada y modelar gráficamente la solución mediante lenguaje unificado de modelado (UML); 3. Diseña el cómo organizar los datos almacenables en archivos; 4. Organiza su tiempo para entregar sus avances de proyecto en tiempo y forma; 5. Desarrolla su capacidad creativa en la solución de problemas, para determinar el diseño o rediseño de un sistema y elegir las estructuras de datos adecuados al problema en cuestión; 6. Implementa soluciones tanto en el paradigma estructurado modular, como en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Liderazgo para la toma de decisiones en el diseño de una solución; 3. Negociación en cuanto a las ventajas y desventajas de las propuestas de los compañeros de su equipo;	Conocimientos sobre manejo de archivos de texto, adquiridos de la unidad de aprendizaje I5886 Estructuras de datos I **1.1. Clases administradoras de objetos**	**5%**

	4. Manejo de tiempo respecto a la planificación de entrega de avances de proyecto; 5. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 6.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 7. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 8. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 9. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.		
1.3. Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad), donde esta última entidad sea acorde al diseño UML actualizado; que mantenga su información en memoria principal usando una estructura de datos estática, la cual guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo mediante la técnica de registros de longitud variable con delimitadores; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo.	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos de texto; 7. Registros de longitud variable con delimitadores; Saber hacer: * los mismos que para la actividad 1.2 Saber ser: * los mismos que para la actividad 1.2	**1.2. Registros de longitud variable** **1.2.1 Técnica con delimitadores**	**10%**
1.4. Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad diferente a la de la actividad anterior, donde esta última entidad sea acorde al diseño UML actualizado; que mantenga su información en memoria principal usando una estructura de datos dinámica, la cual guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos binarios; 7. Registros de longitud variable con campos de dimensión; Saber hacer: * los mismos que para la actividad 1.2 Saber ser:	**1.2. Registros de longitud variable** **1.2.2 Técnica con campos de dimensión**	**10%**

mediante la técnica de registros de longitud variable con campos de dimensión; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo.	* los mismos que para la actividad 1.2		
1.5. Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, incluya la implementación de un tipo de entidad diferente a la de actividades anteriores, donde esta última entidad sea acorde al actual diseño UML; que mantenga su información en memoria principal usando una estructura de datos dinámica (diferente a la estructura de datos usada en la actividad anterior), que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivo mediante la técnica de registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tal archivo.	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos binarios; 7. Registros de longitud fija con acceso directo; 8. Fragmentación interna; 9. Fragmentación externa. Saber hacer: * los mismos que para la actividad 1.2 Saber ser: * los mismos que para la actividad 1.2	**1.3. Registros de longitud fija con acceso directo** **1.5. Fragmentación de archivos** **1.5.1 Fragmentación interna** **1.5.2 Fragmentación externa**	**10%**
1.6 Implementación de una aplicación software ABC orientada a objetos, cuyo código fuente incluya toda la implementación de la actividad anterior (con todas las correcciones en UML e implementación, detectadas durante la retroalimentación de tal actividad); que además, reorganice todos los módulos del programa encargados de escribir y leer archivos a su forma orientada a objetos en la forma de objetos de acceso a datos (DAOs), aplicando la actualización correspondiente al diseño UML.	Saber: 1. Diseño de software mediante diagramas de clases en UML; 2. Clases administradoras de objetos; 3. Nombres de archivos y extensiones; 4. Organización de archivos en clústeres de disco; 5. Modos de acceso de archivos; 6. Escritura y lectura de archivos de texto; 7. Registros de longitud variable con delimitadores; 8. Escritura y lectura de archivos binarios; 9. Registros de longitud variable con campos de dimensión; 10. Registros de longitud fija con acceso directo; 11. Serialización; 12. Fragmentación interna; 13. Fragmentación externa; 14. Objetos de acceso a datos. Saber hacer: * los mismos que para la actividad 1.2 Saber ser: * los mismos que para la actividad 1.2		**10%**

1.7. Implementación de una aplicación software que modele 3 entidades relacionadas, sin relaciones recursivas entre una y la otra, que almacenen cada una diversos atributos primitivos y también, un apuntador hacia otro objeto; se lleve a cabo la serialización del objeto principal del modelo y posteriormente la deserialización de dicho objeto.	Saber: 1. Nombres de archivos y extensiones; 2. Organización de archivos en clústeres de disco; 3. Modos de acceso de archivos; 4. Registros de longitud variable con campos de dimensión; 5. Serialización; 6. Fragmentación interna; 7. Fragmentación externa. Saber hacer: 1. Diseña el cómo organizar los datos almacenables en archivos; 2. Organiza su tiempo para entregar el producto en tiempo y forma; 3. Desarrolla su capacidad creativa en la solución de problemas, para elegir las estructuras de datos adecuadas al problema en cuestión; 4. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Manejo de tiempo respecto a la planificación de entrega de su producto; 3. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 4.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 5. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 6. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 7. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.	**1.4. Serialización** **1.5. Fragmentación de archivos** **1.5.1 Fragmentación interna** **1.5.2 Fragmentación externa**	**5%**
1.8. Examen de programación, consistente en completar la implementación de una aplicación software ABC orientada a objetos (diseñada e implementada parcialmente por el profesor, previo al examen); que incluya la implementación de un tipo de entidad de la realidad; que mantenga su información en memoria principal usando diversas estructuras de datos (dinámicas y/o estáticas), que	Saber: 1. Nombres de archivos y extensiones; 2. Clases administradoras de objetos; 3. Modos de acceso de archivos; 4. Escritura y lectura de archivos de texto; 5. Registros de longitud variable con delimitadores; 6. Escritura y lectura de archivos binarios; 7. Registros de longitud variable con campos de dimensión;	**1.1. Clases administradoras de objetos** **1.2. Registros de longitud variable** **1.2.1 Técnica con delimitadores** **1.2.2 Técnica con campos de dimensión** **1.3. Registros de longitud fija con acceso directo**	**20%**

guarden apuntadores a objetos; que almacene los datos de forma persistente en archivos mediante todas y cada una de las técnicas de: registros de longitud variable con delimitadores, registros de longitud variable con campos de dimensión y registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.	8. Registros de longitud fija con acceso directo; Saber hacer: 1. Diseña el cómo organizar los datos almacenables en archivos; 2. Organiza su tiempo para entregar su producto en tiempo y forma; 3. Desarrolla su capacidad creativa en la solución de problemas, para elegir las estructuras de datos adecuadas al problema en cuestión; 4. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Manejo de tiempo respecto a la planificación de entrega del producto; 2. Calidad en la presentación del código fuente; 3.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema;		
2.1. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando estructuras de datos dinámicas, tanto para un índice como para un listado de objetos, ambas estructuras guardando apuntadores a objetos; que almacene los datos de forma persistente en archivos, tanto el índice como el listado de objetos, lo anterior mediante la técnica de registros de longitud fija con acceso directo; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.	Saber: 1. Búsqueda lineal; 2. Búsqueda binaria; 3. Índices primarios en memoria principal; 4. Índices secundarios en memoria principal; 5. Índices en memoria secundaria; 6. TDA Lista basada en cursores; 7. Listas invertidas; Saber hacer: 1. . Organiza su tiempo para entregar programas en tiempo y forma; 2. Desarrolla su capacidad creativa en la solución de problemas, para elegir la estructura de datos adecuada al problema en cuestión; 3. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Manejo de tiempo respecto a la planificación de entrega de productos de software; 3. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 4.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema;	**2.1. Indización** **2.1.1 Índices primarios** **2.1.2 Índices secundarios** **2.1.3 TDA Lista basada en cursores** **2.1.4 Listas invertidas** **2.1.5 Intercambio entre memoria principal y secundaria**	**4%**

	5. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 6. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 7. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.		
2.2. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando una estructura de datos dinámica de tipo tabla de dispersión, que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivos, tanto la tabla de dispersión como el listado de objetos; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.	Saber: 1. Tabla de dispersión en memoria principal; 2. Manejo de colisiones en tabla de dispersión; 3. Saturación progresiva en tabla de dispersión; 4. Compartimientos en tabla de dispersión; Saber hacer: * los mismos que para la actividad 2.1 Saber ser: * los mismos que para la actividad 2.1	**2.2. Tablas de dispersión** **2.3.1 Manejo de colisiones** **2.3.2 Saturación progresiva** **2.3.3 Compartimentos** **2.3.4 Intercambio entre memoria principal y secundaria**	**4%**
2.3. Implementación de una aplicación software ABC orientada a objetos, que gestione un tipo de entidad de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que mantenga su información en memoria principal usando una estructura de datos dinámica de tipo árbol binario de búsqueda, que guarde apuntadores a objetos; que almacene los datos de forma persistente en archivos, tanto el árbol binario paginado como el listado de objetos; y que al iniciarse dicha aplicación, busque y en su caso cargue los datos de tales archivos.	Saber: 1. Árbol binario de búsqueda; 2. Árbol binario paginado. Saber hacer: * los mismos que para la actividad 2.1 Saber ser: * los mismos que para la actividad 2.1	**2.3. Árbol binario paginado** **2.3.1 Árbol binario de búsqueda** **2.3.2 Paginación en disco duro**	**4%**
3.1. Implementación de una aplicación software ABC orientada a objetos, que gestione al menos un tipo de entidad de la realidad (por ejemplo: Aeropuerto, Avión); que mantenga su información en memoria principal usando una estructura de datos grafo estático.	Saber: 1. Representación gráfica de un grafo; 2. Propiedades de los grafos; 3. Grafos no dirigidos; 4. Grafos dirigidos; 5. Grafos no ponderados; 6. Grafos ponderados; 7. TDA Grafo estático; 8. TDA Grafo dinámico. Saber hacer: 1. . Organiza su tiempo para entregar programas en tiempo y forma;	**3.1. TDA Grafo Estático** **3.1.1 Grafos dirigidos / no dirigidos** **3.1.2 Grafos ponderados / no ponderados** **3.2. TDA Grafo Dinámico** **3.2.1 Grafos dirigidos / no dirigidos** **3.2.2 Grafos ponderados / no ponderados**	**4%**

	2. Desarrolla su capacidad creativa en la solución de problemas, para elegir la estructura de datos adecuada al problema en cuestión; 3. Implementa soluciones en el paradigma orientado a objetos. Saber ser: 1. Respeto tanto a las virtudes como a los defectos de las soluciones de sus compañeros; 2. Manejo de tiempo respecto a la planificación de entrega de productos de software; 3. Calidad en la presentación del código fuente y de la interfaz de usuario de la aplicación desarrollada; 4.. Confianza en sí mismo para argumentar la viabilidad de la solución que propone al problema; 5. Formación de una opinión personal, reflexiva o crítica respecto a su propuesta de solución de problemas en contraste con otras soluciones. 6. Evalúa las soluciones software diseñadas por sus pares y las compara con las propias, evaluando tanto la utilización de recursos como en el desempeño de la solución software; 7. Trabaja en equipo en la resolución de situaciones triviales en la escritura de programas de computadora.		
3.2. Implementación de una aplicación software que cumpla en código fuente con todos los requisitos de la actividad anterior (con todas las correcciones en implementación, detectadas durante la retroalimentación de tal actividad); pero que mantenga su información en memoria principal usando una estructura de datos grafo dinámico, requiriendo este entregable únicamente el cambio de la sentencia "#include" que hace referencia a un TDA Grafo estático, para en su lugar incluir un TDA Grafo dinámico.	Mismos que la actividad 3.1	**Mismos que la actividad 3.1**	**4%**

Producto final		
Descripción	**Evaluación**	
Título del producto #1: Aplicación software para gestión de información en disco. **Título del producto #2:** Portafolio de soluciones independientes..	**Criterios de fondo:** Para cada una de las actividades de aprendizaje (producto de cada unidad temática), entregar un video ilustrando: 1. El diseño UML actual del proyecto;	**Ponderación** **5%**
Objetivo del producto #1: Diseñar e implementar una aplicación software que resuelva un problema de la realidad, analizar la serie de requisitos a cumplir acorde al levantamiento de requerimientos, aplicar su capacidad creativa en modelar gráficamente la solución, implementar el producto software utilizando las técnicas adecuadas acordes a la naturaleza de		

los datos a almacenar, con el fin de evaluar la solución propuesta, en contraste con las soluciones de otros compañeros. **Objetivo del producto #2:** Diseñar e implementar soluciones de software que resuelvan problemas pequeños y aislados, demostrar los conocimientos adquiridos para aquellos saberes (conocimientos) que no hayan sido integrados en el Producto #1 (debido solo al diseño y alcances de dicho producto) con el fin de evaluar las soluciones propuestas, en contraste con las soluciones de otros compañeros.	2. El código fuente, mostrando de este exclusivamente los nombres de clases, sus atributos y los prototipos de métodos (sin mostrar la implementación de las subrutinas); 3. El programa en ejecución, mostrando todas las características del programa visibles al usuario de la aplicación; 4. El contenido de los archivos generados por la aplicación, resultado de aplicar la técnica de escritura, propia de cada actividad de aprendizaje, haciendo énfasis en mostrar la ubicación de los datos de tipo cadena y de tipo entero, usando para lo anterior un bloc de notas para abrir el archivo 5. El funcionamiento de la aplicación, que sea resultado de aplicar todas las correcciones solicitadas al código fuente, derivadas de la retroalimentación recibida durante la evaluación de cada una de las actividades de aprendizaje. **Criterios de forma:** Para cada una de las actividades de aprendizaje (producto de cada unidad temática), entregar un video (y no varios videos de una misma actividad), en formato MP4 (cuyo nombre de archivo sea el especificado por el profesor), el cual muestre toda la pantalla de la computadora, e incluya una explicación oral por parte del alumno durante todo el video	
Caracterización del Producto #1: La aplicación de software consiste en una solución, diseñada (mediante lenguaje unificado de modelado, UML solo para el diagrama de clases y sus relaciones) e implementada aplicando el paradigma orientado a objetos (con el lenguaje de programación C++), la cual represente al menos cinco tipos de entidades (clases) de la realidad (por ejemplo: Persona, Libro, Materia, Automóvil, Ciudad); que permita operaciones ABC (al menos incluyendo altas, bajas, y consultas de información para listados de registros/objetos); que mantenga su información en memoria principal usando estructuras de datos (por ejemplo y sin limitar: arreglos, listas, pilas, colas, índices, tablas de dispersión, grafos), ya sea estáticas y/o dinámicas; que almacene los datos de forma persistente en archivos (mediante las técnicas de: registros de longitud variable con delimitadores, registros de longitud variable con campo de dimensión, registros de longitud fija con acceso directo, serialización); y que finalmente, incluya estructuras de datos para búsqueda de información (por ejemplo: índices y/o tablas de dispersión, gestionados en memoria principal y en disco duro, y/o árboles binarios paginados). Dicha aplicación será propuesta y especificada a detalle por el profesor del curso, tanto para los requerimientos a cumplir como en el alcance, con la finalidad de que los estudiantes contrasten la solución propia con la desarrollada por otros estudiantes de su mismo grupo. **Caracterización del Producto #2:** Elaborar pequeños productos de software, diseñados e implementados aplicando el paradigma orientado a objetos (en el lenguaje de programación C++), los cuales modelen cada uno al menos un tipo de entidad de la realidad, que mantengan su información ya sea solo en estructuras de datos en memoria principal y/o sean almacenados los datos de forma persistente en archivos. Dichos productos serán propuestos y especificados a detalle por el profesor del curso, tanto para los requerimientos a cumplir como en el alcance de cada uno, con la finalidad de que los estudiantes contrasten la solución propia con la desarrollada por otros estudiantes de su mismo grupo.		

Otros criterios

Criterio	Descripción	Ponderación

6. REFERENCIAS Y APOYOS

Referencias bibliográficas

Referencias básicas

Autor (Apellido, Nombre)	Año	Título	Editorial	Enlace o biblioteca virtual donde esté disponible (en su caso)
Folk, Michel y B. Zoellick	2000	Estructura de Archivos	Addison Wesley México	
Folk, M. J.; Zoellick, B.; Riccardi, G.	1997	File Structures: An Object-Oriented Approach with C++	Addison Wesley; 3 edition	
Karumanchi, N.	2011	Data Structures and Algorithms Made Easy: 700 Data Structure and Algorithmic Puzzles.	CreateSpace; 1st edition	

Referencias complementarias

Knuth, D. E.	2011	Art of Computer Programming	Addison-Wesley Professional; 3rd edition.	

Apoyos (videos, presentaciones, bibliografía recomendada para el estudiante)

Unidad temática 1:

Unidad temática 2:
http://www.c.conclase.net/ficheros/?cap=007
http://sedici.unlp.edu.ar/bitstream/handle/10915/21978/Documento_completo.pdf?sequence=1

Unidad temática 3:

Unidad temática 4:

Unidad temática 5: