4115 – FUNDAMENTOS DE ALGORÍTMICA Y PROGRAMACIÓN

I - Datos de identificación de la asignatura

Carrera:	Licenciatura en Análisis de Sistemas		
Código:	4115	**Plan:**	2024
Denominación:	Fundamentos de algorítmica y programación		
Área:	Fundamentos de la informática		
Año:	Primer		
Horas con acompañamiento docente (HTD), semanal			4
Horas de Trabajo Independiente del estudiante (HTI), semanal			6
Horas semanales (HS)			10
Cantidad de sesiones			32
Total Horas de Trabajo con el docente (THTD)			128
THD teóricas	64	**THD prácticas**	64
Total de Horas de Trabajo Independiente del estudiante (THTI)			192
Total Horas Académicas (THA)			320
Crédito académico (CA)			12,8
Pre-requisito:	Aprobar el Curso Probatorio de Ingreso.		

Uso exclusivo para fines informativos

II - Fundamentación

La asignatura tiene como objetivo principal proporcionar a los estudiantes los conocimientos fundamentales en programación y algorítmica necesarios para desarrollar software eficiente y lógico.

En la era actual de la información, la capacidad de transformar datos en información es crucial para el logro de los objetivos organizacionales y personales. El software se ha convertido en una parte integral de nuestras vidas, y su desarrollo se ha enfocado en ampliar la utilidad de la computación a través de una variedad de aplicaciones. La programación, como lenguaje de computación y lógica, nos permite secuenciar y ordenar instrucciones de manera que obtengamos resultados correctos y perceptibles.

En esta asignatura, se abordan las estructuras lógicas, los algoritmos y las funciones aritméticas como elementos centrales del aprendizaje de la programación. Los estudiantes aprenderán a ingresar, almacenar, transformar y generar datos de manera deliberada para informar decisiones y automatizar procesos. La programación, en esencia, nos permite interactuar con la computadora, aprovechando las capacidades crecientes de datos y computación para lograr objetivos específicos.

El aprendizaje de la programación no solo se trata de dominar un lenguaje de programación en particular, sino de moldear la mente y el razonamiento de manera que podamos expresar y perfeccionar los requisitos humanos de datos y resultados informáticos. Los estudiantes desarrollarán habilidades de resolución de problemas, pensamiento lógico y diseño de algoritmos, lo que les permitirá enfrentar desafíos complejos y crear soluciones eficientes y robustas.

La asignatura se enfoca tanto en la teoría como en la práctica. Los estudiantes no solo adquirirán conocimientos conceptuales sobre estructuras lógicas, algoritmos y programación, sino que también aplicarán estos conocimientos en proyectos prácticos. A

través de ejercicios y desafíos, los estudiantes desarrollarán su habilidad para escribir código, depurar programas y optimizar la eficiencia de sus soluciones.

Al finalizar la asignatura, los estudiantes tendrán una base sólida en programación y algorítmica, lo que les permitirá abordar problemas de programación de manera estructurada y eficiente. Estarán preparados para enfrentar cursos más avanzados en el área de desarrollo de software y podrán contribuir de manera efectiva en la creación de soluciones informáticas en el ámbito profesional.

La asignatura proporcionar a los estudiantes los conocimientos y habilidades necesarios para desarrollar software eficiente y lógico. A través del estudio de estructuras lógicas, algoritmos y funciones aritméticas, los estudiantes aprenderán a utilizar la programación como herramienta para transformar datos en información útil. La asignatura tiene una naturaleza teórica-práctica, permitiendo a los estudiantes aplicar sus conocimientos en proyectos prácticos y desarrollar habilidades de resolución de problemas y pensamiento lógico. Al finalizar la asignatura, los estudiantes estarán preparados para enfrentar desafíos en el campo de la programación y contribuir de manera efectiva en el desarrollo de software.

III - Competencias a desarrollar

Competencias genéricas

1. Actuar con autonomía.
2. Demostrar capacidad de abstracción, análisis y síntesis.
3. Identificar, plantear y resolver problemas.
4. Identificar, analizar, abstraer, formular y resolver problemas relacionados con sus áreas de competencia.
5. Conocer y saber aplicar técnicas y herramientas actualizadas en sus áreas de competencia.
6. Diseñar, programar, ejecutar, analizar e interpretar resultados de pruebas realizadas en sus áreas de competencia.

Competencias específicas

1. Diseñar y analizar con eficacia algoritmos computacionales, considerando criterios de optimización de recursos y tiempo de ejecución, para resolver problemas mediante soluciones eficientes y sostenibles.
2. Utilizar de forma comprensiva los elementos básicos de la computadora (variables, estructuras de control y operadores), para expresar la lógica de resolución algorítmica dentro del entorno de programación.
3. Aplicar correctamente operadores aritméticos, de asignación y de transposición, para realizar manipulaciones de datos y transformaciones lógicas, optimizando procesos en el desarrollo de programas.
4. Organizar y modularizar con claridad el diseño algorítmico de un programa, utilizando funciones, subrutinas o métodos, para promover la reutilización del código, su escalabilidad y mantenimiento.
5. Seleccionar y utilizar adecuadamente estructuras de datos lineales y no lineales (listas, pilas, colas, árboles, grafos), para administrar conjuntos de datos en memoria y optimizar su procesamiento.
6. Implementar con buenas prácticas la documentación de código fuente, mediante comentarios claros, descripciones funcionales y nomenclaturas coherentes, para facilitar la comprensión y mantenimiento del software.

Uso exclusivo para fines informativos

IV - Cuerpo de conocimientos

Unidad 1: Creación y análisis de algoritmos con eficacia y eficiencia.
Contenidos:

- Algoritmos y la lógica que los impulsa
- Representación de algoritmos (Diagramas de flujo y pseudocódigo)
- Modelización de problemas del mundo real.
- Verificación de algoritmos. Prueba de escritorio.
- Resolución de un problema usando un algoritmo
- Eficiencia de un algoritmo.

Unidad 2: Resolución algorítmica de problemas utilizando elementos de la computadora y su representación.
Contenidos:

- Estructuración de la progresión del programa usando estructuras de secuencia y diseño top-down
- Concepto de Programa. Diferencia entre programa y algoritmo.
- Concepto de lenguaje de programación. Formas de traducir algoritmos a lenguaje de programación.
- Sintaxis y semántica de un programa.
- Arquitectura y funcionamiento de un programa.
- Palabras reservadas.
- Identificadores.
- Tipos de datos simples: numérico, lógico y cadena.
- Constantes y variables (locales y globales).
- Operadores y expresiones.
- Operaciones de asignación.
- Funciones internas.
- Operadores de entrada y salida.
- Estilo de codificación: identificadores, comentarios e indentación.

Unidad 3: Transformación de datos mediante operadores aritméticos, de asignación y unarios.
Contenidos:

- Aplicación de operadores aritméticos para la transformación de datos
- Utilización de operadores de asignación para establecer los valores que se almacenarán en variables asociadas con la memoria volátil
- Utilización la precedencia de operadores y su impacto en la especificación y validación de operaciones
- Aplicación de operadores unarios y transposicionales que modifican los valores almacenados en la memoria volátil
- Aplicación de cualquier palabra clave específica del lenguaje que modifique la disponibilidad o el uso de valores almacenados en la memoria volátil

Unidad 4: Funciones, métodos, subrutinas o estructuras organizativas similares.
Contenidos:

- Modularización. La descomposición de problemas.
- Reusabilidad.
- Conceptos de argumentos y parámetros.
- Conceptos de variables locales y variables globales.
- Concepto algebraico básico de una función matemática
- La modularidad como estrategia de organización y gestión de la complejidad

Uso exclusivo para fines informativos

- Definición de una subrutina/método/función
- Definición de una subrutina/método/función que defina los parámetros que se pasarán como argumentos
- Definición de un valor de retorno funcional para una subrutina/método/función
- Utilización (llamar) de una subrutina/método/función desde otra parte del código
- Aplicación de los beneficios de subrutinas/métodos/funciones bien nombradas que expresan procesos o resultados como acciones o comportamientos (verbo)
- Utilización de cualquier palabra clave específica del lenguaje que modifique la disponibilidad, la persistencia o la visibilidad de una subrutina/método/ función.

Unidad 5: Estructuras de datos lineales y no lineales.
Contenidos:

- Almacenamiento y acceso a un conjunto de valores relacionados desde una única estructura de memoria lógica o física
- Estructuras de datos lineales contiguas (arreglos unidimensionales o vectores y arreglos bidimensionales o matrices)
- Algoritmos de búsqueda en arreglos de una dimensión.
- Algoritmos de ordenación: Selección.
- Estructuras de datos lineales no contiguas y de tamaño dinámico
- Necesidad de utilización estructuras de datos no lineales
- Elementos básicos de los gráficos.
- Principios matemáticos detrás de los algoritmos hash para el almacenamiento y recuperación de datos de la memoria volátil
- Identificación y utilización de bibliotecas de colecciones específicas del lenguaje como implementaciones probadas de estructuras de datos lineales y no lineales

Unidad 6: Buenas prácticas de documentación en la programación.
Contenidos:

- Documentación de una aplicación
- Ventajas y desventajas de documentar una solicitud
- Propósito del código fuente legible
- Las mejores prácticas en la escritura de código fuente
- Programa estructurado
- Corrección de algoritmos.
- Análisis de algoritmos según su tiempo de ejecución y su utilización de memoria.

V - Estrategias didácticas a ser implementadas en el proceso de enseñanza aprendizaje. (abarcando actividades de formación e investigación)

La cátedra consta de actividades teóricas y prácticas.

En la cátedra se organizan actividades por equipos de trabajo, con 2 a 4 alumnos en las actividades prácticas. En principio los alumnos son “pares” sin roles determinados en el equipo, aunque dado un problema a resolver, ellos pueden definir sus roles (notar que se trata de una asignatura del primer año).

Los equipos deben demostrar capacidad de aprender (a partir de problemas planteados en la práctica y ejemplos desarrollados en la teoría), teniendo la posibilidad de consultar a sus docentes. Cada comisión/equipo debe documentar la solución de los ejercicios que se plantean y son examinados en forma individual en las evaluaciones

Uso exclusivo para fines informativos

prácticas (por escrito) y pueden tener que defender sus soluciones en un coloquio de teoría.

La cátedra mantiene planillas que permiten calificar diferentes aptitudes de los miembros del equipo (conocimientos / modo de expresarse / predisposición al trabajo colaborativo). Estas planillas son reunidas por el docente para ser tenidas en cuenta en las evaluaciones parciales y finales de los alumnos.

En el seguimiento y evaluación de los alumnos se trata de formarlos en una metodología de ir del "caso problema del mundo real" a su solución efectiva con herramientas informáticas limitadas al paradigma imperativo, ejemplificadas en programas informáticos y/o lenguajes alternativos (recordar que son alumnos iniciales de la carrera y es su primer curso de Algoritmos). Para ello se pone énfasis en el modo de abstraer el problema y diseñar una solución verificable. La "calidad" de la solución se mide con métricas simples (tiempo de ejecución, estudio de posibles errores).

Dado el contenido del programa que se enfoca en algoritmos y en el empleo de estructuras de datos lineales (vectores y listas básicamente) el alumno es evaluado en todos los aspectos relacionados con las competencias constando el resultado de esta evaluación en la corrección de las pruebas (parciales y finales) del alumno. Se pone énfasis en detallar los aspectos técnicos que debe perfeccionar hacia el futuro en asignaturas que correlacionan con conceptos de Algoritmos, Datos y Programas.

VI - Estrategias de evaluación.

La evaluación será formativa y procesual, se realizará a través de pruebas (exámenes) que podrán ser escritas, orales o de ejecución que a su vez podrá ser mediante trabajos individuales o grupales. La materia consta de dos pruebas parciales, con un recuperatorio y tres oportunidades para la prueba final.

En estos parciales, así como en el examen final, se evaluarán las competencias alcanzadas a través de actividades de contenido teórico y práctico que permitan dar cuenta del avance conceptual en los temas que se han desarrollado, se incorporan preguntas específicas tipo sobre "donde cree Ud. que es aplicable este conocimiento/método" y se refleja en la corrección de las pruebas del alumno.

En algunos temas se trabaja también con ejercitaciones de aplicación en clase, que requieren de un ejercicio de integración de conceptos y que complementan la evaluación a través de los parciales.

Para la obtención de calificaciones parciales y finales se tendrá en cuenta el Reglamento Académico de la universidad..

VII - Actividades de extensión y de responsabilidad social universitaria.

Rige de acuerdo al reglamento de la Universidad y el reglamento interno de la facultad.

VIII - Fuentes bibliográficas

Básica

- Algoritmos, datos y programas con aplicaciones en Pascal, Delphi y Visual Da Vinci. De Giusti, Armando et al. 1er edición. Prentice Hall 2001.

Uso exclusivo para fines informativos

- Estructuras de Datos y Algoritmos. Hernández R., Dormido R., Lazaro J. Ros S. Pearson Education. 2000.
- Introduction to algorithms Comen, Leiserson. MIT Press 2001.
- Estructuras de Datos y Algoritmos. Aho Alfred, Hopcroft John y Ullman Jeffrey. Addison Wesley Publishing Company. EUA. 1998.
- Programación en Pascal Joyanes Aguilar, Luis. Mc Graw Hill. 2006
- Fundamentos de Programación. Libro de Problemas. Joyanes Aguilar L., Fernandez M., Rodríguez L. Mc Graw Hill. 2003.
- Data structures, algorithms and software principles. Standish, T. A. Addison Wesley Publishing Company. 1994.
- Estructuras de Datos y Algoritmos. Weiss, M.A. Addison Wesley. 1995.
- Fundamentos de Programación. Joyanes Aguilar L., Fernandez M., Rodríguez L. Mc Graw Hill. 1999.
- Algoritmos y estructuras de datos y programación orientada a objetos. Flórez Rueda. Ecoe Ediciones. Bogotá. 2005. ISBN 958648394/0
- Programación En C Metodología, Algoritmos Y Estructura De Datos. Joyanes Aguilar Luis – Zahonero Martínez. Segunda Edición –Editorial Mc Graw Hill. España - Edición 2007

Complementaria

- Introduction to Computer Science with applications in Pascal. Garland, S.J. Addison Wesley Publishing Company. 1986.
- Estructuras de Datos. Franch Gutierrez, Xavier. Alfaomega Grupo Editor Argentino.2002
- Estructura de Datos. Joyanes Aguilar C., Zahonero Martinez I. Mc Graw Hill. 1998.
- Estructuras de Datos. Libro de Problemas. Joyanes Aguilar L., Fernandez M., Rodríguez L. Mc Graw Hill. 1999.
- Estructuras de Datos. Lipschutz, S. Mc Graw Hill. 1997.
- Programación estructurada en Turbo Pascal 7. Lopez Roman, L. Alfaomega Grupo Editor Argentino. 1998.
- Estructuras de Datos. Martinez Román, Quiroga Elda. Thomson International. 2002
- Estructura de Datos y Algoritmos. Sisa, Alberto Jaime. Editorial Prentice. 2002.
- Pascal Estructurado. Tremblay, Jean Paul. Mc Graw Hill.1980.
- Data structures, algorithms and performance. Wood, D. Addison Wesley Publishing Company. 1993.
- Structures and Algorithm Analysis in Java. Weiss, M.A. Data, 3rd Edition, Pearson/Addison Wesley, 2011
- Data Structures and Algorithms using C#. M. McMillan. Cambridge University Press, 2006

Uso exclusivo para fines informativos