ANEXO

Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía, Física y Computación

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Matemática Discreta I	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

- Aplicar el principio de inducción a diversas situaciones.
- Entender la mecánica de las demostraciones matemáticas.
- Enfrentar problemas de combinatoria y conteo.
- Entender los principios de divisibilidad básicos.
- Resolver ecuaciones de congruencias y problemas relacionados.
- Entender las nociones básicas de la teoría de grafos.

CONTENIDO

1. Números enteros

Números naturales y enteros. Aritmética. Principio de buena ordenación. Definiciones recursivas. El principio de inducción.

2. Conteo

Principios básicos. Selecciones ordenadas con repetición. Selecciones ordenadas sin repetición. Selecciones sin orden. El teorema del binomio.

3. Divisibilidad

Cociente y resto. Algoritmo de Euclides. Desarrollo en bases. Divisibilidad. El máximo común divisor y el mínimo común múltiplo. Números primos. Factorización en primos

4. Aritmética Modular

Congruencias. Ecuación lineal de congruencia. Teoremas de Fermat y Wilson. Algoritmo RSA.

5. Grafos

Grafos y sus representaciones. Isomorfismo de grafos. Valencias. Caminatas, recorridos, caminos y ciclos. Ciclos hamiltonianos, caminata elueriana y circuitos eulerianos, Árboles. Coloreando los vértices de un grafo. El algoritmo greedy para coloración de vértices.

BIBLIOGRAFÍA BÁSICA

- Tiraboschi, Alejandro. Notas de Matemática Discreta. Para descarga: https://www.famaf.unc.edu.ar/~tirabo/Apunte_MD1_2023.pdf, 2023.
- Biggs, Norman. Matemática Discreta. Barcelona : Vives V., 1998.

BIBLIOGRAFÍA COMPLEMENTARIA

- Gentile, Enzo R. Notas de álgebra I. Buenos Aires : EUDEBA, 1988.
- Patricia Kisbye y Roberto Miatello. Álgebra I / Matemática Discreta I. (Publicaciones de la FaMAF, Serie C).
- Ross, Kenneth A; Wright, Charles R. B. Matemáticas Discretas. México : Prentice-Hall, 1990.
- Ricardo Podestá y Paulo Tirao. Álgebra. Una introducción a la Aritmética y la Combinatoria.

METODOLOGÍA DE ENSEÑANZA

La materia se organiza a partir de una secuenciación progresiva de contenidos que va de los conceptos básicos a los más complejos, comenzando con lógica proposicional y teoría de conjuntos, continuando con relaciones y funciones, axiomas de los números naturales, principios de conteo, grafos, y nociones de inducción y recursividad. Las clases adoptan un formato teórico-práctico, donde la exposición de los conceptos fundamentales se complementa con ejemplos y la resolución guiada y autónoma de ejercicios.

Los recursos didácticos incluyen apuntes, guías prácticas, plataformas virtuales con foros, cuestionarios de autoevaluación y entrega de trabajos. La dinámica de trabajo promueve tanto el aprendizaje individual como el colaborativo, a través del intercambio entre pares.

FORMAS DE EVALUACIÓN

Los/as estudiantes deberán rendir 2 parciales presenciales.
Habrá una instancia de recuperar un solo parcial.
La escala de notas de cada parcial será de 1 a 10 con un decimal y se aprueba cada parcial con 4 o más puntos, lo que corresponde a un 50% del parcial correcto.

REGULARIDAD

Para obtener la regularidad el/la estudiante deberá aprobar al menos 2 parciales. En caso de reprobar un parcial, lo podrá recuperar al final de la materia.

PROMOCIÓN

- Cumplir un mínimo de 80% de asistencia a clases teóricas y prácticas.
- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Análisis Matemático I	**AÑO:** 2025
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Esta es una de las primeras materias que cursan los/as ingresantes a las Licenciaturas en Ciencias de la Computación, en Matemática Aplicada y en Hidrometeorología, por lo cual contribuye al desarrollo del pensamiento matemático de los/as estudiantes, además de constituir uno de los espacios de iniciación en la vida académica universitaria en un centro científico-educativo.

Este curso tiene como objetivo general introducir a sus estudiantes en los conceptos básicos del cálculo matemático en una variable, abordando contenidos fundamentales de funciones, límite, continuidad, derivada e integrales. Estas nociones revolucionaron la matemática del siglo XVII y hoy son básicas en el estudio de otras ciencias. Las mismas surgen de la necesidad de comprender distintos fenómenos permitiendo modelarlos, compararlos y predecir comportamientos futuros.

Los contenidos de la materia pretenden desarrollar principalmente un pensamiento analítico y crítico, que relacione la interpretación geométrica y gráfica con la formulación algebraica. Es por ello que se intentará presentar los distintos temas en forma numérica, gráfica y simbólica.

El objeto central de la materia es el estudio de las funciones reales en una variable. La noción de función permitió concentrar la información de una diversidad de datos o mediciones asociados a una variable específica, en un solo objeto matemático. El análisis de este objeto permite inferir el comportamiento de estos datos. Dentro del comportamiento de las funciones y el análisis de sus gráficos se analiza el crecimiento, los puntos críticos, la continuidad, su comportamiento cercano a un valor determinado de la variable o cuando la variable crece indefinidamente. Se introduce la noción de derivada como herramienta para medir el crecimiento de una función alrededor de un valor determinado de la variable que permite profundizar más detalladamente en el comportamiento de las funciones. Asimismo, se presenta el concepto de integral como herramienta para medir áreas. Ambos conceptos son fundamentales en el análisis de las funciones que trasciende las técnicas de cálculo, por lo cual es importante la comprensión de los mismos en su sentido geométrico y analítico.

El valor del alcance y la profundidad del estudio de las funciones es incompleto si no se comprende su utilidad y se analizan sus aplicaciones en situaciones del entorno cotidiano o de otras ciencias. Es importante en la formación de los/as estudiantes desarrollar las capacidades de interpretación de diversas situaciones en términos matemáticos y la interpretación de los resultados matemáticos obtenidos en el contexto de procedencia del problema. En ese sentido, esta materia contribuye en

ese desafío y se aborda en forma transversal en toda la asignatura, en la medida que los temas en particular así lo permitan.
Los objetivos a lograr en este curso es que los/as estudiantes desarrollen capacidad o adquieran destreza y habilidad para:
- Aprender la simbología matemática básica inherente a los números reales y las funciones, como así también su utilización en la escritura de afirmaciones y demostraciones en lenguaje matemático.
- Manipular e interpretar el sentido de las desigualdades y del valor absoluto en el contexto de este curso.
- Interpretar el gráfico de funciones y reconocer funciones algebraicas, exponenciales, logarítmicas y trigonométricas.
- Comprender la noción de límite de una función cuando la variable tiende a un valor determinado o crece indefinidamente, como así también reconocer la definición formal. Saber calcular límites.
- Dominar la noción de continuidad y las propiedades de las funciones continuas.
- Comprender el concepto de derivada de una función en un punto, su significado geométrico y su sentido como medida del crecimiento de la función. Saber calcular derivadas.
- Aplicar los conceptos de límite y derivada para estudiar máximos y mínimos de funciones. Poder resolver problemas simples de optimización.
- Utilizar las herramientas analíticas trabajadas durante el curso para graficar funciones.
- Comprender la noción de integral de una función y su significado geométrico. Saber calcular integrales y aplicarlas en la resolución de problemas sencillos.
- Ser capaz de traducir un problema planteado en lenguaje coloquial a lenguaje matemático, resolverlo e interpretar su solución en el contexto del planteo del problema.
- Realizar demostraciones sencillas de algunas afirmaciones matemáticas.

CONTENIDO

Unidad I: Números y Funciones

Números enteros, racionales y reales. Desigualdades. Valor absoluto. Funciones. Definición. Ejemplos. Gráficas de funciones. Funciones inyectivas, suryectivas y biyectivas. Rectas, parábolas, circunferencia, elipse. Funciones trigonométricas. Funciones exponenciales y logarítmicas. Propiedades, ejemplos y aplicaciones

Unidad II: Límite y continuidad

Definición intuitiva de límite. Ejemplos. Límites laterales. Relación entre la existencia de límites laterales y la de límite. Límites infinitos. Límite cuando la variable tiende a infinito. Límites infinitos cuando la variable tiende a infinito. Límites notables. Definición de continuidad en un punto. Continuidad por derecha y por izquierda. Definición de continuidad en un intervalo. Propiedades. Teorema de Weierstrass. Aplicaciones.

Unidad III: Derivada

Definición de función derivable en un punto. Ejemplos. Reglas de derivación. Propiedades. Regla de la cadena. Derivadas de orden superior. Derivada de funciones trigonométricas. Derivada de funciones exponenciales. Derivada de la

función inversa. Derivada de funciones trigonométricas inversas. Algo sobre el número e. Derivada de funciones logarítmicas. Aplicaciones.

Unidad IV: Valores máximos y mínimos. Gráficas
Definición de punto de máximo (mínimo) y de valor máximo (mínimo) locales y absolutos. Ejemplos. Teorema de Fermat. Máximos y mínimos en intervalos cerrados. El Teorema de Rolle y el Teorema del valor medio. Teorema del valor medio de Cauchy. La regla de L`Hopital. Funciones crecientes y decrecientes. Propiedades. Concavidad y puntos de inflexión. Prueba de concavidad. Prueba de la segunda derivada. Gráficas. Aplicaciones.

Unidad V: Integrales
La integral indefinida de una función continua. Área. Suma de Riemann. Teorema fundamental del cálculo. Propiedades básicas de la integral indefinida. Técnicas de integración: Método de sustitución, integración por partes. Aplicaciones al cálculo de áreas y volúmenes. Aplicaciones.

BIBLIOGRAFÍA BÁSICA

- M. Urciuolo, P. Kisbye, Notas de Análisis Matemático I, 2019.
- L. Leithold, El cálculo, México : Oxford University, 1998.
- S. Lang, Cálculo, Buenos Aires : Addison-Wesley Iberoamericana, 1990.

BIBLIOGRAFÍA COMPLEMENTARIA

- Stewart, J. Cáculo, Trascendentes tempranas, México : Cengage Learning, 2013.
- Spivak, M. Calculus, Barcelona : Reverté, 1970.

METODOLOGÍA DE ENSEÑANZA

La metodología de enseñanza consiste en clases teórico-prácticas en 2 encuentros de 4 horas semanales cada una y en grupos de estudiantes reducidos a cargo de un/a docente. Esta modalidad permite que el/la docente pueda hacer un seguimiento más preciso y cuidado del aprendizaje de los/as estudiantes.

Se utiliza como material de estudio principal unas notas escritas por docentes de la Facultad que incluyen guías de ejercicios y problemas prácticos para las diferentes unidades de la materia. Recuperando las producciones del estudiantado, el/la docente gestiona en clases la discusión en torno a la resolución de algunos puntos de las guías, dejando otros para que los/as estudiantes resuelvan de manera autónoma en interacción con sus pares. Los/as docentes responden consultas que puedan surgir del trabajo sobre las guías.

Se usa el aula virtual (Moodle) donde, además de las guías de prácticos, se suele agregar información adicional, como resoluciones grabadas de ejercicios, problemas adicionales, cuestionarios de autoevaluación, etc.

FORMAS DE EVALUACIÓN

- Se tomarán dos exámenes parciales, que deberán ser aprobados como uno de los requisitos para regularizar la materia. Solo se podrá recuperar uno de ellos.
- El examen final constará de una evaluación escrita sobre todos los contenidos teórico-prácticos desarrollados en el curso. Los/as estudiantes libres deberán resolver correctamente una serie de ejercicios extra antes de acceder al examen final.

REGULARIDAD

- Cumplir un mínimo de 70% de asistencia a clases teórico-prácticas.
- Aprobar los 2 (dos) exámenes parciales, o un parcial y el recuperatorio de otro, con calificación mayor o igual a 4 (cuatro).

PROMOCIÓN

- Cumplir un mínimo de 80% de asistencia a clases teórico-prácticas.
- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Introducción a los Algoritmos	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Introducción a los Algoritmos es la primera materia de la Licenciatura en Ciencias de la Computación directamente relacionada con la programación. Se busca que el/la estudiante pueda adquirir por un lado cierta familiaridad en la manipulación de un lenguaje formal, comenzando con la aritmética y continuando con un lenguaje de programación funcional, lógica proposicional y lógica de primer orden; y por el otro, comprender a los programas como un objeto formal, con una sintaxis y semántica bien definida, cuyo comportamiento puede describirse rigurosamente. Como paradigma de programación que atraviesa estos contenidos se elige el paradigma funcional, debido a la simplicidad de su sintaxis.

Los objetivos que se buscan en esta materia son que el/la estudiante adquiera:
- capacidad de análisis de problemas
- formalización a soluciones de problemas
- manipulación de expresiones formales
- pruebas de corrección de expresiones formales
- familiaridad con conceptos básicos de programación.

CONTENIDO

Unidad I: Introducción
Historia de la Computación. Software libre.
Introducción a la metodología de trabajo con expresiones aritméticas. Precedencia y tipado. Validez y satisfacibilidad. Funciones.

Unidad II: Introducción a la programación funcional
Formalismo básico. Números naturales.
Tuplas. Listas, constructores y operadores, propiedades. Modelo computacional. Diseño de programas recursivos. Demostraciones por inducción.

Unidad III: Semántica de la lógica proposicional
Operadores Booleanos. Tablas de Verdad. Equivalencia, disyunción, conjunción, implicación, negación, discrepancia. Representación del conocimiento en lógica proposicional. Introducción al análisis de razonamientos.

Unidad IV: Cálculo proposicional

Estructura de las pruebas formales. Axioma y teoremas. Propiedades de la lógica proposicional. Demostraciones: Equivalencia, disyunción, conjunción, implicación, negación, discrepancia.

Unidad V: Cálculo de predicados
Noción de predicado. Cuantificador universal. Cuantificador existencial. Enfoque semántico (interpretación) y enfoque sintáctico (leyes). Demostraciones.

Unidad VI: Especificaciones
Representación del conocimiento en lógica de predicados. Concepto de especificación formal de un problema. Ejemplos y resolución de problemas.

BIBLIOGRAFÍA BÁSICA

- *Cálculo de Programas*, J. Blanco, D. Barsotti, S. Smith, 2009.
- *Discrete Mathematics Using a Computer*, John O'Donnell, Cordelia Hall and Rex Page. 2nd Edition, Published by Springer, 2006. Versión traducida al español 2015.
- *Learn You a Haskell for Great Good! A Beginner's Guide* by Miran Lipovača. No Starch Press. 2011. Versión traducida al español bajo el título Aprendé Haskell por el bien de todos. Disponible en http://aprendehaskell.es/ bajo licencia CC 3.0.

BIBLIOGRAFÍA COMPLEMENTARIA

- Material de Estudio. Acosta, Cherini, Losano, Pagano, 2014.
- Apuntes de clases. Areces, Benotti, Fervari. 2022.

METODOLOGÍA DE ENSEÑANZA

Los contenidos del programa se presentan a los/as estudiantes organizados en cuatro guías de trabajo. Estos materiales didácticos contienen junto a los ejercicios y problemas, material de lectura relevante a los temas a trabajar que sirve como resumen y repaso de los contenidos que abordan. Los contenidos cubiertos en las dos primeras guías son evaluados en el primer parcial, mientras que los contenidos de las dos últimas guías son el foco del segundo parcial. De todas formas, por la naturaleza de los contenidos dados, cada guía construye sobre los contenidos de las anteriores. Los temas cubiertos en las diferentes guías son los siguientes:
Guía 1: noción de algoritmo, diferencias entre algoritmo e implementación, tipos de datos, funciones, composición de funciones, funciones por casos, tuplas.
Guía 2: listas, tipos de listas, tipos complejos, tipos polimórficos, recursión, funciones de tipo map, filter y fold, inducción, demostración de propiedades.
Guia 3: la lógica proposicional, satisfacción y validez, tablas de verdad, el tipo Bool, funciones booleanas, cálculo proposicional.
Guia 4: la lógica de predicados, cálculo de predicados, especificación de funciones, propiedades de listas, verificación.
La gran mayoría de las clases se desarrollan en formato teórico práctico, donde los contenidos son presentados gradualmente e inmediatamente ejemplificados con ejercicios y problemas. Los puntos de las guías están organizados de acuerdo a su complejidad, de forma que los/as alumnos/as pueden realizar fácilmente los

primeros, que sirven como ejemplos de los contenidos presentados, para luego pasar a ejercicios y problemas más complejos. Se promueve el trabajo en grupo (e.g., de a dos estudiantes, trabajando en una computadora), y las soluciones obtenidas se comparten y discuten luego con toda la clase. Antes de los parciales se organizan clases puramente prácticas y de consulta, para afianzar los contenidos que serán evaluados. Nunca se presentan contenidos nuevos la clase anterior a la evaluación, que siempre se dedica a ejercitación y consultas.
La materia se desarrolla en comisiones presenciales y comisiones virtuales, de presencialidad remota sincrónica, de acuerdo a lo establecido por la OHCS 8/22.
La asignatura cuenta además con un Aula Virtual en la plataforma Moodle donde se pone a disposición del estudiantado los diferentes materiales didácticos (guías de trabajo, bibliografía y material de lectura adicional, que puede utilizarse para completar los temas presentados durante las clases). El Aula Virtual cuenta también con un foro de consultas abierto al estudiantado de todas las comisiones, es decir las consultas de los/as alumnos/as, y respuestas de los/as profesores/as son accesibles a todos/as. Finalmente, en la plataforma Moodle los/as docentes comparten videos con resolución de ejercicios puntuales, así como la grabación de las clases dictadas por las comisiones virtuales. Estos videos son utilizados por muchos/as estudiantes (tanto de presencialidad física como remotal) como material de repaso y afianzamiento de contenidos.
Las comisiones virtuales implementan, además, grupos de chat específicos por comisión, como vías de comunicación adicionales entre pares, y entre docentes y estudiantes. Se crean dos grupos por comisión. Uno de los grupos es de bajo tráfico, y es utilizado principalmente por docentes para realizar anuncios importantes (e.g., nuevas guías, fechas de exámenes, etc.). El otro grupo es de alto tráfico, los/as docentes promueven el uso de este canal como extensión áulica y estimulan a los/as estudiantes a usar este medio para comunicar dudas, hacer preguntas, y discutir entre pares. Aquí, las intervenciones docentes atienden principalmente a realizar aclaraciones, o corrección de errores en las soluciones propuestas. También se comunican por este medio noticias interesantes sobre temas generales de Ciencias de la Computación.

FORMAS DE EVALUACIÓN

- Dos exámenes parciales escritos, presenciales, con sus respectivos recuperatorios.
- Examen final escrito.

REGULARIDAD

- Aprobar las dos evaluaciones parciales o sus correspondientes recuperatorios.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).
- Cumplir un mínimo de 80% de asistencia a clases teóricas, prácticas, o de laboratorio.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Álgebra	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 2°
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Los contenidos de esta materia tienen como eje adquirir los conceptos básicos relacionados con espacios vectoriales y transformaciones lineales. Para ello se buscará comprender la resolución de sistemas de ecuaciones lineales, sistemas que aparecen naturalmente en las ciencias exactas y naturales. En primer lugar se definirá el concepto de cuerpo, seguido por el de espacio vectorial, junto con numerosos ejemplos. A continuación se estudiarán sistemas de generadores y conjuntos linealmente independientes, para abordar el concepto de sistemas lineales y sus espacios de soluciones. Ello permitirá manejar adecuadamente la descripción de dichas soluciones, identificar y plantear sistemas de ecuaciones lineales y resolverlos utilizando el Método de Gauss.
La sistematización del Método de Gauss conduce al empleo del lenguaje matricial que permite manipular de forma ágil los sistemas de ecuaciones utilizando las operaciones del Álgebra de matrices. También, posibilita buscar criterios sobre la existencia o no, la unicidad o multiplicidad de soluciones de un sistema en término de la matriz asociada. Aquí, aparecen las nociones de Determinante y Matriz inversa.
Las transformaciones lineales entre espacios vectoriales junto a las nociones de coordenadas y cambios de bases permiten formalizar ideas usadas intuitivamente al principio de la materia, por ejemplo, la utilización de vectores en vez de polinomios para resolver ecuaciones referidas a estos últimos.
En este recorrido vislumbramos tres etapas en las que avanzaremos en generalización y abstracción de ideas y propiedades, permitiendo así reconstruir el recorrido del quehacer matemático. Asimismo, se pretende reafirmar en cada una de estas etapas el valor de la demostración rigurosa en la matemática como ciencia.

Se proponen como objetivos de este curso que los y las estudiantes desarrollen capacidades o adquieran destrezas y habilidades en:
- Manejar los conceptos de espacios vectoriales, dimensión, transformaciones lineales, núcleo e imagen de una transformación, sus significados y relaciones con sistemas de ecuaciones.
- Comprender la relación existente entre los sistemas de ecuaciones y los conceptos abstractos de álgebra de matrices y espacios vectoriales.
- Identificar problemas que involucren sistemas de ecuaciones lineales en diferentes contextos, plantearlos matemáticamente y resolverlos con las técnicas estudiadas y expresar las respuestas de forma pertinente.
- Aprender la simbología matemática básica inherente a matrices, espacios vectoriales y transformaciones lineales. También, su utilización en la escritura de

afirmaciones y demostraciones en lenguaje matemático.
- Comprender las demostraciones de los teoremas principales relacionados con los contenidos de la materia reafirmando el valor de una demostración rigurosa en la matemática como ciencia.

CONTENIDO

Unidad I:
Cuerpos. Definición y Ejemplos. El cuerpo de los números complejos. Descomposición polar, Teorema de Moivre, raíces n-ésimas, raíces de la unidad.

Unidad II:
Sistemas de ecuaciones lineales, sistemas de ecuaciones equivalentes, matriz asociada a un sistema de ecuaciones, operaciones elementales por filas, matrices reducidas por filas en escalera, matrices equivalentes por filas. Matrices, operaciones con matrices, propiedades de las operaciones con matrices, matrices invertibles.

Unidad III:
Definición y cálculo de determinantes, alternancia, desarrollo por una fila o columna, determinante de un producto. Matrices invertibles y determinantes.

Unidad IV:
Espacios vectoriales, subespacios, combinación lineal de vectores, conjuntos linealmente independientes y linealmente dependientes, bases y dimensión, Teorema de la dimensión de la suma de subespacios. Bases ordenadas, coordenadas lineales, matriz de cambio de base, aplicación de las operaciones por filas al cálculo de subespacio generado por un conjunto finito de vectores.

Unidad V:
Transformaciones lineales, imagen y núcleo, teorema de la dimensión, el álgebra de las operadores lineales, matriz de una transformación lineal, rango fila igual a rango columna de una matriz, dimensión del espacio de las transformaciones lineales, cambio de bases, caracterización de las transformaciones lineales biyectivas, isomorfismos, matrices semejantes, funcionales lineales, el espacio dual, la transpuesta de una transformación lineal.

Unidad VI:
Autovalores y autovectores de un operador lineal, polinomio característico, operadores diagonalizables.

Unidad VII:
Espacios con producto interno, desigualdad de Cauchy-Schwarz y desigualdad triangular. Bases ortogonales y ortonormales, ortogonalización de Gram-Schmidt.

BIBLIOGRAFÍA BÁSICA

- GERONIMO, G., SABIA, J., TESAURI, S. Álgebra Lineal. Universidad de Buenos

Aires. (2008). https://cms.dm.uba.ar/depto/public/Cursodegrado/fascgrado2.pdf
- HOFFMAN,K., KUNZE,R. Álgebra Lineal. México: Prentice-Hall,1973.

BIBLIOGRAFÍA COMPLEMENTARIA

- ANTON,H. Introducción al álgebra lineal, Limusa Wiley (2011).
- GENTILE, E. Espacios Vectoriales. Buenos Aires, 1968.
- HEFFERON, J. Linear Algebra, A Free text for a standard US undergraduate course (2021). http://joshua.smcvt.edu/linearalgebra/
- MEYER, C. Matrix analysis and applied linear algebra. Philadelphia : Society for Industrial and Applied Mathematics. SIAM, c2000.

METODOLOGÍA DE ENSEÑANZA

Las clases son de cuatro horas, dos veces por semana y se organizan en clases teóricas y prácticas.
Las primeras dos horas se destinan a mostrar resultados teóricos junto con sus aplicaciones a la resolución de problemas. Las clases teóricas están a cargo de los/as docentes responsables de la materia. Si bien su carácter es principalmente expositivo existen interacciones con el estudiantado tendientes a responder dudas que surgen del desarrollo de los contenidos enseñados.
Las siguientes dos horas están a cargo de los/as docentes del práctico y se organizan en comisiones con un número más reducido de estudiantes. Las clases prácticas se centran en la actividad y producción de los/as estudiantes y en ellas se propone al estudiantado trabajar sobre guías prácticas diseñadas por el cuerpo docente para cada unidad del programa. La gestión de las clases prácticas involucra la discusión y resolución de ejercicios y problemas de las guías en el pizarrón. Asimismo se destina tiempo al trabajo autónomo y colaborativo de los/as estudiantes y se atienden consultas a partir de las dificultades que estos encuentran durante el proceso de resolución las guías prácticas.
En el aula virtual de la materia se comparten referencias bibliográficas, las guías de ejercicios y diferentes materiales que son relevantes para la materia. Allí, los/as estudiantes disponen de un foro de consulta.

FORMAS DE EVALUACIÓN

- La materia consta de dos evaluaciones parciales, con un recuperatorio cada una. Cada parcial tiene un ochenta por ciento de contenidos prácticos y el restante veinte se utiliza para evaluar nociones teóricas (enunciados o definiciones).
- El examen final es teórico-práctico y escrito. De un total de 100 puntos, 70 se reservan a ejercicios prácticos y los 30 restantes para enunciar y probar resultados de la parte teórica. Para aprobarlo, se deberán aprobar cada una de las partes (teórica y práctica) por separado.

REGULARIDAD

- Para obtener la regularidad se deben aprobar ambos parciales o sus correspondientes recuperatorios (con 4 puntos o más), y además se necesita el 70%

de asistencia a clases.

PROMOCIÓN

- No hay régimen de promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Análisis Matemático II	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El concepto de integral y los rudimentos del cálculo multivariado constituyen herramientas fundamentales en las ciencias básicas y en ciencias de la computación.
Esta materia tiene por objetivo que sus estudiantes puedan resolver problemas clásicos del cálculo de integrales usando los distintos métodos (directo, sustitución, por partes, fracciones simples, funciones racionales de senos y cosenos). Además que puedan aproximar funciones por su desarrollo de Taylor y estimar el error. Estudiar con detalle las sucesiones y series y sus criterios de convergencia (incluyendo el radio de convergencia para series de potencias). Respecto al cálculo multivariable (dos y tres variables) se espera que los/as estudiantes puedan entender los conceptos básicos de derivadas parciales, direccionales y gradientes (usándolos para encontrar extremos locales y globales), Es importante que comprendan los conceptos de curvas y superficies de nivel.

CONTENIDO

Unidad I: Integración
Repaso de la noción de límite, derivada y cálculo de derivadas. Primitivas o antiderivadas, sumas superiores e inferiores, integral definida. Integral indefinida y Teorema Fundamental del Cálculo. Regla de Barrow. Métodos de integración: sustitución, por partes, fracciones simples, funciones racionales de seno y coseno.

Unidad II: Sucesiones y series numéricas
Paradojas de Zenón de Elea. Límite de sucesiones, propiedades y criterios de convergencia. Ejemplos. Límite de series, propiedades y criterios de convergencia.

Unidad III: Series de potencias y series de Taylor
Series de potencias. Radio e intervalo de convergencia. Polinomios de Taylor, estimación del error, serie de Taylor.

Unidad IV: Cálculo vectorial
Cálculo vectorial en espacios de dos y tres dimensiones. Ecuación vectorial de una recta, ecuaciones implícita y vectorial de un plano. Distancia de un punto a un plano y a una recta. Curvas en el espacio, vector tangente.

Unidad V: Cálculo multivariable
Funciones reales de dos y tres variables. Derivadas parciales, direccionales y

gradiente. Composición de funciones y regla de la cadena. Gráfico de funciones de dos variables, plano tangente. Curvas y superficies de nivel, ecuación del plano tangente. Máximos y mínimos locales de funciones de dos variables, puntos críticos, máximos globales.

Unidad VI: Integrales Múltiples
Integración en varias variables. Volumen de una región en el espacio. Coordenadas curvilíneas. Cambio de variable.

BIBLIOGRAFÍA BÁSICA

- Stewart, James. Cálculo de una variable: trascendentes tempranas. México : International Thomson, 1998.
- Stewart, James. Cálculo de varias variables: trascendentes tempranas. México : International Thomson, 1999.

BIBLIOGRAFÍA COMPLEMENTARIA

- Boyallián, Carina; Ferreyra, Élida Vilma; Urciuolo, Marta Susana; Will, Cynthia Eugenia. Un segundo curso de cálculo.Córdoba, Argentina : Universidad Nacional de Córdoba. Facultad de Matemática, Astronomía y Física, 2007.

METODOLOGÍA DE ENSEÑANZA

Los contenidos del programa se organizan en unidades temáticas las cuales se relacionan entre sí de manera gradual, a cada una de ellas le corresponde una guía de trabajos prácticos.
Para el desarrollo de la asignatura se opta por combinar clases teóricas (2 hs) con clases prácticas (2 hs.). En las clases teóricas el/la docente presenta los conceptos centrales de la materia. Si bien estas clases se caracterizan por ser principalmente expositivas, se estimula la participación del estudiantado a través de la presentación de dudas o inquietudes que el desarrollo teórico pueda suscitar. Las clases prácticas hacen foco en la actividad y la producción de los/as estudiantes mediante la invitación a resolver, reunidos en pequeños grupos, los ejercicios y problemas de las guías diseñadas para cada unidad de la materia. Durante la gestión de las clases prácticas el/la docente discute y resuelve en el pizarrón problemas seleccionados por revestir una dificultad particular o por ilustrar, de forma modélica, el vínculo con el desarrollo teórico. Además, se atienden dudas y consultas que surgen del trabajo de los/as estudiantes con las guías.
La materia cuenta con un Aula Virtual donde se pone a disposición todo el material bibliográfico y las guías de trabajos prácticos. En este entorno los/as estudiantes pueden realizar consultas a través de un foro creado para tal fin.

FORMAS DE EVALUACIÓN

Habrá dos exámenes parciales, con sus correspondientes recuperatorios, un tercer parcial para obtener la promoción y un examen final escrito.

EX-2025-01045526- -UNC-ME#FAMAF

REGULARIDAD

Cumplir un mínimo de 70% de asistencia a clases teóricas y prácticas. Aprobar los dos exámenes parciales o sus correspondientes recuperatorios.

PROMOCIÓN

- Cumplir un mínimo de 80% de asistencia a clases teóricas y prácticas.
- Aprobar las tres evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía, Física y Computación

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Algoritmos y Estructuras de Datos I	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Es habitual que una primera materia de programación presente las construcciones más comunes a los lenguajes de programación (ya sean tipos de datos básicos y estructuras de control para lenguajes imperativos o tipos de datos básicos, condicionales y esquemas de recursión para lenguajes funcionales). Además de someter a las idiosincracias propias del lenguaje elegido, se dan explicaciones intuitivas sobre la semántica operacional de cada construcción.
Una manera alternativa de introducir la programación es partiendo de su especificación, es decir, de una descripción detallada y precisa (eventualmente en un lenguaje formal) de lo que el programa resuelve. A partir de aquella se pueden utilizar técnicas formales para construir (derivar) el programa de manera que el mismo satisfaga su especificación; es decir, que el programa sea correcto por construcción. Varias de esas técnicas se pueden utilizar para verificar si un programa dado satisface una especificación.
Más allá de la formalidad involucrada en la derivación y verificación de los programas, partir de la especificación permite introducir conceptos y abstracciones asociadas a la programación a pequeña escala relacionándolos con nociones análogas en otros dominios (como los números naturales). Finalmente, la noción de corrección de programas respecto a su especificación tiene un correlato con la semántica operacional del lenguaje.

Objetivos:
• Adquirir capacidad de usar un lenguaje formal para especificar algoritmos sencillos.
• Comprender la distinción entre especificación e implementación y la noción de corrección.
• Familiarizarse con los conceptos fundamentales de la programación funcional: reducción de expresiones, tipos, funciones de alto orden, recursión, acumular resultados parciales, tipos de datos algebraicos.
• Familiarizarse con los conceptos fundamentales la programación imperativa: estado, pre-condición y post-condición, invariante y función de terminación, arreglos, programa como transformador de predicados.
• Adquirir capacidad para derivar y verificar programas funcionales sencillos respecto a su especificación formal.
• Desarrollar capacidad de programar en un lenguaje funcional (distinción entre expresiones y tipos; reducción de expresiones; funciones de alto orden; composición de funciones; definición de tipos; organización modular).

• Adquirir capacidad para derivar y verificar programas imperativos sencillos respecto a su especificación formal.
• Desarrollar capacidad de programar en un lenguaje imperativo.
• Comprender la relación entre la especificación y la semántica operacional.
• Adquirir capacidad y hábito de identificar abstracciones al abordar un problema.
• Familiarizarse con técnicas frecuentes de diseños de algoritmos.

CONTENIDO

1. Expresiones Cuantificadas

Repaso de especificaciones con cuantificadores lógicos, revisión de la sustitución y la regla de Leibniz, reglas generales para las expresiones cuantificadas, cuantificadores aritméticos y lógicos.

2. Construcción de programas funcionales

Repaso de cuestiones elementales de un lenguaje funcional: tipos, términos, reducción, pattern-matching. Especificaciones, verificación y derivación.

3. Técnicas elementales para la programación funcional

Definiciones recursivas, modularización, generalización. Segmentos de listas.

4. Modelo computacional de la programación imperativa

Estados, predicados sobre estados. Lenguaje de programación imperativo (skip, abort, asignación, composición secuencial, alternativa, repetición). Ejecución de un programa imperativo a través de la transición de estados (semántica operacional).

5. Especificación y corrección de programas imperativos

Pre-condición, post-condición e invariantes.
Pre-condición más débil de cada construcción del lenguaje.

6. Cálculo de programas imperativos

Uso de obligaciones de prueba para verificación y derivación a partir de la precondición más débil. Derivación de ciclos. Técnicas para determinar invariantes.

7. Programas imperativos sobre arreglos

Definición de arreglos, invariantes sobre arreglos.

Proyectos de Laboratorio

Linux y consola. Haskell, GHCI.
Proyecto 1: Funciones estándares sobre listas. Ejemplos tipos de datos. Tipos de datos, deriving, case, Maybe.
Proyecto 2: Módulos, TADs, instanciaciones de clases. Lista con invariante de orden. Tipos. Polimorfismo ad-hoc. Type Clases.
Proyecto 3: Modelo computacional imperativo comparado con modelo funcional. Programación C, GDB.
Proyecto 4: Teórico de Arreglos, Código Arreglo, Inicialización de arreglos. Estructuras.
Los proyectos se realizan en las computadoras de los laboratorios de la institución.

Los lenguajes de programación utilizados son Haskell y C con solo las construcciones necesarias para resolver los ejercicios.

BIBLIOGRAFÍA BÁSICA

- Cálculo de programas. Javier Blanco, Silvina Smith, Damián Barsotti, Córdoba: Universidad Nacional de Córdoba, 2008.

BIBLIOGRAFÍA COMPLEMENTARIA

- Programming: the derivation of algorithms, Anne Kaldewaij, Prentice-Hall, 1990.

METODOLOGÍA DE ENSEÑANZA

El abordaje didáctico adoptado en la materia se basa en un enfoque activo y orientado a la práctica, diseñado específicamente para favorecer la comprensión temprana de los conceptos fundamentales de algoritmia y programación. En lugar de impartir clases teóricas extensas y monolíticas, se trabaja con un modelo en el que los contenidos conceptuales surgen como respuesta a problemas concretos planteados en los trabajos prácticos. De este modo, la teoría no aparece como un cuerpo de conocimiento abstracto o desconectado, sino como una herramienta necesaria para resolver situaciones reales. Esta dinámica favorece la participación, mantiene la atención del estudiantado en intervalos más breves y conectados con tareas específicas, y facilita la transferencia del conocimiento hacia la práctica.

Para acompañar esta metodología, las clases presenciales se desarrollan mediante presentaciones dinámicas proyectadas desde una computadora. Estas presentaciones combinan explicaciones guiadas, ejemplos progresivos, esquemas visuales y fragmentos de código que ilustran de forma clara la construcción de soluciones algorítmicas. El uso de recursos visuales y multimedia contribuye a un aprendizaje más ágil, reduce la carga cognitiva inicial y permite avanzar de manera incremental, mostrando cómo las ideas se integran paso a paso.

Además de las instancias presenciales, se ponen a disposición de los/as estudiantes clases grabadas en video. Este material audiovisual les permite revisar los contenidos con mayor calma, reforzar los conceptos que presentaron más dificultad o recuperar una clase en caso de ausencia. Las grabaciones también favorecen la autonomía y el ritmo individual de aprendizaje, ya que cada estudiante puede pausar, retroceder o revisar los ejemplos tantas veces como lo necesite.

En conjunto, esta metodología promueve un aprendizaje significativo, activo y accesible, en el que la teoría y la práctica avanzan de manera integrada, y en el que los/as estudiantes encuentran múltiples apoyos —presenciales y virtuales— para construir sus primeras competencias en programación y resolución algorítmica de problemas.
Todos los materiales y recursos didácticos referidos son puestos a disposición de los/a estudiantes a través de un Aula Virtual Moodle. En este entorno, se establecen interacciones pedagógicas a través de un foro de consultas donde los/as estudiantes pueden plasmar sus dudas e inquietudes respecto a los contenidos que se desarrollan en clase.

FORMAS DE EVALUACIÓN

- 2 evaluaciones parciales de teórico/práctico donde, se podrá recuperar una instancia.
- 2 evaluaciones de trabajo de laboratorio, donde se podrá recuperar una instancia.

La evaluación final consiste en un examen escrito y un examen de laboratorio. Quienes rindan regular, sólo deberán rendir el examen escrito.

REGULARIDAD

- Aprobar los dos trabajos prácticos de laboratorio o sus correspondientes recuperatorios (podrá recuperar una sola instancia).

PROMOCIÓN

La materia no tiene promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Análisis Numérico	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Es de gran importancia que estudiantes de las Licenciatura en Ciencias de la Computación, Licenciatura en Matemática y Licenciatura en Matemática Aplicada adquieran las herramientas básicas para formular y resolver problemas de matemática aplicada, utilizando de manera óptima algoritmos y computadoras.

En esta materia el/la estudiante logrará:

- conocer los algoritmos para resolver problemas básicos de matemática aplicada;
- discernir acerca de la técnica más conveniente para resolver cada problema;
- implementar el algoritmo en un lenguaje de programación;
- interpretar los resultados obtenidos computacionalmente.

CONTENIDO

1. Análisis de errores

Error absoluto y relativo. Redondeo y truncamiento. Propagación de errores. Sistemas de punto fijo y punto flotante. Errores de representación. Propagación de errores. Estrategias para evitar cancelación de dígitos significativos.

2. Solución de ecuaciones no lineales

Métodos de Bisección, Newton, Secante y de punto fijo. Resultados de convergencia y algoritmos.

3. Interpolación numérica

Interpolación polinomial. Teorema de existencia y unicidad del polinomio interpolante. Formas de Lagrange y de Newton. Diferencias divididas. Análisis de error del polinomio interpolante. Splines lineales y cúbicos.

4. Aproximación de funciones

Teoría de cuadrados mínimos. Caso discreto y caso continuo. Ecuaciones normales. Polinomios ortogonales.

5. Integración numérica

Reglas simples y compuestas: rectángulo, punto medio, trapecio y Simpson. Reglas Gaussianas.

6. Solución de sistemas de ecuaciones lineales

Eliminación Gaussiana y factorización LU. Algoritmos. Conteo operacional. Métodos

iterativos: Jacobi y Gauss-Seidel.

7. **Introducción a la Programación Lineal**

Convexidad y desigualdades lineales. Programación lineal. Introducción al método Simplex.

BIBLIOGRAFÍA BÁSICA

- D. Kincaid, W. Cheney. Numerical Analysis. Mathematics of scientific computing. 3rd. edition. AMS, 2002.
- R. Burden, J. Faires. Análisis Numérico. Thomson Learning, 2002.

BIBLIOGRAFÍA COMPLEMENTARIA

- L. Eldén, L. Wittmeyer-Koch, Numerical Analysis: an introduction. Academic Press, 1990.
- I. Griva, S. Nash, A. Sofer. Linear and nonlinear optimization. SIAM, 2009.

METODOLOGÍA DE ENSEÑANZA

Se dictan semanalmente: 2 clases teóricas de 2 horas cada una y 2 clases prácticas de 2 horas cada una. Las clases teóricas cubren los principales resultados del programa, haciendo énfasis tanto en las ideas subyacentes como en sus demostraciones rigurosas. Su dinámica es principalmente expositiva, aunque incorpora la participación de los/as estudiantes mediante la inferencia y anticipación de resultados. Las clases prácticas están enfocadas en la resolución de problemas cuya herramienta principal proviene de los contenidos teóricos. En ellas se promueve el trabajo colaborativo entre estudiantes y docentes, complementado con la gestión de resoluciones de problemas particularmente relevantes en el pizarrón a cargo del/ de la profesora.
Paralelamente, la materia cuenta con un Aula Virtual en el entorno Moodle en el que se fomenta la participación a través de foros y se incluye material de estudio: guías de trabajos prácticos, bibliografía, enlaces a sitios relevantes, entre otros recursos.

FORMAS DE EVALUACIÓN

Se tomarán 2 (dos) parciales y sus correspondientes instancias de recuperación.
Se tomarán 2 (dos) trabajos prácticos de laboratorio y sus correspondientes instancias de recuperación.
- Examen final escrito

REGULARIDAD

- Aprobar los 2 (dos) parciales, o uno de ellos y el recuperatorio del otro.
- Aprobar los 2 (dos) trabajos de laboratorio, o uno de ellos y el recuperatorio del otro

PROMOCIÓN

- No se prevé régimen de promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Algoritmos y Estructuras de Datos II	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Se pretende que el/la estudiante adquiera: capacidad para comprender y describir el problema que resuelve un algoritmo (el “qué”) y diferenciarlo de la manera en que lo resuelve (el “cómo”); capacidad para analizar algoritmos, compararlos según su eficiencia en tiempo y en espacio; capacidad y hábito de identificar abstracciones relevantes al abordar un problema computacional; familiaridad con técnicas de diseño de algoritmos de uso frecuente; familiaridad con la programación (en el lenguaje c, entre otros) de algoritmos y estructuras de datos, familiaridad con la utilización de diversos niveles de abstracción y lenguajes de programación.

CONTENIDO

1. Análisis de Algoritmos

Motivación

Problema de Ordenación. Diferentes maneras de ordenar. Ordenación por selección. El ciclo for. Conteo de operaciones de un programa. Esquemas de conteo. Conteo de comparaciones de la ordenación por selección. Incidencia del crecimiento del tamaño de los datos en la performance del algoritmo. Introducción del término “del orden de”. Ordenación por inserción. Conteo. Peor caso, mejor caso y caso medio.

La notación O

Significado de peor caso y caso medio. Operaciones elementales. Análisis aproximado. La notación O. Ejemplos. Insignificancia de las constantes aditivas y multiplicativas. Reflexividad y transitividad. Igualdad entre los O's de funciones. Equivalencia entre logaritmos de diferente base. Regla del límite. Jerarquía: logaritmos, polinomios, exponenciales, factoriales. El O de la suma y el producto. El O de un polinomio. Terminología: funciones y algoritmos logarítmicos, cuadráticos, cúbicos, polinomiales, exponenciales. Balance entre tiempo y espacio de los algoritmos.

Ejemplos

Búsqueda lineal. Análisis de mejor caso, peor caso y caso medio. Búsqueda lineal en un arreglo ordenado. Análisis de mejor caso, peor caso y caso medio. Búsqueda binaria. Análisis de mejor caso, peor caso y caso medio. Contraste entre el algoritmo lineal y el logarítmico cuando el tamaño de la entrada crece.

Motivación de la recurrencias

Transformación gradual de la ordenación por selección en la ordenación por intercalación. Versión funcional de la ordenación por intercalación. Versión imperativa. Análisis de la ordenación por intercalación. Resolución de la recurrencia.

Recurrencias
Recurrencias divide y vencerás. Formulación y resolución. Ejemplos. Demostración de la resolución de recurrencias divide y vencerás.

2. Estructuras de Datos

Introducción
Importancia de la elección de estructuras de datos adecuadas. Los tipos concretos como concepto relativo a un lenguaje de programación. Los tipos abstractos como concepto asociado a un problema que se quiere resolver. Tipos abstractos y sus diferentes representaciones.

Estructuras concretas
Estructuras concretas más comunes en los lenguajes de programación. Arreglos. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Eficiencia de las operaciones. Diferentes tipos de índices. Tipos enumerados. Ciclo for generalizado. Listas como tipos concretos. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Eficiencia de las operaciones. Registros. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Problema de aliasing.

Tipos abstractos de datos (TAD's)
Tipos abstractos más usuales. Tipos abstractos como concepto que surge de un problema a resolver. Chequeo de paréntesis balanceados: TAD Contador, operaciones, ecuaciones. Chequeo de delimitadores balanceados: TAD Pila, operaciones, ecuaciones. Representaciones posibles de contadores. Ejemplo: versión iterativa de la ordenación por intercalación usando una pila. Ejemplo: evaluación de expresiones en notación polaca inversa usando una pila. TAD Lista. Operaciones. Ecuaciones. Representaciones usando arreglos. Representaciones de pilas usando arreglos y listas. Transmisión de datos: TAD cola, operaciones, ecuaciones. Representaciones usando arreglos y listas. Listas enlazadas. Representación gráfica. Representaciones de listas, pilas y colas usando listas enlazadas, listas enlazadas con puntero al último y listas circulares. Aliasing y errores usuales al programar con punteros. Manejo de memoria en ejecución. Diccionarios: TAD árbol binario. Representación gráfica. Operaciones. Ecuaciones. Terminología botánica y genealógica. Posiciones. Subárbol correspondiente a una posición. Posiciones de un árbol. Elemento alojado en una posición de un árbol. Representación usando punteros. Árboles binario de búsqueda (ABB). Operaciones: versiones recursiva e iterativa. Eficiencia. TAD cola de prioridades. Operaciones. Ecuaciones. Heap. Implementación de cola de prioridades usando un heap. Eficiencia de las operaciones. Heap usando arreglos. Eficiencia. Ordenación con heap. Eficiencia. Ordenación con heap sin arreglo auxiliar.

Otras estructuras
Problema unión-find. Inicialización virtual.

3. Estrategias conocidas de resolución de problemas

Uso de heurísticas en algoritmos. Estrategias de diseño de algoritmos.

Algoritmos voraces
Propiedades generales de los algoritmos voraces (o greedy o glotones o golosos). Esquema general. Problema de la moneda simplificado. Problema de la mochila simplificado. Problema del camino de costo mínimo. Algoritmo de Dijkstra. Problema del árbol generador de costo mínimo. Algoritmos de Prim y de Kruskal.

Divide y vencerás
Propiedades generales de la técnica divide y vencerás. Esquema general. Búsqueda binaria. Ordenación por intercalación. Ordenación rápida (quicksort). Cálculo eficiente de la potencia n-ésima de un número. Multiplicación de grandes números.

Backtracking
Motivación: algoritmo para salir de un laberinto. Problema de la moneda. Problema de la mochila. Problema de los caminos de costo mínimo.

Programación dinámica
Funciones recursivas potencialmente exponenciales. Confección de tablas. Fibonacci. Problema de la moneda. Problema de la mochila. Funciones con memoria. Revisión de los problemas de la moneda y de la mochila. Problema de los caminos de costo mínimo. Algoritmo de Floyd. Cómputo de números combinatorios. Reducción del espacio necesario para las tablas.

Recorrida de grafos y más backtracking
Recorrida de árboles binarios. Pre-orden, in-orden y pos-orden de izquierda a derecha y de derecha a izquierda. In-orden para listar ordenadamente un ABB. Recorrida de árboles finitarios. Precondicionamiento. Pre-orden y pos-orden para resolver el problema del ancestro. Recorrida de árboles dirigidos o no. DFS recursivo e iterativo con pila. BFS con cola. Grafos implícitos. Problema de las ocho reinas. Podas graduales al grafo de búsqueda

BIBLIOGRAFÍA BÁSICA

- Fundamentos de Algoritmia, Gilles Brassard, Paul Bratley. Prentice-Hall, 1997.
- Fundamentals of Algorithmics. Gilles Brassard, Paul Bratley. Prentice-Hall, 1995.
- Introduction to Algorithms. Thomas Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Cambridge, 2009.
- Introduction to Algorithms: A Creative Approach. Udi Manber. Addison-Wesley, 1989.

BIBLIOGRAFÍA COMPLEMENTARIA

- Programación Metódica. José Luis Balcázar. McGraw-Hill, 1993.
- Matemática Discreta. Norman L. Biggs. Vives V., 1998
- Cálculo de Programas. Javier Blanco, Silvina Smith, Damián Barsotti. Universidad Nacional de Córdoba, 2008.
- Programming: the Derivation of Algorithms. Anne Kaldewaij. Prentice-Hall, 1990.

EX-2025-01045526- -UNC-ME#FAMAF

METODOLOGÍA DE ENSEÑANZA

Los contenidos del programa se organizan en tres partes: análisis de algoritmos, estructuras de datos y estrategias de diseño algorítmico. Cada una de esas partes presenta coherencia temática y brinda herramientas necesarias para abordar las siguientes.
Para el desarrollo de la asignatura se combinan clases teóricas, prácticas y de laboratorio. En las clases teóricas se formulan preguntas y/o problemas disparadores, estimulando la participación a través de preguntas, comprensión del problema y propuestas de solución. Las clases prácticas se desarrollan en aulas comunes, donde estudiantes de manera individual o en grupos resuelven una guía de ejercicios y problemas, realizando consultas a sus docentes cuando lo requieren. Ejercicios y problemas seleccionados se discuten y resuelven de manera colectiva. Las clases de laboratorio se desarrollan en aulas con computadoras, agrupando a los/as estudiantes en diferentes comisiones. Se resuelven en la computadora problemas modélicos que ya se han estudiado en clases teóricas y prácticas. También se proponen desarrollos más complejos que dan lugar a proyectos de varias semanas de duración realizados de manera grupal y colaborativa bajo la orientación de sus docentes. Los programas deben implementarse en el lenguaje de programación C. En las clases de laboratorios se dictan contenidos necesarios sobre el lenguaje de programación. Las aulas de laboratorio cuentan con suficiente número de computadoras, con todas las herramientas necesarias bajo el sistema operativo Linux Ubuntu). Estudiantes que deseen traer su propia notebook pueden hacerlo y realizar los desarrollos en el lenguaje C en cualquier plataforma (Linux, Windows o MacOS).
Entre las tareas que se propone a los/as estudiantes podemos mencionar: lectura del material empleado en las clases teóricas, resolución en grupo o individual de los ejercicios y problemas de las guías, implementación en computadoras de soluciones a los problemas planteados en el laboratorio sujeto a las pautas formuladas en los enunciados y con el acompañamiento y orientación docente. Se promueve la participación en foros del Aula Virtual, la interacción con docentes y compañeros/as.
Además del material bibliográfico y las guías de ejercicios y problemas diseñadas para la materia, se brinda acceso a los apuntes del/ de la docente de teóricos y a las presentaciones multimedia empleadas en el dictado. Se ponen a disposición de estudiantes simuladores de acceso libre existentes en Internet que permiten visualizar el funcionamiento de los algoritmos y comprender de manera visual las estructuras de datos y los algoritmos que se estudian. También se brinda acceso a manuales del lenguaje de programación y del sistema operativo. Todo el material es accesible desde el aula virtual de la materia, que cuenta asimismo con la posibilidad de la interacción con docentes y pares a través de los foros.

FORMAS DE EVALUACIÓN

Se toman dos evaluaciones parciales escritas evaluando los contenidos teóricos y prácticos, a través de ejercicios similares a los de las guías de práctico. Se evalúan dos trabajos prácticos de laboratorio mediante presentaciones breves de lo realizado.
Se puede recuperar un examen parcial teórico/práctico y un examen parcial de laboratorio.

El examen final para regulares consiste de una evaluación escrita teórico-práctica.
El examen final para libres consiste de una evaluación escrita teórico-práctica. y una evaluación de laboratorio, es necesario aprobar ambas.

REGULARIDAD

- Aprobar las dos evaluaciones parciales de laboratorio, o sus correspondientes recuperatorios (podrá recuperar una sola instancia).

PROMOCIÓN

- Aprobar las dos evaluaciones parciales de laboratorio, o sus correspondientes recuperatorios (podrá recuperar una sola instancia).
- Aprobar las dos evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Organización del Computador	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Que el/la estudiante sea capaz de reconocer las unidades constitutivas básicas de un sistema de computación, comprender su funcionamiento interno y la interacción entre ellas.

CONTENIDO

1. Circuitos Lógicos Combinacionales

1.1-Sistemas binarios de numeración.
1.2-Representación de números negativos.
1.3-Puntos fijo y flotante.
1.4- Errores en la representación de los datos a nivel máquina.
1.5-Funciones lógicas. Postulados del álgebra de conmutación (Boole). Minimización.
1.6-Circuitos lógicos de bajo y medio nivel de integración.
1.7 Nociones de Lenguajes de Descripción de Hardware

2. Circuitos Lógicos Secuenciales

2.1-Celda básica de memoria (“Flip-Flop D”).
2.2-Circuitos lógicos secuenciales sincrónicos.
2.3-Autómatas de Moore y Mealy.
2.4-Introducción a los circuitos lógicos secuenciales programables.
2.5- “Latchs” y “Shift Registers”

3. Procesadores

3.1-Líneas de direccionamiento, datos y control.
3.2-Registros internos.
3.3-Modos de direccionamientos.
3.4-Instrucciones (Incluye conceptos sobre lenguaje ensamblador (“assembly”)).
3.5-Interrupciones.
3.6 Procesores Tipo Von Newman
3.7 Procesadores Tipo Harward

4. Memorias

4.1- Conceptos fundamentales sobre memorias “Read Only Memory” - ROM, “Programmable Read Only Memory” - PROM, “Erasable Programmable Only Memory” - EPROM y “Electricaly Erasable Programmable Read Only Memory” -

EEPROM (Introducción a los “Programmable Logic Devices” - PLD). Memoria “FLASH”.
4.2-Conceptos fundamentales sobre memorias “Random Access Memory” - RAM estáticas (SRAM) y dinámicas (DRAM).
4.3-Estructuración o decodificado de bancos de memorias (“Memory Mapped”).
4.4- Otros tipos de Memorias. Ancho de banda. Jerarquía de memorias. Componentes principales de la jerarquía. Organización funcional.
4.5-Sistemas de detección de errores en datos almacenados en memoria

5. Puertos de Entrada/Salida

5.1-Puerto paralelo. Su estructuración y utilización.
5.2- Puerto serie. Su estructuración y utilización.

Trabajos Prácticos o Laboratorios

Laboratorio 1: propuesta formativa de reconocimiento y reparación de computadoras. Se propone el desarmado y armado de una computadora, reconocimiento de sus partes y chips. Se incentiva a comprender que computadoras consideradas obsoletas son en realidad arquitecturas avanzadas con todos los elementos que verán en la materia. Se espera que las y los alumnos logren a través de la materialidad que implica la manipulación de una computadora, comprender que los circuitos combinacionales y secuenciales forman una computadora. Para la realización de este trabajo los alumnos disponen de laboratorios experimentales en la Facultad con comodidades suficientes para la tarea encomendada. El tiempo presencial dedicado por los alumnos para este laboratorio es de 4 horas. En caso de necesidad se asignan distintos turnos para que todos los alumnos puedan realizar el laboratorio de manera cómoda y adecuada.

Laboratorio 2: propuesta formativa de programación en ensablador ARM64 para la creación de un demostración gráfica en un entorno emulado. Se propone una consigna sencilla y abierta, para lograr un actividad con "piso bajo, techo alto y paredes anchas", donde a creatividad sea la fuerza principal de tracción hacia otros contenidos más profundos como la necesidad de modularización, auto-programación o compilación y de optimización para un buen desempeño. Por ejemplo, se solicita a los alumnos que desarrollen una versión modificada de un juego de computadora, pero escribiendo el código en lenguaje ensamblador. Para la realización de este trabajo los alumnos disponen de laboratorios de computación en la Facultad con comodidades y computadoras suficientes para la tarea encomendada. El tiempo presencial dedicado por los alumnos para este laboratorio es de 8 horas. En caso de necesidad se asignan distintos turnos para que todos los alumnos puedan realizar el laboratorio de manera cómoda y adecuada.

BIBLIOGRAFÍA BÁSICA

- Patterson, David y Hennesy, John. Computer Organization and Design. The Hardware and Software Interface. ARM Edition. Editorial Morgan Kaufman 2017.
- Morris Mano, M.: “Diseño Digital - Tercera Edición”. Pearson 2003.

BIBLIOGRAFÍA COMPLEMENTARIA

- Patterson, David y Hennesy, John. Estructura y diseño de computadores. La interfaz hardware/software. Editorial Reverté. 4ta. Edición. Año 2011.
- Stallings, William. Organización y arquitectura de computadores..Prentice Hall, 2007.
- Morris Mano, M.: “Ingeniería Computacional, diseño del hardware”. Prentice Hall Hispanoamericana S.A., 1992.
- Tanenbaum, A. S.: “Organización de Computadoras, un enfoque estructurado”. Prentice Hall Hispanoamericana S. A., 2000.está y Paulo Tirao. Álgebra. Una introducción a la Aritmética y la Combinatoria.

METODOLOGÍA DE ENSEÑANZA

La secuencia de contenidos arranca con la representación de los principales tipos de números conocidos por los estudiantes (naturales, enteros, reales) por medio de formatos digitales estándares utilizados normalmente en computación. Se aprovecha este punto para introducir conceptos tales como errores relativos y absolutos en la representación de dichos números. Como siguiente paso natural en la secuencia se analiza la forma de implementar operaciones básicas sobre este tipo de números tales como la suma y la resta. Para ello se presentan las nociones de circuitos básicos combinacionales y compuertas básicas haciendo un repaso del álgebra de Boole. Se aprovecha para presentar a los sistemas combinacionales de media escala de integración tales como multiplexores y decodificadores que son útiles como módulos constructivos básicos para los sistemas estudiados. También se van presentando nociones básicas de lenguajes de descripción de hardware a manera de refuerzo o vía alternativa para entender a los sistemas estudiados. Continuando con la secuencia, se introducen los sistemas secuenciales para solucionar necesidades concretas de la computación tales como multiplicación y división de números de punto flotante. Se aprovecha para presentar módulos constructivos típicos tales como flip-flops, latchs, shift registers, etc., y se comienza a hablar de nociones de memorias tipo ROM y RAM aunque sin profundizar todavía en este punto. El siguiente paso es comenzar a describir la organización básica y elemental de un procesador muy simplificado mencionando las principales diferencias entre arquitecturas tipo Von-Newman y tipo Harvard. Se presentan conceptos y ejemplos de programación en lenguaje ensamblador. A continuación, se describen en más detalles los sistemas de memorias incluyendo conceptos de mapeo básico y nociones básicas de jerarquías de memorias. Finalmente, para completar la secuencia se presentan los sistemas de entrada y salida básicos (paralelo y serie) junto con los conceptos básicos asociados a sistemas de interrupciones.
En cuanto al formato utilizado para el dictado de clases teóricas, se recurre principalmente a exposiciones utilizando todos los recursos audiovisuales disponibles tales como proyectores, pizarrón, etc., pero siempre fomentando la activa participación e interacción con el/la estudiante. Asociados a las clases teóricas se dan clases prácticas donde se trabaja en la resolución de problemas, siguiendo lineamientos similares pero con mayor interacción e iniciativa del estudiantado, abriendo el espacio para que planteen sus dudas y consultas. Se pone a disposición de los/as estudiantes un Aula Virtual con contenidos, foros e incluso con links a videos, guías de problemas y guías de laboratorios. Los/as estudiantes pueden consultar tanto presencialmente en los horarios de clases y consultas, como así también mediante los foros en el Aula Virtual. Para el desarrollo de los

laboratorios la Facultad dispone de Laboratorios de computación especiales dotados con suficientes computadoras y además de laboratorios experimentales donde los alumnos pueden realizar tareas manuales como por ejemplo desarmar y reparar computadoras (Ver Laboratorio 1). Para el Laboratorio 2 se utilizan simuladores apropiados para lenguaje ensamblador que permiten hacer compilaciones y ejecuciones cruzadas simulando el entorno del procesador estudiado en cualquier computadora, ya sean las disponibles en la Facultad o bien los dispositivos personales de los/as alumnos/as. Para el desarrollo del laboratorio 01 los alumnos emplean unas 4 horas presenciales en los laboratorios experimentales. Para el desarrollo del Laboratorio 02, los alumnos utilizan 8 horas presenciales en los laboratorios de computación. En todos los casos hay suficientes recursos y profesores/as para atender a los/as alumnos/as. En cuanto a la dinámica de interacción con los/as estudiantes, además del/de la profesor/a encargado/a del teórico existe un número suficiente y adecuado de docentes para el desarrollo de las clases de problemas y laboratorios.

FORMAS DE EVALUACIÓN

Los/as estudiantes serán evaluados en instancias de evaluación formativas e instancias sumativas.
Instancias de evaluación formativas: Se refiere a ocho trabajos prácticos o laboratorios donde los/as estudiantes resolverán dos proyectos asociados a la materia y se les tomará una exposición oral tipo coloquio donde además de evaluar se aprovechará para diagnosticar el estado de aprendizaje del/de la estudiante e identificar necesidades de ayuda pedagógica apropiados para el/la estudiante. Cada trabajo práctico tendrá dos niveles de complejidad. Uno para regularizar y uno para promocionar.
Respecto de instancias de evaluación sumativas: Serán dos parciales tomados en forma presencial.
Habrá dos parciales cada uno con su propio recuperatorio. La nota del recuperatorio reemplaza la nota del parcial recuperado.
- Examen final
Los/as estudiantes libres o regulares rendirán un examen final escrito similar a los parciales y además deberán tener presentados o presentar los trabajos prácticos del año en curso. Se les podrá tomar un examen oral de los mismos y de los temas del examen escrito, en función de los antecedentes registrados de la actuación previa del/de la estudiante en la materia.

REGULARIDAD

- Aprobar al menos dos evaluaciones parciales o sus correspondientes recuperatorios con nota mayor o igual a cuatro. (Se toman 2 parciales y 2 recuperatorios, uno para cada parcial) 2.
- Aprobar al menos el 60% de los trabajos prácticos o de laboratorio (Se desarrollan al menos cinco trabajos prácticos o laboratorios).

Los trabajos prácticos o de laboratorio podrán ser tenidos en cuenta para la nota final de un parcial.

EX-2025-01045526- -UNC-ME#FAMAF

PROMOCIÓN

- Aprobar las dos evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete). Habrá dos parciales y dos recuperatorios. Cada parcial tiene su propio recuperatorio. La nota del recuperatorio reemplaza la nota recuperada.
- Aprobar todos los trabajos prácticos o de laboratorio con su correspondiente defensa oral. Los trabajos prácticos o de laboratorio podrán ser tenidos en cuenta para la nota final de promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Introducción a la Lógica y la Computación	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Se han definido para esta materia tres grandes ejes de contenidos teóricos que contribuirán a lograr los objetivos propuestos.
El primer eje trata de estructuras ordenadas, que constituyen la base para la definición de modelos matemáticos, tanto de los lenguajes de programación como de las lógicas que se utilizan para razonar sobre los programas.
El segundo eje aborda la lógica proposicional a través de una presentación diferente a la ofrecida en materias anteriores, que no pone énfasis en el cálculo, sino en el concepto de demostración. Este abordaje establece las bases para conectar la lógica con otras áreas fundamentales de las Ciencias de la Computación, como el cálculo lambda (a través del isomorfismo de Curry-Howard), y la inteligencia artificial.
Por último, el tercer eje trata sobre mecanismos de computación y formas de definición de lenguajes formales, con aplicaciones directas en el desarrollo de los lenguajes de programación, por ejemplo mediante las técnicas de parsing.

OBJETIVOS
En esta materia se abordan contenidos que constituyen algunos de los pilares teóricos de las Ciencias de la Computación. El objetivo general es proveer un marco teórico que tenga aplicaciones tanto en la práctica profesional como en la investigación científica.
Entre los objetivos específicos se espera que las y los estudiantes adquieran destrezas relativas a:
1) La aplicación de los diversos algoritmos que involucran estructuras matemáticas surgidas de las teorías del orden, de los autómatas y lenguajes formales y de la lógica.
2) Manejo de los conceptos de inducción y recursión estructural.
3) Desarrollo de demostraciones matemáticas y formales involucrando los conceptos de la materia.

CONTENIDO

1. **Relaciones y orden**

Noción de Relación y propiedades. Relaciones de Equivalencia y Particiones. Órdenes Parciales. Conjuntos Parcialmente Ordenados (“posets”). Máximos, mínimos, elementos maximales y minimales, ínfimos y supremos. Diagramas de Hasse. Isomorfismo de posets y sus propiedades.

2. **Reticulados y Álgebras de Boole**

Posets reticulados. Versión algebraica: retículos. Equivalencia de dichas definiciones. Isomorfismo de retículos. Equivalencia con isomorfismo de posets. Pruebas de desigualdades en reticulados. Reticulados acotados y complementos. Reticulados distributivos. Álgebras de Boole y sus propiedades. Teoremas de Representación. Representación de las álgebras de Boole finitas como álgebras de conjuntos. Teorema de Birkhoff de Representación de reticulados distributivos finitos. Caracterizaciones de la distributividad en reticulados.

3. Cálculo Proposicional: Sintaxis y Semántica

La sintaxis de las proposiciones (PROP). Definición inductiva de PROP como un conjunto de cadenas. La inducción estructural; recursión sobre PROP. Noción de verdad: Asignaciones y valuaciones. Tautologías y la relación de consecuencia. Lema de coincidencia y tablas de verdad.

4. Cálculo Proposicional: Deducción Natural

Noción de demostración: el sistema de deducción natural de Gentzen-Prawitz. Caso intuicionista y clásico: la reducción al absurdo. Inducción estructural en derivaciones. Conjuntos consistentes, maximales. Teoremas de Corrección y Completitud del cálculo proposicional. Álgebra de Lindenbaum.

5. Autómatas Finitos y Lenguajes Regulares

Alfabetos, Cadenas y Lenguajes. Codificación de problemas con lenguajes. Autómatas finitos deterministas (DFA). Trazas. Lenguajes regulares como los aceptados por un DFA. Autómatas no deterministas (NFA) y con movimientos silenciosos (ε-NFA). Determinización de ε-NFAs. Expresiones regulares y propiedades de clausura de los lenguajes regulares. Teorema de Kleene. Lema de bombeo (“Pumping”) como método para ver no regularidad.

6. Gramáticas

Gramáticas Libre de Contextos (CFG). Derivación. Lenguajes Libres de Contexto. Gramáticas Regulares; equivalencia con lenguajes regulares. Ejemplo de autómata a pila. Lenguajes contextuales (CSL). Introducción a la computabilidad y la Jerarquía de Chomsky.

BIBLIOGRAFÍA BÁSICA

- Apunte de Cátedra: “Lenguajes y Autómatas”, Alejandro Tiraboschi y colaboradores, 2009.
- Apunte de Cátedra: “Lógica Proposicional”, Pedro Sánchez Terraf, 2004. Edición 2022.
- Apunte de Cátedra: “Reticulados y Álgebras de Boole”, Alejandro Tiraboschi y Héctor Gramaglia, 2020.

BIBLIOGRAFÍA COMPLEMENTARIA

- B. Davey, H. Priestley, "Introduction to Lattices and Order", Cambridge University Press, 1997.
- Jeffrey Ullman; John Hopcroft; Rajeev Motwani. "Introducción a la teoría de autómatas, lenguajes y computación". Pearsons, 2008.

- D. Van Dalen, "Logic and Structure". Springer, 1997.

METODOLOGÍA DE ENSEÑANZA

Los contenidos del programa se presentan agrupando en tres partes, las unidades 1 y 2, 3 y 4, y finalmente 5 y 6. Cada una de esas partes presenta una especial coherencia temática.
Para el desarrollo de la asignatura, optamos por un esquema de clases teóricas y clases prácticas. En las clases teóricas el/la docente responsable de la materia presenta los contenidos del programa, ejemplifica su uso para la resolución de problemas y responde preguntas que surgen durante su exposición en clase o, con posterioridad, a través del Aula Virtual. Las clases prácticas se centran en la actividad del estudiante y sus producciones para la consolidación de los contenidos desarrollados en la clase teórica.
Se requiere que los/as estudiantes anticipen el abordaje del material teórico con antelación al desarrollo de cada clase mediante indicación de lecturas pautadas por el/la docente. Luego, en las clases prácticas, se espera que los/as estudiantes interioricen los contenidos desarrollados mediante la discusión y resolución de guías de ejercicios y problemas diseñadas para cada unidad del programa.
A través del Aula Virtual de la materia, se proveen presentaciones multimedia y apuntes teóricos de los contenidos de la materia, guías de trabajos prácticos, grabaciones de clases y uso de diferentes recursos disponibles en el aula virtual: foros, encuestas, cuestionarios, etc.
En lo que respecta a las dinámicas de trabajo propuestas a los/as estudiantes, se incentiva la participación en las clases teóricas, invitando al estudiantado a realizar preguntas de los contenidos desarrollados en clase y a través de un foro de consultas en el Aula Virtual creado para tal fin. La gestión de las clases prácticas promueve que los/as estudiantes discutan y resuelvan en interacción con sus pares y docentes las actividades propuestas. El/la docente invita al estudiantado a presentar sus producciones, avances y dificultades, consultando a los/as estudiantes por su progreso. Se resuelven y discuten con toda la clase problemas que tienen una importancia central y modélica para el desarrollo de la materia. Durante las clases prácticas los/as estudiantes pueden hacer preguntas al equipo docente, y consultas fuera de horario a través del Aula Virtual.

FORMAS DE EVALUACIÓN

Se tomarán 3 (tres) exámenes parciales, pudiendo recuperarse uno de los dos primeros. Las evaluaciones parciales serán sobre contenidos teórico-prácticos.
El examen final de la materia será escrito.

REGULARIDAD

- Aprobar las dos primeras evaluaciones parciales, o una de ellas y el recuperatorio de la otra.

PROMOCIÓN

- Cumplir un mínimo de 80% de asistencia a clases teóricas, prácticas, o de laboratorio.

- Aprobar las tres evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Probabilidad y Estadística	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El propósito del curso es proporcionar una base sólida, a nivel universitario, en teoría de probabilidad y estadística, destacando su importancia en la resolución de problemas de diversas disciplinas.
Se espera que el alumno, al finalizar el curso, esté preparado para calcular probabilidades en diferentes situaciones (Probabilidad), describir el comportamiento del conjunto de datos (Estadística descriptiva) y tomar decisiones sobre posibles hipótesis planteadas (Inferencia estadística) para la población en estudio. En este curso se impartirán las herramientas básicas de la inferencia estadística. Se espera que el alumno sea crítico al interpretar resultados estadísticos de documentos publicados.
También pretendemos que los alumnos puedan interpretar los resultados del software estadístico Python para las situaciones consideradas en el curso.

CONTENIDO

1. **Probabilidad.**

Modelos matemáticos: determinísticos y aleatorios. Elementos de un modelo aleatorio o probabilístico: espacio muestral, familia de eventos, función de probabilidad. Propiedades. Probabilidad de unión de eventos. Espacios finitos equiprobables. Probabilidad condicional. Propiedades. Fórmula multiplicativa, fórmula de la probabilidad total, teorema de Bayes. Independencia de eventos. Esquema de extracción sin reposición.

2. **Variables aleatorias discretas.**

Variable aleatoria (v.a.): definición. Variable aleatoria discreta. Distribución de probabilidad o función de probabilidad de masa. Función de distribución acumulada de una variable aleatoria. Propiedades. Esperanza, valor esperado o media de una variable aleatoria discreta. Valor esperado de las funciones de una variable aleatoria discreta. Varianza y desviación estándar. Propiedades de varianza. Ejemplos de v.a. Discretas: distribución de probabilidad binomial, media y varianza. Distribución de Poisson. Aproximación binomial a la distribución de Poisson. Media y varianza de la distribución de Poisson. Distribución hipergeométrica. Esperanza y varianza de la distribución hipergeométrica. Aproximación binomial a la hipergeométrica. Distribución binomial negativa. Esperanza y varianza.

3. **Variables aleatorias continuas.**

Definición de variable aleatoria continua. Función de densidad de probabilidad. Función de distribución acumulada. Percentil de una v.a. con densidad f. Valor esperado o valor medio de una v.a. continua. Valor esperado de funciones de v. a. discretas. Varianza y desviación estándar.
Ejemplos de distribuciones de v.a. continuas. Distribución uniforme y normal. Media y varianza. Distribución normal estándar. Uso de tablas normales. Cálculo de percentiles de una distribución normal en términos de la distribución normal estándar. Distribución Gamma. Casos particulares: distribución exponencial y distribución chi-cuadrado. Distribución lognormal. Distribución de Weibull. Media y varianza de todas las variables mencionadas.

4. **Distribución de probabilidad conjunta.**

Distribución de probabilidad conjunta. Función de probabilidad de la masa conjunta de dos variables aleatorias discretas. Caso continuo: función de densidad de probabilidad conjunta. Funciones marginales de la densidad de probabilidad. Variables aleatorias independientes. Caracterización en términos de la factorización de la función de densidad de probabilidad conjunta o de la función de probabilidad de masa conjunta. Cálculo de la esperanza usando la distribución de probabilidad conjunta. Covarianza. Coeficiente de correlación. Propiedades.
Teorema central del límite. Ley de los grandes números y desigualdad de Chebyshev.

5. **Distribución de muestreo y estimación puntual estadística.**

Muestra aleatoria. Media muestral. Distribución en el caso normal. Enunciado del Teorema Central del Límite. Ejemplos. Aproximación normal a la binomial. Esperanza, varianza y covarianza de combinaciones lineales de variables aleatorias. Caso de muestra aleatoria de una distribución normal. Estimación puntual. Parámetros de una población o de una distribución. Estimadores insesgados. Error estándar estimado. Métodos de estimación puntual: Método de los Momentos y Método de Máxima Verosimilitud (EMV). Propiedad de invarianza del EMV.

6. **Intervalos de confianza basados en una sola muestra.**

Intervalos de confianza. Nivel de confianza. Intervalo de confianza para la media de una distribución normal con varianza conocida. Longitud del intervalo de confianza. Intervalo de confianza con muestras grandes para la media poblacional y la proporción poblacional. Selección del tamaño muestral para alcanzar una longitud especificada. Intervalo de confianza para la media de una distribución normal con varianza desconocida. Distribución t de Student con n grados de libertad. Uso de las tablas de la distribución t de Student para el cálculo de probabilidades.
Intervalo de confianza para la diferencia de medias basado en datos apareados, cuyas diferencias tienen una distribución normal con varianza desconocida.
Uso de tablas de la distribución de chi cuadrado con v grados de libertad. Intervalo de confianza para la varianza de una distribución normal.

7. **Pruebas o tests de hipótesis. Pruebas o tests de hipótesis.**

Elementos de un test de hipótesis: hipótesis nula y alternativa, estadístico de prueba, región de rechazo, errores de tipo I y II, nivel y potencia del test. Tests unilaterales y bilaterales. Pruebas de media para una muestra aleatoria con distribución normal y varianza conocida. Función de potencia. Determinación del tamaño muestral para

obtener una potencia prefijada en una alternativa fija. Tests de nivel aproximado para muestras grandes. Tests para la media de una m.a. con distribución normal y varianza desconocida. Test de hipótesis para la diferencia de medias basado en datos apareados, cuyas diferencias siguen una distribución normal con varianza desconocida.
Tests de varianza para una m.a. con distribución normal. Tests de muestras grandes para proporción desconocida. P-valor. Relación entre los tests bilaterales y los intervalos de confianza.
Test de hipótesis para la diferencia de medias considerando muestras aleatorias de distribuciones normales independientes. Caso de varianza conocida y de varianza desconocida.

8. **Elementos de estadística descriptiva**.

Recopilación de datos. Organización de la información. Presentación y tabulación. Análisis de resultados.
Modelos matemáticos: determinísticos y aleatorios. Población y muestra. Estadística descriptiva de conjuntos de datos numéricos. Métodos gráficos y tabulares para resumir y describir. Histogramas. Distribución de frecuencia de la muestra. Formas cualitativas de histogramas. Medidas de posición: media muestral, mediana muestral y cuartiles. Medidas de variabilidad: desviación estándar, distancia intercuartílica. Box-plot. El coeficiente de variación.Gráficos multivariados para datos tabulares. Uso de Jupyter, Colab y Python para la descripción y visualización de datos tabulares.

BIBLIOGRAFÍA BÁSICA

- Notas de curso Elementos de Probabilidad y Estadística. Dra Ana Georgina Flesia. 2024
- Devore, Jay. Probabilidad y estadística para ingeniería y ciencias. Cengage Learning, novena edición (2015).

BIBLIOGRAFÍA COMPLEMENTARIA

- Hoel, Paul; Port, Sidney, and Stone, Charles. Introduction to Probability Theory. Houghton Mifflin College, Boston, 1971.
- Ross, Sheldon. Introducción a la Estadística. Editorial Revertè, 2007.
- Wackerly, Dennis; Mendenhall, William, and Scheaffer, Richard. Estadística Matemática con Aplicaciones. Cengage Learning, séptima edición (2010).
- Walpole; Myers y Myers. Probabilidad y estadística para ingeniería y ciencias. Pearson educación, novena edición (2012).

METODOLOGÍA DE ENSEÑANZA

El horario asignado a esta materia por clase se divide en una presentación magistral de dos horas del temario y dos horas de discusiones sobre resoluciones prácticas en grupos reducidos.
Los contenidos han sido descriptos en el orden en que serán presentados a lo largo de la asignatura durante las horas de clase magistral teórico-práctica. Se utilizan

presentaciones multimedia que incluyen definiciones, proposiciones y gráficos para ampliar la información de cada tema. Se usa el pizarrón de forma limitada, ya que no se obtiene una buena visualización de lo escrito desde todas las posiciones del aula. Se usa un micrófono para que la clase sea escuchada por todos los alumnos del aula.
Durante las horas de discusión general de resoluciones prácticas, sí se utiliza el pizarrón para una explicación más efectiva de las resoluciones de los ejercicios prácticos. El pizarrón se divide en tercios para que hasta tres profesores puedan tener discusiones con grupos reducidos de estudiantes al mismo tiempo, mientras que el resto de los/as docentes lo hace en los bancos, de forma individual.
El resumen del curso, presentado en forma de diapositivas durante las clases magistrales, está disponible en el Aula Virtual, así como el libro asociado y la colección de ejercicios y problemas prácticos a realizar durante el cuatrimestre.
Los/as estudiantes tienen acceso a un cronograma de temas por día de clase, que se actualiza constantemente y está disponible en el Aula Virtual. En el cronograma están las fechas de parciales, recuperatorios y entrega del trabajo especial computacional que evalúa la unidad 8, elementos de estadística descriptiva. El primer parcial comprende todas las unidades de probabilidad y el segundo, todas las unidades de estadística.
Para presentar los temas en los que se usan los programas computacionales, se graba una clase que queda a disposición de los/as estudiantes en el Aula Virtual, y estos/as se trasladan a los laboratorios de la Facultad para las consultas prácticas.

FORMAS DE EVALUACIÓN

Deberán entregar todos los trabajos prácticos derivados de la ejercitación general. Tendrán dos instancias de evaluación parcial en el aula.

REGULARIDAD

- Aprobar con 4 (cuatro) al menos dos evaluaciones parciales o sus correspondientes recuperatorios.
- Aprobar el 60% de los trabajos prácticos.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis) y obtener un promedio no menor a 7 (siete).
- Aprobar todos los trabajos prácticos.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Sistemas Operativos	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Fundamentación: El sistema operativo es un programa fundamental dentro de toda la pila de software y hardware que compone una computadora moderna. Esto es así no solamente porque aísla a los programas de usuario de los detalles del hardware subyacente, sino que además provee fuertes abstracciones que han perdurado a lo largo de las décadas: procesos, memoria (virtual) y sistema de archivos.
La necesidad de aprovechar mejor el hardware hizo que apareciera el concepto de multiprogramación, donde el no-determinismo de los programas secuenciales se presenta de manera concreta, además de presentar el Área de la Teoría y la Práctica de la Concurrencia.

Objetivos:
Teórico
- Comprender las abstracciones principales de un Sistema Operativo: procesos, memoria, sistema de archivos.
- Resolver problemas simples que se plantean en la práctica para estas abstracciones.
- Comprender, reparar y programar algoritmos concurrentes de baja complejidad.
- Entender la problemática de la seguridad en general, y para los Sistemas Operativos en particular.
- Resolver problemas sencillos que involucren algunos de los aspectos sobresalientes de la seguridad y la entrada/salida en sistemas operativos.
- Entender las relaciones de compromiso de los algoritmos y estructuras de datos internas del Sistema Operativo. Comprender cómo algunos cambios tecnológicos afectan fuertemente estas relaciones de compromiso.
- Comprender la relación entre algunas partes del diseño de la arquitectura del microprocesador con el Sistema Operativo.
- Poder asimilar los conceptos utilizando ejemplos concretos de Sistemas Operativos.

Laboratorio
- Avanzar en la práctica de la programación en general.
- Trabajar en grupo tanto en objetivos individuales como en objetivos grupales.
- Ser capaz de leer, modificar y comprobar código dentro de Sistemas Operativos completos y funcionales.
- Utilizar herramientas de apoyo para el desarrollo del software: editores, detectores de errores en código estático, debuggers, chequeadores de memoria, etc.

- Utilizar herramientas de desarrollo colaborativo de proyectos y las prácticas asociadas a desarrollos remotos.
- Generar independencia para la búsqueda de soluciones técnicas en el proceso de desarrollo y/o modificación de código.
- Realizar entregas de proyectos dentro de límites de tiempo prefijados.
- Programar abstracciones de dispositivos de bajo nivel a partir de la especificación dada por su hoja de datos.
- Realizar modificaciones a partes fundamentales del Sistema Operativo: procesos, memoria virtual y sistema de archivos.
- Comprender en general la problemática del desarrollo del software dentro del núcleo del Sistema Operativo.

El teórico de la materia será clase expositiva tradicional con mezcla de live-coding dependiendo de tener un beamer en la sala de dictado.

Los laboratorios serán presenciales.

CONTENIDO

1. Virtualización

Virtualización de CPU: procesos, API de procesos, ejecución directa limitada, planificación, planificación multinivel. Virtualización de RAM: espacio de direcciones, API de memoria, traducción de direcciones, segmentación, administración de memoria libre, introducción a la paginación, TLB, tablas de páginas avanzadas, archivo de intercambio.

2. Concurrencia

Concurrencia e hilos, API de hilos, locks, variables de condición, semáforos, bugs de concurrencia.

3. Persistencia

Dispositivos de Entrada/Salida, discos duros rotacionales, RAID, archivos y directorios, implementación de sistemas de archivos, sistemas de archivos rápidos, fsck y bitácora, sistemas de archivos con registro.

BIBLIOGRAFÍA BÁSICA

- Remzi Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Operating Systems: Three Easy Pieces, University of Wisconsin-Madison, November, 2023 (Version 1.10), https://pages.cs.wisc.edu/~remzi/OSTEP/#book-chapters
- Gunnar Wolf, Esteban Ruiz, Federico Bergero, Erwin Meza. Fundamentos de Sistemas Operativos, 2015.
- Andrew S. Tanenbaum, Herbert Bos, Sistemas Operativos Modernos, Quinta Edición. Prentice Hall, 2022.
- Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Operating System Concepts 10th Edition, Wiley, 2021.

BIBLIOGRAFÍA COMPLEMENTARIA

- Michael Kerrisk, The Linux Programming Interface, No Starch Press, 2010.

- Russ Cox , Frans Kaashoek , Robert Morris, xv6 a simple, Unix-like teaching operating system, MIT, 2012, https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf
- Raphael Finkel. An operating systems Vade Mecum, Segunda Edición. Prentice Hall, 1988.
- Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. Linux Device Drivers, Third Edition. O'Reilly, 2005.

METODOLOGÍA DE ENSEÑANZA

La secuenciación de contenidos se estructura alrededor del libro OSTEP de la materia: virtualización, concurrencia y persistencia. Los laboratorios siguen esta secuencia de manera exacta, aunque no siempre sincronizada con el teórico.
Los tiempos de las 3 partes se dividen en dos, hasta el primer parcial que ocurre en la semana del estudiante que incluye el 21 de septiembre entra virtualización y para el parcial final en la 3ra semana de noviembre concurrencia y virtualización. Este aparente desbalance sirve para empezar lento y luego ir acelerando a medida que las/os estudiantes se afianzan en la lógica de la materia, que es la primera que tienen donde se mezclan conocimientos fundamentales (concurrencia), con tecnologías y formas de su aplicación (virtualización y persistencia).

Las clases teóricas son expositivas con prácticas en papel para realizar y que se van resolviendo en un teórico-práctico que ocurre en 4 de las 8hs semanales disponibles. El laboratorio se estructura en base a un proyecto introductorio a la materia y de preparación (Lab0) y tres laboratorios uno por tema. En el laboratorio se presentan los proyectos y luego el cuerpo docente orienta el trabajo de los grupos, integrados por 4 estudiantes. Como parte de la gestión de las clases, se realizan presentaciones sobre manejo de herramientas que son necesarias para el laboratorio (frameworks de testing, sistemas de control de versiones, etc.)

Las actividades en los teóricos se basan en la resolución de problemas en papel y lápiz o sea las guías de práctico. Además, se van compartiendo de manera regular noticias absolutamente actuales sobre las diferentes temáticas que se van tocando, a fin que las/os alumnos/as entiendan que el conocimiento de Sistemas Operativos es actual y tiene implicancias sociales. Se promueve el uso del canal de Zulip #sistop-2025 a fin de discutir problemáticas, dudas, ejercicios y todo lo que compete a la materia.
En los laboratorios las actividades son la compleción de proyectos donde se presenta una pieza de software y cada grupo debe lograr extenderla a fin de cumplir con los objetivos del laboratorio. Cada laboratorio tiene casos de testing bien definidos. Los proyectos de laboratorio incluyen la realización de un video de 7 minutos donde el grupo debe exponer dificultades y aprendizajes del proceso. Es parte de los materiales entregados los commits en el repositorio de cada grupo con pautas claras acerca de cómo se deben realizar estos commits.

Los recursos didácticos del teórico se basan principalmente en material generado a lo largo de los años por estudiantes y docentes. En este sentido, la plataforma Zulip funciona como repositorio de todos estos materiales.
Para el laboratorio además de Zulip se cuenta con Bitbucket para la administración

de versiones y material seleccionado a lo largo de más de 15 años en formatos de guías, videos y guías de estudio que se acceden a través de la plataforma Moodle.

La dinámica de trabajo del teórico es clásica, de tipo clase magistral, con la búsqueda de interacción a partir de los conocimientos o preconceptos de las y los estudiantes. En los laboratorios se promueve el trabajo colaborativo y autónomo al interior de los grupos . El cuerpo docente interviene para atender las dudas e inquietudes que se presentan a medida que los grupos avanzan con sus producciones.

FORMAS DE EVALUACIÓN

- Dos parciales teórico-práctico presenciales de 2hs cada uno.
- Cuatro (4) proyectos grupales. Todos los proyectos permiten una re-entrega.
- Dos (2) parciales de laboratorio en plataforma Moodle, cada uno con recuperatorio.
- Régimen de promoción.
- Examen final teórico-práctico presencial de 4hs, más coloquio sobre los Laboratorios si el alumno/a está libre.

REGULARIDAD

- Para regularizar la materia se necesita aprobar todos los laboratorios con 6 o más.
- Se toman parciales de laboratorio individuales a través de la plataforma Moodle, donde tienen que obtener un 6 o más en ambos para lograr la regularidad.

PROMOCIÓN

- Para promocionar la materia se necesita:
- Aprobar los dos parciales de teórico con un promedio de 7 (cada parcial con más de 6). Aprobar todos los laboratorios y sus parciales con 6 o más. En caso de promocionar la calificación final será:

$$notaPromoción = min(10,\ 0.35 * p1 + 0.35 * p2 + 0.4 * lab)$$

donde *p1* y *p2* son las calificaciones del primer y segundo parcial respectivamente, y *lab* es la calificación del desempeño en los laboratorios.

Cualquier estudiante puede rendir libre la materia. En este caso deberá presentar 14 días antes de la fecha de examen todos los laboratorios completos. Primero se tomará un examen en máquina con ejercicios relacionados a los laboratorios, luego el/la estudiante deberá defender sus laboratorios en un coloquio a posteriori de que haya aprobado el examen en máquina. Luego de pasar estas dos instancias podrá rendir el examen teórico-práctico de la materia.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Paradigmas de Programación	**AÑO:** 2025
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El objetivo de la materia es conocer e instrumentalizar conceptos fundamentales de los lenguajes de programación, poder identificar y explicar la semántica de los programas en diferentes lenguajes, identificar causas de comportamientos inesperados, conocer las semejanzas y diferencias entre los diferentes lenguajes de programación y las decisiones de diseño subyacentes a los diferentes paradigmas de programación.

CONTENIDO

1. Introducción, Historia y Alcance

Introducción a la materia. Historia de los lenguajes de programación. Alcance de los lenguajes de programación.

2. Sintaxis y Semántica

Distinción entre sintaxis y semántica.
Estructura y función de los compiladores.
Niveles de los compiladores.
Semántica denotacional, lambda cálculo y semántica operacional.
Fundamentos de semántica operacional.

3. Tipos

Concepto de tipo y subtipo.
Jerarquías de tipos.
Mecanismos de inferencia de tipos.
Tipado fuerte y tipados débiles.
Sobrecarga y polimorfismo.

4. Conceptos Fundamentales Variables

Lenguajes estructurados en Bloques.
Bloques nombrados, funciones.
Pasaje de parámetros.
Alcance estático y dinámico. Excepciones.
Recolección de basura.

5. Programación Funcional

Propiedades de las componentes de software declarativas.
Transparencia referencial.
Efectos secundarios.

6. Programación Orientada a Objetos

Abstracciones de la orientación a objetos.
Encapsulación, interfaz e implementación.
Herencia, herencia múltiple, mecanismos de herencia múltiple aproximada.
Niveles de visibilidad. Particularidades de diferentes lenguajes orientados a objetos: Simula, Smalltalk, C++, Java.

7. Programación Concurrente y Distribuida

Semántica de concurrencia. Abstracciones de concurrencia.
Frameworks de programación distribuida.
Concurrencia funcional.
Paradigma de actores.

8. Programación Lógica

Motor de inferencia, búsqueda.
Unificación.
Mundos cerrados.
Cut.

9. Scripting

Decisiones de diseño en los lenguajes de scripting. Lenguajes pegamento y lenguajes de dominio.

10. Frameworks

Concepto de boilerplate.
Hotspot y Frozen spot.
Inyección de dependencia.

11. Seguridad en Lenguajes de Programación

Vulnerabilidades por manipulación de bajo nivel.
Vulnerabilidades por debilidad en el sistema de tipos.
Programación defensiva y programación ofensiva.

12. Programación Asistida por Inteligencia Artificial

Introducción al aprendizaje automático y modelos de lenguaje.
Herramientas de asistencia a la programación basadas en Inteligencia Artificial.
Usos y limitaciones.

BIBLIOGRAFÍA BÁSICA

- John Mitchell. 2002. Concepts in programming languages. CUP. ISBN: 1139433482

BIBLIOGRAFÍA COMPLEMENTARIA

- Michael L Scott. 2016. Programming Language Pragmatics. Morgan Kaufmann. ISBN: 9780124104099

- Benjamin Pierce. 2002. Types and Programming Languages. MIT Press. ISBN: 9780262162098
- Van Roy & Haridi. 2004. Concepts, Techniques, and Models of Computer Programming. MIT Press. ISBN: 9780262220699
- Norman Ramsey. 2022. Programming Languages: Build, Prove, and Compare. CUP. ISBN: 9781107180185
- Saverio Perugini. 2021. Programming Languages: Concepts and Implementation. O'Reilly. ISBN: 9781284222739

METODOLOGÍA DE ENSEÑANZA

La materia se dicta en modalidad híbrida: presencial en aulas de FAMAF con transmisión sincrónica a través de meet (según lo establecido por ORD HCS 8/22). La retransmisión queda grabada y disponible para los/as estudiantes de la materia, que la pueden consultar en cualquier momento. Se reciben y contestan preguntas de los/as estudiantes presentes en el aula presencial de FAMAF y también de estudiantes presentes en la videollamada de Meet, de forma oral y escrita (por chat de meet).

La parte teórica de la materia se aborda a través de las clases magistrales, pero también con una guía de lectura creada por la cátedra, que acompaña las lecturas obligatorias con comentarios y aclaraciones, además de ejercicios prácticos que cubren los diferentes contenidos presentados, y exámenes de años anteriores. Esta parte de la materia se desarrolla de forma individual, con resolución de ejercicios en común en la clase, y respuesta de dudas, especialmente en las horas dedicadas a ejercicios.

De forma asincrónica, se resuelven consultas a través del canal correspondiente de Zulip, la plataforma de chat organizado para las materias de la Sección Computación de FAMAF.

La mitad del tiempo de dictado se dedica a clases magistrales de contenido teórico-práctico, con un 80% del tiempo dedicado a la exposición secuencial de los temas de la currícula, y el 20% del tiempo dedicado al planteo y resolución en el pizarrón de ejercicios prácticos. La otra mitad del tiempo de dictado se dedica al laboratorio de programación, con un 10% - 20% dedicado a clases magistrales sobre consignas, metodología, resolución de problemas y abordaje de los laboratorios en general, y el 80% dedicado a consultas sobre los 3 proyectos de programación propuestos. Las clases de laboratorio se llevan a cabo en formato híbrido, presencialmente en los laboratorios de computación de la FAMAF, que cuentan con más de 100 computadoras con el software necesario instalado, y también a través de meet sincrónicos para el seguimiento de grupos de trabajo con presencialidad remota. Las clases magistrales de laboratorio quedan grabadas para su posterior consulta.

El trabajo en el laboratorio se desarrolla en grupos de 3 o 4 personas (según la ratio docente-estudiante de cada año). Los estudiantes implementan 3 laboratorios según las consignas presentadas por los/as docentes, y a partir de un esqueleto de programa de partida. Cada uno de los laboratorios está orientado a desarrollar uno

de los paradigmas de programación presentados en el teórico. A modo de evaluación, el grupo tiene que entregar el proyecto implementado, que es revisado por los/as docentes a cargo, y cada estudiante resuelve de forma individual, en instancia de examen presencial, un ejercicio de programación relacionado con cada uno de los proyectos.

FORMAS DE EVALUACIÓN

- Tres evaluaciones parciales de teórico y tres entregas de proyectos de laboratorio, con un recuperatorio de teórico y uno de laboratorio. Las evaluaciones de teórico consisten en un examen escrito, las de laboratorio en entregas de código y defensa oral.

REGULARIDAD

- Para que un/a estudiante pueda obtener la condición de estudiante regular deberá aprobar al menos dos evaluaciones parciales o sus correspondientes recuperatorios y aprobar al menos dos de las tres entregas de laboratorio.

PROMOCIÓN

- Para adquirir la condición de estudiante promocional, un/a estudiante deberá aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), obteniendo un promedio no menor a 7 (siete), y aprobar todas las entregas de laboratorio con una nota no menor a 6 (seis).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Matemática Discreta II	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Esta materia aborda temas de Matemática Discreta, Teoría de Códigos de Corrección de Errores, Teoría de Complejidad y rudimentos de inteligencia artificial.

La parte principal de la materia es el estudio de algoritmos sobre grafos y networks, y especialmente el análisis de la corrección y las complejidades de los mismos.

El objetivo de esta parte es que los/as estudiantes comprendan que en muchas aplicaciones no basta dar un algoritmo sino que hay que demostrar su correctitud, dar una cota de su complejidad y demostrarla. Convergen el desarrollo de habilidades de algoritmia y matemática.

Además de esta parte central la materia brinda formación general sobre códigos de corrección de errores, el problema P-NP, especialmente pertinente al tratarse problemas, en la primera parte, para los que no se conocen algoritmos polinomiales. Por último, se abordan problemas de inteligencia artificial y se proponen algoritmos genéticos para ejemplificar posibles cursos de acción cuando no se cuenta con algoritmos polinomiales. También se dan algunos conceptos básicos de Machine Learning.

CONTENIDO

1. **Coloreo de Grafos**

Repaso de la noción de grafo. Notaciones. Coloreo de Grafos. Número cromático. Algoritmo de fuerza bruta. Problema k-Color. Definición de bipartito. Conectitud. Componentes conexas. Repaso de BFS y DFS. Algoritmo polinomial para determinar biparticidad. Propiedad: un grafo es bipartito si y solo si no tiene ciclo impares. Algoritmo Greedy de Coloreo. Ejemplos de aplicación. Ejemplo de que no siempre Greedy devuelve el número cromático.
Ejemplo de que tan mal puede dar.
Teorema importante (central para el proyecto):
Sea G=(V,E) un grafo cuyos vértices están coloreados con un coloreo propio c con r colores {0,1,...,r-1}. Sea P una permutación de los números 0,1,...,r-1. Sea V[i]={x en V tal que c(x)=i}, i=0,1,...,r-1. Ordenemos los vértices poniendo primero los vértices de V[P(0)], luego los de V[P(1)],etc, hasta $V[P(r-1)]. Entonces Greedy en ese orden coloreará G con r colores o menos. Propiedad: El número cromático es menor o

igual que Delta +1. Ejemplos donde se alcanza la cota.
Teorema de Brooks. (prueba solo para el caso no regular).
Teorema de los cinco colores.

2. **Fundamentos de inteligencia artificial**

Algoritmos de Búsqueda. Hill Climbing. Simulated Annealing. Algoritmos Genéticos: Codificación del problema. Fitness. Reproducción de Población. Terminación. Selección, Crossover, Mutación, Reemplazo. Algunas posibilidades de Mutación. Algunas posibilidades de Crossover. Single point, double, multiple points o máscara. Crossover en el caso de permutation based codifications: crossover básico, Partial Mixing Crossover y Cíclico.
Algunas posibilidades de Selección: Ruleta, SUS, Rank-based selection. Sigma based selection. Otras posibilidades de estructura: catástrofes e islas. con migraciones.

3. **Flujos Maximales**

Grafos Dirigidos. Ejemplos. Networks (redes). Flujos sobre redes. Valor de un flujo. Flujos maximales. Diversos ejemplos. Algoritmo Greedy para encontrar flujo maximal. Ejemplo donde no necesariamente encuentra flujo maximal. Definición de corte y capacidad de un corte. Caminos aumentantes de Ford-Fulkerson. Algoritmo de Ford-Fulkerson.
Propiedad: Al aumentar el flujo a lo largo de un camino aumentante de Ford-Fulkerson lo que se obtiene sigue siendo flujo.
Max Flow Min Cut Theorem:
a) El valor de todo flujo es menor o igual que la capacidad de todo corte.
b) Si f es un flujo, las siguientes afirmaciones son equivalentes:
1) f es maximal.
2) Existe un corte S tal que v(f)=cap(S).
3) No existen f-caminos aumentantes.
Ejemplos de aplicación del algoritmo de Ford-Fulkerson. Debilidades del algoritmo de Ford-Fulkerson: Ejemplo donde la complejidad no depende del numero de vértices o lados. Ejemplo donde el algoritmo no termina.
Refinamientos: Algoritmos fuertemente polinomiales: Algoritmo de Edmonds-Karp. Complejidad. Algoritmo de Dinic (o Dinitz). Complejidad de sus 2 versiones. Algoritmos de pre-flow/push: algoritmo ``wave'' de Tarjan. Complejidad.

4. **Matchings**

Matchings en grafos bipartitos, Matchings perfectos y Matchings completos. Ejemplos.
Algoritmo para encontrar matchings como aplicación de los algoritmos para encontrar flujos maximales. Modificaciones. Uso de matrices.
Definición de Gamma (S). Condición de Hall. Teorema de Hall.
Teorema del Matrimonio. (Todo grafo bipartito regular tiene un matching perfecto).
Problemas de Matchings Óptimos en grafos bipartitos con pesos.
Resolución del ``bottleneck problem'': problema del asignamiento óptimo cuando se desea minimizar el máximo (o maximizar el mínimo) de los pesos.
Resolución del problema del asignamiento óptimo cuando se desea minimizar (o maximizar) la suma de los pesos: Algoritmo Húngaro.

Codificación de complejidad O(n al cubo) del algoritmo Húngaro.

5. **Códigos de corrección de errores.**

Códigos de corrección de errores. Definiciones básicas. Distancia de Hamming. Detección y Corrección de errores. Ejemplos de códigos. Chequeo de paridad. Códigos de repetición. Cota de Hamming.
Códigos Lineales. Propiedad: Si C lineal entonces delta(C) es igual al mínimo peso no nulo.
Matrices Generadoras. Códigos lineales como espacios filas de una matriz.
Códigos lineales como núcleos de matrices. Matrices de chequeo.
Equivalencia entre matrices generadoras y de chequeo. Propiedad: todo código lineal tiene una matriz de chequeo. Proposición: Si en la matriz de chequeo no hay columnas repetidas ni nulas entonces el código correspondiente corrige al menos un error. Generalización de esta propiedad a corrección de más errores: (Teorema:) Si H es una matriz de chequeo de C, entonces delta(C)=Min{j: existe un conjunto de j columnas linealmente dependientes de H}.
Algoritmo para corregir un error. Códigos de Hamming. Códigos perfectos. Propiedad: Hamming es perfecto. Singleton Bound. Códigos MDS
Códigos Cíclicos. Rotación de una palabra. Códigos cíclicos. Códigos cíclicos mirados como polinomios. Propiedad: todo código lineal binario tiene un único polinomio no nulo de menor grado. Definición de Polinomio generador de un código cíclico. Propiedades del polinomio generador. Uso del polinomio generador para codificación: dos métodos. Matrices generadoras asociadas a los dos métodos. Obtención en forma directa a partir del polinomio generador de una matriz de chequeo con la identidad a izquierda. Polinomio chequeador.
Corrección de errores: error trapping.
Códigos de Reed-Solomon.

6. **P-NP.**

Las clases P y NP. Ejemplos. El problema SAT. El problema k-COLOR. Reducción polinomial. Las clases de problemas NP-hard y NP-completo.
Teorema de Cook: SAT es NP-completo. Teorema: 3-SAT es NP-completo. Teorema: 3-COLOR es NP-completo.
Teorema: Matrimonio trisexual es NP-completo.

BIBLIOGRAFÍA BÁSICA

- Matemática Discreta. N. Biggs,1989
- Applied Combinatorics. Roberts, 1989, Prentice-Hall.
- Data Structures and Network Algorithms. R.E. Tarjan, 1983, Society for Industrial and Applied Mathematics.
- Computers and Intractability: A Guide to the Theory of NP-completeness. Garey and Johnson, 1979, Bell Telephone Laboratories.
- Apuntes del docente de la materia, Daniel Penazzi.
- Combinatorial Optimization: Algorithms and Complexity. Papadimitriou-Steiglitz, 1998, Dover Publications.

BIBLIOGRAFÍA COMPLEMENTARIA

- Applied Combinatorics. A. Tucker, 2nd Ed., 1984
- Network Flows: Theory, Algorithms and Applications. Ahoja-Magnani-Orlin, 1993, Prentice-Hall

METODOLOGÍA DE ENSEÑANZA

Secuenciación de los contenidos:
Se empieza con la parte de grafos y coloreos de grafos.
Esto permite que los estudiantes estén haciendo el práctico de coloreo, que es largo, mientras se dictan los temas de la siguiente parte en el teórico , que requiere una introducción teórica larga de ciertos temas antes de que puedan comenzar a hacer el práctico.
Ese tema es flujos maximales, donde se dan las definiciones de networks, se plantea el problema, y se muestran algoritmos iniciales que no funcionan pero sientan las bases para los algoritmos que sí funcionan (Edmonds-Karp, Dinitz, Wave) que se dan luego de desarrollar las bases teóricas necesarias.

Luego se usa lo aprendido en esta parte para facilitar pruebas del siguiente tema, que es matchings maximales en grafos bipartitos, con y sin pesos en los lados, con dos criterios distintos de minimizar peso, reduciendo el problema a uno de flujo maximal.

Luego se pasa al tema de códigos de corrección de errores, que requiere más matemática que los anteriores (álgebra lineal) y sobre el final se da el tema de P-NP, que se da después del último parcial pues es un tema teórico.
El tema de algoritmos genéticos se da en algún momento entre estos temas, dependiendo de la distribución temporal de los parciales que nos toque. Usualmente entre matchings y códigos, o entre códigos y P-NP o como último tema.

Como es usual en Famaf, el formato son dos clases a la semana de 4 horas cada una. (8hrs/semana) Las primeras dos horas son de dictado teórico y las siguientes dos horas son de ejercicios prácticos.

El docente de teórico da el teórico, respondiendo dudas que vayan apareciendo. En el teórico no solo se da la teoría sino ejemplos de aplicación y resolución de algunos ejercicios prácticos similares a los que figuran en los prácticos. En el práctico los/as estudiantes trabajan en los ejercicios y problemas propuestos de manera autónoma y colaborativa con sus pares, consultando a los/as docentes cuando tienen una duda o se traban. Además, los/as docentes del práctico hacen un par de ejercicios al frente al final de la clase, eligiendo de entre aquellos que detectaron mayores dificultades.

FORMAS DE EVALUACIÓN

- Habrá dos evaluaciones parciales con sus respectivos recuperatorios.
- La evaluación final constará de dos partes: una parte teórica, sobre demostraciones de resultados teóricos y una parte práctica sobre resolución de ejercicios (que no es necesaria si se aprobaron ambos parciales).

REGULARIDAD

- Para regularizar los/as estudiantes deberán aprobar aprobar los dos parciales o sus respectivos recuperatorios.

PROMOCIÓN

- La materia no se promociona.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Redes y Sistemas Distribuídos	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Las redes de computadoras y las aplicaciones basadas en redes de computadoras son fundamentales para el trabajo profesional y son recursos valiosos para quienes hacen investigación y docencia. Para la formación del/de la estudiante no solo se espera que sepan usar las redes de computadoras y las aplicaciones basadas en ellas, sino también comprender cómo se arman las redes, cuáles son sus componentes y los protocolos de software para las mismas; esto les ayudará a eventualmente poder construir y administrar redes de computadoras. Los/as estudiantes aprenderán los fundamentos sobre los sistemas operativos de redes; esto les podría servir en el futuro para participar en el desarrollo de protocolos de redes o de partes de sistemas operativos de redes. En el mundo moderno hay distintos paradigmas de desarrollo de software sobre redes: cliente-servidor, peer to peer, middlewares, etc. Los/as estudiantes adquirirán las primeras experiencias de desarrollo de aplicaciones de redes basándose en algunos de dichos paradigmas y en algunos protocolos de redes. En la materia seguimos el enfoque de organizar los sistemas operativos de redes como una arquitectura de capas donde cada capa tiene sus protocolos y se abstrae de ciertos problemas; esta forma de dar la materia ayuda a organizarla y a que los alumnos la comprendan (la capa de más abajo tiene que ver con el hardware de las redes y las dos capas de más arriba son necesarias para aprender a construir aplicaciones de redes). En cada capa hacemos énfasis en conceptos fundamentales, en cómo resolver los problemas asociados a ella, y en comprender y evaluar los protocolos más importantes usados hoy en día.

Objetivos:
Los/as estudiantes deberán alcanzar los siguientes:
Conocer el hardware de las redes y entender los límites teóricos de velocidad de transferencia.
Comprender los conceptos y problemas a resolver para las distintas capas de sistemas operativos de redes (SOR) arriba del hardware de las redes. Poder hacer razonamientos acerca de protocolos de red (mediante cálculos - usando recursos del álgebra, la aritmética, el análisis matemático, y la probabilidad y estadística – el uso de los conceptos en los que se basan los protocolos, y el empleo de las reglas de los protocolos).
Poder llevar a cabo evaluaciones de cómo se comporta un protocolo de acuerdo a las propiedades que importan para el mismo.
Poder evaluar la cantidad de los recursos que un protocolo de red consume y así como explicar bajo qué circunstancias un protocolo se comporta bien y en cuáles

casos se comporta mal.
Poder comparar las alternativas de protocolos para una cierta capa de SOR entre sí desde distintos puntos de vista.
Poder programar aplicaciones distribuidas que usan APIs de comunicación de redes: aquí nuevamente los/as estudiantes deberán conocer los protocolos intervinientes y tener en cuenta las reglas por ellos definidas.

CONTENIDO

1. **Introducción**

Redes de computadoras. Servicios proporcionados por las redes de computadoras. Tipos de redes. Internet de las cosas. Sistemas operativos de red. Jerarquías de protocolos. Modelos de referencia. Protocolos de internet de las cosas. Cómputo en la nube.

2. **La Capa de Aplicación**

Enfoques para desarrollar aplicaciones de red. Estilos de arquitectura de aplicaciones de red: cliente-servidor y peer-peer. Protocolos de capa de aplicación. La web: panorama de la arquitectura, navegadores web, plug-ins y aplicaciones de ayuda, servidores web, protocolo de transferencia de hipertexto (HTTP), documentos web estáticos (HTML), páginas dinámicas, generación de páginas web del lado del servidor usando PHP, Cookies, manejo de cookies con PHP.

3. **La Capa de Transporte Primitivas y sockets**

Elementos de los protocolos de transporte. Conceptos básicos de TCP, problemas elementales sobre envío y recepción de mensajes en TCP, encabezado de TCP. Direccionamiento. Direccionamiento en TCP. Protocolos para transferencia de datos confiable: parada y espera, retroceso N, y repetición selectiva. Control de flujo, protocolos de control de flujo, control de flujo en TCP. Control de congestión. Control de congestión en TCP: distintos protocolos. Establecimiento y fin de conexiones. Establecimiento y liberación de conexiones en TCP. Administración de temporizadores en TCP. Protocolo UDP.

4. **La Capa de Red**

Aspectos de diseño de la capa de red. Conmutación de paquetes de almacenamiento y reenvío. Servicios proporcionados a la capa de transporte. Servicio no orientado a la conexión. Servicio orientado a la conexión. Algoritmos de enrutamiento: principio de optimización, enrutamiento de ruta más corta, inundación, enrutamiento de vector de distancia, enrutamiento por estado del enlace, enrutamiento jerárquico. Control de congestión: principios generales del control de congestión, políticas de prevención de congestión, control de congestión en subredes de datagramas, desprendimiento de carga. Interconectividad: cómo difieren las redes, conectando redes, fragmentación. Capa de red de Internet: protocolo IP, formatos de direcciones IP, subredes, CIDR, traducción de dirección de red (NAT), Ipv6. Protocolo OSPF (abrir primero la ruta más corta). Protocolos de puerta de enlace exterior, BGP.

5. **La Capa de Enlace de Datos**

Funciones de la capa de enlace de datos. Tramas. Servicios provistos a la capa de

red. El problema de la asignación del canal. Protocolos de acceso múltiple sin detección de portadora, protocolos de acceso múltiple con detección de portadora. Ethernet: cableado Ethernet, formato de trama, cálculo de tamaño de trama mínima, algoritmo de retroceso exponencial binario, Ethernet conmutada. Fast Ethernet. Gigabit Ethernet. Redes inalámbricas: tipos de redes inalámbricas, problemas de las redes inalámbricas, protocolo CSMA/CA; protocolo 802.11 PCF.

6. **La Capa Física**

Bases teóricas de comunicación de datos. Análisis de Fourier. Resultados de Niquist y Shannon. Coversiones entre señales digitales y analógicas. Módems. Medios de transmisión guiados y no guiados. Multiplexión. Sistema telefónico público conmutado. DSL. Sistema telefónico móvil. Internet por cable. Fibra a la casa.

BIBLIOGRAFÍA BÁSICA

- Andrew S. Tanenbaum and David J. Wetherhall. Computer Networks (6th Edition). Prentice Hall, 2021.
- Kurose, J. F. and Ross, K. W. Computer Networking – A Top Down Approach. Seventh Edition, Pearson, 2017.

BIBLIOGRAFÍA COMPLEMENTARIA

- Douglas E. Comer. Computer Networks and Internets. 5th edition, Prentice Hall, 2009.
- William Stallings. Data and Computer Communications. 8th edition, Prentice Hall, 2007.
- Larry L. Peterson and Bruce S. Davie. Computer Networks. 5th edition, Morgan Kaufmann, 2011.

METODOLOGÍA DE ENSEÑANZA

Formatos empleados para el desarrollo de las clases

Se usan clases teórico-prácticas híbridas (i.e. con asistencia presencial y remota), donde se estudia la teoría de la materia y se desarrollan actividades prácticas bajo supervisión de los/as docentes.

El laboratorio de la materia combina explicaciones conceptuales con instancias de desarrollo de código y de experimentación guiada. Las clases se desarrollan tanto en modalidad presencial física como remota, con instancias sincrónicas para acompañamiento técnico y conceptual. Se pone a disposición además, distintos material (videos, guías y enunciados) a través del Aula Virtual. Cada laboratorio incluye demostraciones, sesiones de preguntas y ejercicios en vivo para afianzar los conceptos antes de la entrega.

Secuenciación de los contenidos

Para el teórico-práctico se comienza con la introducción de la materia; luego se usa enfoque top-down recorriendo las diferentes capas de red desde las más abstractas a las más concretas; estas son: capa de aplicación, capa de transporte, capa de red,

capa de enlace de datos y capa física.

Los contenidos del laboratorio se organizan de manera incremental siguiendo el enfoque top-down de la materia. Se inicia con el desarrollo de aplicaciones de red en la capa de aplicación (Lab 1 y Lab 2), centradas en programación con sockets en Python y protocolos estandarizados o diseñados ad hoc. Luego, se avanza hacia la experimentación en capas inferiores mediante simulaciones con OMNeT++ (Lab 3, Lab 4 y Lab 5), donde se analizan fenómenos de transporte, enrutamiento y acceso al medio. Esta secuencia permite que el/la estudiante transite desde la construcción de aplicaciones distribuidas hasta la modelización y evaluación de mecanismos internos de una red real, tal como se detalla en los enunciados de los laboratorios.

Tareas y actividades que realizan los estudiantes

Para clases teórico-prácticas: i) Derivar protocolos para las diferentes capas. Ii) Comparar protocolos que cumplen la misma función dentro de una capa. iii) Hacer cálculos de uso de recursos que involucran los protocolos (latencia, utilización de canal, ventanas, espacios de secuencia, temporizadores, representación interna de la red entre otros). iv) Simular secuencias de mensajes para protocolos bajo ciertas condiciones. v) Construcción de mensajes para ciertas necesidades. vi) Diseñar redes de distintos tipos para diferentes necesidades.

Para el laboratorio de la materia las actividades principales consisten en: (i) implementación de aplicaciones cliente/servidor con sockets (Labs 1 y 2), (ii) desarrollo y extensión de modelos en OMNeT++ para estudiar control de flujo y congestión (Lab 3), enrutamiento estático y dinámico (Lab 4), y protocolos de enlace como ALOHA (Lab 5), (iii) análisis cuantitativo de resultados mediante métricas obtenidas de simulaciones (.sca, .vec), (iv) elaboración de informes técnicos y presentaciones orales grupales de hasta 10 minutos, según lo solicitado en los enunciados. Además, los/las estudiantes realizan lecturas guiadas y resuelven ejercicios de modelización, depuración de código y discusión conceptual sobre el comportamiento observado en sus implementaciones.

Materiales y recursos didácticos usados para la enseñanza y aprendizaje de la materia

Para el teórico práctico: usamos un Aula Virtual en Moodle que contiene: presentaciones audiovisuales de los temas de la materia, videos de clases grabadas, parciales de ejemplo y su resolución, listas de ejercicios y problemas de los capítulos de la materia, ejercicios y problemas resueltos

El laboratorio utiliza: Python 3 para programación con sockets (Labs 1 y 2), OMNeT++ para simulación en capas inferiores (Labs 3 a 5), repositorios Git provistos por la Facultad para entregas, presentaciones multimedia, videos tutoriales y ejemplos de código. El Aula Virtual contiene cuestionarios, foros de consulta, cronogramas, enunciados y material complementario. También se emplean herramientas de línea de comando, Wireshark, contenedores Docker en talleres prácticos y APIs específicas presentadas por la cátedra.

Asimismo, en el entorno Moodle, los/as estudiantes disponen de un foro de consultas a donde pueden hacer preguntas de los temas desarrollados en las clases teórico-prácticas, así como en los laboratorios.

Dinámicas de trabajo propuestas a los estudiantes

Para el teórico-práctico: En las clases se intercalan presentación de contenido teórico con actividades de ejercitación. Estas actividades son supervisadas por el/la docente. Algunas clases son solo de ejercitación donde hay actividades de resolución de guías de ejercicios y problemas. Antes de los parciales siempre hay una clase donde se resuelve un parcial modelo y se atienden consultas sobre los contenidos a evaluar. Los/as estudiantes pueden usar el canal de Zulip de la materia para hacer consultas sobre la materia o plantear inquietudes.

Para la parte de laboratorio las actividades se realizan en grupos de tres estudiantes, promoviendo la interacción entre pares mediante el diseño colaborativo de código, la discusión de decisiones de ingeniería y el co-análisis de resultados. Las consultas se canalizan a través de foros (Moodle y Zulip) y de sesiones sincrónicas con docentes y ayudantes. En las evaluaciones orales, la interacción es individual, lo que permite verificar la comprensión conceptual y la capacidad de justificar las decisiones de implementación. La dinámica fomenta el aprendizaje activo y el razonamiento crítico sobre el funcionamiento real de las redes de computadoras.

FORMAS DE EVALUACIÓN

- Aprobación de los 2 parciales, o de 1 parcial y de 1 recuperatorio.
- Aprobar al menos el 60% de los trabajos de laboratorio.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7.
- Entrega y aprobación de todos los trabajos de laboratorio en las fechas establecidas con nota no menor a 6.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Bases de Datos	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

En el mundo moderno las empresas y las organizaciones públicas necesitan manejar información para poder llevar a cabo sus actividades. Para poder, consultar, definir, gestionar esa información resulta imprescindible el diseño y manejo de bases de datos lo cual se puede llevar a cabo con la ayuda de herramientas de modelado y de sistemas de gestión de bases de datos.

El/la esrudiante deberá estar capacitado para:
• Diseñar modelos de datos de calidad y definir restricciones de integridad que deben cumplir los datos.
• Tomar decisiones de diseño para el modelado de datos y justificarlas.
• Poder evaluar un diseño de entidad-relación usando diferentes criterios.
• Poder evaluar el diseño de una base de datos relacional usando diferentes criterios.
• Comprender y aplicar los algoritmos de normalización enseñados para producir diseños de bases de datos relacionales de calidad.
• Poder mapear diagramas de entidad-relación a tablas de modelo relacional.
• Especificar consultas, disparadores y restricciones de integridad en SQL, a partir de descripciones en lenguaje natural provistos por clientes.
• Leer una consulta expresada en un lenguaje de consultas y entender su significado. Esto incluye SQL, MongoDB y álgebra de tablas.
• Programar usando algún sistema comercial de gestión de bases de datos. Esto incluye algún motor que soporta SQL y MongoDB.
• Poder definir índices tanto en SQL como en MongoDB.
• Poder estimar el costo de evaluar una consulta de acuerdo a un plan.
• Poder aplicar técnicas de optimización de consultas.
• Poder escribir prompts para robots de chat.
• Poder usar sistemas de retorno de la información y entender su funcionamiento.

CONTENIDO

1. **Introducción**

¿Qué es una base de datos? Aplicaciones de bases de datos. Esquemas y ejemplares. Modelos de los datos. Modelo relacional. Modelos de datos no relacionales. Lenguajes consulta. SQL. Álgebra relacional. Diseño de base de datos relacionales. Diseño de entidad-relación. Teoría de normalización. Traducción de diseño de entidad-relación a tablas. Sistemas gestores de bases de datos.

Arquitectura. Gestión del almacenamiento. Procesamiento de consultas. Transacciones. Planificaciones. Gestión de transacciones. Arquitectura de aplicaciones de bases de datos.

2. **Diseño de Entidad-Relación**

Diagramas de entidad-relación. Entidades, atributos y conjuntos de entidades. Superclaves, claves candidatas y claves primarias de conjuntos de entidades. Relaciones y conjuntos de relaciones. Clasificación de Atributos. Correspondencia de cardinalidades. Restricciones de participación. Notación de intervalos. Conjuntos de entidades débiles. Especialización y generalización. Restricciones de diseño sobre las generalizaciones. Decisiones de diseño al construir un diagrama de entidad-relación. Estructura básica de las bases de datos relacionales. Esquema de una base de datos relacional. Claves primarias. Claves foráneas. Reducción de un esquema de entidad-relación a tablas.

3. **Algebra de tablas**

Lenguajes de consulta. Álgebra relacional. Limitaciones del álgebra relacional. Algebra de tablas. Listas y sus operaciones. Tablas y sus esquemas. Operadores: proyección generalizada, selección, producto cartesiano, reunión selectiva, reunión natural, renombramiento, concatenación, resta, intersección, remoción de duplicados, agregación, agrupación, ordenamiento. Definiciones locales. Consultas usando el álgebra de tablas. Propiedades de los operadores en el álgebra de tablas.

4. **SQL**

Lenguaje de definición de datos: tipos de dominios en SQL, definición de esquemas en SQL. Restricciones de los dominios en SQL. Cláusulas select, from y where. La operación de renombramiento. Variables tupla. Operaciones sobre Cadenas. Operaciones sobre conjuntos. Funciones de agregación. Manejo de valores nulos. Subconsultas anidadas. Vistas. Modificación de la base de datos. Reunión de relaciones.

5. **Integridad y Seguridad**

Integridad referencial. Integridad referencial en SQL. Aserciones. Aserciones en SQL. Disparadores. Disparadores en SQL. Seguridad y autorización: medidas de seguridad en varios niveles, autorizaciones, concesión de privilegios, papeles. Autorización en SQL: privilegios en SQL, papeles, el privilegio de conceder privilegios.

6. **Procesamiento de Consultas**

Organización de archivos. Organización de registros en archivos. Almacenamiento del diccionario de datos. Buffer de la base de datos. Índices. Índices ordenados. Índices árboles B+ y sus extensiones. Definición de índices en SQL.
Pasos en el procesamiento de consultas. Cómo medir el costo de una consulta. Costo de operadores: selección, ordenamiento, reunión natural, eliminación de duplicados, proyección, agregación, operaciones de conjuntos. Evaluación de expresiones de consulta. Materialización. Canalización.

7. **Optimización de Consultas**

Planes de evaluación. Transformación de expresiones relacionales. Reglas de

equivalencia. Optimización basada en transformación. Optimización basada en costo. Programación dinámica en optimización. Optimización heurística. Optimizadores de consultas.

8. MongoDB

Bases de datos NoSQL. Categorías de bases de datos NoSQL. Qué es MongoDB. Bases de datos, colecciones y documentos. Documentos BSON. MongoDB Shell: Comandos. Operaciones CRUD en MongoDB. Sintaxis típica de una consulta en MongoDB. Operaciones InsertOne e InsertMany. Operación Find. Operadores de comparación. Consultas en arreglos. Consultas en documentos embebidos. Operaciones updateOne y updateMany. Operaciones deleteOne y deleteMany. Operadores de consulta, de proyección y de actualización.

9. Asuntos avanzados de MongoDB

Pipeline de agregación. $match, $project, $addFields, $group, $lookup, $unwind, $sort, $skip, $limit. Validación de esquemas de datos: Validación con JSON Schema. Validación con Operadores de Consulta. Modelado de distintos tipos de relaciones en MongoDB. Creación de índices en MongoDB para la optimización de consultas.

10. Dependencias Funcionales

Dependencias funcionales: conceptos básicos, cierre de un conjunto de dependencias funcionales, cierre de un conjunto de atributos, implicación lógica, deducción, teorema de completitud, recubrimiento canónico. Descomposición. Propiedades deseables de una descomposición: descomposición de reunión sin pérdida y preservación de las dependencias.

11. Formas Normales

Forma normal de Boyce-Codd (FNBC): definición, chequeo de FNBC, algoritmo de descomposición de un esquema relacional en FNBC. Tercera forma normal (3FN): definición, chequeo de 3FN, algoritmo de descomposición de un esquema en 3FN. Comparación de Forma normal de Boyce-Codd con tercera forma normal.

12. Retorno de la Información

Retorno de la información. Lenguajes de consulta para retorno de la información. Relevancia. Modelos booleanos. Modelos de espacio vectorial. Selección de términos de un documento. índices invertidos. Consultas usando índices invertidos. Medición de relevancia de resultados de una consulta. Máquinas de búsqueda en la web. Rastreadores web. Relevancia de documentos en la web. Popularidad de sitios web. Algoritmo pageRank.

BIBLIOGRAFÍA BÁSICA

- Silberschatz, Korth y Sudarshan. Fundamentos de Bases de Datos. Mc Graw Hill, Ediciones: Cuarta Edición (2002), Quinta (2005), Sexta (2011), o Séptima (2020).
- Elmasri, R., Navathe, S. Fundamentals of Database Systems. Pearson. Quinta Edición (2007), Séptima edición (2016).
- Ziliani F., Bordone, M. Álgebra de Tablas. 2020.

BIBLIOGRAFÍA COMPLEMENTARIA

- García-Molina, Ullman, Widom. Database System Implementation. Prentice Hall (2000).

METODOLOGÍA DE ENSEÑANZA

Formato de desarrollo de las clases

Se usan clases teórico-prácticas donde se enseñan los fundamentos de diseño de bases de datos y de sistemas de bases de datos. En las clases se ejercitan decisiones de diseño y se hacen ejercicios de diseño en base a enunciados en distintos formatos en el aula. Con respecto a los sistemas de bases de datos se ejercita definición lógica de operadores para consultas, su implementación usando ciertos recursos como índices; además se analizan estas implementaciones; adicionalmente, se estudian técnicas para evaluar una consulta y estimar su costo. Otro tipo de ejercitación es la aplicación de técnicas de optimización de consultas.

Se usan clases de laboratorio, donde los/as estudiantes aprenden a usar sistemas gestores de bases de datos comerciales para implementar bases de datos usando SQL y MongoDB. Esto involucra: diseño de los datos con sus restricciones de integridad, definición de índices sobre los datos, escritura de consultas sobre los datos, creación de disparadores y definición de transacciones.

Secuenciación de los contenidos

Para el teórico-práctico se comienza con una introducción de la materia donde se presentan algunos conceptos que se van a usar a lo largo de la materia. Luego, se aborda el diseño de bases de datos, donde se estudia modelado de entidad-relación, y su traducción a esquemas de tablas; luego se estudia diseño de bases de datos relacionales mediante algoritmos de normalización. A continuación, se desarrolla lo referente a sistemas de bases de datos. Aquí, estudiamos sistemas gestores de bases de datos relacionales y sistemas de retorno de la información.

El laboratorio acompaña estos temas con actividades 100% prácticas en computadora, donde los/as estudiantes crean modelos de datos y los traducen a tablas, escriben consultas en SQL y MongoDB, implementan disparadores, índices y restricciones, analizan planes de ejecución para optimizar consultas, utilizan motores reales de bases de datos para resolver ejercicios y practican la lectura y escritura de consultas a partir de enunciados.

Actividades que realizan los/as estudiantes

Consideramos los siguientes tipos de actividades en la materia: I) En base a un enunciado en lenguaje natural hacer un diseño de entidad relación. II) En base a diagrama de entidad relación traducirlo a esquemas de bases de datos relacionales. III) En base a enunciado en lenguaje natural, obtener los atributos del problema y las restricciones de integridad. IV) En base a el esquema universal y las restricciones de integridad (p.ej. dependencias funcionales), aplicar algoritmos de normalización. V) Chequear si los esquemas relacionales están en una determinada forma normal. VI) Especificar un operador de consultas (usando definiciones recursivas, teoría de conjuntos o cálculo de predicados). VII) Implementar un operador de consultas y estimar su costo. VIII) Tomar decisiones de implementación de los operadores de una consulta. IX) Estimar el costo de evaluar una consulta entera. X) Optimizar una

consulta usando heurísticas y programación dinámica. XI) diseñar y escribir consultas en SQL, MongoDB a partir de enunciados, XII) especificar restricciones de integridad, XIII) definir índices, XIV) Implementar disparadores.

Materiales y recursos didácticos usados

Para el teórico práctico: se trabaja presentaciones multimedia de los temas de la materia, videos de clases grabadas, parciales de ejemplo y su resolución, listas de ejercicios de los capítulos de la materia y sus correspondientes resoluciones.

Para el laboratorio se emplean guías prácticas, entornos de bases de datos instalados en las computadoras del aula, ejemplos de consultas y scripts, así como datasets y casos de uso que permiten trabajar de manera aplicada los contenidos vistos en clase.

Dinámica de trabajo propuesta a los/as estudiantes

Para el teórico práctico: se intercalan presentaciones de conceptos, técnicas, métodos y decisiones de diseño con su ejercitación. Se graban las clases y se provee acceso a ellas a través del Moodle. En algunas clases se resuelven parciales de ejemplo o parciales que ya se evaluaron. Hay listas de ejercicios de los capítulos de la materia. Algunas clases se dedican a resolver ejercicios de estas listas bajo la orientación de los/as docentes; otra modalidad es la gestión a cargo del/de la docente de la discusión y registro de las soluciones propuestas por los/as estudiantes en el pizarrón, corrigiendo y compilando lo presentado.

En el laboratorio: los/as estudiantes trabajan directamente en las computadoras, resolviendo problemas en motores reales de bases de datos, escribiendo consultas, creando estructuras, implementando restricciones y optimizaciones. El/la docente guía el trabajo, responde dudas en el momento y propone ejercicios progresivos para aplicar los contenidos del teórico-práctico en situaciones concretas. En algunas clases se resuelven parciales de ejemplo o parciales que ya se evaluaron.

FORMAS DE EVALUACIÓN

- Dos (2) evaluaciones parciales del teórico-práctico, cada una correspondiente a aproximadamente la mitad de los capítulos de la materia.
- Dos (2) recuperatorios de esos parciales.
- Hay trabajos individuales de taller con evaluación.

REGULARIDAD

- Aprobación de los 2 parciales del teórico-práctico, o de 1 parcial y de 1 recuperatorio (del parcial no aprobado).
- Aprobar al menos el 60% de los trabajos individuales de taller.

PROMOCIÓN

-Aprobación de los 2 exámenes parciales.
- Deberá tener notas no menores a 6 en cada parcial y promedio no menor a 7 en los parciales.

- Aprobar un coloquio sobre el teórico-práctico de la materia.
- Entrega y aprobación de todos los trabajos individuales de taller en las fechas establecidas con nota no menor a 6.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Ingeniería del Software I	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

La materia se organiza en clases teóricas, clases prácticas y actividades de laboratorio. En las clases teóricas se brindan los contenidos fundamentales de la asignatura. En las clases prácticas se ejercita sobre los temas cubiertos en la teoría, con especial énfasis en actividades de análisis y diseño orientado a objetos, especificación de requisitos y testing. Las clases de laboratorios se utilizan para llevar adelante un proyecto de desarrollo, de tamaño mediano, que es resuelto en grupos y en el cual los alumnos experimentan los problemas que surgen en el desarrollo de un sistema real, siguiendo todas las etapas que involucra el desarrollo de un proyecto real. Las clases teóricas son complementadas con charlas de temáticas variadas vinculadas a la ingeniería de software, brindadas por docentes de la asignatura e invitados de la industria local.

Lograr que el alumno/a sea capaz de:
Entender y aplicar actividades de análisis y especificación de requerimientos, diseño, codificación y testing de software.
Manejar elementos de planificación, especificación y documentación de proyectos usando los paradigmas funcionales y orientado a objetos para el análisis y el diseño de sistemas.

CONTENIDO

1. Introducción

El dominio del problema. La "Crisis del Software" y su Legado.
El desafío de la Ingeniería del Software, como disciplina para la Calidad.
El enfoque de la Ingeniería del Software, y su importancia frente a la actualidad donde los sistemas computacionales son ubicuos.

2. El proceso del software Procesos.

Modelo de procesos. Componentes. Enfoque ETVX.
Características deseadas del proceso del software: Predecible y repetible, Tolerante a cambios, Testeable y Mantenible.
Proceso de desarrollo del software, etapas fundamentales.
Modelos de procesos de desarrollo: Cascada, Prototipado, Iterativo.
Otros procesos del software: Administración del proyecto, Proceso de inspección, Administración de configuración, Administración de cambios, Administración del

proceso (CMM).

3. Análisis y especificación de los requerimientos del software

Requerimientos del software, Necesidad de la especificación de requerimientos, Proceso de requerimientos.
Análisis del problema: Enfoque informal, Modelo de flujo de datos (DFD), Modelo orientado a objetos (UML), Prototipado.
Especificación de los requerimientos del software: Características, Componentes, Lenguajes de especificación, Estructura de un documento.
Especificación funcional con Casos de Uso: Conceptos, Estructura, Abstracción. Validación.
Métricas: Tamaño, Calidad.

4. Arquitectura del software

Rol de la arquitectura del software.
Vistas: Módulos, Componentes y conectores, Asignación de recursos.
Vista de Componentes y Conectores (C&C;). Estilos arquitectónicos para C&C;: Tubos y Filtros, Datos compartidos, Cliente-servidor, Publish-subscribe, Peer-to-peer, Procesos que se comunican. Patrones arquitectónicos: Monolito, Multi-tier, Web architecture - REST API, Microservices, Event driven.
Documentación del diseño arquitectónico.
Arquitectura en comparación con el diseño. Preservación de la integridad de una arquitectura.
Evaluación de las arquitecturas (método de análisis ATAM).

5. Planeamiento del proyecto de software

Planeamiento del proceso.
Estimación del esfuerzo: Incertidumbres, Construcción de los modelos (estimaciones top-down y bottom-up). El modelo COCOMO. Planificación y recursos humanos: Planificación global y detallada, Estructura del equipo de trabajo.
Plan del Control de Calidad: Introducción y eliminación de errores, Enfoques, Plan.
Administración del Riesgo: Conceptos, Evaluación, Control.
Planeamiento del seguimiento del proyecto: Mediciones, Seguimiento observacional, Registro del seguimiento.

6. Diseño orientado a funciones

Niveles en el proceso de diseño.
Principios del diseño: Particionado y jerarquía, Abstracción, Modularidad. Estrategias top-down y bottom-up.
Acoplamiento y Cohesión.
Notación y especificación del diseño.
Metodología de diseño estructurado: Cuatro pasos elementales, Heurísticas de diseño, Análisis de transacción.
Verificación.
Métricas: de red, de estabilidad y de flujo de información.

7. Diseño orientado a objetos

Conceptos de la orientación a objetos: Clases, Objetos, Relación entre objetos, Herencia, Polimorfismo.
Conceptos de diseño: Acoplamiento, Cohesión, Principio abierto-errado. UML.
Una metodología de diseño: Modelado dinámico, Modelado funcional, Definición de clases y operaciones, Optimización. ORM.
Métricas.

8. Diseño detallado

Lenguaje de diseño de procesos (PDL). Diseño lógico (del algoritmo). Modelo de estado de clases (Autómatas de estado nito). Refinamiento en abstracciones de datos e invariantes de representación.
Verificación: Recorrido del diseño, Revisión crítica (bajo proceso de inspección), Verificación de consistencia y Uso de técnicas formales.
Métricas: Complejidad Ciclomática, Vínculos de datos, Métricas de cohesión.

9. Codificación

Principios y pautas para la programación: Errores comunes, Programación estructurada, Ocultamiento de la información, Practicas de programación, Estándares de Codificación. Proceso de Codificación: Incremental, Dirigido por test, Programación de a pares, Control del código fuente y construcción (build).
Refactorización: Conceptos básicos, Malos olores, Refactorizaciones comunes.
Verificación: Inspección del código, Test de unidad, Análisis Estático, Métodos formales.
Métricas: Tamaño y Complejidad.

10. Testing

Conceptos fundamentales: Defecto y desperfecto (fault & failure), Oráculos, Casos de test y criterios de selección, Psicología del test.
Testing de caja negra: Particionado por clases de equivalencia, Análisis de valores l mites, Grafo de causa-efecto, Testing de a pares, Casos especiales, Testing basado en estados (Maquinas de estado finitas).
Testing de caja blanca: Criterios basados en flujo de control, Criterios basados en flujo de datos, Testing por mutación, Generación de casos de tests y herramientas de soporte.
El proceso de testing: Niveles, Plan, Especificación de los casos de test, ejecución de los casos de test, Análisis, Registro de defectos y seguimiento. Análisis y Prevención de los defectos.
Métricas. Estimación de la con habilidad.

11. Aspectos profesionales y sociales

Computación y sociedad.
Propiedad intelectual, licencia de software y contratos informáticos.
Aspectos legales.
Responsabilidad y ética profesional.

BIBLIOGRAFÍA BÁSICA

- El curso sigue fundamentalmente el libro: Pankaj Jalote. An Integrated Approach to Software Engineering, Third Edition. Springer. 2005. ISBN: 0-387-20881-X.

BIBLIOGRAFÍA COMPLEMENTARIA

- F. Brooks. The Mythical Man Month. Addison-Wesley. 1995.
- R. Glass. The Relationship Between Theory and Practice in Software Engineering. Communications of the ACM, 39(11):1113. Nov. 1996.
- IEEE. Estándares de la IEEE sobre la Ingeniería del Software.
- B. Meyer. Object-Oriented Software Construction (2nd Edition). Prentice Hall. 2000.
- D. Parnas. Software Engineering: An Unconsummated Marriage. Communications of the ACM, 40(9):128. Sep. 1997.
- Sun Microsystems. Java Code Convention. 1997.
- R. Stallman et al. GNU coding standards. 2007.

METODOLOGÍA DE ENSEÑANZA

El curso se desarrolla mediante un enfoque teórico-práctico, que combina la fundamentación teórica con la aplicación práctica, simulando entornos reales de desarrollo de software. La metodología es la siguiente:

1. Clases Teóricas:
    - Impartidas siguiendo el libro de texto base (Jalote) complementado con material extra (artículos, casos de estudio, etc.) para actualizar y profundizar los temas.
    - Se enfocan en la exposición de conceptos, principios y técnicas de la Ingeniería del Software, fomentando la participación y el debate.
2. Sesiones Prácticas o de laboratorio:
    - Los/as estudiantes resuelven ejercicios y problemas prácticos.
    - Durante estas sesiones, los/as docentes responden dudas y proporcionan retroalimentación inmediata sobre los ejercicios y problemas, asegurando la comprensión de los temas vistos en la teoría.
3. Tareas (Take-Home):
    - Se asignan tareas para realizar fuera del aula, con un plazo aproximado de dos semanas para su entrega.
    - Los /as estudiantes tienen la oportunidad de consultar y discutir con sus docentes durante las clases prácticas, para aclarar dudas y recibir orientación.
    - Cada tarea va acompañada de clases específicas que preparan a los/as estudiantes para resolver los problemas planteados.
4. Proyecto Integrador:
    - Los/as estudiantes, organizados en grupos, desarrollan un proyecto de software que abarca todo el ciclo de vida.
    - Cada grupo es asignado a docentes tutores que realizan seguimiento y brindan asesoría personalizada.
    - Se utilizan herramientas típicas de la industria (como sistemas de control de versiones, entornos de desarrollo integrado, gestores de proyectos) y se sigue un proceso de desarrollo iterativo (por ejemplo,

Scrum) para gestionar el trabajo.
- El proyecto permite aplicar los conocimientos teóricos en un contexto real, desarrollando habilidades técnicas y de trabajo en equipo.

5. Evaluación Continua:
   - La evaluación se basa en exámenes parciales de las clases, y los proyectos, donde se valora no solo el resultado, sino también el proceso, la calidad del trabajo y la capacidad de aplicar metodologías de ingeniería del software.

FORMAS DE EVALUACIÓN

- Dos evaluaciones parciales y un recuperatorio.
- Dos proyectos prácticos.
- Un coloquio para estudiantes que aspiren a la promoción.
- Un examen final.

REGULARIDAD

- Cumplir un mínimo de 70% de asistencia a clases prácticas, o de laboratorio.
- Aprobar con una nota no menor a 4 (cuatro), correspondiente al 60%, al menos dos evaluaciones parciales o un parcial y el recuperatorio del otro.
- Aprobar el 60% de los trabajos prácticos y laboratorios.

PROMOCIÓN

-Cumplir un mínimo de 80% de asistencia a clases teóricas, prácticas, o de laboratorio.
-Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis) correspondiente al 73%, y obteniendo un promedio no menor a 7 (siete) correspondiente al 80%.
- Aprobar los trabajos prácticos.
- Aprobar un coloquio.

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Lenguajes Formales y Computabilidad	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 4° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Lograr que el/la estudiante maneje con madurez los siguientes conceptos:

• lenguajes libres de contexto
• máquinas de estado finito (autómatas a pila y máquinas de Turing)
• funciones recursivas, funciones computables y funciones Turing computables, y su equivalencia
• computabilidad efectiva y Tesis de Church
• conjuntos recursivamente enumerables y conjuntos recursivos
• el halting problem

Estos conceptos le permitirán acceder a ideas y habilidades fundamentales para el desempeño en la ciencia de la computación teórica.

CONTENIDO

1. Gramáticas y Autómatas a pila

Gramáticas libres de contexto. Lenguajes libres de contexto. Derivaciones lefmost. Autómatas a pila. Equivalencia de lenguajes aceptados por vaciamiento de pila y por alcance de estado final. Equivalencia entre los lenguajes libres de contexto y los lenguajes aceptados por autómatas a pila.

2. Funciones Σ-recursivas

Funciones Σ-mixtas. Identificación entre Σ^* y ω para un orden total sobre Σ. Funciones Σ-recursivas y Σ-recursivas primitivas. Conjuntos Σ-recursivos y Σ-recursivos primitivos. Lema de división por casos. Iteración de funciones Σ-recursivas primitivas. Cuantificación acotada de predicados Σ-recursivos primitivos. Minimización acotada de predicados Σ-recursivos primitivos. Lema de independencia del alfabeto (sin demostración).

3. Lenguaje S

El lenguaje imperativo S asociado a un alfabeto finito Σ. Sintaxis y semántica. Macros. Funciones Σ-computables. Equivalencia entre funciones Σ-computables y Σ-recursivas. Forma normal de Kleene. El halting problem. Caracterización de los conjuntos Σ-recursivamente enumerables.

4. **Máquinas de Turing**

Máquinas de Turing. Lenguaje aceptado por una máquina de Turing (por detención y por alcance de estado final). Equivalencia entre funciones Σ-Turing computables y Σ-recursivas y entre lenguajes Σ-recursivamente enumerables y lenguajes aceptados por máquinas de Turing.

BIBLIOGRAFÍA BÁSICA

- Apunte de la materia por Diego Vaggione (2025)
- Hopcroft, John E., et al. Introduction to Automata Theory, Languages, and Computation, 3rd Edition. United Kingdom, Pearson/Addison Wesley, 2006.
- Soare, Robert I.. Turing Computability: Theory and Applications. Germany, Springer Berlin Heidelberg, 2016.

METODOLOGÍA DE ENSEÑANZA

Las clases son teórico-prácticas. Se alterna entre la exposición, a cargo de docentes, de desarrollos teóricos, con momentos para la discusión y resolución ejercicios y problemas relevantes al tema. Los contenidos de la materia están desarrollados en una secuencia de 9 guías teórico-prácticas. Contienen definiciones, teoremas, ejercicios y problemas, de manera que la/el estudiante al seguir la secuencia va madurando y afianzando incrementalmente todos los conceptos de la materia. Paralelamente, está disponible el apunte de la cátedra, el cual es más abarcativo que las guías, si bien completamente compatible con éstas. Tanto el apunte como las guías están disponibles online en el Aula Virtual y son actualizados permanentemente.

Además de la resolución de ejercicios y problemas, como estrategia de aprendizaje, la/el estudiante cuenta con una tómbola de vofois disponible online. Un vofoi es un enunciado que puede ser Verdadero, Falso o Impreciso. En la tómbola es posible seleccionar al azar un número de vofois a resolver y luego le da como salida la lista de respuestas correctas y el porcentaje de aciertos que tuvo. De esta forma se entrena al alumnado en las definiciones y conceptos básicos de la materia. En particular, el hecho de que la/el alumna/o debe detectar si un enunciado es impreciso, hace el entrenamiento aún más efectivo.

Al margen de los recursos antes mencionados, el Aula Virtual de la materia cuenta con videos explicativos donde se desarrollan conceptos particulares y se discuten y proponen soluciones a problemas modélicos. Así como también un foro de consultas en el que estudiantes pueden plasmar inquietudes respecto tanto de aspectos teóricos como referentes a las dificultades que encuentran al enfrentarse a las tareas propuestas en las guías.

FORMAS DE EVALUACIÓN

Se toman tres parciales de una duración aproximada de tres horas. Los exámenes finales consisten de una parte práctica y una teórica, en general tomadas por separado. La parte práctica se toma por medio de un escrito de cuatro horas aproximadamente y la parte teórica se toma ya sea por medio de un escrito de dos horas o por medio de un examen oral de duración aproximada de una hora.

REGULARIDAD

Aprobar al menos dos evaluaciones parciales o sus correspondientes recuperatorios.

PROMOCIÓN

- La materia no contempla régimen de promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Modelos y Simulación	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 4° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

La simulación es una herramienta importante utilizada en las Ciencias de la Computación para modelar sistemas y procesos complejos en un entorno virtual. El objetivo principal de la simulación es imitar escenarios del mundo real, de forma tal que sea posible explorar, probar y optimizar diferentes variables y parámetros sin los riesgos y costos asociados con la experimentación en el mundo físico.

En esta asignatura se presentan distintos modelos probabilísticos y se desarrollan variadas técnicas para la simulación de eventos y procesos estocásticos, continuos y discretos, y el análisis estadístico de datos simulados.

Son objetivos de esta asignatura que el/la estudiante logre:
- Relacionar conceptos de probabilidad y estadística con técnicas de modelado y simulación.
- Interpretar resultados obtenidos y tomar decisiones en base a ellos.
- Diseñar, desarrollar e implementar modelos adecuados a un sistema real.
- Seleccionar las técnicas adecuadas de acuerdo al tipo de sistema a simular.

Estos objetivos alcanzados permitirán que el/la estudiante adquiera una formación sólida de los conceptos y técnicas utilizados en la simulación de sistemas, a través del procesamiento digital de modelos matemáticos probabilísticos.

CONTENIDO

1. **Revisión de fundamentos de Probabilidad y Estadística.**

Axiomas de probabilidad, probabilidad condicional e independencia. Variables aleatorias. Valor esperado y varianza. Desigualdad de Chebyshev y Ley de los grandes números.
Variables aleatorias discretas: Distribuciones binomial, Poisson, geométrica, binomial negativa, hipergeométrica.
Variables aleatorias continuas: Uniforme, normal, exponencial, gamma.

2. **Procesos de Poisson**

Procesos de Poisson homogéneos. Caracterización. Distribución del número de eventos. Distribución del tiempo entre arribos y de tiempos de arribo. Superposición y refinamiento de procesos de Poisson.
Procesos de Poisson no homogéneos. Función de intensidad y tasa media de arribos.

3. **Generación de números pseudoaleatorios**

Concepto y propiedades de un generador de números seudoaleatorios. Revisión histórica de generadores de números seudoaleatorios. Generadores congruenciales y combinaciones. Métodos actuales.

4. **Método de Monte Carlo**

El método de Monte Carlo. Aplicaciones del método de Monte Carlo para el cálculo de integrales: integración en el intervalo (0,1), en el intervalo (a, b) y en intervalos infinitos. Estimación del número pi.

5. **Generación de variables aleatorias discretas**

Método de la transformada inversa. Método de la transformada inversa. Simulación de variables uniformes discretas, Bernoulli, geométricas, de Poisson y binomial. Aplicaciones: cálculo de promedios y simulación de una permutación aleatoria. Método de composición. Métodos alternativos: el método del alias y métodos de la urna.

6. **Generación de variables aleatorias continuas.**

Método de la transformada inversa. Método de aceptación y rechazo. Simulación de variables exponenciales. Aplicación para simular variables aleatorias discretas de Poisson y variables Gamma(n, lambda). Métodos para simular variables aleatorias normales. Método polar. Simulación de procesos de Poisson homogéneos. Simulación de Procesos de Poisson no homogéneos. Método de refinamiento y mejora del método.

7. **Técnicas de validación estadística**

Tests de bondad de ajuste. El test chi-cuadrado para datos discretos. El test de Kolmogorov-Smirnov para datos continuos. Técnicas de bondad de ajuste con parámetros no especificados. El problema de dos muestras: test de rangos de Mann-Whitney o Wilcoxon.

8. **Cadenas de Markov**

Cadenas de Markov: Propiedad de Markov. Probabilidades de transición. Diagrama de transición. Estructura de clases. Clasificación de estados. Cadenas periódicas. Tiempos de alcance y probabilidades de absorción. Tiempo medio de retorno. Distribución estacionaria.

9. **Análisis estadístico de datos simulados**

Técnicas de inferencia estadística. Histogramas, distribución empírica. Estimación de parámetros de una distribución. Estimadores de máxima verosimilitud. Propiedades de un buen estimador. Error cuadrático medio y varianza de un estimador.
La media muestral y la varianza muestral. Fórmulas recursivas para el cálculo de la media muestral y la varianza muestral. Estimador de la proporción. Fórmula recursiva para el estimador de la proporción. Estimadores por intervalos del valor esperado y de una proporción.

BIBLIOGRAFÍA BÁSICA

- Apuntes de Clase: Kisbye, Patricia, "Modelos y Simulación". Disponible en el aula virtual de la materia. (Última revisión: 2025)
- Sheldon M. Ross, Simulation, Academic Press, 6th. edition, (2022)
- Sheldon M. Ross, Simulación, Prentice Hall, 2da. edición, (1999).
- Averill M. Law, W. David Kelton, Simulation Modelling and Analysis, Mc. Graw Hill, 3ra. edición, 2000.
- Averill M. Law, W. David Kelton, Simulation Modelling and Analysis, Mc. Graw Hill, 5th. edition, 2015.

BIBLIOGRAFÍA COMPLEMENTARIA

- George Marsaglia and Arif Zaman, Some portable very-long-period random number generators, Computers in Physics,(8)1, 117 (1994).
- Numerical Recipes: http://www.nr.com/oldverswitcher.html
- Donald Knuth: “Art of Computer Programming, The: Seminumerical Algorithms“, Volume 2. Addison Wesley (1997).

METODOLOGÍA DE ENSEÑANZA

Las clases se distribuyen en dos horas de clases teóricas y dos horas de clases prácticas en laboratorio. En las clases teóricas se utiliza una pizarra digital compartida, que permite que los/as estudiantes puedan tomar nota de las explicaciones del/de la docente y disponer de las anotaciones realizadas en clase. Los/as estudiantes tienen acceso con anticipación a la clase del material teórico que será desarrollado.
Las clases prácticas están coordinadas con los temas desarrollados en el teórico. Los/as estudiantes disponen de guías de ejercicios y problemas, algunos de desarrollo en papel y otros de programación. Un subconjunto representativo de estos ejercicios y problemas son explicados al frente por los/as docentes de práctico recuperando las producción, dudas e inquietudes de estudiantes.
Los/as estudiantes disponen en el Aula Virtual del material de estudio, guías de ejercicios y problemas, repositorios con códigos específicos y material complementario. También un foro de novedades y el acceso a los enunciados de trabajos prácticos especiales. El lenguaje de programación utilizado para la resolución de ejercicios, problemas y trabajo especial es Python 3.
La comunicación con los/as estudiantes se complementa con el grupo de chat con las cuentas institucionales de docentes y estudiantes.

Además de las guías de problemas y ejercicios, los/as estudiantes realizan un trabajo especial en base a un enunciado dado por los/as docentes. Este trabajo se realiza en grupos de a dos, o excepcionalmente tres estudiantes. Debe ser desarrollado en las últimas semanas de clase, aproximadamente, y culmina con la presentación de un informe y el código empleado para la obtención de resultados. Cada grupo cuenta con el seguimiento de un/a docente, desde el inicio hasta la evaluación final del proyecto.

EX-2025-01045526- -UNC-ME#FAMAF

FORMAS DE EVALUACIÓN

Se prevén:
- Tres (3) evaluaciones parciales escritas. Los/as estudiantes podrán recuperar una sola evaluación parcial.
- Un (1) trabajo práctico especial, realizado en forma individual.
- Tres (3) actividades de seguimiento no obligatorias, previas a cada parcial. Su aprobación sumará puntaje al parcial siguiente.

REGULARIDAD

Para regularizar el/la estudiante deberá cumplir los siguientes requisitos:
- Aprobar dos parciales, o un parcial y un recuperatorio.
- Aprobar el trabajo práctico especial.

PROMOCIÓN

Para promocionar el/la estudiante deberá cumplir los siguientes requisitos:
- Aprobar los tres parciales, o dos parciales y un recuperatorio, con nota no menor a 6 (seis) y promedio no menor a 7 (siete).
- Aprobar el trabajo práctico especial con una nota no menor a 6(seis)

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Lógica	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 4° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Lograr que el alumno maneje con madurez conceptos básicos de la lógica de primer orden. Estos conceptos le permitirán acceder a ideas y habilidades fundamentales para el desempeño en las ciencias de la computación teórica.

CONTENIDO

Capítulo 1:
Conjuntos parcialmente ordenados. Diagramas de Hasse. Elementos maximales, máximos y supremos. Homomorfismos de posets. Reticulados. Equivalencia de la definición geométrica y la algebraica. Subreticulados. Homomorfismos de reticulados. Congruencias de reticulados. Relación entre congruencias y homomorfismos. Reticulados acotados. Subreticulados acotados. Homomorfismos y congruencias de reticulados acotados. Reticulados complementados. Subreticulados complementados. Homomorfismos y congruencias de reticulados complementados. El teorema del filtro primo. Lema de Rasiowa y Sikorski.

Capítulo 2:
Tipos de primer orden. Términos. Unicidad de la lectura de términos. Fórmulas. Unicidad de la lectura. Variables libres y acotadas. Reemplazos.

Capítulo 3:
Estructuras de tipo T. Valor de un término para una asignación en una estructura. Valor de verdad de una fórmula para una asignación en una estructura (Tarski). Substitución. Sentencias universalmente válidas. Equivalencia de fórmulas.

Capítulo 4:
Tipos algebraicos. Álgebras. Subuniversos y subalgebras. Producto directo de dos álgebras. Homomorfismos. Congruencias. Teorema del isomorfismo. El álgebra de términos. Identidades y el teorema de Completitud de la lógica ecuacional (Birkhoff).

Capítulo 5:
Teorías de primer orden. Modelos. Concepto de prueba formal. Teorema de corrección. Consistencia. El álgebra de Lindenbaum de una teoría. Teorema de completitud de Godel. Teorema de compacidad. Aplicaciones.

Capítulo 6:

La aritmética de Peano. Algunos teoremas básicos. Inducción completa. El modelo estándar. Existencia de modelos no estándar. Análisis de recursividad del lenguaje de primer orden: los teoremas forman un conjunto recursivamente enumerable. Funciones representables. La función β de Godel. Toda función primitiva recursiva es representable. Teorema de incompletitud de Godel.

BIBLIOGRAFÍA BÁSICA

- Apunte de Lógica, por Diego Vaggione (2025). Disponible en la página de la materia.
- Ebbinghaus, Flum and Thomas, **Mathematical Logic** 3rd ed., UTM Springer-Verlag (2021).
- Mendelson, Elliott. **Introduction to Mathematical Logic**. United States, CRC Press (2015).

METODOLOGÍA DE ENSEÑANZA

Las clases son teórico-prácticas. Se alterna entre la exposición, a cargo de docentes, de desarrollos teóricos, con momentos para la discusión y resolución ejercicios y problemas relevantes al tema. Los contenidos de la materia están desarrollados en una secuencia de 14 guías teórico-prácticas que contienen definiciones, teoremas, ejercicios y problemas, de manera que la/el estudiante al seguir la secuencia va madurando y afianzando incrementalmente todos los conceptos de la materia. Paralelamente, está disponible el apunte de la cátedra, el cual es más abarcativo que las guías, si bien completamente compatible con éstas. Tanto el apunte como las guías están disponibles en el Aula Virtual y son actualizados permanentemente.

Además de la resolución de ejercicios y problemas, como estrategia de aprendizaje, la/el estudiante cuenta con una tómbola de vofois disponible online (granlogico.com). Un vofoi es un enunciado que puede ser Verdadero, Falso o Impreciso. En la tómbola es posible seleccionar al azar un número de vofois a resolver y luego le da como salida la lista de respuestas correctas y el porcentaje de aciertos que tuvo. De esta forma se prepara al alumnado en el dominio de las definiciones y conceptos básicos de la materia. En particular, el hecho de que la/el estudiante debe detectar si un enunciado es impreciso, hace el entrenamiento aún más efectivo.

Al margen de los recursos antes mencionados, el Aula Virtual de la materia cuenta con videos explicativos donde se desarrollan conceptos particulares y se discuten y proponen soluciones a problemas modélicos. Así como también un foro de consultas en el que estudiantes pueden plasmar inquietudes respecto tanto a aspectos teóricos como referentes a las dificultades que encuentran al enfrentarse a las tareas propuestas en las guías.

FORMAS DE EVALUACIÓN

Se toman tres parciales de una duración aproximada de tres horas. Los exámenes finales consisten de una parte práctica y una teórica, en general tomadas por separado. La parte práctica se toma por medio de un escrito de cuatro horas aproximadamente y la parte teórica se toma ya sea por medio de un escrito de dos

horas o por medio de un examen oral de duración aproximada de una hora.

REGULARIDAD

- Aprobar dos de tres evaluaciones parciales o sus correspondientes recuperatorios.

PROMOCIÓN

- No se contempla régimen de promoción.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Física	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 4° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

- Conocer conceptos fundamentales de la Física Clásica.
- Conocer y valorar el método científico de las ciencias naturales.
- Adquirir el lenguaje y los alcances de la Física para facilitar la realización de modelos y la integración interdisciplinaria.
- Conexión entre la Física y las Ciencias de la Computación.
- Obtener herramientas para la comprensión de fenómenos actuales.

CONTENIDOS

Se incluyen las unidades de **Mecánica, Electricidad y Magnetismo y Termodinámica** según el programa vigente que se detalla a continuación.

MECÁNICA

1. Cinemática en una y dos dimensiones

Repaso de magnitudes físicas fundamentales y vectores. Punto material y sistemas de referencia. Movimiento rectilíneo: posición, velocidad y aceleración; movimientos uniformes y uniformemente acelerados. Caída libre y tiro vertical. Movimiento en dos dimensiones: .Vector posición y velocidad. trayectoria, componentes de la aceleración normal y tangencial. Tiro parabólico. Movimiento angular, velocidad y aceleración angular. Movimiento circular uniforme y no uniforme. Movimiento relativo.Problemas de encuentro.

2. Dinámica

Masa inercial y concepto de fuerza. Leyes de Newton. Fuerzas típicas: normal, fricción, tensión, peso y fuerzas elásticas (Ley de Hooke). Diagramas de cuerpo aislado. Sistemas masa–resorte y péndulo simple. Momento lineal y centro de masa. Conservación del momento lineal en una y dos dimensiones. Extensión a dos dimensiones.

3. Energía mecánica

Trabajo de una fuerza. Teorema del trabajo y la energía. Energía potencial y fuerzas conservativas. Fuerzas disipativas y rozamiento. Conservación y transformaciones de la energía mecánica. Potencia.

4. Momento angular y campos centrales

Noción de momento angular y su conservación. Campos de fuerzas centrales. Problemas de dos cuerpos en interacción. Campo gravitatorio clásico. Nociones básicas de cuerpo rígido: rotación, energía rotacional y equilibrio.

ELECTRICIDAD Y MAGNETISMO

5. Campo Eléctrico

Carga eléctrica y cuantización. Ley de Coulomb. Definición de campo eléctrico. Representación mediante líneas de campo.

6. Ley de Gauss y potencial eléctrico

Flujo de un campo vectorial. Ley de Gauss y aplicaciones. Distribuciones de carga. Potencial eléctrico y diferencia de potencial.Campo y potencial en conductores. Energía potencial eléctrica. Dipolo eléctrico. Estructura eléctrica de la materia. Experimento de Millikan.

7. Capacidad y dieléctricos

Capacidad y combinaciones de condensadores. Energía almacenada. Medios dieléctricos y desplazamiento eléctrico. Dipolo eléctrico. Carga y descarga de un condensador.

8. Corriente eléctrica y circuitos

Conductividad eléctrica. Corriente eléctrica, Ley de Ohm y resistencia. Leyes de Kirchhoff y análisis de circuitos simples.

9. Campo magnético e inducción

Fuerza de Lorentz. Movimiento de cargas en campos magnéticos. Dipolo magnético. Fuerza entre corrientes. Ley de Ampère. Ley de Biot–Savart. Flujo magnético y Ley de Faraday.

TERMODINÁMICA

10. Conceptos fundamentales de termodinámica

Temperatura y Ley cero. Escalas de temperatura (Celsius, Fahrenheit) Escala absoluta. Expansión térmica. Variables macroscópicas y sistemas termodinámicos.

11. Gases ideales

Ecuación de estado del gas ideal. Parámetros intensivos. Procesos elementales y equilibrio térmico. Energía interna y equilibrio. Calor y trabajo. Primera ley de la termodinámica. Calorimetría.

12. Entropía y segunda ley

Definición de entropía. Direccionalidad de los procesos. Segunda ley de la termodinámica. Máquinas térmicas y ciclo de Carnot.

13. Fundamentos estadísticos

Nociones de estados microscópicos y macroscópicos. Probabilidad en sistemas físicos. Secuencias binarias, leyes de los grandes números. Información mutua y método de máxima entropía. Aplicaciones.

BIBLIOGRAFÍA BÁSICA

Se utiliza en particular *Halliday & Resnick, Fundamentals of Physics* (última edición) y *Serway & Jewett, Physics for Scientists and Engineers* (volúmenes 1 y 2, en español o inglés), disponibles en la biblioteca. Se fomenta explícitamente su uso para fortalecer y extender la comprensión desarrollada en clase. Se utilizan apuntes de la carrera de Licenciatura en Física: *Introducción a la Física y Física General I.*

BIBLIOGRAFÍA COMPLEMENTARIA

• Callen, H. B. Termodinámica e Introducción a la Termoestadística.
• Ford, K. Classical and Modern Physics.
• García, N.; Damask, A. Physics for Computer Science Students.
• Ingard, U.; Kraushaar, W. Introducción a la Mecánica, Materia y Ondas.
• Feynman, R. The Feynman Lectures on Physics (Volúmenes 1, 2 y 3).
• Feynman, R. Six Easy Pieces y Six Not-So-Easy Pieces.
• Tipler, P.; Mosca, G. Physics for Scientists and Engineers.

METODOLOGÍA DE ENSEÑANZA

a) Secuenciación de los contenidos

La secuenciación respeta la estructura del programa analítico oficial y organiza los contenidos de manera progresiva, *desde los conceptos más sencillos hacia los más complejos*. Se utiliza un *cronograma de clases* que los/las estudiantes conocen con anticipación.

Se inicia con la Mecánica, continúa con Electricidad y Magnetismo y culmina con Termodinámica. Cada unidad introduce primero ideas básicas, luego *leyes fundamentales* (Newton, Gauss, Ampère, primera y segunda ley de la termodinámica) y finalmente su aplicación a problemas concretos.

La progresión está pensada para construir comprensión conceptual de forma acumulativa y favorecer una *nueva forma de pensar (físicamente)*, mediante razonamientos encadenados, interpretación de modelos y articulación entre temas.

Se repasan temas específicos y necesarios de *matemáticas*. En cada unidad se retoman y articulan conceptos vistos previamente, promoviendo una comprensión acumulativa.

b) Formatos empleados para el desarrollo de las clases

La asignatura se dicta mediante *clases teóricas, clases prácticas y clases demostrativas con experimentos sencillo*s.

- Las clases teóricas son *dinámicas, dictadas con tiza y pizarrón*, promoviendo una *escucha activa*, la toma de apuntes y la participación. La exposición se realiza de forma dialogada, destacando los *procesos de razonamiento*, la conexión con los *hechos físicos naturales,* los pasos lógicos en la comprensión de las leyes físicas y el *marco histórico* del desarrollo de la Física.
- Las clases prácticas se orientan a la *resolución de ejercicios de aplicación*, priorizando el razonamiento lógico y el análisis conceptual por sobre el uso mecánico de fórmulas.
- En algunas clases se incorporan *demostraciones experimentales simples* (péndulo, resortes, colisiones, carga y descarga de un capacitor, circuitos, dinamómetro), utilizadas para observar fenómenos y vincular teoría con experiencia directa.

c) Tipo de tareas y actividades propuestas a los/as estudiantes

Se proponen diversas actividades formativas:

- Resolución guiada de problemas teóricos y prácticos.
- Lecturas asociadas a los capítulos pertinentes de los libros básicos disponibles en biblioteca, fomentando la relación entre clase y bibliografía, para ampliar y profundizar conocimientos.
- Actividades de modelización cualitativa, donde se identifican variables, fuerzas relevantes, supuestos y predicciones.
- *Coloquios y exposiciones* finales sobre temas seleccionados, incluyendo presentaciones orales y audiovisuales. El objetivo es enfatizar la conexión de la física con las ciencias de la computación, y utilizar los conocimientos adquiridos en la materia. Temas de los últimos coloquios: Costo energético de los cálculos computacionales, termodinámica y computación, computación cuántica, experimento de Millikan, leyes de Kepler, generación de energía en centrales termoeléctricas.
- Discusiones abiertas durante las clases, orientadas a estimular el pensamiento crítico y la formulación de preguntas.
- Uso ocasional de un *prototipo de herramienta basada en IA* en modalidad conversacional, con el objetivo de explorar y discutir ideas, no de obtener soluciones automáticas.

Si bien se emplean algunos recursos digitales, *no se hace un uso extensivo de simulaciones*, ya que la prioridad es que los/as estudiantes comprendan los *conceptos fundamentales de la física* antes de abordar modelizaciones computacionales. El objetivo pedagógico es construir una base conceptual sólida que luego permita avanzar hacia simulaciones y aplicaciones.

d) Materiales y recursos didácticos empleados

Los recursos utilizados incluyen:

- *Guías de Trabajos prácticos* (impresos o pdfs) con ejercicios de aplicación para enfatizar la construcción del conocimiento desde lo mas sencillo hasta lo complejo.
- *Intercambio de ideas teórico-prácticas* (tiza y pizarrón) como soporte principal para el desarrollo conceptual.
- *Presentaciones multimedia,* utilizadas principalmente en coloquios estudiantiles.
- *Videos breves o demostraciones simples*, para ilustrar fenómenos.
- Recursos del *Aula Virtual* (cuestionarios, materiales en PDF, consignas de tareas y entregas).
- Realización grupal de e*xperimentos simples montados,* con toma de datos, análisis de los mismos incluyendo estadísticas y presentación de resultados. Experimentos realizados: Efecto fotoeléctrico, Millikan, Órbitas en campos magnéticos.

e) Dinámicas de trabajo propuestas

Se favorece una dinámica de *interacción constante* entre estudiantes, docentes y contenido.

- En las clases teóricas se promueve un clima de participación, preguntas espontáneas y construcción conjunta de los argumentos físicos.
- En las clases prácticas, los/as estudiantes trabajan individualmente y en pequeños grupos, discutiendo estrategias de resolución y contrastando resultados.
- En actividades experimentales se aplica la secuencia *predicción–observación–explicación*, reforzando el vínculo entre modelo y evidencia.
- Las *exposiciones estudiantiles* fomentan la producción autónoma, el análisis crítico y la comunicación oral de ideas científicas.
- El enfoque general es *constructivo*, orientado a que los/as estudiantes desarrollen progresivamente herramientas de razonamiento físico aplicables en su formación en computación.

FORMAS DE EVALUACIÓN

- Se registra la *Asistencia* a teóricos y prácticos.
- Evaluación: Dos parciales y un recuperatorio.
- Se incluye el régimen de Promoción directa.
- Regularidad: asistencia al 70% y aprobación de 2 parciales.
- Promoción: asistencia al 80%, aprobación de parciales y realización de un coloquio.

- En caso de no alcanzar la promoción directa, se requerirá aprobar un *examen final* escrito de resolución de problemas integradores, al estilo de los parciales de la materia. En algunos casos se tomará una instancia de *examen final oral*.

REGULARIDAD

Se requiere:

- Asistencia al 70% de las clases teóricas y prácticas
- Aprobar 2 (dos) evaluaciones parciales con nota mayor o igual a 4 (cuatro).
- Se tomará 1 (un) parcial recuperatorio al final del curso para aquellos alumnos que no hayan aprobado uno de los dos parciales.

PROMOCIÓN

Se requiere:

- Asistencia al 80 % de las clases teóricas y prácticas.
- Aprobar 2 (dos) evaluaciones parciales con nota no menor a 6 (seis), y con promedio no menor a 7 (siete).
- Realizar un trabajo práctico y un coloquio grupal.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Arquitectura de Computadoras	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 4° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

Que el/la estudiante sea capaz de interpretar el funcionamiento de los bloques "internos" asociados a Arquitectura de Computadoras No Convencionales (No “Von Neumann”, Procesadores de Alta Prestación y Computadoras Reconfigurables). Que el/la estudiante pueda entender los conceptos básicos asociados con el incremento de rendimiento de las computadoras y técnicas de hardware y software subyacentes.

CONTENIDO

1. **PROCESADORES RISC**

Arquitectura de un procesador RISC. Implementación mediante el uso de lenguaje de descripción de hardware. Excepciones. Técnicas de mejora de rendimiento de entradas/salidas.

2. **SEGMENTACIÓN DE CAUCE (PIPELINE)**

Concepto de pipelining. Segmentación de cauce en un procesador. Hazard: de datos, estructurales y de control. Técnicas de manejo de hazard estáticas y dinámicas: Stall, Forwarding-stall.

3. **JERARQUÍA DE MEMORIAS**

Organización jerárquica de memorias. Principio de localidad de las referencias: espacial y temporal. Memoria caché. Criterios de correspondencia: Directa, Full Asociativa; Asociativa por conjuntos. Algoritmos de sustitución. Niveles.

4. **PREDICCIÓN DE SALTOS**

Predicción de saltos. Predictores dinámicos: locales y globales. Predictores híbridos. Predictores por torneo.

5. **PROCESADOR SUPERESCALAR**

Concepto de paralelismo a nivel de instrucción. Procesador superescalar estático en orden. Implementación de un procesador “multiple-issue”. Tipos de dependencia de datos. Técnicas de mejora de rendimiento estáticas: scheduling, register renaming, loop-unrolling. Procesador superescalar dinámico fuera de orden. Algoritmo de Tomasulo. Especulación.

6. **PROCESAMIENTO PARALELO**

Arquitecturas paralelas. Tipos de paralelismo: de instrucciones y datos. Límites del paralelismo. Procesador multithreading. GPU.

7. **NOCIONES DE COMPUTACIÓN RECONFIGURABLE (CR) Y DE ALTA PERFORMANCE (HPC)**

Conceptos generales, historia y estado del arte de la CR. Uso de HDL en computación reconfigurable. Nociones de codiseño Hardware-Software y su aplicación en CR. Conceptos generales, historia y estado del arte de HPC. Nociones de HPC y distribuida. Nociones básicas de clusters y arquitecturas Grid. B

Trabajos Prácticos o Laboratorios

Laboratorio 1: En el primer laboratorio se trata de poner en práctica lo aprendido sobre lenguajes de descripción de hardware y segmentación encausada (pipe-line). Para ello se les entrega una descripción HDL de un procesador de un solo ciclo y se les solicita que la modifiquen para obtener un pipeline funcional, y que evalúen el aumento de rendimiento logrado.

Laboratorio 2: En el segundo laboratorio se pretende que el/la estudiante sea capaz de comparar y analizar el impacto de modificar características de la microarquitectura de un procesador en la velocidad de ejecución de distintos códigos. Para esto se utiliza el simulador, el cual tiene una precisión de, al menos, el 95% en la determinación de la cantidad de ciclos de CLK consumidos en la ejecución de un código en procesadores.

Para la realización de cada uno de estos trabajos prácticos los/as estudiantes disponen de laboratorios de computación en la Facultad con comodidades y computadoras suficientes para la tarea encomendada. El tiempo presencial dedicado por los/as alumnos/as para este laboratorio es de 8 horas. En caso de necesidad se asignan distintos turnos para que todos/as los/as alumnos/as puedan realizar el laboratorio de manera cómoda y adecuada.

BIBLIOGRAFÍA BÁSICA

- Patterson, Hennesy, “Computer Organization and Design - ARM edition”, 2017.
- Hennesy, Patterson, “Computer architecture - A quantitative approach”, Sexta edición, 2019.

METODOLOGÍA DE ENSEÑANZA

La secuencia de dictado comienza con el estudio de arquitecturas RISC, utilizando como herramienta de estudio su descripción mediante el uso de lenguaje de descripción de hardware. Se estudian con cierto detalle los sistemas de excepciones y de entrada y salida y se bosqueja un panorama general de las distintas técnicas de mejora de rendimiento a ser estudiadas a continuación. Una vez planteado este panorama general se continúa con el estudio detallado de los conceptos de pipe-line, incluyendo los conceptos de peligros (hazards) de datos, estructurales y de control, con sus correspondientes soluciones (técnicas de manejo de hazard estáticas y dinámicas). Luego, se continúa asociando lo aprendido con el concepto de jerarquía de memorias profundizando la relación entre las memorias cache, los bancos de memoria y los pipe-lines. Se estudian, a continuación, los distintos tipos de predictores de salto y su influencia en el rendimiento general del sistema. Luego,

se estudian los procesadores multiple-issue, tanto estáticos como dinámicos. Se llega, posteriormente, al estudio de los procesadores con multithreading de manera natural. El siguiente paso es abstraer el nivel de descripción para estudiar los sistemas multiprocesadores. Finalmente, se dan nociones básicas y generales de computación reconfigurable, computación de alta performance, computadoras grid. El nivel de estos últimos temas es introductorio y motivacional para que el/la estudiante interesado/a pueda continuarlos en otras materias optativas o trabajos finales relacionados.

En cuanto al formato utilizado para el dictado de clases teóricas, se recurre principalmente a exposiciones utilizando todos los recursos audiovisuales disponibles tales como proyectores, pizarrón, etc., pero siempre fomentando la activa participación e interacción con el grupo de estudiantes. Se utiliza como herramienta auxiliar el grabado de clases para que los/as estudiantes puedan verlas después en caso que lo necesiten. Asociados a las clases teóricas se dan clases de resolución de problemas, siguiendo lineamientos similares pero con mayor interacción e iniciativa de los/as estudiantes estimulando que planteen sus dudas y consultas y compartan y discutan sobre sus producciones. Las clases prácticas también son grabadas.

Se pone a disposición de los/as estudiantes un Aula Virtual con contenidos, foros e incluso con links a videos, guías de problemas y guías de laboratorios. Los/as estudiantes pueden consultar tanto presencialmente en los horarios de clases, como así también mediante los foros disponibles para tal fin en el Aula Virtual. Para el desarrollo de los laboratorios la Facultad dispone de laboratorios de computación especiales dotados con suficientes computadoras.

En cuanto a la dinámica de interacción el grupo de estudiantes, además del/de la profesor/a encargado/a del teórico existe un número suficiente y adecuado de docentes para el desarrollo de las clases de problemas y laboratorios. Se incentiva la participación de los alumnos tanto en las clases teóricas como en las clases de soluciones de problemas. Todo ello tanto en formato presencial como a través de las herramientas del Aula Virtual tales como foros de consultas.

FORMAS DE EVALUACIÓN

1. Se tomarán dos exámenes parciales escritos sobre temas teórico-prácticos de la materia. Para cada parcial habrá una instancia de recuperación. Se aprueban con 4 (cuatro).
2. Se realizarán dos trabajos prácticos o de laboratorio obligatorios.
3. El examen final para los alumnos regulares y libres será escrito, sobre temas teórico-prácticos de la materia y podrá incluir una instancia oral, sobre temas de los laboratorios de la materia.

REGULARIDAD

- Aprobar al menos dos evaluaciones parciales o sus correspondientes recuperatorios.
- Aprobar al menos el 60% de los trabajos prácticos o de laboratorio.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y

obteniendo un promedio no menor a 7 (siete).
- Aprobar todos los trabajos prácticos o de laboratorio.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Ingeniería del Software II	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El curso introduce metodologías y técnicas avanzadas para la construcción de software confiable y seguro. Los temas tratados a lo largo del curso brindan el conocimiento fundamental y las herramientas para asegurar que el software que será parte de sistemas de alta complejidad, del cual pueden depender vidas humanas o respondan a misiones críticas, brinde un servicio correcto y efectivo.

Objetivos:
Al finalizar la materia los/as estudiantes estarán en condiciones de:
• comprender la problemática de los sistemas críticos (incluyendo sistemas concurrentes y de tiempo real) y los requerimientos fundamentales que estos deben satisfacer;
• elaborar modelos operacionales de estos tipos de modelos en lenguajes formales;
• expresar formalmente los requerimientos de estos sistemas complejos;
• seleccionar y manipular las herramientas y técnicas adecuadas para hacer los distintos tipos de análisis y verificación de modelos y especificaciones;
• comprender los fundamentos matemáticos y algorítmicos detrás de las distintas herramientas de análisis y verificación.

CONTENIDO

1. El problema de la corrección del software

(1) Definición de sistemas críticos, (2) Limitaciones del testing y la simulación, (3) Discusiones sobre verificación.

2. Programación concurrente

(1) Definición de sistemas reactivos, (2) Interacción entre procesos, (3) Los problemas de la concurrencia, (4) Semántica de los programas concurrentes, (5) Interleaving y no determinismo, (6) Razonamiento sobre programas concurrentes, (7) La necesidad de abstraer para modelar, (8) El lenguaje de modelado FSP: sintaxis y semántica, (9) La herramienta LTSA.

3. Sincronización de procesos concurrentes

(1) Recursos compartidos: interferencia y exclusión mutua, (2) Detección de errores, (3) Monitores, sincronización condicional e invariantes del monitor, (4) Semáforos y su invariante, (5) Buffers acotados, (6) Bisimulación como equivalencia de procesos,

(7) Comunicación mediante pasaje de mensajes, (8) Pasaje sincrónico de mensajes, (9) Recepción selectiva, (10) Pasaje asincrónico de mensajes, (11) Rendezvous. (12) Transacciones distribuidas.

4. Propiedades de los sistemas concurrentes

(1) Categorías de propiedades: alcanzabilidad, safety, liveness, y fairness, (2) Necesidad de la categorización de propiedades, (3) Propiedades como conjuntos de trazas, (4) Lenguajes ω-regulares, (5) Formalización de las propiedades de safety y liveness, (6) Otras propiedades, (7) Análisis automatizado de propiedades usando FSP: deadlock, safety y liveness.

5. Lógicas temporales

(1) Limitaciones de los métodos previos y de las lógicas usuales, (2) Lógicas modales, (3) Introducción a las lógicas temporales, (4) La lógica temporal lineal LTL, (5) Sintaxis y semántica,(6) Operadores derivados y leyes, (7) Especificación de propiedades con LTL: Safety y Liveness, (8) Fairness: incondicional, débil y fuerte, (9) Otros tipos propiedades en LTL.

6. Model checking

(1) El modelo de un sistema, (2) Autómatas de Büchi: definición y uso para presentar programas y propiedades, (3) Model Checking de propiedades LTL con enfoque en la teoría de autómatas, (4) Herramientas de model checking, (5) El model checking de propiedades descritas en LTL Spin, (6) Promela: modelado y análisis, (7) El model checker de propiedades descritas en CTL (computational tree logic) SMV, (8) El model checker de propiedades de tiempo Uppaal, (9) Otros model checkers.

7. Especificaciones de sistemas

(1) Características de los lenguajes de especificación, (2) Las lógicas como lenguajes de especificación, (3) Lógica proposicional: Sintaxis, semántica y poder expresivo, (4) SAT solving en la lógica proposicional: ventajas y desventajas, (5) Lógica de primer orden: Sintaxis, semántica y poder expresivo, (6) SAT solving en la lógica de primer orden, (7) El álgebra relacional. Sintaxis, Semántica y Axiomas.

8. El lenguaje de especificación Alloy

(1) Sintaxis del lenguaje Alloy, (2) Características de Alloy, (3) Uso de Alloy para la resolución de problemas con restricciones (constraint solving), (4) Modelos de ejecuciones, (5) Uso de Alloy para verificar refinamientos, (6) Análisis de especificaciones en Alloy: Cotas, cuantificadores no acotados, axiomas de generación.

9. Algoritmos para verificar satisfactibilidad en lógica proposicional

(1) Algoritmos simples: Tablas de verdad y argumentos semánticos, (2) Algoritmos avanzados, (3) Tablas de verdad revisadas, (4) Conversión a forma normal conjuntiva, (5) Regla de resolución clausal, (6) Propagación de restricciones booleanas, (7) El algoritmo de Davis, Putnam, Logemann & Loveland, (8) Cláusulas de Horn, (9) Linealidad de la resolución en la lógica de Horn, (10) La lógica de Horn como base de la programación lógica y los demostradores automáticos de teoremas.

10. Testing

(1) Definición del testing basado en modelos, (2) Testing con modelos formales, (3) El proceso de testing formal, (4) Conformidad corrección y exhaustividad, (5) La teoría de conformidad de testing basada en entradas y salidas (ioco: Input/Output Conformance Testing), (6) Extensión con tiempo y canales de la teoría ioco, (7) Definición de cubrimiento semántico.

BIBLIOGRAFÍA BÁSICA

[1] J. Magee y J. Kramer. Concurrency: State Models & Java Programs, 2nd edition. Wiley 2006.
[2] C. Baier and J.-P. Katoen. Principles of Model Checking. MIT Press, 2008.
[3] D. Jackson. Software Abstractions: Logic, Language, and Analysis (Revised Edition). MIT Press, 2011.
[4] A.R. Bradley y Z. Manna. The Calculus of Computation: Decision Procedures with Applications to Verification. Springer, 2007.

BIBLIOGRAFÍA COMPLEMENTARIA

[5] G. J. Holzmann. The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, 2003.
[6] B. Alpern y F. Scneider. Defining Liveness. Information Processing Letter 21:181-185. 1985
[7] B. Alpern y F. Scneider. Recognizing Safety and Liveness. Distributed Computing 2 (3): 117-126. 1987.
[8] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen. Systems and Software Verification Model-Checking Techniques and Tools. Springer, 2001.
[9] E.M. Clarke, O. Grumberg, D. Peled. Model Checking. MIT Press, 1999.
[10] P. Jalote. An Integrated Approach to Software Engineering, Third Edition. Springer. 2005.
[11] M. Müller-Olm, D. Schmidt, B. Steffen. Model Checking: A Tutorial Introduction. En A. Cortesi, G. Filé (Eds.), Procs. Of SAS'99, LNCS 1694, pp. 330-354. Springer 1999.
[12] J. Tretmans. A formal Approach to Conformance Testing. PhD Thesis. Univeristeit Twente,
The Netherlands, 1992.

METODOLOGÍA DE ENSEÑANZA

La dependencia de las distintas unidades se organiza según la Fig. 1. Las líneas sólidas indican una dependencia necesaria entre las unidades dado que la mayoría de los temas que comprenden la unidad requieren conocimiento de los temas enseñados en las unidades de las cuales dependen. Las líneas de punto, en cambio señalan una dependencia recomendada, en el sentido de que la enseñanza de los temas de dicha unidad se verían facilitados si ciertos conceptos previos ya fueron expuestos.
Los colores indican grupos temáticos. La primera unidad corresponde a los temas introductorios y es más bien descriptiva de lo que transcurrirá a lo largo del curso. Las Unidades 2 a 6 (en celeste) corresponden a la primera parte de la materia y agrupa lo que corresponde al modelado, análisis y fundamentos comprendiendo a

los sistemas de forma operacional. Esta parte agrupa poco más de la mitad del contenido de la materia y es la que se evalúa en el primer parcial. Las Unidades 7 a 9 (en rosado) corresponden al modelado, análisis y fundamentos comprendiendo a los sistemas y sus requerimientos en forma declarativa, es la que se dicta en la segunda parte de la materia y normalmente se evalúa en el segundo parcial. La Unidad 10 (en amarillo) se enfoca en testing basado en modelos operacionales y es la unidad que usualmente queda excluida del dictado normal de la materia por cuestiones de tiempo. Esto no afecta a los contenidos de la carrera ya que esta unidad presenta una visión más específica de conceptos básicos que ya fueron expuestos en la materia de Ingeniería del Software I pero también se trabajaron de manera menos sistemática en casi todas las materias que incluyen un laboratorio.

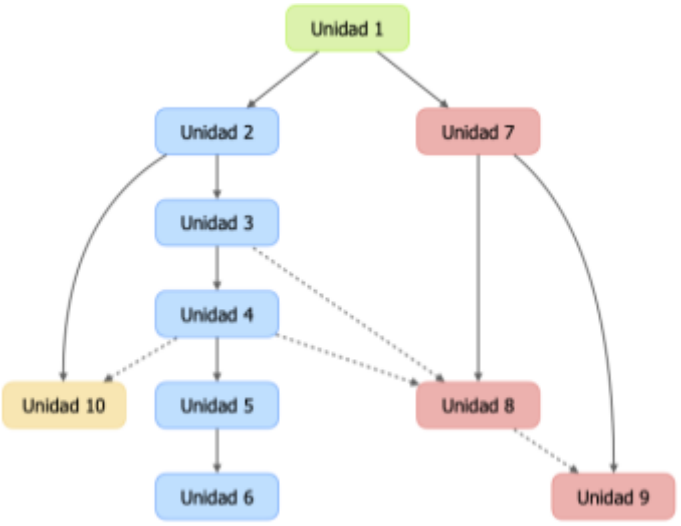

Fig. 1: Dependencia entre las distintas unidades de la materia

El dictado de la materia se organiza en teóricos y prácticos, destinando aproximadamente 60 horas al primero y 60 horas al segundo. En cada clase, usualmente, se utilizan las dos primeras horas para el teórico y las dos horas restantes para el práctico. El teórico se dicta mayormente con un programa de presentación multimedia complementado con el uso del pizarrón. Se fomenta la interacción durante las clases buscando que los/as estudiantes puedan ir descubriendo, o al menos intuyendo por sí mismos/as las soluciones a las problemáticas que presentan los distintos temas. Por otro lado, los prácticos tienen como objetivo la consolidación de los temas aprendidos en el teórico a través de la resolución de problemas presentados en guías de ejercicios prácticos. Se espera que los/as estudiantes resuelvan los problemas mayormente de manera autónoma y utilicen el tiempo del práctico para resolver consultas. Estas consultas pueden ser respondidas personalmente o también a través de la elaboración de soluciones de ejercicios paradigmáticos en el pizarrón y con la colaboración de toda la clase.

El trabajo práctico de la materia se realiza en grupo y es un trabajo de investigación integral que comprende el desarrollo de un breve manuscrito asociado y una presentación oral, que se complementa con las actividades de la materia y orientan a la consolidación de la elaboración y diseminación de trabajos de investigación y desarrollo en el área. Las consultas necesarias para la elaboración del trabajo práctico también se efectúan en el horario del práctico.

Destacamos que la materia hace uso del Aula Virtual, donde se encuentra disponible todo el material necesario para el desarrollo de la materia. Esto incluye: las presentaciones multimedia de cada unidad y las de presentación y explicación del trabajo práctico, las guías de ejercicios prácticos, la bibliografía utilizada incluyendo links en los casos que el material es de público acceso, las herramientas de software utilizadas (tanto LTSA y ALLOY, utilizadas respectivamente en la primera y segunda parte de la materia, como todas las herramientas de software que serán objeto de

estudio en el trabajo práctico), las grabaciones de cada clase teórica para poder ser accedida por estudiantes en caso de ausencia a la clase, además de información administrativa, como el programa de la materia, cronograma con fechas importantes del curso (exámenes, entregas, etc) y las condiciones de promoción y regularidad.
Además, el aula virtual provee un foro para anuncios generales de la materia y otro foro para consultas e interacción entre estudiantes (para esto último también hay habilitado un canal de Zulip). Finalmente, el Aula Virtual también provee el marco para seguimientos y entregas del trabajo práctico. Por todo esto, es importante que los/as alumnos/as mantengan activos sus correos electrónicos dado que las comunicaciones de los foros se reenvían a través de email.
En la materia se alienta a que las consultas que corresponden a la resolución de problemas y a la comprensión de los distintos temas se haga de forma pública, sea esto en clases o a través de los foros de consulta. Esta metodología permite enriquecer el aprendizaje del grupo de estudiantes ya sea porque ellos/as mismos/as son quienes realizan las consultas o porque las consultas pueden despertar nuevas dudas que de otra manera no hubieran surgido a término.

FORMAS DE EVALUACIÓN

La materia consta de dos evaluaciones parciales y la elaboración de un trabajo práctico con múltiples instancias de evaluación.
El trabajo práctico, determinante para la obtención de la regularidad, es un trabajo de investigación integral que comprende el desarrollo de un breve manuscrito asociado y una presentación oral, que se complementa con las actividades de la materia y orientan a la consolidación de la elaboración y diseminación de trabajos de investigación y desarrollo en el área.
Las dos evaluaciones parciales complementan al trabajo práctico en lo que respecta a la promoción de la materia

REGULARIDAD

- Aprobar al menos el 60% de los trabajos prácticos o de laboratorio.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).
- Aprobar todos los trabajos prácticos.

EX-2025-01045472- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Redes Neuronales	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El curso tiene como principal objetivo dotar a los estudiantes avanzados del área de neurociencia teórica y computacional del Doctorado en Neurociencias de la Universidad Nacional de Córdoba y de otras carreras afines de herramientas matemáticas y computacionales que le permitan encarar el desafío de modelar procesos neuronales, desde nivel molecular y celular hasta grandes redes de neuronas artificiales. El curso cubrirá tanto el modelado biológico de sistemas neuronales, como el estudio y uso de redes neuronales como instrumentos del aprendizaje automático. En particular se darán nociones básicas de aprendizaje profundo.

CONTENIDO

1. Elementos de Sistemas dinámicos.

El concepto de sistema dinámico. El proceso de modelado. Linealidad vs. no linealidad. Describiendo un sistema dinámico desde el punto de vista matemático. Ecuaciones diferenciales ordinarias. Clasificación de Sistemas Dinámicos. Sistemas autónomos y no autónomos. Sistemas estacionarios vs. sistemas no estacionarios. Comportamiento caótico.
El caso unidimensional. Análisis geométrico de las soluciones. Puntos de equilibrio y el concepto de estabilidad. Análisis de estabilidad lineal. Existencia y unicidad. Diagramas de fases. Métodos numéricos para la resolución de ecuaciones diferenciales ordinarias. Método de Euler y métodos de Runge-Kutta de 2 y 4 orden. Análisis de bifurcaciones. El caso bidimensional. Análisis de estabilidad lineal. Clasificación de los puntos fijos. El plano de fase. Puntos fijos y linealización. Bifurcaciones en sistemas bidimensionales. El caso tridimensional y de dimensiones mayores a tres. El ejemplo del sistema de Lorenz. El concepto de caos. Atractores extraños. Sensibilidad a las condiciones iniciales. El exponente de Liapunov. El efecto de la dimensionalidad del sistema en su dinámica.
Sistemas discretos. Mapas unidimensionales. Puntos fijos. El mapa logístico. La ruta de duplicación de período al caos.

2. Modelado matemático de neuronas.

Propiedades eléctricas de las neuronas. ¿Qué es una neurona artidicial?. Neurona de McCulloch-Pitts. Modelos “integrateand-fire”. Conductancias dependientes del voltaje. El modelo de Hodgkin-Huxley. Modelados de canales. Conductancia sináptica. Sinapsis en neuronas “integrate-and-fire”.

3. **Introducción a las redes neuronales.**

¿Qué es el aprendizaje automático? Repaso y presentación de diferentes problemas y técnicas. Aprendizaje de conceptos. Arboles de decisión. Evaluación de hipótesis. Aprendizaje Bayesiano. Conjuntos de clasificación. Reducción de dimensionalidad. Regresión lineal. Regresión no lineal y logística. Neuronas artificiales. Inspiración biológica. Historia. Redes de neuronas artificiales. La función de activación. Posibles arquitecturas.

4. **Redes neuronales Feed-forward:**

Reglas de la plasticidad sináptica. Aprendizaje no supervisado. El Perceptron simple. Neuronas escalón, lineales y no lineales. El método del descenso por el gradiente. El Perceptrón multicapas. Separabilidad lineal. El método de back-propagation y algoritmos asociados. Generalización. Aproximación de funciones continuas. Algoritmos de crecimiento de arquitecturas. Aprendizaje no supervisado. Condicionamiento clásico. Aprendizaje reforzado. Aprendizaje representacional. Aprendizaje competitivo. Aplicaciones.

5. **Redes neuronales recurrentes**

Inspiración biológica. Funciones de base radial. Redes de base radial. Algoritmos. Aplicacionees. El modelo de Hopfield para memoria asociativa. Capacidad de almacenamiento. Neuronas estocásticas. El modelo de la pseudo inversa. Dilución sináptica. Mapas auto organizados. Red neuronal de Kohonen. La máquina de Boltzmann. Autoencoders.

6. **Aprendizaje profundo**

Introducción al aprendizaje profundo. Autoencoders apilados. Redes convolucionales. Red de creencia profunda. La máquina de Boltzmann profunda. Modelos generativos profundos. Aplicaciones y casos de éxito.

BIBLIOGRAFÍA BÁSICA

- “Nonlinear dynamics and chaos”, S.H. Strogatz, Addison-Wesley Publishing Company, 1994.
- “Introduction to the Theory of Neural Computation”, J. Hertz, A. Krogh and R.G. Palmer, Santa Fe Institute, 1991.

BIBLIOGRAFÍA COMPLEMENTARIA

- “Theoretical neuroscience: computational and mathematical modeling of neural systems”, P. Dayan and L. Abbot, MIT Press, 2001.
- “Machine Learning”, T.M. Mitchell, McGraw-Hill, 1997.
- “Deep learning”, Ian Goodfellow, Yoshua Bengio and Aaron Courveille, MIT Press, 2016.
- “Neural Networks and Deep Learning”, Michael A. Nielsen, Determination Press, 2016.

METODOLOGÍA DE ENSEÑANZA

El método de enseñanza combina exposiciones teóricas, actividades prácticas y

espacios de trabajo autónomo, organizados de manera progresiva según la complejidad de los contenidos. La secuenciación parte de los fundamentos conceptuales neurobiológicos y se plasman en la formulación matemática de modelos neuronales. Se comienza con modelos de neuronas individuales, incrementando gradualmente la complejidad al tiempo que se avanza con los contenidos de sistemas dinámicos, aumentando la dimensionalidad de los modelos, al tiempo que se presentan diferentes modelos icónicos, y se culmina en el estudio de redes neuronales y técnicas de aprendizaje profundo. El enfoque parte de una mirada desde la neurociencia teórica, haciendo mucho hincapié en el modelado físico de sistemas neurobiológicos. Esta progresión permite que los estudiantes integren gradualmente los distintos niveles de descripción, desde conceptos biológicos simples hasta arquitecturas computacionales complejas.

Las clases se desarrollan en formatos teóricos y prácticos. Las clases teóricas introducen los conceptos centrales y sus fundamentos matemáticos, mientras que las instancias prácticas se orientan a la resolución de problemas de simulación utilizando Python, la implementación de algoritmos y el análisis de simulaciones.

Además, se promueve el desarrollo de pequeños proyectos en los que los estudiantes implementan y analizan, así como la producción de informes breves que integran resultados teóricos y computacionales.

Los materiales didácticos incluyen guías de trabajos prácticos, presentaciones multimedia, simuladores dinámicos y ejemplos de código en lenguajes de programación en el lenguaje Python. El aula virtual se utiliza para disponibilizar lecturas, proponer cuestionarios de autoevaluación, habilitar foros de consulta y gestionar la entrega de tareas.

La dinámica de trabajo favorece la interacción continua entre los/as estudiantes y el equipo docente. Se fomenta el intercambio entre pares mediante discusiones de problemas, análisis colaborativo de resultados y revisión conjunta de implementaciones. Los/as estudiantes cuentan con instancias de consulta permanente, tanto en clase como en el entorno del Aula Virtual, y se promueve una participación activa mediante preguntas, presentaciones breves y la discusión de estrategias de modelado.

En la última etapa del curso se invita a destacados investigadores e investigadoras a que expongan, en clases de una hora, resultados específicos obtenidos utilizando las más avanzadas técnicas de aprendizaje automático.

FORMAS DE EVALUACIÓN

Se darán dos trabajos prácticos que se deben entregar a lo largo del curso que se realizan en grupos de dos o tres estudiantes. Estos se evalúan con calificación de 0 a 10 puntos. Además de aprobar los estos dos trabajos prácticos, los alumnos regulares deberán defender, el día del examen final, un trabajo final integrador individual dispuesto por el equipo docente. Con tres días de antecedencia al examen, los estudiantes regulares deben enviar un informe escrito (formato PDF) del trabajo final integrador.

REGULARIDAD

1. cumplir un mínimo de 70% de asistencia a clases teóricas, prácticas, o de laboratorio.

2. aprobar al menos dos evaluaciones parciales o sus correspondientes recuperatorios.

PROMOCIÓN

No se contempla régimen de promoción.

CORRELATIVIDADES

Para cursar: Algoritmos y estructura de datos (regular) Análisis Matemático II (aprobada).
Para rendir: Algoritmos y estructura de datos (aprobada).

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Lenguajes y Compiladores	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

El establecer el significado de las frases de un lenguaje de programación es un problema de múltiples aristas en tanto puede tener variados objetivos, que van desde la necesidad de comprensión humana, hasta el imperativo de que una máquina los pueda interpretar o traducir a una secuencia de instrucciones ejecutables. Un manual de usuario/a, una estrategia de compilación, o alguna herramienta teórica destinada a desentrañar los principios básicos de su diseño, constituyen todas vertientes de significado que responden a distintos intereses y usos de los lenguajes de programación. En las últimas décadas variados desarrollos matemáticos y lógicos dieron forma a una teoría que se posicionó en un lugar privilegiado para el acceso a la comprensión profunda del significado de un lenguaje. La misma permite conectar la descripción intuitiva de un sentido finito y dinámico (un manual), con una modalidad estática del significado, vigente en la lógica formal y la matemática (denotación). A partir del desarrollo de la Teoría de Dominios la semántica denotacional adquiere una relevancia especial, no sólo por tratarse de objetos matemáticos perfectamente definidos en el contexto de una teoría particular, sino además porque comienza a ser utilizada como "la definición" del lenguaje y luego, si se proponen otras semánticas (operacional, axiomática), se las demuestra correctas con respecto a dicha definición.

El objetivo general de la asignatura es lograr que los/las estudiantes se apropien de las herramientas más importantes que actualmente se utilizan para definir sintácticamente y dar significado a las frases de un lenguaje de programación, poniendo énfasis en la utilidad de estas herramientas para comprender los principios básicos que subyacen en su diseño.

Dentro de los objetivos específicos, mencionamos como relevantes:
• apropiarse de conceptos fundamentales sobre la estructura gramatical de lenguajes de programación, tales como sintaxis abstracta, variables ligadas y alcance,
• tomar contacto con un lenguaje teórico basado en Standard ML, en tanto lenguaje que ha sido definido formalmente de manera completa, y cuyos principios básicos coinciden con los lenguajes más populares,
• acceder al uso de herramientas matemáticas apropiadas para el estudio de los lenguajes de programación,
• disponer de recursos para evaluar las características principales de lenguajes

cercanos a lenguajes reales actualmente en uso,
• reconocer propiedades deseables en lenguajes de programación y las herramientas para garantizarlas,
• proveer de recursos para que el/la estudiante pueda diseñar e implementar lenguajes de programación.

CONTENIDO

1. **Herramientas básicas para dar significado a los lenguajes de programación**

• Nociones en relación a la sintaxis: gramática, gramática abstracta, sintaxis abstracta, lenguaje y metalenguaje.
• Distintas formas de dar significado a los lenguajes de programación. Semántica denotacional: las nociones de frase, dominio semántico y función semántica. Semántica operacional: las nociones de configuración, regla de transición y ejecución.
• Nociones en relación a la definición del significado: dirección por sintaxis, semántica composicional.
• Variables y ligadura (en la lógica de predicados). Sustitución y el problema de la captura. Propiedades de coincidencia y renombre.
• El problema de dar significado a ecuaciones recursivas. Dominios, función continua y teorema del menor punto fijo. Análisis de las soluciones de una ecuación recursiva a la luz del teorema del menor punto fijo.

2. **Lenguajes imperativos.**

• Conjunto de estados. Semántica denotacional de las construcciones básicas de un lenguaje imperativo.
• El problema de dar significado a la iteración. Significado de la iteración utilizando el teorema del menor punto fijo.
• Propiedades de coincidencia y renombre.
• Fallas y manejo de excepciones. Output. Input.
• Semántica operacional para el lenguaje imperativo.
• Corrección respecto de la semántica denotacional.

3. **Lenguajes aplicativos**

• Las nociones de reducción y evaluación en el Cálculo Lambda. El problema de la terminación. La noción de forma canónica. Modalidad de evaluación: Eager y Normal.
• El problema de la semántica denotacional: el modelo D infinito y sus variantes.
• Lenguaje aplicativo. Sintaxis. Semántica operacional eager y normal: la noción de evaluación, formas canónicas y reglas de evaluación. Tratamiento de errores.
• Semántica denotacional directa del lenguaje aplicativo. Sintaxis y semántica de la recursión en las modalidades eager y normal. Propiedades.

4. **Lenguajes aplicativos con una componente imperativa.**

• Los problemas de la combinación de paradigmas.
• Las nociones de estado, ambiente, identificador y variable.
• Un lenguaje que combina los paradigmas. Semántica denotacional y operacional.
• Construcciones imperativas como abreviaturas. Propiedades.

• Funciones y procedimientos. Pasaje de parámetros.

5. Sistema de tipos simples para el cálculo lambda.

• Tipos simples (atómicos y funcionales) y contextos de tipado.
• Juicios, reglas de inferencia; tipado explícito. Preservación.
• Semántica denotacional: extrínseca e intrínseca.
• Relación entre las dos semánticas mediante relaciones lógicas.

BIBLIOGRAFÍA BÁSICA

- Fridlender, Daniel y Gramaglia Héctor. Apuntes de Cátedra (basados en el libro de Reynolds), 2022.
- Reynolds, John. Theories of Programming Languages, Cambridge University Press, 1998.

BIBLIOGRAFÍA COMPLEMENTARIA

- Harper, Robert, Practical foundations for programming languages. Cambridge University Press, 2013.
- Tennet, Robert D., Semantic of Programming Languages, Prentice Hall, 1991.
- Hindley, J. Roger y Seldin, Johnatan P. Lambda-Calculus and Combinators, an Introduction, Cambridge University Press, 2008.
- Astarte, Troy. Formalising Meaning – a History of Programming Language Semantics (Tesis doctoral) School of Computing, Newcastle University, Reino Unido, 2019.

METODOLOGÍA DE ENSEÑANZA

Cada unidad se desarrolla en clases teóricas expositivas donde se introducen los conceptos fundamentales de la unidad motivándolos con ejemplos de lenguajes de programación que las/os estudiantes han visto en la carrera y estableciendo vínculos con temas de otras asignaturas. Cada clase teórica tiene una duración de dos horas y habitualmente se hace una pausa de cinco minutos a la mitad. Se fomenta la participación a través de la discusión sobre los conceptos y cómo se aplican para la resolución de problemas concretos (por ejemplo, discutiendo posibles estrategias de prueba para un teorema, presentando un ejercicio y evaluando grupalmente posibles formas de encarar su solución).

A continuación de cada clase teórica sigue una clase de prácticos que también tiene una duración de dos horas. Hay una guía de prácticos por cada unidad. En las horas de práctico se espera que las/os estudiantes resuelvan los problemas de la guía que corresponde a la unidad que se está viendo; se invita a que este sea un trabajo grupal. Además, hay una o dos profesoras/es en el aula para consultas. Al final de cada clase práctica se presenta y discute en el pizarrón la solución de algún ejercicio que sea de interés para la clase (ya sea por su dificultad o porque el ejercicio es importante para comprender la unidad).

La materia incluye la presentación grupal (de hasta tres integrantes) de un tema a convenir entre el grupo y la cátedra. Además, de la presentación cada grupo debe

realizar un breve informe escrito que es entregado con algunos días de anticipación. El objetivo de esta actividad es el desarrollo de habilidades para presentar un tema a una audiencia de pares; por ello, al estructurar el informe (y la presentación) se debe incluir una introducción al tema incluyendo la vinculación con lo desarrollado en la materia. Si el tema es sobre un lenguaje de programación se deben resaltar los aspectos fundamentales del lenguaje; si el tema es más teórico, se deben presentar los teoremas más importantes junto con aplicaciones de esos resultados. La presentación es oral con soporte audiovisual y en ella los/as estudiantes responsables de la presentación deben responder preguntas tanto de sus pares como de la cátedra. La exposición oral es de 15 minutos y hay un tiempo adicional de 5 minutos de preguntas y respuestas. En la evaluación se tendrá en cuenta tanto el informe (estructura, escritura, contenido) como la claridad en la exposición de la presentación y la capacidad de responder preguntas.

Todos los materiales de estudio de la cátedra se ponen a disposición de los estudiantes a través del aula virtual (exceptuando libros con copyright). La materia habilita la consulta asincrónica virtual de dudas a través del foro del aula virtual.

La materia cuenta con un único trabajo de laboratorio que consiste en la implementación de un intérprete. Este taller es una actividad autónoma e individual con guía de la cátedra. Cada estudiante puede elegir el lenguaje a interpretar y también en qué lenguaje programar el intérprete; esta elección debe ser aprobada por la cátedra a fin de determinar la viabilidad del proyecto y que no sea trivial. Se da la posibilidad de realizar un intérprete en Haskell para un lenguaje imperativo simple (con input/output y fallas) a partir de una base de código que deben completar. La evaluación del taller es de aprobado o no aprobado. Las consultas que puedan surgir sobre la implementación del intérprete se atienden de manera asincrónica, vía foros del aula virtual y mails a docentes de la cátedra o personalmente, tanto en clases de teórico como de práctico.

FORMAS DE EVALUACIÓN

La materia cuenta con un aula virtual donde se encuentra información más detallada, como las fechas de las evaluaciones.

• Se tomarán 2 (dos) exámenes parciales. Las evaluaciones parciales serán sobre contenidos teórico-prácticos. El formato de estas evaluaciones consistirá en la resolución de actividades en el aula.

• Los/as estudiantes realizarán un taller (como parte de la carga horaria de práctico) que consiste en la elaboración de un intérprete de un lenguaje de programación pequeño.

• En las últimas tres semanas les estudiantes realizarán, en grupo de tres personas, un informe y una breve presentación sobre algún tema no presentado por la cátedra. El objetivo de esta actividad es la práctica de escritura académica y también de realizar una exposición oral.

• La aprobación de la materia se dará por promoción, o mediante la aprobación de un examen final en las fechas destinadas a exámenes en el calendario académico. El examen final constará de una evaluación escrita con un formato similar al de los parciales sobre contenidos teórico-prácticos; en caso que el tribunal lo considere necesario, puede complementarse el examen teórico con preguntas orales.

EX-2025-01045526- -UNC-ME#FAMAF

REGULARIDAD

- Aprobar las dos evaluaciones parciales o sus correspondientes recuperatorios. Se podrá recuperar uno de los dos parciales en la última semana de dictado de la materia. Aprobar el taller.

PROMOCIÓN

- Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete). Aprobar el taller. Aprobar el informe y la presentación oral.

EX-2025-01045526- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Computación Paralela	**AÑO:** 2025
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación **PLAN:** 2002	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

FUNDAMENTACIÓN: la tecnología de los microprocesadores modernos contiene un alto nivel de paralelismo. Para poder utilizar eficientemente estas arquitecturas se debe conocer los modelos de ejecución y las formas de sacar provecho a este paralelismo.
OBJETIVOS: Que el estudiante comprenda las tres dimensiones de paralelismo que actualmente posee una arquitectura de microprocesador: paralelismo de instrucciones (ILP), de datos (DLP) y de hilos (TLP), tanto en sus variantes de CPU como de GPU. Comprender las soluciones de compromiso de cada una de estas arquitecturas para obtener alto desempeño tanto en cálculo como en acceso a memoria. Saber discernir si un proceso está realizando un uso adecuado de todas las capacidades de la máquina.
Al final de la materia los estudiantes deben ser capaces de adaptar programas a fin de utilizar estas tres dimensiones del paralelismo, tanto en CPU como en GPU.

CONTENIDO

1. Introducción

- Escalado. Leyes de: Amdahl, Gustafson, Little. Eficiencia.
- Factores que degradan el desempeño: inanición, latencia, sobrecarga, contención.
- Paralelización: descomposición en tareas, orden y agrupamiento de tareas, descomposición de datos, datos compartidos.
- Sincronización: condiciones de carrera, instrucciones atómicas.
Primitivas de sincronización: mutexes, spinlocks, semáforos, barreras y fences.
- Predicción de desempeño: modelo roofline. Medición de desempeño.

2. CPU

- Paralelismo de instrucción (ILP): pipelining, procesadores superescalares, ejecución fuera de orden, SMT.
- Memoria: jerarquía y asociatividad de caché, alineamiento de memoria, algoritmos cache-aware y cache-oblivious. Memoria virtual: efectos de la TLB en el desempeño. Memoria distribuída: NUMA, coherencia de cache. Afinidad de memoria y pinning de hilos a cores.
- Vectorización: unidades SIMD, SSE intrinsics, técnicas de vectorización.
- OpenMP: constructores work-sharing, atributos para compartir datos, planificadores, sincronización, entorno de ejecución, compilación.
- Aplicaciones: extensiones ISA específicas para aplicaciones, bibliotecas para HPC.

3. **GPU**

- Arquitectura interna.
- Limitaciones de la GPGPU: serialización de saltos, ocultamiento de la latencia, ocupación.
- Jerarquía de memoria, caché de software vs. caché de hardware, unidades de textura.
- CUDA: mapeo hilo-dato, lanzamiento de kernels, comunicación host-device, sincronización, contadores de desempeño y profiling, manejo de errores, compute capabilities, PTX ISA.
- Optimización: aumento de la granularidad de los hilos, uso efectivo de la memoria compartida, código sin saltos, double buffering, reducción del uso de registros, aritmética de precisión mixta, cómo evitar instrucciones atómicas.
- Ejemplos de Algoritmos GPU: reducción, scan segmentado, compactación de streams y sus usos.

Bibliotecas: CUBLAS, CUFFT, CUSPARSE, Thrust, CUDPP, CUB.

4. **Computación heterogénea**

Utilización de sistemas runtime para la paralelización heterogénea (CPU+GPU) de algoritmos. Ejemplos de uso. Bibliotecas de álgebra lineal heterogéneas.

BIBLIOGRAFÍA BÁSICA

- Ruud Van Der Pas, Eric Stotzer, Christian Terboven, Using OpenMP-The Next Step: Affinity, Accelerators, Tasking, and SIMD 1st Edition, 2017.
- B. Chapman, G. Jost, R. van van der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, 2007.
- Wen-mei W. Hwu, David B. Kirk, Izzat El Hajj, Programming Massively Parallel Processors: A Hands-on Approach 4th Edition, 2022.
- NVIDIA Inc., CUDA C Programming Guide, versión 13.1, 2025.
- NVIDIA Inc., CUDA C++ Best Practices, versión 13.1, 2025.
- NVIDIA Inc., PTX ISA 3.2, versión 9.1, 2025.

BIBLIOGRAFÍA COMPLEMENTARIA

- J. Hennessy, D. Patterson, Computer Architecture: A Quantitative Approach 7th Edition, Morgan Kaufmann, 2025.
- J. Hennessy, D. Patterson, Computer Organization and Design MIPS Edition: The Hardware/Software Interface, 6th Edition, Morgan Kaufmann, 2020.

METODOLOGÍA DE ENSEÑANZA

La secuencia de contenidos es bottom-up, desde el nivel más básico de paralelismo (ILP) hasta el más externo el TLP. Una vez vistas las 3 dimensiones del paralelismo de un microprocesador moderno CPU, se muestra una implementación distinta a la CPU que se denomina GPU. En el laboratorio se sigue exactamente la misma secuencia, con una sincronización (de barrera) casi perfecta.

Las clases teóricas son expositivas, y utilizando en gran medida la CLI (interfaz de línea de comandos), yendo y viniendo desde “C” a ensamblador para entender cómo es la manera de sacarle mejor provecho a las arquitecturas actuales. Se utilizan muchas herramientas de profiling y monitoreo para diagnosticar problemas de desempeño. Durante las clases los errores y la búsqueda de soluciones es la práctica común que se presenta en la línea de comandos. También se utiliza mucho el desarrollo de hipótesis de desempeño previas a la ejecución, a fin de derribar mitos. Las clases de laboratorio son en su mayoría de trabajo en grupo, y se dan consultas que ayuden a destrabar problemas, sugiriendo también líneas de avance.

El cursado del teórico y del laboratorio forman un solo conjunto y todo lo que se explica en el teórico, se usa de manera directa en los laboratorios para cada uno de los proyectos. Se usan herramientas como tiem, perf, Intel VTune Profiler, NVIDIA Nsight Systems, NVIDIA Nsight Compute, Compiler Explorer y los compiladores más populares: GCC, CLANG e Intel.
Cada proyecto se presenta en forma de video de 5 minutos y luego de cada video se realiza una puesta en común con todos los grupos. Usualmente esto implica que se aplican mejoras aprendidas en esta etapa antes de continuar con el siguiente nivel de paralelismo.

Los recursos didácticos son muchas discusiones actuales en foros y redes sociales sobre Performance Optimization, sobre todo en el ámbito de game engines. Se utilizan muchas herramientas de visualización para incorporar ideas como latencia y caché awareness.

La dinámica de trabajo le valió un premio a esta materia en

- Carlos Bederián, Nicolás Wolovick, A Project-based HPC Course for Single-box Computers, EduHPC'16 (best paper award), SC16, 2016.

Y se basa en hacer grupos de dos personas, una de CS y otra de noCS, donde se potencian los saberes y se baja el prejuicio de ambas partes.

FORMAS DE EVALUACIÓN

El/la estudiante deberá elegir un programa de computación numérica intensiva (simulación), que será paralelizado de 4 formas:
1. ILP, cache-aware.
2. SIMD (instrucciones vectoriales).
3. Multicore (típicamente OpenMP para CPU).
4. Manycore (típicamente CUDA para GPU).

Se entregará un informe final donde se comparen las mejoras obtenidas. En el primer punto se deberá analizar la mejor utilización de las unidades de ejecución, y la mejora en las tasas de cache-hit. En el segundo caso la mejora que se obtuvo al operar de manera vectorial sobre los datos y la estrategia de paralelización utilizada, así como un mínimo análisis de scaling respecto al ancho de la unidad vectorial (SSE4 vs. AVX). Para multicore CPU además de analizar el scaling con respecto a la cantidad de cores, se deberá informar sobre los efectos de la afinidad de memoria-cpu. Finalmente para manycore GPU se harán análisis de weak-scaling y

de utilización del ancho de banda de memoria y potencia de cálculo respecto al pico teórico.

REGULARIDAD

- Aprobar al menos 3 (tres) de los 4 (cuatro) laboratorios.

PROMOCIÓN

- Aprobar todos los laboratorios con una nota no menor de 6 (seis) y promedio de 7 (siete).

CORRELATIVIDADES

Para cursar: Tener regularizadas Sistemas Operativos y Arquitectura de Computadoras, y tener aprobadas Algoritmos y Estructuras de Datos I y Organización del Computador.
Para rendir: Tener aprobadas: Sistemas Operativos y Arquitectura de Computadoras.