EX-2026-00088647- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Algoritmos y Estructuras de Datos I	**AÑO:** 2026
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 1° cuatrimestre / Redictado: 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 180 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 de 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

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.

EVALUACIÓN

FORMAS DE EVALUACIÓN

- 2 evaluaciones parciales de teórico/práctico donde, se podrá recuperar una instancia.
- 2 Trabajos Prácticos 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 al menos dos evaluaciones parciales de teórico/práctico o sus correspondientes recuperatorios (podrá recuperar una sola instancia).
- Aprobar al menos el 60% de los Trabajos prácticos de laboratorio o sus correspondientes recuperatorios (podrá recuperar una sola instancia).

PROMOCIÓN

La materia no tiene promoción.