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

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Lenguajes y Compiladores	**AÑO:** 2026
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
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

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.

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.

EVALUACIÓN

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.

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.