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

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Paradigmas de Programación	**AÑO:** 2026
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

ASIGNATURA: Modelos de Programación	**AÑO:** 2026
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Matemática Aplicada	
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

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.

EVALUACIÓN

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).