1. Objetivos

- Transmitir a los alumnos conceptos fundamentales vinculados al diseño de lenguajes de programación para facilitar el análisis de los distintos paradigmas.
- Hacer operativos los conceptos básicos, mediante procesos de reflexión, creación y ejercitación.
- Desarrollar conocimientos y competencias vinculadas con el diseño y criterios de diseño de lenguajes de programación.
- Desarrollar competencias para la identificación y selección de paradigmas.
- Integrar los conocimientos particulares, vinculados al diseño de lenguajes de programación, con otros conocimientos pertinentes, ya logrados durante el curso de la carrera.
- Ajustar la visión individual conformada desde la teoría y la práctica realizada durante la carrera sobre la programación, los lenguajes de programación y su implementación a su marco teórico actual; completarla.

Al finalizar el curso los alumnos estarán preparados para

- Analizar y juzgar críticamente cualquier lenguaje de programación en función de su aplicabilidad, estructuras, componentes e implementación.
- Clasificar lenguajes de programación en función de su uso, aplicabilidad e implementación.
- Vincular las características de los lenguajes de programación con los criterios que determinaron su diseño (epistemología: conocimientos teóricos, sintaxis, semántica) y a las posibilidades de implementación (estado del arte: hardware, software).
- Vincular los distintos paradigmas con su modelo semántico.

2. Contenidos

Unidad 1 - Introducción al estudio de los lenguajes de programación.
Evolución de los principales lenguajes de programación. Perspectiva epistemológica / histórica de los principales lenguajes de programación. Criterios de implementación. Lenguajes iniciales. Lenguajes de la década del 60. Influencia del Algol en los lenguajes superiores. Lenguajes de la década del 70: Pascal, C. Lenguajes orientados a objetos: C++. Lenguajes visuales.
***Paradigmas de programación*:**. Perspectiva epistemológica: concepto de paradigma, nociones básicas de semántica formal. Caracterización de los paradigmas de lenguajes de programación según elementos significativos de diseño e implementación.
Paradigmas imperativo, estructurado, funcional,lógico, concurrente, orientado a objetos, scripting. Su relación co la evolución histórica de los lenguajes de programación.

Unidad 2. Herramientas de compilación.

Análisis léxico y sintáctico. Traducción. Gramáticas independientes del contexto. Gramáticas regulares. Sintaxis de los lenguajes de programación. Reglas sintácticas. Notaciones. Reglas de producción. Derivaciones.
Semántica de los lenguajes de programación. Nociones básicas de semántica formal. Concepto de máquina virtual.
***Compilación*.** Árbol de Parsing. Árboles de análisis sintáctico. Fases de compilación de un programa. Estructura de un compilador. Diferencias entre compilador e intérprete.

Unidad 4. Teoría de la programación.
Conceptos básicos
Valores y tipos. Tipos primitivos. Tipos compuestos. Tipos recursivos. Sistemas de tipos. Expresiones. Variables y almacenamiento. Tiempo de vida. Variables globales y locales. Variables heap. Variables persistentes.Punteros y tipos recursivos. Tratamiento de punteros
Ligadura y ámbito. Entidades y ligaduras. Noción de ligadura. Ligadura y entorno. Ámbito y visibilidad. Ambitos estáticos y dinámicos. Declaraciones. Efectos colaterales. Ámbito de las declaraciones. Bloques. Tratamiento de casos de implementación. Relación con los paradigmas.
Abstracción procedural. Funciones y procedimientos. El principio de abstracción.
Parámetros y argumentos. Mecanismos de copia, de referencia.
Estudio de cuestiones de implementación, en relación con distintos paradigmas.
Conceptos avanzados. Abstracción.
Abstracción de datos. Objetos y clases. Clases, subclases y herencia, clases abstractas, herencia simple vs. herencia múltiple. Paquetes genéricos. Clases genéricas. Verificación de tipos y ámbitos. Verificación fuerte de tipos. Conversiones entre tipos. Polimorfismos, niveles de polimorfismos.
Abstracción de procesos. Subprogramas y su implementación. Registro de activación. Llamado a subrutinas. Administración de registros de activación en Algol. Variables dinámicas. Arreglos flexibles. Organización de los registros de activación. Cadenas estáticas y dinámicas. Pasajes de parámetros.
Flujo de control. Secuenciadores. Saltos. Escapes. Excepciones. Cuestiones de implementación.
Concurrencia. Programas y procesos. Problemas con la concurrencia. Interacción entre procesos. Primitivas de concurrencia. Abstracciones de control. Casos de implementación.

Unidad 5. Paradigmas
Programación imperativa. Programación Orientada a Objetos. Programación concurrente. Programación funcional. Programación lógica. Scripting.
Análisis de componentes y su implementación, de uno o más lenguaje de programación correspondientes al paradigma.

3 Bibliografía:

Obligatoria

WATT, D. :Programming Languages Concepts and Paradigms", Hoare Series Editor, Prentice Hall, (1990) http://www.acm.org

Bibliografía complementaria:

1. BACKUS, J. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. In: Comm. of the ACM, Vol. 21, Nro. 8, pp. 613-641 (1978). http://www.stanford.edu/class/cs242/readings/backus.pdf
2. CARDELLI, L. and WEGNER, P. On Understanding Types, Data Abstraction, and Polymorphism. In: Computing Surveys, Vol 17 n. 4, pp 471-522, (1985) http://lucacardelli.name/Papers/OnUnderstanding.A4.pdf
3. COLMERAUER, A. y ROUSSEL, P.: The birth of Prolog (1992) http://alain.colmerauer.free.fr/ArchivesPublications/HistoireProlog/19november92.pdf
4. KNUTH, D. E. von Neumann's First Computer Program. In: Computing Surveys, Vol. 2, Nro. 4, pp: 247-260 (1970) http://www.acm.org
5. PARNAS, D.L. On the Criteria To Be Used in Decomposing Systems into Modules. In: Comm. of the ACM, Vol. 15, Num. 12 , pp: 1053-1058 (1972) http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf

Herramientas de trabajo :

- Emulador de Assembler
- Compilador Pascal o C
- Intérprete Lisp
- Intérprete Prolog
- Entorno de trabajo Java

4. Metodología de enseñanza

- **Presentación** de los contenidos de las unidades que integran el programa analítico.
- **Reflexiones y conclusiones** sobre los temas, con participación de los alumnos.
- **Producción monográfica**. Grupos de hasta tres alumnos, con la participación y consejo del docente, seleccionarán un tipo de trabajo monográfico publicado en la Guía de Trabajos Prácticos de la materia. El objetivo del trabajo monográfico es servir de hilo conductor para facilitar la incorporación de los contenidos teóricos que se introducirán a lo largo del curso. En cada clase se tratarán contenidos relacionados con las monografías en curso. Esto permitirá que los alumnos participen activamente a través de preguntas y consultas, por un lado. Por el otro, facilitará la vinculación de los conceptos teóricos con las características de diseño e implementación de lenguajes de distintos paradigmas. La producción monográfica incluirá ejemplos de programación en

los lenguajes incluidos en el estudio.

- **Prácticas** sobre distintos paradigmas. Los alumnos serán introducidos a la codificación en distintos paradigmas, mediante experiencias puntuales.

Se recomienda que los trabajos prácticos ocupen un **45% o más** de la duración de la materia.

5. Evaluación

a. Examen parcial
Los alumnos serán evaluados con respecto a los contenidos presentados en clase mediante a) preguntas sobre contenidos teóricos y la resolución de ejercicios y b) sobre la autoría y dominio del trabajo que están desarrollando mediante preguntas orientadas a relacionar los contenidos de la materia con cada trabajo en particular; también se indagará sobre la participación del alumno en la producción de dicho trabajo.

b. Examen final
El examen final se desarrollará a partir del trabajo que el alumno produjo a lo largo de toda la materia. Se indagará al alumno sobre el dominio que tiene de los conceptos propios de la materia y su capacidad de ponerlos en juego. Se tomará como tema de evaluación el trabajo monográfico realizado a lo largo del curso y se evaluará la capacidad del alumno para relacionar su trabajo con cualquier contenido incluido en el programa analítico.