LENGUAJES, TECNOLOGÍAS Y PARADIGMAS DE PROGRAMACIÓN

Presentación y Normas de la Asignatura Curso 2020-2021

¿Por qué estudiar una asignatura sobre Lenguajes y Paradigmas de programación?

- Existen conceptos comunes a muchos lenguajes aunque se expresen de forma diferente en cada uno de ellos.
 - instrucciones condicionales
 - Iteración
 - estructuras de datos ...
- Existen características que identifican familias de lenguajes
 - concurrencia
 - herencia ...

Aprender estos conceptos nos permitirá

¿Por qué estudiar una asignatura sobre Lenguajes y Paradigmas de programación?

- Facilitar el aprendizaje de un nuevo lenguaje.
- Simular características en lenguajes que carecen de ellas.
- Mejorar la habilidad de desarrollar algoritmos eficientes.
- Mejorar el uso del lenguaje de programación disponible.
- Aumentar el vocabulario del programador.
- Facilitar el diseño de un nuevo lenguaje.

¿Por qué estudiar una asignatura sobre Lenguajes y Paradigmas de programación?

Entender el diseño e implementación de los lenguajes.

- sintaxis las reglas de construcción de los programas
- semántica el significado de los programas
- implementación cómo se ejecutan
- pragmática los aspectos prácticos de su uso

Elegir el lenguaje más apropiado para una aplicación dada.

- Imperativo: Cobol/Fortran Ej: Aplicaciones de gestión/científicas
- OO: C++/Ada/Java Ej: Programación de sistemas
- Funcional: Haskell/LISP/ML Ej: Aplicaciones simbólicas
- Lógico: Prolog/Mercury ... Ej: Aplicaciones de toma de decisiones

Objetivo general

El objetivo del curso es introducir los conceptos fundamentales que podemos encontrar en los lenguajes de programación, presentar las características y las principales aplicaciones de los paradigmas clave en los que se enmarcan los lenguajes de programación, y las tecnologías de soporte.

Aprender principios generales de los lenguajes de programación ilustrados en diversos lenguajes

Competencias

Al finalizar el curso, seréis capaces de:

- <u>diferenciar</u> las características propias de los principales <u>paradigmas</u>
- resolver un mismo problema en diferentes estilos y escribir programas equivalentes en los distintos lenguajes
- saber elegir el lenguaje más apropiado para cada aplicación y <u>usar las técnicas más eficientes</u> para programar en ese lenguaje

Relación con otras asignaturas

- (2°A) TAL: Definición sintáctica de un lenguaje de programación
- (2°B) EDA: Técnicas de programación
- (2°B) Concurrencia y Sistemas Distribuidos: Programación concurrente
- (3°A) Ingeniería del Software: Programación OO
- Rama de Computación
 - (4°A) Lenguajes de Programación y Procesadores de Lenguajes: Modelos de Cómputo, Teoría de Lenguajes de Programación, Herramientas de procesamiento
- Rama de Ingeniería de Computadores
 - (4°A) Lenguajes y Entornos de Programación Paralela
- Rama de Ingeniería del Software
 - (3°B) Métodos formales industriales: Herramientas y técnicas de verificación y certificación.
 - (4°A) Análisis, depuración y validación de software: Herramientas y técnicas de análisis, depuración y validación

LTP: Contenidos

Tema 1: Introducción (2.5 semanas)

- Motivación
- Conceptos básicos (tipos, polimorfismo, reflexión,...).
- Principales modelos de programación: imperativo, funcional, lógico, OO, concurrente.
- Otros paradigmas. Basado en interacción, emergentes.

Tema 2: Fundamentos de los lenguajes de programación (3 semanas)

- Sintaxis y semántica estática de los lenguajes de programación.
- Semántica dinámica de los lenguajes de programación. Estilos de definición semántica.
 Semántica operacional. Semántica axiomática.
- Propiedades semánticas: Corrección, Completitud, Equivalencia. Especificación versus programación.
- Implementación de los lenguajes de programación: máquinas virtuales y lenguajes intermedios.

LTP: Contenidos

Tema 3: Paradigma funcional (4.5 semanas)

- Breve introducción a la notación funcional
- Tipos en Programación Funcional
 - Tipos e inferencia de tipos.
 - Sistemas de tipos (predefinidos, funcionales, algebraicos y parametrizados).
- Tipos de polimorfismo: genericidad, coerción y sobrecarga.
- Modelo de cómputo (reducción y evaluación).
- Características avanzadas
 - Funciones anónimas y composición de funciones.
 - Iteradores y compresores (foldr).

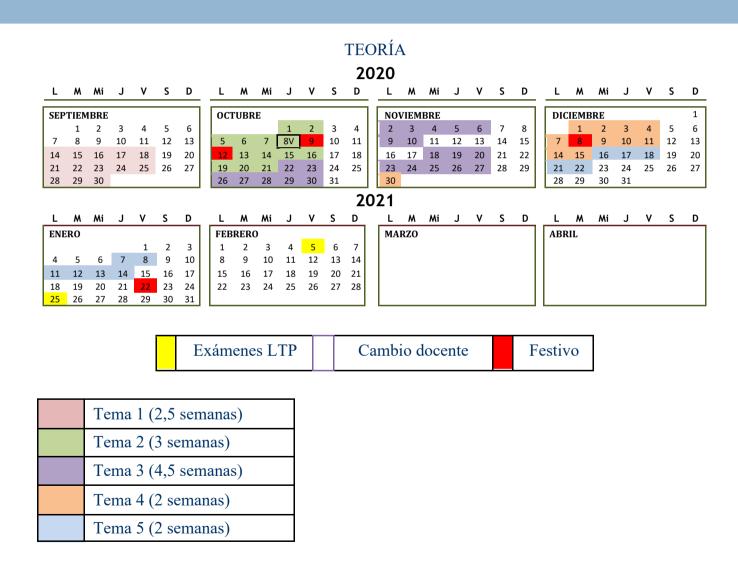
Tema 4: Paradigma lógico (2 semanas)

- Paradigma Lógico: variables lógicas y unificación.
- Modelo de cómputo (resolución SLD)

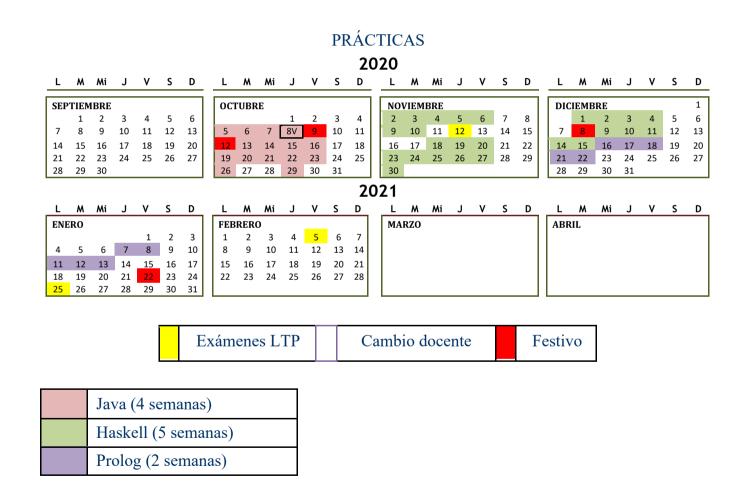
Tema 5: Tecnologías y herramientas de soporte (2 semanas)

Depuración y validación de programas.

LTP: Planificación Teoría



LTP: Planificación Laboratorio



LTP: Evaluación

- □ **Teoría (7 puntos)**: Consta de dos partes:
 - Parte 1 (Temas 1 y 2, 40% de la nota de teoría)
 - Se evaluará con dos exámenes en el aula (uno de cada tema)
 - La fecha de los exámenes será anunciada por el profesor de cada grupo
 - Parte 2 (Temas 3, 4 y 5, 60% de la nota de teoría)
 - Se evaluará con un exámen tipo test común para todos los alumnos
 - El examen se realizará el 25 de enero

Habrá un examen de **recuperación** de cada parte el <u>5 de febrero</u>. Hay que sacar un mínimo de 4 puntos sobre 10 en cada parte para mediar con el resto de notas.25

LTP: Evaluación

- Laboratorio (3 puntos): Hay que sacar un mínimo de 4 puntos sobre 10 en esta parte para mediar con el resto de notas. Habrá dos exámenes parciales:
 - 12 de noviembre (examen online, mediante PoliformaT/Teams): Parte I (Java). VALORACIÓN: 1 punto
 - 25 de enero (coincidiendo con el parcial de teoría, en el aula): Partes II y III (Haskell y Prolog). VALORACIÓN: 2 puntos

Recuperación de las prácticas: habrá una recuperación de cada uno de los parciales el <u>5 de febrero</u>.

Material para la asignatura

- Todo el material de la asignatura (transparencias, boletines de ejercicios, actividades no presenciales, ...) estará accesible en PoliformaT.
- Las referencias bibliográficas generales están disponibles en la biblioteca del Centro y/o en la de la Universidad.