Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

PROGRAMA ANALÍTICO DE ASIGNATURA

DEPARTAMENTO: Ingeniería en Sistemas de Información

CARRERA: Ingeniería en Sistemas de Información

NOMBRE DE LA ACTIVIDAD CURRICULAR: Paradigmas de Programación

Año Académico: Plan 2023
Área: Programación
Bloque: Tecnologías Básicas
Nivel: 2º
Tipo: Obligatoria
Modalidad: Cuatrimestral

Cargas horarias totales:

Horas reloj	*Horas cátedra*	*Horas cátedra semanales*
96	128	8

OBJETIVOS

- Reconocer los fundamentos de los paradigmas de programación asociados a lenguajes de programación concretos para resolver distintas situaciones problemáticas.
- Identificar criterios en la selección del paradigma de programación a utilizar para cada caso concreto.
- Comprender los fundamentos de los paradigmas de programación asociados a lenguajes de programación concretos.
- Aplicar los diferentes paradigmas en la resolución de problemas.
- Adquirir criterios para la selección del paradigma de programación a utilizar en un caso concreto.

CONTENIDOS

Contenidos mínimos

- Concepto de Paradigmas de Programación
- Paradigma Funcional
- Lenguajes de Programación Funcional
- Paradigma Lógico
- Lenguaje de Programación Lógica

- Paradigma Orientado a Objetos
- Lenguajes de Programación Orientados a Objetos

Contenidos analíticos

Unidad 1: Paradigmas de Programación

Concepto de paradigma de programación. Necesidad de la existencia de diferentes paradigmas de programación. Concepto de programa: definiciones generales y específicas. Diferencia entre lenguaje y paradigma de programación. Concepto de tipo: representación de los tipos en los diferentes lenguajes de programación. Importancia del concepto de tipo en relación a la implementación de sistemas complejos y cambiantes. Comparación de los diferentes esquemas de chequeo de tipos. Ubicación de los mecanismos de control de flujo en un programa. Declaratividad: importancia de la separación del control de flujo de la lógica del dominio a modelar. Abstracción y popularización: definición y mecanismos de implementación. Orden superior: concepto e implicancias en el desarrollo de programas. Utilización de las variantes del polimorfismo en los diferentes paradigmas. Comparación entre los diferentes paradigmas de programación.

Logros pedagógicos

- Aprender el concepto de paradigma.
- Identificar claramente la diferencia conceptual entre cada uno de los paradigmas.
- Comprender las aplicaciones de cada uno de ellos.
- Profundizar el concepto de tipo y variable ya visto.
- Conceptualizar al concepto de abstracción.
- Implementar los conceptos de abstracción.

Unidad 2: Paradigma de Objetos

Concepto de Objeto. Concepto de mensaje, estado y comportamiento. Encapsulamiento. Visión de programa entendido como un conjunto de objetos que envían mensajes. Ambientes de objetos: diferencia con la programación tradicional. Los métodos como mecanismo de resolución de mensajes. Concepto de polimorfismo. Concepto de Clase como modelo/molde de objetos. Delegación y responsabilidad. Concepto de referencia. Interfaz e implementación: encapsulamiento del estado interno, ocultamiento de datos. Tipos de mensaje. Herencia. Variables y métodos de clase. Igualdad e identidad. Relaciones entre clases: asociación, composición; relación con delegación. Aplicación del concepto de tipo en el paradigma de objetos. Efecto de lado y declaratividad en el paradigma de objetos. Concepto de orden superior en la programación orientada a objetos. Introducción al manejo de errores. Lenguaje asociado: Smalltalk. Imagen, ambiente de objetos, definición y uso de clases y objetos. Herramientas de navegación (object browser, class browser, otros). Uso de workspaces. Estudio algunas clases propias de Smalltalk: String, Integer, Date, otras. Estudio del protocolo de colecciones. Bloques. Garbage collection.

Logros pedagógicos

- Identificar y comprender el Paradigma Orientado a Objetos.
- Comprender las características y propiedades de los objetos.
- Comparación en el análisis y diseño de la solución con otros paradigmas.
- Práctica y aplicaciones de este paradigma.

Unidad 3: Paradigma Funcional

Concepto de función. La función como bloque de construcción de programas. Concepto de programa en el paradigma funcional. Efecto de lado. Concepto de variable. Definición de tipo y valor. Definición de funciones. Funciones definidas por ramas. Pattern matching. Inferencia de tipos. Funciones recursivas. Prueba por inducción. Manejo de listas. Listas por comprensión. Funciones de orden superior. Currificación y aplicación parcial de funciones. Evaluación diferida y listas infinitas. Composición de funciones. Sistemas de tipos. Polimorfismo y tipos genéricos. Tuplas. Expresiones lambda.

Lenguaje asociado: Haskell. Entorno de trabajo, definición de programas, uso del intérprete. Notación bidimensional. Módulos. Notación de listas [n..m]. Notación de listas por comprensión. Operadores infijos y prefijos. Reglas de precedencia. Prelude de Haskell. Funciones incorporadas en el prelude para manejo de listas, de tuplas, de funciones de orden superior.

Logros pedagógicos

- Identificar y comprender el Paradigma Funcional.
- Comprender el concepto de Recursividad y su utilización.
- Práctica y aplicaciones de este paradigma.

Unidad 4: Paradigma Lógico

Fundamentación lógica. Predicados. Razonamientos y silogismos. Relaciones, hechos y reglas. Consultas. Tipos de consultas. Definición de programa en Paradigma Lógico. Motor de inferencia, ubicación del control en un programa lógico. Diferencia entre una función y una relación. Concepto de variable o incógnita. Unificación. Múltiples resultados. Inversibilidad. Aritmética, evaluación de expresiones aritméticas. Negación. Listas. Pattern Matching. Predicados de orden superior. Functores. Polimorfismo.

Lenguaje asociado: Prolog. Entorno de trabajo, manejo de archivos. Realización de consultas. Ayuda. Trace y debug. Limitaciones de inversiblidad: generación de valores.

Logros pedagógicos

- Identificar y comprender el Paradigma Lógico.
- Comprender la inferencia y los conceptos axiomáticos.
- Práctica y aplicaciones de este paradigma.

BIBLIOGRAFÍA OBLIGATORIA

- Chitta, B. y Gelfond, M. (2002). Logic Programming and Knowledge Representation. Ed. University of Texas.
- Hal, D. (2017). Yet Another Haskell. Ed. Prentice Hall.
- Hudak, P. Peterson, J. y Fasel, J. (2019). Haskell's semantics. Ed. Prentice Hall.
- Lipovača, M. (2018). Learn You a Haskell for Great Good!. Ed. Prentice Hall
- Miller, K. (2017). Haskell is not for production and other tale. Ed. Prentice Hall.
- O'Sullivan, B., Don Stewart y Goerzen, J, (2014). Real World Haskell. Ed. O'Reilly Media.
- Peyton-Jones, S. (2017). Escape from the ivory tower: the Haskell journey. Ed. Prentice Hall.
- Van Roy, P. y Haridi, S. (2003). Concepts, Techniques, and Models of Computer Programming. Ed. The MIT Press.

BIBLIOGRAFÍA COMPLEMENTARIA

- Watt, D. (1990). Programming Languages Concepts and Paradigms. Ed. Prentice Hall.
- Wilkerson, B. y Wiener, L. (1990). Designing Object-Oriented Software, Wirfs- Brock. Ed. Prentice Hall.