PROGRAMA DE CURSO
LENGUAJES DE PROGRAMACIÓN

A. Antecedentes generales del curso:

Departamento	Ciencias de la Computación					
Nombre del curso	Lenguajes de programación	Código	CC4101	Créditos	6	
Nombre del curso en inglés	*Programming Languages*					
Horas semanales	Docencia	3	Auxiliares	1,5	Trabajo personal	5,5
Carácter del curso	Obligatorio	X		Electivo		
Requisitos	CC3102: Teoría de la Computación					

B. Propósito del curso:

Al término del curso, los y las estudiantes demuestran que manejan conceptos y mecanismos asociados a los lenguajes de programación, su semántica, implementación y aplicaciones. Los y las estudiantes definen incrementalmente lenguajes de programación, con su respectiva sintaxis y semántica, a través de la construcción de un intérprete, estudiando paso a paso distintos mecanismos, como funciones, recursión, estado, y finalmente objetos. Asimismo, identifican mecanismos para extender lenguajes existentes, en particular macros. Como la definición de los lenguajes (construcción de intérpretes) se hace en *Scheme*, los y las estudiantes manejan este lenguaje y la programación funcional en general.

Además, el curso entrega elementos precisos para comparar, clasificar, relacionar y evaluar lenguajes de programación en general, incluyendo tanto lenguajes ya existentes (C, Java, Lisp, bash, Haskell, ML, JavaScript, Scala, Self, y Smalltalk, etc.), como lenguajes que emergerán durante la vida profesional de los y las estudiantes.

Como la casi totalidad del material bibliográfico está en inglés, los y las estudiantes profundizan, a través de la lectura, su manejo de inglés técnico.

El curso tributa a las siguientes competencias específicas (CE) y genéricas (CG):

CE1: Analizar problemas computacionales, construir modelos, expresándolos en representaciones y lenguajes formales adecuados.

CE5: Concebir, diseñar y construir soluciones de software, siguiendo un proceso sistemático y cuantificable, acorde a los fundamentos, eligiendo el paradigma y las técnicas más adecuadas.

CE6: Desarrollar software en una amplia variedad de plataformas y lenguajes de programación.

CG2: Comunicación en inglés

Leer y escuchar de manera comprensiva en inglés una variedad de textos e informaciones sobre temas concretos o abstractos, comunicando experiencias y opiniones, adecuándose a diferentes contextos y a las características de la audiencia.

CG3: Compromiso ético Actuar de manera responsable y honesta, dando cuenta en forma crítica de sus propias acciones y sus consecuencias, en el marco del respeto hacia la dignidad de las personas y el cuidado del medio social, cultural y natural.

C. Resultados de aprendizaje:

Competencias específicas	Resultados de aprendizaje
CE1, CE5, CE6	RA1: Define, en forma incremental, lenguajes de programación, considerando el uso de intérpretes como mecanismo de especificación formal para la sintaxis y semántica de los lenguajes.
CE5, CE6	RA2: Utiliza distintos mecanismos de los lenguajes de programación tales como funciones, recursión, estado, y finalmente objetos, prediciendo el comportamiento de programas según las variantes adoptadas por el lenguaje.
	RA3: Compara y clasifica, según sus ventajas, lenguajes de programación emergentes y tradicionales, tales como C, Java, Lisp, Bash, Haskell, ML, JavaScript, Scala, Self, y Smalltalk.
CE1, CE6	RA4: Diseña e implementa intérpretes de lenguajes de programación en Scheme, utilizando principios básicos de programación funcional, tales como, funciones de primera clase, abstracciones de datos, recursión y pattern matching.
Competencias genéricas	**Resultados de aprendizaje**
CG2	RA5: Lee diversos textos o documentación especializada en inglés para extraer y utilizar información acerca de conceptos técnicos y uso de herramientas de los lenguajes de programación.
CG3	RA6: Trabaja en la ejecución de sus tareas y actividades académicas, actuando con responsabilidad y honestidad en su proceder.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA4	Elementos básicos de programación funcional en Scheme	1,5 semanas
Contenidos		**Indicador de logro**	
1.1. Programación básica en el lenguaje Scheme. 1.2. Funciones de primera clase. 1.3. Pattern matching. 1.4. Procesamiento de estructuras recursivas.		El/la estudiante: 1. Identifica los elementos básicos de programación funcional. 2. Escribe programas en Scheme, utilizando conceptos fundamentales de la programación funcional.	
Bibliografía de la unidad		[2].	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
2	RA1	Anatomía de un lenguaje de programación	0,5 semanas
Contenidos		**Indicador de logro**	
2.1. Sintaxis: concreta v/s abstracta; parsing. 2.2. Semántica estática, tipos. 2.3. Semántica dinámica. 2.4. Interpretación v/s compilación.		El/la estudiante: 1. Reconoce la diferencia entre sintaxis y semántica en el contexto de lenguajes de programación. 2. Identifica y analiza las distintas fases de procesamiento para ejecutar un programa.	
Bibliografía de la unidad		[1] Cap 1.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA1, RA2, RA3, RA4, RA5, RA6	Sustitución y funciones de primer orden	2 semanas
Contenidos		**Indicador de logro**	
3.1. Interpretación de un lenguaje de cálculo aritmético elemental. 3.2. Introducción de identificadores en el lenguaje. 3.3. Sustitución: definición del significado de los identificadores. 3.4. Sustitución temprana y perezosa 3.5. Funciones de primer orden.		El/la estudiante: 1. Escribe un intérprete para un lenguaje de cálculo aritmético básico. 2. Define una función sustitución para un lenguaje de cálculo aritmético básico que incorpora definiciones locales y funciones de primer orden. 3. Distingue y utiliza posibles variaciones de sustitución (temprana o perezosa), sopesando sus limitaciones y beneficios. 4. Lee diversos textos técnicos en inglés para extraer conceptos disciplinares aplicables sobre sustitución y funciones de primer orden.	

Bibliografía de la unidad	[1] Cap 2-5.

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
4	RA1, RA2, RA3, RA4, RA5, RA6	Funciones de primera clase y alcance	1,5 semanas
Contenidos		**Indicador de logro**	
4.1. Introducción de ambientes para manejar identificadores 4.2. Control del alcance de las variables: estático vs. dinámico. 4.3. Funciones de primera clase. 4.4. Funciones anónimas. 4.5. Clausuras vs. punteros de función.		El/la estudiante: 1. Usa ambientes para definir intérpretes. 2. Compara beneficios y limitaciones del alcance estático y dinámico de los lenguajes de programación. 3. Utiliza clausuras para garantizar el alcance estático. 4. Compara las ventajas respectivas de clausuras (Python, Scheme, etc.) y punteros de función (C). 5. Implementa un intérprete para funciones de primera clase con alcance estático. 6. Lee diversos textos técnicos en inglés para extraer y utilizar conceptos sobre funciones de primera y segunda clase.	
Bibliografía de la unidad		[1] Cap 5-6. [3].	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
5	RA1, RA2, RA3, RA4, RA5, RA6	Estrategias de evaluación de programas	2 semanas
Contenidos		**Indicador de logro**	
5.1. Introducción al lenguaje Haskell. 5.2. Evaluación perezosa vs. evaluación temprana. 5.3. Ventajas y oportunidades de la evaluación perezosa: eficiencia y expresividad (manejo de estructuras de datos infinitas). 5.4. Transparencia referencial y razonamiento ecuacional.		El/la estudiante: 1. Identifica los beneficios de distintas estrategias de evaluación, reconociendo los escenarios de aplicación de cada uno. 2. Implementa un intérprete para un lenguaje con evaluación perezosa, dada su eficiencia y expresividad. 3. Determina bajo qué condiciones técnicas de *memoization* y *caching* son aplicables a un programa. 4. Lee diversos textos en inglés para extraer y utilizar conceptos de estrategias de evaluación de programas.	
Bibliografía de la unidad		[1] Cap 7-8.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
6	RA1, RA2, RA3, RA5, RA6	Recursión	2 semanas
Contenidos		Indicador de logro	
6.1. Funciones recursivas de primera clase. 6.2. Ambiente recursivo mediante mutación y punto fijo. 6.3. Recursión e iteración: recursión por la cola y optimizaciones. 6.4. Definición funcional de la recursión.		El/la estudiante: 1. Determina las ventajas de las llamadas a función por la cola, especialmente, funciones recursivas por la cola. 2. Implementa un intérprete de lenguajes con funciones recursivas, usando ambientes recursivos y mutación. 3. Analiza y evalúa el enfoque puramente funcional para implementar funciones recursivas. 4. Lee diversos textos en inglés para extraer y utilizar conceptos propios de lenguaje de programación sobre recursión.	
Bibliografía de la unidad		[1] Cap 9-10 [4].	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
7	RA1, RA2, RA3, RA4, RA5, RA6	Representación procedural y meta-interpretación	0,5 semanas
Contenidos		Indicador de logro	
7.1. Uso de funciones en vez de estructura de datos: abstracción procedural. 7.2. Aplicación a ambientes y definición de intérpretes. 7.3. Intérprete sintáctico, intérprete meta.		El/la estudiante: 1. Determina la relación entre abstracción de datos y abstracción procedural. 2. Escribe intérpretes usando una representación procedural de ambientes. 3. Asocia los distintos tipos de interpretación para cada mecanismo de lenguaje: sintáctica y meta, considerando sus beneficios y limitaciones. 4. Escribe intérpretes sintácticos, meta e híbridos. 5. Lee diversos textos técnicos en inglés para usar conceptos de representación procedural y meta-interpretación.	
Bibliografía de la unidad		[1] Cap 11.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
8	RA1, RA2, RA3, RA4, RA5, RA6	Estado y mutación	2 semanas
Contenidos		Indicador de logro	
8.1. Mutación y orden de evaluación. 8.2. Estructuras de datos mutables. 8.3. Variables y asignación. 8.4. Extensión del modelo con el "almacén" (*store*). 8.5. Patrón de hilamiento en el intérprete. 8.6. Estrategias de paso de parámetros (por valor y por referencia).		El/la estudiante: 1. Determina pro y contras de usar estado mutable, considerando cuándo es necesario usarlo y cuándo es preferible evitarlo. 2. Caracteriza diferentes formas de mutación: variables y estructuras de datos mutables. 3. Reconoce y analiza diferentes mecanismos de paso de parámetros (por valor y por referencia). 4. Escribe un intérprete para un lenguaje con diferentes formas de mutación y diferentes formas de paso de parámetros. 5. Lee en inglés, utilizando lo aprendido en un nuevo contexto de aplicación para lenguajes de programación.	
Bibliografía de la unidad		[1] Cap 12-14.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
9	RA1, RA2, RA3, RA4, RA5, RA6	Extensión sintáctica de lenguajes	1 semana
Contenidos		Indicador de logro	
9.1. Introducción a macros. 9.2. Macros higiénicas.		El/la estudiante: 1. Determina la ventaja de proveer buenas abstracciones a los programadores, facilitando el desarrollo y comprensión de código. 2. Usa el sistema de macros higiénicas de Scheme para definir abstracciones sintácticas. 3. Lee diversos textos técnicos en inglés para extraer datos e información aplicable sobre extensión sintáctica de lenguajes.	
Bibliografía de la unidad		[1] Cap 35-37.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
10	RA1, RA2, RA3, RA4, RA5, RA6	Objetos	2 semanas
Contenidos		Indicador de logro	
10.1. Abstracción procedural y objetos. 10.2. Objeto como unidad de encapsulación. 10.3. Interfaz de un objeto. 10.4. Recursión y self. 10.5. Delegación y prototipos. 10.6. Clases y herencia. 10.7. Opciones de diseño para OOP.		El/la estudiante: 1. Clasifica y compara distintos mecanismos de lenguajes orientados a objetos, considerando el paso de mensajes, resolución de nombres, entre otros. 2. Implementa distintos mecanismos de programación orientada a objetos, usando tanto macros como intérpretes. 3. Lee diversos textos en inglés para extraer y usar conceptos sobre recursión.	
Bibliografía de la unidad		[5].	

E. Estrategias de enseñanza -aprendizaje:

El curso considera las siguientes estrategias:

- **Clases expositivas:** en sesión el o la profesor(a) combina la explicación teórica de términos clave con una demostración donde se implementan los conceptos vistos, a través de la implementación de intérpretes en vivo (**demostraciones computacionales**).
- **Resolución de problemas:** en clases auxiliares se complementa el proceso de enseñanza aprendizaje al repasar y trabajar con conceptos complejos vistos en clase, analizar ejemplos más extensos, resolver ejercicios propuestos para poner en acción los aprendizajes adquiridos.

F. Estrategias de evaluación:

Al inicio de cada semestre, el académico o académica informará a los y las estudiantes sobre los tipos de evaluaciones, así como las ponderaciones correspondientes.

Para esta propuesta el curso considera las siguientes estrategias de evaluación:

- **Controles (máximo 3):** evalúa de manera integradora los aprendizajes declarados en las unidades informadas al inicio del semestre, usando conceptos fundamentales de los lenguajes de programación y su aplicabilidad para la resolución de problemas específicos.
- **Tareas (máximo 5):** consisten generalmente en diseñar e implementar lenguajes con varios mecanismos. Las tareas buscan fundamentalmente que los y las estudiantes comprenda cómo definir un lenguaje, y cómo/cuándo aplicar los mecanismos ofrecidos por este.
- **Examen (1):** evalúa de forma integradora los aprendizajes declarados para el curso, considerando los conceptos fundamentales de lenguajes de programación como sus implicancias concretas.

G. Recursos bibliográficos:

Bibliografía obligatoria:

[1] Krishnamurthi, S (2007). *Programming Languages: Application and Interpretation.* Online: http://www.cs.brown.edu/~sk/Publications/Books/ProgLangs/.
[2] Tanter, É. PrePLAI: Scheme y Programación Funcional. Online. http://users.dcc.uchile.cl/~etanter/preplai/.
[3] Tanter, É. *A Note on Dynamic Scope*. Online. https://users.dcc.uchile.cl/~etanter/scope/.
[4] Tanter, É. A note on Recursion. Online. https://users.dcc.uchile.cl/~etanter/recursion/.
[5] Tanter, É. *Object Oriented Programming Languages: Application and Interpretation*. Online. http://users.dcc.uchile.cl/~etanter/ooplai/.

Bibliografía complementaria:

[6] Friedman, D., Wand, M., Haynes. C. (2001). Essentials of Programming Languages. MIT Press: 2nd edition.
[7] Friedman, D., Felleisen, M., Sussman, G. (1995). The Little Schemer, MIT Press: 4th edition.
[8] Sitaram, D. (2004). Teach Yourself Scheme in Finum Days. Online: http://www.ccs.neu.edu/home/dorai/t-y-scheme/

H. Datos generales sobre elaboración y vigencia del programa de curso:

Vigencia desde:	Primavera, 2022
Elaborado por:	Éric Tanter, Federico Olmedo
Validado por:	Validador académico par: Jocelyn Simmonds Validación CTD de Computación
Revisado por:	Área de Gestión Curricular