PROGRAMA DE CURSO

PROGRAMACIÓN FUNCIONAL

A. Antecedentes generales del curso:

Departamento	Ciencias de la Computación					
Nombre del curso	Programación funcional					
Nombre del curso en inglés	Functional programming					
Código	CC5115		Créditos	6		
Horas semanales	Docencia	3.0	Auxiliares	1.5	Trabajo personal	5.5
Carácter del curso	Obligatorio	0		Electivo de Especialización	X	
Requisitos	CC3001: Algoritmos y estructura de datos CC3101: Matemáticas discretas para la computación					

B. Propósito del curso:

El propósito de este curso es introducir la *programación funcional* a través del lenguaje Haskell. La programación funcional es reconocida por dar lugar a un estilo mucho más declarativo –y elegante– de programación, y tres características que la hacen particularmente atractiva son su conveniencia para *verificar* programas, razonando algebraicamente de la misma manera que lo hacemos en matemática, para *paralelizar* la ejecución de programas, ganando órdenes de magnitud en eficiencia, y por último, para *abstraer* patrones de programación recurrentes, generando código más compacto, robusto y reusable.

Lenguajes populares como Java, JavaScript o Scala han sabido explotar estas características adoptando un enfoque multi-paradigma. Para los objetivos de este utilizaremos sin embargo Haskell, un lenguaje puramente funcional. Haskell se considera en la frontera del diseño de lenguajes de programación y hoy en día tiene una penetración no menor en la industria, siendo usado por compañías como Microsoft, Facebook, IBM, Galois, AT&T, JaneStreet y la NASA.

Las competencias específicas (CE) y genéricas (CG) a las que tributa el curso son:

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

CE2: Analizar, diseñar y/o adaptar algoritmos y estructuras de datos que cumplan con las garantías requeridas de correctitud y eficiencia.

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.

CG1: Comunicación académica y profesional

Comunicar en español de forma estratégica, clara y eficaz, tanto en modalidad oral como escrita, puntos de vista, propuestas de proyectos y resultados de investigación fundamentados, en situaciones de comunicación compleja, en ambientes sociales, académicos y profesionales.

CG2: Comunicación en inglés

Leer y escuchar de manera comprensiva en inglés variados tipos de textos e informaciones sobre temas concretos o abstractos, comunicando experiencias y opiniones, adecuándose a diferentes contextos de acuerdo 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	RA1: Detecta, encapsula y explota patrones de cómputo recurrentes, a fin de obtener programas más robustos, concisos y reutilizables.
	RA2: Razona formalmente sobre programas, a fin de probar rigurosamente propiedades sobre el comportamiento de los mismos.
CE2	RA3: Diseña soluciones algorítmicas paralelizables, para mejorar las cualidades de eficiencia de los programas.
CE5	RA4: Valida software de manera automática, a través del testing aleatorio, para aumentar la confianza en el software desarrollado.
CE6	RA5: Desarrolla software siguiendo el paradigma funcional, logrando una descripción más declarativa y concisa de los programas.

Competencias genéricas	Resultados de aprendizaje
CG1	RA6: Comunica en forma oral y escrita, sobre distintos aspectos de lenguajes de programación, haciendo uso del lenguaje técnico, para establecer un lenguaje común y colaborar con sus pares en las tareas asignada.
CG2	RA7: Lee en inglés, de manera analítica y comprensiva, artículos científicos y libros de texto sobre lenguajes de programación a fin de generar nuevos conocimientos atingentes y aplicables a la solución computacional de problemas.
CG3	RA8: Cumple obligaciones y acuerdos, respetando los compromisos adquiridos, reflexionando sobre sus acciones y asumiendo las consecuencias.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA5-RA6-RA7-RA8	Introducción	1 semana
Contenidos		**Indicador de logro**	
1.1. Programación funcional y modelo de cómputo subyacente. 1.2. Características fundamentales de Haskell y de su entorno de desarrollo. 1.3. Elementos básicos de Haskell: ● tipos de datos básicos y operadores predefinidos; ● elementos básicos para la defincón de funciones: *pattern matching*, condicionales, guardas, definiciones locales; ● estructuras de datos simples: tuplas y listas; ● definición de listas por comprensión.		El estudiante: 1. Identifica las nociones fundamentales de expresión, tipo y valor, y reconoce a la reducción de expresiones como el modelo de cómputo subyacente de la programación funcional. 2. Identifica las características fundamentales del lenguaje funcional Haskell. 3. Reconoce sus construcciones sintácticas básicas. 4. Escribe programas simples utilizando elementos básicos de Haskell. 5. Escribe programas complejos sobre listas, combinando funciones predefinidas y la definición de listas por comprensión. 6. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 7. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad.	

	8. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[1] Cap. 0 (Introducción), Cap. 1.{1-5}, Cap. 3 [2] Cap. 1, Cap. 2, Cap. 3.{1-5}, Cap. 4 [3] Cap. 2.{1-3}

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
2	RA1-RA5-RA6-RA7-RA8	Conceptos avanzados sobre funciones	1 semana
Contenidos		Indicador de logro	
2.1. Funciones de alto orden. 2.2. Funciones anónimas. 2.3. Currificación. 2.4. Estilo libre-de-punto (*point-free*) y estilo punto-a-punto (*point-wise*)		El estudiante: 1. Abstrae diversos procesos de cómputo a través de funciones de alto orden. 2. Utiliza λ-expresiones para definir programas de manera más compacta. 3. Explota la aplicación parcial de funciones currificadas. 4. Define funciones de manera más concisa usando el estilo *point-free*, y/o el operador de composición. 5. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 6. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 7. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.	
Bibliografía de la unidad		[1] Cap. 5.{1-4,6-7} [2] Cap. 7.{1-2,5-7}	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA4-RA5-RA6-RA7-RA8	Testing automático de programas	1 semana
Contenidos		Indicador de logro	
3.1. Testing de programas basado en propiedades (*property-based testing*). 3.2. Generación aleatoria de *test cases*. 3.3. Librería QuickCheck.		El estudiante: 1. Identifica propiedades relevantes para testear distintos tipos de programas.	

	2. Aplica la técnica de testing aleatorio basado en propiedades para validar de manera automática la corrección de programas. 3. Identifica las limitaciones inherentes a esta técnica. 4. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 5. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[4] Cap. 11 [5] Cap. 14

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
4	RA1-RA2-RA5-RA6-RA7-RA8	Recursión	1.5 semanas
Contenidos		**Indicador de logro**	
4.1. Definición de funciones recursivas, sobre los naturales y listas. 4.2. Recursividad sobre múltiples argumentos y recursividad mutua. 4.3. Definición de funciones recursivas a través de *fold*. 4.4. Pruebas por inducción estructural. 4.5. Leyes de fusión.		El estudiante: 1. Aplica la técnica divide-y-vencerás y la recursión para resolver problemas complejos. 2. Reconoce un criterio que garantiza el buen comportamiento (i.e. la terminación) de definiciones recursivas. 3. Reconoce y captura el patrón de recursión estructural sobre los naturales y las listas. 4. Define funciones recursivas usando dicha abstracción, y explota las oportunidades de optimización que esto provee. 5. Razona ecuacionalmente sobre programas y prueba propiedades sobre funciones recursivas a través del principio de inducción estructural. 6. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 7. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad.	

	8. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[1] Cap. 4, Cap. 5.5 [2] Cap. 6, Cap. 7.{3-4}, Cap. 16.{1-6} [3] Cap. 6.3

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
5	RA2-RA5-RA6-RA7-RA8	Conceptos avanzados sobre tipos	1.5 semanas
Contenidos		Indicador de logro	
5.1. Tipos de datos algebraicos simples (i.e., no recursivos) 5.2. Algunos constructores de tipo relevantes: *Either*, *Maybe*, etc. 5.3. Variables de tipos. 5.4. Clases de tipos.		El estudiante: 1. Explota todo el potencial del sistema de tipos de Haskell para generar código más robusto y reusable. 2. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 3. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 4. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.	
Bibliografía de la unidad		[1] Cap. 2.{3-4}, Cap. 7.{1-6,8-9} [2] Cap. 3.{7,9}, Cap. 8.{1-3,5} [3] Cap. 6.3	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
6	RA1-RA2-RA5-RA6-RA7-RA8	Tipos recursivos	2 semanas
Contenidos		Indicador de logro	
6.1. Estructuras de datos recursivas: definición y manipulación. 6.2. Catamorfismos (o *fold's*). 6.3. Casos de estudio: chequeador de tautologías y el problema de la cuenta regresiva.		El estudiante: 1. Modela estructuras de datos recursivas mediante tipos de datos algebraicos. 2. Captura en una función de alto orden (catamorfismo o *fold*) el patrón recursivo asociado a un tipo recursivo. 3. Define funciones sobre tipos recursivos usando tanto recursión explícita como el catamorfismo asociado.	

	4. Razona ecuacionalmente sobre programas que manipulan tipos recursivos, utilizando el principio de inducción estructural. 5. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 6. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 7. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[1] Cap. 7.7 [2] Cap. 8.{4-6}, Cap. 9 [6]

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
7	RA5-RA6-RA7-RA8	Evaluación perezosa	1 semanas
Contenidos		Indicador de logro	
7.1. Evaluación perezosa. 7.2. Formas normales. 7.3. Estructuras de datos infinitas.		El estudiante: 1. Reconoce de manera más precisa la estrategia usada por Haskell para reducir expresiones. 2. Reconoce el beneficio de esta estrategia en términos de eficiencia. 3. Explota esta estrategia para definir y manipular estructuras de datos infinitas. 4. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 5. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.	
Bibliografía de la unidad		[2] Cap. 15.{1-6} [3] Cap. 7.1 [7]	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
8	RA1-RA5-RA6-RA7-RA8	Eficiencia	1 semana
Contenidos		Indicador de logro	

8.1. Consideraciones sobre *fold* y leyes de fusión. 8.2. Acumuladores. 8.3. Tupling. 8.4. Aplicaciones a algoritmos de ordenación: *mergesort* y *quicksort*.	El estudiante: 1. Reconoce ciertos patrones que menguan la eficiencia de programas y son amenos a optimización. 2. Aplica transformaciones de programas (basadas en leyes de fusión, acumuladores y tupling) destinadas a mejorar sus propiedades de eficiencia. 3. Prueba formalmente que dichas transformaciones preservan la semántica de los programas. 4. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 5. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[3] Cap. 7

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
9	RA1-RA5-RA6-RA7-RA8	Entrada/salida	1 semana
Contenidos		Indicador de logro	
9.1. Interpretación de la entrada/salida como un efecto. 9.2. Mónada de entrada/salida. 9.3. Primitivas monádicas de entrada/salida. 9.4. Casos de estudio: juego del ahorcado y juego de la vida.		El estudiante: 1. Reconoce las limitaciones de los lenguajes funcionales puros para soportar efectos, en particular, entrada/salida. 2. Reconoce el enfoque monádico para modelar entrada/salida y lo utiliza efectivamente para escribir programas interactivos. 3. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 4. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 5. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.	
Bibliografía de la unidad		[1] Cap. 8 [2] Cap. 10	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
10	RA1-RA2-RA5-RA6-RA7-RA8	Mónadas y funtores	2 semanas
Contenidos		Indicador de logro	
10.1. Funtores. 10.2. Funtores aplicativos. 10.3. Mónadas.		El estudiante: 1. Utiliza la abstracción de funtores para capturar un patrón de programación recurrente, y desarrollar software más robusto y reusable. 2. En particular, utiliza la abstracción de mónadas para capturar distintos tipos de efectos dentro de un lenguaje puro. 3. Utiliza la notación aplicativa (resp. la notación *do*) para escribir programas en idioma aplicativo (resp. monádicos) de manera natural y concisa. 4. Explota las leyes algebraicas de los functores para probar equivalencias de programas. 5. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 6. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 7. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.	
Bibliografía de la unidad		[1] Cap. 11, Cap. 13, Cap. 14 [2] Cap. 12	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
11	RA3-RA5-RA6-RA7-RA8	Paralelización	2 semanas
Contenidos		Indicador de logro	
11.1. Primitivas de paralelización. 11.2. Herramientas de profiling.		El estudiante: 1. Diferencia las nociones de concurrencia y paralelismo. 2. Identifica oportunidades de paralelizar programas, y explota el paralelismo eligiendo estructura de datos y diseño de funciones adecuados.	

	3. Utiliza técnicas de profiling para cuantificar los efectos de la paralelización. 4. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 5. Planifica y entrega sus tareas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre programación funcional, determinando sus ideas principales.
Bibliografía de la unidad	[4] Cap. 24. [8]

E. Estrategias de enseñanza:

El curso se estructura en base a distintas metodologías que incluyen principalmente:

- **Clases expositivas** que combinan el uso de diapositivas, la pizarra, y la programación en vivo para presentar el material. Para promover el aprendizaje activo, durante todas las clases se presentarán mini-problemas que los alumnos deben resolver grupalmente, y cuya solución se discutirá luego de manera colectiva.
- **Clases auxiliares** donde se afianzarán los conceptos presentados en las clases expositivas a través de la resolución de diversos tipos de problemas por parte de los estudiantes, y que servirán también de ayuda para la realización de las tareas.
- **Casos de estudio**, donde se desarrollarán soluciones a problemas no-triviales, de la vida real, y cuya solución exitosa requiere de múltiples conceptos desarrollados en clase.

E. Estrategias de evaluación:

Durante el curso se realizará una *evaluación continua* a través de tareas periódicas y una presentación oral, distribuidas de la siguiente manera:

- **5 tareas** que se promedian en partes iguales para obtener la nota de tareas (NT). Las tareas evaluarán tanto aspectos prácticos como teóricos, correspondiendo a los resultados de aprendizaje RA1, RA2, RA3, RA4, RA5, RA6 y RA8.
- **1 presentación oral**, que el alumno deberá dar individualmente sobre un tema relacionado al curso, y que deberá preparar de manera independiente, obteniendo una nota de presentación (NP). Los temas serán propuestos por el profesor de cátedra. Se evalúan los resultados de aprendizaje RA6, RA7 y RA8.
- **1 examen**, en el que obtendrá su la nota de examen (NE). Se evalúan los resultados de aprendizaje RA1, RA2, RA3, RA4, RA5, RA6 y RA8.

La nota final (NF) se calcula ponderando las notas de los tres instancias de evaluación de la siguiente manera:

NF = 0.3 * NE + 0.5 * NT + 0.2 * NP.

Si NT≥5.5 y NP≥5.5 el alumno se exime de rendir el examen, y en ese caso su nota final será:

NF = 0.8 * NT + 0.2 * NP.

F. Recursos bibliográficos:

Bibliografía obligatoria:

1. *Learn you a Haskell for great good: A beginner's guide.* Lipovača M., No Starch Press, 1º Edición, 2011. Disponible online: http://learnyouahaskell.com
2. *Programming in Haskell.* Hutton. G. Cambridge University Press. 2º Edición, 2016. Disponible online: http://www.cs.nott.ac.uk/~pszgmh/pih.html
3. *Thinking functionally with Haskell.* Richard Bird. Cambridge University Press. 1º Edición, 2015.
4. *Real world Haskell: Code you can believe in.* O'Sullivan, B., Goerzen, J. y Stewart, D.B. O'Reilly Media, Inc. 1º Edición, 2008. Disponible online: http://book.realworldhaskell.org/
5. *Haskell programming from first principles*. Allen C. y Moronuki J. Allen and Moronuki Publishing. 1º Edición, 2016.
6. *What's in a Fold: The basic catamorphism in recursion-schemes.* Daniel Mlot. 2017. Disponible online: https://duplode.github.io/posts/whats-in-a-fold.html
7. *Why functional programming matters.* Hughes J. The Computer Journal, 32:2, 1989. Disponible online: https://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf
8. *Parallel and concurrent programming in Haskell.* Marlow S. Central European Functional Programming School 2011. Springer. 2011. Diapositivas disponibles online: http://www.it.uu.se/edu/course/homepage/avfunpro/ht17/haskell-parallel-marlow.pdf

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

Vigencia desde:	Otoño 2020
Elaborado por:	Federico Olmedo
Validado por:	Sergio Ochoa
Revisado por:	Área de Gestión Curricular