FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA
U.N.R.

PROGRAMA ANALÍTICO DE LA ASIGNATURA: Análisis de Lenguajes de Programación

Código R-313

PLAN DE ESTUDIOS: 2010
CARRERA: Lic. en Cs. de la Computación
DEPARTAMENTO: Cs. de la Computación
PROFESOR: Mauro J. Jaskelioff

2011 **HASTA AÑO**

~~TENTATIVO~~ **DEFINITIVO** **DE EXAMEN**

PROGRAMA

~~ANUAL~~ ~~SEMESTRAL~~ **CUATRIMESTRAL**

Táchese lo que no corresponda.

OBSERVACIONES:

PRESUPUESTO HORARIO SEMANAL PROMEDIO

TEORÍA:	**3**	**1**
PRÁCTICA:	**3**	**2**
LABORATORIO:	**2**	**3**
TOTAL ASIGNADO:	**8**	**4** **1+2+3**
DEDICACIÓN DEL ALUMNO FUERA DE CLASE:	**4**	**5**
PRESUPUESTO TOTAL:	**12**	**6** **5+4**
PROGRAMA BASADO EN SEMANAS ÚTILES :	**15**	**7**
HORAS TOTALES ASIGNADAS:	**120**	**7x4**
HORAS TOTALES PRESUPUESTAS:	**180**	**7x6**

OBJETIVOS: (qué debe saber el alumno al concluir el curso)

1. Comprender el concepto de programación como actividad rigurosa de abstracción, y como se comparan los diferentes paradigmas de programación.
2. Explorar y utilizar con seguridad los conceptos del paradigma funcional.
3. Conocer las principales técnicas formales de desarrollo de programas, comprendiendo su importancia y utilidad como medio de garantizar la corrección del software producido.
4. Comprender, manejar y familiarizarse con conceptos fundamentales de la programación y su importancia en la tarea de programar:
   - abstracción (en particular las nociones de función y módulo)
   - inducción (y las herramientas asociadas: inducción estructural y recursión)
   - demostración de propiedades
   - transformación de programas

UBICACIÓN EN LA CARRERA Y CARACTERÍSTICAS GENERALES:

La asignatura se ubica en el 1º cuatrimestre de 3º año de la carrera de Lic. en Cs. de la Computación.
Los lenguajes funcionales sirven no sólo como base teórica para el estudio de problemas y técnicas computacionales complejas, sino que hoy día también se están usando a nivel industrial como lenguajes de prototipado y en algunos casos, de producción. Es por ello que una materia que aborde esta temática es fundamental.

MATERIAS RELACIONADAS:

Previas: R-223 Lógica.
Simultáneas recomendadas: - - -
Posteriores: R-322 Sistemas Operativos I; R-422 Compiladores.

.....	….....	.....	….....
Firma Profesor	**Fecha**	**Aprob. Escuela**	**Fecha**

Aprobado en reunión de Consejo Académico de fecha: .....

CONTENIDO TEMÁTICO

Ordenar temas utilizando codificación decimal

1. Conceptos Preliminares
   1.1. Revisión de la noción de programación y el concepto de programa.
   1.2. Propiedades deseables de los programas. Razonamiento y demostración de dichas propiedades.
   1.3. Dificultades del modelo clásico de programación para el razonamiento sobre programas.
   1.4. Descripción del modelo de programación funcional.
   1.5. Características principales de los lenguajes funcionales: transparencia referencial, alto orden, currificación y sistemas de tipos.
   1.6. Comparación de paradigmas: imperativo, orientado a objetos, lógico y funcional.

2. Modelo de Computación del Paradigma Funcional
   2.1. Valores y expresiones. Las funciones como valores.
   2.2. Mecanismos de definición de expresiones y valores. Ecuaciones orientadas para definir funciones. Sintaxis.
   2.3. Visión denotacional y operacional de las expresiones. Modelos de computación mediante reducción. Semántica.
   2.4. Ordenes de reducción: reducción aplicativa y reducción normal.
   2.5. Sistema de Tipos Hindley-Milner. Tipos básicos. Constructores de tipos. Polimorfismo. Sintaxis para valores de cada tipo (caracteres, tuplas, listas, strings, funciones)
   2.6. Funciones parciales y totales.
   2.7. Funciones de alto orden. Currificación.
   2.8. Modelo de cómputo. Computación. Reducción. Normalización.
   2.9. Orden de evaluación. Evaluación *Lazy.*

3. Técnicas Formales
   3.1. Demostración de propiedades
   3.2. Noción de propiedad y de demostración. Diferentes formas de garantizar propiedades: por construcción, por chequeo automático, por demostración manual.
   3.3. Algunas propiedades interesantes de los programas: corrección, terminación, equivalencia de programas.
   3.4. Inducción/Recursión.
   3.5. Definición inductiva de conjuntos. Relaciones bien fundadas.
   3.6. Definición recursiva de funciones sobre esos conjuntos.
   3.7. Demostraciones inductivas sobre dichas funciones.
   3.8. Ejemplos: programas, expresiones aritméticas, listas.

4. Aplicación de Conceptos: Listas
   4.1. Listas por comprensión. Definición y ejemplos. Semántica de listas por comprensión mediante reducción.
   4.2. Listas como tipo inductivo. Funciones básicas sobre listas (append, head, tail, take, drop, reverse, sort, elem, etc.).
   4.3. Funciones de alto orden sobre listas. Patrón de recorrido: map. Patrón de selección: filter. Patrón de recursión: foldr.
   4.4. Demostración de propiedades sobre listas y funciones sobre listas.

5. Sistemas de tipos.
   5.1. Nociones básicas. Sistemas de tipado fuerte. Ventajas y limitaciones de los lenguajes de programación con tipos.
   5.2. Lenguaje de tipos. Asignación de tipos a expresiones. Propiedades interesantes de esta asignación. Algoritmo de inferencia.
   5.3. Mecanismos de definición de tipos nuevos y de funciones sobre ellos. Tipos algebraicos. Ejemplos: enumeraciones, listas, árboles binarios, árboles generales.
   5.4. Clases de tipos e instancias.
   5.5. Esquemas de alto orden. Programas genéricos.

6. Técnicas de Diseño Funcional.
   6.1. Transformación y Síntesis de Programas .
   6.2. Motivación. Obtención de programas a partir de especificaciones. Mejoramiento de eficiencia, con

corrección por construcción.

6.3. Transformación de expresiones que utilizan listas por comprensión en expresiones que utilizan map, filter y concat.
6.4. Transformación y síntesis de programas. Técnicas y ejemplos.

7. Lambda Cálculo
   7.1. Definición del lenguaje. Sintaxis. Definición de substitución.
   7.2. Modelo de computación. Nociones de alfa, beta y eta reducción. Semántica operacional.
   7.3. Lambda cálculo como modelo teórico de los lenguajes funcionales. Representación de booleanos, pares, números, listas, y otras construcciones.

8. Parsers
   8.1. Definición.
   8.2. Parsers básicos.
   8.3. Parsers compuestos.
   8.4. Ejemplo de parser para un lenguaje simple.

9. Mónadas
   9.1. Motivación.
   9.2. Definición.
   9.3. Entrada/Salida en Haskell.
   9.4. Ejemplos de mónadas comunes: error, reader, writer, state transformer. Listas como mónadas. Mónada de términos.
   9.5. Uso de mónadas en la implementación de lenguajes embebidos.
   9.6. Escribiendo mónadas en Haskell. Notación **do**.

TRABAJOS PRÁCTICOS

a) Enumeración:

Ejercicios de práctica sobre los siguientes temas:

1. Introducción a la sintaxis de Haskell y al ambiente GHCi. Expresiones y valores. Tipos. Notación Lambda. Currificación.
2. Alto orden. Reducción. Órdenes de evaluación. Tipos algebraicos. Pattern matching. Tipos algebraicos recursivos. Listas. Árboles.
3. Demostraciones. Propiedades de programas: terminación, equivalencia. Inducción. Recursión. Relaciones bien fundadas.
4. Funciones de alto orden. Patrones genéricos de recursión. Funciones sobre árboles.
5. Derivación de programas.
6. Lambda cálculo. Programación con λ-cálculo. Sustitución. α-equivalencia. B-reducción.
7. Esquemas de Recursión Genéricos. Representación de datos en λ-cálculo.
8. Evaluación Peresoza. Tipos de datos infinitos.
9. Parsers.
10. Entrada/Salida.
11. Mónadas.

Trabajos prácticos:

1. **Trabajo práctico nº 1**
   Implementación de un lenguaje imperativo simple a partir de su especificación. Sintaxis abstracta. Sintaxis concreta. Semántica denotacional para expresiones. Semántica operacional estructural para comandos.

2. **Trabajo práctico nº 2**
   Implementación de un intérprete interactivo para λ-cálculo sin tipos. Sintaxis abstracta. Sintaxis concreta. Variables libres y variables ligadas. Notación de De Bruijn. Máquina abstracta de Krivine.

3. **Trabajo práctico nº 3**
   Proyecto de programación.

b) Guías de trabajos prácticos publicadas: (con su código de publicación)

BIBLIOGRAFÍA

a) Adecuada al programa. Ordenada por temas y con su codificación de biblioteca, incluidas las publicaciones de la Cátedra con su código de publicación.

1. R. S. Bird.
   Introduction to Functional Programming using Haskell.
   Prentice-Hall, 1998.
2. G. Hutton.
   Programming in Haskell Theories of Programming Languages.
   Cambridge University Press, 2007.
3. S. L. Peyton Jones, John Hughes, et al.
   Report on the programming language Haskell'98.
   Technical report, Yale University, February 1999.
   http://www.haskell.org/onlinereport
4. J. R. Hindley and J. P. Seldin.
   Lambda-Calculus and Combinators.
   Cambridge University Press 2008.

b) Complementaria para profundización o extensión de temas.

1. Sistemas inductivos. Ordenes parciales completos (CPOs).
2. Tipos de datos abstractos generalizados.
3. Lambda cálculos con tipos y con sustituciones explícitas.