

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

**PROGRAMA ANALÍTICO DE LA ASIGNATURA:** Análisis de Leng. de Programación II Código T-321

<p><b>PLAN DE ESTUDIOS:</b> 1993  <b>CARRERA:</b> Lic. en Cs. de la Computación  <b>DEPARTAMENTO:</b> Cs. de la Computación  <b>PROFESOR:</b> Pablo E. Martínez López</p> <p style="text-align: center;">2006    HASTA AÑO</p> <p style="text-align: center;">TENTATIVO    DEFINITIVO    DE EXAMEN</p> <p><b>PROGRAMA</b></p> <p style="text-align: center;">ANUAL    SEMESTRAL    TRIMESTRAL</p> <p style="text-align: center;">Táchese lo que no corresponda.</p> <p><b><u>OBSERVACIONES:</u></b></p>	<p><b>PRESUPUESTO HORARIO SEMANAL PROMEDIO</b></p> <table style="width: 100%; border-collapse: collapse;"> <tr><td>TEORÍA: 3</td><td style="text-align: right;">1</td></tr> <tr><td>PRÁCTICA: 3</td><td style="text-align: right;">2</td></tr> <tr><td>LABORATORIO: 2</td><td style="text-align: right;">3</td></tr> <tr><td>TOTAL ASIGNADO: 8</td><td style="text-align: right;">4</td></tr> <tr><td>DEDICACIÓN DEL ALUMNO FUERA DE CLASE: 4</td><td style="text-align: right;">1+2+3</td></tr> <tr><td>PRESUPUESTO TOTAL: 12</td><td style="text-align: right;">6</td></tr> <tr><td>PROGRAMA BASADO EN SEMANAS ÚTILES : 15</td><td style="text-align: right;">5+4</td></tr> <tr><td>HORAS TOTALES ASIGNADAS: 120</td><td style="text-align: right;">7x4</td></tr> <tr><td>HORAS TOTALES PRESUPUESTAS: 180</td><td style="text-align: right;">7x6</td></tr> </table>	TEORÍA: 3	1	PRÁCTICA: 3	2	LABORATORIO: 2	3	TOTAL ASIGNADO: 8	4	DEDICACIÓN DEL ALUMNO FUERA DE CLASE: 4	1+2+3	PRESUPUESTO TOTAL: 12	6	PROGRAMA BASADO EN SEMANAS ÚTILES : 15	5+4	HORAS TOTALES ASIGNADAS: 120	7x4	HORAS TOTALES PRESUPUESTAS: 180	7x6
TEORÍA: 3	1																		
PRÁCTICA: 3	2																		
LABORATORIO: 2	3																		
TOTAL ASIGNADO: 8	4																		
DEDICACIÓN DEL ALUMNO FUERA DE CLASE: 4	1+2+3																		
PRESUPUESTO TOTAL: 12	6																		
PROGRAMA BASADO EN SEMANAS ÚTILES : 15	5+4																		
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, incluyendo el paradigma de programación funcional.
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 2º 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 prototipación y en algunos casos, de producción. Es por ello que una materia que aborde esta temática es fundamental.

**MATERIAS RELACIONADAS:**

**Previas:** T-222 Arquitectura del Computador; T-123 Análisis de Lenguajes de Programación I.  
**Simultáneas recomendadas:** - - -  
**Posteriores:** T-411 Análisis de Sistemas.

.....  
Firma Profesor

1-6-07  
Fecha

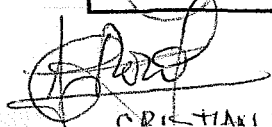
Dr. RAUL E. KANTOR  
DIRECTOR  
Escuela de Ciencias Exactas y Naturales  
FCEIA

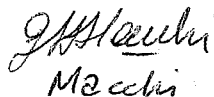
.....  
Aprob. Escuela

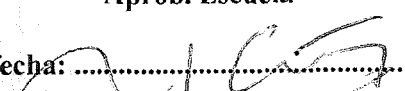
23/8/07  
Fecha

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

.....  
RAUL E. KANTOR  
DIRECTOR  
Escuela de Ciencias Exactas y Naturales  
FCEIA

  
CRISTIAN DORA

  
Pablo E. Martínez López

 5/10/07

# 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.
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.
  - 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. Esquemas de alto orden. Dualidad.
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 sustitución.

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. Mónadas

8.1. Motivación.

8.2. Definición.

8.3. Ejemplos de mónadas comunes: error, reader, writer, state transformer. Listas como mónadas.

8.4. Uso de mónadas. Algoritmos en grafos.

8.5. Escribiendo mónadas en Haskell.

- Clases.

- do-notación.

## TRABAJOS PRÁCTICOS

### a) Enumeración:

#### **Ejercicios de práctica sobre los siguientes temas:**

1. Introducción a la sintaxis de Haskell y al ambiente Hugs. Expresiones y valores. Tipos. Notación Lambda.
2. Currificación. Alto orden. Reducción. Órdenes de evaluación. Demostraciones. Propiedades de programas: terminación, equivalencia. Inducción. Recursión.
3. Tipos algebraicos. Pattern matching. Tipos algebraicos recursivos. Listas. Árboles.
4. Funciones de alto orden. Patrones genéricos de recursión. Funciones sobre árboles.
5. Dualidad.
6. Derivación de programas.
7. Lambda cálculo. Programación con  $\lambda$ -cálculo. Sustitución.  $\alpha$ -equivalencia.  $\beta$ -reducción.
8. 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  $\lambda$ -cálculo sin tipos. Sintaxis abstracta. Sintaxis concreta. Variables libres y variables ligadas. Sustitución. Conversión. Reducción. Igualdad. Estrategias de reducción.

### 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. John C. Reynolds.  
*Theories of Programming Languages.*  
Cambridge University Press, 1998.
3. Benjamin C. Pierce.  
*Types and Programming Languages.*  
The MIT Press, 2002.
4. S. L. Peyton Jones.  
*The implementation of functional programming languages.*  
Prentice-Hall - C.A.R. Hoare Series Editor, 1987.
5. Richard Bird and Oege de Moor.  
*Algebra of Programming.*  
Prentice Hall, 1996.
6. 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>
7. M. J. C. Gordon.  
*Programming language theory and its implementation.*  
Prentice-Hall - C.A.R. Hoare Series Editor, 1988.
8. L.C. Paulson.  
*ML for the working programmer.*  
Cambridge University Press, 1991.
9. A. J. T. Davie.  
*An Introduction to Functional Programming Systems Using Haskell.*  
Cambridge University Press, 1992.

b) Complementaria para profundización o extensión de temas.

1. Sistemas inductivos. Ordenes parciales completos (CPOs).
2. Formalización del sistema de tipos HM monomórfico.
3. Lambda cálculos con tipos y con sustituciones explícitas.