TEC | Tecnológico de Costa Rica

Programa del curso IC-4700

Lenguajes de Programación

Escuela de Computación
Carrera de Ingeniería de Computación, Plan 410.

TEC | Tecnológico de Costa Rica

I parte: Aspectos relativos al plan de estudios

1 Datos generales

Nombre del curso:	Lenguajes de Programación
Código:	IC-4700
Tipo de curso:	Teórico - Práctico
Nº de créditos:	4
Nº horas de clase por semana:	4
Nº horas extraclase por semana:	12
Ubicación en el plan de estudios:	Curso del 4° semestre dela carrera de Ingeniería en Computación
Requisitos:	IC3002 Análisis de Algoritmos. IC3101 Arquitectura de Computadoras
Correquisitos:	Ninguno
El curso es requisito de:	IC5701 Compiladores e Intérpretes
Asistencia:	Obligatoria
Suficiencia:	No
Posibilidad de reconocimiento:	Sí.
Vigencia del programa:	I semestre 2012.

2 Descripción general

Este curso se estudian con detalle los paradigmas fundamentales de los lenguajes de programación. Se aprenden lenguajes representativos de cada paradigma y se estudia la teoría más relevante del diseño de lenguajes.

3 Objetivos

Objetivo General

Estudiar los conceptos fundamentales y los principales principios de diseño de los lenguajes de programación. Estudiar los cuatro paradigmas principales de programación y ofrecer criterios para comparar, evaluar y seleccionar lenguajes de programación en relación con sus diferentes usos.

Objetivos Específicos

- Aprender un lenguaje ejemplo para cada uno de los cuatro paradigmas principales de programación, así como conocer varios otros lenguajes para representar conceptos que no tengan los lenguajes ejemplo.
- Comprender algunos principios generales de diseño de lenguajes.
- Ofrecer criterios para comparar, evaluar y seleccionar lenguajes de programación.
- Dar criterios para discernir cuál es el lenguaje de programación más adecuado para un problema dado.

Introducción y conceptos fundamentales.

4 Contenidos

- Evolución histórica de los lenguajes de programación.
- Principios de diseño de lenguajes de programación.
- Sintaxis, semántica y pragmática.
- Conceptos fundamentales

  Valores, datos y tipos.
- Expresiones.
- Almacenamiento y control.
- Asociación ("binding").
- Abstracción.
- Encapsulación.
- Secuenciadores.

Programación imperativa. (Lenguajes: C o Ada o CLU)

- Modelos de almacenamiento (punteros, arreglos, etc.)
- Mecanismo de paso de parámetros
- Portabilidad
- Preprocesamiento, entrada y salida estándar
- Modularidad
- Generalidad
- Ocultamiento de información
- Independencia de representación
- Referencia a los conceptos fundamentales

Programación funcional. (Lenguaje: preferiblemente Standard ML y Haskell. Otros lenguajes: Lisp, Scheme)

- Expresiones y aritmética
- Funciones como "ciudadanos de primera clase"
- Manejo de listas como primitivas
- Principios de diseño y programación funcional
- Streams (evaluación perezosa)
- Programación funcional con tipos
- Polimorfismo paramétrico

Programación lógica. (Lenguaje: Prolog o Turbo Prolog. Otros lenguajes ilustrativos: Oz o Gödel)

- Relaciones vs funciones
- Hechos y consultas
- Calculo de predicados
- Dominios, datos compuestos y listas.
- Unificación.
- Control de flujo

  Backtracking y orden de descripción.
- Corte y Fail.

Programación orientada a objetos. (Lenguaje Smalltalk, Eiffel, Java. Otros lenguajes ilustrativos: BETA, C++, Object Pascal (Delphi))

- Objetos y Mensajes
- Expresiones y aritmética
- Clases y Métodos
- Instancia y tipos de variables
- Herencia y polimorfismo
- Jerarquía de clases
- Colecciones
- Principios de diseño y programación OO
- Bloques de código y mensajes en cascada
- Estudio analítico de otros lenguajes (lenguaje ilustrativo: Java)

Elementos avanzados de lenguajes de programación

- Concurrencia, paralelismo y distribución.

  (Modelo de concurrencia de Java)

  (Herramientas para programación máquinas multinúcleo)

  (Lenguajes: occaml, POOL, C-Linda, Orca).
- Sistemas de tipos.
- Ligas entre programas de distintos lenguajes.
- Elementos del diseño de lenguajes de programación.
- Evaluación y selección de lenguajes de programación.

II II parte: Aspectos operativos

5 Metodología de enseñanza y aprendizaje

Los estudiantes harán programas no triviales para cada paradigma fundamental (imperativo, orientado a objetos, funcional y lógico)

Para cada uno de los paradigmas fundamentales se usará un lenguaje principal como ejemplo de estudio. Se ilustrarán variante del paradigma con lenguajes afines.

Habrá pruebas cortas en que se evaluará tanto la materia teórica vista en clase como la implementación de las tareas programadas.

6 Evaluación

El curso será evaluado con tres rubros principales: proyectos programados, exámenes parciales y un último rubro que comprende pruebas y tareas cortas.

Proyectos programados (60%): Se asignarán cuatro proyectos programados, uno por cada paradigma estudiado. Se requerirá de un manejo adecuado de los conceptos fundamentales de cada pardigma para poder construir los proyectos exitosamente. Los proyectos se trabajarán individualmente. Las entregas están programadas en las semanas 7, 10, 13 y 16.

Exámenes parciales (20%): Se aplicarán dos exámenes parciales, cada uno correspondiente a 10% del rubro. Los exámenes están programados para en las semanas 14 y 19.

Pruebas y tareas cortas (20%): Se asignarán cuatro tareas cortas programadas, una por cada paradigma de programación estudiado. Adicionalmente se aplicarán pruebas cortas durante el semestre, éstas serán anunciadas con una semana de antelación.

Proyectos programados (4)	60%
Exámenes parciales (2)	20%
Pruebas y tareas cortas	20%

No es posible eximirse de ninguna evaluación del curso.

7 Bibliografía

Obligatoria

Scott, M. "Programming Language Pragmatics" 2da ed, Morgan Kaufmann, 2006.

Kernighan, B., Ritchie, D. "El lenguaje de programación C". Prentice Hall Hispanoamericana, 1985.

Lipovača, M. "Learn you a Haskell for Great Good! A beginners guide" Disponible en: www.learnyouahaskell.com

Blackburn, P.; Bos, J.; Striegnitz K. "Learn Prolog Now!" Disponible en: www.learnprolognow.org

Complementaria

Barendregt, H.P. "The Lambda Calculus: its Syntax and Semantics" North-Holland, Amsterdam, 1985.

Borland International. Turbo Prolog Reference Guide V 2.0. Borland International, 1988.

Camacho, L. "Lógica Simbólica" Editorial Tecnológica, Cartago (última edición).

Copi, I.M. "Lógica Simbólica" EUDEBA, Buenos Aires, 1997.

Digitalk Inc. Smalltalk V: Tutorial and programming Handbook. Digitalk Inc. 1986.

Friedman. "From Babbage to Babel and beyond: A brief history of programming languages". Computer Language 17, 1992. pp. 1-17.

Friedman, D.; Felleisen, M. "The Little Schemer", IV Ed, Science Research Associates Inc, SRA Inc, The MIT Press, 1999.

Helo, José. "Introducción a la Programación con Scheme", Editorial Tecnológica de Costa Rica, 2000.

Hoare. "Hints on programming language design". Originalmente publicado en 1973. En Hoare y Jones (ed.), "Essays in Computing

Science", Prentice-Hall, 1989. También en Horowitz (ed.), "Programming languages, a grand tour". Computer Science Press / Springer-Verlag.

Gordon, M. "Programming Language Theory and its implementation" Prentice-Hall, New York, 1988.

Louden, K. "Lenguajes de Programación", 2da Ed, Thomson, 2004.

Lennan, B. "Principles of Programming Languages: Design, Evaluation, and Implementation", 2nd. edition. Holt, Rinehart & Winston (ahora Oxford University Press), 1986 (1995)

Pratt, T.;Zelkowitz, M. "Lenguajes de programación: diseño e implementación", Prentice-Hall Hispanoamericana, 1998.

Sethi, R. Lenguajes de programación: conceptos y constructores. Addison-Wesley, 1992. (Trad. edición en Inglés de 1989.)

Stansifer. "The study of programming languages". Prentice-Hall, 1995.

Tennent. "Principles of programming languages". Prentice-Hall, 1981.

Watt, D. "Programming language design concepts". John Wiley & Sons, 2004.

Wirth. "On the design of programming languages". Originalmente publicado en 1974. En Horowitz (ed.), Programming languages, a grand tour. Computer Science Press / Springer-Verlag.

8 Profesor

Diego Munguía Molina tiene estudios de Ingeniería en Computación (ITCR) y es egresado de la maestría en Ciencias Cognoscitivas (UCR), actualmente se encuentra desarrollando su tesis de graduación. Profesionalmente se ha desarrollado como arquitecto de software, acumulando experiencia en ingeniería de software, computación de alto rendimiento e integración de sistemas. Ha laborado como docente para la institución desde el 2012 en la Escuela de Computación impartiendo los cursos de Introducción a la Programación, Taller de Programación, Lenguajes de Programación,

Compiladores e Intérpretes, Diseño de Software e Inteligencia Artificial.

Medio oficial electrónico: TEC-Digital (www.tec-digital.itcr.ac.cr)
Medio de comunicación fuera de clase: Telegram
Correos electrónicos: dmunguia@itcr.ac.cr
Oficina de Ingeniería en Computación, SIUA
Horario de consulta: J 1:00pm-6:00pm