

Código: EIF-400
Nombre: Paradigmas de Programación
Requisitos: Programación 3
Naturaleza: Curso teórico/práctico
Área disciplinaria: Ingeniería de Software
Período: II Ciclo del III Nivel
II Ciclo 2008
Profesores: Georges Alfaro Salazar (coordinador)
Oldemar Rodríguez Rojas.

Créditos	Horas semanales	Horas presenciales (pueden variar según los contenidos específicos)		Horas de estudio independiente
		Teoría	Práctica	
4	10			6
		3 (2)	1 (2)	

OBJETIVO GENERAL

Al elaborar un modelo para resolver un problema mediante programación, existen diferentes enfoques sobre cómo se debe realizar la abstracción de los diferentes elementos de dicho problema. Así, dependiendo de la situación que se desea modelar, cada uno de estos distintos enfoques o paradigmas de programación tiene ventajas y desventajas, que facilitan o entorpecen la construcción de un programa. El objetivo principal del curso es estudiar de manera comparativa los diferentes paradigmas de programación existentes, y conocer los criterios más importantes para seleccionar un lenguaje determinado. El curso busca complementar el conocimiento de los estudiantes de ingeniería informática en los paradigmas y técnicas de modelado que no se han estudiado en los cursos regulares de programación.

OBJETIVOS ESPECÍFICOS

Al terminar el curso, se pretende que el estudiante haya adquirido el conocimiento necesario para:

1. Esbozar la historia de los lenguajes de programación y la manera en que estos han evolucionado para adaptarse a las necesidades de desarrollo de software.
2. Reconocer e identificar los tres principales paradigmas de programación existentes (según el modelo teórico que los sustenta) y sus características principales.
3. Identificar las características de cada lenguaje de programación, su implementación y ambiente de ejecución.
4. Identificar las particularidades de un lenguaje de programación que pueden afectar de una u otra manera la construcción de un programa.

5. Aprovechar las características de los lenguajes y las técnicas de programación orientada a objetos.
6. Emplear técnicas funcionales en la construcción de aplicaciones generales.
7. Comprender la especificación formal de condiciones de salida para la comprobación de resultados de un programa.

CONTENIDOS

Se propone que el desarrollo de cada una de las cuatro partes en que se divide el curso tenga una duración aproximada de cuatro semanas y media (equivalente a 9 lecciones). Dependiendo del desarrollo del curso, es posible que se reasignen algunas lecciones para poder cubrir adecuadamente cada uno de los temas propuestos.

Parte 1 - Características generales de los lenguajes de programación

Tema 1 - Introducción a los lenguajes de programación (Objetivos 1,2)

Historia de los lenguajes de programación.
Problemas de traducción.
Gramáticas y definición formal de lenguajes de programación.
Analizadores sintácticos y generación de código.

Tema 2 - Lenguajes de programación y arquitectura (Objetivo 2)

Tipos de datos y mecanismos de abstracción.
Métodos de encapsulamiento.

Tema 3 - Especificación y control (Objetivos 2,4)

Control de secuencia.
Control de subprogramas.

Tema 4 - Herencia y polimorfismo (Objetivos 2,5)

Conceptos generales e implementación.

Parte 2 - Programación imperativa

Tema 5 - Programación imperativa (Objetivos 2,3,4)

Máquinas de Turing y computabilidad.
Las estructuras básicas y la programación estructurada por control.
Comparación entre la programación imperativa secuencial y la programación algorítmica (estructurada).

Tema 6 - Programación guiada por eventos (Objetivos 2,3,5)

Programación guiada por eventos (*event-driven programming*) y programación guiada por flujo (*flow-driven programming*).
Modelo básico de la programación guiada por eventos.
El problema del manejo de la estructura de control básica (ciclo de atención de eventos).
Relación entre la implementación de las interfaces gráficas de usuario (GUIs) y la arquitectura de una aplicación de software.
Serialización de interfaces de usuario.

Tema 7 - Programación orientada a objetos (Objetivos 2,5)

Conceptos fundamentales y propósito de la POO (Programación Orientada a Objetos).

Principios empíricos de la POO.

Ventajas y desventajas de la POO. Consecuencias del empleo de técnicas de POO en el desarrollo de sistemas de información.

Patrones de diseño y su aplicación.

El patrón MVC (Modelo-Vista-Controlador)

Uso de intermediarios y adaptadores.

El patrón observador para el manejo de actualizaciones.

Programación en capas (*n-tier*).

Parte 3 - Programación funcional

Para esta parte del curso, se utilizará *Scheme* como lenguaje de programación funcional para estudiar los conceptos teóricos y algunas técnicas prácticas de programación.

Tema 8 - Principios de la programación funcional (Objetivos 2,3)

Justificación del modelo funcional.

Cálculo lambda (λ -calculus).

Tema 9 - Introducción al cálculo lambda (λ -calculus) (Objetivos 2,4,6)

Historia y descripción.

Evaluación por conversión y reducción.

Aritmética.

Recursividad.

Funciones computables.

Evaluación por sustitución.

Tema 10 - Recursividad (Objetivos 3,4,6)

Recursividad y funciones recursivas primitivas.

Recursividad simple, recursividad lineal y de cola.

Limitaciones de la programación funcional.

Parte 4 - Programación declarativa

Para esta parte del curso, se utilizará *Prolog* como lenguaje de programación declarativo para estudiar la implementación de los modelos lógicos descriptivos.

Tema 11 - Programación declarativa (Objetivos 2,3,4,7)

Programación declarativa vs. programación lógica.

Principios teóricos de la programación declarativa.

Lógica cuantificada de primer orden.

El algoritmo de unificación y el método de resolución.

Declaraciones recursivas.

ESTRATEGIA METODOLÓGICA

El curso se basa en la exposición magistral por parte del profesor de cada uno de los contenidos descritos. Además, se harán sesiones de resolución de problemas y prácticas de laboratorio para enfrentar al estudiante de manera directa con las principales dificultades y técnicas utilizadas en programación. De esta manera también se logra que el estudiante aplique los conceptos expuestos en situaciones específicas.

También habrá proyectos de programación que requieren más tiempo del que se dispone durante las lecciones, donde los estudiantes resuelven en grupo ejercicios de dificultad media o alta, para conocer, estudiar y resolver problemas representativos.

También se harán exposiciones sobre diferentes temas, cuyo estudio no puede hacerse dentro de las limitaciones de tiempo de la clase, pero cuyo conocimiento se considera importante para conseguir cumplir cabalmente el objetivo general del curso.

EVALUACIÓN

Los exámenes en el curso buscan medir y evaluar la comprensión de cada estudiante del material estudiado durante el curso y del trabajo realizado en los proyectos. Los **exámenes** deben realizarse y entregarse **individualmente**, aunque la realización de los **proyectos** pueden completarse en **grupos de dos personas como máximo**.

La suma de los porcentajes obtenidos por el estudiante en cada aspecto señalado determina su nota de aprovechamiento. Si ésta es superior o igual a 70%, el estudiante aprueba el curso. De lo contrario, el estudiante pierde el curso. Por ser un curso que incluye el desarrollo de proyectos prácticos, no existe la posibilidad de realizar un examen extraordinario.

Descripción	Porcentaje
Primer examen parcial Cubre los temas cubiertos en las dos primeras partes del curso: <ul style="list-style-type: none">• Características generales de los lenguajes de programación• Programación imperativa	15%
Segundo examen parcial Cubre los temas cubiertos en las dos segundas partes del curso: <ul style="list-style-type: none">• Programación funcional• Programación declarativa	20%
Quices y tareas Se realizarán 5 (cinco) exámenes cortos (quices) para evaluar el progreso del grupo. Los exámenes cortos se realizarán según el cronograma. También se asignarán tareas cortas que se asignarán con por lo menos una semana de	10%

anticipación por el profesor.	
<p>Trabajo de investigación</p> <p>Se hará un trabajo de investigación sobre algún lenguaje de programación particular o un tema relacionado con los objetivos del curso. Se hará en grupos de a lo más dos personas. El grupo de trabajo para la investigación estará compuesto preferiblemente por los mismos integrantes que para la realización de los proyectos.</p>	10%
<p>Proyectos</p> <p>Los proyectos servirán para evaluar aspectos prácticos concretos de los temas estudiados en el curso. Pueden realizarse en grupos de a lo más dos personas. Los proyectos asignados serán preferiblemente programados, pero pueden tratar también de algún tipo de desarrollo teórico.</p> <p>Los grupos de trabajo se formarán al inicio del curso, y permanecerán integrados de la misma manera para cada trabajo grupal asignado.</p>	<p>45%</p> <p>(se harán 3 proyectos, con un valor de 15% cada uno).</p> <p>Los proyectos deberán entregarse en clase, en la segunda lección de la semana correspondiente.</p>

CRONOGRAMA

Punto de evaluación	Fecha estimada
Primer examen parcial	Semana 9 15-20 septiembre 2008
Segundo examen parcial	Semana 18 17-22 noviembre 2008
Exámenes cortos y tareas	Durante todo el curso
Trabajo de investigación	El tema se asignará antes de la fecha del primer examen parcial. Se entrega al final de curso, en la segunda lección de la semana 16 (3-8 noviembre 2008).
Proyectos	<p><i>Proyecto 1:</i> Semana 8 8-13 septiembre 2008</p> <p><i>Proyecto 2:</i> Semana 13 13-18 octubre 2008</p> <p><i>Proyecto 3:</i> Semana 18 17-22 noviembre 2008</p>

BIBLIOGRAFÍA

Material de referencia y consulta:

Pratt, Terrence W., Zelkowitz, Marvin V. *Lenguajes de Programación. Diseño e implementación.*
Prentice Hall, México, 1998.

Watt, David A. *Programming Language Concepts and Paradigms.*
Prentice Hall, USA, 1990.

Aho A., Sethi Ravi, D. Ullman Jeffrey. *Compiladores: Principios, Técnicas y Herramientas.*
Addison Wesley, 1990.

Clocksin, W. F. *Clause And Effect.*
Springer-Verlag, 1997.

Clocksin, W. F., Mellish, C. S. *Programming in Prolog: Using the ISO Standard.*
Springer-Verlag; quinta edición, 2003.

Budd, Timothy. *Introduction to Object-Oriented Programming.*
Addison Wesley; tercera edición, 2001

Hankin, Chris. *An Introduction to Lambda Calculi for Computer Scientists.*
King's College Publications. Londres, 2004.

Rodríguez, Oldemar. *Introducción a la Programación Orientada a Objetos en C++ para Ambiente Windows.*
Editorial Tecnológica ITCR. Primera edición, 1997, ISBN 9977-66-099-9

Helo Guzmán, José E. *Introducción a la programación con Scheme.*
Editorial Tecnológica, ITCR. Segunda edición, 2005, ISBN 9977-66-176-6.

Además de los textos anteriores, se utilizarán los manuales respectivos de cada uno de los lenguajes estudiados, y bibliografía o material adicional que se anotará oportunamente durante el curso.

Software:

Para la elaboración de los proyectos se utilizará software provisto de manera oportuna por el profesor, que también podrá ser descargado de Internet sin costo ni necesidad de licencias de uso.

Las direcciones correspondientes para consultar documentación o descargar software son:

http://www.mingw.org/	gcc (para Windows)
http://www.c-frame.com/	IDE gcc (Windows y LINUX)
http://java.sun.com/	Java SDK
http://www.drjava.org/	DrJava
http://www.drscheme.org/	DrScheme
http://www.mv.com/ipusers/xlisper/	xLISP
http://www.dobrev.com/	Strawberry Prolog
http://www.swi-prolog.org/	SWI Prolog

NOTAS Y OBSERVACIONES GENERALES

El trabajo de investigación podrá referirse, pero no necesariamente limitarse, a alguno de los siguientes lenguajes de programación:

Ada
APL
AppleScript
B
BCPL
D
Eiffel
Forth
Haskell
Javascript
Leda
Miranda
ML
Modula-2
Oberon
Objective-C
Ocaml
Perl
PL/1
Python
Ruby
Self
SIMULA
SNOBOL4
Smalltalk
ZPL