ORGANIZACIÓN DOCENTE del curso 2011-2012

1. DATOS GENERALES DE LA ASIGNATURA

NOMBRE	**PROCESADORES DE LENGUAJE**	**PÁGINA WEB**	
CÓDIGO	5415		
DEPARTAMENTO	MATEMATICAS, ESTADISTICA Y COMPUTACION		
PLAN DE ESTUDIOS	INGENIERO EN INFORMATICA	**CURSO**	4º

PROFESORADO	**Nombre**	**e-mail**
	José Luis Balcázar Navarro	joseluis.balcazar@unican.es
	Domingo Gómez	

CRÉDITOS ALUMNO	Teóricos (1)	Prac. Problemas (2)	Prac. Laboratorio	Prac. Computador	TOTALES
	4,5		4,5		9

LUGAR DE IMPARTICIÓN(*)	Teóricos	Prac. Problemas	Prac. Laboratorio	Prac. Computador

HORARIO PREVISTO(*)	Teóricos	Prac. Problemas	Prac. Laboratorio	Prac. Computador

Observaciones:	

(*) Lo rellenará la secretaría del centro
(1) Se corresponde con clases magistrales de teoría en aula
(2) Se corresponde con clases prácticas (problemas, experiencias de cátedra,….) en aula

2. PROGRAMA DE LA ASIGNATURA

Tema 1: Introducción. Los lenguajes de programación y los procesos de interpretación y de compilación. Analogías y diferencias con otros procesos de lenguaje. Estructura y tipologías de compiladores: monopaso y multipaso, "front-end" y "back-end"; fases: léxica, sintáctica, semántica, generación de código, optimización.

Tema 2: Análisis sintáctico. Gramáticas incontextuales, RTNs, grafos sintácticos: su equivalencia. "Parsing": parsers universales, parsers descendentes LL, parsers ascendentes LR; indeterminismo. "Syntax error": su detección en el parsing LR y las estrategias de solución.

Tema 3: Representaciones internas. Arboles de derivación, árboles de sintaxis abstracta; introducción al cálculo de atributos: recorridos de árboles. XML: parsers basados en expat, árboles DOM.

Tema 4: Análisis léxico. Gramáticas regulares, TN's, autómatas finitos, expresiones regulares. Expresiones regulares extendidas, su uso en contextos Unix. Prefijos viables de lenguajes incontextuales: explicación del parsing LR en base a los lenguajes regulares. Uso de generadores de analizadores léxicos (flex). El "Toolbox" de Unix (sed, tr, cmp, nl, wc, join, cut).

Tema 5: Tablas de símbolos. Visibilidad, reglas de "overriding", "scopes". Intérpretes y scripting: awk, Tcl/Tk, Perl, Python.

Tema 6: Análisis semántico. Rutinas semánticas, cálculo de atributos por síntesis, cálculo de atributos por herencia. Análisis semántico mediante árboles de sintaxis abstracta. Uso de generadores de analizadores sintácticos (bison). Su extensión para el análisis semántico, para la construcción de árboles de sintaxis abstracta y para el cálculo de atributos. SAX y eventos en el proceso XML.

Tema 7: Generación de código. Código intermedio. Código de tres direcciones comparado con código de una dirección. Generación de código en compiladores multipaso.

Tema 8: Optimización de código. Descomposición en bloques, grafo de control de flujo. Optimizaciones "peephole".

Tema 9: Aspectos avanzados. Serialización. Compilación de lenguajes lógicos. Compilación de lenguajes funcionales. Compilación de lenguajes orientados a objetos. El "lenguaje natural": las dificultades adicionales que plantea.

Asignaturas que se recomienda al alumno haber cursado o estar cursando

3. OBJETIVOS GENERALES DE LA ASIGNATURA

Se pretende que, tras cursar esta asignatura, cada alumno sea mejor programador en diversos sentidos: de una parte, al conocer mejor cómo se trata en detalle cada construcción de sus programas, será más ágil en interpretar los mensajes del compilador y capaz de identificar más fácilmente posibles errores; de otra, el conocimiento de los principios que subyacen en la construcción de compiladores le permitirá detectar, en la carrera primero y en su vida profesional después, oportunidades de aplicación de estos mismos principios. A fin de obtener un buen rendimiento en este segundo aspecto, se trabajarán ejemplos de uso de las herramientas disponibles que permiten aplicar dichos principios con un esfuerzo relativamente bajo: generadores de analizadores de distintos tipos, uso de analizadores para XML, herramientas conceptuales apropiadas para la implementación de tablas de símbolos. Se espera que esta asignatura contribuya de manera importante a la capacidad del alumno para desarrollar procesos mentales con rigor y precisión y, a la vez, con creatividad y autonomía, tanto en la programación a pequeña escala (precisión) como en la concepción de un proyecto de programación de un tamaño relativamente grande. Adicionalmente, tras cursar esta asignatura el alumno ha de ser capaz de explicarse y explicar los principios básicos de funcionamiento de un compilador habitual.

4. OBJETIVOS ESPECIFICOS: APTITUDES/DESTREZAS

Manejo ágil de los lenguajes regulares, tanto conceptualmente (expresiones regulares, autómatas finitos) como a través de los programas que los usan (flex, egrep, sed…)

Manejo ágil de los lenguajes incontextuales, tanto conceptualmente (gramáticas incontextuales, analizadores de pila, análisis ascendente) como a través de los programas que los usan (bison, expat, SAX, DOM)

Comprensión de los diversos mecanismos de ocultación y visibilidad en lenguajes de programación de diversos paradigmas

Comprensión ágil de la programación en paradigmas y lenguajes no conocidos (scripting, funcional…)

5. BIBLIOGRAFÍA

Básica

Aho, A. V.; Lam, M. S.; Sethi, R.; Ullman, J. D.: "Compilers. Principles, Techniques and Tools" 2ª ed. Addison-Wesley. 2007. (O la traducción al castellano de la edición de 1985.)

Appel, A. W.; Palsberg, J.: "Modern Compiler Implementation in Java". 2ª ed. Cambridge University Press. 2002.

Appel, A. W.: "Modern Compiler Implementation in C". Cambridge University Press.

Levine, J. R.; Mason, T.; Brown, D.: "Lex & Yacc". O'Reilly. 1992.

Complementaria

Grune, D.; Bal, H.; Jacobs, C; Langendoen, K.: "Modern Compiler Design". John Wiley & Sons. 2000.

Torben Ægidius Mogensen: Basics of compiler design – edición electrónica en lulu.com (ver http://www.diku.dk/_torbenm/Basics)

Wilhelm, R.; Maurer, D.: "Compiler Design". Addison-Wesley. 1995.

Appel, A. W.: "Modern Compiler Implementation in ML". Cambridge University Press.

Jones, C.A.; Drake, F.L.: Python and XML . O'Reilly. 2001.

6. ACTIVIDADES A DESARROLLAR EN LA ASIGNATURA

* Desarrollo de conceptos teóricos por el profesor.

* Familiarización con dichos conceptos en sesiones de problemas, guiados por el profesor pero resueltos por los alumnos.

* Implementación de pequeños analizadores.

* Comprensión de la aplicabilidad de los métodos descritos para otras tareas informáticas.

* Desarrollo autónomo de un proyecto relacionado con la compilación a lo largo del curso.

7. MÉTODO DE EVALUACIÓN

Establecer en cada caso el peso en porcentaje que tiene en la evaluación de la asignatura la parte de la evaluación continua, y la correspondiente a la prueba del examen final.

Descripción de la evaluación continua: actividades que debe desarrollar el alumno y su valoración

La nota final es la suma de dos calificaciones: la nota de comprensión básica, que contribuye entre 0 y 6 puntos, y la nota de comprensión profunda, que contribuye entre 0 y 4 puntos.

A su vez, la nota de comprensión básica es la suma de dos notas: la obtenida a lo largo del desarrollo de la asignatura a través de la evaluación continua, que contribuye entre 0 y 6 puntos, y la nota de la parte básica del examen final, que contribuye asimismo entre 0 y 6 puntos: estas dos notas se suman y, en caso de superar el 6, el resultado se trunca a 6.

La nota de comprensión profunda se obtiene en la parte avanzada del examen final.

La evaluación continua se desarrolla como sigue: con una periodicidad a determinar (posiblemente quincenal), se habrán de entregar y defender desarrollos de aplicaciones de los conceptos tratados, que podrán tomar la forma de resolución de problemas o de implementación mediante el uso de las herramientas adecuadas; asimismo, antes de la fecha de entrega que se acuerde a final de curso, se habrá de completar un proyecto de implementación relacionado con los compiladores.

Cada una de estas actividades recibirá una nota que se irá acumulando a las anteriores, permitiendo alcanzar a final de curso un máximo de 6 puntos sobre 10.

Descripción del examen final (duración, se pueden llevar apuntes o no, tiene partes diferenciadas o no, se promedian teoría y problemas o no, etc).

La duración no superará las tres horas, sólo constará de problemas, y se podrá llevar apuntes. Estará estructurado en dos partes: la parte básica contribuye a la nota de comprensión básica, y la parte avanzada proporciona la nota de comprensión profunda.

La parte básica permitirá obtener hasta 6 puntos, que se sumarán a los obtenidos en la evaluación continuada; la suma obtenida se truncará a 6 puntos. De este modo, cada alumno puede obtener mediante evaluación continuada la parte de la calificación de comprensión básica que más convenga a su propia situación, entre dos extremos: quien haya superado a la perfección la evaluación continuada se presenta al examen final sabiendo que tiene la asignatura aprobada y a fin de subir nota, y quien no haya podido esforzarse suficientemente a lo largo del curso tiene aún la opción de alcanzar los 6 puntos de esta parte en el examen final.

La parte avanzada es la única oportunidad de obtener notas superiores al 6, constituye la nota de comprensión profunda, y su calificación se suma a la nota de comprensión básica, proporcionando así la nota final.

8. OBSERVACIONES

Indicar en este apartado si se prevé el uso de algún tipo de Software (nombre y versión), además de cualquier otra observación que se desee.

Únicamente herramientas estándar de Unix/Linux o software libre basado en C++, Java o Python.