COMPILADORS II
ENGINYERIA D'INFORMÀTICA 8° Semestre

Créditos: 3 (teoría) + 1'5 (problemas) + 1'5 (Prácticas)
Horas: 26 (teoría) + 11 (problemas) + 30 (Prácticas)

TEMARIO

TEMA 1. PARADIGMAS DE LOS LENGUAJES DE PROGRAMACIÓN.

(teoría 6h.)

Lección 1. CARACTERÍSTICAS GENERALES. Valores. Almacenamiento. Ligaduras. Abstracción.

Lección 2. SISTEMAS DE TIPOS Y LENGUAJES IMPERATIVOS. Monomorfismo. Polimorfismo. Tipos Paramétricos. Inferencia de Tipos. Coerciones. Subtipos. Flujos de control. Excepciones.

Lección 3. ENCAPSULACIÓN Y PARADIGMAS DE OBJETOS. Tipos abstractos. Objetos. Clases. Herencia.

Lección 4. PROGRAMACIÓN FUNCIONAL Y PROGRAMACIÓN LÓGICA. Funciones. Pattern matching. Valores y tipos.. Funcionales. Programación lógica. Cláusulas, relaciones y términos. Concepto de mundo cerrado.

Lección 5. CONCURRENCIA. Programas. Procesos. Deadlock. Interacción de procesos. Primitivas de bajo nivel. Programación concurrente estructurada.

Lección 6. ENTRADA, SALIDA Y FICHEROS. Entrada / salida incluida en las primitivas del lenguaje o en bibliotecas de funciones. Tipos de datos y la entrada / salida. Conceptos de entrada / salida en un lenguaje. Ejemplos: Pascal, C, C++.

TEMA 2. ESPECIFICACIÓN FORMAL DE LENGUAJES.

(teoría 4h., problemas 5h.)

Lección 7, 8 y 9. SEMÁNTICA INTERPRETATIVA. Máquina virtual. Cálculo. Resultado. Estado. Construcción abstracta. Lenguaje de la máquina virtual. Construcción de la semántica interpretativa para un lenguaje. utilización de una semántica interpretativa.

Lección 10, 11, 12 y 13. GRAMÁTICAS DE ATRIBUTOS. Atributo. Atributos calculados. Atributos heredados. Condiciones. Funciones. Gramática de atributos para un lenguaje de programación.

TEMA 3. GRAMÁTICAS Y ANÁLISIS SINTÁCTICO: PARSER BOTTOM-UP.

(teoría 2h., problemas 1h.)

Lección 14, 15 y 16. PARSERS LR: LR(0), LR(1), SLR(1). Parsers LR(0). Corrección de un parser LR(0). Parsers LR(1). Corrección de un parser LR(1). Parsers SLR(1). Corrección y Limitaciones. Construcción de un parser LALR(1). Corrección de un parser LALR(1).

Lección 17. CONSIDERACIONES PRÁCTICAS. Parsers LR prácticos. Propiedades del análisis sintáctico LR. ¿Utilizar LL(1) o LALR(1)?.

TEMA 4. ANÁLISIS SEMÁNTICO AVANZADO.

(teoría 2h., problemas 1h.)

Lección 18. OBJETOS, SOBRECARGA Y HERENCIA.

Lección 19. TIPOS PARAMÉTRICOS.

Lección 20. INFERENCIA DE TIPOS. Reglas de inferencia de tipos y Algoritmos.

TEMA 5. CÓDIGO INTERMEDIO.

(teoría 1h.)

Lección 21. FORMATOS INTERNOS INTERMEDIOS DE LOS PROGRAMAS FUENTE. Notación polaca. *n*-tuplas. Árboles sintácticos abstractos. Código enhebrado. Código para máquinas abstractas.

TEMA 6. OPTIMIZACIÓN DE CÓDIGO.

(teoría 5h., problemas 2h.)

Lección 22. OPTIMIZACIONES BÁSICAS. Introducción. Bloques básicos. Ensamblamiento. Eliminación de subexpresiones redundantes.

Lección 23. OPTIMIZACIONES DENTRO DE BUCLES. Expansión de bucles. Reducción de frecuencia. Reducción de potencia.

Lección 24. OPTIMIZACIÓN GLOBAL. Construcción del grafo de flujo. Análisis del grafo de flujo. Aplicaciones de la optimización de programas.

OPTIMIZACIÓN DEPENDIENTE DE MÁQUINA. Introducción. Asignación de Registros. Arquitectura máquina y Generación de Código Real.

TEMA 7. INTERPRETES.

(teoría 4h., problemas 2h.)

Lección 25. INTRODUCCIÓN. Ciclo de interpretación. Estrategias de interpretación: Código intermedio y compilador-intérprete. Organización de memoria usuales en los intérpretes.

Lección 26. INTERPRETACIÓN ITERATIVA. Interpretación de código máquina. Interpretación de lenguajes de comandos. Interpretación de lenguajes imperativos. Ejemplo: Basic.

Lección 27. FUNCIONALES. LISP: Ciclo de interpretación. Nodo LISP. Listas. Símbolos. Listas de propiedad. Entornos y ligaduras. Garbage collection.

Lección 28. LÓGICOS. PROLOG: Pila de backtracking. Árbol sintáctico abstracto y constructores. Unificación. tratamiento del desconocido. Ciclo de interpretación de PROLOG.

TEMA 8. GENERADORES DE COMPILADORES Y HERRAMIENTAS.

(teoría 2h.)

Lección 29. GENERADORES DE PARSERS. Introducción. Generadores de parsers. Generadores de compiladores. YACC.

Lección 30. OTRAS HERRAMIENTAS. Entornos de compilación. Bancos de trabajo. Depuradores. Editores guiados por la sintaxis.

Práctica

Practica 1. IMPLEMENTAR UN COMPILADOR. Generando código máquina (ensamblador) y con optimización de código generado.

Grupos de 2 personas.

A partir de:

- Definición de la gramática.
- Restricciones Semánticas.
- Control de Calidad. Juego de pruebas.
- Documentación de usuario.
- Implementación de un analizador / generador de código intermedio.

Presentado y explicado en clase de prácticas . 10 h. / persona

Se requiere:

- Estrategias de Optimización.	3 h. / persona
- Estrategia de la Generación de Código.	(20 h./ Grupo
- Implementación.	General 160 h.)
- Documentación Técnica.	5 h. / Grupo
- Pruebas.	9 h. / Grupo

Total: 30 h. / persona

Bibliografia Básica.

The Theory and Practice of Compiler Writing, Jean-Paul Tremblay & Paul G. Sorenson, MCGRAW-HILL, 1985.

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

Thèorie des programmes. Schémes, preuves, sémantique, C. Livercy, Dunod, 1984.

Formal Specification of Programming Languages: A Panoramic Primer, Frank G. Pagan, Prentice Hall Inc., 1981.

Bibliografia de Consulta.

Compiladores: Principios, técnicas y herramientas, Alfred V. Aho, Ravi Sethi & Jeffrey D. Ullman, Addison-Wesley, 1990.

Programming Language Processors, David A. Watt, Prentice Hall, 1993.

Functional Programming, Anthony J. Field & Peter G. Harrison, Addison-Wesley, 1988.

Crafting a Compiler with C, Charles N. Fisher & Richard J. Leblanc jr., The Benjamin / Cummings Publishing Company inc., 1991.

Programming Languages: Design and Implementation, Terrence W. Pratt, Prentice Hall International Editions, 1984.

Compiler Construction. An Advanced Course, Edited by G. Goos and J. Hartmanis, Springer-Verlag, 1974.

The Design of an Optimizing Compiler, William Wulf et all., North Holland, 1980.

An Implementation Guide to Compiler Writing, Jean-Paul Tremblay & Paul G. Sorenson, MCGRAW-HILL, 1982.

Writing Compilers & Interpreters: An Applied Approach, Ronald Mak, Wiley, 1991.

Writing Interactive Compilers and Interpreters, J. P. Brown, Wiley, 1979.

Compiler Design in C, Allen Y. Holub, Prentice Hall, 1990.

Método de evaluación

Nota final: 0.6 * Nota Teoría + 0.3 * Nota examen prácticas + 0.1 * Práctica
Notas mínimas:5 (teoría), 5 (examen prácticas), 5 (práctica)
Prácticas obligatorias para aprobar el curso: **SI**
La nota de prácticas será función del número de módulos implementados y del número de grupos que los utilicen.

Para el resto de temas rige la normativa interna de la Unidad de Proceso de Imágenes.

Profesores

Grupo	Clase	Profesor	Despacho	Consultas
1	Teoría Problemas	Francisco Navarro	C5-335 C5-337	Lunes 18:30 - 19:30 Viernes 19:30 - 20:30
1,2,3	Prácticas	Gemma Sánchez	C5-337 C5-337	Martes 10:30 - 11:30 Martes 18:30 - 19:30