

## **Assignatura : Compiladors**

### **Objectius**

---

Conocimientos:

- Análisis lexicográfico, sintáctico y semántico.
- Organización de memoria de los programas durante su ejecución.
- Generación de código.
- Gestión de memoria.

Habilidades y competencias:

- Un mayor conocimiento sobre el funcionamiento de los compiladores.
- Aprovechar mejor las capacidades de los compiladores.
- Escribir programas más óptimos y legibles.
- Entender los errores de compilación.
- Depuración más rápida de programas.
- Programación más abstracta.

### **Capacitats prèvies**

---

Conocimientos sobre programación imperativa.

### **Continguts**

---

- **TEMA 1. INTRODUCCION. CONCEPTOS BASICOS**
  - [Lecciones 1 y 2.] **GENERALIDADES.** Código fuente. Código objeto. Gramáticas y lenguajes de programación. Etapas de compilación: análisis lexicográfico, análisis sintáctico, análisis semántico y generación de código. Tabla de símbolos. Compiladores y Recursividad.
- **TEMA 2. ANALISIS LEXICOGRAFICO: SCANNER**
  - [Lección 3.] **ANALISIS LEXICOGRAFICO.** Símbolo. Scanner. Expresiones regulares. Autómatas finitos y scanners.
  - [Lección 4.] **SCANNER.** Consideraciones prácticas. Palabras reservadas. Final del código fuente. Análisis de múltiples caracteres hacia delante. Recuperación de errores léxicos.
- **TEMA 3. GRAMATICAS Y ANALISIS SINTACTICO: PARSER.**

- [Lección 5.] **GRAMÁTICAS.** Gramáticas Libres del Contexto. Definición de gramática. Clasificación: regulares, libres del contexto, dependientes del contexto, recursivas. Notación BNF extendida
- [Lecciones 6 y 7.] **NOTACIONES.** Notación. Diagramas Sintácticos. Traducción de BNF a diagramas sintácticos.
- [Lección 8]. **GRAMÁTICAS Y LENGUAJES DE PROGRAMACIÓN.** Características de las Gramáticas y Lenguajes de Programación. Árbol sintáctico. Ambigüedad gramatical. Estructura de un lenguaje y su gramática
- [Lección 9]. **PARSERS.** Gramáticas y Parsers LL(1): Análisis un símbolo en adelante sin vuelta a tras. Primeros y siguientes. Definición de gramática LL(1).
- [Lección ] **PARSERS.** Construcción de un parser LL(1): traducción de las reglas BNF en un parser, traducción de los diagramas sintácticos en un parser
- [Lección 15]. **ERRORES.** Recuperación de Errores.
- **TEMA 4. ANÁLISIS SEMÁNTICO**
  - [Lección 16]. **TRADUCCIÓN DIRIGIDA POR LA SINTAXIS.** Rutinas semánticas, símbolos de acción, atributos.
  - [Lección 17]. **TABLA DE SÍMBOLOS I.** Contenido: símbolos y atributos. Técnicas básicas de búsqueda: secuencial, árbol binario, tabla hash. Estructura de bloques de la tabla de símbolos para: lenguajes no estructurados, lenguajes estrucados por bloques anidados, etc.
  - [Lección 18]. **TABLA DE SÍMBOLOS II.** Representación de atributos recursivos (tipos). Referencias hacia delante. Consideraciones de orden práctico: Creación y destrucción en la tabla de símbolos.
  - [Lección 19]. **ANÁLISIS SEMÁNTICO DE DECLARACIONES.**
  - [Lección 20]. **ANÁLISIS SEMÁNTICO DE EXPRESIONES E INSTRUCCIONES.**
- **TEMA 5. ORGANIZACIÓN DE MEMORIA EN TIEMPO DE EJECUCION.**
  - [Lección 21]. **AMBIENTE DE EJECUCIÓN.** Organización de memoria de un programa. Asignación estática de memoria: Arrays. Registros. Conjuntos.
  - [Lección 22 y 23]. **PILA DE EJECUCIÓN.** Asignación de la Pila. Registro de activación. Enlaces estáticos y dinámicos. Displays. Llamada y retorno de una función. Resultados intermedios.
  - [Lección 24]. **HEAP.** Asignación Dinámica de Memoria. Liberación explícita. Liberación Implícita: contador de referencias, garbage collection, compactación.
- **TEMA 6. GENERACION DE CODIGO.**
  - [Lección 25]. **GENERACIÓN DE CÓDIGO.** Procesamiento de declaraciones y tabla de símbolos. Compilación de Estructuras de Control.
  - [Lección 26 y 27]. **GENERACIÓN DE CÓDIGO DE EXPRESIONES.** Notación postfija. Pila para los resultados intermedios. Casos: expresiones PASCAL y C.
  - [Lección 28 y 29]. **DIRECCIONES.** Procesar referencias a Estructuras de Datos. Selectores como operadores. Cálculo de la dirección. Casos: estructuras de datos PASCAL y C.

## Metodologia docent

---

**Teoría:** Clases magistrales.

**Problemas:** clases en las que se reafirman los conocimientos adquiridos en teoría con problemas donde se aplican los métodos estudiados en teoría.

**Prácticas:** Implementación de los conceptos adquiridos en teoría (implementar un micro compilador). El objetivo de la práctica es realizar un compilador de un lenguaje imperativo simple que genera código de una máquina abstracta de la que se implementa el correspondiente intérprete.

El lenguaje a compilar es no tipado, por lo tanto en las declaraciones no es necesario especificar el tipo de datos de las variables, funciones, parámetros, etc. La verificación de los tipos de datos se realizará en tiempo de ejecución. Tiene variables locales y globales, funciones globales, las instrucciones mínimas de control de flujo. Los tipos de datos con los que podrá trabajar serán números, strings y vectores.

La práctica se realiza en grupos de dos alumnos.

## Avaluació

---

(assignatures que no estan al PLA PILOT)

| Avaluació continuada   | Examen final<br>(1ª convocatòria)   | Examen final<br>(2ª convocatòria)   |
|--|---|---|
| <input type="checkbox"/> NO  | <input checked="" type="checkbox"/> No n'hi ha  | <input type="checkbox"/> Només per als estudiants que compleixin les condicions següents: |
| <input checked="" type="checkbox"/> SÍ<br>En què consisteix?<br>Evaluación de la pràctica. | <input type="checkbox"/> Només per als estudiants que compleixin les condicions següents: |   |
|  | <input type="checkbox"/> Obligatori per a tothom  | <input type="checkbox"/> Oberta a tothom  |

### Método de evaluación:

La nota se obtiene a partir de la práctica. Se realiza una entrega de seguimiento por cada parte de la práctica (scanner, parser, analizador semántico y generador de código) y una entrega final. La entrega final de la práctica es presencial y consta de dos partes: en la primera el profesor comprueba el correcto funcionamiento de la práctica y si esta pasa el mínimo exigido, cada componente del grupo de prácticas responde un cuestionario sobre la práctica que han realizado.

## **Bibliografia bàsica**

---

- **Crafting a Compiler with C, Charles N. Fisher & Richard J. Leblanc jr., The Benjamin / Cummings Publishing Company inc., 1991.**
- **Modern Compiler Design. Dick Grune et al. Wiley, 2000.**
- **The Theory and Practice of Compiler Writing, Jean-Paul Tremblay & Paul G. Sorenson, MCGRAW-HILL, 1985.**
- **Compiladores: Principios, técnicas y herramientas, Alfred V. Aho, Ravi Sethi & Jeffrey D. Ullman, Addison-Wesley, 1990.**

## **Bibliografia complementària**

---

- **Programming Language Concepts and Paradigms, David A. Watt, Prentice Hall, 1990.**
- **Programming Language Processors, David A. Watt, Prentice Hall, 1993.**
- **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 al., 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.**
- **Garbage Collection, Richard Jones y Rafael Lins, Wiley 1996.**