

ASIGNATURA: 14107 - FUNDAMENTOS DE LENGUAJES

CENTRO: Escuela de Ingeniería de Telecomunicación y Electrónica

TITULACIÓN: Ingeniero de Telecomunicación

DEPARTAMENTO: INGENIERÍA TELEMÁTICA

ÁREA: Ingeniería Telemática

PLAN: 13 - Año 2000**ESPECIALIDAD:**

CURSO: Cuarto curso **IMPARTIDA:** Segundo semestre **TIPO:** Optativa

CRÉDITOS: 4,5

TEÓRICOS: 3

PRÁCTICOS: 1,5

Información ECTS

Créditos ECTS: 3,6

Horas de trabajo del alumno: 130

Horas presenciales: 7,0

- Horas teóricas (HT): 0
- Horas prácticas (HP): 0
- Horas de clases tutorizadas (HCT): 5,0
- Horas de evaluación: 2,0
- otras: 0,0

Horas no presenciales:

- trabajos tutorizados (HTT): 31,5
- actividad independiente (HAI): 11,5

Idioma en que se imparte: Español

Descriptores B.O.E.

Diseño de Lenguajes: Conceptos y Paradigmas; valores, almacenamiento, declaraciones, ámbitos, tipos de datos, abstracción y encapsulado; Paradigma imperativo, concurrente, orientado a objetos, funcional y lógico. Procesadores de Lenguajes: traductores e intérpretes: análisis léxico, análisis semántico y generación de códigos.

Temario

Tema 1: Introducción a los lenguajes de programación. Modelos de computación. Características comunes de los lenguajes modernos. Sintaxis y semántica. Gramáticas: clasificación de Chomsky. Compiladores e intérpretes (6 horas)

PRIMERA PARTE: PROCESADORES DE LENGUAJES

Tema 2: Procesadores de lenguajes. Análisis léxico. Separadores, operadores, palabras reservadas e identificadores. Expresiones regulares. Autómatas. Alternativas de implementación. (3 horas)

Tema 3: Análisis sintáctico. Métodos descendente. Descenso recursivo. Métodos guiados por tablas. (3 horas)

Tema 4: Análisis semántico. Atributos y gramáticas con atributos. Atributos heredados y sintetizados. Resolución de nombres. (2 horas)

Tema 5: Generación de código. Estructuras de control. Evaluación eficiente de expresiones. Técnicas básicas de optimización de código. (3 horas)

Tema 6: Soporte de ejecución. Registro de activación de subprogramas. Gestión de ámbitos. Gestión de parámetros. Funciones con número de parámetros variable. Objetos de tamaño dinámico. Gestión de memoria dinámica. (2 horas)

SEGUNDA PARTE: CONCEPTOS DE LENGUAJES DE PROGRAMACIÓN

Tema 7: Lenguajes imperativos. Estructuras de control. Tipos de datos y su representación. Declaraciones de tipos y subtipos de datos. Subprogramas y módulos. Paso de parámetros. Anidación. Soporte para módulos en Ada, Java y C++. (4 horas)

Tema 8: Programación orientada a objetos. Constructores y destructores. Herencia simple. Polimorfismo. Dynamic Dispatching. Clases abstractas. Herencia múltiple. Comparación entre Ada, Java y C++. (2 horas)

Tema 9: Programación Concurrente. Declaración de tareas. Activación de tareas. Comunicación y sincronización. Programación funcional: gestión de listas. Programación Lógica: gestión de predicados. (2 horas)

Tema 10: Presentación y defensa de trabajos. (3 horas)

Requisitos Previos

Se recomienda que el alumno tenga aprobadas las asignaturas Fundamentos de Programación y Programación de la ETSIT, así como la mayoría de las asignaturas de tercer curso del plan de estudios de la titulación de Ingeniero de Telecomunicación o estudios similares, y que tenga conocimientos básicos de uso de sistemas UNIX (por ejemplo, Linux).

Objetivos

1.0 Objetivos conceptuales:

- 1.1 Situar la asignatura en el contexto de las Telecomunicaciones
- 1.2 Diferenciar entre traductores e intérpretes
- 1.3 Conocer la estructura de un traductor
- 1.4 Comprender la relación entre los componentes de un traductor
- 1.5 Reconocer conceptos comunes de lenguajes de programación modernos y su traducción
- 1.6 Conocer los principales paradigmas de programación actuales

2.0 Objetivos procedimentales:

- 2.1 Construir un traductor
- 2.2 Utilizar la programación orientada a objetos para desarrollar herramientas telemáticas
- 2.3 Aplicar el desarrollo en espiral para la construcción de prototipos

3.0 Objetivos Actitudinales

- 3.1 Interesarse por los recursos que ofrecen los traductores a las Telecomunicaciones
- 3.2 Comunicar de forma oral los problemas encontrados durante el desarrollo del prototipo

Metodología

La disposición Transitoria Cuarta del Reglamento de Planificación Académica de la ULPGC establece que las asignaturas de los títulos no adaptados tendrán el primer año de su extinción una carga docente del 25% de las horas contempladas en el plan de estudios para la realización de actividades de docencia y evaluación, y de un 10% el segundo año.

Puesto que el curso 2013-2014 es el primer año de extinción de ésta asignatura de 3 créditos de teoría y 1,5 de prácticas, se impartirán 11,25 horas distribuidas como sigue:

- a) 7,5 horas de tutoría presencial de la parte de teoría durante las cuales se facilitará a los alumnos el seguimiento secuencial de la asignatura resolviendo dudas y proponiendo temas y ejercicios para la siguiente sesión.
- b) 4,25 horas de tutoría presencial de la parte práctica durante las cuales se facilitará a los alumnos el seguimiento de la parte de laboratorio de la asignatura.

Las actividades de tutoría se realizarán en el laboratorio de Arquitecturas del Departamento de Ingeniería Telemática, en el recinto C204 del Pabellón C de los Edificios de Telecomunicación del Campus de Tafira.

El periodo de realización de las tutorías corresponderán a 1,25 horas en la segunda semana, 2 horas en la cuarta semana, 2 horas en la sexta semana, 2 horas en la octava semana, 2 horas en la décima semana y 2 horas en la duodécima semana, según el periodo lectivo de la Universidad.

En el Campus Virtual el alumno dispondrá de material complementario de teoría, así como la documentación necesaria para la realización de la práctica.

Criterios de Evaluación

Actividades que liberan materia:

- La presentación y defensa de un trabajo teórico consistente en la exposición y defensa de un artículo técnico libera hasta un máximo del 20% de la nota final.
- El ejercicio práctico de la asignatura libera hasta un máximo del 80% de la nota final.

La evaluación del ejercicio práctico se realiza de forma continua (no es una prueba escrita) durante las tutorías de la asignatura; el alumno debe realizar entregas mensuales por correo electrónico del estado de su práctica; la evaluación del ejercicio teórico se realiza de forma presencial en una tutoría presencial con el profesor de la asignatura.

En cada convocatoria oficial de la asignatura la evaluación se realiza de forma presencial concertando una tutoría presencial con el profesor para defender el trabajo realizado y realizar la exposición de su trabajo teórico.

Consideraciones generales:

- En las convocatorias oficiales de la asignatura el alumno deberá realizar la defensa de su trabajo teórico (puntuación hasta un 20% de la asignatura) y de su trabajo práctico (puntuación hasta un 80% de la asignatura).
- La nota final de la asignatura se obtiene mediante la suma de ambas puntuaciones. Por tanto, no es necesario aprobar ambas partes para aprobar la asignatura.

Descripción de las Prácticas

Las prácticas se pueden desarrollar de forma no presencial o en el Laboratorio de Arquitecturas del Departamento de Ingeniería Telemática.

A lo largo del curso se realiza una única práctica consistente en el diseño de un pequeño lenguaje basado en el paradigma de programación imperativa y su compilador. El lenguaje debe tener un soporte mínimo para declaraciones de variables, gestión de expresiones aritméticas y lógicas, registros, arrays y procedimientos anidados. Además los alumnos pueden incorporar en su lenguaje características de programación orientada a objetos, excepciones, etc. El objetivo de la práctica es aplicar los aspectos fundamentales de diseño y desarrollo de procesadores de lenguajes de esta asignatura.

El desarrollo completo de esta práctica consta de los siguiente sub-apartados (o sub-prácticas):

- 1) Desarrollo del módulo de análisis léxico. Este módulo se encarga de reconocer las palabras aceptadas por el lenguaje diseñado por el alumno.
- 2) Desarrollo del módulo de análisis sintáctico. Este módulo se encarga de reconocer las frases válidas en el lenguaje de programación diseñado por el alumno, así como de notificar los errores de sintaxis existentes en los programas de prueba utilizados para comprobar el traductor.
- 3) Desarrollo del módulo de análisis semántico. Este módulo se encarga de realizar las verificaciones de tipos de datos asociadas al lenguaje, así como de identificar sin ambigüedades todos los objetos y tipos de datos del programa que se traduce.
- 4) Desarrollo del módulo de generación de código. Este módulo se encarga de realizar la traducción del programa al lenguaje destino. Se recomienda que los alumnos generen código para la plataforma .NET ya que proporciona un nivel de abstracción alto que facilita el desarrollo de este módulo de forma independiente de la arquitectura. Además, la elección de .NET simplifica la complejidad de implementación este módulo del traductor. Los alumnos interesados en experimentar con arquitecturas específicas pueden proponer la generación de código para CPUs específicas (por ejemplo pueden generar código ensamblador de la arquitectura x86).

Esta práctica se realiza de forma individual.

Bibliografía

[1 Básico] Compilers: principles, techniques and tools /

Alfred V. Aho ...[et al.].
Addison Wesley,, Boston : (2007) - (2nd ed.)
0321491696 (ed. int.)

[2 Básico] Concepts of programming languages /

Robert W. Sebesta.
Addison-Wesley,, Boston (Massachusetts) : (2002) - (5th ed.)
0201752956

[3 Básico] Programming languages: design and implementation /

Terrence W. Pratt, Marvin V. Zelkowitz.
Prentice Hall,, Upper Saddle River, New Jersey : (2000) - (4th. ed.)
0-13-027678-2

[4 Recomendado] Concurrent and real-time programming in Ada 2005 /

Alan Burns and Andy Wellings.
Cambridge University Press,, Cambridge [etc.] : (2007)
9780521866972

[5 Recomendado] Concurrent and real-time programming in Java /

Andy Wellings.
John Wiley,, Chichester, West Sussex, England ; (2004)
047084437X

[6 Recomendado] The design and evolution of C++ /

Bjarne Stroustrup.
Addison-Wesley,, Boston : (1994)

[7 Recomendado] Programming in ADA 2005 /

John Barnes.
Addison Wesley,, Harlow : (2006)
0321340787

[8 Recomendado] Compiling for the .NET common language runtime (CLR) /

John Gough.
Prentice Hall PTR,, Upper Saddle River, New Jersey : (2002)
0-13-062296-6

[9 Recomendado] Programming language pragmatics /

Michael L. Scott.
Morgan Kaufmann,, San Francisco : (2000)
1-55860-578-9

[10 Recomendado] Garbage collection :algorithms for automatic dynamic memory management /

Richard Jones, Rafael Lins.
John Wiley,, Chichester : (2006) - (reimp. 2006.)
9780471941484

[11 Recomendado] Comparative programming languages.

Wilson, Leslie B.
Addison-Wesley,, Wokingham, England : (1993) - (2nd ed.)
0201568853

Organización Docente de la Asignatura

Contenidos	Horas					Competencias y Objetivos
	HT	HP	HCT	HTT	HAI	
Tema 1	0,0	0,0	0,0	6,0	1,0	1.1, 1.2, 1.3, 1.4, 3.1
Tema 2	0,0	0,0	0,0	3,0	0,5	1.4, 2.1, 2.2, 2.3
Tema 3	0,0	0,0	0,0	16,5	0,0	1.4, 2.1, 2.2, 2.3

Contenidos	Horas					Competencias y Objetivos
	HT	HP	HCT	HTT	HAI	
Tema 4	0,0	0,0	0,0	12,0	2,0	1.4, 2.1, 2.2, 2.3
Tema 5	0,0	0,0	0,0	6,0	1,0	1.4, 2.1, 2.2, 2.3
Tema 6	0,0	0,0	0,0	6,0	1,0	1.4, 2.1
Tema 7	0,0	0,0	0,0	12,0	2,0	1.5, 1.6
Tema 8	0,0	0,0	0,0	5,0	2,0	1.6
Tema 9	0,0	0,0	0,0	5,0	1,0	1.6
Tema 10	0,0	0,0	0,0	5,0	1,0	1.6, 3.2

Equipo Docente

FRANCISCO JAVIER MIRANDA GONZÁLEZ

(COORDINADOR)

Categoría: TITULAR DE UNIVERSIDAD

Departamento: INGENIERÍA TELEMÁTICA

Teléfono: 928451240 **Correo Electrónico:** javier.miranda@ulpgc.es

WEB Personal: <http://www.iuma.ulpgc.es/users/jmiranda>

Resumen en Inglés

Descriptor

Languages design: concepts and paradigms: values, storage, declarations, scopes, types, abstraction. Imperative programming, concurrent programming, object-oriented programming. Language processors: translators and interpreters: scanner, parser, semantic analyzer and code generator.

Requirements

The student should have passed the subjects Fundamentos de Programación and Programación of the first course of the ETSIT (or subjects with similar contents) to successfully follow this course. It is also recommended to have passed most of the subjects of the third course of the ETSIT, and be familiar with the use of some variant of the UNIX Operating System (for example, Linux) at the user level.

Methodology:

- The instructor will present the theoretical concepts in class.
- The instructor will propose exercises to facilitate understanding such concepts.
- The students will write a language processor in the laboratory to understand the implementation of these concepts.

- The students will have further information in the Virtual Campus Server of this University (available at <http://www.ulpgc.es>).

Evaluation:

- The defense of a technical report has a maximum value of 20% of the final punctuation.
- The compiler developed by each student in the laboratory has a maximum value of 80% of the final punctuation.

The final punctuation is calculated adding these two values. Hence, it is not required to pass the two parts to pass the whole subject.

Laboratory Assignments

We will work in the Laboratory of Architectures of the Departamento de Ingenieria Telematica. The goal of this practical work is to know implementation details of language processors. Each student will design an imperative programming language (or a subset of some well known language) and implement a small compiler for that language.