Departamento de INGENIERÍA y CIENCIA de los COMPUTADORES

IS04. Metodología y Tecnología de la Programación

PRIMER CURSO. ING. TÉCN. en INFORMÁTICA de SISTEMAS

Escuela Superior de Tecnología y Ciencias Experimentales

Curso 2006/2007.

Anual.

15 créditos.

Profesores:

López Malo, Ángeles,

Martínez Vidal, Gloria,

Ortíz Gómez, Mª del Carmen,

Rubio Moreno, Rafael.

Índice

1. Introducción. **1**

2. Temario y Objetivos. **1**

2.1. Parte I: Introducción al Diseño de Algoritmos. 1

2.2. Parte II: Gestión básica de la información. 3

2.3. Parte III: Introducción a la Algorítmica. 4

2.4. Temario de Prácticas. 4

2.5. Comentarios al Temario. 5

3. Bibliografía. **6**

3.1. Bibliografía básica. 6

3.2. Bibliografía de ampliación. 6

4. Metodología. **7**

5. Evaluación. **7**

1. Introducción.

El desarrollo de *software* es una de las tareas más importantes dentro del ámbito de la informática. En esta asignatura se pretende sentar las bases que permitan introducirse en el mundo de la programación, aprendiendo a construir algoritmos que permitan resolver problemas concretos.

Es un error frecuente concebir la programación como una labor que consiste en aprender la sintaxis de un lenguaje y, a partir de ahí, adquirir una experiencia, mediante el "entrenamiento", que permita ir generando programas más y más complejos.

La visión fundamental en esta asignatura será que la programación requiere de una técnica, cuya aplicación se debe intentar sistematizar al máximo, al igual que se hace en cualquier otra rama de la ingeniería. Ello permitirá que se incremente la productividad en el desarrollo de algoritmos y programas y, además, pensando en el trabajo a realizar por un futuro profesional de la informática, permitirá sentar las bases de técnicas más avanzadas de planificación, análisis y desarrollo de sistemas informáticos.

En la asignatura se utilizará el llamado *paradigma de programación imperativa*, debido a su mayor uso. Las clases teóricas desarrollarán los principales conceptos de dicha programación, con el objetivo de poder concebir algoritmos válidos, en principio, para cualquier lenguaje de programación imperativo. Las clases prácticas (problemas y laboratorio) tendrán como objetivo implementar dichos algoritmos generales en lenguajes de programación concretos. Estos lenguajes, durante el presente curso, serán C y Python.

2. Temario y Objetivos.

La asignatura tiene una carga lectiva de 15 créditos (9 en clases de teoría y problemas y 6 de prácticas en laboratorio). Se ha dividido sus contenidos en tres partes, que se desarrollan a continuación.

2.1. Parte I: Introducción al Diseño de Algoritmos.

Objetivos teóricos: Al finalizar esta parte el alumno *debería*,

1. conocer qué es un algoritmo y qué es un programa, y sus etapas básicas de desarrollo,
2. conocer qué objetos forman el entorno de un algoritmo y cómo manipularlos: constante/variable/parámetro,
3. conocer cuáles son los tipos básicos y las expresiones que permiten obtener valores de cada tipo, y el concepto de coherencia de tipos,
4. conocer qué son las estructuras de control de flujo de datos, cómo funcionan y cuándo es correcto utilizar cada una de ellas,
5. conocer qué es el diseño descendente y la especificación básica de un algoritmo, además de distinguir entre especificación y diseño,
6. conocer los conceptos básicos de la descomposición modular, la definición de subprogramas y de bibliotecas,

7. conocer los esquemas básicos de recorrido y búsqueda en estructuras de datos lineales.

Objetivos prácticos: Al finalizar esta parte el alumno *debería*,

1. ser capaz de manipular correctamente variables: cómo asignar valores, bien mediante expresiones, bien mediante acciones interactivas del usuario, y en qué orden,
2. ser capaz de definir completamente el entorno de los algoritmos que diseñe y de manipular correctamente cada objeto del entorno,
3. ser capaz de interpretar correctamente el funcionamiento de una secuencia de instrucciones,
4. ser capaz de manejar correctamente las estructuras de control de flujo de datos, utilizando la programación estructurada,
5. ser capaz de identificar la especificación de Datos/Resultados de los algoritmos, y de identificar el intercambio de comunicación entre ellos,
6. ser capaz de realizar el proceso básico de arrays y tuplas y de recorrer correctamente arrays de una dimensión,
7. ser capaz de plasmar adecuadamente lo anterior en un entorno de programación real; es decir, sabrá traducir dichos conceptos a algún lenguaje de programación.

Temario teórico:

TEMA 1: Introducción.

1. Conceptos básicos de programación.
2. La Programación como disciplina de ingeniería.

TEMA 2: Conceptos Básicos.

1. Variables. Tipos básicos y expresiones.
2. Concepto de estructura secuencial. Flujo de instrucciones. Trazas.
3. Estructura básica de un programa. Bibliotecas estándar. Fase de compilación.

TEMA 3: Programación Estructurada.

1. Introducción a la programación estructurada. Importancia de los predicados lógicos
2. Estructura condicional. Tipos de condicionales.
3. Estructuras iterativas.
4. La Lógica de las estructuras de control.

TEMA 4: Introducción al Diseño Descendente.

1. Estructuración del trabajo y reutilización del trabajo.
2. Especificación básica de algoritmos.
3. Concepto de parámetro.
4. Definición de subprogramas e intercambio de información.
5. Descomposición modular. Definición de bibliotecas.

TEMA 5: Estructuras de Datos Estáticas.

1. Necesidad de las estructuras de datos. Estructuras homogéneas. Estructuras heterogéneas.
2. Algoritmos de manipulación de las estructuras de datos.
3. Influencia de las estructuras de datos en el diseño de algoritmos.

2.2. Parte II: Gestión básica de la información.

Objetivos teóricos: Al finalizar esta parte el alumno *debería*,

1. haber reforzado la noción de tipo, demostrando conocimiento de los tipos básicos y los tipos estructurados elementales, así como de las operaciones que permiten manipularlos.
2. haber reforzado los conceptos básicos de la descomposición modular, la definición de subprogramas y de bibliotecas.
3. conocer qué es la gestión dinámica de la memoria, el concepto de puntero y los tipos enlazados básicos,
4. conocer qué es un fichero y su gestión básica.
5. conocer la existencia de distintos estilos de programación y nociones básicas de POO.

Objetivos prácticos: Al finalizar esta parte el alumno *debería*,

1. ser capaz de utilizar y definir módulos y una biblioteca de programas,
2. ser capaz de manejar correctamente arrays de cualquier dimensión y tuplas,
3. ser capaz de manejar correctamente el tipo puntero y las estructuras dinámicas básicas,
4. ser capaz de realizar operaciones básicas de ficheros,
5. ser capaz de realizar la documentación básica de los programas realizados,
6. ser capaz de plasmar adecuadamente lo anterior en un entorno de programación real; es decir, sabrá traducir dichos conceptos a algún lenguaje de programación.

Temario teórico:

TEMA 6: Estructuras de Datos Dinámicas.

1. Memoria dinámica. Concepto de puntero.
2. Estructuras enlazadas.
3. Influencia de las estructuras de datos en el diseño de algoritmos.

TEMA 7: Introducción al Uso de Ficheros.

1. Ficheros. Manipulación básica

TEMA 8: Otros estilos de programación.

1. Scripts.
2. Introducción a la programación Orientada a Objetos.
3. Introducción a la programación Orientada a Eventos.

2.3. Parte III: Introducción a la Algorítmica.

Objetivos teóricos: Al finalizar esta parte el alumno *debería*,

1. conocer qué diferencias hay entre un algoritmo iterativo y un algoritmo recursivo,
2. conocer qué es la complejidad computacional y cómo comparar algoritmos,
3. conocer los esquemas básicos de ordenación en estructuras lineales.

Objetivos prácticos: Al finalizar esta parte el alumno *debería*,

1. ser capaz de interpretar correctamente el funcionamiento de algoritmos recursivos, y de diseñar algoritmos recursivos simples,
2. ser capaz de evaluar casos simples de complejidad computacional,
3. ser capaz de utilizar correctamente algoritmos elementales de ordenación y búsqueda, y de adaptarlos correctamente a instancias concretas.

Temario teórico:

TEMA 9: Recursividad.

1. Concepto de algoritmo recursivo.
2. Coste de la recursividad.
3. Utilidad de la recursividad. Tipos de recursividad.

TEMA 10: Introducción al Análisis de Algoritmos.

1. Concepto de complejidad computacional. Talla de un problema.
2. Coste en el mejor y en el peor de los casos.
3. Comportamiento asintótico y orden de coste. Problemas intratables.
4. Algoritmos elementales de ordenación.

2.4. Temario de Prácticas.

Parte 1 20 horas (2 créditos). Lenguaje de Programación: C.

- Práctica 1: Condicionales y Bucles I (4 horas).
- Práctica 2: Condicionales y Bucles II (4 horas).
- Práctica 3: Funciones y Procedimientos (4 horas).
- Práctica 4: Vectores (I) (4 horas)
- Práctica 5: Vectores (II). Introducción a los módulos (4 horas).

Parte 2 28 horas (2,8 créditos). Lenguaje de Programación: C, si bien se realizarán prácticas puntuales en Python.

- Práctica 6: Matrices (4 horas).
- Práctica 7: Punteros (I) (4 horas).
- Práctica 8: Punteros (II) (4 horas).
- Práctica 9: Punteros (III). Definición de un módulo para listas enlazadas (4 horas).
- Práctica 10: Ficheros (4 horas).
- Práctica 11: Introducción a Python (4 horas).
- Práctica 12: Programación con eventos en Python (4 horas).

Parte 3 12 horas (1,2 créditos). Lenguaje de Programación: C o Python.

- Prácticas 13 y 14: Recursividad (8 horas).
- Práctica 15: Análisis de Algoritmos de Ordenación (4 horas).

2.5. Comentarios al Temario.

El temario se ha dividido en tres partes, cada una de ellas con una orientación específica.

La primera parte está orientada al conocimiento de las bases del diseño algorítmico, el manejo de las variables de tipo básico y el conocimiento de las estructuras de control de flujo que se identifican en *Programación Estructurada*.

Para dar mayor énfasis al uso de las estructuras de control y al diseño del algoritmo, se propone que el trabajo durante esta parte sea principalmente teórico, orientado hacia el diseño del algoritmo, así como en la identificación, en cada problema, de Datos y Resultados. De esta forma, al comenzar las sesiones prácticas se habrá realizado ya un buen número de algoritmos: el objetivo de esta primer bloque de prácticas será, entonces, familiarizarse con un lenguaje de programación concreto, y su entorno de trabajo, verificando además el funcionamiento de los diseños realizados previamente. Se pretende también que el alumno comprenda la importancia de desarrollar un algoritmo de forma previa a su implementación como programa.

La segunda parte pretende reforzar la primera y, además, introducir al alumno en el tratamiento de estructuras de datos fundamentales y en la formalización del entorno de programación. Se pretende también hacer especial hincapié en el uso y definición de bibliotecas de programas.

La elección del lenguaje C se ha hecho, principalmente, atendiendo al perfil curricular de la Ingeniería Técnica en Informática de Sistemas, además de por ser uno de los lenguajes de más amplia difusión hoy en día. Además, al final de esta segunda parte se introducirán las nociones básicas del lenguaje Python. Este lenguaje, interpretado, tiene una sintaxis similar al lenguaje C; el objetivo es presentar al alumno un lenguaje de gran difusión en los últimos tiempos, principalmente por su versatilidad a la hora de realizar fácilmente scripts y al permitir realizar una introducción a cuestiones relacionadas con programación orientada a objetos, programación orientada a eventos, programación CGI,etc. Por lo tanto, en el futuro le puede ser de utilidad.

La tercera parte tiene como objetivo introducir aspectos formales de programación. En ella se presentan las nociones fundamentales de *recursividad* y las definiciones básicas de *complejidad computacional* que un informático debe conocer. Así, esta tercera parte podría servir también como introducción si algún alumno deseara ampliar su formación en programación.

En la descripción del temario, se han ido marcando los objetivos que pretenden cubrir cada una de estas tres partes. Los objetivos indicados hasta el momento son los objetivos de *conocimiento* (teóricos) y los objetivos de *destrezas* (prácticos). Tal y como se indica en la sección 5, Evaluación, es muy importante recalcar estos objetivos puesto que a ellos se referirán las cuestiones y problemas del examen. Pero, además, a estos objetivos hay que añadir dos objetivos fundamentales de *actitud*: a lo largo de la asignatura se intentará que el alumno no sólo adquiera conocimientos o destrezas, también

- debe asumir la importancia y necesidad de trabajar siguiendo las directrices de la programación estructurada,
- debe adoptar la costumbre de realizar un algoritmo como paso previo y fundamental en la elaboración de programas.

3. Bibliografía.

3.1. Bibliografía básica.

- Apuntes de la asignatura. Disponibles en Reprografía y en la página web de la asignatura, `http://www.unoweb-s.uji.es/IS04`.
- "El Lenguaje de Programación `C`. Diseño e Implementación de Programas". Félix García, Jesús Carretero, Javier Fernández & Alejandro Calderón. Pearson Education, Madrid 2002.
- "Introducción a la Programación. Tomo I: Algorítmica y Lenguajes"[QA76.6.B5618 1985]. Joëlle Biondi & Gilles Clavel. Ed. Masson, S.A. 1985.
- "Algorithms in C (parts I-IV)" [QA76.73.C15 S43 1998]. Robert Sedgewick. Ed. Addison-Wesley. 1998.
- "How to Think like a Computer Scientist". Allen B. Downey, Jeffrey Elkner & Chris Meyers. 2001. Versiones en Python, C++ y Java. `http://www.ibiblio.org/obp/thinkCS/`.
- "The C programming lenguage"[QA76.73.C15 K47 1988]. Brian W. Kernigham & Dennis M. Ritchie. Prentice-Hall Inc. 1988. Existe una versión en castellano del año 1991.

3.2. Bibliografía de ampliación.

- "The Art of Computer Programming (vol. 1)"[QA76.6 .S43 1988]. Donald E. Knuth. Ed. Addison-Wesley. 1973. Texto fundamental de programación, pero la exposición ha quedado un poco anticuada. Hay una versión en castellano, [QA76.6.K5818 1986].
- "Introduction to Algorithms. A Creative Approach"[QA76.9.D35 M36 1989]. Udi Manber. Ed. Addison Wesley. 1989. Lectura avanzada.
- "Fundamentos de Algoritmia"[QA9.58.B7318 1997]. Brassard & Bratley. Ed. Prentice Hall. 1997. Lectura avanzada.
- "Compared to What? An Introduction to the Analysis of Algorithms". [QA76.9.A43 R39 1992] G.J.E. Rawlins. Ed. Computer Science Corp. 1992. Lectura avanzada.
- "The Science of Programming"[QA76.6.G75 1981]. David Gries. Ed. Springer-Verlag. 1981. Lectura muy avanzada.

4. Metodología.

La asignatura está dividida en 9 créditos de clase teórica y 6 de laboratorio. En las clases teóricas se desarrollarán los contenidos del temario, intentando ofrecer un número importante de ejemplos y problemas solucionados. Además, servirán para proponer nuevos problemas, algunos de los cuales se desarrollarán en el laboratorio de prácticas; pero también se propondrán problemas que el alumno debe intentar solucionar por sí mismo. Para potenciar esto, cada tema de teoría incluye una colección de problemas.

Se realizarán 15 prácticas, cada una dividida en dos sesiones de dos horas. En cada práctica se propondrán una serie de problemas obligatorios a resolver. Dependiendo de la práctica concreta, cabe la posibilidad de ofrecer además una serie de propuestas voluntarias. En cada práctica se indicará al alumno qué cuestiones debe entregar por escrito de forma previa a su realización y qué cuestiones debe entregar, bien por escrito bien como programa ejecutable, al finalizarla.

En una asignatura como ésta, es importante el trabajo regular y, además, personal. Se trata de aprender a diseñar algoritmos y a construir programas. En ningún momento se va a exigir la asistencia obligatoria a ninguna clase; pero, evidentemente, es un aspecto que ayudará a cualquier alumno de cualquier disciplina técnica.

En este sentido, cabe destacar la importancia de las tutorías, no sólo para solucionar las dudas surgidas en clase sino también para corregir los trabajos realizados de forma voluntaria.

5. Evaluación.

Para la evaluación se tendrá en cuenta tanto la nota obtenida en los exámenes como la nota obtenida por un correcto aprovechamiento de las prácticas y la entrega periódica de ejercicios y pequeñas cuestiones (lo que en adelante se denominará *"entregables"*). La asignatura queda superada si se obtiene una calificación final superior o igual a 5 en la convocatoria de Junio, o bien en la de Septiembre.

Fechas y Objetivos de los exámenes. Las fechas previstas para los exámenes son:

- Examen parcial (EP), 23 de Enero de 2007.
- Examen final de Junio (EF), 15 de Junio de 2007.
- Examen de Septiembre (SEP), 3 de Septiembre de 2007.

En cualquiera de estos tres exámenes se podrán utilizar los apuntes de la asignatura. El principal objetivo será medir la capacidad de razonamiento en el diseño de la solución algorítmica de problemas; además, también habrá cuestiones de tipo teórico que permitan evaluar el conocimiento de los contenidos fundamentales. Como ya se ha comentado, en general los problemas y cuestiones se **referirán a los objetivos teóricos y prácticos marcados en el temario** de la asignatura.

Prácticas. El correcto aprovechamiento de la asistencia a prácticas permitirá obtener un máximo de *3 puntos*.

Por *"un correcto aprovechamiento de las prácticas"*, se entiende que el alumno tendrá que entregar **puntualmente** el trabajo previo y posterior de **todas** y **cada una** de las 15 sesiones prácticas. El trabajo previo habrá de entregarse aproximadamente *10 días* antes de la realización de la sesión de prácticas[1] y el trabajo posterior *al final* de las 4 horas de una sesión de prácticas. Para cada práctica se calificará el trabajo previo, la asistencia a las sesiones y la calidad del trabajo final entregado, así como el cumplimiento de las actividades complementarias propuestas en cada práctica (por ejemplo, hay prácticas en las que se pide una crítica constructiva del trabajo de un compañero). La calificación de las prácticas se irá publicando periódicamente a lo largo del curso, de forma que antes antes del examen final se conocerá la calificación obtenida en prácticas.

Convocatoria de Junio. Con el objetivo de realizar una evaluación continuada de la asignatura, para obtener la calificación de la convocatoria de Junio, se tendrán en cuenta:

- Examen parcial (EP), máximo de 1 punto,
- Examen final de Junio (EF), máximo de 5 puntos,
- Nota de las prácticas (PR), máximo de 3 puntos,
- Nota de los "entregables", máximo de 1 punto.

La calificación, por regla general, será la suma de todas estas notas.

Convocatoria de Septiembre. Para obtener la calificación de la convocatoria de Septiembre, se tendrán en cuenta:

- Examen final de Septiembre (SEP), máximo de 6 puntos,
- Nota de las prácticas (PR), máximo de 3 puntos,
- Nota de los "entregables", máximo de 1 punto.

La calificación será la suma de estas notas, y no se tendrá en cuenta la nota del examen parcial.

¿Qué ocurre si no se realizan las prácticas o los "entregables" o el primer parcial? De lo anterior se desprende que es posible aprobar la asignatura sin realizar las prácticas, ni los "entregables", ni el primer parcial si se alcanzan los 5 puntos en el examen de Junio o Septiembre, pero nótese entonces cuál es la nota máxima a la que se podrá aspirar.

[1] Al publicarse el cuaderno de prácticas correspondiente al curso 2005/06, se hará público el calendario completo de entregas de cada práctica.