La asignatura Introducción a los ordenadores en el contexto de la experiencia Áncora, en la ETSETB-UPC

Beatriz Otero, Pau Bofill
Dept. Arquitectura de Computadors
Universitat Politècnica de Catalunya
08034 Barcelona
e-mail: {botero,pau}@ac.upc.edu

Resumen

Este artículo describe el proyecto docente de la asignatura Introducción a los ordenadores (programación en C), del cuatrimestre 1A de Ingeniería de Telecomunicación (ETSETB-UPC), en el contexto de la experiencia *Áncora* de aprendizaje organizado por tareas.

El curso se plantea en tres etapas de forma que, en vez de acumular, el estudiante construya sobre lo que ha aprendido en la etapa anterior. El artículo describe los objetivos de la asignatura organizados por etapas, así como las acotaciones y los recursos del lenguaje utilizados en cada etapa.

En el contexto de la experiencia Áncora, la asignatura se ofrece como una materia instrumental de tipo transversal, en el que los programas realizados por los estudiantes son de utilidad en las demás asignaturas. Para cada etapa, se formula una tarea, a desarrollar en equipo, que cubre los objetivos correspondientes.

El artículo describe también la metodología de trabajo cooperativo utilizada, y la estrategia de evaluación formativa. En su conjunto, la propuesta fomenta la diversidad de estilos de aprendizaje y permite a los estudiantes adquirir experiencia en habilidades transversales como el trabajo en equipo o la gestión de tareas.

1. Introducción

Este artículo describe el proyecto docente de la asignatura Introducción a los ordenadores (IO), del cuatrimestre 1A de Ingeniería de Telecomunicación (ETSETB-UPC) [**?**], en el contexto de la experiencia *Áncora* de aprendizaje organizado por tareas [**?**].

La experiencia Áncora es un proyecto de innovación educativa que engloba todas las asignaturas troncales del primer cuatrimestre (Álgebra, Cálculo, Fisca, Circuitos Electrónicos e Introducción a los ordenadores). La experiencia se realiza solamente en uno de los seis grupos del primer cuatrimestre y, en lugar de las clases expositivas habituales, pretende recrear las actividades propias de un entorno laboral, con una sala de trabajo, una jornada completa, y un contrato. La actividad de los estudiantes se organiza mediante tareas, que son pequeños proyectos o problemas abiertos que conducen a la consecución de los objetivos formativos. El trabajo se realiza fundamentalmente en equipo (5 estudiantes) y el rol principal de los profersores es dinamizar las actividades del día a día, hacer un seguimiento del trabajo de los equipos, y ofrecer realimentación personalizada. Áncora se imparte como experiencia piloto en uno de los 6 grupos de primero, con 60 estudiantes como los demás grupos, y a coste 0. En [**?**] se describe la experiencia con detalle. La primera edición de la experiencia Áncora se llevará a cabo durante el cuatrimestre de primavera de 2005.

Los *objetivos globales* de IO son los siguientes: Introducción a la progamación en lenguaje C e implementación de pequeñas aplicaciones. Noción de ordenador y sus bloques funcionales, y nociones de representación de datos.

En nuestra propuesta IO pone el énfasis en la programación por encima de la algorítmica, y se promueve la programación modular y constructiva, usando funciones desde el primer día.

En el plan de estudios actual IO tiene asignados 6 créditos. Esto significa, incluyendo las horas de trabajo personal, unas 120 horas (4 créditos ECTS).

2. Enfoque y articulación de contenidos por etapas

La mayoría de libros y cursos de programación siguen un enfoque temático en la secuenciación de contenidos, incluso cuando son cursos prácticos. Se empieza por expresiones, luego estructuras algorítmicas, luego estructuras de datos, punteros, funcio-

nes, ficheros, etc. Después de cada tema únicamente se pueden hacer ejercicios que utilicen los elementos estudiados, de forma que se construyen programas que más adelante habrá que reescribir (para introducir funciones, por ejemplo).

Un enfoque alternativo es estructurar el curso *por etapas*, de forma que en cada etapa se utilice un subconjunto consistente del lenguaje y en la etapa siguiente se construya sobre lo aprendido, fijando los conocimientos y ampliando el repertorio a un subconjunto más extenso. El énfasis en cada etapa se pondrá entonces en *aprender a programar* con un conjunto de recursos dado, por contraposición a aprender los elementos del lenguaje.

En el caso de IO proponemos tres etapas. El objetivo de la primera etapa es introducir al estudiante en la lógica de la programación. Para los iniciados, la programación de ordenadores parece una disciplina fácil porque sigue una lógica interna muy coherente y no requiere la comprensión de conceptos o teorías complejos. Pero la dificultad estriba, precisamente, en entrar en esta lógica. La causa fundamental de fracaso en IO es la dificultad de comprender la relación entre un fragmento de código y lo que ocurrirá cuando se ejecute. Por esto, cuando estos estudiantes tienen que codificar un programa, por sencillo que sea, el resultado es a menudo incoherente. En nuestro caso, el objetivo principal de la primera etapa será que todos los estudiantes consigan iniciarse en la lógica de la programación.

Una vez dentro, programar es como un juego de construcciones. Se trata de conseguir un resultado a partir de un repertorio de elementos o piezas. El objetivo de la segunda etapa es conocer el repertorio de piezas, y las posibilidades de cada pieza, y saber como combinarlas para conseguir el resultado deseado. No se trata de construir programas excesivamente complejos (demasiadas piezas), sino programas sencillos bien estructurados (con una buena selección y combinación de piezas). En nuestro caso, en esta etapa hay que aprender la semántica en relación a la máquina de cada uno de los elementos del lenguaje, y la correcta construcción de estructuras de datos, con sus operaciones (funciones) asociadas. Las aplicaciones a desarrollar serán todavía muy sencillas.

En la tercera etapa ponemos todos los recursos en juego y nos centramos en el desarrollo de una aplicación. Se trata de definir un pequeño proyecto, acotado y, con asistencia por parte del profesor, abordar su diseño e implementación. El objetivo en este caso es resolver el problema a partir de piezas conocidas, al mismo tiempo que se consolida el manejo de dichas piezas.

3. Objetivos por etapas

Los objetivos de Introducción a los ordenadores se definen aquí en 3 etapas. Para cada etapa se definen los objetivos generales y específicos [**?**], se enumeran los contenidos temáticos (el subconjunto del lenguaje que se va a utilizar), y se da una acotación al nivel de complejidad de los programas a realizar en dicha etapa.

3.1. Primera etapa: gráficos de tortuga y expresiones

El objetivo de esta etapa es comprender el concepto de "programabilidad" (que el comportamiento de un ordenador depende de un programa), y descubrir que programar es *fácil*. Los gráficos de tortuga facilitan un entorno muy interactivo en el que el estudiante "ve" con claridad el resultado de sus esfuerzos y aprende a corregir sus propios errores de forma intuitiva. La "extensibilidad" del lenguaje mediante la definición de funciones aparece de forma natural. La evaluación de expresiones matemáticas permite de forma sencilla la integración con otras materias del curso, al tiempo que introduce al estudiante al cálculo con operadores y la gestión de pantalla y teclado.

Objetivos prácticos. Realizar gráficos de tortuga, evaluar expresiones matemáticas, etc.

Objetivos formativos generales

- Escribir e interpretar programas sencillos mediante el subconjunto de elementos del lenguaje descrito más adelante.
- Usar funciones constructivamente para resolver problemas de mayor complejidad.
- Identificar el conjunto de primitivas disponible para un problema (librerias, en el sentido descrito más adelante).
- Construir librerías con las funciones primitivas asociadas a una familia de problemas.

Objetivos formativos específicos

- Editar, compilar y ejecutar programas.
- Identificar variables y sentencias.
- Relacionar el comportamiento de un fragmento de código con la secuencia de sentencias que lo describe.
- Escribir un fragmento de código con un comportamiento especificado.
- Seguir la ejecución de un programa paso a paso.
- Identificar los bloques funcionales de un programa (las funciones).
- Dado un gráfico de tortuga, escribir una función que lo realice y un programa principal para probarla.
- Dado un programa correspondiente a un gráfico de tortuga, anticipar el gráfico resultante.
- Dada un expresión matemática, escribir una función que permita evaluarla y un programa principal para probarla.
- Dado un programa correspondiente a una expresión matemática conocida, identificar de qué expresion se trata.
- Construir librerías con las funciones primitivas asociadas a una familia de problemas.
- Utilizar un repertorio de funciones dado (una o más librerías) para escribir un programa o función.

Los contenidos temáticos en esta etapa son los siguientes: Entorno de programación Visual C++. Modo consola (excepto los gráficos de tortuga para los que se ofrece un entorno predefinido). Primitivas gráficas de tortuga (Inicializa, Finaliza, Avanza, Gira, SubeLapiz, BajaLapiz). Operadores básicos del lenguaje (+, -, *, /, %). Variables y tipos elementales. Secuenciamiento de sentencias. Sentencia de asignación. Entrada y salida simple por teclado y pantalla (scanf, printf). Selección IF. Iteración FOR. Llamada a funciones con parámetros por valor y resultado funcional (la llamada a funciones se interpreta como una sentencia elemental). Definición de funciones con parámetros por valor y resultado funcional.

Los programas que se realicen en esta etapa deben cumplir las acotaciones siguientes: Fragmentos de código de no más de 5 o 6 líneas. Funciones de no más de 10 líneas de código. Expresiones con un máximo de 6 operadores. No más de 2 ó 3 sentencias IF en una misma función. No más de 2 sentencias FOR por programa o función, sin anidar. Combinadas con 1 ó 2 sentencias IF. Sin BREAK.

3.2. Segunda etapa: estructuras de datos y sus operaciones asociadas

El objetivo de esta etapa es conocer el comportamiento de los elementos del lenguaje y su combinación para construir estructuras sencillas. Con este objetivo se construirá un conjunto de primitivas asociado a una estructura de datos no trivial (por ejemplo, listas), introduciendo paulatinamete más complejidad en los datos, punteros para el paso por referencia y ficheros.

En esta etapa también se hará hincapié en la semántica de datos y sentencias.

Objetivos prácticos. Implementación de un conjunto de datos estructurado y sus operaciones asociadas (típicamente, variantes del tipo abstracto lista como las cartas de la baraja, o los coeficientes de un polinomio).

Objetivos formativos generales

- Comprender la estructura del ordenador y sus bloques funcionales.
- Comprender la semántica de datos (mapa de memoria), sentencias y llamadas a funciones.
- Escribir programas sencillos que utilicen las operaciones primitivas de la estructura de datos elegida.
- Realizar las operaciones primitivas de la estructura de datos elegida.

Objetivos formativos específicos

- Describir los bloques funcionales de un ordenador y sus funciones.Relacionarlos con los recursos del lenguaje.
- Depurar un programa utilizando el entorno de programación.

- Dada una declaración de tipos y variables, dibujar el mapa de memoria correspondiente, y viceversa.
- Dado un mapa de memoria, escribir la declaración de tipos y variables asociado a dicho mapa.
- Dada la descripción textual de un conjunto de datos, especificar los tipos de datos adecuados y las variables necesarias para representarlos.
- Seguir la traza de ejecución de un fragmento de código sobre el mapa de memoria asociado a las variables activas.
- Seguir la traza de paso de parámetros y resultados en una llamada a una función.
- Identificar los parámetros, resultados, entradas y salidas de una función.
- Describir la interficie y funcionalidad de las operaciones primitivas de gestión de listas o la estructura de datos elegida, incluyendo lectura y escritura a fichero.
- Escribir un programa (o función) que utilice las operaciones primitivas de gestión de listas.
- Dado un tipo de datos, realizar las operaciones primitivas asociadas a una lista de datos de dicho tipo.

A los contenidos de la etapa anterior, en esta etapa se añaden los siguientes temas: Estructura del ordenador: bloques funcionales. Mapas de memoria. Trazas y diagramas de flujo. Tipos estructurados (vectores, structs). Operadores lógicos y relacionales. Sentencias SWITCH, WHILE, DO..WHILE. Implementación del paso de parámetros por referencia con punteros (en punteros, *únicamente* lo necesario para comprender el paso de parámetros). Búsqueda y recorrido en un vector. Primitivas de la gestión de ficheros de texto (fopen, fclose, fscanf, feof, fprintf, fflush).

La acotación del código en esta etapa es la siguiente: Programas: código organizado por funciones de no más de 15 líneas (aprox). Máximo 3 niveles de anidamiento en los tipos estructurados. Máximo 3 niveles de anidamiento explícito entre iteraciones y selecciones en una misma función. Punteros sólo para el paso de parámetros por referencia.

3.3. Tercera etapa: diseño y realización de aplicaciones

El objetivo de esta etapa es iniciarse en la programación de un problema real, a partir de su especificación funcional y de la descomposición en tipos y funciones.

Paralelamente, esta etapa tiene también por objetivo conocer los principios básicos de representación de datos.

Objetivos prácticos. Realizar aplicaciones sencillas relacionadas preferentemente con alguna de las demás materias del cuatrimestre.

Objetivos formativos generales

- Codificar caracteres, enteros y reales en los formatos ascii, complemento a 2 y coma flotante, respectivamente.
- Dada la descripción de un problema algorítmico, diseñar un algoritmo para resolverlo, codificarlo y probarlo.
- Escribir un programa completo a partir de su descomposición en funciones y tipos de datos.
- Conocer la complejidad organizativa del software y comprender los principios de la organización modular.

Objetivos formativos específicos

- Interpretar el código ASCII para la codificación de caracteres y distinguir dígitos de números.
- Traducir un entero de decimal a ca2-N y viceversa.
- Convertir un real a coma flotante IE3 y viceversa.
- Dado un problema, describir el algoritmo para resolverlo.
- Dado un algoritmo, escribir un programa para codificarlo.
- Dado un programa, utilizar herramientas de depuración para verificar su correcto funcionamiento.
- Proponer problemas y aplicaciones sencillos.

- Escribir un programa o función concreto a partir de uno o varios conjuntos de primitivas y sus datos asociados.
- Dada la especificación de sus funciones y tipos de datos, realizar un conjunto de primitivas.
- Dada la descomposición en funciones y tipos de datos, escribir un programa para realizar una aplicación completa.

Adicionalmente, en esta etapa se incluyen los contenidos temáticos siguientes: Representación de tipos de datos. Primitivas de gestión de strings.

La acotación en esta etapa es la siguiente: Programas: código organizado por funciones de no más de 20 líneas (aprox.). Máximo 4 niveles de anidamiento en los tipos estructurados. Máximo 3 niveles de anidamiento explícito entre iteraciones y selecciones en una misma función. Problemas y algoritmos: problemas que puedan ser codificados mediante un máximo de 2 ó 3 funciones que, en conjunto, no impliquen más de 4 niveles de anidamiento. Conjuntos de primitivas: prototipo y descripción detallada de cada una de sus funciones. No más de 10 primitivas. Aplicaciones: programas completos con unas 20 funciones, acotadas según lo dicho anteriormente.

4. Las tareas de Introducción a ordenadores en Áncora

En el entorno de la experiencia Áncora, IO se ofrece como una disciplina de caracter transversal, en el sentido de que los programas realizados pueden estar al servicio de cualquiera de las otras materias del curso, como herramienta de cálculo, para ilustrar algún concepto o para realizar simulaciones.

En este apartado proponemos tareas para cubrir los objetivos formativos de cada una de las tres etapas. La primera tiene una duración de 24h ECTS, y la segunda y la tercera una duración de 32h ECTS cada una. Las horas restantes hasta las 120 del curso se dedicarán a la puesta a punto inicial, y a la preparación de exámenes al final de curso.

4.1. Primera tarea: formulario para el cuatrimestre 1A

Esta tarea consiste en la realización en equipo de un formulario para las demás asignaturas del cuatrimestre: Álgebra, Cálculo, Circuitos electrónicos y Física.

Especificación. Para cada una de las fórmulas que se utilicen en las asignaturas mencionadas debe escribirse uno o más programas que, reiteradamente, dadas las variables independientes calcule el valor de las variables dependientes. El programa pedirá los datos al usuario y mostrará los resultados en la pantalla.

Restricciones para la realización. Cada programa debe pedir los datos y mostrar los resultados desde el programa principal. Hay que utilizar funciones con los nombres adecuados para la realización de los cálculos y utilizar por lo menos una vez las sentencias IF y FOR, pero sin anidaciones.

Fases de elaboración y metodología. No todos los miembros del equipo tendrán la misma facilidad en realizar estos programas. La tarea del equipo no finaliza hasta que *cada uno* sea capaz de programar por lo menos 3 fórmulas individualmente, y hasta que el formulario este completo.

Resultados y formato de presentación. Los programas se irán incorporando al formulario de cada estudiante o equipo, y se pondrán a prueba para realizar ejercicios de las demás materias.

Estrategia de valoración y mejora. Los programas desarrollados se verificarán a nivel de funcionamiento y se analizarán a nivel de código fuente mediante la comparación con modelos. Los programas que no cumplan los requisitos de estructuración tendrán que repetirse.

4.2. Segunda tarea: una estructura de datos y sus operaciones

Para esta segunda etapa sugerimos diferentes objetivos, a elegir, que incluyen estructuras de datos y sus operaciones asociadas. Estas estructuras de datos estarán relacionadas con el contenido de una o varias materias del curso. A continuación, a modo de ejemplo, se proponen una calculadora de polinomios y la implementación del tipo abstracto conjunto.

Especificación. Opción a) El programa debe comportarse como una *calculadora de polinomios* que permita realizar las siguientes operaciones: producto de un escalar por un polinomio, suma, resta, multiplicación y división entre polinomios, integración y derivación. Para esto, los estudiantes deberán definir los tipos de datos que consideren más adecuado y desarrollar mediante funciones cada una de las operaciones.

Opción b) Los estudiantes deben definir una estructura de datos adecuada para representar *conjuntos*, facilitando que se pueda cambiar fácilmente el tipo de datos de los elementos. Para los conjuntos deben implementarse las operaciones de unión, intersección, unión exclusiva y complementariedad.

Fases de elaboración y metodología. Primera fase, análisis matemático: esta fase comprende el estudio/repaso de los conceptos matemáticos implicados. Segunda fase, familiarización con programas similares. Tercera fase, especificación del programa, con la definición de las estructuras de datos y las operaciones asociadas. Última fase, juego de pruebas y verificación de resultados.

Resultados y formato de presentación. Documento en formato electrónico en donde se especifique: La descripción matemática del problema, la descripción de la metodología utilizada, la descripción de las estrucutas de datos y especificación de funciones, el código fuente en Lenguaje C, y estudio de ejemplos y verificación de resultados.

Estrategia de valoración y mejora. Se realizará una demostración pública de las tareas de cada uno de los equipos, y se analizarán las soluciones (ya sea en sesiones colectiva o en consultas), con contribución de las opiniones de los compañeros. Los equipos deberán incorporar las mejoras sugeridas al programa y actualizar la documentación, que se hará pública en la plataforma web de la asignatura.

4.3. Tercera tarea: diseño y realización de aplicaciones

En esta tercera etapa se pide a los estudiantes que propongan su propia aplicación, que debe ser acotada y estar relacionada con alguna de las otras asignaturas.

Especificación. Se trata de proponer y desarrollar una tarea relacionada con otras materias, verificando que tiene el nivel de dificultad aproximado descrito en los objetivos de la tercera etapa (apartado **??**), y desarrollarla.

Fases de elaboración y metodología. Inicialmente se realizará una sesión de *brainstorming* para eligir las tareas, sopesando su dificultad. Las tareas se llevarán a cabo considerando las fases y metodología aprendidas en la segunda etapa.

Resultados y formato de presentación. Se escribirá la documentación de la aplicación, incluyendo la descripción del problema (en los términos de la otra materia), el manual de usuario, la implementación y las pruebas.

Estrategia de valoración y mejora. Se realizará una demostración pública de las tareas de cada equipo, y se hará una crítica constructiva. Las mejoras sugeridas se incorporarán al programa y a la documentación, como en el caso anterior.

5. Metodología, progresión del aprendizaje y calificación

El desarrollo del curso será principalmente práctico utilizando extensivamente el ordenador personal de los propios estudiantes (los laboratorios de la escuela ofrecen puestos de trabajo suficientes para los que no dispongan de ordenador personal). No obstante, el estudiante también tiene que ser capaz de diseñar y codificar algoritmos sobre el papel.

El trabajo se realizará en equipo, pero cada estudiante deberá utilizar el material de la asignatura para el estudio autónomo (apartado **??**). La función principal del profesor será el seguimiento del desarrollo de las tareas, y ofrecer realimentación en el momento preciso. También se utilizará la revisión de ejercicios entre compañeros y se localizarán estudiantes de cursos superiores para que actúen como monitores a cambio de créditos de libre elección.

El aprendizaje de la programación en un entorno como Áncora facilita la participación de estudiantes con distintos estilos de aprendizaje y formas de aprender, y permite el desarrollo de distintos itinerarios de aprendizaje. Algunos abordarán directamente

las tareas y tratarán de resolver las dificultades sobre la marcha, otros preferirán ejercitar previamente, paso a paso, con el material de la asignatura, otros se estudiarán la especificación del lenguaje con un manual de referencia, mientras otros construirán sus programas partiendo de un ejemplo parecido y modificándolo. Pero para todos el objetivo explícito será resolver las tareas.

Cuando el aprendizaje de la programación se organiza por temas, puede darse la situación de que los estudiantes aprendan todo a medias. Por contraposición, la organización por etapas garantiza una progresión del aprendizaje en la que lo que se va aprendiendo, se aprende bien, y se consolida en la etapa siguiente. En un intento de conseguir que *todos* los estudiantes entren en la lógica de la programación, la primera tarea no se dará por terminada hasta que *todos* alcancen los objetivos correspondientes, ayudándose los unos a los otros si hace falta. De la misma forma, tampoco tiene sentido empezar la tercera etapa sin dominar los objetivos de la segunda. Dado que al final de la segunda etapa los conocimientos alcanzados incluyen ya todos los elementos del lenguaje, consideramos que las dos primeras etapas cubren los objetivos básicos del curso. La tercera etapa, entonces, es en cierto sentido complementaria.

La calificación en IO se divide en un 50 % de evaluación interna, más un 50 % de un examen final externo, común a todos los grupos de primero. Si la nota del examen final supera la nota de evaluación interna, la calificación final es el 100 % de la del examen. En lo que se refiere a la evaluación interna, nuestro criterio es otorgar el *apto* (6 puntos) a quien supere los objetivos de la primera y la segunda etapas, y la puntuación del 6 al 10 se pacta *a priori* en el momento de especificar la tercera tarea.

6. Recursos de apoyo al aprendizaje

Como apoyo al aprendizaje autónomo en IO hemos elegido el libro "C, guía de autoaprendizaje" [**?**], que incluye el uso de funciones desde el primer capítulo.

Como material de elaboración propia ofrecemos, para la primera etapa, una libreria de gráficos de tortuga para realizar ejercicios de iniciación, y una colección de ejercicios resueltos de dificultad progresiva. Para la segunda etapa, un par de ejemplos resueltos de estructuras de datos con sus operaciones asociadas, y para la tercera etapa, algunas aplicaciones completas y algunos exámenes finales resueltos de otros cursos.

Estos recursos estarán a disposición de los estudiantes en la plataforma *moodle* [**?**]. Ofrecemos, además, los recursos generales de la escuela (biblioteca, laboratorios), aunque contamos con que la mayoría de los estudiantes tienen acceso a un ordenador personal.

7. Conclusiones

En este artículo se ha descrito la organización en tres etapas de la asignatura Introducción a los ordenadores (IO) de la escuela de Telecomunicación de la UPC, en Barcelona, en el contexto de la experiencia de innovación docente Áncora.

Se han descrito los objetivos específicos de cada etapa y se han formulado tareas que los estudiantes deben resolver para alcanzar dichos objetivos. Las tareas propuestas se integran de forma transversal con las demás asignaturas del primer cuatrimestre de la titulación, en el contexto de Áncora.

En cuanto a la metodología, destaca el cambio de rol del profesor que pasa de impartir clases expositivas a realizar un seguimiento continuado del aprendizaje de los estudiantes.

En cuanto a la progresión y la evaluación internas, la propuesta se basa en el criterio de no pasar a la siguiente etapa hasta superar los objetivos de la anterior, de forma que las dos primeras etapas sean suficientes para alcanzar el apto.

No lo podemos asegurar, pero tenemos la confianza de que los estudiantes, después de conseguir el apto en la evaluación interna, conseguirán también buenos resultados en el examen final externo.

Referencias

[1] Navarro J.J., Valero-García M., Sánchez F. y Tubella J., *Formulación de los objetivos de una asignatura en tres niveles jerárquicos*, Actas de JENUI 2000, pp. 457-462.

[2] Bofill P., Otero B., Toribio E., Aroca JM., Breitman M, Garcias P. y Sancho JM., *ÁNCORA: Aprendizaje Organizado por Tareas*, actas de JENUI 2005, pendiente de publicación.

[3] H. Schildt, *C, guía de autoaprendizaje*, Osborne McGraw-Hill, 2001, ISBN 8448132041.

[4] Sitio web del proyecto Áncora, http://www.etsetb.upc.es/estudis/telecos/ancora/

[5] Sitio web de la Escuela Técnica Superior de Ingeniería de Telecomunicación, ETSETB, http://www.etsetb.upc.es

[6] Sitio web de la plataforma Moodle, http://moodle.org