Metodología basada en descomposición funcional y orientación a objetos en la introducción a la programación

Mercedes Gómez Albarrán
Dpto. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
28040 Madrid
e-mail: albarran@sip.ucm.es

Resumen

Este trabajo realiza una propuesta docente para un amplio cuerpo de materia de iniciación a la programación en el que tienen cabida la Programación Orientada a Objetos (POO) y una metodología de programación más tradicional como es la basada en descomposición funcional. Se muestra cómo podría quedar reflejada dicha propuesta en el plan de estudios de la Ingeniería en Informática de la Universidad Complutense de Madrid.

1. Introducción

A comienzos de los años noventa, en línea con las recomendaciones de *Computing Curricula 1991*, las asignaturas introductorias a la programación y la presentación de la POO ocupaban posiciones muy distintas en el currículo: las primeras presentaban una metodología de diseño procedimental usando un lenguaje de programación imperativo, con frecuencia Pascal, mientras que la POO aparecía en asignaturas de cursos avanzados.

En nuestros días vivimos lo que podríamos denominar un período de *(r)evolución* en lo que respecta a la enseñanza de la materia de introducción a la programación. Unos apuestan por mantener el enfoque "tradicional"; otros apuestan por incluir la orientación a objetos (OO) en el currículo introductorio. No existe un consenso en la comunidad docente en lo que respecta a la forma de abordar la introducción a la programación, incluso el propio *Computing Curricula 2001* [1] no se decanta por una estrategia concreta.

La tendencia a incluir la OO en el currículo introductorio es cada vez mayor. La cuestión es, ¿cómo se incluye? Podemos distinguir dos enfoques: el enfoque "objetos más tarde" (*objects-late*), en el que se presenta la OO tras la metodología basada en descomposición funcional, y el enfoque "objetos primero" (*objects-first*), que comienza presentando directamente los fundamentos de la POO[1].

La propuesta que aquí se plantea se enmarca dentro del enfoque "objetos más tarde" y se articula en torno a las siguientes ideas:

- Es innegable el papel fundamental que tiene la OO en el desarrollo de software actual y parece claro que será el modelo de programación dominante en un futuro próximo. De ahí que apostemos por no relegarlo a una enseñanza de "segundo nivel" –entendiendo por "segundo nivel" una presentación alejada del primer o segundo año de estudios–, sino por que forme parte de la formación básica en programación de los alumnos.
- La metodología basada en descomposición funcional ha sido y sigue siendo una metodología válida, que permite a los alumnos enfrentarse al desarrollo de algoritmos aislados y pequeñas aplicaciones. En consecuencia, tampoco consideramos que deba ser excluida. En este sentido, un enfoque "objetos primero" discrimina a la metodología basada en descomposición funcional, tal y como se indica en [15].

[1] Existen diversas tendencias dentro del enfoque "objetos primero" según se dé más importancia al uso de clases, a la creación de clases, a la construcción de jerarquías, etc. [3][7].

- La descomposición funcional y la descomposición conducida por los datos (en la que se apoya el diseño OO) no son metodologías disjuntas y mutuamente excluyentes. La Ingeniería del software nos aconseja que el diseño conducido por los datos (la abstracción de datos) domine el desarrollo de aplicaciones de cierta envergadura. Pero cuando hablamos de abstracción de datos no hablamos sólo de los valores sino también de operaciones asociadas. Las operaciones requieren algoritmos. La descomposición funcional es aplicada para diseñar esos algoritmos, y también puede serlo en la coordinación de las interacciones entre datos que representa el código conductor de la aplicación.

Con estas ideas subyacentes, en el bloque de materia de iniciación a la programación que se propone, se presentarían, en este orden:

- Descomposición funcional para el desarrollo de algoritmos concretos y aplicaciones a pequeña escala.
- Descomposición conducida por los datos, para el diseño de aplicaciones no triviales. En este punto la abstracción procedimiental no es eliminada sino subyugada a la abstracción de datos que debe ocurrir antes en el proceso de diseño.
- Orientación a objetos, una técnica basada en el diseño conducido por los datos que incluye mecanismos que facilitan la creación de software extensible y reutilizable.

A continuación, pasamos a describir los objetivos de una propuesta como la nuestra. Seguidamente se presentan los contenidos de la propuesta, reflejando cómo puede ser aplicada en un plan de estudios concreto que es el de la Ingeniería en Informática de la Universidad Complutense de Madrid. Así mismo, en relación con la propuesta, abordaremos la cuestión del/de los lenguaje(s) de programación a utilizar.

2. Objetivos de la propuesta

Consideramos que el objetivo general de un bloque de materia de iniciación a la programación, como integrante del grueso de asignaturas dedicadas a la enseñanza de la programación con las que los alumnos se enfrentan a lo largo de la carrera, es contribuir en la capacitación de los alumnos para desarrollar programas de calidad. Entendemos por programas de calidad aquellos que cuentan con las siguientes características: claridad, corrección, extensibilidad, reusabilidad y eficiencia.

A nivel más concreto, la propuesta que aquí se presenta pretende alcanzar los siguientes subobjetivos:

- Dejar claro que la programación es un acto de resolución de problemas. Saber programar no es aprender uno o más lenguajes de programación concretos sino conocer y saber utilizar métodos que permitan la construcción sistemática de algoritmos.
- Presentar los conceptos de algoritmo y programa.
- Reconocer la necesidad de manipular datos cuando programamos y exponer el concepto de tipo de datos.
- Resaltar el papel fundamental que juega la abstracción en todo proceso de resolución de problemas y, en particular, en el proceso de desarrollo de software. Se deben exponer procesos de abstracción aplicados a diferentes entidades, presentando a lo largo de la materia la tendencia a incluir mecanismos de abstracción cada vez de más alto nivel, tendencia que acompaña a la evolución de la Programación como disciplina.
- Aportar elementos de programación básicos y metodologías de programación para la resolución de problemas.

  Se propone la metodología de descomposición funcional para el desarrollo tanto de algoritmos aislados como de aplicaciones completas de pequeño tamaño. Al plantearse la necesidad de tipos de datos más complejos se presentan la noción de Tipo Abstracto de Datos (TAD) y herramientas para su definición.

  La programación con TADs se introduce en el marco de la programación modular como una metodología de diseño que permite organizar los programas en torno a los datos que manipulan y no en base a las operaciones a realizar. Este diseño conducido por los datos se plantea como la técnica preferente a usar en el diseño de aplicaciones de mayor envergadura. Como ya se ha señalado, la metodología basada en descomposición

funcional y la conducida por los datos no son técnicas mutuamente excluyentes y en la presentación de los contenidos a los alumnos se debe hacer hincapié en ello.

- Introducir a los alumnos en la POO, presentándola como un paradigma que se apoya en la abstracción de datos e incluye mecanismos que permiten construir software extensible y reutilizable.
- Examinar los conceptos fundamentales de la OO.
- Dar a conocer los pasos esenciales que conlleva proporcionar una solución orientada a objetos a un cierto problema.
- Aplicar los conceptos presentados para solucionar problemas usando una aproximación orientada a objetos.
- Adquirir conocimiento acerca de (al menos) un lenguaje de programación. Dicho lenguaje debe contar con los mecanismos necesarios para dar soporte a las metodologías presentadas. Otra característica preferible, aunque no primordial en un primer lenguaje de programación, es que sea de proyección profesional.

3. La organización de los contenidos de la propuesta

La tabla 1 muestra los temas que componen el contenido de nuestra propuesta. Con la organización propuesta se espera realizar una transición relativamente "suave" de unas cuestiones a otras.

El primer tema presenta nociones básicas sobre Informática y programación, tales como la diferencia entre el hardware y el software, el concepto de algoritmo y el concepto de programa.

En el segundo tema hacemos que los alumnos caigan en la cuenta de que los algoritmos trabajan sobre datos y que es necesario representarlos y manipularlos. Se presentan el concepto de tipo de datos, las nociones de variable y constante, la instrucción de asignación. Se trata también la comunicación vía consola con el usuario.

El objetivo del tercer tema es la presentación de las construcciones alternativas e iterativas. Esas instrucciones se deben presentar como un paso en el camino hacia la abstracción, comparándolas con lo que ocurre en los lenguajes máquina. Se

1. Introducción a las computadoras y a la programación
2. Instrucciones y tipos elementales
3. Estructuras de control
4. Abstracción funcional
5. Introducción a la recursividad
6. Abstracción de datos: una introducción a los TADs
7. Colecciones
8. El camino hacia la orientación a objetos
9. Más sobre clases y objetos
10. Herencia
11. Polimorfismo y vinculación dinámica
12. Introducción al diseño orientado a objetos: aplicación de conceptos

Tabla 1. Temario de la propuesta.

esquematizan algoritmos iterativos de uso frecuente como son los recorridos y las búsquedas.

En la tendencia a incluir mecanismos de abstracción cada vez de más alto nivel, se presenta en el tema 4 la abstracción funcional. Se aborda el diseño de algoritmos aplicando diseño descendente.

La recursión se presenta como una técnica de resolución de problemas en el tema 5.

Llegados a este punto de la materia, se plantea a los alumnos la necesidad, a medida que aumenta la complejidad de los problemas abordados, de que el programador cree sus propios tipos de datos. Se presentan así en el tema 6 las ideas de abstracción de datos, TAD y las clases como la herramienta para implementar TADs –al permitir aquellas la encapsulación de datos y operaciones, hacer privados la representación de los datos y la implementación de las operaciones y proteger de accesos incontrolados. En este mismo tema se hará hincapié en que la metodología de descomposición conducida por los datos es la predominante en el diseño de aplicaciones de envergadura y que la metodología basada en descomposición funcional vista sigue siendo aplicable al desarrollo de las operaciones de los

nuevos tipos creados. Estas consideraciones son muy importantes para que los alumnos no tengan la sensación de que lo que han visto hasta ese momento deja de ser útil.

Muchos problemas requieren manejar cierto número de entidades de un determinado tipo. Si el número es considerable no se puede recurrir al uso de variables "independientes". La solución viene de la mano del uso de colecciones de entidades homogéneas. El tema 7 se dedica a estas cuestiones.

Tras haber presentado los conceptos de abstracción y encapsulación de datos y las clases como una herramienta para implementar los TADs, en el tema 8 se presenta la POO como una programación con TADs junto una serie de mecanismos que permiten alcanzar mejoras en la calidad del software. En este tema se ofrece una visión preliminar de esos mecanismos. Básicamente, se trata de transmitir a los alumnos de manera intuitiva las "bondades" de la OO.

El tema 9 aborda las cuestiones de creación, inicialización y destrucción de objetos. Para que los alumnos puedan llegar a apreciar mejor las características incorporadas en los diferentes lenguajes que soportan la OO, deberían exponerse algunos temas relacionados tales como la gestión de asignación de memoria y la diferencia existente entre las copias superficiales y las copias profundas.

Aprender a "programar orientado a objetos" implica comprender uno de los pilares de la filosofía que hay detrás: la realización de una tarea se materializa en la interacción de objetos, cada uno de los cuales tiene un estado y proporciona una serie de servicios (comportamiento). Llegados a este punto, los alumnos cuentan ya con los conocimientos necesarios para definir clases, crear objetos y conseguir que éstos se comuniquen entre sí mediante pasos de mensajes. El siguiente paso es establecer relaciones entre clases, siendo la herencia y la composición las dos típicas utilizadas en la organización de las clases. El tema 10 se dedica a la herencia.

Desde el punto de vista más pragmático, la herencia se presentará a los alumnos como un mecanismo que permite reutilizar código. Efectivamente, permite compartir código: las características comunes a un conjunto de clases se extraen y definen en una clase más general a la que luego especializan. Las descripciones de las clases resultado de esa especialización se complementan con la descripción de la superclase. De esta forma, disminuyen el tiempo y el esfuerzo necesarios para desarrollar las nuevas clases. Y el hecho de que el código de las clases esté organizado en una jerarquía permite darle una estructura al conjunto de módulos –haciéndolo más comprensible– así como facilitar los cambios, pues una modificación realizada en una clase se propaga automáticamente a sus subclases.

Aún con todo, estas ventajas no son las principales. Lo verdaderamente relevante aquí, y hay que hacérselo ver a los alumnos, es que la herencia, complementada con la vinculación dinámica y el polimorfismo (aspectos introducidos brevemente en el tema 8 y que serán tratados en detalle en el siguiente), permite construir descripciones a distintos niveles de abstracción.

En el tema 11 se revisa el concepto de polimorfismo brevemente descrito en el tema 8. Conviene relacionar, siempre que sea posible, lo que se vaya viendo con aspectos y ejemplos que ya hayan surgido. En este caso se hace ver a los alumnos que ya hemos encontrado entidades polimórficas (por ejemplo, los nombres de los métodos polimórficos debido a sobrecarga y redefinición). Aquí nos centramos, sin embargo, en las variables polimórficas.

Es interesante que los alumnos distingan entre el tipo estático y el dinámico de una variable y que se analice cómo influye la forma de reservar espacio de memoria en la existencia de variables polimórficas.

Asimismo, hay que resaltar el polimorfismo y las clases abstractas como aspectos clave en la reutilización.

Para finalizar el tema, se analizan las implicaciones que conlleva la existencia de variables polimórficas, presentando la distinción entre vinculación estática y dinámica.

El último tema del grueso de materia propuesto lo consideramos indispensable. En él se lleva a cabo una introducción al diseño OO: se presentan algunas consideraciones generales que conducen a buenos diseños –la alta cohesión, el bajo acoplamiento, la trazabilidad, etc.– y un esquema simple con las tareas básicas de todo diseño OO –detección de clases, determinación de relaciones entre clases, diseño de la coordinación de los objetos. Los conceptos expuestos a lo largo

del bloque de materia se aplican al desarrollo de soluciones orientadas a objetos a ciertos proyectos de programación. Consideramos que cualquier asignatura donde se introduzca la OO necesita de un tema como éste, en el que los alumnos pongan a prueba su comprensión del significado y la potencia de la OO.

Aparte de las correspondientes clases en las que abordar todos estos temas, sería altamente recomendable que los alumnos pudiesen hacer prácticas donde poner a prueba todos los conocimientos adquiridos.

3.1. La propuesta reflejada en el plan de estudios de la Ingeniería en Informática de la Universidad Complutense de Madrid

Esta sección refleja cómo una propuesta como la descrita se puede llevar al plan de estudios de la Ingeniería en Informática de nuestra universidad.

Las asignaturas directamente relacionadas con la programación en los dos primeros años de dicha titulación son:

- Primer curso: *Introducción a la Programación* y *Laboratorio de Programación I* son las asignaturas de toma de contacto con la programación. La primera es anual (9 créditos, 6 teóricos y 3 prácticos); la segunda (4,5 créditos, todos prácticos) se imparte en el segundo cuatrimestre como laboratorio de soporte a la primera.
- Segundo curso: *Estructura de Datos y de la Información* es una asignatura anual (15 créditos, 10 teóricos y 5 prácticos); *Programación Orientada a Objetos* (4,5 créditos, 3 teóricos y 1,5 prácticos), que se imparte en el primer cuatrimestre; *Laboratorio de Programación II* es anual (9 créditos, todos prácticos).

Las asignaturas que escogeríamos para impartir los contenidos presentados son *Introducción a la Programación* y *Programación Orientada a Objetos*. Grosso modo, el reparto de contenidos entre dichas asignaturas queda como sigue:

- Presentación de la metodología basada en descomposición funcional en el primer cuatrimestre de *Introducción a la Programación* como colofón a la presentación de la abstracción funcional. El segundo cuatrimestre de *Introducción a la Programación* se dedica a la metodología basada en descomposición conducida por los datos. Podríamos pues decir que en esta asignatura se seguiría un enfoque "basado en objetos" (*object-based*): programación con objetos (entidades o datos) como bloques principales de construcción, pero sin herencia. En definitiva, los temas del 1 al 7 quedarían recogidos en esta asignatura.
- Tomando como punto de partida los conocimientos adquiridos en la anterior asignatura, la asignatura *Programación Orientada a Objetos* se dedicaría a profundizar en la OO a lo largo de los temas 8 al 12. La descomposición funcional se seguirá presentando como técnica candidata a utilizar en el diseño del comportamiento de los objetos de una clase.

La tabla 2 recoge la estimación de la asignación de tiempo (medida en horas) para cada uno de los temas del programa propuesto.

Por lo que respecta a las asignaturas *Laboratorio de Programación I* y *Laboratorio de Programación II* pueden emplearse para la realización de prácticas en las que los alumnos utilicen los conocimientos adquiridos en *Introducción a la Programación* y *Programación Orientada a Objetos*.

4. La elección del lenguaje de programación

Dentro del planteamiento general de la propuesta que aquí se realiza queda una cuestión más que abordar: el/los lenguajes de programación a utilizar.

Hay que indicar que consideramos conveniente presentar los diferentes conceptos de la forma más desligada posible de un lenguaje de programación específico. Pero, a la vez, la ejemplificación con un lenguaje de programación concreto resulta interesante de cara a que los alumnos puedan más fácilmente poner en práctica sus conocimientos, ya sea en su propia casa o en el laboratorio (bien sea en horas regladas de las asignaturas de laboratorio que sirvan de complemento o en horas libres).

A la hora de elegir lenguaje, debemos tener en cuenta las necesidades de cada parte del grueso de materia. Para los siete primeros temas necesitamos

	Teoría	Práctica	Total
Introducción a la Programación			
1. Introducción a las computadoras y a la programación	3	–	3
2. Instrucciones y tipos elementales	5	3	8
3. Estructuras de control	10	6	16
4. Abstracción funcional	12	6	18
5. Introducción a la recursividad	4	3	7
6. Abstracción de datos: una introducción a los TADs	12	7	19
7. Colecciones	14	5	19
Programación Orientada a Objetos			
8. El camino hacia la orientación a objetos	2	–	2
9. Más sobre clases y objetos	6	2	8
10. Herencia	8	4	12
11. Polimorfismo y vinculación dinámica	5	2	7
12. Introducción al diseño orientado a objetos: aplicación de conceptos	9	7	16

Tabla 2. Asignación temporal (en horas) para los temas.

un lenguaje que soporte la abstracción funcional y la de datos. Para los cinco temas restantes necesitamos un lenguaje que soporte la orientación a objetos.

Podríamos optar por utilizar dos o más lenguajes. Sin embargo, hemos decidido utilizar un único lenguaje con el fin de evitar cambios de sintaxis dentro del bloque de materia. Actualmente en el primer curso de programación en la Facultad de Informática de la Universidad Complutense de Madrid se emplea Pascal. En segundo curso, en los últimos años se ha intentado unificar el lenguaje tanto para la asignatura *Programación Orientada a Objetos* como para *Laboratorio de Programación II*, siendo C++ el lenguaje utilizado. Los profesores de segundo curso somos conscientes de los problemas iniciales que tienen no pocos alumnos de segundo curso al cambiar de sintaxis.

En esta propuesta hemos elegido el lenguaje multiparadigma C++. Somos conscientes de que C++ es un lenguaje muy amplio y permisivo. Ante esta situación, estamos de cuerdo con lo que se indica en [1]:

> "*... many of the languages used for object-oriented programming in industry –particularly C++, but to a certain extent Java as well– involve significantly more detail complexity than classical languages. Unless instructors take special care to introduce the material in a way that limits this complexity, such details can easily overwhelm introductory students.*"

Conscientes de su amplitud, hay que indicar que no pretendemos en absoluto presentar el lenguaje en su totalidad. Para los cinco primeros temas utilizaremos un subconjunto de la parte procedimental de C++. Esta propuesta va en la línea de [9]. Hay que decir que, a este respecto, nos sentimos totalmente respaldados por los libros seleccionados como bibliografía básica. Estos libros son [5] y [16] para los siete primeros temas y [6] para los cinco restantes.

Por lo que respecta a la permisividad, que en ocasiones conduce a la obtención de código excesivamente críptico, estamos de acuerdo con lo que se indica en [5]: la solución viene de la mano de un estilo de programación directo, disciplinado

	Curso 1995-96	Curso 1996-97	Curso 1997-98	Curso 1998-99	Curso 1999-00
Pascal	36%	23%	6%	2%	5%
Ada	12%	18%	19%	7%	6%
C	17%	14%	11%	20%	19%
Scheme	2%	4%	4%	1%	---
C++	32%	39%	47%	50%	54%
Java	---	---	9%	22%	22%

Tabla 3. Primer lenguaje de programación usado en centros estadounidenses [10] [11] [12] [12] [14].

y libre de características intrincadas del lenguaje, de forma que los alumnos aprendan a utilizar C++ para producir código claro y legible.

El lenguaje de programación Java se va abriendo camino como primer lenguaje de programación (especialmente en centros estadounidenses está alcanzando niveles de uso nada despreciables). No hay más que echar un vistazo a los estudios que desde el curso 1995-96 son realizados acerca de la situación en departamentos que ofertan programas oficiales de Informática en Estados Unidos [10] [11] [12] [13] [14]–programas acreditados por CSAC/CSAB (*Computer Science Accreditation Commission of the Computing Sciences Accreditation Board*). La tabla 3 muestra los resultados que, relativos al primer lenguaje de programación usado, reflejan los citados estudios. Los resultados aparecen en forma de porcentaje de departamentos encuestados que utilizan cada uno de los lenguajes indicados.

Aunque hoy por hoy C++ es el lenguaje de programación más utilizado en universidades y *colleges* de Estados Unidos, vemos que Java que ha experimentado en tres años un crecimiento considerable[2].

Java es, indudablemente, un lenguaje muy interesante. Sin embargo no lo consideramos apropiado en nuestro marco por ciertos motivos. Por un lado, Java no es especialmente adecuado para presentar una metodología basada únicamente en descomposición funcional. Podemos usar Java siguiendo un estilo básicamente procedimental (usando métodos estáticos en una única clase). Pero los programas resultantes en Java serían, a nuestro juicio, un poco artificiosos. Además Java limita las posibilidades del flujo de datos a través de la interfaz de un bloque funcional cuando se trabaja exclusivamente con tipos primitivos (y no con objetos), que sería la situación en la que nos encontraríamos en los cinco primeros temas. Por otro lado, cuestiones nada triviales como son las excepciones aparecen desde el primer momento relacionadas con la entrada de datos. Finalmente, están las revisiones que probablemente aún sufran el lenguaje y sus bibliotecas, fruto de un lenguaje que aún no ha madurado lo suficiente.

El hecho de que Java sea un lenguaje reciente también se acusa en la bibliografía. No hemos conseguido encontrar un número significativo de libros de Java especialmente dedicados a un primer curso de programación. Por último, señalar un aspecto que nos ha llamado la atención en la bibliografía de Java. Entre los textos que hemos analizado existe una gran diferencia en la presentación de un tema que inevitablemente aparece desde las primeras clases y de forma recurrente: la entrada de datos. Entre los libros manejados nos hemos encontrado con los que utilizan las clases proporcionadas por Java [2], otros que "recubren" dichas clases con clases propias [8] y otros que utilizan interfaces gráficas desde el primer momento [4]. Esto hace que la resolución de un mismo ejemplo según el texto considerado tenga aspectos muy diferentes, algo

[2] Una visión más global del panorama actual en relación con el primer lenguaje de programación se puede obtener a partir de los informes Reid (accesibles desde la página personal del encargado de mantenimiento de los mismos, Frances L. Van Scoy, http://www.csee.wvu.edu/~vanscoy). Dichos informes recogen la situación de centros de todo el mundo.

que consideramos puede crear confusión en alumnos que (al menos teóricamente) se enfrentan por primera vez con la programación.

5. Conclusiones

Hemos presentado una propuesta de contenidos para un bloque de materia de iniciación a la programación que aúna la metodología procedimental junto con la orientación a objetos. La propuesta se ejemplifica llevándola a un plan de estudios concreto, el de la Ingeniería en Informática de la Universidad Complutense de Madrid.

Entre los diferentes aspectos tenidos en cuenta cabría destacar, por un lado, la conveniencia de desligar (en la medida de las posibilidades) la presentación de los conceptos de un lenguaje de programación concreto, y, por otro, la conveniencia de que se realice una transición "suave" entre contenidos, intentando que los alumnos vean las "bondades" de cada una de las metodologías y cómo se construyen unas sobre otras.

Agradecimientos: Este trabajo ha sido financiado parcialmente por el proyecto TIC 2000-0737-C03-01.

Referencias

[1] ACM/IEEE-CS Joint Task Force, *Computing Curricula 2001*, final draft (diciembre 2001).

[2] Arnow, D., Weiss, G., *Introducción a la programación con Java*, Addison-Wesley, 2001.

[3] Culwin, F., "Objects Imperatives!", *Procs. of the SIGCSE Technical Symposium on Computer Science Education*, 1999.

[4] Dale, N., Weems, C., Headington, M., *Introduction to Java and Software Design*, Jones and Bartlett, 2001.

[5] Dale, N., Weems, C., Headington, M., *Programming and Problem Solving with C++, 2nd Edition*, Jones and Bartlett, 2000.

[6] Lafore, R., *Object-Oriented Programming in C++*, SAMS Publishing, 1999.

[7] Lewis, J., "Myths about Object-Orientation and its Pedagogy", *Procs. of the SIGCSE Technical Symposium on Computer Science Education*, 2000.

[8] Liang, Y.D., *Introduction to Java*, Prentice Hall, 2001.

[9] Llopis Pascual, F., Pérez López, E., "C++-OO como lenguaje introductorio a la programación", *Actas de las VII Jornadas de Enseñanza Universitaria de la Informática*, 2001.

[10] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1999 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report CoC/CS TR# 2000-9-1, Department of Computer Science, College of Charleston, octubre 2000.

[11] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1995 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-96-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1996.

[12] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1996 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-97-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1997.

[13] McCauley, R.A., Manaris, B.Z., Comprehensive Report on the 1997 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs, Technical Report TR-98-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1998.

[14] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1998 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-99-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1999.

[15] Marion, W., "CS1: What Should We Be Teaching?", *SIGCSE Bulletin*, 31, (4), 1999.

[16] Savitch, W., *Resolución de problemas con C++. El objetivo de la programación*, Prentice Hall, 2000.