Propuesta de metodología para un curso universitario introductorio a la programación

Fernando Llopis Pascual

Departamento de Lenguajes y Sistemas Informáticos

Universidad de Alicante

e-mail: llopis@dlsi.ua.es

RESUMEN

Es destacable la importancia que tienen, en la formación del alumno que cursa estudios en informática, las asignaturas de primer ciclo más relacionadas con los temas de programación. Aunque la difusión de la informática ha aumentado en los últimos años, en la mayoría de los casos, estas asignaturas son la primera toma de contacto que el alumno tiene con la resolución de problemas a través de la realización de programas. El objetivo del presente artículo es presentar las experiencias adquiridas por la aplicación de un nuevo programa a la asignatura de Fundamentos de Programación en las titulaciones en informática que oferta la Universidad de Alicante.

1. INTRODUCCIÓN

La enseñanza de la materia de las asignaturas introductorias a la programación ha sufrido una evolución a lo largo del tiempo determinada por el cambio que se ha producido en la tecnología informática. Es por ello, que el enfoque docente con que se debe abordar la enseñanza de dicha materia en un contexto universitario debe ser coherente con esta evolución. Existe un principio básico en este enfoque: "el objetivo de las asignaturas de primer ciclo más relacionadas con temas de programación es conseguir que el alumno alcance un cierto dominio en el desarrollo de programas, sea cual sea la magnitud de los mismos".

Con el fin de alcanzar este objetivo, es necesario que el alumno sea capaz de:

1) Escribir programas en un tiempo razonable, que funcionen correctamente, estén bien documentados y sean legibles.

2) Abordar eficazmente la división de los programas de gran magnitud en módulos y especificar adecuadamente los mismos.

3) Razonar sobre la corrección y eficiencia de las soluciones desarrolladas y compararlas con distintas soluciones alternativas.

4) Identificar problemas desconocidos con familias de problemas conocidos y ser capaces de aplicar la solución algorítmica correspondiente.

Además el alumno debe darse cuenta del hecho de que programar no es sólo escribir programas que funcionen, sino conseguir que esos programas satisfagan otras características como puedan ser su claridad, modificabilidad, eficiencia, etc. Por ello

se tratará que el alumno se dé cuenta de lo erróneo de la frase:

" ..... pero si el programa funciona y es suficiente "

Además considero importante que la enseñanza universitaria de fundamentos de programación debe formar al alumno de tal forma que le prepare para el estudio crítico de cualquier lenguaje de programación así como para la asimilación de los cambios futuros. Este planteamiento no implica en ningún caso que deba sacrificarse el aprendizaje y uso de lenguajes de programación concretos, sino que por el contrario este uso debe completar la formación teórica del alumno.

2. EVOLUCION DE LA ASIGNATURA

Tradicionalmente, la enseñanza de la asignatura *Fundamentos de la Programación* en la Universidad de Alicante se ha enfocado desde una perspectiva que podríamos calificar como "orientada a un lenguaje de programación". Partiendo de una presentación más o menos exhaustiva de los fundamentos de la informática y de la presentación de las distintas notaciones para la representación de algoritmos, se pasaba a explicar un lenguaje de programación, normalmente Pascal o C.

Este enfoque tiene una serie de problemas:

- Se centra excesivamente en las características de un lenguaje de programación en concreto, haciendo que el alumno pensara más en la especificación de la solución en un lenguaje, mas que en una solución genérica, lo que en algunos casos podría dificultar el aprendizaje de otras herramientas o lenguajes de programación.

- El diseño de lenguajes de programación es un campo que evoluciona continuamente. Los lenguajes nacen, envejecen y mueren [Pratt98], y por ello se debe evitar una excesiva dependencia de esta asignatura con respecto a algún lenguaje de programación en concreto, esta debe plantearse de forma que un cambio de lenguaje de programación utilizado no afecte a los fundamentos de la propuesta para la asignatura.

Evidentemente, tampoco se puede obviar el aprendizaje de lenguajes de programación, porque esto facilita la comprensión de muchos conceptos y además se ha de conseguir que los alumnos adquieran cierto nivel de programación para estar preparados para las asignaturas del siguiente curso.

Así el planteamiento que se propone se basa en la introducción de una base teórica y genérica, para después pasar a estudiar lenguajes de programación en concreto. El primer bloque de estudio debe conseguir que el alumno aprenda a programar en general y por ello se estudiarán diferentes herramientas y técnicas independientes de los lenguajes de programación. El segundo bloque se centrará en la aplicación de los conceptos vistos en el primer bloque utilizando lenguajes de programación. Dentro de este segundo bloque se ha de destacar tres aspectos: el primero es el estudio de dos lenguajes de programación y no uno, el segundo, el paradigma o paradigmas de programación a estudiar y el tercer aspecto, la elección de los lenguajes de programación a utilizar.

3. ESTUDIO DE DOS LENGUAJES DE PROGRAMACIÓN

A lo largo de la carrera los alumnos deberán estudiar y trabajar con varios lenguajes de programación. Es importante que cuando el alumno se inicie en el estudio de los mismos, pueda empezar a diferenciar qué conceptos teóricos son dependientes de un lenguaje de programación y cuáles no lo son. También es positivo que el alumno no se centre únicamente en un lenguaje de programación, ya que el conocimiento de más de un lenguaje facilita el aprendizaje de uno nuevo y además le permite comparar entre diferentes lenguajes y poder valorar cuál es el más efectivo para solucionar un problema en concreto. Además el estudio de dos

lenguajes de programación puede permitir que los hábitos adquiridos en el primero se trasladen al segundo tal como se indica en [Walker96].

4. SELECCIÓN DE UN PARADIGMA DE PROGRAMACIÓN

De entre los paradigmas de programación disponibles cabe comentar cuál podría ser el más adecuado para la iniciación a la programación. De los tres paradigmas de programación, imperativo, declarativo y orientado a objetos, realmente dos ya que el tercero se puede aplicar tanto a lenguajes imperativos como declarativos, he seleccionado el imperativo por varios motivos. Principalmente debido a que considero más sencillo que el alumno comprenda los algoritmos como una secuencia ordenada de pasos que permiten realizar una serie de tareas, porque es algo más natural a su conocimiento. Además los paradigmas declarativos y orientados a objetos son estudiados en otras asignaturas.

No obstante, cabe indicar que han aparecido propuestas en las que se indica que es más recomendable introducirse a la programación a través del modelo orientado a objetos [Arnold98]. Yo por mi parte, he contrastado la dificultad que tienen los alumnos de entender algunos de los conceptos de dicho modelo en la asignatura Programación Orientada a Objetos, pero quizá esto pueda ser debido a que el planteamiento es diferente, ya que en este caso se trata de estudiar la programación orientada a objetos cuando el alumno se ha acostumbrado a trabajar de forma procedural.

5. SELECCIÓN DE UN LENGUAJE DE PROGRAMACIÓN

Quizá una de las cuestiones más debatidas y que más controversias provoque en la enseñanza de la programación, sea la elección del primer o primeros lenguajes de programación. Ello se puede comprobar al observar los diversos lenguajes de programación que se enseñan o incluso a que su estudio no se realice en clases de teoría sino únicamente en las clases prácticas.

En base a los objetivos propuestos los lenguajes de programación ideales deberían cumplir las siguientes características:

- Ser un lenguaje de propósito general.
- Debe facilitar los hábitos de programación estructurada y programación modular.
- Debe posibilitar al usuario la definición de tipos de datos adicionales.
- Debe existir un estándar de dicho lenguaje.

Además sería recomendable que estuviera difundido, que existieran versiones de dicho lenguaje para los entornos más utilizados por los alumnos (Linux, MS-DOS, Windows95) y que sus requisitos mínimos de ejecución no fuesen especialmente elevados. También es positivo que dispusiesen de suficiente documentación y bibliografía.

La evolución y cambios que han sufrido los lenguajes de programación en la enseñanza ha sido enorme. De hecho si miramos hacia atrás, no es extraño ver que lenguajes como el FORTRAN, COBOL y BASIC eran utilizados en asignaturas de programación en muchas universidades. Hoy en día, quizá solo el FORTRAN se sigue utilizando en la Universidad en ámbitos muy concretos.

Con respecto a los lenguajes introductorios, cabe indicar que el lenguaje Pascal ha sido el más utilizado, no obstante su uso hoy en día está empezando a decaer y su sustituto natural, el Modula-2 (o Modula-3) no ha conseguido ni mucho menos ocupar su lugar.

Pascal se diseñó para ser una herramienta para la enseñanza de conceptos de programación, no obstante herramientas basadas en lenguaje Pascal (más bien en Object Pascal la versión orientada a objetos de Pascal) tienen actualmente una gran aceptación en la industria (Borland Delphi). Del

lenguaje Pascal cabe decir que es un lenguaje sencillo de aprender, y facilita la adquisición de hábitos de programación estructurada y la comprensión de otros lenguajes de programación no tan sencillos.

Desde mi punto de vista, quizá los mayores defectos de Pascal son:

- No disponer del concepto de módulos compilados de forma independiente, problema ya resuelto en Modula-2.

- El lugar donde se deben definir las variables del programa principal, tan alejadas de su código y que pueden ser utilizadas como variables globales por las funciones y procedimientos que se declaran a continuación

- Pascal standard en cierta forma ha sido absorbido por el estándar de hecho, el Turbo-Pascal, de hecho hay más libros disponibles de Turbo-Pascal que de Pascal y algunos compiladores de Pascal han asumido algunas de las especificaciones del Turbo-Pascal.

- Otro de los problemas que tiene Pascal, es que difícilmente puede seguir siendo utilizada en otras asignaturas de cursos superiores como Tipos Abstractos de Datos o Programación orientada a Objetos.

En algunas Universidades se ha optado por el lenguaje C. De C cabe decir muchas cosas, sobre todo que es un lenguaje que no deja indiferente. Sus detractores dicen de él que es un lenguaje difícil de entender y que dificulta utilizar la programación estructurada. Sobre el primer aspecto cabe indicar que estoy de acuerdo parcialmente con él, creo que es tan fácil o difícil de entender como el Pascal, exceptuando un aspecto que ciertamente complica mucho su comprensión y es la gestión de los parámetros por valor y por referencia y la utilización en algunos comandos de la variable o de la dirección de la variable. Es este aspecto el que pienso que crea cierta frustración en los alumnos, y se desmoralizan al no saber claramente cuando deben colocar los símbolos & y * delante de cada variable. Por otro lado cabe destacar que C tiene la ventaja de disponer una continuación o mejora del mismo, C++, C orientado a objetos, que puede ser utilizada en cursos posteriores.

Cuando hablamos de futuro, cabe hablar de dos lenguajes de programación, Ada y Java. Ada es un buen lenguaje de programación, que cubre muchos de los requisitos solicitados, pero tiene una complejidad que lo aleja de lo recomendable en un curso de iniciación y además está poco difundido y se dispone poca bibliografía sobre él. Su viabilidad dependerá de la aceptación del nuevo estándar Ada95. Java es un lenguaje de tercera generación con características de programación orientada a objetos, multiplataforma, y especialmente preparado para el desarrollo de aplicaciones en Internet. Java está basado en gran medida en el C++ pero con una gestión de memoria menos propensa a errores de programación. No obstante Java tiene una eficiencia considerablemente menor que el C++, debido a ser un lenguaje interpretado o seudointerpretado. Su popularidad ha experimentado un gran crecimiento en los últimos años, de hecho hay propuestas que lo tratan como un buen lenguaje introductorio a la programación [Arnold98].

6. TEMARIO PROPUESTO

El temario propuesto se basa en dividir la asignatura en dos bloques temáticos, el primero dedicado al estudio de técnicas y herramientas independientes de los lenguajes de programación y el segundo donde se aplicarán dichas herramientas y técnicas al estudio de dos lenguajes de programación, el lenguaje Pascal como lenguaje introductorio a la programación y al lenguaje C, como lenguaje de continuación. En este segundo bloque se profundizará más en el C, ya que es el lenguaje que los alumnos utilizarán en asignaturas de cursos siguientes.

Bloque I: Análisis y diseño de algoritmos
Tema 1 Introducción. Ordenadores y programas.
Tema 2 Tipos de datos elementales.
Tema 3 Lenguaje algorítmico. Concepto de programa.
Tema 4 Programación modular.
Tema 5 Estructuras de datos.
Tema 6 Lenguajes de programación.

Bloque II: Lenguajes de programación
Tema 7 Programación elemental en Pascal.
Tema 8 Tipos de datos estructurados en Pascal.
Tema 9 Programación elemental en C.
Tema 10 Tipos de datos estructurados en C.
Tema 11 Ficheros en C.
Tema 12 Estructuras de datos dinámicas en C.

7.CONSIDERACIONES METODOLÓGICAS

El modelo docente tradicional seguido en la Universidad Española ha estado basado principalmente en la transmisión y recepción de conocimientos ya elaborados. Reflejo de este modelo es el método docente clásico, seguido en la enseñanza de las ciencias, que se resume en el siguiente ciclo:

- Presentación de los conceptos teóricos.
- Resolución de problemas por parte del profesor.
- Propuesta de nuevos problemas.

En el caso de las asignaturas estudiadas se suelen ir presentando las diferentes estructuras que se pueden utilizar para elaborar algoritmos y posteriormente se plantean y resuelven problemas sobre el tema tratado.

Suele ser, entonces, habitual presentar la estructura que soluciona el problema antes que el problema que se pretende solucionar. Yo considero que es mejor invertir los términos, el alumno suele entender mejor las estructuras cuando conoce de antemano el problema que resuelven. más en línea citada en [Gil94] y [Abboud94].

Un ejemplo donde se puede ver mi planteamiento es la forma que yo suelo seguir para el estudio de las estructuras de repetición en los algoritmos:

1) Una vez los alumnos ya conocen distintas estructuras de entrada y salida y las de selección se plantea un ejercicio que es elaborar un algoritmo que solicite dos números y escriba el mayor de ellos. Un gran porcentaje de alumnos suelen resolverlo.

2) A continuación se plantea una ligera variación, que consiste en calcular el mayor de tres números. Un mayor porcentaje de alumnos resuelve este ejercicio, ya que sólo supone una ligera variación del anterior.

3) Se vuelve a plantear otra ligera variación, ahora se debe calcular el mayor de cinco números. Este ejercicio suele provocar cierto desconcierto, ya que en muchos casos los alumnos intentan reflejar todas las combinaciones posibles entre los cinco números, antes de darse cuenta que esto se puede hacer comparando el nuevo número con el mayor de los dos anteriores.

4) Se solicita que el algoritmo ahora calcule el mayor de 15 números. Los alumnos indican que se debería modificar la solución anterior pero previendo la entrada de 15 números, no obstante, a muchos ya no les parece muy adecuado que el

algoritmo se haga más grande y complicado dependiendo del número de valores a considerar.

5) Ahora se indica que el algoritmo en primer lugar debe solicitar un valor y tras ello debe calcular el mayor de tantos números como indique dicho valor. Ahora esta ligera variación del algoritmo hace que sea casi imposible de resolver con los planteamientos conocidos hasta el momento.

6) Ahora se plantea que sería necesario poder disponer para poder resolver este último algoritmo y facilitar la elaboración del anterior. Tras los comentarios más o menos acertados de los alumnos paso a explicar la primera de las estructuras de repetición y como con ellas se podría elaborar dichos algoritmos.

Suele ser habitual que en las clases se planteen problemas y a continuación se solucionen por el profesor directamente o bien tras haber dado cierto tiempo al alumno a realizarlo. Este método tiene ventajas como las que permite realizar un gran número de problemas en clase, pero como su mayor inconveniente, cabe indicar que facilita que los alumnos adopten frecuentemente actitudes pasivas frente al aprendizaje y hábitos memorísticos y de reproducción mecánica de los modos de razonamiento presentados. Los alumnos se limitan a copiar la solución que da el profesor al problema, ya que consideran esta solución mucho mejor que la que podrían obtener ellos, sin plantearse posibles alternativas. Esto puede provocar un mal aprendizaje dadas las características de los problemas planteados, ya que los propios alumnos reconocen que entienden claramente la solución que da el profesor al problema, pero cuando ellos intentan resolverlo, en bastantes casos el día del examen, se ven totalmente incapaces de hacerlo.

También considero muy importante plantear ejercicios o pequeñas modificaciones de los ya resueltos, que los alumnos deberán resolver fuera de horas de clase, además haciendo hincapié en las ventajas de este tipo de problemas. Los problemas de programación no suelen ser como por ejemplo los de resolver a que velocidad cae cierto producto por un plano inclinado, ya que es imposible poder darse cuenta si la solución planteada es la correcta o no. Los algoritmos o programas que resuelven cierto problema pueden ser probados y el alumno puede darse cuenta de muchos de los errores que tiene su propuesta, hasta llegar a una solución definitiva.

No obstante esto tiene el inconveniente que si bien los alumnos desean realizar muchos ejercicios, pero aunque en clase sí que los intentan realizar, nunca (o en muy pocas ocasiones) los realizan en casa. Esto dificulta notablemente la realización de algunos ejercicios de mayor complejidad y que los alumnos lleven los conceptos teóricos estudiados al día. Este problema se puede solucionar en las clases de prácticas de laboratorio.

Las prácticas de laboratorio son un medio excelente para que el alumno potencie su iniciativa y capacidad crítica. Además son imprescindibles para conseguir un buen aprendizaje en materias científicas y tecnológicas, permitiendo proyectar los conocimientos acumulados sobre problemas reales. Por todos estos motivos es necesario prestar a las prácticas una gran atención y dedicación, para conseguir hacerlas útiles y sugestivas.

Además las clases prácticas permiten al profesor estar en mayor contacto con los alumnos debido al número reducido de alumnos por cada grupo, permitiéndole realizar un mejor seguimiento de cada alumno.

El enfoque que considero adecuado utilizar es el siguiente :

1. Las prácticas se plantean como una serie de problemas de complejidad *creciente.*

Las primeras prácticas de cada una de las partes serán de corta duración con el objeto de que el alumno vaya asimilando los conceptos y pueda ir corrigiendo los errores que va cometiendo. A

medida que transcurra el curso el alumno deberá resolver una serie de problemas de mayor complejidad que requieran varias sesiones.

2. - Se evalúan *continuamente las prácticas*

Es muy importante realizar un seguimiento continuado a las tareas que realiza el alumno para evitar que :

- El alumno inicie sus prácticas con planteamientos incorrectos, lo que hace que los arrastre durante toda la práctica.

- El alumno no le dedique mucho tiempo a las prácticas en las primeras sesiones y se dedique a otras tareas de mayor urgencia, con lo que se dispone de poco tiempo para la realización de las mismas.

Por otro lado se consigue indicar al alumno qué errores está cometiendo y cómo debe solucionarlos.

3. - Las prácticas se corregirán en presencia del alumno.

Es importante que siempre que sea posible, las prácticas se corrijan en presencia del alumno por varios motivos:

- Facilita notablemente la interrelación entre profesor y alumno, incrementando la confianza de éste para exponer sus dudas al profesor.

- Permite al profesor indicar cuáles han sido los fallos cometidos o qué elementos podrían ser mejorados.

- Permite conocer la aportación de cada alumno en las prácticas entregadas por el grupo.

4. - Es importante destacar la importancia de las prácticas en la asignatura.

- Es decir, ya no sólo debido a que el alumno que no realice las prácticas no aprobará, sino que el que no realice las prácticas difícilmente podrá adquirir los conocimientos que le permitan superar un examen escrito.

5. - Las prácticas deberán estar sincronizadas con las clases de teoría.

Así se consigue que el alumno conozca los aspectos teóricos previamente al planteamiento del trabajo práctico, pero de tal forma que no haya mucha diferencia en el tiempo en el que se impartieron los aspectos teóricos y cuando se realizan las clases practicas, ya que éstas pueden ayudar mucho a la comprensión de dichos aspectos teóricos.

8 CONCLUSIONES

Las conclusiones que indico a continuación se han obtenido a partir de la observación de los trabajos realizados por los alumnos en prácticas y exámenes, así como a partir de la encuesta realizada a los alumnos, donde ellos debían especificar sus opiniones positivas y negativas sobre la asignatura, sus contenidos y la forma en la que se ha impartido.

- Se ha conseguido motivar al alumno para que participe en clase y que por lo menos intenten resolver los ejercicios planteados y se atrevan a comentar sus soluciones.

- El aprendizaje de C, de forma posterior al seudocódigo y a la introducción del lenguaje Pascal, ha permitido que los alumnos escriban los programas en C notablemente influenciados por la estructura del Pascal. No obstante al dedicar menor tiempo al estudio del lenguaje C que en años anteriores, no se consigue profundizar en la misma medida en muchos de los conceptos del lenguaje, sobre todo en la gestión de memoria dinámica.

- La realización de muchas prácticas de corta duración y sincronizadas con las clases teóricas, ha permitido a los alumnos que practicaran los conceptos estudiados.

- A la mayor parte de los alumnos les ha gustado el enfoque de la asignatura en cuanto a la metodología, no obstante un porcentaje de alumnos, sobre todo alumnos que no cursaban la asignatura por primera vez, han indicado que de esta forma el número de problemas resueltos en clase es mucho menor.

- El realizar un gran número de prácticas ha creado controversia, por una parte los alumnos han considerado que ha sido positivo, ya que les permite llevar la asignatura al día de forma más amena, pero que por otro lado les obliga a realizar un trabajo adicional a las clases de laboratorio para poder realizar las prácticas. Lo que ha sido muy bien aceptado ha sido la corrección de las prácticas junto con el profesor, ya que les permite conocer errores cometidos y la forma de solucionarlos.

- Donde si han mostrado en general su desacuerdo ha sido en la temporalización de la asignatura, dado que el lenguaje C es el más utilizado en cursos posteriores, los alumnos consideran que han dispuesto de poco tiempo para su estudio.

9. BIBLIOGRAFÍA

[Abboud94] Abboud,M. *Problem Solving Design: A Pedagogical Approach.* Computer Science Education, Vol 5, Nro 1, 1994

[Antonakos97] Antonakos,J; Mansfield,K. *Programación estructurada en C.* Prentice Hall, 1997.

[Arnow98] Arnow,D; Weiss, G. *Introduction to programming using Java.* Addison-Wesley, 1998.

[Castro94] Castro, J. Cucker, F.; Messeguer,X.; Rubio, A; Solano, A; valles,B. *Curso de programación* McGraw-Hill, 1994.

[Feuer82] Feuer,A; Gehani,N. *A Comparison of the programming Languajes C and Pascal.* ACM Computer Surveys Vol 14 Núm1. 1982.

[Gil94] Gil, D.; Carrascosa, J.; Bringing *Pupils'learning Closer to a Scientific Construction of Knowledge: A Permanent Featura in Innovationes in Science Teaching* Science Education, Vol.78, Nro. 3, 1994.

[Gottfried97] Gottfried,B. *Programación en C. Segunda edición.* McGraw-Hill, 1997.

[Grogono96] Grogono,P. *Programación en Pascal. Edición revisada.* McGraw-Hill, 1996.

[Joyanes96a] Joyanes,L. *Fundamentos de programación. Algoritmos y estructuras de datos.* Segunda edición. McGraw-Hill, 1996.

[Pratt98] Pratt,T.;Zelkowitz,M. *Lenguajes de programación. Diseño e implementación.* Tercera edición Prentice Hall, 1998.

[Walker96] Walker, K.; Schach,S. *Obstacles to Learning a Second Programming language: An empirical study.* Computer Science Education, Vol 7, Nro 1, 1996.

[Wirth88] Wirth, N. *Programming in Modula-2* Springer-Verlag, Berlín 1988.