Tres Lenguajes Distintos y un Solo Objeto Verdadero: una Propuesta de Introducción a la Programación

Alberto Gómez Mancha, Julia González Rodríguez, Adolfo Lozano Tello
Área de Lenguajes y Sistemas Informáticos
Departamento de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: {agomez, juliagon, alozano}@unex.es

Resumen

Se presenta una propuesta de introducción a la programación para el primer curso de las titulaciones universitarias de Informática, tanto para las Ingenierías Técnicas como para la Ingeniería Superior. Todo el desarrollo del temario se basa en el paradigma orientado a objetos, utilizando como base un pseudocódigo, para después usar varios lenguajes en el desarrollo de las clases prácticas. Con esta idea se persigue que los alumnos no se centren en las características particulares de un lenguaje, sino en la comprensión de los conceptos fundamentales comunes a todos ellos.

1. Introducción y motivación

En la Escuela Politécnica de Cáceres, de la Universidad de Extremadura, actualmente se imparten tres titulaciones de Informática: Ingeniería Técnica en Informática de Sistemas, Ingeniería Técnica en Informática de Gestión e Ingeniería Informática. El plan de estudios actual entró en vigor en el curso 1998-1999.

Las asignaturas básicas de programación, comunes a todas las titulaciones, se distribuyen de la forma siguiente: durante el primer curso, dos asignaturas con una carga total de 15 créditos (7,5 de teoría y 7,5 de prácticas), y durante el segundo año, otras dos asignaturas con las mismas características.

- Elementos de programación: asignatura troncal anual de 9 créditos (6 teóricos y 3 prácticos), primer curso. Se imparten 3 horas de teoría y 2 de prácticas durante el primer cuatrimestre, y sólo una hora de teoría en el segundo, para coordinarla con la asignatura siguiente.
- Laboratorio de programación I: asignatura troncal del segundo cuatrimestre de 6 créditos (1'5 de teoría y 4'5 de prácticas), primer curso.
- Estructura de datos y algoritmos: asignatura troncal anual de 9 créditos (6 teóricos y 3 prácticos), segundo curso. Se imparten 4 horas semanales en el primer cuatrimestre, y 2 horas en el segundo.
- Laboratorio de programación II: asignatura obligatoria de segundo cuatrimestre, con 6 créditos (1'5 teóricos y 4'5 prácticos).

Las dos asignaturas del primer curso han ido variando gradualmente, año tras año, intentando adecuar, actualizar y mejorar los contenidos, los métodos y la evaluación, aunque algunos de los cambios previstos no se han podido llevar a cabo debido a problemas administrativos o de infraestructura.

Los contenidos del primer curso están orientados principalmente a la construcción de aplicaciones de tamaño medio con estructuras de datos simples (vectores, ficheros, estructuras dinámicas sencillas). El contenido formal es bastante limitado (por ejemplo, se inicia el estudio de la complejidad algorítmica), dejando la mayor parte para el segundo curso. El lenguaje utilizado para las prácticas actualmente es Pascal (con orientación a objetos en el segundo cuatrimestre).

Partiendo de conceptos básicos de programación, como variables, estructuras de control y módulos, se pretende desde el inicio hacer hincapié en el concepto y utilización de tipos abstractos de datos (TAD), que les ayuden

en el diseño y la implementación de programas. El paradigma orientado a objetos (POO) se introduce a finales del primer cuatrimestre, enfoque en el que se apoyan las prácticas de ‘Laboratorio de programación I’, aunque sin entrar en conceptos claves como la herencia y el polimorfismo, tratados en el segundo curso. Por tanto, seguimos un enfoque centrado en el objeto [1].

En general, los profesores que impartimos estas asignaturas estamos relativamente satisfechos con su contenido, aunque no con los resultados, y vemos la necesidad de hacer algunas modificaciones para actualizarlas y mejorar el nivel de aprendizaje de los alumnos.

Los principales problemas que tenemos son ajenos al contenido de las asignaturas, aunque influyen mucho en todas las decisiones posteriores: el elevado número de alumnos (bastantes de ellos poco motivados) y los pocos recursos, tanto humanos como técnicos. Por ejemplo, este curso tenemos 580 alumnos en 'Elementos de programación' y 666 en 'Laboratorio de programación I', con tres grupos de teoría y 18 de prácticas. Para intentar mejorar los resultados, sin bajar el nivel de exigencia, es necesario un mayor seguimiento y tutorización de los alumnos, tarea difícil en las condiciones anteriores.

Por lo referente a los contenidos de las dos asignaturas, se ve necesario un incremento de formalización en la explicación de los mismos. En muchos alumnos, sobre todo en aquellos que tienen algo de experiencia anterior en programación, se detecta una tendencia a ignorar las directrices de diseño aconsejadas, y conformarse simplemente con que el programa 'funcione', sin atender a su eficiencia y claridad. Con la introducción, aunque sea de manera informal, de conceptos de complejidad (que ya se explican actualmente), verificación y diseño, los alumnos podrán apreciar mejor la importancia de ser rigurosos a la hora de escribir programas.

También existe una gran dependencia del lenguaje utilizado por parte de los alumnos. Por algunos comentarios escuchados y que aparecen en encuestas realizadas en años anteriores, los alumnos parecen creer que están aprendiendo a 'programar en Pascal', no utilizando Pascal para aprender a programar. Nos encontramos con que los alumnos asocian directamente la implementación de un problema a este lenguaje, y relacionan la dificultad de diseñar el algoritmo a los problemas de manejo del compilador o la sintaxis concreta. Se acostumbran a la utilización de un entorno específico y les cuesta excesivo trabajo cambiar. Por esto, no saben aprovechar sus conocimientos de programación en otros ámbitos (como la programación en ensamblador para la asignatura de 'Introducción a los computadores'), y les cuesta cambiar a otros lenguajes posteriormente. Además, el uso de Pascal se entiende por parte de algunos alumnos como una falta de actualización de los contenidos de las asignaturas y un alejamiento de los lenguajes 'reales' de programación.

2. Propuesta de introducción a la programación sin vinculación a un lenguaje

En nuestra propuesta para un primer curso de introducción a la programación para alumnos de carreras de Informática queremos incidir en los conceptos abstractos, comunes a todos los lenguajes de programación que siguen el paradigma orientado a objetos, sin entrar en los detalles de los lenguajes particulares hasta que no sea necesario, y justamente tratándolos como detalles, como meras cuestiones sintácticas para conseguir una semántica determinada.

Con el enfoque actual, los alumnos aprenden los conceptos de programación en un lenguaje concreto, acarreando problemas de adaptación a otros lenguajes, no sólo debido al cambio en la sintaxis, sino al significado que los conceptos tienen para ellos.

Nuestra propuesta se basa en la explicación de los conceptos del temario en un pseudocódigo simple, con las características más comunes de los lenguajes orientados a objetos más utilizados. Con este código se explicarían los conceptos fundamentales y se resolverían los primeros problemas.

Como es necesario que el alumno vea también sus programas en ejecución, a la hora de realizar las prácticas, propondremos dos o tres lenguajes distintos (Pascal Orientado a Objetos, C++ y Java), viendo cómo se escriben las construcciones básicas del pseudocódigo en cada uno de ellos, y presentando sus características diferenciadoras. Para el desarrollo de las prácticas, nuestra idea es

utilizar tablas de conversión entre el pseudocódigo y los distintos lenguajes, de manera que el alumno pueda consultarlas en cualquier momento para resolver las dudas que se le presenten, hasta que adquiera experiencia en la sintaxis de cada uno de los lenguajes.

Con esta propuesta pretendemos que los conceptos principales queden mucho más claros, que los alumnos sean capaces de pensar de forma abstracta, sin darle tanta importancia a la codificación, centrándose más en el diseño y eficiencia del algoritmo. Además, aunque en las primeras semanas pueda costarles un trabajo suplementario la implementación en un lenguaje determinado, debido a la posible confusión entre los varios presentados, creemos que al finalizar el curso pueden llegar a un nivel aceptable en los lenguajes concretos, que afianzarán en cursos posteriores a medida que utilicen unos u otros lenguajes. Posiblemente, los errores cometidos serán fallos sintácticos sin demasiada importancia, que el compilador en la mayoría de los casos detectará. De la misma manera que un niño pequeño puede aprender dos idiomas distintos a la vez sin mezclarlos, y a un adulto le cuesta bastante más aprender un nuevo idioma, pensamos que se puede aprender a programar usando varios lenguajes distintos a la vez, sin los problemas del aprendizaje consecutivo de lenguajes.

La elección del paradigma y del primer lenguaje de programación en la enseñanza de la informática es un tema de debate abierto y en continua evolución. (Se puede ver, por ejemplo, cómo han ido variando las propuestas en los boletines y conferencias patrocinados por ACM [2]). Actualmente, muchas de las propuestas se centran en la programación orientada a objetos como paradigma inicial, y el uso de Java como primer lenguaje va siendo habitual. En ediciones anteriores de las JENUI se han presentado propuestas que inciden en la presentación consecutiva o simultánea de varios lenguajes de programación en cursos de programación básicos [3] [4] [5], lo que nos hace pensar que nuestra propuesta no es desacertada.

Además de variar el método de enseñanza buscando la focalización en el contenido y no en la forma, proponemos comenzar la introducción a la programación desde las primeras clases con Programación Orientada a Objetos (POO). La POO se adapta a las nuevas necesidades del mercado al que nuestros alumnos se enfrentarán, permite crear programas más reutilizables y, sobre todo, hace que el alumno idee y organice tanto las estructuras de representación, como las operaciones que se pueden realizar con estas estructuras para resolver problemas.

Basándonos en la POO desde el principio, y centrándonos en las definiciones de los objetos y sus necesidades, se inducirá al alumno a diseñar y analizar los programas de manera más eficiente y ordenada. Los objetos se diseñarán para que mantengan la información, de forma que no sólo puedan ser utilizados para el problema propuesto, sino que puedan ser reutilizados en otros.

Entendemos que aprender a programar mediante POO no crea una dificultad adicional para utilizar posteriormente programación procedural, ya que ésta se puede usar en la implementación de funciones auxiliares, así como en la implementación de métodos. Sin embargo, realizar el aprendizaje en sentido inverso, (aprender a programar en procedural para después estudiar el paradigma de POO), según nos indica nuestra experiencia crea confusiones y conflictos que llevan a la no utilización de este modelo o a su utilización ineficaz, donde a los alumnos les cuesta entender el porqué de las necesidades del diseño de objetos y de sus operaciones básicas.

3. Organización de las asignaturas

La puesta en marcha de la propuesta, que a nuestro entender ha de ser progresiva, se iniciará en el curso 2001/02, en el que se actualizarán las asignaturas de 'Elementos de Programación' y 'Laboratorio de Programación I', variando el año siguiente el programa de las asignaturas de segundo curso 'Estructuras de Datos y Algoritmos' y 'Laboratorio de Programación II'. De este modo, el cambio gradual afectará a una misma promoción, y el modelo de aprendizaje estará desarrollado en dos años consecutivos.

La propuesta, tal y como especificamos en el punto anterior, esencialmente consiste en enseñar a los alumnos a centrarse en conceptos fundamentales de programación olvidándose de los problemas de la sintaxis de un lenguaje concreto y los generados por su compilador. Para esto hemos diseñado un pseudocódigo que nos permite abstraernos de estas singularidades. Este

pseudocódigo es la base de la asignatura de 'Elementos de programación'; lo utilizaremos para aplicar y explicar los conceptos fundamentales tratados, así como la implementación de ejemplos. La carga práctica de esta teoría equivale a un 25% de la carga total de la asignatura. En sesiones prácticas de dos horas a la semana se implementan propuestas concretas de los problemas estudiados en las clases de teoría.

Las sesiones de prácticas se desarrollan proponiendo una serie de problemas relacionados con la materia teórica vista hasta ese momento. La propuesta consistirá en la realización de módulos que podrán ser reutilizados en sucesivas sesiones prácticas y que han de resolverse en el tiempo disponible.

Proponemos que en las sesiones prácticas se realicen implementaciones en los tres lenguajes: Pascal Orientado a Objetos, C++ y Java. Elegimos estos tres lenguajes porque son utilizados frecuentemente tanto en otros entornos académicos como en entornos laborales, existiendo para los tres suficiente bibliografía y material de apoyo para los alumnos. No desistimos de Pascal Orientado a Objetos por ser un lenguaje muy estructurado, de fácil aprendizaje, y que nos va a permitir reutilizar parte del material con el que contamos actualmente. Aunque presenta algunas dificultades en su enseñanza, seleccionamos C++ puesto que es un lenguaje muy utilizado en distintos ámbitos y fundamental en asignaturas específicas como Sistemas Operativos; además así se acercan a un lenguaje de nivel de abstracción medio, donde los mensajes proporcionados por el compilador no son tan precisos como los de Pascal; es un lenguaje más flexible y con más responsabilidad para el programador. Optamos como tercer lenguaje por Java, debido a la demanda de su conocimiento en el entorno laboral y su uso en otras asignaturas posteriores como 'Programación concurrente'. Además, este lenguaje promueve la creación de aplicaciones a partir de librerías y clases existentes, permite diseñar interfaces gráficas y aplicaciones orientadas a eventos fácilmente, y generar documentación externa a partir de los comentarios incluidos. El uso de Java va a motivar a los alumnos, ya que aprenden un lenguaje relacionado con Internet, de máxima actualidad.

Como hemos indicado, cada uno de los lenguajes seleccionados nos aporta características particulares, pero nuestra idea es enseñar su codificación aproximando sus estructuras sintácticas, evitando las singularidades propias de cada uno de ellos. El desarrollo de las sesiones prácticas se hará igual que hasta ahora, mediante la aplicación práctica de conceptos. La diferencia radica en la forma de implementación. Algunas sesiones prácticas se desarrollarán paralelamente en los tres lenguajes (sesiones de explicación de sintaxis y manejo de compiladores), en las que estarán ayudados por las **tablas de conversión (**ver la siguiente tabla abreviada de ejemplo).

Concepto	Pseudocódigo	Pascal OO	C++	Java
Declaración variables	`variables`	`var`		
Inicio de bloque Fin de bloque	`inicio` `fin`	`begin` `end`	`{` `}`	`{` `}`
Declaración de módulos	`módulo`	`function` `procedure`		
Declaración de clases	`clase`	`Object`	`class`	
Métodos privados Métodos públicos	`privado` `público`	`private`	 `public`	`private` `public`
Escribir	`Escribir` *`Cad`*	`WriteLn(`*`Cad`*`)`	`cout <<` *`Cad`*	`System.out.println(`*`Cad`*`)`
Sentencias condicionales	**`Si`** (*`condición`*) **`entonces`** *`bloque sentencias`* **`si no`** *`bloque`*	**`if`** (*`condición`*) **`then`** *`bloque de sentencias`* **`else`** *`bloque de`*	**`if`** (*`condición`*) *`bloque de sentencias`* **`else`** *`bloque de sentencias`*	**`if`** (*`condición`*) *`bloque de sentencias`* **`else`** *`bloque de sentencias`*

	sentencias **fin_si**	*sentencias*		

Las tablas de conversión son especificaciones sobre algún concepto de programación que se formulan con la sintaxis de varios lenguajes. Se facilitarán a los alumnos antes de comenzar las sesiones prácticas. En ellas aparecerán en el pseudocódigo utilizado en las clases de teoría y en los tres lenguajes citados anteriormente, el concepto expresado con las cuatro notaciones. Viene a ser como la piedra de Rosetta, donde el alumno puede ver en paralelo la relación que existe entre los conceptos indicados en pseudocódigo y la sintaxis formalizada en los tres lenguajes.

Aunque vayamos a utilizar varios lenguajes distintos, tampoco pretendemos que los alumnos acaben siendo expertos en todos ellos, ni que conozcan todas sus posibilidades. Así, donde sea posible, unificaremos y simplificaremos la sintaxis, siendo el pseudocódigo presentado el núcleo común a todos ellos. Por ejemplo, aunque en C++ o Java se pueden declarar variables en cualquier punto, en nuestras explicaciones siempre las definiremos al principio del método, como se hace en Pascal.

Las prácticas consistirán en la utilización de clases ya existentes para resolver problemas, y en la implementación de nuevas clases desde el principio, en cualquiera de los lenguajes de programación. En algunas prácticas, para potenciar la idea de que lo importante es el concepto subyacente, se realizará la misma implementación en los tres lenguajes, y puntualmente en otros casos, se dará un objeto escrito en un lenguaje y el estudiante deberá implementarlo en otro distinto. Existirá un proyecto práctico tutorizado de cierta importancia donde el tema será el mismo para todos los alumnos; ellos podrán elegir entre los tres lenguajes propuestos. La evaluación de las sesiones prácticas de la asignatura 'Elementos de Programación' consistirá en pruebas periódicas de implementación de pequeños problemas en los tres lenguajes, y la entrega del último proyecto propuesto. Con estos exámenes se evita que alumnos malintencionados intenten copiar las prácticas. Además, no sabrán en qué lenguaje tendrán que hacer el examen hasta que no se les proponga en su sesión. Por otro lado, la parte teórica se evaluará mediante un examen escrito. Este examen escrito tendrá como finalidad la resolución de varios problemas en el pseudocódigo aprendido. De esta forma diferenciaremos el diseño de algoritmos de la implementación en un lenguaje concreto con una sintaxis específica.

La asignatura 'Laboratorio de Programación I', pertenece al segundo cuatrimestre del primer curso. Cuando esta asignatura comienza, se han impartido dos tercios de la docencia de 'Elementos de Programación', y los alumnos cuentan con los conceptos fundamentales para afrontar proyectos de tamaño medio. Proponemos que estos proyectos sean explicados y desarrollados en pseudocódigo y que la implementación, siempre siguiendo el paradigma de la POO, sea libre en cuanto al lenguaje elegido.

Dejar que los alumnos decidan el lenguaje que van a utilizar en la implementación nos permitirá recabar información sobre la asimilación de los lenguajes, y sobre sus preferencias.

Los principales temas que se explicarán en estas dos asignaturas son, por este orden aproximadamente:

- Objetos básicos: Clases, instancias, atributos, métodos.
- Variables, tipos simples, expresiones.
- Módulos: procedimientos y funciones.
- Estructuras básicas de control.
- Recursividad.
- Complejidad temporal (notación O).
- Vectores.
- TAD lineales: uso e implementación estática.
- Gestión dinámica de memoria.
- Implementación dinámica de TAD lineales
- Ficheros secuenciales y de texto.

4. Ventajas y dificultades

Hemos comentando que con el uso de pseudocódigo para explicar los conceptos fundamentales de programación, y mediante la implementación de programas en las sesiones prácticas usando tres lenguajes de programación diferentes, perseguimos que el alumno desvincule el proceso de diseño de algoritmos de la sintaxis

de los lenguajes de programación usados en la implementación. También hemos argumentado que usando POO desde el principio presuponemos que los alumnos diseñarán sus programas de una forma más estructurada, consistente y reutilizable.

Además, esperamos que el uso desde el principio de varios lenguajes y entornos de programación no sea una dificultad añadida, sino una ventaja en los cursos posteriores, donde actualmente se detectan problemas al cambiar de lenguajes y paradigmas.

En cambio, vemos a priori algunos inconvenientes que pueden surgir al utilizar este método. En primer lugar, aunque hagamos siempre el paralelismo entre el pseudocódigo y cada uno de los lenguajes de programación, puede que a estos programadores noveles les cueste recordar la sintaxis de estos cuatro lenguajes y mezclen aspectos de sus gramáticas. Para solucionar esto, somos conscientes de que tenemos que poner excesivo cuidado en hacer una simetría con las estructuras sintácticas de todos los lenguajes, no comentando más posibilidades en los formatos de las instrucciones de las que nos convienen para hacer este paralelismo.

Además, para recordar las estructuras sintácticas de los formatos de las instrucciones, facilitaremos las tablas de conversión, que en cualquier momento (tanto en el desarrollo de sus prácticas como en los exámenes) el alumno podrá consultar.

Por otro lado, tendremos que hacer frente al hecho de que vamos a enseñar iniciación a la programación mediante el paradigma de la POO. Aunque inicialmente no vamos a utilizar aquellos conceptos que dotan de mayor potencia a la POO (herencia, polimorfismo), pretendemos que estén familiarizados con los conceptos básicos y sobre todo con la metodología de diseño.

5. Conclusiones

En este trabajo se propone un método de iniciación a la programación que abarcará el cambio en el temario de dos asignaturas troncales del primer curso de las titulaciones de Informática impartidas en la Escuela Politécnica de Cáceres, en la Universidad de Extremadura. Su principal característica es evitar la vinculación del desarrollo de programas con un lenguaje de programación concreto. Para ello, proponemos que los conceptos teóricos sean explicados con un pseudocódigo, y las prácticas de aplicación de estas nociones teóricas sean llevadas a cabo mediante tres lenguajes de programación diferentes. Buscamos de esta forma que el alumno sea capaz de diferenciar el diseño de un programa de la sintaxis del lenguaje de programación en el que se implemente. Además, proponemos que el aprendizaje de la programación se realice desde el principio mediante POO, ya que estamos convencidos que con este paradigma se consiguen programas mejor estructurados, legibles y reutilizables.

En la puesta en marcha de esta iniciativa prevemos dos dificultades: el alto número de alumnos y la escasez de recursos con los que contamos. Por estas razones, para el curso que viene, deberemos iniciar esta propuesta empleando sólo dos de los tres lenguajes, Pascal y C++ (que ya se emplean en las asignaturas de los dos primeros cursos), y aplazar el uso de Java hasta poder adecuar los recursos con los que contamos y ver cómo resulta la experiencia.

Referencias

[1] Joel C. Adams. *Object-Centered Design. A Five-Phase Introduction To OOP In CS1-2.* ACM SIGCSE Bulletin Vol.28, No.1, Pág.78-82, Marzo 1996.

[2] SIGCSE: ACM Special Interest Group in Computer Science Education. http://www.acm.org/sigcse.

[3] Fernando Llopis Pascual. *Propuesta de metodología para un curso universitario introductorio a la programación.* Actas IV JENUI, Andorra, 1998.

[4] Antonio Ortega Tello y colegas. *Programación Multilenguaje con Component Pascal y Java en un 1º Curso de Programación.* Actas V JENUI (La Almunia de Dª Godina), 1999.

[5] Xavier Franch y Joaquim Gabarró. *Transición de Modula-2 a Java en un curso de iniciación a la programación.* Actas V JENUI (La Almunia de Dª Godina), 1999.