

Propuesta para la enseñanza de Informática en titulaciones de Ingeniería Química

Ana Belén Moreno Díaz
Dept. de Informática, Estadística y
Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
e-mail: a.b.moreno@escet.urjc.es

Juan José Pantrigo
Dept. de Informática, Estadística y
Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
e-mail: j.j.pantrigo@escet.urjc.es

Rosalía Peña
Dept. de Informática, Estadística y
Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
y Dept. CC. Computación.
Universidad de Alcalá de Henares
28871 Alcalá de Henares (Madrid)
e-mail: rpr@uah.es

Resumen

Este trabajo describe el enfoque y desarrollo de los cursos de introducción a la informática impartidos en las titulaciones Técnica y Superior de Ingeniería Química en la Universidad Rey Juan Carlos de Madrid. Desde hace varios años, hemos venido realizando cambios en las asignaturas para adecuar su contenido a las necesidades de los ingenieros químicos, aumentar el grado de consecución de los objetivos y fomentar el interés y aprovechamiento de los alumnos. En este artículo presentamos nuestra experiencia. Se analizan los aspectos que se deben considerar al elaborar los temarios. Se evalúa la adecuación de *Fortran90* como lenguaje soporte para un primer acercamiento a la programación y se justifica su elección. Se estudia qué tipo de prácticas contribuye a aumentar la motivación de los alumnos y favorece el aprendizaje de la metodología de la programación.

1. Introducción

En el diseño de una asignatura hay que considerar diversos factores: los descriptores oficiales, las recomendaciones de las asociaciones profesionales, la evolución y el estado actual de la materia, el plan de estudios, la formación previa del alumnado, el número de créditos teóricos y prácticos y la experiencia de los profesores.

En este trabajo se expone y justifica la propuesta de las asignaturas denominadas "Informática" y "Fundamentos de Informática", de las titulaciones *Ingeniería Química* (Superior) e

Ingeniería Técnica Industrial en Química Industrial (Técnica), respectivamente. Ambas son obligatorias, y se imparten en el primer cuatrimestre del primer curso (15 semanas), con cinco horas lectivas por semana, dos de teoría y tres de práctica.

Al elaborar los temarios hemos tenido en cuenta, entre otros factores, la opinión de profesores del área de *Ingeniería Química* de esta universidad.

Al tratarse de una introducción a la informática de corta duración, el lenguaje de programación debe ser imperativo, sencillo, expresivo y fácil de asimilar, para disponer de más tiempo para el trabajo práctico.

Para motivar a los alumnos se proponen prácticas cercanas a la ingeniería, que en numerosas ocasiones, requieren conocimientos sobre química, matemáticas o física, haciendo énfasis en la metodología de la programación estructurada y del diseño descendente y en la adquisición de buenos hábitos. A la elaboración de este tipo de prácticas ha contribuido que los profesores (y autores del trabajo), pertenecientes al área de Lenguajes y Sistemas de Información e Ingeniería del Software, son titulados superiores en Química o en Física.

La estructura del artículo es la siguiente: la sección 2 presenta los conocimientos de informática más relevantes en el ámbito de la Ingeniería Química. La sección 3 expone los objetivos docentes. La sección 4 resume el contenido teórico y el material bibliográfico. La sección 5 justifica la elección del software. La sección 6 describe la organización y el enfoque de

los contenidos prácticos, mostrando algunos ejemplos. La sección 7 concreta la metodología empleada para el seguimiento de la asignatura y los criterios de evaluación. En la sección 8 se valora esta oferta docente, proponiendo futuras mejoras y en la sección 9 se exponen las conclusiones.

2. Informática en Ingenierías Químicas

Los titulados en Ingeniería Química en el mercado laboral en España van a necesitar para su ejercicio profesional conocimientos de informática, por lo que se recomienda su inclusión en los planes de estudio. Los conocimientos que requerirán con mayor probabilidad [1][2][3][4][5] son:

- *Ofimática e Internet.*
- *Diseño asistido por computador (CAD)*, mediante paquetes como *AutoCAD*, *AutoSketch* o *CADKEY*, que permiten definir objetos (por ejemplo, componentes de plantas químicas) y manipularlos gráficamente.
- *Simuladores de procesos químicos*: además de las funciones programadas, suelen proporcionar flexibilidad, admitiendo módulos externos (programados muy frecuentemente en *Fortran90* o *Visual Basic*).
- *Paquetes de cálculo matemático*, como *MATLAB* o *Maple*, que disponen de librerías predefinidas para la realización de cálculos y visualización de gráficos.
- *Programación*: las herramientas comerciales no resuelven todos los problemas, a veces es necesario realizar desarrollos propios. Además, en los casos en que el entorno de trabajo cuenta con equipos multidisciplinarios, el conocimiento de conceptos de programación favorece la comunicación entre químicos e informáticos.

2.1. Elementos curriculares

Los contenidos de nuestras asignaturas siguen las directrices de las asociaciones de profesionales de Informática y Química sobre formación en informática. El Documento de Política de Educación Científica [2] de la *American Chemical Society* recomienda el uso habitual de ordenadores modernos y el acceso a Internet en las titulaciones de Química. También recomienda que la

simulación por ordenador complemente, pero no sustituya, al laboratorio de química.

En el *Computing Curricula 2001* [6] (Volumen *Computer Science* [7]) se dan recomendaciones para cubrir la docencia de informática en titulaciones no informáticas. La definición de cursos de informática para estas titulaciones ha de seguir las fases:

- *Especificación*: los objetivos del curso se deben consensuar entre profesores informáticos y no informáticos (en este caso, químicos).
- *Diseño*: los objetivos educativos se tienen que concretar, especificando las destrezas y conceptos que deben alcanzar los alumnos.
- *Implementación*: para la estructuración del curso se deben tener en cuenta los conocimientos previos, los métodos de trabajo, las motivaciones de los alumnos y los recursos disponibles.
- *Evaluación*: debe ser activa y redundar en una constante actualización.

2.2. Formación previa de los alumnos

A estas asignaturas se incorporan todos los años 75 alumnos nuevos, y entre veinticinco y treinta repetidores; en total, hay cien alumnos de media en cada grupo. La mayoría de los alumnos de la Superior, provienen de selectividad. Sin embargo, la procedencia de los alumnos de la Técnica es de Módulos de Formación Profesional, Bachillerato y de Titulaciones de Ingeniería Superior que a veces no han superado. La calificación necesaria para acceder a esta titulación es menor, por lo que su formación previa es inferior. En cuanto a sus conocimientos de informática, la situación es también muy variada: algunos conocen un lenguaje de programación (normalmente Pascal, C o Basic) desde una perspectiva muy práctica, pero deficiente desde el punto de vista conceptual y metodológico, lo que dificulta nuestra tarea. Otros solamente conocen el sistema operativo *Windows* y el manejo de algunos programas, y para un 30% de los alumnos este es su primer contacto con el ordenador.

2.3. Contexto académico

En cuanto a las materias relacionadas con la que nos ocupa conviene señalar que en ambas titulaciones hay una asignatura de Diseño Asistido por Ordenador, en la que se aprende el manejo de *AutoCAD*. En algunas asignaturas de los últimos cursos los alumnos trabajan con simuladores de procesos químicos y para cursarlas conviene tener conocimientos de programación. En las asignaturas de Matemáticas realizan prácticas de laboratorio utilizando *MAPLE*. También existe una asignatura de libre elección sobre Internet. En el tercer curso de la titulación Superior se imparte una optativa de ofimática y *Visual Basic*.

3. Objetivos de la asignatura

Debido a que gran parte de nuestros alumnos no ha tenido contacto con un ordenador anteriormente, esta asignatura tiene un marcado carácter introductorio. Por otro lado, ésta es la única asignatura obligatoria en la que se estudian conceptos de informática, por lo que debe cubrir las necesidades profesionales del ingeniero. Por estos motivos, el objetivo central es poner a disposición del alumno *los conocimientos necesarios para que consiga identificar y comprender la utilidad de los conceptos fundamentales de la informática, haciendo énfasis en el algoritmo y en la estructura física y lógica de los ordenadores actuales*. Concretamente, se pretende que, finalizado el proceso formativo, el alumno:

- conozca de modo sucinto los conceptos básicos referentes a la evolución de la informática hasta nuestros días,
- comprenda cómo funciona un ordenador y conozca sus componentes básicos,
- conozca las funciones del sistema operativo, y maneje el entorno de uno de ellos,
- conozca los elementos básicos de un lenguaje de alto nivel y su sintaxis en *Fortran90*, más concretamente se presta atención a: (a) tipos de datos simples y compuestos; (b) estructuras secuenciales, selectivas, repetitivas y subprogramación, (c) normas de estilo de programación referentes a la construcción y documentación de programas;

- conozca algoritmos clásicos (recorrido, búsqueda, ordenación, entre otros),
- sea capaz de diseñar algoritmos y de crear programas sencillos utilizando la metodología de la programación estructurada y el diseño descendente,
- conozca la utilidad de algún paquete de software científico para la Ingeniería Química, y maneje *MATLAB* (este objetivo sólo se introduce en la titulación Superior).

4. Contenido teórico del curso

Los contenidos teóricos de la asignatura *Informática* (Superior) están divididos en tres partes. La primera ocupa un cuarto del curso y trata los conceptos básicos de la informática. En la segunda se lleva a cabo una introducción a la programación estructurada con ayuda de *Fortran90*, y ocupa la mayor parte del cuatrimestre. La tercera, y menos extensa, se dedica al estudio de los paquetes de software matemáticos, utilizando *MATLAB*. A continuación, se muestra un esquema de los temas especificando las horas teóricas y prácticas:

Parte I: Introducción a la Informática

Tema 1: Conceptos básicos (3T+1P)

Tema 2: Arquitectura de ordenadores (3T+1P)

Tema 3: Sistemas operativos (3T+4P)

Parte II: Programación en lenguaje *Fortran90*

Tema 4: Los primeros pasos en la programación en *Fortran90* (2T)

Tema 5: Tipos de datos, constantes, variables y expresiones. Sentencias de asignación y entrada/salida (4T+7P)

Tema 6: Sentencias de selección (2T+5P)

Tema 7: Sentencias de repetición (2T+7P)

Tema 8: Metodología del diseño descendente. Subprogramas (3T+5P)

Tema 9: Arrays (3T+7P)

Tema 10: Ficheros (2T+4P)

Parte III. Paquetes de software científico

Tema 11. Software de cálculo científico para la Ingeniería Química (3T+4P)

En la titulación Técnica se omiten los apartados *registros* del tema 5, *módulos* del tema 8 y los *paquetes de software científico*.

Al principio de cada tema se presentan los objetivos que los alumnos deben alcanzar. Las

clases teóricas se imparten con la ayuda de un ordenador portátil y un cañón de proyección. Se recomienda al alumno el uso de bibliografía.

No hemos encontrado un único libro que abarque la asignatura en su totalidad, siendo los libros más utilizados:

- J. Glenn Brookshear, *Introducción a las ciencias de la computación*, 4ª Edición, Addison-Wesley Iberoamericana, 1995.
- F. García Merayo, *Lenguaje de Programación Fortran90*, Ed. Paraninfo, 1999.
- T.M.R. Ellis, Ivor R. Philips and T.M. Lahey, *Fortran90 programming*, Addison.Wesley 1994.
- C. Pareja, A. Andeyro y M. Ojeda, *Introducción a la Informática: I. Aspectos Generales*, Ed. Complutense, 1994.
- [Etter97] Delores M. Etter, *Solución de problemas de Ingeniería con MATLAB*, 2ª edición, Prentice Hall, 1997.

5. Otros aspectos considerados

A continuación se justifica la elección del sistema operativo, el lenguaje, el entorno de programación y el paquete de software científico.

5.1. Elección del sistema operativo

Es fundamental entender el contexto en el que se ejecutan los programas en el ordenador, y por tanto, el alumno debe comprender los conceptos fundamentales del sistema operativo. En entornos personales, el usuario debe ser autónomo, al menos en el manejo cotidiano de sus ficheros. Sin embargo, en grandes sistemas, se suele disponer de personal especializado para administración y mantenimiento, que dará soporte al especialista Químico que se acerca al sistema con unos fines concretos.

Adicionalmente, se desea exista la mayor disponibilidad de puestos para los alumnos en horas de trabajo personal, todo ello nos conduce a elegir *Windows* como sistema operativo.

5.2. Valor pedagógico de *Fortran90*

Varios autores [8-9] han estudiado las características que deben poseer los lenguajes utilizados para el primer acercamiento a la

programación imperativa, coincidiendo en las siguientes:

- 1 un sistema de tipos fuerte
- 2 simplicidad
- 3 entrada/salida sencillas, para poder presentarlas desde el inicio del curso
- 4 homogeneidad (sistemática o coherencia en su sintaxis)
- 5 controles que eviten errores conceptuales
- 6 todos los mecanismos necesarios para facilitar la programación modular y estructurada
- 7 características que permitan fomentar buenos hábitos de programación (comentarios, sangrado sistemático de los bloques de ejecución, por ejemplo), y
- 8 tipos de datos derivados.

A continuación, se examina cada una de estas características en *Fortran90*, comentando la forma en que las implementamos en nuestra docencia.

- 1 Se fuerza la declaración explícita de tipos mediante la instrucción `IMPLICIT NONE`. *Fortran90* dispone de funciones intrínsecas de conversión entre tipos, permitiendo no presentar las reglas de compatibilidad y forzar la conversión explícita, impidiendo expresiones de tipos mixtos.
- 2 *Fortran90* es un lenguaje muy rico, versátil y potente, que no se puede calificar de simple; sin embargo, en nuestra aproximación docente, hemos optado por introducir, exclusivamente, las estructuras que posibilitan la programación modular y estructurada y que detallamos en los puntos siguientes. Fomentar el empleo de las estructuras más generales del lenguaje conlleva la ventaja de que las habilidades adquiridas por nuestros estudiantes son directamente transportables a otros lenguajes imperativos, con sólo disponer de una tabla de conversión de la sintaxis.
- 3 Del mismo modo, optamos por restringirnos a la entrada/salida sin formato que es sencilla e intuitiva:

```
READ*, variable[,variable]n
PRINT*, {expresión/literal}
[, {expresión/literal}]n
```

Sintaxis 1. Entrada/salida

- 4 Las estructuras que presentamos a los alumnos disponen sistemáticamente de cabecera y fin de estructura. El hecho de que el fin sea

específico para cada una (IF/END IF, FUNCTION/END FUNCTION, DO/END DO, etc.) aumenta la legibilidad frente al uso del mismo terminador para todas ellas. La sintaxis es homogénea, de modo que los modificadores o parámetros de cada estructura se ubican de forma sistemática (en su versatilidad, *Fortran90* posibilita formas abreviadas, o sintaxis alternativas, pero manejamos solamente la forma más estándar). Cada instrucción por defecto es una línea, pero puede especificarse su continuidad en los casos excepcionales en que se necesite (& carácter de continuación).

- 5 *Fortran90* dispone de controles que impiden realizar manipulaciones inadecuadas, por ejemplo: (a) modificar la variable de control de un bucle con contador dentro de su ámbito; (b) la dirección del flujo de los argumentos de subprogramas se declara de forma explícita IN|OUT|INOUT; (c) por defecto, todas las variables declaradas, tanto en el programa principal como en subprogramas son locales, aunque existe la posibilidad de otorgar visibilidad, expresándolo de forma explícita.
- 6 Las estructuras selectivas que manejan nuestros alumnos son IF y CASE (Sintaxis 2). Las tres estructuras repetitivas clásicas son (Sintaxis 3): bucle controlado por contador y bucles controlados por expresión lógica pre y post-probados. Aunque la sintaxis de esta última pueda generar, en principio, alguna reticencia, su funcionalidad es exactamente la misma que el REPEAT/UNTIL de Pascal, saliendo a la instrucción inmediatamente posterior a END DO si se satisface la condición, e iniciando una nueva iteración en caso contrario, de modo que no hay motivo para omitirla.

```

IF (exp_logica) THEN
    conjunto_instrucciones
[ELSE conjunto_instrucciones]
END IF

SELECT CASE (exp_E|C|L)
[CASE (valor:valor)
    conjunto_instrucciones]1-n
[CASE DEFAULT
    conjunto_instrucciones]
END SELECT

```

Sintaxis 2. Estructuras de selección

```

DO var_Entera=valInicial,valFinal
    [,incremento]
    conjunto_instrucciones
END DO

```

```

DO WHILE (expresión lógica)
    conjunto_instrucciones
END DO

```

```

DO conjunto_instrucciones
IF (expresión_lógica) EXIT
END DO

```

Sintaxis 3. Estructuras de repetición

Como ya se ha comentado, se declara explícitamente la dirección del flujo de los argumentos de los subprogramas, conforme a la Sintaxis 4:

```

SUBROUTINE nombreSubrutina [(arg
    [,arg]n)]
    [tipo, INTENT(IN|OUT|INOUT)]::
    arg [,arg]n]
END SUBROUTINE nombreSubrutina

```

```

tipo FUNCTION nombreFuncion
    ([argumento][,argumento]n)
    [tipo, INTENT(IN)]::arg[,arg]n]
END FUNCTION nombreFuncion

```

Sintaxis 4. Estructuras de selección

- 7 *Fortran90* dispone de tipos de datos derivados, y las variables de éstos pueden ser empleadas en cualquier lugar en que se pudiera usar una variable de tipo simple, incluido el valor devuelto por las funciones.
- 8 Se pueden incluir espacios en blanco, tabuladores, líneas en blanco y comentarios, en cualquier punto del programa, lo que permite imponer estilos de sangrado, y documentación adecuados. Adicionalmente, *Fortran90* no es sensible al modo mayúscula/minúscula de los caracteres, y los identificadores pueden tener hasta 31 caracteres, características que facilitan la diferenciación entre los componentes del lenguaje y la elección adecuada de identificadores. Hemos optado por la escritura de las palabras reservadas del lenguaje en mayúsculas y los identificadores en minúsculas, a excepción de la primera letra de cada palabra en los nombres compuestos. Entendemos que establecer desde el primer momento del curso estas

pautas facilita la legibilidad del código y la conveniencia de adoptar una metodología.

Estas características permiten concluir que *Fortran90* es pedagógicamente adecuado para la ejercitación de alumnos noveles.

5.3. Elección del lenguaje

El hecho ser uno de los lenguajes más empleados en la programación de simuladores en química nos condujo a evaluarlo para su utilización en estas asignaturas.

Además, *Fortran90* es muy potente para cálculos matemáticos, concretamente, para el manejo de *arrays*, tan frecuentemente utilizados en Química (por ejemplo, para el estudio de energías de enlace o simetrías moleculares); en particular, es posible realizar cálculos entre *arrays* n-dimensionales con la misma sintaxis empleada para variables simples (Sintaxis 5).

```
INTEGER, DIMENSION [N, M] : a, b, c
c=2*b+a-c
PRINT*, c
```

Sintaxis 5. Declaración y uso de arrays

En [10] se describen más características de este lenguaje.

Por todo lo anterior y por su adecuación pedagógica, llegamos a la conclusión de que *Fortran90* es una elección correcta para nuestra asignatura.

5.4. El entorno de programación

Para Lahey [8] las características pedagógicas del entorno de programación son tan importantes como las del lenguaje en el entrenamiento de alumnos principiantes y propone el entorno *Elf90*, desarrollado específicamente para fines docentes.

Las sesiones prácticas de programación se están desarrollando sobre *Plato* (de *Salford*). Es un entorno que permite tener abiertos simultáneamente varios archivos, copiar información de unos a otros y compilar y ejecutar sin salir de la edición. Sin embargo, los mensajes del compilador son muy escuetos, con frecuencia imprecisos y no dispone de utilidades para trazar y depurar. La mejora de estos aspectos haría más confortable y eficiente el aprendizaje. Con frecuencia, en nuestras universidades resulta más

difícil adquirir software, que incurrir en cualquier otro tipo de gastos, este es el motivo de mantener el actual entorno.

5.5. Elección del paquete de software científico

Es aconsejable que en estas titulaciones se aprenda algún lenguaje interactivo, con herramientas para la producción de gráficos y construcción de prototipos, como *Mathematica*, *MATLAB*, *Maple* o *MATCAD*, siendo todos ellos potentes herramientas de cálculo matemático. Nuestros alumnos trabajan con *Maple* en asignaturas de matemáticas. Con objeto de que se aproximen a otro paquete de software científico, muy extendido en las empresas del sector, se ha escogido *MATLAB*, que posee una amplia gama de librerías específicas para distintas especialidades de la ingeniería; por su sencillez de manejo y por la semejanza de su sintaxis a la de *Fortran90*, resulta adecuado para esta asignatura [11], ya que una vez conocido *Fortran90*, es rápido, sencillo, intuitivo y casi inmediato de aprender.

6. Prácticas del curso

Los 4,5 créditos prácticos se invierten en la realización de ejemplos, ejercicios y prácticas de programación. Semanalmente se dedica una hora a resolver y comentar ejercicios en el aula y dos en el laboratorio.

Durante el curso se proponen alrededor de 60 ejercicios en hojas de problemas, de los que gran parte se resuelven posteriormente en clase y del resto se proporcionan las soluciones comentadas. Los ejercicios propuestos cubren preguntas cortas sobre el contenido de la parte I del curso, construcción de programas sencillos en *Fortran90* sobre cada uno de los temas y ejercicios de *MATLAB*.

Las prácticas de *MS-DOS*, *Windows* y *MATLAB* son tutoriales que contienen conceptos teóricos intercalados con ejercicios prácticos. Las prácticas de programación tienen carácter obligatorio y deben ser entregadas al profesor. Cada una de ellas consiste en la realización de un programa que ayuda a sedimentar los conceptos teóricos correspondientes al capítulo más reciente, ponerlos en práctica y descubrir su utilidad para

resolver problemas. Como muestra, se describen algunas de estas prácticas:

- **Integral definida**

Objetivo: Practicar el diseño y la codificación de algoritmos con estructuras de repetición controladas por contador.

Práctica: Se propone la construcción de un programa para calcular una integral definida de una función conocida, mediante el método de los trapecios. Este método aproxima el valor de la integral de una función en un intervalo $[a, b]$ de la recta real, a la suma de las áreas de los trapecios bajo la curva, que resultan al efectuar N divisiones en el intervalo.

- **Combinatoria**

Objetivo: Practicar el diseño y la codificación de algoritmos que usen estructuras de selección y repetición controladas por contador, por expresión lógica y subprogramación.

Práctica: Se propone la construcción de un programa que ofrezca al usuario un menú en el que puede optar entre calcular combinaciones ($Cnk = n!/[(k!)(n-k)!]$), permutaciones ($Pnk = n!/k!$), o salir. Posteriormente se pedirán los datos n y k . Puesto que el factorial de un número se requiere en 5 puntos del programa, el alumno descubre la utilidad de codificar una función para calcularlo. El algoritmo iterativo para el cálculo del factorial es conocido por el alumno.

- **Química del Carbono**

Objetivo: Practicar el diseño y la codificación de algoritmos que usen cadenas de caracteres, estructuras de selección y repetición controladas por expresión lógica y subprogramación.

Práctica: Se propone la construcción de un programa que analice una fórmula de la química del carbono proporcionada por el usuario en una cadena de caracteres. Este programa debe llamar a un subprograma que valide la entrada (caracteres permitidos: 'C', 'H', 'O', 'N', '2', '3', '4', '5' y '6') y si es correcta, calcula el número de átomos de cada

tipo que contiene la fórmula y su peso molecular.

En todo momento se respetan unas normas de estilo acordadas, que unifican la notación empleada, haciendo más legible el código. Estas normas hacen referencia, entre otras cuestiones, a la elección de identificadores significativos, uso de constantes no literales, tabulado de instrucciones según su bloque de ejecución, inclusión de comentarios (para especificar objetivos, valores de entrada y salida, pre y post-condiciones y documentar segmentos no obvios), separación de bloques de programa, evitar operaciones mixtas e impresión de literales explicando al usuario el dato pedido o mostrado.

7. Metodología y evaluación de alumnos

Las prácticas de ordenador son obligatorias. El alumno las entrega durante la semana siguiente a la que fueron propuestas, de manera que tiene un tiempo razonable para realizarlas. Se devuelven comentadas semanalmente, de forma que el alumno conoce la calidad de su trabajo y puede corregir sus errores.

Considerando el número de alumnos proponemos un método tradicional de evaluación, se realiza un examen final escrito, dividido en dos partes: teoría y práctica. Han de superar ambas para aprobar la asignatura. El alumno que suspende las prácticas en febrero, debe realizar otras distintas para septiembre. Cuando todas las prácticas están aprobadas, sus puntuaciones se promedian y normalizan entre 0 y 1 y se suman a la nota del examen final si este está aprobado. De esta forma se fomenta el seguimiento semanal de las prácticas.

La realización de hojas de problemas es voluntaria, pero muy recomendable para incrementar las habilidades de programación.

8. Evaluación de la asignatura

Algunos puntos de las encuestas docentes relacionados con el interés de la asignatura han mejorado al acercar los contenidos prácticos a problemas relacionados con su ámbito de conocimiento. A los alumnos, en general, les gusta la programación, aunque la encuentran

difícil. Valoran positivamente que se les entreguen comentadas sus prácticas de forma personalizada.

Reclaman disponer de un libro en español que cubra los conceptos de la metodología de la programación estructurada y el diseño descendente sobre *Fortran90* y que presente ejemplos y ejercicios.

Los resultados obtenidos en las dos titulaciones son diferentes. Un mayor número de alumnos de la titulación Técnica tiene dificultades para alcanzar los objetivos. Es posible que esto sea debido a la diferente formación previa.

Consideramos adecuado reducir la primera parte del temario para dedicar más tiempo a la programación.

Dado que la informática está cada vez más extendida, posiblemente el manejo del entorno del sistema operativo se podrá realizar de manera optativa, en horarios no lectivos, en los próximos cursos.

9. Conclusiones

Los estudiantes de Ingeniería Química necesitan conocimientos de: ofimática, *CAD*, paquetes matemáticos y desarrollo de habilidades algorítmicas, que les doten de cierta autonomía y les faciliten la comunicación en equipos de trabajo multidisciplinarios. La responsabilidad de buena parte de dicha formación recae en estas asignaturas.

El contenido de la asignatura ha tenido que ser ajustado en cada una de las titulaciones, en función de las características del colectivo.

Fortran90 muestra ser un lenguaje pedagógicamente adecuado como soporte en un curso de iniciación a la programación.

Fortran90 y *MATLAB* son una buena elección para alumnos de ingenierías químicas.

El acercamiento de las prácticas a su contexto contribuye a que los alumnos encuentren mayor sentido a la presencia de esta asignatura en su plan de estudios, mejorando su interés y rendimiento.

En un futuro próximo, se tenderá a trasladar el aprendizaje del entorno de *Windows* a seminarios fuera de horas lectivas y a mejorar el entorno de programación.

Referencias

- [1] L. E. Romero Zúñiga, *Los titulados en ingeniería química ante el mercado laboral*, Ingeniería Química, Julio-Agosto 2000, pp. 201-206.
- [2] *Education Policies for Sustainable Reform*, Society Committee on Education SOCED of the American Chemical Society ACS, Science, 2001.
- [3] American Institute of Chemical Engineers, <http://www.aiche.org/education/>
- [4] Asociación Nacional de Químicos de España, <http://www.anque.es/index.htm>
- [5] European Federation of Chemical Engineering, <http://www.efce.info>.
- [6] The Joint Task Force on Computing Curricula IEEE-CS/ACM. *Computing Curricula 2001*. <http://www.computer.org/education/cc2001/>
- [7] The Joint Task Force on Computing Curricula IEEE-CS/ACM. *Computing Curricula 2001*, Computer Science Volume, December 2001, <http://www.computer.org/education/cc2001/final/>.
- [8] Lahey T. M., Walker T., *Elf90-A first Programming Language*, Proceedings ASEE Annual Conference & Exposition, Washington, I, 1996.
- [9] Kölling, M. et al. *Requirements for a first year OO teaching language*, SIGCSE 1995, 173-77.
- [10] Lahey T. M., *A look at Fortran90*, <http://www.lahey.com/lookat90.html>.
- [11] Delores M. Etter, *Solución de problemas de Ingeniería con MATLAB*, 2ª edición, Prentice Hall, 1997.