INICIACIÓN A LA INVESTIGACIÓN
REVISTA ELECTRÓNICA
UNIVERSIDAD DE JAÉN

Ini Inv, e5: a8 (2013)

Experiencia en el Grado en Ingeniería Informática para reforzar conceptos algebraicos a través de la Programación de Ordenadores

Eva Gibaja, Amelia Zafra, María Luque y Sebastián Ventura

Departamento de Informática y Análisis. Universidad de Córdoba. Campus de Rabanales, 14071, Córdoba.
egibaja@uco.org

Resumen

Este trabajo presenta la experiencia llevada a cabo en primero de Grado en Ingeniería Informática en las asignaturas de Programación. Esta experiencia ha estado orientada al refuerzo de conceptos de matemática discreta y álgebra lineal. Para ello planteamos dos recursos didácticos en los que el alumnado pone en práctica no solo los conceptos vistos en clase de programación, sino también los contenidos de otras asignaturas del mismo cuatrimestre, favoreciendo el refuerzo y la interiorización de algoritmos y conceptos matemáticos que suelen resultarle difíciles de aprender.

INTRODUCCIÓN

La programación es una materia cuyo aprendizaje requiere de la realización de numerosos ejercicios prácticos de diferente complejidad para que el alumno practique los conceptos aprendidos. Ya que en última instancia un programa se desarrollará para solucionar un problema real, es razonable considerar que las asignaturas de programación constituyen un lugar desde el que practicar programando no solo los conceptos propios de programación, sino también conceptos relacionados con otras asignaturas que el alumno está cursando.

Este artículo presenta la experiencia docente en dos asignaturas: *Introducción a la Programación* (IP) y *Metodología de la Programación* (MP) del Grado en Ingeniería Informática que se impartirse en la Universidad de Córdoba.

El Grado consta de 4 cursos, el primero de ellos constituye el *módulo de formación básica* con contenidos de Informática, Matemáticas, Informática y Empresa. Hemos enfocado las prácticas realizadas en estas asignaturas a la resolución mediante ordenador de conceptos sobre aritmética modular y criptografía que se imparten de forma simultánea en otras dos asignaturas *Álgebra lineal* y *Matemática discreta*.

Para ello presentamos dos propuestas: una para la asignatura IP que se imparte en el primer cuatrimestre y otra para MP que se imparte en el segundo cuatrimestre de tal modo que los alumnos, además de practicar conceptos sobre programación (Joyanes y Zahonero, 2005. *Programación en C. Metodología, algoritmos y estructuras de datos*. Mc Graw Hill; Joyanes et al., 2003. *Programación en C. Libro de problemas*. Mc Graw Hill; Kernigham and Ritchie, 1989, *El lenguaje de programación C*. Prentice-Hall), también practicarán conceptos matemáticos que son necesarios para las asignaturas de matemática discreta (Rosen, 2004, *Matemática Discreta y sus aplicaciones*. McGrawHill; García et al., 2002, *Matemática Discreta. Problemas y ejercicios resueltos*. Prentice Hall) y

álgebra lineal (Noble y Daniel, 1989, *Álgebra Lineal Aplicada*. Prentice Hall Hispanoamericana; Grossman, 2005, Álgebra Lineal. McGraw-Hill) citadas anteriormente.

Propuesta 1. Métodos de cifrado de textos

Motivación. Esta práctica la planteamos en la asignatura IP, de primer cuatrimestre. La asignatura IP presenta al alumno los recursos más básicos de programación en lenguaje C: tipos de datos básicos, estructuras de control, funciones, matrices, vectores, estructuras y cadenas. Para practicar estos conceptos proponemos la implementación del algoritmo extendido de Euclides y métodos de cifrado de cadenas considerando el alfabeto descrito en la **Tabla 1**.

Tabla 1. Alfabeto a utilizar para resolver la práctica.

* → 0	A → 1	B → 2	C → 3
D → 4	E → 5	F → 6	G → 7
H → 8	I → 9	J → 10	K → 11
L → 12	M → 13	N → 14	O → 15
P → 16	Q → 17	R → 18	S → 19
T → 20	U → 21	V → 22	W → 23
X → 24	Y → 25	Z → 26	(→ 27
) → 28	, → 29	? → 30	! → 31

Algoritmo de Euclides extendido. Implementar una función que calcule el algoritmo extendido de Euclides para calcular el máximo común divisor de un número que devuelva la siguiente información: el máximo común divisor de dos números *a* y *b*, el valor *u* y el valor *v*. El algoritmo de Euclides se describe en la **Tabla 2** y puede verse un ejemplo de aplicación en la **Tabla 3**.

Tabla 2. Algoritmo de Euclides extendido.

EuclidesExtendido(a, b)
P1 Leer a y b
P2 u' = 1, v = 1, u = 0, v' = 0, c = a, d = b
P3
q = cociente de dividir c entre d
r = resto de dividir c entre d
P4 si r = 0 entonces d = au + bv FIN
P5 si no, entonces
c = d, d = r
t = u', u' = u, u = t - qu,
t = v', v' = v, v = t - qv
P6 ir al Paso 3

Tabla 3. Ejemplo de uso del algoritmo de Euclides extendido.

u'	u	v'	v	c	d	q	r
1	0	1	1	1769	551	3	116
0	1	0	-3	551	116	4	87
1	-4	-3	13	116	87	1	29
4	5	13	16	87	29	3	0
mcd(1769, 551) = 29 = 5*1769 - 16*551							

Cifrado/descifrado Cesar. Implementar el cifrado César de un mensaje, es decir, dado un mensaje, y una clave privada, *k*, se transformará cada letra del mensaje en un número, *m*, (utilizando la **Tabla 1**). A dicho número se le suma una clave privada *k* y se hace módulo 32. La codificación de un mensaje se puede expresar como: $c = (m+k) \ mod32$. Después se vuelve a convertir *c* en una letra. Por ejemplo, si el mensaje, *m*, es ''MAR'' y la clave privada es 25, el mensaje cifrado, será "FZK".

- M =13 → (13+25) mod 32 = 6 → F
- A =1 → (1+25) mod 32 = 26 → Z
- R =18 → (18+25) mod 32 =11 → K

Un mensaje se descifra a partir de un mensaje cifrado y la clave privada, *k*, utilizada para cifrarlo, la función devolverá el mensaje sin cifrar. Para ello, se convierte cada carácter del mensaje cifrado en un número, *c*, utilizando la tabla anterior y se calcula $m = (c-k) \ mod32$. Después se transforma m en el carácter correspondiente. Por ejemplo, si desciframos el mensaje anterior ''FZK'' tenemos:

- F =6 → (6-25)=-19 + 32=13 → M
- 26 → (26-25) mod 32 = 1 → A
- 11 → (11-25) =-14+32=18 → R

Cifrado/descifrado Afín. Para cada carácter, hay que obtener su código, *m*, y el cifrado afín hace la operación: $c = (am + b) \ mod32$ siendo *a* y *b* las claves privadas. Posteriormente transforma el valor, *c*, obtenido por su carácter correspondiente. Por ejemplo, si queremos cifrar el mensaje ''MAR'' con el cifrado afín con las claves: *a* =7 y *b* =3, se haría:

- M =13 → (7*13+3) mod 32= 30 → ?
- A = 1 → (7*1+3) mod 32=10 → J
- R = 18 → (7*18+3) mod 32=1 → A

La función debe comprobar que la constante *a* es válida, es decir, que tiene inverso en Z_{32}[1], ya que si esto no fuera así el mensaje no se podría descifrar. La función devuelve -1 en este caso y 1 en caso contrario. Para comprobar esta condición se utilizará el algoritmo de Euclides extendido que ya ha sido implementado en un apartado anterior. El descifrado afín de un mensaje se implementará aplicando el método inverso al cifrado. Es decir, cada carácter del mensaje se convertirá en un número, *c*, y se calculará: $m = (a^{-1}(c-b)) \ mod32$[2]. Donde a^{-1} es el inverso de *a* en Z_{32}.

- ? =30 → 23*(30-3) mod 32= 13 → M
- J = 10 → 23*(10-3) mod 32 = 1 → A
- A =1 → 23*(1-3+32) mod 32= 18 → R

Cifrado/descifrado Vernam. El cifrado de Vernam funciona de la siguiente forma: un número *m* del texto sin cifrar se le hace la suma XOR bit a bit con el número correspondiente k de la clave obteniendo el número c del mensaje cifrado: $c = m \oplus k$[3]. Tener en cuenta que la longitud de la clave debe ser igual que la del

[1] Este valor existe si y sólo si $mcd(a, m) = 1$. Más aún, si al usar el algoritmo de Euclides extendido se obtiene $1 = au + mv$, entonces el valor *u* es el inverso modular de *a* módulo m

[2] Si a^{-1} fuese negativo, podríamos sumarle 32 las veces que hiciera falta hasta alcanzar un número positivo, denominado *representante canónico*. Análogamente, si a^{-1} fuera mayor que 32 podríamos restarle 32 las veces necesarias hasta que estuviese comprendido entre 0 y 32.

[3] Recordar que en C el operador XOR se corresponde con ^, así 13^24=13⊕24=21.

mensaje. Por ejemplo, supongamos que el mensaje que deseamos cifrar es: ''MAR'' y que la clave es ''XYZ''. Entonces:

• M = 13 = 01101; X = 24 = 11000

01101 ⊕ 11000 = 10101 = 21 → U

• A = 1; Y = 25

1 ⊕ 25=24 → X

• R = 18; Z = 26

18 ⊕ 26 = 8 → H

Para descifrar bastará con hacer: $m=c \oplus k$. Tomando el ejemplo anterior:

• U = 21; X = 24

21 ⊕ 24 = 13 → M

• X = 24; Y = 25

24 ⊕ 25 = 1 → A

• H = 8; Z = 26

8 ⊕ 26 = 18 → R

Cifrado/descifrado RSA. Cada carácter del mensaje se encripta utilizando la clave pública del receptor (n, e). Para ello, la función de cifrado es: c=me mod n. Por ejemplo, tomando n=55, e=7 y d=23, para cifrar el mensaje ''MAR'' tendríamos:

• M =13 → 13^7 mod 55 = 7 → G

• A = 1 → 1^7 mod 55 = 1 → A

• R = 18 → 18^7 mod 55 = 17 → Q

El mensaje cifrado se descifra utilizando el valor de la clave privada (*n*, *d*). Para ello, la función de descifrado es: $m = c^d \ mod \ n$. Para descifrar el ejemplo anterior tendríamos:

• G =7 → 7^{23} mod 55 = 13 → M

• A = 1 → 1^{23} mod 55 = 1 → A

• Q = 17 → 17^{23} mod 55 = 18 → R

NOTA: Para evitar que las potencias desborden la capacidad del tipo de dato, calcularemos $m^e \ mod \ n$ como un producto acumulado[4]:

Tabla 4. Producto acumulado.

```
DESDE (i=1; i<=e; i++)
{
  prod = (prod * a) mod n;
}
```

En la **Tabla 5** hay algunos ejemplos para realizar y probar el cifrado RSA.

[4] No olvidar el caso en que el exponente vale cero.

Tabla 5. Valores para probar el cifrado RSA.

p=5, q=11 → n=55, ϕ(n)=40
e=3 y d=27
e=7 y d=23
e=13 y d=37

Calcular parámetros del algoritmo RSA. Dados un par de primos *p* y *q* (tal que *p*q>32*), devuelve un conjunto de valores válidos para la clave pública y privada del receptor del mensaje (*d*, *e*, *n*). A continuación se describen los pasos que sigue el algoritmo:

• En privado, el receptor del mensaje, *R*, escoge dos números primos *p* y *q*[5], y los multiplica, obteniendo *n* = *pq* (*n*q* es mayor que el cardinal del alfabeto). Los valores de *p* y *q* no se hacen públicos.

• También en privado, el receptor obtiene el valor *(p-1)(q-1)* denominado función multiplicativa de Euler, $\phi(n)$.

• En privado, el receptor escoge un número *e* tal que $1 < e < \phi(n)$ de manera que sea primo relativo con $\phi(n)$[6], y le calcula su inverso módulo $\phi(n)$ que llamaremos $d=(e^{-1})_{\phi(n)}$. Para esto basta aplicar el algoritmo de Euclides extendido[7].

• El par de números *(d, n)* es la clave privada y el par de números *(e, n)* es la clave pública.

• Cuando el emisor, *E*, desea enviar un mensaje a *R*, lo hace utilizando la clave pública de *R* encriptando del siguiente modo: $c=m^e \ mod \ n$. Cosa que puede hacer, pues conoce los números *e* y *n* que *R* hizo públicos. Ahora envía el mensaje cifrado, *c*.

• El receptor recibe el mensaje cifrado y lo descifra aplicando $m=c^d \ mod \ n$. Es decir, utiliza su clave privada para descifrar.

En la **Fig. 1** tenemos un ejemplo de cifrado/descifrado con RSA. Los parámetros usados aquí son muy pequeños con respecto a los que maneja el algoritmo.

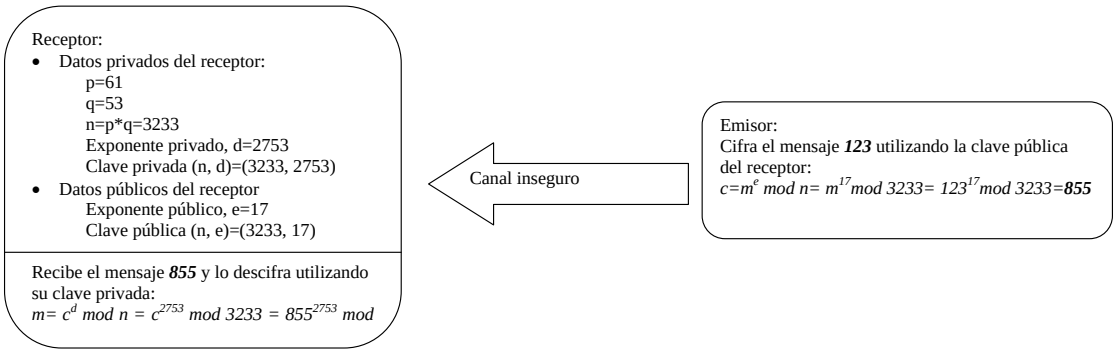

Figura 1. Ejemplo de cifrado RSA.

[5] Para que el algoritmo sea seguro, *p* y *q* han de tener valores muy grandes.
[6] $mcd\ (e, \phi(n)) = 1$
[7] Si al usar el algoritmo de Euclides extendido con $mcd(e, \phi(n))$ se obtiene $1 = eu + \phi(n)\ v$, entonces *u* se corresponde con *d*, el inverso modular de *e* módulo $\phi(n)$.

Propuesta 2. Métodos de cifrado de imágenes

Motivación. Esta práctica la planteamos en la asignatura MP, de segundo cuatrimestre. La asignatura MP presenta al alumno conceptos avanzados de programación en lenguaje C: ficheros, memoria dinámica, punteros, listas pilas y colas, ordenación y búsqueda, y herramientas de programación (bibliotecas, makefiles, etc.). Para practicar estos conceptos proponemos la implementación de algoritmos de cifrado de imágenes, de este modo deberán trabajar con ficheros de imágenes cargarlos en memoria y procesarlos. Las imágenes no tienen un tamaño fijo, por lo que habrá que recurrir a la utilización de memoria dinámica. Dado que la complejidad de los programas implementados comienza a aumentar, también se hará uso de herramientas de programación como *makefiles*, bibliotecas, argumentos en línea de órdenes, etc. Mediante el cifrado de una imagen se pretende que la imagen cifrada sea totalmente ilegible a la vista común. El proceso de descifrar así mismo debe garantizar la obtención de una imagen totalmente idéntica a la original. La esteganografía, por otro lado, es la rama de la criptología que trata sobre la ocultación de mensajes, para evitar que se perciba la existencia del mismo.

Cifrado de imágenes con el método Vernam. Para cifrar con este método, lo primero que deben hacer los dos usuarios será generar una clave secreta. En este caso que nos ocupa, la clave será una imagen, generada aleatoriamente, del mismo tamaño de la imagen a cifrar. Esta clave debe permanecer totalmente en secreto por ambos usuarios, ya que va a servir para cifrar y descifrar. El cifrado de Vernam utiliza la clave secreta para cifrar mediante la operación XOR. Cuando uno de ellos le quiera enviar al otro una imagen cifrada hará lo siguiente: cogerá la imagen original *A* y la clave *K*, obteniendo la imagen cifrada *B* mediante un XOR bit a bit entre cada par de píxeles. Por ejemplo, si el nivel de gris del primer elemento de la imagen original *A* es 129 y el primer elemento de la clave *K* es 231, entonces el primer elemento de la imagen cifrada *B* será 102 ya que: $(129 = 10000001) \oplus (231 = 11100111) \rightarrow 01100110 = 102$. Para descifrar sencillamente haremos la operación a la inversa: $(102 = 01100110 \oplus 231 = 11100111) \rightarrow 10000001 = 129$.

Cifrado matricial. El cifrado matricial[8] consiste en coger los niveles de gris de la matriz de dos en dos, empezando en la esquina superior izquierda de la matriz y moviéndonos de izquierda a derecha y de arriba a abajo: el primer bloque será $\{a_{11}, a_{12}\}$, el segundo bloque será $\{a_{13}, a_{14}\}$, y así sucesivamente. Cada bloque de dos niveles de gris de la imagen original se va a transformar en otros dos números mediante un producto matricial con una matriz secreta *K* de tamaño 2x2. Supongamos que el bloque que estamos procesando es: {125,137} y que la matriz secreta es $K = \begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix}$.

Hacemos el producto matricial: $\begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix}\begin{pmatrix} 125 \\ 137 \end{pmatrix} = \begin{pmatrix} 740 \\ 13073 \end{pmatrix} = \begin{pmatrix} 252 \\ 17 \end{pmatrix} (\mathrm{mod}\, 256)$.

Como vemos los números {740, 13073} exceden el valor 255 y por lo tanto no se corresponden con niveles de gris. Para conseguir que el resultado proporcione números entre 0 y 255, tomaremos módulo 256. Ahora el píxel que tenía nivel de gris 125 lo pondremos a 252 y el píxel de al lado que tenía nivel de gris 137 será puesto a 17. De esta manera, vamos transformando cada par de valores de gris consecutivos por otro par de valores de gris diferentes. De esta forma conseguimos codificar la imagen inicial.

[8] Este método se puede generalizar para claves de mayor tamaño realizando el cálculo de la inversa mediante el método de Gauss.

El receptor de la imagen cifrada, conoce cuál es la matriz secreta de cifrado *K*. Para deshacer el proceso y poder recuperar así la imagen original necesitará averiguar la matriz inversa de cifrado. Sabemos que:

la inversa de una matriz $K=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ es la matriz $K^{-1}=\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

Entonces para que dicha inversa exista es necesario que $|K|=ad-bc\neq 0$ pero en nuestro caso, como estamos trabajando módulo 256, es necesario además que el número $|K|$ sea primo relativo con el módulo para que le podamos calcular el inverso y poder hacer los cálculos anteriores. Para hacer esta comprobación podemos utilizar el algoritmo extendido de Euclides. Por ejemplo, si *K* fuera:

$$K=\begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix} \Rightarrow |K|=1029=5 \pmod{256}$$

Como 5 y 256 son primos relativos, el 5 tendrá inverso módulo 256 que calcularemos usando Euclides: $256(1)+5(-51)=1 \Rightarrow 5^{-1}=-51=205 \pmod{256}$. Por lo tanto la inversa será:

$$K^{-1}=\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}=\frac{1}{5}\begin{pmatrix} 79 & -35 \\ -18 & 21 \end{pmatrix}=205\begin{pmatrix} 79 & -35 \\ -18 & 21 \end{pmatrix}=$$

$$=\begin{pmatrix} 16195 & -7175 \\ -3690 & 4305 \end{pmatrix}=\begin{pmatrix} 67 & 249 \\ 150 & 209 \end{pmatrix}(\mathrm{mod}\,256)$$

¿Qué ocurre en caso de que el número de columnas de la matriz no sea par? En este caso, al cifrar se añadirá una columna ficticia, por ejemplo de ceros que se utilizará para cifrar (observar que siempre se generará el mismo valor para todos los píxeles de la columna ficticia). Estos píxeles ficticios no se guardarán en la imagen cifrada. Al descifrar añadiremos la columna ficticia cifrada (es un valor sencillo de calcular disponiendo de la clave de cifrado), nuevamente estos píxeles ficticios no se incluirán en el fichero de la imagen descifrada.

Mapa del gato de Arnold. Este método realiza una permutación de los píxeles de la imagen utilizando para ello una matriz secreta. Este algoritmo solo puede aplicarse a matrices cuadradas, por lo que *nfil=ncol*. Supongamos que *{x, y}* es la posición de un píxel (*x* es la fila entre *0* y *nfil-1*, *y* es la columna entre *0* y *ncol*-1) y que usamos una matriz secreta:

$K=\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$. Para cifrar, realizaremos la operación:

$$\begin{pmatrix} x' \\ y' \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}(\mathrm{mod}\,nfil)$$

Esto significa que el píxel que ocupa la posición *{x, y}* pasa a la posición *{x', y'}*. Por ejemplo, en una imagen de 124x124 el píxel (1,1) iría a la posición:

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix}(\mathrm{mod}124)$$

Para descifrar, utilizaremos la inversa, del mismo modo que se ha descrito en el cifrado matricial.

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}=\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}=\begin{pmatrix} 2 & 123 \\ 123 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 3 \end{pmatrix}(\mathrm{mod}124)$$

Esteganografía. La esteganografía se puede aplicar a imágenes mediante diferentes técnicas, una de ellas consiste en la modificación del bit de menor peso (*LSB: less significant bit*) de algunos pixeles de la imagen. Al tratarse del bit de menor peso de un píxel, éste se ve sometido a un cambio imperceptible de color.

Ejemplo de codificación. Sea la imagen original de la **Tabla 6** y el mensaje a codificar: ''so''.

Tabla 6. Imagen original y codificada.

	0	1	2	3	4	5	6
0	115	120	125	125	125	125	125
1	115	120	125	125	125	125	130
2	115	120	125	130	130	130	130
3	115	120	125	130	130	130	130
4	115	120	125	130	130	130	130
5	115	120	125	130	130	130	130

	0	1	2	3	4	5	6
0	114	121	125	125	124	124	125
1	115	120	125	125	124	125	131
2	115	121	124	130	131	130	131
3	114	121	124	130	130	131	130
4	115	120	125	130	130	130	131
5	114	121	124	131	130	130	130

• Añadir al mensaje la secuencia ''***'' de fin de mensaje: ''so***''

• Comprobar si el mensaje cabe dentro de la imagen. Cada carácter del mensaje está codificado con 1 byte (8 bits), habrá que tener en cuenta, además, la secuencia ''***'' de fin de mensaje

Tamaño mínimo de la imagen: 5 caracteres x 8 píxeles/carácter = 40 píxeles.

Tamaño de la imagen actual: 7x6=42 píxeles.

• Obtener el mensaje en binario binario: 01110011 01101111 00101010 00101010 00101010

• Esconder el mensaje en los bits menos significativos de la imagen original (ver tabla 6).

Ejemplo de decodificación. Sea la imagen con mensaje de la tabla 6.

• Obtener los bits menos significativos. Si el nivel de gris es par, el LSB será 0 y 1 si el nivel de gris es impar.

• Extraer el mensaje. Cada carácter está codificado con 8 bits. El mensaje comienza en el primer píxel de la imagen y termina con la secuencia ''***'' de fin de mensaje: 01110011 01101111 00101010 00101010 00101010.

• Decodificar el mensaje: ''so***''.

CONCLUSIONES

Este trabajo ha presentado dos interesantes propuestas de recursos didácticos para el aprendizaje de conceptos matemáticos a través de asignaturas relacionadas con la programación, dentro del título de Grado en Ingeniería informática. Se trata de dos aportaciones prácticas en las que tiene que

implementar algoritmos de distinta dificultad vistos en clase que pueden servir como recurso didáctico a otros profesores que imparten materias similares en otras titulaciones y universidades. Las principales ventajas de utilizar este tipo de recursos se detallan a continuación:

- El hecho de que el alumno tenga que implementar y probar los algoritmos le hará reflexionar sobre el funcionamiento de los mismos, lo que le llevará a una mejor comprensión.
- La implementación de algoritmos de cifrado permite al alumno practicar los conceptos vistos en ambas asignaturas con ejemplos de aplicación real, lo que le motiva al estudio de todas las materias implicadas.
- El hecho de que los conceptos practicados en programación sean vistos en las asignaturas de matemáticas simplifica el número de tareas que tiene que realizar el alumno.
- La práctica de conceptos como matrices y memoria dinámica a través de imágenes también supone un estímulo muy favorable para el alumno.

El siguiente paso sería una coordinación más fuerte entre asignaturas de modo que los trabajos realizados se calificaran en todas las materias implicadas.

AGRADECIMIENTOS

Los autores agradecen la financiación aportada por los proyectos P08-TIC-3720 y TIN2011-22408.