Los concursos de programación como herramienta didáctica

Agustín Cernuda del Río Daniel Gayo Avello

Departamento de Informática
Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo
(Universidad de Oviedo)
C/ Calvo Sotelo, S/N – 33007 Oviedo
Tf.: 985 10 50 94 - Correo-e: {guti, dani}@lsi.uniovi.es

Resumen

Algunos docentes aprecian un creciente desinterés del alumnado hacia las materias específicas de Informática, y en particular de la programación. A fin de promover y valorar la afición a estas materias, los concursos de programación entre estudiantes pueden ser una buena opción.

A la hora de organizar estas actividades puede ser conveniente plantearse los diversos pros y contras de las posibles modalidades, o conocer experiencias realizadas en otros centros.

1. Introducción

La evolución de la Informática como disciplina laboral en los últimos años ha transformado su imagen ante la sociedad. Hay que recordar que hace dos décadas casi no se conocía el ordenador como instrumento de uso doméstico. En la actualidad, aunque aún falte un gran trecho para que la situación de la ingeniería informática se normalice, es conocido al menos que hay una notable demanda de profesionales [11].

Esto ha transformado el perfil de los estudiantes. Hace años era muy frecuente que el alumno de Informática fuese aficionado a la materia, su familia no entendiese bien a qué se dedicaba y sus problemas académicos estuviesen centrados en asignaturas supuestamente *no informáticas*. Hoy en día, sin embargo, suele darse el caso de alumnos que se matriculan movidos por las salidas laborales, espoleados por su familia, y que de hecho completan el ciclo de estudios teniendo aún pendientes diversas asignaturas de programación. Ese parece ser el caso, al menos, en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo.

Ante este panorama parece oportuno promover la afición a la informática o el ejercicio lúdico de la misma, en especial en facetas relacionadas con la programación. Los concursos de programación son un recurso conocido para ello (en [14] puede verse incluso un enfoque para promocionar el estudio de la teoría y los métodos formales mediante concursos de programación), pero su organización puede plantear ciertas dificultades.

En este artículo se examinan desde diversas perspectivas los concursos de programación. A continuación se ofrecen diversas ideas sobre posibles modelos de concurso y posteriormente se describe una experiencia concreta llevada a cabo en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo (EUITIO).

2. Ventajas e inconvenientes de los concursos de programación

¿Qué interés puede tener desde un punto de vista didáctico la organización de un concurso de programación? Cabe perseguir los fines siguientes:

- La mera publicidad del concurso permite llamar la atención de todo el alumnado (participantes y no participantes) y recordarles la programación y la relación directa con los ordenadores.
- El que se organicen estos actos también extiende la idea de que la programación no sólo es un aspecto importante en su profesión, sino que también puede ser algo divertido. Efectivamente, existe una suerte de *vocación* informática.
- Centrándonos en los alumnos participantes, el concurso no deja de ser un ejercicio más, que les obliga a plantearse retos y buscar soluciones. Dependiendo del tipo de concurso se puede promover incluso una preparación específica en algún lenguaje, técnica o familia de algoritmos.

Sin embargo, no conviene perder de vista los inconvenientes asociados a estos concursos:

- Cierto tipo de concursos puede intimidar a los alumnos e inhibir la participación.
- Paradójicamente, los concursos que requieran una preparación específica o gran cantidad de trabajo pueden resultar demasiado absorbentes para los participantes motivados e influir negativamente en su rendimiento académico.
- Es posible que el resultado del concurso parezca arbitrario, lo que minará la motivación de futuros aspirantes.
- La organización de estos concursos puede resultar bastante compleja y requerir notables recursos del centro o del profesorado. Si la organización resulta deficiente, también se verá afectada la motivación de los alumnos.

A la hora de pensar en un concurso de programación es necesario tener en cuenta los siguientes aspectos:

- **Duración del concurso**. Puede tratarse de horas o de meses; recuérdese la posible influencia en la dedicación de los alumnos a las asignaturas.
- **Criterios de valoración**. Se puede tener en cuenta el diseño, el estilo de codificación, la eficiencia (en términos de memoria o de velocidad de ejecución), la calidad (referida al número de defectos), etc.
- **Trabajo individual o en equipo**. El concurso puede ser individual o admitir la participación de equipos de alumnos, cosa que plantea desafíos adicionales a los participantes [6].
- **Número de problemas a resolver**. Se puede plantear un solo problema o muchos, con un sistema de puntuación; también es posible que el concursante pueda elegir los problemas que resuelve.
- **Subjetividad del fallo del jurado**. Dependiendo de lo que se someta a valoración, el fallo puede ser casi arbitrario ("elegancia de la solución") o claramente objetivo ("número de fallos frente a un banco de pruebas determinado").
- **Dificultad para el jurado**. La revisión de los programas puede requerir mucho tiempo o consistir en una simple prueba de ejecución.
- **Dificultad de organización**. En clara conexión con la duración del concurso, infraestructura necesaria o tareas que el jurado deba realizar.
- **Perfil del jurado**. En ciertos tipos de concurso podría plantearse el que fueran los propios alumnos los que puntuasen, por votación, los programas; esto puede resultar de interés didáctico para los votantes, aunque quizás no tanto para los concursantes (que podrían perderse implementando características *deslumbrantes*).
- **Lenguajes de programación**. Se puede orientar el concurso a un lenguaje dado o dar libertad. También conviene pensar en diferencias entre lenguajes imperativos, funcionales, etc., en especial al plantear el enunciado.

3. Grandes concursos y *parties*

La idea de un concurso de programación no es, en modo alguno, nueva; muchas organizaciones promueven diversos tipos de concurso. Es bien conocido el concurso de programación de ACM para estudiantes [1] (en adelante ACM-ICPC), que celebra en 2003 la final de su 27ª edición. En él toman parte equipos de más de 1.000 universidades de todo el mundo, a través de diversas fases eliminatorias.

Evidentemente, resulta muy apetecible obtener cualquier buen resultado en un torneo de estas características. No obstante, no todos los alumnos se sienten preparados para participar a este nivel (independientemente de que lo estén o no). Además, tanto el entrenamiento como la participación pueden requerir un notable esfuerzo, incluyendo en algunos casos desplazamientos, estancias en otras ciudades o países o conocimiento de idiomas. Es una opción válida, pero no parece la más asequible para fomentar una alta participación. Como ejemplo, la Universidad de Valladolid otorga créditos de libre configuración por participar en diversas fases del ACM-ICPC.

Otro tipo de eventos de cierta envergadura son los encuentros denominados *parties*, en cuyo marco se suelen organizar concursos orientados a la programación de *demos*, videojuegos y similares, a la seguridad, o también a la programación en términos generales. En España existen varios encuentros de este tipo [3]. Algunos, de hecho, han sido organizados por instituciones docentes universitarias [7]. Al igual que los concursos como ACM-ICPC, son una opción atractiva para los estudiantes; en este caso,

sobre todo, porque estos eventos constituyen una gran oferta de ocio en relación con la informática. No obstante, adolecen de problemas similares a los de los grandes concursos de programación:

- Necesidad de desplazamientos si la *party* no se celebra cerca del lugar de residencia o estudio.
- Pueden no adaptarse bien al perfil académico del alumnado (especialmente por el grado de dificultad) o no cubrir la temática que se busca (el uso de ciertos lenguajes de programación o el estudio de cierto tipo de problemas).
- Pueden no ser adecuados en términos de fechas y duración o no tener interés de forma global para el alumno.
- Muchas instituciones académicas no pueden afrontar la organización de un evento propio de estas características.

Una respuesta es abordar la realización de concursos más específicos, adaptados a las necesidades (y posibilidades) formativas de cada centro.

4. Concursos de programación

A continuación se comentarán algunos posibles tipos de concurso, en relación con las ventajas y desventajas que ofrecen.

4.1. Proyecto de desarrollo integral

Al estilo de algunos concursos de programación de juegos, *demos* o similares, los concursantes desarrollan durante un tiempo (semanas, meses) un producto que luego entregan. Se puede valorar cualquier aspecto (análisis, diseño...), incluyendo el número de características implementadas, o incluso la estética.

Este tipo de concursos permite trabajo en equipo, y además el alumno revisa todo el ciclo de desarrollo. Como contrapartida, la participación puede resultar muy costosa para el alumno, exige también bastante trabajo del jurado, y además el fallo puede parecer arbitrario dependiendo de qué y cómo se valore.

4.2. Satisfacción de requisitos

Se trata de construir un programa que simplemente cumpla los requisitos planteados. Una posibilidad es plantear un problema de gran dificultad, en el que la mera capacidad de análisis y síntesis sea definitiva. Un inconveniente es que existe la posibilidad de que ningún concursante acierte a dar con la solución (además, la programación no es decisiva en el resultado).

Otra opción es plantear problemas de dificultad normal (o incluso sencillos) y puntuar según cuántos se resuelvan en un tiempo dado. Este enfoque es el que se utiliza en el ACM-ICPC.

Los concursos basados en la mera satisfacción de requisitos tienen la ventaja de que se puede verificar de manera automatizada la solución; basta establecer restricciones claras sobre las entradas y las salidas. Este sistema se utiliza en el sitio web que la Universidad de Valladolid mantiene en relación con el ACM-ICPC y que cuenta con un *juez en-línea* [2].

Como mezcla de los dos enfoques, se puede plantear un problema cuya solución básica sea relativamente asequible pero cuyas ramificaciones sean de gran dificultad. Así ha hecho en el ICFP Programming Contest [8][13].

4.3. Eficiencia

El objetivo del concurso es resolver un problema mediante un programa que sea lo más eficiente posible; por ejemplo, el ganador es el programa que (resolviendo el problema) requiere menos tiempo de ejecución. Es una buena forma de recompensar un diseño cuidadoso de los algoritmos, y una de las pocas medibles. (Otras medidas de eficiencia, como el uso de recursos, administración de memoria, etc. pueden ser mucho más problemáticas).

La dificultad estriba en que en ocasiones es difícil hacer comparaciones de eficiencia si la diferencia no es grande, ya que hay muchos factores del entorno que pueden influir: los lenguajes utilizados, los equipos, etc.

4.4. Diseño

En este caso se valoraría el diseño, la elegancia de la solución, etc. Didácticamente resulta muy conveniente en el sentido de que se premian algunas de las características más significativas del perfil académico del alumno. Sin embargo, puede resultar difícil establecer criterios claros de valoración, y en consecuencia el fallo puede percibirse como subjetivo o ambiguo. Además,

requiere un notable trabajo del jurado, incluyendo la resolución de posibles diferencias de opinión.

4.5. Velocidad de programación

Se trata de una variante del concurso de satisfacción de requisitos; se premia exclusivamente el tiempo que el alumno tarda en construir un programa que resuelva un problema dado (asequible). Las ventajas de este enfoque son que resulta muy fácil de organizar frente a otros (la infraestructura y organización son muy similares a las de un examen práctico). Además, el resultado es objetivo y el criterio de valoración muy claro. Sin embargo, es evidente que la velocidad de programación no es precisamente el factor más importante en el perfil de un alumno, y ni siquiera cabe fomentarlo demasiado desde un punto de vista académico.

4.6. Lucha entre programas

Otra versión clásica de los concursos de programación es organizar enfrentamientos entre programas que puedan competir de alguna manera. Existen juegos de programación basados en esta idea; CROBOTS [4] fue lanzado en 1985, y hay otras versiones más recientes para otros lenguajes [10]. En estos juegos los programas "luchan" en la memoria del ordenador, pero como evolución de esto se han creado versiones parecidas con una representación visual tridimensional, como en el caso de los D-Robots, cuyo lenguaje es similar a Pascal [5] o más recientemente Terrarium para .NET [15].

Estos juegos son interesantes porque se fomenta el trabajo en equipo y se pueden crear proyectos relativamente creativos manteniendo sin embargo un criterio de valoración uniforme y objetivo (la victoria en las partidas). Además, la infraestructura suele ser asequible[1], y en el caso de que haya una representación gráfica los estudiantes que no concursan pueden involucrarse y seguir las partidas como cualquier competición deportiva. Los problemas vienen porque su preparación requiere del estudiante cierto esfuerzo y posiblemente un aprendizaje específico.

[1] La versión actual de D-Robots exige tarjetas de vídeo con capacidad de aceleración 3D.

4.7. Relevos

Se puede organizar una competición basada en alguna de las demás modalidades (por ejemplo, velocidad de programación) pero por turnos; un concursante empieza a resolver el problema durante un tiempo límite, y cuando el tiempo acaba le sustituye el otro miembro del equipo. Puede haber más turnos hasta llegar a la solución. Se puede permitir una breve charla en el seno del equipo al principio o, por el contrario, impedir toda comunicación fuera del código y / o la documentación.

Este tipo de concurso tiene importantes ventajas; frente a la velocidad de programación pura, que no resulta del todo apropiada como objetivo didáctico, se premia la claridad, el buen diseño, la documentación, el desarrollo de cara al mantenimiento... Además, el criterio de valoración puede ser muy claro (siguiendo con el ejemplo, la velocidad de programación), pero a la vez el buen diseño, programación y documentación son necesarios por motivos prácticos. Los inconvenientes son que hay que elegir los problemas cuidadosamente (para que no los resuelva el primer relevo y a la vez sean abordables por turnos). También son más difíciles de organizar y de desarrollo previsiblemente más largo.

4.8. Duelo

Este tipo de concurso se centra en la resolución de problemas. Los concursantes se enfrentan a pares, como en un torneo individual de tenis o ajedrez; uno plantea al otro un problema que tienen que resolver ambos en cierto tiempo, y luego el segundo plantea su propio problema. A cada jugador se le valora el resultado (del tiempo de programación, por ejemplo) de ambos programas.

Es interesante porque obliga al concursante a pensar problemas variados que, pudiendo resolver él mismo en un tiempo limitado, tengan alguna dificultad para el oponente. Además, cada alumno se enfrentará a diversos problemas. Como inconveniente, es posible traer problemas rebuscados que el oponente sea incapaz de resolver aunque la programación sea fácil cuando uno conoce la solución.

```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n')()#}w'){){nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
  :0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Figura 1. Ejemplo de C *enrevesado (obfuscated)*.

4.9. *Obfuscated*

Tienen gran tradición las secciones de *obfuscated code* o *código enrevesado* en las revistas de programación. El código enrevesado es código deliberadamente oscuro, generalmente recurriendo a sutilezas del lenguaje de programación, que resulta difícil de leer e interpretar. A veces parece imposible que compile sin errores; otras parece correcto pero su comportamiento no es el esperado. Algunos lenguajes son especialmente *apropiados* para esto, y existen diversos concursos; C tiene una larga tradición [9]. En la Figura 1 puede verse un famoso ejemplo (un programa que imprime por pantalla un poema).

Es un ejercicio interesante para conocer a fondo un lenguaje de programación, y puede ser apropiado para la modalidad de *duelo*. Pero también es cierto que esta habilidad, por sí misma, no resulta muy significativa de cara a la formación de un perfil profesional.

5. El I Concurso de Velocidad de Programación en la EUITIO

5.1. Antecedentes y convocatoria

Con el fin de fomentar el interés por la programación y, sobre todo, servir de recordatorio sobre la posibilidad de entender la programación como una actividad lúdica, la dirección de la EUITIO decidió organizar un concurso de programación. Este concurso se enmarcaba en las actividades de conmemoración del XX Aniversario de la Escuela.

De cara a la organización había varias dificultades; en primer lugar, los recursos disponibles eran muy limitados. La actividad diaria de la EUITIO y otros actos del XX Aniversario ya no dejaban mucho margen para trabajar en la organización del concurso.

Pensando en evitar problemas de interpretación y en las dificultades de encontrar miembros para el jurado, se decidió organizar un concurso de velocidad puro: un solo problema, que se resolvería en el momento y en el menor tiempo posible. De este modo, las dificultades de organización serían mínimas.

No teníamos mucha confianza en una respuesta entusiasta por una serie de razones, pero en la medida de lo posible intentamos presentar cada inconveniente como una ventaja o desafío:

- **El concurso se parece a un examen práctico contra el reloj, experiencia no muy agradable.** En los anuncios y bases del concurso se desafiaba a los participantes a "soportar la presión" (Fig. 2).
- **No se podía valorar la elegancia o el diseño, porque eso plantearía un mayor trabajo de organización.** Se incidió en la oportunidad de programar *libremente*, sin normas (¡por una vez!) sobre documentación, análisis o diseño.
- **Los medios de hardware, software y personal de soporte eran limitados**. Se advertía de que no cabía reclamar al respecto ("este es un concurso para programadores, no para llorones").
- **No había posibilidad de ofrecer premios en metálico**. Se ofreció como premio "sólo la gloria"; recibir un documento acreditativo en la entrega de diplomas y ver el propio nombre en la página web de la Escuela.

Figura 2. Cartel del concurso.

Se redactaron unas bases informales para el concurso, que pueden consultarse en [12].

5.2. Inscripción y participación

De este modo se abrió el plazo de inscripción, y se inscribieron 34 personas. Fue una agradable sorpresa, ya que las previsiones eran simplemente alcanzar un número de participantes que permitiese celebrar dignamente el concurso (al menos 5). El día del concurso acudieron 18 participantes. Las salas de ordenadores de la EUITIO albergan a unos 25 alumnos como máximo, por lo que preveíamos la utilización de dos salas y un esfuerzo extra para sincronizar en ambas el desarrollo del concurso; finalmente, no fue necesario.

Un dato curioso es la participación de las mujeres; sólo 2 se inscribieron (y acudieron al concurso), lo que representa un 6% de los inscritos y un 11% de los participantes reales. Esta proporción no representa en modo alguno la proporción de sexos en el alumnado de la EUITIO (que puede verse en la Figura 3). En la pasada Campus Party 2002 [3] el porcentaje de mujeres participantes fue de un 9%, frente a un 91% de hombres.

	♂		♀	
Plan antiguo				
I.T.I.Sist.	350	75,75 %	112	24,25 %
I.T.I.Gest.	204	43,68 %	263	56,32 %
Plan nuevo				
I.T.I.Sist.	179	81,73%	40	18,27%
I.T.I.Gest.	171	70,95%	70	29,0º5%

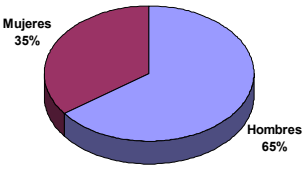

Figura 3. Distribución del alumnado por sexos en la EUITIO.

Sería interesante saber si los alumnos (en general) que no respondieron a la convocatoria no lo hicieron porque la materia (o la informática en sí misma) no les interesa, porque no creen que puedan ganar el concurso, porque la "gloria" y la competitividad no les parecen atractivas, porque les resulta demasiado estresante programar contra el reloj... y en qué medida estos motivos tienen más peso en uno u otro sexo.

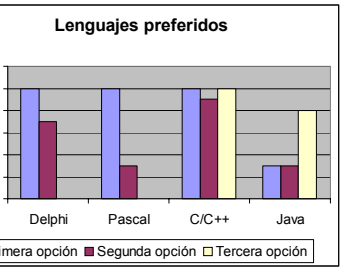

Figura 4. Distribución de lenguajes elegidos.

En el mensaje de inscripción cada alumno debía elegir sus lenguajes preferidos, por orden, a fin de tomar decisiones sobre el software a instalar. La distribución de las preferencias puede verse en la Figura 4 (incluye sólo los lenguajes más solicitados; por ejemplo, dos alumnos anotaron Visual Basic como segunda opción).

5.3. Preparación del problema

El programa a construir tenía que leer un fichero de texto de entrada y ordenar (línea por línea) sus palabras en un fichero de salida, con la salvedad de que debía ser capaz de manejar (teóricamente) un fichero de entrada de tamaño arbitrariamente grande. Es decir, no se considerarían válidas las soluciones que cargasen en memoria toda la información. Básicamente, se trataba de un simple algoritmo de ordenación secuencial de ficheros.

Por razones de disponibilidad de salas y personal de administración y servicios, el concurso podía durar sólo 3 horas en total. Con anterioridad a la celebración del concurso un miembro de la organización hizo la prueba, creando una versión del programa en C, relativamente compleja (porque incluía alguna pequeña mejora) que constituía una solución válida; se trataba de un programa de unas 200 líneas que estuvo terminado en algo menos de 50 minutos. Contando con una probable diferencia de velocidad a favor del profesor, parecía razonable esperar que algún alumno diera con la solución en el plazo de 3 horas.

5.4. Mecánica del concurso

Una vez vistas las preferencias respecto a lenguajes, se vio que el concurso podría celebrarse con la infraestructura disponible en los laboratorios. En concreto, unos días antes se informó a los alumnos de que estarían disponibles los compiladores de la Figura 5, a fin de que pudieran familiarizarse con el entorno o la línea de órdenes antes del concurso si lo necesitaban. (Suele haber otros compiladores, pero sólo se garantizaba la disponibilidad de estos).

Tras recibir a los participantes en una sala de teoría, se les expuso la mecánica del concurso y se les explicó también el enunciado del problema. Una vez atendidas las preguntas, se pasó al laboratorio donde tendría lugar la competición.

Lenguaje	Compiladores
C/C++	Borland C++ 5.02 Microsoft Visual Studio 6.0
Pascal/Delphi	Borland Delphi 5
Java	Borland Personal JBuilder 7

Figura 5. Compiladores disponibles en el concurso

Los alumnos podían utilizar papel y lápiz, pero no podían disponer de ningún material informático propio; el primer paso, además, fue desconectar fisicamente las conexiones de red de todos los equipos. Acto seguido se les facilitó una copia impresa del enunciado, así como un disquete con un par de pequeños programas; uno servía para generar un caso de prueba, y el otro para verificar la solución que se había dado a dicho caso de prueba.

Una vez repartido el material, se puso en marcha el cronómetro. Cada alumno que creyese tener una solución avisaba a los organizadores; en ese momento, se paraba el cronómetro y todos los participantes debían interrumpir su actividad y quedarse de pie junto a su puesto mientras que se comprobaba la solución. Si esta no era buena, el cronómetro se ponía de nuevo en marcha y los concursantes podían continuar.

Estas interrupciones formaban parte del juego; de hecho, antes del concurso hubo algunas sugerencias acerca de incorporar estrategias de distracción (o incluso cortes de luz, según las versiones más radicales), que en principio se desestimaron hasta ver cómo se desarrollaba esta primera experiencia. Hubo tres "falsas alarmas". Dos de ellas se debían a programas que funcionaban, pero que de alguna manera cargaban toda la información en memoria. La tercera fue debida a un programa que, durante la verificación, falló (por un aserto).

Finalmente, este programa, una vez corregido, se dio por bueno a los 53 minutos de empezar el concurso. Se trata de un algoritmo muy ineficiente pero muy simple; menos de 60 líneas de código en C [12]. De acuerdo con las normas del concurso, este programa resultó ganador.

Para nuestra sorpresa, muchos participantes estaban dispuestos a continuar, por lo que se decidió seguir adelante hasta encontrar un segundo programa válido (unos 6 minutos después y en lenguaje Delphi) y un tercero (un par de minutos más tarde, también en Delphi). En este punto se dio por terminado el concurso.

5.5. Comentarios al resultado

A pesar de las muchas restricciones de material, personal y tiempo de organización, el concurso resultó ser un rotundo éxito. Hemos recibido diversos mensajes de felicitación de los alumnos,

pero además al término del concurso varios alumnos comentaron que les gustaría participar en otros similares, proponiendo diversas modalidades. A pesar de lo "árido" que puede resultar en algunos aspectos un concurso de velocidad de este tipo, los participantes lo vivieron realmente como una actividad lúdica. Ha tenido, además, una cierta repercusión entre el alumnado, y el alumno ganador recibió su premio en el acto académico de la fiesta de la Escuela, junto con los diplomas de la 18ª promoción de la EUITIO.

El ganador es un buen programador, pero por las preguntas que realizó sobre el enunciado sabemos que cuando entró en el laboratorio no sabía cómo solucionar el problema. Además, otros alumnos con gran afición por la programación y un excelente nivel no quedaron entre los tres primeros. Esto no tiene nada de sorprendente; este enfoque de la programación es, en gran medida, "deportivo", y como en cualquier encuentro deportivo aislado influyen en el resultado muy diversos factores, muchos de ellos circunstanciales.

Ciertamente, el ejercicio de programación planteado no es de gran nivel; además, resulta muy difícil medir el efecto real del concurso. Pero ha habido una respuesta entusiasta de muchos alumnos y el resultado es excelente dadas las limitaciones del planteamiento.

6. Conclusiones

A pesar de que el perfil típico del estudiante de Informática no necesariamente responde al de un aficionado a la programación, un concurso de programación puede despertar el interés de suficientes personas, sirviendo como un acicate en su formación para los participantes y como un "recordatorio" para el resto de los alumnos.

Aunque existen diversos concursos y ciclos de actividades con cierta tradición, es perfectamente posible organizar concursos propios, con recursos muy limitados y más adecuados al perfil e intereses de los estudiantes. Muchas de las limitaciones organizativas pueden transformarse en desafíos que incluso hagan más atractivo el concurso.

La respuesta a nuestra primera experiencia de este género ha sido sorprendentemente positiva, a pesar de que otro tipo de competiciones puede parecer más vistoso a priori. Los alumnos se han mostrado entusiastas ante la organización de este evento y esperan más en el futuro.

En la EUITIO tenemos intención de repetir este tipo de actividades, y tenemos especial interés en alguna modalidad de tipo D-Robots y en un concurso por relevos.

Referencias

[1] The ACM International Collegiate Programming Contest. http://www.acm.org/contest

[2] ACM International Collegiate Programming Contest. Universidad de Valladolid. http://acm.uva.es/problemset

[3] Campus Party. http://www.campus-party.org

[4] CROBOTS. http://www.nyx.net/~tpoindex/crob.html

[5] D-Robots. http://www.plasmacode.com

[6] Ernst, Fabian; Moelands, Jeroen; Pieterse, Seppo. *Teamwork in Programming Contests: 3 * 1 = 4*. ACM Crossroads Student Magazine, nº 3.2 (invierno 1996).

[7] FIBERparty. http://www.fiberparty.net

[8] International Conference on Functional Programming (ICFP). http://www.math.luc.edu/icfp

[9] International Obfuscated C Code Contest. http://www.es.ioccc.org/main.html

[10] JRobots. http://www.mobydisk.com/java/jrobots

[11] Martínez, Gloria; Fabregat, Germán. *Perfil profesional y académico de la informática en España*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática, Cáceres, julio de 2002.

[12] Página web sobre el I Concurso de Velocidad de Programación de la EUITIO. http://www.euitio.uniovi.es/actividades/concursovelprog-i.php3

[13] Ramsey, Norman; Scott, Kevin. *The 1999 {ICFP} Programming Contest*. SIGPLAN Notices, Vol. 35, nº 3 (Marzo de 2000), págs. 73-83.

[14] Shilov, Nikolay V.; Yi, Kwangkeun. *Engaging Students with Theory through ACM Collegiate Programming Contests*. ACM Communications, Vol. 45, nº 9 (Septiembre de 2002), págs. 98-101.

[15] Terrarium. http://www.gotdotnet.com/terrarium