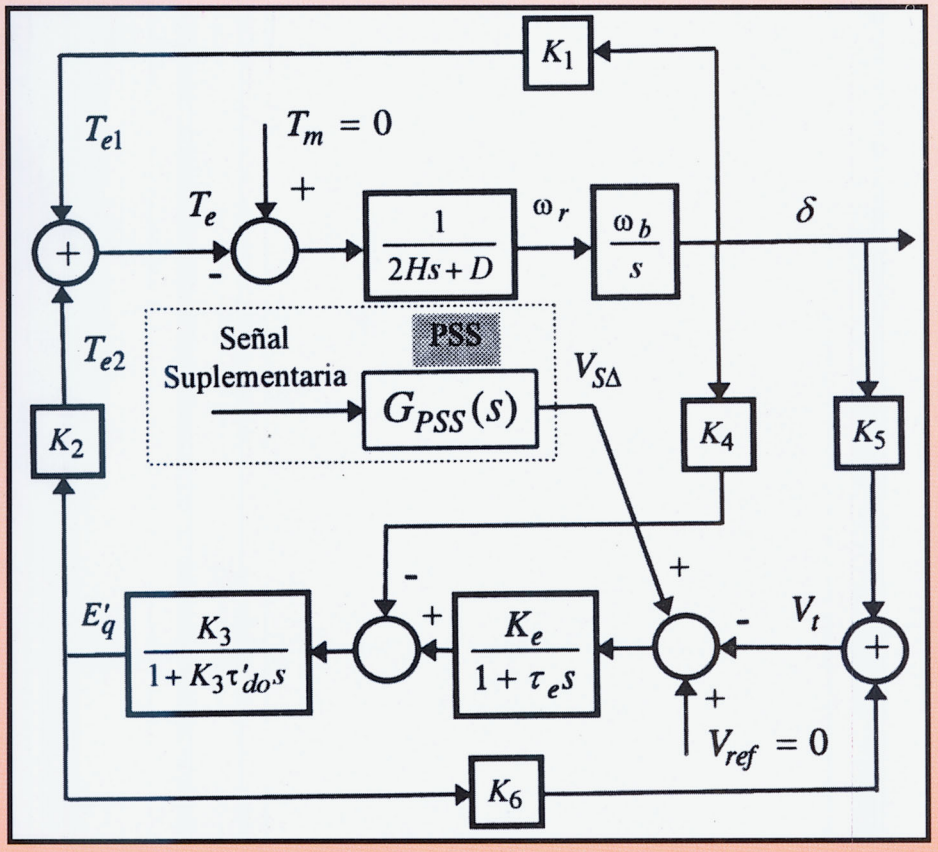
ISSN 1409-2441

Ingeniería

Revista de la Universidad de Costa Rica
Enero/Junio 1997 VOLUMEN 7 N° 1



ISSN 1409-2441

INGENIERIA

Revista Semestral de la Universidad de Costa Rica
Volumen 7, Enero/Junio 1997 Número 1

CORRESPONDENCIA Y SUSCRIPCIONES

Editorial de la Universidad de Costa Rica
Apartado Postal 75
2060 Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

CANJES

Universidad de Costa Rica
Sistema de Bibliotecas, Documentación e Información
Unidad de Selección y Aquisiciones-CANJE
Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

Suscripción anual:
Costa Rica: ¢ 1 000,00
Otros países: US $ 25,00

Número suelto:
Costa Rica: ¢ 750,00
Otros países: $ 15,00

Edición aprobada por la Comisión Editorial de la Universidad de Costa Rica

Ciudad Universitaria Rodrigo Facio
San José, Costa Rica.

Revisión Filológica: *Lorena Rodríguez*

Diseño Gráfico, Diagramación y Control de Calidad:
Sergio Aguilar Mora

Impreso en la Oficina de Publicaciones
de la Universidad de Costa Rica

Revista
620.005
I-46i Ingeniería / Universidad de Costa Rica. ——
Vol. I, no. 1 (ene./jun. 1991)— . —— San José, C. R. : Editorial de la Universidad de Costa Rica, 1991— (Oficina de Publicaciones de la Universidad de Costa Rica)
v. : il

ISSN 1409-2441

Semestral.

1. Ingeniería - Publicaciones periódicas.

CCC/BUCR—250

Análisis y Comentarios

Ingeniería Vol. 7 Nº 1 - 1997: 111-123; San José, Costa Rica

PROPUESTA PARA MEJORAR EL CURSO PRINCIPIOS DE INFORMÁTICA

Adolfo Di Mare Hering [1]

RESUMEN

Presento mis ideas sobre cómo debe impartirse el curso CI-0202 Principios de Informática, que es el curso introductorio de Informática para Ingeniería en la Universidad de Costa Rica. Concluyo que se puede lograr una significativa mejora si las lecciones se imparten usando la ayuda audiovisual de un equipo DataShow en lugar de dar las clases en el laboratorio de computadores de la Facultad.

SUMMARY

I share my ideas on how to teach CI-0202 Principios de Informática, the first Informatic course for Engineering at Universidad de Costa Rica. My conclusion is that a significant improvement can be achieved by using a DataShow instead of teaching at the computer lab.

Después de más de diez años de impartir lecciones sólo a estudiantes de la Escuela de Ciencias de la Computación e Informática [ECCI], tuve la oportunidad de impartir el curso CI-0202 Principios de Informática, a dos grupos de estudiantes de la Facultad de Ingeniería, en la Universidad de Costa Rica [UCR]. Este cambio me ha ayudado mucho a entender mejor qué significa docencia en computación y en la Universidad. Por eso escribo estas líneas, con la esperanza de que contribuyan al mejor desarrollo de nuestros estudiantes.

Espero que mis ideas contribuyan a mejorar la calidad de nuestros cursos, o al menos a ayudar a que el material que los profesores entregamos a nuestros estudiantes tenga más relevancia a su futuro quehacer profesional. De todas maneras, en toda academia es importante propiciar estas discusiones curriculares.

1. PROGRAMA ACTUAL DEL CURSO

En el primer semestre 1997 el programa del curso CI-0202 Principios de Informática fue el siguiente:

OBJETIVOS GENERALES

1. Que el estudiante comprenda la utilidad y alcances de una herramienta de programación en forma tanto general como específica dentro de su quehacer profesional futuro.
2. Que el estudiante pueda enfrentarse ante un problema, diseñar su solución algorítmica -si la tiene- y programarla utilizando un lenguaje de alto nivel.

1 Profesor Escuela de Ciencias de la Computación e Informática. Facultad de Ingeniería.

OBJETIVOS ESPECIFICOS

1. Dominio de los comandos básicos del sistema operativo DOS (o el que utilice el equipo con el que se trabaje).

2. Manejo del Lenguaje de Programación Pascal para el desarrollo de aplicaciones de mediana complejidad.

3. Dominio de la Hoja de Cálculo Excel.

CONTENIDOS

1. Introducción a la computación y al DOS.

   - Componentes básicos simples de un computador tanto de "hardware" como de "software" (incluyendo la descripción simple de traductores, en particular de los compiladores). Funcionamiento de un computador.
   - El sistema operativo DOS.

2. Algoritmos.

   - Concepto de algoritmo, relación entre "algoritmización" y computabilidad.
   - Ejemplos.

3. El Lenguaje Pascal.

   3.1 Tratamiento secuencial en Pascal.

   - Ejemplos de programas.
   - Estrucura secuencial simple.
   - Alfabeto, identificadores, palabras clave, constantes, tipos (*integer, real, char, string, boolean*).
   - Encabezado, declaración de constantes, declaración de variables, secuencia de instrucciones.
   - Asignación de variables numéricas.
   - Expresiones aritméticas, operadores aritméticos y relacionales, funciones de biblioteca (*Read, ReadLn, Write, WriteLn, ClrScr, Delay*).

   3.2 Tratamiento condicional e iterativo.

   - *IF...THEN...ELSE...*
   - IF's anidados.
   - Expresiones booleanas.
   - *FOR, WHILE, REPEAT, UNTIL*

   3.3 Registros y archivos secuenciales.

   - Registros.
   - Archivos, Read, Write, ReWrite, Assign.

   3.4 Arreglos.

   - Arreglos unidimensionales.
   - Arreglos multidimensionales (matrices).

   3.5 Procedimientos.

   - Ejemplos.
   - Parámetros por referencia y por valor.
   - Variables locales y globales.

   3.6 Funciones.

   - Declaración.
   - Parámetros.
   - Variables locales y globales.

4. La Hoja Electrónica Excel.

   - Concepto de hoja electrónica.
   - Modos de operación.
   - Celdas, bloques, filas, columnas.
   - Introducción de datos, tipos de datos y formatos.
   - Edición de celdas (copiar, cortar, trasladar, llenar, limpiar, borrar).
   - Creación, almacenamiento y recuperación de una hoja de trabajo.
   - Impresión de una hoja, o de un área de ella.
   - Referencia absoluta y referencia relativa.
   - Edición y fórmulas.
   - Fijación de títulos, ordenamiento de datos.

- Gráficos.
- Macros.

5. Concepto para la utilización de Internet.

- Qué es Internet.
- Correo electrónico.
- Utilidad en investigación.
- Uso de los principales hojeadores (browsers).
- Uso de motores de búsqueda Internet.

EVALUACION

Tareas cortas, quices, concepto del profesor	10%
Tareas programas	15%
Examen parcial #1 (temas 1 - 3.2)	30%
Examen parcial #2 (temas 1 - 3.6)	30%
Examen parcial #3 (tema 4)	15%

Este programa ha evolucionado con la tecnología, pues, en sus primeras versiones, el curso de Principios de Informática era un curso del lenguaje Fortran. Posteriormente fue adoptado el lenguaje Pascal porque tiene muchas cualidades como lenguaje de programación [Set-92]. A continuación las menciono, y aprovecho la oportunidad para justificarlas.

- Pascal incluye las construcciones de programación estructurada que le faltan a Fortran (*IF-THEN-ELSE, WHILE, REPEAT*).

  La tecnología de programación de computadores ha avanzado relativamente poco en comparación con la de construcción de computadores. Pese a esto, a principios de los años 80 se llegó al consenso de que cualquier lenguaje de programación debe permitir programas estructurados; incluso Fortran en sus versiones recientes incluye este tipo de construcciones.

- Todo compilador Pascal verifica que las interfaces entre módulos sean correctas mediante la técnica de la verificación de tipos.

  Esta técnica facilita mucho la construcción de programas, pues delega en el compilador la tarea de encontrar muchos errores de programación que de otra forma debería hallar el programador.

- Pascal es un lenguaje más fácil de aprender, lo que facilita mucho la enseñanza de la programación.

  Todo el proceso de enseñanza - aprendizaje debe llevar a alcanzar los objetivos cognoscitivos. Por eso es mejor aprovechar aquellas opciones que requieran un esfuerzo menor; de nada vale lograr lo mismo a un costo más alto, que es la alternativa si no se usa Pascal como primer lenguaje de programación.

- No existen compiladores eficientes de Fortran, mientras que el Turbo Pascal corre, prácticamente, en cualquier computador.

  Aunque las versiones más recientes de Pascal requieren ambientes de trabajo como Windows 95, el compilador versión 5.5 corre en máquinas con un megabyte de memoria, y una unidad de diskette [BI-88].

Como la computación es una tecnología joven, no tiene la tradición que otras profesiones han acumulado. Por eso, los primeros programas para cursos de cómputo estaban dirigidos a aprender lenguajes de programación, pues para muchos "saber computación" era "saber programar" y, para simplificar las cosas, la enseñanza se concentró en un lenguaje específico. Así, en Principios de Informática primero se enseñó Fortran y luego Pascal. También se probó enseñar Dbase III o Fox, y luego se agregó la hoja de cálculo Excel.

No es resorte de la ECCI decidir si aprender a programar es relevante para la formación profesional de estudiantes de otras carreras, en particular de las ingenierías, como tampoco puede la Escuela de Matemática definir si la capacidad de razonamiento para la carrera de Medicina se logra mejor con materias de Cálculo que de Filosofía. Esta decisión la deben tomar los docentes de cada carrera.

Sin embargo, muchos profesores en la ECCI coincidimos en que la enseñanza de programas específicos como Word y Excel no debería ser materia de cursos de nivel universitario, pues son menudencias que pueden ser impartidas, a un costo muy bajo pero con gran eficacia, por otras instituciones educativas como escuelas, colegios, o institutos de enseñanza especializados en ese ramo. De hecho, muchos estudiantes ya aprenden a usar esos paquetes de computación solos en casa o en su colegio de secundaria. Este argumento lleva a la siguiente conclusión:

No debería enseñarse Word o Excel en Principios de Informática

2. EL CONTEXTO ACTUAL

Con un gran esfuerzo financiero la Facultad de Ingeniería ha logrado montar un laboratorio de cómputo que, en primera instancia, es utilizado por los estudiantes del curso Principios de Informática. Además de contar con un administrador del sistema, la Facultad ha comisionado a varios asistentes quienes ayudan a los usuarios del laboratorio y cuidan que las máquinas no sufran daños por terceros mal intencionados. Este esfuerzo es encomiable.

Los profesores de Principios de Informática imparten dos lecciones de 100 minutos cada semana, y una de las sesiones se desarrolla en el laboratorio, en donde los estudiantes se sientan en parejas ante las computadoras. Como hay 14 máquinas disponibles (aunque con frecuencia al menos una de ellas no funciona), la matrícula del curso está restringida a poco más de 25 cupos. En la práctica el director de la ECCI permite que se matriculen más estudiantes, pues siempre ocurre que unos pocos abandonan las aulas por razones personales.

El cupo se restringe porque, principalmente, un profesor no puede atender en forma adecuada a 30 estudiantes, y menos a 40 ó 50. En resumen, en estos momentos, en el curso de Principios de Informática se invierten una gran cantidad de recursos docentes, de laboratorio y de asistentes; pero, pese a esta inversión, persiste semestre a semestre una demanda insatisfecha de más del 50% de los cupos disponibles. Por eso conviene cambiar la cosas, para poder ofrecer un mejor servicio docente y reducir el costo de prestarlo.

3. SENTIR DE LOS ESTUDIANTES

Muchos estudiantes que llegan al curso ya conocen varios paquetes de computación. Algunos tienen computadora en su casa y otros han recibido cursos de Excel o Word en el colegio. Otros, simplemente, han practicado empíricamente en las computadoras de la Universidad o en casa de amigos. En general todos tienen alguna idea de qué pueden hacer con las computadoras.

A los estudiantes les gustaría que, en lugar de enseñarles Pascal, les permitieran manejar muchos programas o paquetes de computación. Como muchos vienen de otras facultades, principalmente de la Facultad de Ciencias Económicas, muestran poco interés en la programación y toman el curso como un castigo necesario para graduarse. Por eso los estudiantes sienten que el curso debería empezar por lo que les es más útil: la hoja de cálculo Excel.

Los profesores sabemos que toma un par de lecciones aprender Excel, pero que para programar es necesario madurar algunos conceptos; por eso le damos mucho más énfasis a la programación Pascal. De hecho, para conseguir profesores, se busca a quien sepa programar, y nunca se pregunta siquiera si el profesor conoce Excel.

En la mente de los estudiantes, el curso Principios de Informática es una incomodidad más en su carrera, por lo que les interesa pasarlo con el mínimo esfuerzo. Por eso muchos recurren a copiar las tareas, o incluso a comprarlas. Este hecho ya es conocido por los profesores, y explica por qué el valor de las tareas programadas es tan bajo: sólo un 15%. Lo mismo ocurre en los exámenes: los estudiantes a veces no consideran

que copiar en este curso sea "realmente malo", pues no lo perciben como fundamental para su desenvolvimiento profesional, mientras que sostienen que por su entereza personal no deben copiar en los cursos propios de su carrera.

¿Debe eliminarse este curso del pensum de cada carrera? Eso no le corresponde a la ECCI determinarlo. Pero es natural que la ECCI se oponga a impartir cursos de paquetes, y, si los imparte, es porque la Universidad los solicita con ahínco. De todas formas, los profesores de computación siempre daremos más importancia a la parte de programación.

4. MEJORA EN LA ENSEÑANZA DE LA PROGRAMACIÓN

Para muchos profesores de computación es mejor enseñar a programar en Pascal que en cualquier otro lenguaje. Sólo el lenguage C [Ker-86] es más popular que Pascal[2]. Pero tanto C, como C++ y Ada son lenguajes muy difíciles de aprender, aún para los profesionales de la computación, por lo que si el objetivo del curso es obtener destreza en la construcción y manipulación de algoritmos, no conviene derrochar esfuerzos en un lenguaje cuyas cualidades no serán aprovechadas. Ciertamente, Pascal es el lenguaje óptimo para aprender algoritmos.

Otra gran ventaja de Pascal sobre otros lenguajes es que hay muchos buenos libros de texto en español[3] ([Gol-93], [Gro-86], [Pas-93], etc.). La dificultad más grande que deben sortear los estudiantes para aprender a programar es entender por qué y cómo secuenciar instrucciones. Esto requiere de un gran apoyo del profesor, y de intensa concentración del estudiante. Los profesores sabemos que después de comprender qué es un algoritmo, para concluir su aprendizaje el estudiante sólo necesita memorizar las reglas sintácticas del lenguaje de programación.

La práctica docente nos ha mostrado que la programación se aprende más con ejemplos que con clases teóricas. Por eso en cualquier curso de programación, en cualquier parte del mundo, los profesores usamos mucho tiempo para exponer ejemplos. Más aún, es factible no impartir clases de "teoría" si se cuenta con un buen libro de texto, pues los estudiantes lo que buscan en clase es entender la mecánica algorítmica que se muestra en los ejemplos de programación.

Precisamente, el curso Principios de Informática se imparte en el laboratorio para que los estudiantes puedan apreciar mejor los ejemplos que su profesor les enseña. Para aprender a programar hay que ver muchos algoritmos, desde muchas perspectivas.

Los profesores contamos con una gran cantidad de programas Pascal que podemos usar como ejemplos en nuestras lecciones. Uno que es particularmente interesante de analizar es el programa Armstrong.pas.

```
PROGRAM Armstrong;

{ Determina cuantos numeros son de Armstrong, desde 1
hasta numero. Un numero es de Armstrong si la suma de
los digitos que lo componen al cubo es igual al número.
Por ejemplo 153 es un numero de Arms.}

CONST  n = 3000;

VAR
numero, suma,
digito, temp : INTEGER;

begin
  WriteLn('Numeros encontrados (1 .. 3000)');
  WriteLn(' que son de Armstrong:');
  WriteLn;
  for numero:=1 to n do begin
    suma:=0;
    temp:=numero;
     WHILE temp < > 0 DO BEGIN  { suma de digitos }
   digito:= temp MOD 10;      { al cubo }
   suma:= suma+(digito*digito*digito);
   temp:= temp DIV 10
    END;
IF suma = numero then
   WriteLn(numero,'Suma de su digitos al cubo ', suma);
  end;{ FOR }
  ReadLn;
END. { Armstrong }
```

Figura 1: Armstrong.pas

2 Más bien C++ [Str–86]. Hay muchos lenguajes de programación que gozan de gran popularidad (Basic, Fox, etc.), pero Pascal es ideal para aprender algoritmos.

3 Hay pocos libros de C y C++ en español. Para los estudiantes de computación es imprescindible conocer inglés, pero este requisito no es tan importante en otras carreras.

El programa Armstrong.pas de la **Figura 1** tiene varias cualidades. Primero, es muy corto, pues consta de unas decenas de instrucciones. Además, incluye las primitivas de control más importantes: FOR, IF y WHILE. También utiliza la operación aritmética MOD. Este programa es muestra de que en una sola página, con unas cuantas instrucciones Pascal, se puede ver casi toda la materia de programación del curso de Principios de Informática.

Con mucha facilidad cualquier profesor de un curso de programación de computadores puede obtener o escribir programas como Armstrong.pas, lo que lleva a la siguiente inferencia. Como en el curso lo más difícil es lograr que el estudiante entienda cómo funciona un algoritmo, y como la manera más rápida de llegar a esta meta es exponer al estudiante a muchos ejemplos prácticos, entonces hay que enseñar de forma que el estudiante pueda ver muchos algoritmos funcionando.

La primera aproximación a esta solución ha sido impartir las clases de programación en el laboratorio. Sin embargo, esta manera de hacer las cosas presenta varios problemas:

- Los estudiantes tienen dificultad en poner atención a lo que su profesor les dice, pues deben concentrarse en hacer funcionar su máquina.
- Como pueden practicar programación en horas de clase, no tratan de estudiar por su cuenta, por lo que no profundizan en su estudio.
- Para evitar trabajar, muchas veces los alumnos se pasan los ejercicios ya sea en diskette o por medio de correo electrónico.
- Se fomenta la algarabía y el desorden en clase porque los estudiantes sienten que en las horas de práctica necesitan hablar, y terminan hablando de lo que no es relevante para el curso.
- Como el profesor está junto a ellos, los estudiantes encuentran más sencillo preguntarle cualquier pequeña duda que tengan, lo que los desalienta a esforzarse para aprender solos. En muchas ocasiones pierden tiempo esperando que el profesor les resuelva un duda que, en otras circunstancias, ellos no vacilarían en solventar por sí mismos.
- Se fomenta una actitud pasiva en el estudiante, quien llega al convencimiento de que sólo debe saber lo que ha practicado en horas de clase. Esta pérdida de la capacidad autodidacta del estudiante perjudica directamente el nivel académico del curso, pues hasta los profesores llegan a pensar que es incorrecto esperar del estudiante algo más de lo que se vio en horas de práctica.
- Hay una gran variabilidad con la velocidad en que los estudiantes pueden hacer cada práctica, por lo que quienes son rápidos se pasan a veces más de la mitad de la lección esperando que "pase algo interesante".
- Se desperdicia mucho el esfuerzo del profesor, y también su experiencia, pues invierte la mayor parte del tiempo en contestar preguntas triviales. En esto ayuda un poco que el asistente del profesor asista al laboratorio, pero muchas veces esto causa choques porque los asistentes piensan que las cosas deben hacerse de forma diferente a como el profesor las quiere.
- Por razones didácticas, a veces los ejemplos usados en clase no tienen relación con la carrera del estudiante, lo que refuerza su creencia de que el curso no es muy relevante para su futuro quehacer profesional.
- Es caro mantener un laboratorio completo para dar clases. Además, como las clases hacen uso exclusivo del laboratorio, los estudiantes que no están en el curso deben abandonar el laboratorio frecuentemente, cada vez que comienza una lección.

La principal ventaja de aprender en el laboratorio es que los estudiantes pueden practicar inmediatamente lo que su profesor les explica en la pizarra. Es natural que en el pasado los profesores de la ECCI, por medio de su director, hayan sugerido impartir las lecciones en el laboratorio, pues la práctica nos ha mostrado que para aprender paquetes de computación (Word, Excel, AutoCAD, etc) conviene recibir lecciones frente al computador.

Pero aprender a programar es mucho más difícil que aprender a usar un paquete de programas. Requiere mayor intensidad intelectual, mayor concentración y mucho más práctica. Por eso

en la Universidad cabe aprender a programar, en tanto que los paquetes se pueden aprender en otro lado, a un costo mucho más bajo.

Recientemente la ECCI adquirió una herramienta audiovisual que permite mostrar a los estudiantes el funcionamiento dinámico de un algoritmo, sin necesidad de sentarlos en el laboratorio de computadoras. Este dispositivo se conecta como si fuera un monitor del computador, y, junto con un proyector de transparencias, permite desplegar en una pantalla gigante lo que aparece en la del computador. A estos aparatos los llamamos *DataShow*, posiblemente porque esa es la marca del primero que obtuvimos.

El *DataShow* tiene sus desventajas. Primero, para usarlo se necesita disminuir la intensidad de la luz en el salón de lecciones, para que todos los estudiantes puedan ver la pantalla. Además, se necesita contar con un computador debidamente configurado al que esté conectado el *DataShow*, lo que efectivamente fuerza a que el computador permanezca anclado en el salón de clases; el aula debe permaner cerrada para que no se roben el equipo. El *DataShow* es un aparato pequeño que cabe en un maletín, y cuesta desde $700 hasta $3,000 [4]. Sin embargo, estos impedimentos se pueden solventar con un poco de organización administrativa, y con un desembolso monetario relativamente pequeño. Las ventajas operativas y didácticas que se derivan de impartir lecciones con este aparato cubren con creces estos inconvenientes.

Con un *DataShow* el profesor puede mostrarles a sus estudiantes los algoritmos en acción. Para esto basta que cargue en el computador el ambiente de programación, Turbo Pascal en este caso, y que con ayuda del Depurador Simbólico muestre a sus alumnos el efecto en las variables que se observa al ejecutar paso a paso el programa.

5. ENSEÑANZA CON EL DATASHOW

Para dar lecciones con el *DataShow* el profesor debe traer, posiblemente en un diskette, los programas y ejercicios de ejemplos que mostrará en clase. Luego de arrancar el computador, debe cargar el ambiente de programación (o su depurador simbólico), para cargar el programa que mostrará a sus estudiantes. En este momento es muy importante manejar psicológicamente al estudiantado, para que no se vean intimidados por este primer programa Pascal.

Como estudiantes y profesores están acostumbrados a que la instrucción sea gradual, de manera automática sentirán el deseo de rechazar cualquier enfoque didáctico que un alto porcentaje de la materia del curso en una de las primeras lecciones. Lo que hay que hacer no decir de entrada que un programa de ejemplo como Armstrong.pas contiene el 80% de la materia que hay que cubrir en el curso. También es muy importante que el equipo *DataShow* funcione de manera óptima: si la luz en el salón de clases es muy fuerte, o si la intensidad en la pantalla es demasiado tenue, los estudiantes usarán su capacidad de concentración para tratar de ver la pantalla, y disminuirá su capacidad para entender la explicación del profesor. Por eso también ayuda mucho repartir el texto del programa impreso en papel.

Si el profesor usara el programa Armstrong.pas de la **Figura 1** como su primer ejemplo de un programa Pascal, entonces lo primero que haría es corregirle varios defectos. Para esto puede seguir el estándar definido en las "Convenciones de programación para Pascal" [DiM-88]:

- Tildar las palabras que no están tildadas ("número", "dígitos", etc.).
- Corregir la redacción de los comentarios.
- Poner en letra mayúscula las palabras reservadas de Pascal.
- Usar espacios en blanco para que se note mejor cada uno de los componentes del programa.
- Indentar adecuadamente el programa.

Después de aplicar el maquillaje de "las convenciones" al programa de la **Figura 1**, el resultado es el programa de la **Figura 2**. Con facilidad el profesor puede mostrar a sus pupilos por qué es importante trabajar con orden al programar. Como los estudiantes necesitan obtener destreza en el manejo del ambiente de programación, particularmente en el uso del editor de tex-

4 En la ECCI se han robado una cada año

```
PROGRAM Armstrong;
{ RESULTADO
Determina cuantos números son de Armstrong, desde 1
hasta N.
- Un número es de Armstrong si la suma de los dígitos
  que locomponen elevados al cubo es igual al número.
- Por ejemplo 153 es un número de Armstrong porque
        153 = 1^3 + 5^3 + 3^3 = 1^3 + 5^3 + 3^3 }

CONST
 N = 3000;
VAR
 numero, suma,
 digito, temp : INTEGER;

BEGIN { Armstrong }
 Write('Números encontrados (1 .. ');
 WriteLn(N:1, ') que son de Armstrong:');
 WriteLn;
 FOR numero := 1 TO N DO BEGIN
  suma := 0;
  temp := numero;
  WHILE temp < > 0 DO BEGIN {suma de dígitos }
   digito := temp MOD 10;    { al cubo }
   suma   := suma + (digito * digito * digito);
   temp   := temp DIV 10;
  END;
  IF suma = numero THEN BEGIN
   WriteLn(numero,' Suma de sus dígitos al
           cubo ', suma);
  END;
 END;
END. { Armstrong }
```

Figura 2: Versión corregida de Armstrong.pas

tos, también el profesor puede mostrarles cuáles teclas usa para manipular el código y acomodar el programa para obtener esta versión de Armstrong.pas.

Luego de editar el programa Armstrong-.pas, el profesor puede mostrar cómo funciona algorítmicamente. Para esto hay que mostrar el efecto de cada instrucción en las variables. En la **Figura 3** se muestra una sección del programa junto con el valor de las variables en ese momento (la siguiente instrucción por ejecutar está marcada { < - }). Con paciencia el profesor explicará qué significa cada instrucción, cómo cambian las variables, etc. O sea, que su trabajo consiste en explicar el algoritmo como si tuviera a un solo alumno sentado junto a su computador. La diferencia la marca el *DataShow*, que se convierte en el centro de atención, pues todos los alumnos estarán concentrados en lo que ocurre en la máquina, mirando la pantalla que cambia al comando del profesor.

```
suma := 0;
temp := numero;
WHILE temp < > 0 DO BEGIN
 digito := temp MOD 10;    { < - }
 suma   := suma + (digito * digito * digito);
 temp   := temp DIV 10;
END;
```

Figura 3:Ejecución de Armstrong.pas

```
FUNCTION IS_ArmstrongCASE( {+} n:
LONGINT ) : BOOLEAN;
{ RESULTADO
  Retorna TRUE si el número "n" es un número de Arms-
trong, esto es, si la suma de los dígitos que lo componen
elevados al cubo es igual al número.
  - Por ejemplo 153 es un número de Armstrong porque
        153 = 1^3 + 5^3 + 3^3 = 1^3 + 5^3 + 3^3
- [0, 1, 153, 370, 371, 407]   }

BEGIN { IS_ArmstrongCASE }
 CASE n OF
  0, 1, 153, 370, 371, 407 : IS_ArmstrongCASE := TRUE;
 ELSE
  IS_ArmstrongCASE := FALSE;
 END; { CASE }
END;  { IS_ArmstrongCASE }
```

Figura 4: Armstrong.pas implementado con CASE

El programa Armstrong.pas tiene otras grandes cualidades, pues es un algoritmo que se puede implementar de muchas formas diferentes. Por ejemplo, después de correr el programa de la **Figura 2** es fácil concluir que sólo existen unos pocos números de Armstrong: [0, 1, 153, 370, 371, 407]. Entonces el profesor puede implementar el mismo algoritmo usando, por ejemplo, una instrucción CASE como en la **Figura 4**.

Para retomar el control de su clase, que ha perdido cuando imparte lecciones en el laboratorio, el profesor cuenta ahora con un nuevo dispositivo audiovisual: el *DataShow*. Los estudiantes tienen la oportunidad de ver más ejemplos, pues el profesor ya no tiene que copiarlos uno por uno en la pizarra. En una sola sesión es posible cubrir entonces cinco, diez o hasta veinte veces más material porque el profesor deja de ser un escribano quien repella pizarras con texto, pues cuenta con una máquina para realizar ese trabajo.

Otra gran ventaja de usar esta herramienta didáctica es que el profesor ya no necesita recurrir a los diagramas de flujo que se usaron en el pasado. La única ventaja de ese antiguo recurso didáctico es que un diagrama de flujo es una herramienta gráfica que le permite al estudiante entender mejor cómo es la dinámica de un programa, para que luego pueda formular algoritmos por su cuenta. Pero con el uso interactivo del depurador simbólico esta ventaja queda totalmente superada, pues el estudiante puede apreciar en pantalla el efecto de ejecutar un algoritmo. La desventaja más grande de los diagramas de flujo es que no sirven para representar algoritmos complicados, y además se pierde mucho tiempo al usarlos porque son difíciles de dibujar [5].

Hay una desventaja del uso interactivo del *DataShow* que es difícil descubrir. Algunos profesores se niegan a usarlo porque no quieren que sus estudiantes descubran su poca destreza en el uso del ambiente de programación. Este escollo se produce porque no todos los profesores de computación se sienten atraídos a la programación. Como esta disciplina es materia fundamental en la carrera de computación, es costumbre asumir que cualquier profesional de computación puede enseñar programación. Pero este pequeño tropiezo se puede evitar si se escoge a los profesores para este curso con un poquito de cuidado. Como no todos los profesionales en computación se dedican a programar, no debe verse como un defecto que alguno tenga poca destreza en el manejo de una herramienta de programación.

6. MOTIVACIÓN ESTUDIANTIL

El programa Armstrong.pas es un recurso didáctico muy importante, pues permite exponer la materia en una forma integral y compacta. Sin embargo, a los ojos de los estudiantes aparece como un programa bastante tonto, pues "casi no hace nada importante" [6].

Aunque no es imprescindible para que el proceso de enseñanza - aprendizaje tenga éxito, sí ayuda mucho que los estudiantes estén motivados por su curso: en cualquier curso de programación siempre hay muchas oportunidades para motivarlos.

El programador en realidad lo que hace es construir mundos virtuales, pues cada programa es un contexto completo nuevo que existe no sólo como instrucciones de computador, sino como una gran obra intelectual de quien mejor la puede apreciar: su progamador. A diferencia de otras disciplinas, como la Filosofía o las Matemáticas, en la Programación el resultado final del esfuerzo intelectual del estudiante toma vida en el computador, en donde el estudiante puede apreciar el fruto de su trabajo interactuando con la máquina. Relativamente pocos estudiantes pueden experimentar el placer de resolver un problema matemático, o de encontrar una tautología filosófica, pero varios más sí pueden disfrutar el éxito que implica terminar un programa.

Existen técnicas para lograr que el aprendizaje de la programación sea más atractivo para el estudiante. La más obvia de todas es permitirle al estudiante resolver problemas que le son relevantes. Por ejemplo, en una tarea programada mostrarán más interés los estudiantes de Ingeniería Civil si tienen que calcular el área de un poliedro a partir de varias mediciones, que si tienen que procesar una planilla de sueldos.

Los profesores de la cátedra de Principios de Informática con frecuencia aplican esta buena idea, pero no siempre recuerdan hacerlo. Por eso sería muy saludable que la cátedra contara con el apoyo de un profesor de la Facultad que conozca las aplicaciones de cómputo en las ingenierías. En las reuniones de cátedra este docente externo a la ECCI podría ser el vínculo que permita adaptar semestre a semestre el curso a las necesidades de la carrera, para llenar las expectativas estudiantiles. Por supuesto, para que pueda impartir el curso, este profesor debe conocer del lenguaje Pascal.

También conviene invitar a profesionales de firmas prestigiosas que usan día a día la programación. Por ejemplo, un ingeniero que trabaje en la construcción de equipos electrónicos po-

---

5 Solo queda una excusa para usar diagramas de flujo: los profesores los usan porque aprendieron a programar con ellos y, como enseñamos de forma similar a como aprendemos, a veces nos cuesta dejar prácticas arcaicas para adoptar métodos más modernos.

6 Estas son exactamente las plabras con que un estudiante me criticó en clase.

dría compartir con los estudiantes sus experiencias en programación. Otro podría explicarles cómo usar aplicaciones como diseño y manufactura asistido por computadora (CAD-CAM). Los estudiantes adoran estas actividades, talvez porque les hace soñar sobre lo que disfrutarán cuando concluyan su carrera universitaria.

Es prácticamente imposible que el profesor sólo utilice programas interesantes en el curso, pues como ocurre con Armstrong.pas, los más didácticos no son, necesariamente, los más atractivos para el estudiante. Pero, en definitiva, los profesores podemos lograr una mezcla satisfactoria con relativa facilidad.

Otro recurso para motivar al estudiantado es permitirles programar juegos, aunque necesariamente hay que cuidar que la complejidad del programa no sea tan grande que abrume de trabajo a los muchachos y, por ende, más bien los desanime.

También es conveniente crear una biblioteca electrónica con todo el material didáctico de los cursos, accesible a través de la tecnología Internet. En esta biblioteca se puede mantener el siguiente material:

- Carta el estudiante
- Exámenes
- Tareas programadas
- Soluciones completas
- Ejercicios
- Programas completos
- Documentación de programas

Como profesores debemos aprovechar los recursos tecnológicos disponibles no sólo para motivar a nuestros pupilos, también debemos usar las nuevas herramientas para hacer más eficiente el proceso de enseñanza - aprendizaje.

7. REFINAMIENTOS DIDÁCTICOS

En las secciones anteriores he expuesto varias ideas. En concreto, los cambios que propongo son los siguientes:

1. No enseñar a usar paquetes de computación en la Universidad.
2. No dar lecciones en el laboratorio de computadoras.
3. Usar el *DataShow* para enseñar programación.
4. Usar a profesores externos a la ECCI en el curso.
5. Crear una biblioteca didáctica para el curso.

No sé si mi recomendación será aceptada, pues ya en toda la Universidad está muy asentada la tradición de incluir en el pensum de las carreras cursos de uso de paquetes de computación, y la UCR ha mostrado ser muy lenta en realizar cualquier cambio. Por eso discuto dos posibles escenarios.

Si se eliminan los paquetes de computación del curso Principios de Inform&aacutetica entonces el tiempo que se gana se puede utilizar básicamente de dos maneras: la primera es cubrir más terreno en programación enseñando otros temas. La segunda es impartir un segundo lenguaje como Basic o C[7].

Aprender un segundo lenguaje no es una tarea imposible para el estudiante, aunque posiblemente es mejor profundizar sólo en uno, pues los lenguajes de programación se parecen mucho entre si. Es cierto que C++ y Pascal o Ada son muy diferentes, pero las diferencias sólo las notará el programador profesional cuando utiliza las cualidades más especializadas del lenguaje. Si un estudiante de Principios de Informática alguna vez llegara a usar estas facilidades seguramente estaría desarrollando un trabajo para el que no ha sido entrenado, pues lo natural es que la programación de alta intensidad sea realizada por profesionales en computación.

Existen muchos temas relevantes a la programación que no se cubren en el curso y que podrían impartirse si se elimina el uso de paquetes. Sobresalen principalmente tres: la Programación orientada a objetos, la Recursividad y el Uso de punteros. El primero ha recibido mucha publicidad recientemente, al menos en los círculos de profesionales de la computación: es la moda. La recursividad es una elegante técnica que todos

7 C++ es demasiado complejo y no debe impartirse en ningún curso de servicio.

los programadores conocemos, aunque tiene poca aplicación para profesionales de otras disciplinas. Lo mismo ocurre con el uso de punteros, pues su uso es relevante, principalmente, para implementar programas que requieren de un altísimo rendimiento. De estos tres temas, y sin lugar a dudas, la Programación orientada a objetos es el más importante, pues sirve para construir programas mucho más grandes, mientras que los otros dos temas son bastante esotéricos.

Si es necesario continuar enseñando paquetes de computación, entonces puede trabajarse de una manera un poco diferente. Una de las sesiones semanales se impartiría en el *DataShow* y versaría sobre programación. La otra sería en el laboratorio, pero estaría dedicada al uso de paquetes de computación.

El semestre tiene 16 semanas, pero de todas maneras es conveniente reservar no más de 4 sesiones en el laboratorio para que los estudiantes practiquen el lenguaje de programación. Las otras 12 se pueden usar par cubrir los siguientes temas:

- 2 sesiones: MicroSoft Word
- 2 sesiones: MicroSoft Excel
- 3 sesiones: Correo Electrónico
- 3 sesiones: Uso de Internet
- 2 sesiones: Programación de calculadoras.

A primera vista parece que se cubre demasiado, pero lo importante cuando se usan paquetes es desarrollar la habilidad de aprender a usarlos poco a poco, pues todos tienen ayudas interactivas muy completas y cuentan con mecanismos de guía que facilitan cada tarea. La Programación de calculadoras es un tema que les agrada a los estudiantes de Ingeniería, quienes usan mucho estos aparatos, y en Internet es posible conseguir emuladores de calculadoras que corren bajo el ambiente Windows, como los programas BizWiz o HP12c que se pueden obtener por ejemplo en ftp:// ftp.cdrom.com /simtelnet/win-3/calc/.

Finalmente, es saludable que los estudiantes desarrollen su cultura Internet, para que puedan usar este recurso en su desarrollo profesional.

La cantidad de sesiones dedicadas a cada tema dependerá, por supuesto, del profesor, pero muchos coincidirán en que se puede hacer un trabajo adecuado en el tiempo que sugiero. Por supuesto, los estudiantes no serán "maestros en el manejo de Excel", pero sí adquirirán las bases para que poco a poco, y en su práctica personal, desarrollen cada vez más destreza en el manejo de estos paquetes[8].

Me parece muy importante que la evaluación del curso se base, primordialmente, en verificar la destreza del estudiante para concretar algoritmos, y no en el uso de paquetes. Yo jamás le daría más de un 10% (diez por ciento) a la destreza del estudiante en el uso de paquetes de computación.

La gran ventaja de no usar el laboratorio para impartir lecciones es que quedaría para el uso exclusivo de los estudiantes, quienes ya no se verían forzados a comprar su propio computador personal pues podrían usar los de la Universidad para hacer sus trabajos. Al sacar a Principios de Informática del laboratorio la Facultad se gana un laboratorio completo, y el único costo en que incurre es adquirir un *DataShow* [9].

8. PROGRAMA PROPUESTO PARA EL CURSO

A la vista de lo expuesto en las secciones anteriores, es saludable ahora proponer un programa que acuerpe las ideas que he discutido. Para ahorrar papel (o bits si usted lee la versión electrónica de este artículo), sólo incluyo los temas principales, pues el detalle es similar al del programa del curso expuesto anteriormente, pero tiene las siguientes diferencias:

- He refinado los objetivos y destrezas que el estudiante debe adquirir en el curso.
- He eliminado del curso al paquete Excel.
- Agrego Programación orientada a objetos.
- Agrego una discusión sobre los profesionales en computación.

8 Más que darles pescado, se busca enseñarlos a pescar.

9 Con un poco de suerte se puede conseguir uno barato, pues como Turbo Pascal v5.5 corre en máquinas pequeñas, sin disco duro, basta un DataShow simple que sólo despliegue texto. La ECCI tiene uno de esos, y precísamente lo usan los profesores de cursos de servicio.

La discusión sobre el perfil del profesional en computación busca ayudar a los estudiantes a apreciar cuándo deben buscar la ayuda de otro profesional para resolver un problema específico. Opino que los profesionales de otras carreras no necesitan programar, pues para eso estamos nosotros, y entiendo que la única razón para obligar a todos los estudiantes de cualquier otra carrera a aprender programación es para que puedan comunicarse con nosotros. Si ese es el objetivo, me parece que explícitamente debería cubrirse en el programa del curso.

Por lo demás el programa no tiene ningún otro cambio significativo: es una realización de las ideas que ya he expuesto.

OBJETIVOS GENERALES

1. Desarrollar en el estudiante la habilidad para elaborar y construir algoritmos y programas sencillos.

2. Formar el criterio del estudiante para que pueda determinar el momento en que necesita del apoyo de un profesional de la computación en lugar de usar una herramienta computacional de propósito específico.

OBJETIVOS ESPECIFICOS

1. El estudiante debe ser capaz de diseñar e implementar programas simples en un lenguaje de programación de alto nivel.

2. El estudiante debe adquirir la destreza para usar un ambiente de programación para la construcción, edición y depuración de programas de computadora.

3. El estudiante debe ser capaz de aplicar reglas básicas para la documentación interna y externa de programas.

CONTENIDOS

- Introducción a la computación
- Manejo del sistema operativo
- Algoritmos y programas
- El lenguaje Pascal
- Convenciones de programación
- Programación orientada a los objetos
- Perfil del profesional de la computación

9. CONCLUSIÓN

He propuesto impartir las lecciones de programación con *DataShow* en lugar de hacerlo en el laboratorio de computadoras. Además, propongo eliminar del programa los paquetes de computación, y si esto no es aceptado, pues entonces lo más conveniente es que las lecciones de paquetes sean en el laboratorio de computadoras. De cualquier manera, conviene (aunque no es necesario) que los estudiantes de programación tengan la oportunidad de practicar programación un par de veces en el laboratorio. Para evitar que el curso pierda su rumbo es indispensable una fuerte coordinación con la Facultad. Por último, opino que la evaluación del curso debe concentrarse en las habilidades del estudiante para concretar algoritmos, y no en el uso de paquetes de computación.

Creo que aprender programación es útil y fundamental para los estudiantes de la carrera de Computación, y no lo es para estudiantes de otras carreras. Pero también opino que no somos los profesores de computación quienes debemos definir si un curso como Principios de Informática debe estar en cada curriculum.

Si estas recomendaciones fueran adoptadas, a cambio la Facultad de Ingeniería obtendría un laboratorio que en estos momentos se usa, primordialmente, para impartir lecciones. Posiblemente, la demanda insatisfecha de cupos se reduciría en forma drástica, pues ahora los cupos en el laboratorio son una cota superior para abrir grupos de Principios de Informática. A cambio, la Facultad deberá adquirir un equipo *DataShow* a un costo de varios cientos de dólares.

BIBLIOGRAFÍA

[BI-88] Borland International: ***Turbo Pascal Reference Manual version 5.5***, Borland International, California (U.S.A.), 1988.

[DiM-88] Di Mare, Adolfo: ***Convenciones de Programación para Pascal***, Revisión 2, Reporte técnico ECCI-01-88, ECCI-UCR, también disponible en Internet en http:// www.di-mare.com /adolfo/p/ conv-pas.htm, 1988.

[Gol-93] Goldstein, Larry Joel: Turbo Pascal - ***Introducción a la Programación Orientada a Objetos***, ISBN 0-13-629635-1, Prentice Hall Hispanoamericana S.A., 1993.

[Gro-86] Grogono, Peter: ***Programación en Pascal*** (Edición Revisada), Addison-Wesley Iberoamericana, ISBN 0-201-02471-3, 1984.

[Ker-86] Kernighan, Brian: ***El lenguaje de programación C***, Prentice Hall, 1986.

[Pas-93] Pascual González, Francisco: ***Domine Turbo Pascal 6***, (incluye diskette de programas), Addison-Wesley Iberoamericana, ISBN 0-201-62508-3, 1993.

[Set-92] Sethi, Ravi: ***Lenguajes de Programación - Conceptos y Constructores***, Addison-WesleyIberoamericana, ISBN 0-201-51858-9, 1992.

[Str-86]
Stroustrup, Bjarne: ***The C++ Programming Language***, Addison-Wesley, 1986.

Este trabajo está disponible en Internet en http://www.dimare.com/adolfo/p/modemtel.htm