Redescubrir Fortran: una experiencia educativa.

Laura Angelone[1,2], Jorge Dimarco[1], Alfonso Pons[1], Claudia Reynares[1], Javier Sorribas[1]

*(1) **Facultad Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario, Argentina.***

*(2) **CIFASIS (Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas)-CONICET***

Resumen

La enseñanza de Programación en las carreras de Ingeniería de la FCEIA contempla el uso de un lenguaje de programación para codificar la resolución de problemas en pseudocódigo. En el año 2000, se adoptó el Pascal, clásico para el aprendizaje de la programación a nivel universitario, por ser un lenguaje orientado específicamente a la enseñanza de la programación. No obstante, nos preguntamos porque no volver al lenguaje Fortran, dejado de lado hace 15 años en nuestra institución. Fortran es un lenguaje orientado al cálculo científico, afín a las carreras de Ingeniería, que evolucionó de sus primeras versiones y cuenta con todas las herramientas necesarias para desarrollar código utilizando programación estructurada, con iguales ventajas del Pascal. En este trabajo se presenta una experiencia de aula en la cual se comparó la actitud y el rendimiento de los estudiantes al momento de la codificación en lenguaje Pascal frente al lenguaje Fortran. Para ello se cotejó el avance de estudiantes de la primera asignatura de programación en las carreras de Ingeniería de la FCEIA. En este estudio se concluyó que a los estudiantes, sin conocimientos previos de programación, les resultó más simple escribir el código fuente en lenguaje Fortran que en lenguaje Pascal.

Palabras Clave

Educación, Programación, Fortran, Pascal.

Introducción

La enseñanza de Programación en las carreras de Ingeniería de la FCEIA contempla el uso de un lenguaje de programación para codificar la resolución de problemas en pseudocódigo. Como docentes de la asignatura Informática I del Ciclo Básico de las carreras de Ingeniería, nos preguntábamos desde hace algún tiempo sobre cuál es el lenguaje de programación que más se adecua a la enseñanza de Programación dirigida a estudiantes de carreras de Ingeniería, no afines a las áreas de Ciencias de la Computación, y con distintas especialidades, tales como Civil, Electricista, Electrónica, Industrial y Mecánica. Informática I está ubicada al inicio de las carreras de Ingeniería de la FCEIA-UNR, es una asignatura con un enfoque eminentemente práctico, orientada hacia las tecnologías modernas, y necesaria para múltiples aplicaciones en Ingeniería. En esta asignatura, en el año 2000, se adoptó el lenguaje de programación Pascal, lenguaje específicamente orientado a la enseñanza de la programación, fuertemente estructurado y que genera buenos hábitos de programación. No obstante, nos cuestionábamos por qué no volver al lenguaje Fortran, descartado los últimos 15 años en nuestra instituíción. Fortran es un lenguaje relacionado a las Ciencias Exactas, que además evolucionó de sus primeras versiones y cuenta con todas las herramientas necesarias para desarrollar código utilizando programación estructurada (Reid 2003), con iguales ventajas que el Pascal. A partir de aquí surge el interrogante de saber si realmente existen ventajas comparativas entre el uso del Fortran y del Pascal para la enseñanza de la programación. Para ello se desarrolló una experiencia educativa de aula en la que se contemplaron y compararon la utilización de ambos lenguajes en el proceso de aprendizaje. Se cotejaron los avances de los estudiantes sin conocimientos previos de programación de dos cursos similares de la asignatura Informática I, uno de ellos funcionó como testigo empleándose el

lenguaje Pascal, mientras que en el otro se enseñó el lenguaje Fortran. Con el propósito de realizar una investigación participativa, se eligieron dos cursos cuyos docentes son miembros de esta investigación. En este marco, se buscó observar, analizar y describir actitudes y rendimientos de los estudiantes en el aprendizaje de los lenguajes propuestos.

En su gran mayoría los estudiantes ingresantes a la Universidad tienen una gran carencia de conocimientos, no sólo desde el punto de vista de la formación básica, sino de los métodos de estudio y habilidades metacognitivas. Estos últimos puntos son fundamentales para encarar un curso orientado a la resolución de problemas algorítmicos, donde el pensar, el ordenar ideas, el razonamiento y la lógica son fundamentales para un correcto y eficiente seguimiento del curso.

Por ese motivo, una de las tareas más difíciles que debemos encarar a la hora de transmitir los conceptos de Programación es enseñar al estudiante a pensar. Por ello, al iniciar la enseñanza de resolución de problemas algorítmicos, nos basamos en las sugerencias de George Polya (1981) para responder a la pregunta de cómo resolver problemas. Recurrimos además a una estrategia de aprendizaje que se denomina "Análisis de problemas", basada en el modelo computacional: Datos-Proceso-Salida.

La primera etapa de esta estrategia está dada por entender el enunciado del problema. Es importante que se conozca lo que se desea que realice la computadora; mientras esto no se conozca del todo bien no tiene mucho sentido continuar con la siguiente etapa. Una vez que se ha comprendido lo que se desea que realice la computadora, es necesario definir los datos de entrada, los resultados buscados, y por supuesto la metodología de resolución, es decir, los métodos y fórmulas que se necesitan para procesar los datos y arribar a los resultados.

Teniendo en claro la metodología de resolución, se escribe, respetando ciertas reglas sintácticas y semánticas, el algoritmo en pseudocódigo.

En este punto se guía al estudiante a examinar que el algoritmo esté sin errores tanto sintácticos como semánticos. Este es un camino de procesos de "prueba humana" según Myers (1979). Los métodos son indudablemente simples a esta altura del aprendizaje: inspecciones de la codificación y pruebas de escritorio con valores medios y límites, son algunas de las sugerencias. Sin embargo, nuestra experiencia como docentes nos demuestra que los estudiantes no pueden ver sus errores, por el efecto del fenómeno que se describe como "el ojo ve lo que quiere ver". Errores de dislexia al escribir la sintaxis, incorrectos cierres de estructuras de repetición, ciclos infinitos, comparaciones entre variables con distintos tipos de dato, son algunos de estos ejemplos. Por esta razón, si al algoritmo se lo traduce a un programa, y con ayuda del compilador, el estudiante va descubriendo los errores de sintaxis que comete por desconocimiento o distracción. Luego de depurar al programa de todo error de sintaxis, se pasan datos de prueba a través de él para comprobar su correcto funcionamiento. Así el estudiante puede detectar errores tales como variables cuyo valor no ha sido inicializado, índices fuera de rango, comparaciones o cálculos inconsistentes, división por 0, exceso de rango significativo, entre otros, sin depender sólo de la corrección del docente. Esto ayuda a aquellos estudiantes que no tienen conocimientos previos de programación que su aprendizaje sea más concreto, directo y preciso. Además, al estar en contacto con la computadora para compilar y ejecutar los programas, se logra que el estudiante afronte con mejores posibilidades el análisis crítico de diferentes propuestas para la resolución de situaciones problemáticas, y vaya formando conciencia para intentar la construcción de soluciones a nuevos problemas.

Con esta metodología de trabajo, ambos grupos fueron iniciados en un camino de pensamiento lógico que los condujo a utilizar las mismas técnicas de programación. La estrategia fue comenzar a enseñar Programación utilizando los algoritmos

como recursos esquemáticos (como se describe en Joyanes Aguilar 2003) para plasmar el modelo de resolución del problema real, escritos en pseudocódigo, y luego codificarlo a través del lenguaje de programación elegido. Escribir los algoritmos resulta un tanto tedioso para los estudiantes que están deseosos de utilizar la computadora. Si bien no aparecen graves dificultades con el aprendizaje de del pseudocódigo, se puede comprobar que no resulta una tarea trivial obtener un algoritmo semánticamente correcto. El hecho de reescribir los algoritmos hasta ponerlos a punto no es simple cuando se trabaja con lápiz y papel, pero esto ayuda a que los estudiantes adquieran destrezas en el proceso del pensamiento lógico.

Para obtener los datos para medir esta experiencia, se consideraron tres momentos de evaluación diferenciados: diagnóstica, formativa y sumativa, a través de la aplicación de procedimientos cualitativos y cuantitativos. Esto permitió obtener información sobre el estado de los estudiantes al iniciar la experiencia, sobre su progreso durante la implementación de la misma, y sobre la valoración de los logros alcanzados al finalizar el proceso.

En la sección Métodos se detallan como se realizó la experiencia y en la sección Resultados se indican las derivaciones de la misma.

Métodos

La experiencia en cada curso se desarrolló en la misma institución educativa, en los mismos laboratorios, sobre las mismas PC, con los mismos docentes. Los condicionantes de la tarea se mantuvieron constantes a lo largo del desarrollo de la experiencia, tanto la composición del grupo docente como la disponibilidad de recursos (libros, apuntes y PCs).

Con el propósito de analizar la evolución de los conocimientos del estudiante y los efectos de los diferentes lenguajes de programación, se realizaron tres momentos de evaluación diferenciados: diagnóstica, formativa y sumativa.

Al inicio del curso, y con el objetivo de recabar información acerca del conocimiento inicial de los estudiantes, se instrumentó un test diagnóstico, que permitió indagar sobre los conocimientos previos en Informática. Este diagnóstico se fundó en un Test de Conocimientos Iniciales de Informática y un Test de Procedimientos. El primero tuvo como objetivo observar los conocimientos computacionales del estudiante. En este test se realizaron preguntas con respuestas de elecciones múltiples sobre temas tales como Internet, lenguajes de programación, “inteligencia” de la computadora, discos rígidos y Sistemas Operativos. Mientras que el segundo se basó en interpretación de textos y enunciados de problemas, el cual nos dio una pauta de la habilidad de razonamiento de los estudiantes. Los test pueden ser requeridos por e-mail (ver Datos de contacto).

Este test diagnóstico se realizó en cuatro cursos, para observar el nivel de los mismos, y poder armar así la población inicial formada por dos cursos homogéneos, nivelados en conocimientos informáticos y descartar de este modo todo factor negativo que pudiera dañar la recolección de datos para la muestra poblacional.

A partir de la información obtenida en este diagnóstico, el grupo de investigación organizó sesiones de clases en las que se enfatizaba la estrategia de resolución de problemas mediante Análisis de problemas y Algoritmia. Esto condujo a que ambos grupos tuvieran una base común de la técnica para encarar la resolución de problemas.

A dos meses de comenzada la experiencia, se implementó una evaluación de tipo formativa, es decir, sobre lo enseñado-aprendido en ese período. En la misma se evaluaron conocimientos informáticos tales como esquema interno de una computadora, redes, archivos, y conocimientos básicos del lenguaje de programación (Pascal o Fortran de acuerdo al grupo). Simultáneamente se realizaron observaciones áulicas, en las cuales se consideró: interés de los estudiantes en el aprendizaje del lenguaje, el grado de comprensión de Algoritmia, el

proceso de traducción del algoritmo al programa mediante el lenguaje de programación dado, sin uso de la PC, y en especial el impacto en el uso del ambiente de desarrollo de programación. En la dinámica de laboratorio no se observaron dificultades en el manejo de los entornos de desarrollo. Los estudiantes, una vez ejercitado el entorno, se manejaron con naturalidad. Estas observaciones áulicas, sirvieron de base para detectar los parámetros que teníamos que considerar para medir las dificultades de los estudiantes al momento de la codificación. De este modo se detectó que los errores más comunes de los estudiantes a la hora de la codificación fueron:

- olvido de punto y coma, begin–end, paréntesis,
- error en la sintaxis del if, repeat, while, for/do,
- error en la sintaxis del read y write,
- incorrecta llamada a subprogramas.

Al cuarto mes de clases se realizó la evaluación sumativa. En la misma se proporcionó a los estudiantes algoritmos para su codificación, sin uso de la PC, en papel, como el de Búsqueda Secuencial que se muestra a continuación. Para reforzar el análisis de la comparación de codificación entre ambos lenguajes, se realizó una instancia de Resolución de Problemas. En la misma se planteó un mismo problema a ambos grupos, en el mismo momento. Se requirió el Análisis del problema, el algoritmo de resolución y su posterior codificación al lenguaje de programación Pascal/Fortran, en papel sin uso de la PC. Estas evaluaciones nos permitieron contabilizar los errores de codificación antes mencionados en ambos lenguajes.
En la sección Resultados se pueden observar los porcentajes de estos errores.
Al finalizar el cursado se realizó el último Test de Procedimientos para medir el avance en el pensamiento de resolución de problemas. El mismo consistió en plantear problemas que debían razonar y explicar su resolución en forma escrita, tal como el conocido acertijo de Einstein, ver Paenza (2005).

```
ALGORITMO Búsqueda secuencial
variables
   X, num_sorteado(20), J : enteros
Comienzo
  LECTURA( num_sorteado, 20)
  Escribir “Ingrese su número”
  Leer( X )
  J← 1
  Mientras(X≠num_sorteado(J)yJ< 20)
   J ← J + 1
  fin mientras
  Si X=num_sorteado(J) entonces
       Escribir “Ganamos!!!!!!!!”
       sino Escribir “Perdimos”
  fin si
Fin.

Subalgoritmo LECTURA (T, cant)
Variables k, num: enteros
Desde k = 1 a cant hacer
  Escribir“Ingrese nro. sorteado”
  Leer (num)
  T(k) ← num
fin desde
fin subalgoritmo
```

Resultados

En el primer encuentro se realizó el diagnóstico de conocimientos previos. Se seleccionaron los 2 grupos que presentaron niveles similares en el Test de Conocimientos Iniciales de Informática. Sin embargo, en el Test de Procedimientos Iniciales, los que fueron calificados del 1 a 5, se notó una diferencia marcada entre ambos grupos. El Grupo Pascal presentó una mejor predisposición al razonamiento que el Grupo Fortran, ver Fig.3, situación que no consideramos relevante debido a que el objetivo que perseguíamos con estos tests era minimizar las diferencias en conocimientos informáticos previos entre los 2 grupos a estudiar.
A dos meses de iniciado el curso, se evaluaron los avances en los conocimientos informáticos. El resultado de esta evaluación formativa arrojó que ambos cursos avanzaron de igual manera en su aprendizaje.
Al finalizar la experiencia se disponían las evaluaciones de 72 estudiantes del grupo Pascal y 75 del grupo Fortran.

Con el objetivo de realizar un estudio de los errores cometidos en la codificación en ambos grupos, se analizaron las codificaciones que realizaron los estudiantes sin el uso de la PC, en papel. Se plantearon varios algoritmos del estilo del algoritmo Búsqueda secuencial antes mencionada. Con estas codificaciones se contaron uno a uno y sin repetición por cada examen, los errores antes mencionados. Sin repetición significa que si en un examen un estudiante se olvidó más de un paréntesis, solo se lo contabilizó una vez. Los resultados obtenidos se muestran en la Fig.1. En la gráfica se observa que el 80% de los estudiantes del grupo Pascal se olvidaron al menos un punto y coma. El 63% tuvieron dificultades con al menos un begin-end. Mientras que en la codificación Fortran el mayor porcentaje fue el 22% en los estudiantes que se olvidaron al menos un paréntesis.

En la Resolución de Problemas se hizo foco sólo en la etapa de codificación, y con pautas objetivas y precisas, se calificaron del 0 al 10. Con estas notas se realizó el gráfico que se observan en la Fig.2. Se observa que el grupo Fortran tuvo mejores resultados, esto es, menor cantidad de errores en la codificación.

Al finalizar el curso se plantearon problemas que los estudiantes debían explicar su resolución en forma escrita, Test Final de Procedimientos. Este test sirvió para medir el avance en el pensamiento de resolución de problemas. Se calificó cada evaluación con notas del 1 al 8. Con estas notas se realizó el gráfico de Fig.4. En la gráfica se observa que un 65% del grupo Pascal obtuvo una nota máxima contra un 55% del curso Fortran. Cabe notar que el curso Pascal comenzó con un mejor nivel de razonamiento en la interpretación y resolución de problemas, como se observa en la Fig.3. Sin embargo, este último test arrojó que ambos cursos mejoraron satisfactoriamente sus niveles de interpretación de enunciado, pensamiento lógico, y redacción de la solución a los problemas planteados.

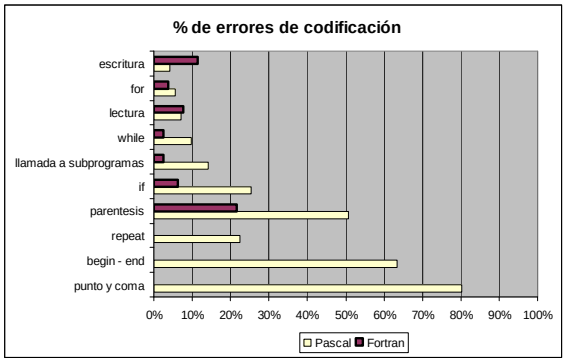

Figura 1. Porcentajes de errores de codificación

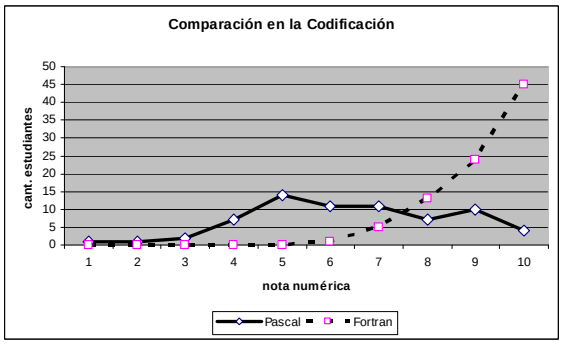

Figura 2. Resultado de la codificación

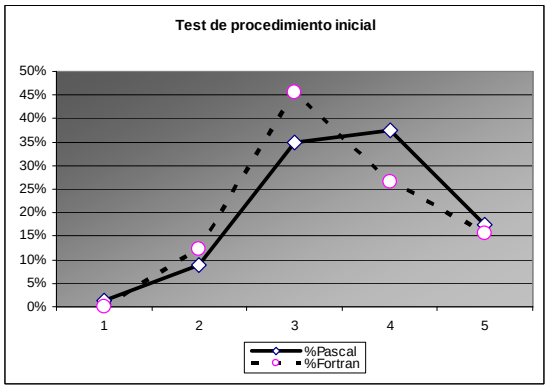

Figura 3. Resultado del test inicial de procedimiento nota vs. % de estudiantes con esa nota

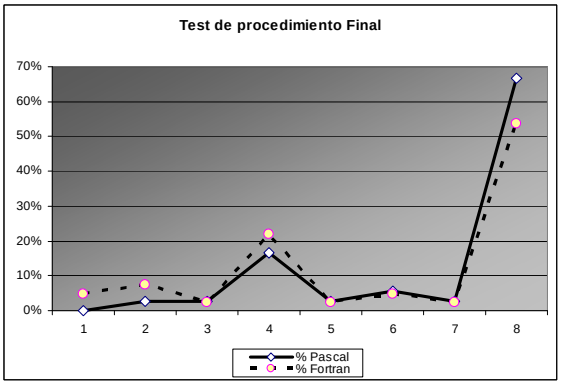

Figura 4. Resultado del test final de procedimiento nota vs. % de estudiantes con esa nota

Conclusión

Basándonos en los resultados obtenidos en las distintas instancias de seguimiento de ambos grupos de estudiantes (Pascal y Fortran), se puede inferir que la codificación en lenguaje Fortran provee resultados coherentes y racionales dentro del marco teórico desarrollado.

En este trabajo de investigación se observó que la codificación de los algoritmos en lenguaje Fortran les resulta más simple a los estudiantes en los inicios de la programación. Esto se concluye debido a que los errores iniciales de sintaxis están disminuidos por la facilidad que tiene este lenguaje.

El lenguaje Pascal fue pensado en sus orígenes como un lenguaje fuertemente estructurado y orientado específicamente a la enseñanza de la programación; este último motivo fue de fundamental importancia para que los docentes de Informática I se decidieran, en el año 2000, a adoptarlo como herramienta para codificar los algoritmos que se desarrollaban en la asignatura. Pensamos que este razonamiento puede ser muy válido en carreras de Ingeniería vinculadas específicamente a Sistemas donde en asignaturas posteriores se estudian otros lenguajes incluso de características muy distintas entre sí, por lo cual se puede considerar que es muy importante iniciar a los estudiantes de primer año en un lenguaje fuertemente estructurado y con reglas sintácticas lo suficientemente rígidas como para generar buenos hábitos de programación. Sin embargo, en nuestro caso, en las carreras de Ingeniería no afines a Sistemas donde el aprendizaje de la Algoritmia y la Programación deben constituirse en una herramienta útil para la posterior resolución de problemas en las distintas áreas de Ingeniería y no en un fin en sí mismo, nos parece más apropiado adoptar un lenguaje con reglas sintácticas más simples, como lo es el Fortran, que deriven en una mayor rapidez en la codificación de algoritmos y una menor posibilidad de cometer errores de sintaxis, dando lugar a disponer de mayor tiempo de razonamiento para encontrar la mejor solución al problema planteado.

Agradecimientos

Queremos agradecer a la Lic. Ma. Inés González, directora del Dpto. de Matemática FCEIA, por su generosidad al brindarnos la oportunidad de realizar la investigación de campo con nuestros propios alumnos. A nuestro compañero Fernando Roldán por sus aportes, fundamentales para la concreción de este trabajo. Finalmente, al PhD. Javier Signorelli por brindarnos desinteresadamente todos sus conocimientos sobre el uso del lenguaje Fortran a nivel profesional.

Referencias (Times New Roman, 10, negrita).

[1] Joyanes Aguilar, L., *Fundamentos de Programación. Algoritmos, Estructuras de datos y Objetos*, 3ª Ed., McGraw-Hill, 2003.

[2] Myers, G., *El arte de probar el software*, El Ateneo, 1979.

[3] Paenza, A., *Matemática.. estás ahí?* ISBN 987-1220-19-7, Siglo XXI Editores Argentina, 2005.

[4] Polya, G., *Cómo plantear y resolver problemas.* Serie de matemáticas. Trillas. México, 1981.

[5] Reid, J., *The New Features of Fortran 2003,* ISO/IEC JTC1/SC22/WG5 N1579, 2003.

Datos de Contacto:

Laura Angelone *langelon@fceia.unr.edu.ar*
Javier Sorribas, *sorribas@fceia.unr.edu.ar*
Claudia Reynares, *reynares@fceia.unr.edu.ar*
Alfonso Pons, *alfonso@fceia.unr.edu.ar*
Jorge Dimarco, *jdimarco@fceia.unr.edu.ar*

FCEIA-UNR. Pellegrini 250 - 2000 Rosario