TED

Tecné, Episteme y Didaxis: TED
ISSN: 2665-3184
ISSN: 2323-0126
revistated.fct@gmail.com
Universidad Pedagógica Nacional
Colombia

Uso didáctico de estrategias inductivas en un curso introductorio de programación estructurada

Miños Fayad, Alejandro

Uso didáctico de estrategias inductivas en un curso introductorio de programación estructurada
Tecné, Episteme y Didaxis: TED, núm. 39, 2016
Universidad Pedagógica Nacional, Colombia
Disponible en: https://www.redalyc.org/articulo.oa?id=614264960009
DOI: https://doi.org/10.17227/01203916.4583

Reporte de caso educativo

Uso didáctico de estrategias inductivas en un curso introductorio de programación estructurada

Using Inductive Teaching Strategies in an Introductory Structured Programming Course

Uso didático de estratégias indutivas num curso introdutório de programação estruturada

Alejandro Miños Fayad
Consejo de Formación en Educación Profesor Interino de Didáctica, Uruguay
alejandromifa@gmail.com

DOI: https://doi.org/10.17227/01203916.4583
Redalyc: https://www.redalyc.org/articulo.oa?id=614264960009

Recepción: 23 Marzo 2015
Aprobación: 10 Octubre 2015

Resumen:

El abordaje de la programación en las carreras informáticas se realiza como un fin en sí mismo desde las asignaturas que estudian el conjunto de técnicas, o como parte de otras asignaturas en las cuales se analizan sus algoritmos específicos. Así, el estudio de las técnicas de programación se convierte en un elemento transversal y estructurante de los planes de estudio de estas carreras. En consecuencia, se debe buscar la realización de actividades en las que el estudiante pueda experimentar y analizar los resultados a fin de construir soluciones viables. Las estrategias inductivas permiten analizar, discutir y luego generalizar propiedades o procedimientos, que se estudian primero en casos particulares, debidamente delimitados. En el caso de la programación, estas estrategias se deben abordar a partir de un programa, sobre el cual se estructura la actividad de aula. En este trabajo se ha indagado, mediante preguntas de respuestas abiertas, acerca de la predisposición hacia estrategias de aprendizaje inductivas de un grupo de estudiantes de un curso de Bachillerato de Informática, en la asignatura Programación I. Las estrategias se centraron en un conjunto de actividades durante el transcurso del año lectivo. En todos los casos, el uso de la codificación de programas en lenguaje Java, su ejecución y posterior análisis y discusión de los resultados fueron parte del trabajo de aula. Hemos podido concluir que en el grupo seleccionado, los estudiantes se sintieron mayoritariamente interesados y predispuestos positivamente hacia actividades de tipo inductivo, en detrimento de las de tipo deductivo. Corroboramos que prefieren el trabajo a partir de ejemplos y casos particulares sobre las actividades que abordan primero los conceptos generales por estudiar, sus características y propiedades, para posteriormente operativizar y trabajar con los casos particulares. Al mismo tiempo, pudimos verificar que ha existido una mayor dificultad en comprender los contenidos que se trabajaron de modo deductivo que los de tipo inductivo.

Palabras clave: Didáctica, informática, métodos inductivos, lenguajes de programación, enseñanza.

Abstract:

Programing in computer science careers is approached as an end in itself in the subjects that study the techniques, or as a part of other subjects, in which specific algorithms are analyzed. Thus, the study of programing techniques becomes a cross-curricular and structuring element for these careers' plans of study. Therefore, attention should be paid to the work with strategies designed to motivate students and maximize their learning. Likewise, it is necessary to perform activities that provide opportunities for students to experiment and analyze results in order to come up with workable solutions. Inductive strategies make it possible to analyze, discuss and generalize properties or procedures, which are first studied in particular cases properly defined. In programming, these strategies must be approached from a program the classroom activity will be based upon. This study inquired, through open-ended questions, about students' attitude towards inductive learning strategies. The sample was composed by a group of students from Computer Programming I, a subject matter from a Computer Science Diploma. The strategies focused on a number of activities carried out during the academic year. In all cases, the use of program coding in Java, its implementation, analysis and discussion of findings were the result of classroom work. It was concluded that most of students in the group were more interested and had a positive attitude towards inductive activities than towards deductive ones. It was confirmed that they prefer dealing with examples and particular cases from the beginning, rather than addressing general concepts to be learned, their features and properties, and finally focusing on particular cases. Furthermore, it was ratified that it is more difficult trying to understand deductively-addressed contents than those that were addressed inductively.

Keywords: Didactics, computer science, inductive methods, computer programming languages, teaching.

Resumo:

A abordagem da programação de cursos de computação realiza-se com o objetivo das disciplinas da grade curricular que estudam o conjunto de técnicas, ou como parte de outras disciplinas nas quais são analisados os algoritmos específicos. Assim, o estudo de técnicas de programação se torna um elemento transversal e estruturante dos planos de estudo destes cursos. Neste sentido, é necessário procurar a realização de atividades, nas quais o aluno possa experimentar e, analisar os resultados no intuito de construir soluções viáveis. As estratégias indutivas permitem analisar, discutir e generalizar propriedades ou procedimentos, que são primeiro estudados em casos particulares, devidamente definidos. No caso da programação, tais estratégias devem ser consideradas, a partir de um programa sobre o qual é estruturada a atividade de aula. Neste trabalho se tem questionado por meio de perguntas a respostas abertas sobre a predisposição de um minicurso de ensino fundamental sobre computação na disciplina de programação I. As estratégias foram focadas em um conjunto de atividades durante o desenvolvimento do ano escolar. Em todos os casos, o uso da codificação programas Java e posterior análise e discussão dos resultados foram parte do trabalho de aula. Temos podido concluir que no grupo selecionado, a maior parte dos alunos estiveram interessados positivamente nas atividades de tipo indutivo em comparação com as atividades de tipo dedutivo. Constatamos que eles preferem o trabalho a partir de exemplos e casos particulares sobre as atividades que implicam primeiro os conceitos gerais, suas características e propriedades para depois trabalhar com casos particulares. Ao mesmo tempo, conseguimos verificar que tem existido uma maior dificuldade na compreensão dos conteudos que foram trabalhados de forma dedutiva que aqueles que foram trabalhados de forma indutiva.

Palavras-chave: didática, computação, métodos indutivos, linguagens de programação de ensino.

Introducción

La programación, como conjunto de técnicas, es estructurante de los planes de estudio de las carreras de Computación. En efecto, la construcción y el estudio de algoritmos son fundamentales a la hora de presentar cualquier solución relacionada con la disciplina, a la vez que parte de su dominio material (Barchini, Sosa & Herrera, 2004). Comprender el currículo de esta forma es sinónimo de entender no solo que todo se programa, sino que sin programación solo es posible trabajar con lo que se denomina computación teórica. De este modo, en la enseñanza de la informática, o computación, pasa a ser determinante el estudio de las distintas técnicas de programación, ya sea estructurada, orientada a objetos o concurrente, entre otras. Típicamente, las carreras técnicas o científicas del área de informática presentan un abordaje paulatino de la enseñanza de la Programación, más allá del paradigma elegido, en el cual resultan de vital importancia los cursos introductorios al estudio de estas técnicas.

Los primeros cursos de programación, o Introducción a la Programación, tienen una doble importancia en los planes: primero como elemento motivador y segundo como formación técnica específica. Desde el componente técnico se hace necesario presentar actividades que resulten interesantes para los alumnos, y fomenten su autoestima y sus capacidades metacognitivas, entre otras. Es aquí donde el docente debe realizar elecciones didácticas de vital importancia, pues debe conjugar el trabajo con los contenidos técnicos y con aquellas actividades motivantes que incentivan los aprendizajes de los alumnos. Mientras los primeros aseguran el dominio técnico específico, indispensable para lograr el perfil académico de egreso, las segundas permiten que el estudiante se sienta interesado, atraído y predispuesto hacia el aprendizaje de los contenidos seleccionados. Es de destacar que muchas veces los cursos de Introducción a la Programación, o Programación I, son los primeros relacionados con la informática a los que se enfrentan los alumnos, lo que a su vez puede determinar que continúen o no en la carrera, por el éxito o fracaso que puede suponer el desarrollo del curso.

Antecedentes y marco teórico

Varios artículos versan sobre la enseñanza de la programación. Estos se agrupan en dos grandes áreas: (1) el análisis curricular, más que en las estrategias de transposición didáctica, y (2) en la codificación y el análisis de las soluciones. Mientras Llorens, Satorre y Puchol (1996) destacan la importancia de comprender el algoritmo independiente del lenguaje escogido y el uso de la computadora, otros artículos abordan el trabajo curricular a fin de reducir "el problema del desplazamiento" (Fernández, Peña, Nava & Velázquez, 2002, p. 3)

que supone el aprendizaje de un nuevo paradigma de programación. Un segundo grupo de trabajos se centran en la programación propiamente dicha, por ejemplo el abordaje que hace Ebrahimi (1994) de los errores asociados a los constructores de cuatro lenguajes, y la importancia de formular un plan de trabajo viable; o el trabajo de Götschi, Sanders y Galpincon (2003), en el cual se analizan los modelos mentales de recursión.

No obstante, son pocos los trabajos que analizan el abordaje de los contenidos programacionales desde un punto de vista del aula, es decir, de las actividades concretas con las que se trabajará. El artículo titulado "Enseñando programación con C++: una propuesta didáctica", de Jorge Díaz (2006), presenta algunos lineamientos de trabajo para la enseñanza del lenguaje C++; en él se indican dificultades observadas y un posible abordaje de los contenidos mencionados en el artículo. Villalobos (2009) plantea algunas dificultades asociadas a la enseñanza de la programación, se destaca el hecho de que el aprendizaje parece darse por imitación y "se da la sensación de que el 'cómo hacer las cosas' no es enseñable, y que es algo que depende de la inspiración y del ingenio del programador" (p. 10). Ferreira y Rojo (2006) abordan la enseñanza de la programación desde la realización de algoritmos de alto nivel, sin "preocuparse por los detalles de programación propios de cada lenguaje", al tiempo que sostienen que es necesario que el alumno "reconozca que la solución de un problema es una actividad metódica, y como tal, involucra una serie de etapas" (p. 4), para dar unidad a la construcción de software. Por último, Pérez, Fuentes y Moreno (2008) afirman que "las técnicas de representación de algoritmos [diagramas de flujo, pseudocódigo, etc.], no parecen ayudar sustancialmente a resolver el problema que presentan los estudiantes, sobre su ineficiente capacidad para desarrollar algoritmos que reflejen correctamente las soluciones solicitadas" (p. 5). Por consiguiente, es necesario acotar, delimitar y analizar convenientemente el problema por resolver.

Tal cual establece la Real Academia Española (2015), la gramática "estudia los elementos de una lengua, así como la forma en que estos se organizan y se combinan". Desde este punto de vista existen similitudes con el estudio de la programación, pues si bien esta última no se limita a la sintaxis de los lenguajes, ni a la combinación de sus elementos, sí estudia lo anterior a fin de resolver un determinado algoritmo. Por tanto, para poder programar deben conocerse los constructores del lenguaje dado, en particular sus características y utilidades, y no solo su sintaxis. Por esta razón, se destaca el trabajo de Rodríguez y Valero (1998), en el cual analizan la incorporación de estrategias inductivas en el aprendizaje de la gramática. Las autoras proponen actividades inductivas, que consideran que potencian el aprendizaje autónomo.

Sin embargo, la mayoría de los trabajos relevados analizan las características generales del enfoque que se le da a un curso, pero no a las estrategias de aula usadas para el trabajo de un contenido propio de la programación. Estos elementos, propios de la Didáctica de la Informática, son fundamentales por cuanto establecen elementos y guías de acción docente a nivel de aula y permiten analizar las relaciones que se dan entre el saber, el docente y el alumno (Houssaye, 1992).

Consideremos un ejemplo: el aprendizaje del concepto de variable en computación y su uso, que representan una dificultad significativa para los estudiantes. En este caso se debe conjugar la conceptualización de lo que es una variable con su posterior aplicación en un programa en concreto, lo cual está atado a un tipo de datos en particular. Una posible opción de trabajo consiste en definir el concepto de variable así como su utilidad, las características asociadas a ella, enumerar y describir los tipos de datos existentes en el lenguaje usado y luego ejemplificar su uso, los casos concretos de aplicación. Otra posibilidad la constituye la definición incompleta del concepto de variables, a partir de un ejemplo, y su trabajo con un tipo de datos en particular, como pueden ser los enteros, para posteriormente presentar y trabajar con los otros tipos de datos existentes en el lenguaje. Mientras la primera opción es más extensa, desde lo conceptual y procedimental, por la necesidad de trabajar con varios tipos de variables, la segunda actividad restringe las posibles aplicaciones a aquellas que se pueden operar con enteros.

Desde una perspectiva estrictamente motivacional, correspondería analizar si resulta adecuado, interesante o desafiante para los alumnos presentar los distintos tipos de variables existentes, sin usarlas en el corto plazo. Al no ser posible trabajar en el transcurso de una clase los tipos enteros, booleano, flotante, byte y real, sino

solamente nombrarlos y presentar brevemente sus características, corresponde preguntar si la instancia de trabajo resultante deviene en una situación estresante y monótona para el alumno.

Ahora bien, restringir la presentación de variables a un solo tipo de datos implica realizar una planificación de actividades más detallada, haciendo énfasis en el tipo de datos seleccionado. Si bien reducir el uso de variables a las de un tipo puede simplificar el proceso de transposición, también puede reducir las actividades futuras a la aplicación de ejercicios que resulten reiterativos. En efecto, si se ha trabajado con variables enteras, solo será posible realizar actividades con este tipo de datos, eliminando el uso de cadenas de caracteres, por ejemplo. Al mismo tiempo, si bien lo anterior puede dar lugar a actividades rutinarias y demasiado orientadas a problemas matemáticos, por el tipo de datos elegido, se podrá incrementar la motivación por el logro, pues los estudiantes verifican la posibilidad de resolver los problemas planteados.

Las estrategias de trabajo no son únicas en un curso, ya que deben tener relación con los contenidos y las opciones metodológicas que elija el docente, las cuales tienen relación con las características del grupo, su contexto y proyecto de centro (Antúnez, del Carmen, Imbernón, Parcerisa & Zabala, 1992). En el ejemplo planteado, aunque simplificado y asumiendo que no son los únicos posibles, podemos observar la existencia de dos grandes lineamientos metodológicos, que guiarán las actividades presentadas por el docente: la deducción y la inducción. Los métodos deductivos se basan en presentar las generalidades y características del objeto de estudio, para luego abordar los casos particulares; los métodos inductivos resaltan los ejemplos y situaciones particulares como forma de generar instancias que permitan la posterior generalización de los mismos. La primera forma de abordar el trabajo con variables, desde la presentación de las características, los usos y elementos generales está fuertemente relacionada con la deducción, en tanto que la segunda actividad se enfocará en el ejemplo, el caso concreto que posteriormente es generalizado, es decir, la inducción. El estudio de casos, propio de los métodos inductivos, solo es posible si permite la posterior generalización de conceptos o procedimientos, a fin de que no se limite al estudio de situaciones muy particulares de difícil aplicación real. Esta división no siempre es clara, Cañadas (2002, p. 36) toma la idea de Poyla en el sentido de que "el razonamiento inductivo aparece como un tipo de razonamiento en el que se parte de hechos particulares y se busca la generalidad de los hechos que acontecen".

Desde estrategias metodológicas inductivas es necesario hacer hincapié en las actividades que permitan generalizar conceptos o procedimientos por parte de los alumnos. Estas actividades pasan a tener un lugar preponderante en la planificación docente, no solo como aplicación o reforzamiento del contenido, sino también como forma de generalizar los conceptos o procedimientos trabajados. Al mismo tiempo, a la hora de secuenciar los contenidos se deben considerar entre otros aspectos: la lógica de la disciplina, la adecuación de los nuevos contenidos a los conocimientos previos de los estudiantes, la correcta delimitación de ideas rectoras de la actividad y la continuidad y progresión de contenidos (Antúnez et ál., 1992). Estos aspectos cobran especial importancia desde estrategias inductivas, pues las mismas ponen de relieve los casos particulares, los cuales pueden incluir singularidades que eventualmente se transformarán en elementos distractores.

Las tareas, como "secuencia ordenada de todas la actividades y recursos que utiliza el profesor... con un fin determinado de aprendizaje" (Antúnez et ál., 1992, p. 123) deberán estar lo suficientemente acotadas para lograr las generalizaciones de procesos o conceptos, identificando el objeto de estudio, sus propiedades, y que se pueda "definir claramente y conceptualmente el problema" (Cuevas, Bautista & Medina, 2013, p. 4). Es así que la actividad podrá ser estructurada de modo que se trabaje con conceptos o procedimientos sin hacer hincapié en ellos, o en su definición formal, reduciendo su uso a lo estrictamente necesario para trabajar el ejemplo. De este modo, estos conceptos o procedimientos actúan como un axioma, en el sentido de que se transforman en una verdad incuestionable para el alumno. El docente deberá cuidar que el axioma no sea en sí mismo un elemento distractor, y que coadyuve al trabajo con el caso particular, en lugar de obstruirlo. Al mismo tiempo debe recordarse que estos "axiomas" son un medio para el aprendizaje, por lo que en algún momento deberán ser retomados y trabajados en clase. Tal cual dijimos, los elementos distractores deberán

controlarse al máximo, de modo que el estudiante enfoque su esfuerzo cognitivo en aquellos elementos que se tratarán en la clase. Por ejemplo, si el objetivo de nuestra actividad es recorrer un vector, debemos centrarnos en ello, diferenciando la recorrida del mismo del ingreso de datos a él. Este último elemento podría ser trabajado axiomáticamente (por ejemplo mediante un método proporcionado por el docente), de modo que se simplifique este aspecto de la clase, y se evite que se constituya en un elemento distractor.

Finalizada la clase, es necesario que el estudiante se haya apropiado, si no del concepto o procedimiento generalizable, sí del problema o elemento que se va a generalizar. Para Bruner, referenciado por Antúnez et ál., será necesario que el estudiante identifique "unas ideas centrales que actúan como eje de desarrollo" (1992, p. 92). En estrategias inductivas este elemento es particularmente delicado, pues el ejemplo o caso particular puede incluir otros contenidos que no han sido trabajados, o lo fueron muy superficialmente. De este modo, el estudiante deberá poder identificar estos aspectos esenciales de la actividad. Por ejemplo, si bien una actividad no necesariamente estará centrada en crear una solución, por lo que no es determinante la capacidad de construir un plan en el sentido dado por Ebrahimi, sí será fundamental que dicho plan se pueda identificar claramente, y que no se constituya en un mero entendimiento de sentencias explicadas por el docente. Al mismo tiempo, y dado que se han identificado las ideas claves, es más probable que el estudiante se sienta motivado por el trabajo, pues "la principal reflexión que hace el sujeto tiene que ver con buscar una explicación al resultado, atribuir una causa al éxito o al fracaso obtenido" (Míguez, 2007, p. 320).

Metodología y población

Dada la ausencia de trabajos que aborden el trabajo de los conceptos básicos de programación desde estrategias de aula a nivel micro, es decir, la forma de presentar y trabajar dichos conceptos desde las distintas actividades de aula, la presente investigación es de tipo exploratorio. Al mismo tiempo, el trabajo ha sido principalmente cualitativo dada la intención de comprender la percepción que tienen los estudiantes sobre la estrategia de aula.

La población analizada correspondió a trece estudiantes de un curso de Programación I, primer año de Enseñanza Media Tecnológica (EMT) en Informática, dependiente del Consejo de Educación Técnico Profesional, de Montevideo, Uruguay. Este tipo de curso, de tres años de duración, tiene por objetivo formar personal calificado, con competencias técnico-tecnológicas en el área informática. El plan de los cursos de EMT se organiza sobre dos grandes ejes curriculares: el espacio tecnológico y el equivalente; el primero centrado en la especificidad técnico-tecnológica del curso, y el segundo común a todos los cursos EMT, con asignaturas como Inglés, Análisis y Producción de Textos o Ciencias Sociales, entre otros. Los EMT, para las distintas especialidades, son parte de la enseñanza media superior, pre terciaria; los alumnos de primer año tienen alrededor de 16 años.

El programa de la asignatura Programación I busca abordar el trabajo computacional desde la programación estructurada. Este curso es particularmente importante desde la Computación, entre otras cosas por las constantes referencias a la programación que se encuentran en los programas de otras asignaturas. Variables, condiciones, iteraciones, cadenas de caracteres y vectores constituyen los grandes ejes temáticos de Programación I. En el curso de Programación II, que se dicta durante el segundo año de la carrera, son abordados brevemente los conceptos de análisis y diseño de programas bajo el paradigma de la programación orientada a objetos, y con mayor profundidad la programación propiamente dicha.

Tanto en Programación I como en Programación II se utiliza como lenguaje de programación Java, el cual está construido y pensado para programar bajo el paradigma de objetos (Deitel & Deitel, 2007). De este modo, y al no variar el lenguaje usado en ambas asignaturas, se logra "mantener la uniformidad y evitar la exposición de los aspectos léxicos, sintácticos y semánticos" que acarrearía el uso de dos lenguajes distintos, aunque "emplear un lenguaje que no está orientado al paradigma explicado genera confusión y hábitos

inadecuados en el alumno" (Fernández et ál., p. 3). Cabe acotar que la asignatura Programación I se presenta como una de las más difíciles del plan de estudios, con altos niveles de reprobación de estudiantes.

Durante el año lectivo 2014 organizamos cinco actividades de tipo inductivo; cada una de ellas se centró en el trabajo con un nuevo contenido, para luego analizar el trabajo de los alumnos, sus intervenciones y actividades posteriores.

A fin de conocer la opinión de los estudiantes con relación a la forma en que prefieren que se presente un determinado contenido, realizamos la primera semana de octubre de 2014 una encuesta anónima entre ellos, la cual constaba de cuatro preguntas abiertas:

- ¿Resulta útil presentar un ejemplo en la computadora para luego explicar su utilidad?
- ¿Se entiende mejor a partir de un ejemplo en la computadora?
- ¿Es mejor presentar un tema de forma "teórica" o a partir de un ejemplo?
- ¿Cuál tema fue el más difícil?

El uso de preguntas abiertas estuvo motivado por el hecho de que "son particularmente útiles cuando no tenemos información sobre las posibles respuestas. [y]... sirven en situaciones donde se desea profundizar una opinión" (Hernández, Fernández & Baptista, 1991, p. 221).

Si bien estas preguntas son muy parecidas entre sí, e incluso algunas de ellas levemente redundantes, no todas se enfocan en el mismo aspecto por estudiar (la presentación de un tema de forma inductiva), aunque sí están relacionadas. Por ejemplo, en el caso de la segunda pregunta, no necesariamente está relacionada con métodos inductivos de trabajo, sino con el uso de un recurso didáctico y su incidencia en la enseñanza de la programación. El análisis de los resultados, y su relación con estrategias inductivas se abordará más adelante.

Actividades de aula

Describiremos a continuación las actividades de presentación de los contenidos usadas en el curso indicado. Luego de codificado el programa (omitimos el método main, el uso de bibliotecas y la clase de inicio), ejecutado y analizada la salida obtenida, procedimos a discutir en grupo los resultados obtenidos. Como estrategia de trabajo usamos la verbalización y la negociación de significados entre los alumnos, asegurándonos de guiar la discusión grupal. En todos los casos, los alumnos utilizaron el entorno de desarrollo Eclipse.

Retomando aspectos metodológicos, debemos agregar que las actividades presentadas hacen hincapié en situaciones particulares, casos de estudio, los que luego son generalizados (Poyla, citado en Cañadas, 2002). Para las actividades trabajadas podemos observar lo antedicho, en el sentido de que, desde un caso particular, se procura generalizar dicho concepto o procedimiento. Es de destacar que las actividades hacen referencia a la presentación del contenido, el cual continuó siendo trabajado posteriormente.

Actividad Variables

(1) int a = 10;

(2) System.out.println ("El número asignado a la variable a es: "+a);

El objetivo de esta actividad fue presentar la salida en pantalla en entorno comando, y una primera aproximación al manejo de variables. La modificación en la asignación de la variable (1), permite que el alumno observe como esto afecta el valor de la misma, pues la salida en pantalla (2) mostrará el valor asignado.

El trabajo con variables se redujo al uso de las de tipo entero (int), procurando evitar la sobrecarga cognitiva y presentar solo aquellos tipos que es probable usar al principio del curso, sin entrar en detalle sobre aquellas que no se aplicarán en las siguientes actividades (como los tipos boolean o String). El uso de las variables

estuvo orientado principalmente a su enfoque desde el concepto de roles de variables (Saavedra & Silveira, 2011), su función y utilidad dentro del programa, más que en una definición formal y estricta de ellas. El trabajo con variable continuó durante el resto del año, y el concepto estuvo sujeto a refinaciones sucesivas. En el caso de la sentencia que permite la salida en pantalla (System.out.println) fue presentada de forma axiomática, sin insistir en el concepto de clase asociado a System, ni a sus métodos.

Si bien los alumnos identificaron cómo operativizar el uso de variables, a partir del ejemplo y la aplicación, esto no se convirtió en sinónimo de identificar aquella información por almacenar en una variable. Por tanto se continuó trabajando el concepto de variable, su necesidad e implicancias algorítmicas.

Actividad If

```
(1) int y;
   (2) System.out.println("Ingrese un número");
   (3) y=leer.nextInt();
   (4) if (y>0)
   (5) System.out.println("el número es positivo");
   (6) else
   (7) System.out.println("el número es negativo");
```

Para esta actividad trabajamos previamente con el ingreso de datos por teclado, mediante objetos de tipo Scanner.

En este caso, nuestro objetivo fue presentar la sentencia if, de modo que los alumnos identificaran su utilidad. Se hizo necesario que el programa fuera ejecutado varias veces, pues los alumnos tendían a ingresar números positivos en lugar de negativos. A fin de comprender la utilidad de la condición lógica, se realizaron múltiples ejecuciones, y fue preciso ser explícitos en que los alumnos probaran el resultado obtenido al ingresar números negativos.

Pudimos observar que los alumnos reconocieron que la existencia de la sentencia if estaba relacionada con el mensaje de salida en pantalla. El ejemplo planteado, así como su aplicación en una situación concreta, permitieron que los alumnos identificaran la utilidad de la sentencia if, más allá de la consigna inicial.

Actividad for

```
(1) int y;
   (2) Scanner dato = new Scanner(System.in);
   (3) for(int i=0;i<10;i++) {
   (4) System.out.println("Ingrese un numero");
   (5) y=dato.nextInt();
   (6) y=y+1;
   (7) System.out.println("El siguiente es: "+y); }
```

Esta actividad tuvo por objetivo presentar el concepto de iteración, y en particular la sentencia for. La ejecución del programa es tal que el estudiante identifica que se le solicita un número (5), y luego se muestra el siguiente. El trabajo con la condición permitió modificar el número de salidas en pantalla, por lo que el concepto de iteración se asimiló convenientemente. La presencia de mensajes por consola es fundamental, pues aquellos estudiantes que no las codificaron presentaron dificultades para entender el funcionamiento del código.

Se presentaron dificultades en el número de iteraciones por realizar, y la forma de controlarlas. Si bien es cierto que los alumnos reconocieron que la sentencia for permite realizar iteraciones, las cuales se conocen

de antemano, la forma de incremento de la variable i no resultó trivial. El incremento del dato solicitado (6), aunque trabajado en otras instancias, supuso una dificultad extra para los estudiantes.

Pudimos observar que en general no fue difícil identificar el uso de la sentencia for, pues reconocieron su utilidad, y aquellas situaciones en las cuales sería necesario su uso. Las mayores dificultades fueron de tipo sintáctico. El paso inductivo, el ejemplo planteado, permitió que los alumnos generalizaran el concepto sin mayores dificultades, aunque el funcionamiento interno de la sentencia resultó no trivial.

Actividad Contador

```
(1) int y;
  (2) int CANTIDAD=0;
  (3) for (int i = 0; i<10 ; i++ ) {
  (4) System.out.println("Ingrese un número");
  (5) y=leer.nextInt();
  (6) if (y>0) CANTIDAD++; }
  (7) System.out.println("El número almacenado en cantidad es: "+cantidad);
```

Ejecutado el programa, vimos que no fue trivial identificar que la variable CANTIDAD se comporta como un contador. A fin de guiar hacia la identificación de su funcionamiento, y dado que esto no se logró de inmediato, sugerimos a los estudiantes los valores que debían ingresar, los cuales se registraron en el pizarrón. Luego de varias ejecuciones los alumnos pudieron inferir que la variable cantidad se comportaba como un contador, y no acumulador, concepto que debimos definir.

Pudimos observar algunas dificultades con relación al trabajo con contadores: la inclusión de varias sentencias distintas genera una dificultad importante para los alumnos; si a la variable se le asigna el valor 1 se genera confusión, pues se tienden a mezclar los conceptos de acumulador y contador; y por último la inicialización del contador (2) resulta no trivial, es necesario insistir en su necesidad y utilidad. Desde esta perspectiva podemos inferir que la actividad, aunque limitada a la presentación de un contenido, resultó acotada como para generalizar el concepto.

Actividad do-while

```
(1) Scanner leer = new Scanner(System.in);
  (2) int y;
  (3) do{
  (4) System.out.println ("Ingrese un número");
  (5) y=leer.nextInt();
  (6) }while (y<=8);
  (7) System.out.println("fin");
```

El abordaje de este contenido, al igual que los otros, es típicamente inductivo, desde lo particular a lo general, del ejemplo concreto a la generalización de procedimientos y conceptos. Ejecutado el programa, pedimos a los estudiantes que ingresaran datos, sin presentar guía alguna. Mientras algunos alumnos identificaron claramente una condición de salida del programa, otros no reconocieron que la salida se debió al dato que ingresaron previamente. Luego de una primera discusión grupal, la puesta en común consistió en guiar a los alumnos al ingreso de datos, indicando explícitamente los números que debían ingresar. Realizadas varias ejecuciones llegamos a identificar que mientras el número ingresado fuese menor o igual a 8 se repetía la solicitud de datos.

El trabajo con esta sentencia se mostró más difícil que en el caso de for, donde el manejo interno de la condición está implícito y se caracteriza por un número de iteraciones previamente conocido. En el caso de la sentencia do-while, se observaron tres dificultades: reconocer la necesidad de usar una sentencia iterativa, al igual que en el for, se convierte en un obstáculo importante para el estudiante; identificar la existencia de una condición que determina la posibilidad de iterar y acota el número de repeticiones, a priori desconocida; el manejo de la condición de salida del ciclo, la cual debió ser presentada como una condición lógica, genera una dificultad semánticas y sintácticas asociadas a la lógica proposicional.

Resultados obtenidos

El uso con casos particulares no es sinónimo de utilizar métodos inductivos, pues las particularidades nos pueden llevar a procesos solamente asociativos. En efecto, si consideramos la actividad Variables, podemos ver que los alumnos pueden asociar que al modificar el valor de la variable el resultado en pantalla también se modifica. Sin embargo, esto último no es lo mismo que aprehender el concepto de variables, en el sentido de apropiarse de él, ya que no implica que el alumno reconozca su importancia o sus posibles usos. Es así que debe diferenciarse un proceso inductivo como tal, que pretende lograr una generalización, de una asociación, la cual se limita a identificar los patrones en común.

Para que los alumnos puedan generalizar el concepto o procedimiento, es preciso guiarlos, mediante actividades debidamente organizadas, que permitan un proceso inductivo. En efecto, el profesor deberá partir del supuesto de que no es suficiente que el alumno reconozca lo que ha sucedido, sino que deberá lograr la extrapolación y generalización de conceptos o procedimientos. Este aspecto es particularmente delicado, pues una guía inadecuada puede dar lugar a conclusiones o generalizaciones parcial o completamente erróneas, lo que generará inconvenientes posteriores de difícil corrección. Es así que cuando las conclusiones a las que llegan los estudiantes, en el transcurso de la clase, "no son completamente correctas... [los estudiantes] deben saber que no lo son, pueden entonces anotarlas como "conclusiones provisorias", que estarán sujetas a revisiones" (Lerner,1996, p. 108), lo que permitirá retomar el contenido posteriormente.

De este modo, la actividad se convierte en un disparador, que permite que el estudiante analice, discuta, reorganice su estructura cognitiva, para "que se coordinen progresivamente los diferentes puntos de vista, que se vaya construyendo un saber común" (Lerner, 1996, p. 102).

Presentamos a continuación algunas preguntas que permiten la construcción o generalización de conceptos o procedimientos, las cuales están relacionadas con las actividades planteadas:

- ¿Qué utilidad tiene el signo + dentro del System.out.println?
- ¿Por qué podríamos querer almacenar un dato en una variable?
- Explique con sus palabras la utilidad de la sentencia if
- ¿Cómo haría para, dada una edad, determinar si la persona tiene más de 40 años? ¿Cómo sabemos que debemos usar una sentencia for?
- Presente un ejemplo en el cual se puede usar una sentencia for, que no implique mostrar un texto en pantalla.
- ¿Qué significa que una variable es un contador?
- ¿Qué podría pasar si no se inicializa el contador?
- ¿Cómo nos damos cuenta si debemos usar while o for?

Cada una de las preguntas indicadas permite que el alumno, a través de la negociación de significados, la prueba y los procesos inductivos, generalice el concepto o procedimiento asociado a ella.

Presentamos las peguntas realizadas y su incidencia en el trabajo, haciendo la aclaración de que no todos los contenidos del curso se han trabajado inductivamente, por lo que los alumnos no se han enfrentado solamente a la dicotomía entre deducción e inducción.

¿Resulta útil presentar un ejemplo en la computadora para luego explicar su utilidad?

En sentido estricto, esta pregunta no implica la preferencia por métodos inductivos o deductivos de los estudiantes. Sin embargo, el trabajo que se haga con el ejemplo y su posterior generalización podrían interpretarse como una predisposición del estudiante hacia un tipo de estrategia en particular.

El 77% de los estudiantes consideró beneficioso comenzar el trabajo a partir de un contenido presentando y explicando un ejemplo del mismo, para luego explicar las características generales del contenido. Como dijimos, el accionar docente influye fuertemente en el hecho de si esto debe considerarse como inducción o no. La ejecución y el análisis de programas están asociados a casos particulares, los cuales posteriormente son generalizados y discutidos, con distinto grado de éxito. Sin embargo, en nuestro caso el trabajo no se limitó a ejecutar el programa propuesto, o explicar línea a línea de código, sino que procuramos intercalar ambas, analizando el caso particular, la utilidad del contenido trabajado, para posteriormente abordar la utilidad y características del contenido. Cabe acotar que la presentación de un ejemplo no necesariamente estará unida a la ejecución en computadora de un programa.

El 23% de los estudiantes consideraron que esta estrategia no era ni mejor ni peor que comenzar explicando la utilidad del contenido; las respuestas se agruparon como "más o menos".

¿Se entiende mejor a partir de un ejemplo en la computadora?

Esta segunda pregunta presenta similitudes con la anterior, a la vez que algunas diferencias. Por un lado podríamos deducir que en caso de responder Si, el alumno prefiere trabajar los conceptos de programación codificando el algoritmo. Al igual que indicamos antes, estas respuestas no necesariamente tienen relación con métodos inductivos o deductivos. Sin embargo, debemos suponer que el ejemplo es un caso particular de la realidad, en el cual la posible generalización da lugar al caso general, trabajando así inductivamente. En nuestro caso el uso sistemático de la computadora tuvo dos motivos: asegurarnos de acercar al alumno al mundo real de la programación, y presentar ejemplos y actividades que permitan una posterior generalización, para lograr que los alumnos observen y deduzcan efectivamente el resultado de los programas. Nuestra forma de trabajo, una vez aplicado, explicado y analizado el ejemplo, tuvo como objetivo la generalización del concepto o procedimiento, por lo que caería dentro de las estrategias inductivas de trabajo, y las potenciaría.

El 77% de los estudiantes fueron explícitos en el sentido de preferir abordar un tema desde la ejemplificación en la computadora, mientas que el resto presentan otras respuestas, no necesariamente opuestas. Este resultado puede ser relativizado, pues desde nuestro accionar docente hemos procurado intensificar el trabajo en computadora, haciéndolo práctico, lo que puede ser considerado positivo por parte del alumno por el solo hecho de que lo hace el docente.

Al mismo tiempo, cabe indicar que algunos alumnos consideraron que si bien resulta útil presentar un ejemplo en la computadora (primera pregunta), esto no significa que se entiende mejor a partir de un ejemplo (segunda pregunta). Estos alumnos estarían afirmando que si bien el uso de la computadora y un ejemplo permitirían potenciar sus aprendizajes, esta estrategia no siempre sería la adecuada (o tal vez en la mayoría de los casos), dependiendo del contenido o del análisis que se haga del ejemplo.

¿Es mejor presentar un tema de forma "teórica" o a partir de un ejemplo?

En el momento de presentar esta pregunta fuimos explícitos en indicar que, tal cual lo consideran en general los estudiantes, presentar un tema "teórico" será sinónimo de presentar sus características constitutivas en lugar de la aplicación o el uso concreto del contenido.

La mayoría de los alumnos, casi el 70%, indicaron que prefieren la presentación del contenido desde un ejemplo. A diferencia de la pregunta anterior, en esta no se hizo referencia al uso de la computadora, por lo que el ejemplo se puede trabajar sin necesidad de codificación alguna. La dificultad en el aprendizaje del lenguaje de trabajo, del paradigma de programación o de la práctica por parte de

los alumnos hará que estos opten por el trabajo a partir de un ejemplo, con o sin codificación. Al igual que en la pregunta anterior, el trabajo con el ejemplo, más allá del uso o no de la computadora, hará que el proceso se pueda definir como principalmente inductivo.

¿Cuál tema fue el más difícil?

Dentro de los contenidos considerados como los más difíciles por parte de los estudiantes, se destacan el uso de las sentencias while, switch y los conceptos y operaciones sobre vectores y métodos. En el caso particular del uso del while, este tiene una gran similitud con do-while, sin embargo resulta difícil para los alumnos. Esta dificultad se observó al analizar los trabajos obligatorios de fin de curso.

Para los contenidos indicados se presentó un gran número de ejercicios para resolver, en forma de repartidos prácticos que posteriormente se trabajaron en clase, como el resto de los contenidos del programa. No obstante el énfasis en las actividades prácticas, estos temas fueron identificados como los más difíciles del programa. Al mismo tiempo es de destacar que el abordaje de estos contenidos fue principalmente deductivo: primero se presentaron su utilidad y sus características generales, más que desde el caso particular, el ejemplo y su posterior generalización, lo cual en ningún caso implicó la ausencia de trabajo práctico. La aplicación de los contenidos a casos concretos y particulares, se realizó con posterioridad a la presentación de las generalidades, propiedades o características del concepto o procedimiento.

Conclusiones

Si bien la población estudiada en el presente trabajo fue acotada, y no necesariamente es representativa del estudiantado que cursa asignaturas como Programación I, los resultados sugieren algunas líneas futuras de investigación y desarrollo, amén de las conclusiones parciales y acotadas al grupo. A modo de síntesis podemos decir que los estudiantes valoran y aprecian el trabajo con ejemplos y métodos inductivos, elementos estos que se ven potenciados por el uso de la computadora como herramienta.

El trabajo con métodos inductivos, entendidos como aquellos que abordan el problema desde los casos particulares para luego proceder a generalizar, parece ser efectivo y aceptado por los alumnos, quienes prefieren esta forma de trabajo frente a las estrategias principalmente deductivas, que suelen resultar de más difícil comprensión que aquellos trabajados inductivamente.

El uso adecuado de los ejemplos y su delimitación constituyen el elemento indispensable de las llamadas estrategias inductivas. Estas actividades deben estar debidamente acotadas, a fin de no generar una sobrecarga cognitiva, al tiempo que deben permitir la generalización de conceptos y procedimientos, y no solo un proceso asociativo. Este aspecto se muestra indispensable para el éxito de este tipo de estrategias de aula.

Los estudiantes aceptan y valoran el ejemplo, trabajado desde la codificación de la solución, y lo prefieren frente al trabajo sin computadora. Podemos concluir, al menos para el grupo analizado, que el trabajo alejado de la computadora y la implementación de la solución resulta ineficiente. La visualización del funcionamiento del ejemplo podría fortalecer las estrategias inductivas.

Referencias bibliográficas

Antúnez, S.; Del Carmen, L.; Imbernón, F.; Parcerisa, S. & Zabala, A. (1992). *Del proyecto educativo a la programación de aula* (2.ª ed.). España: Graó.

Barchini, G.; Sosa, M.; Herrera, S. (2004). La informática como disciplina científica. Ensayo de mapeo disciplinar. *Revistade Informática Educativa y Medios Audiovisuales, 1*(2), 1-11.

Cañadas, M. (2002). *Razonamiento inductivo puesto de manifiesto por alumnos de secundaria.* (Trabajo de investigación tutelada). Recuperado de: http://core.kmi.open.ac.uk/download/pdf/12340977.pdf.

Cuevas, R.; Bautista, H. & Medina, J. (2013). Propuesta metodológica para la enseñanza de la programación en la Unidad Académica de Ingeniería de la Uagro, *Revista vínculos,10*, 105-118. Recuperado de: http://revistavinculos.udistrital.edu.co/.

Deitel, P. & Deitel, H. (2007). *Java™ Cómo programar*. (7.ª ed.). México: Pearson Educación.

Díaz, J. (2006). Enseñando programación con C++: una propuesta didáctica. *Revista de Informática Educativa y Medios Audiovisuales, 3*(7), 12-21. Recuperado de: http://www.portalhuarpe.com/Medhime20/expo_americo/2003/articulos.htm.

Ebrahimi, A. (1994). Novice programmer errors: language constructs and plan composition. *International Journal of Human-Computer Studies,41*, 457-480. Recuperado de: http://www.sci.brooklyn.cuny.edu/~kopec/research/sdarticle1.pdf.

Fernández, L.; Peña, R.; Nava, F. & Velázquez A. (2002). Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos. Recuperado de: http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2002/Cac457_464.pdf.

Ferreira, A. & Rojo, G. (2006). Enseñanza de la programación. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología, 1*(1). Recuperado de: http://teyet-revista.info.unlp.edu.ar/numero-1.htm.

Gotschi, T.; Sanders, I. & Galpin, V. (2003). *Mental models of Recursion*. Recuperado de: http://www.it.uu.se/edu/course/homepage/datadidaktik/ht06/teaching/finalseminar/p346-gotschi.pdf.

Hernández, R.; Fernández, C. & Baptista, P. (1991). *Metodología de la Investigación.* (5.ª ed.). México: McGraw-Hill.

Houssaye, J. (1992). *Le triangle pédagogique* (2.ª ed.). Berna: Peter Lang.

Lerner, D. (1996). La enseñanza y el aprendizaje escolar. Un alegato contra una falsa oposición. En J. Castorina, E. Ferreiro, M. Kohl & D. Lerner. *Piaget-Vigotsky: contribuciones para replantear el debate* (pp. 69-118). Argentina: Paidós.

Llorens, F.; Satorre, R. & Puchol, J. (1996). Enseñar programación en las Ingenierías Informáticas. *II Jornadas Nacionales de Innovación en las Enseñanzas de las Ingenierías: resúmenes de comunicaciones, 2*, 840-847. Recuperado de http://www.dccia.ua.es/~faraon/docs/programacion.pdf.

Míguez, M. (2007). ¿Cómo motivar para aprender? En E. Fiore & J. Lymonié (ed.). *Didáctica práctica para enseñanza media y superior* (pp. 309-340). Montevideo: Magró.

Pérez, I.; Fuentes, A. & Moreno, S. (2008). *Estudio de la problemática presente en el diseño de algoritmos por computadora*. Recuperado de http://repository.uaeh.edu.mx/bitstream/handle/123456789/7943.

Real Academia Española. (2015). Recuperado de: http://www.rae.es/.

Rodríguez, R. & Valero, M. (1998). La gramática para comunicar: una propuesta inductiva. Recuperado de http://cvc.cervantes.es/ensenanza/biblioteca_ele/asele/pdf/09/09_0436.pdf.

Saavedra, J. & Silveira, A. (2011, junio). Algunas dificultades en el aprendizaje del concepto de variable. *Reporte Técnico RT11-07 (Pedeciba Informática-Instituto de Computación-Facultad de Ingeniería-Universidad dela República)*, 1- 9. Disponible en: http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR1107.pdf.

Villalobos, J. (2009). Proyecto cupi2 - una solución integral al problema de enseñar y aprender a programar. Recuperado de: http://www.colombiaaprende.edu.co/html/mediateca/1607/articles-205832_recurso_1.pdf.