EJEMPLIFICACIÓN DE LA SOLUCIÓN ALGORÍTMICA DE PROBLEMAS DE PROGRAMACIÓN COMPUTACIONAL

LA ALGORITMIZACIÓN DE PROBLEMAS DE PROGRAMACIÓN COMPUTACIONAL

AUTORES: Antonio Salgado Castillo[1]
Isabel Alonso Berenguer[2]
Alexander Gorina Sánchez[3]

DIRECCIÓN PARA CORRESPONDENCIA: Facultad de Matemática y Computación. Universidad de Oriente. E-mail: asalgado@csd.uo.edu.cu

Fecha de recepción: 06 - 08 - 2014 Fecha de aceptación: 11 - 09 - 2014

RESUMEN

Se expone una ejemplificación de la solución algorítmica de dos problemas de programación computacional, que se desarrolla a partir de un sistema de procedimientos didácticos creado por los propios autores. El citado sistema se sustenta en un modelo de la dinámica lógico–algorítmica de la resolución de problemas de programación computacional. La ejemplificación se distingue por simular el proceso de resolución llevado a cabo en una clase práctica, empleando pseudocódigos para realizar una algoritmización computacional de situaciones problémicas. Este proceso se conduce por el docente a través de los cuatro procedimientos del sistema, que son la construcción lógico–matemática, la orientación matemático–algorítmica, la estructuración algorítmico–generalizadora y la validación algorítmico–computacional. En cada uno de los procedimientos se simulan las acciones que deben desarrollar el profesor y el estudiante, con el propósito de que sirva de guía a profesores noveles de Programación para preparar y desarrollar sus clases prácticas. La pertinencia y viabilidad de esta forma de enseñar la algoritmización computacional se ha corroborado mediante el desarrollo de un experimento pedagógico con estudiantes de la carrera de Telecomunicaciones y Electrónica de la Universidad de Oriente, Santiago de Cuba. Se concluyó que el sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas, influye positivamente en el aprendizaje de la algoritmización para la resolución de problemas de programación computacional.

PALABRAS CLAVE: sistema de procedimientos didácticos; algoritmización computacional; situación problémica.

[1] Licenciado en Ciencia de la Computación. Doctorante en Ciencias Pedagógicas. Profesor Asistente. Departamento de Ciencia de la Computación. Facultad de Matemática y Computación. Universidad de Oriente. Cuba.
[2] Licenciada en Matemática. Doctora en Ciencias Pedagógicas. Profesora Titular. Departamento de Matemática. Facultad de Matemática y Computación. Universidad de Oriente. Cuba. E-mail: ialonso@csd.uo.edu.cu
[3] Licenciado en Matemática. Doctor en Ciencias Pedagógicas. Profesor Auxiliar. Departamento de Contabilidad y Finanzas. Filial Universitaria Contramaestre. Universidad de Oriente. Cuba. E-mail: gorina@contre.sum.uo.edu.cu

EXEMPLIFICATION OF THE SOLUTION ALGORITHMIC OF PROBLEMS OF PROGRAMMING COMPUTACIONAL

ABSTRACT

It is presented the exemplification of the algorithmic solution of two problems of computer programming, which are developed from a didactics procedures system created by the authors. That system is based on a model of the logical-algorithmic dynamic for computer programming problems solving. The modeling is distinguished by simulating the resolution process conducted in a practical class, using pseudocode to perform a computational algorithmization of problematic situations. This process is conducted by the teacher through the four processes of the system, which are the logical-mathematical construction, the algorithmic-mathematical orientation, the algorithmic-generalizing structuring and the algorithmic-computational validation. In each of the procedures are simulated the actions that should be developed by the teacher and the student, with the purpose to guide novice Programming teachers to prepare and develop their practical classes. The relevance and feasibility of this form of teaching computational algorithmization has been corroborated by developing a pedagogical experiment with students of the career Telecommunications and Electronics, at the Oriente University, Santiago de Cuba. It was concluded that the didactics procedures system supported in the exemplification of algorithmic solutions, have positive influences in learning of algorithmization to solve computational programming problems.

KEYWORDS: didactics procedures system; computational algorithmization; problem solving.

INTRODUCCIÓN

La programación computacional es aquella parte de las ciencias computacionales que proporciona las herramientas necesarias para establecer la secuencia de instrucciones que deben ser aplicadas, empleando un lenguaje específico, para que el ordenador ejecute una tarea, previo análisis de las condiciones y requerimientos de la misma (Microsoft Student, 2009).

Consecuentemente, la actividad de programar requerirá del establecimiento de un conjunto de instrucciones ordenadas para que la computadora lleve a cabo una determinada tarea, de dónde se induce la necesidad de favorecer el desarrollo de las potencialidades de los estudiantes para diseñar, escribir, depurar y mantener el código fuente (instrucciones) de los programas computacionales. Dicho código debe ser escrito en un lenguaje específico, requiriendo de conocimientos de varias áreas, del dominio de un lenguaje, de algoritmos especializados y de la lógica formal, a partir de lo cual se podrán crear programas que exhiban el comportamiento deseado (Martinelli, O., 2006; De Lobos, M. E., 2010).

Así hablar de la programación computacional implica reconocerla como un proceso complejo y creativo, el que ha sido abordado desde una diversidad de

paradigmas que han generado numerosas propuestas y discusiones en la búsqueda de una forma óptima de enseñar a programar, que permita al alumno el desarrollo de sus potencialidades para utilizar un conjunto de abstracciones, interrelacionadas entre sí para la resolución de problemas (Chesñevar, C.I, 2001; Martinelli, O., 2006). Sin embargo, aún en la actualidad la complejidad que caracteriza a la programación computacional hace que al llevarla a cabo se confronten serias dificultades, siendo una de las principales la falta de éxito que tienen los estudiantes en el análisis y solución de situaciones problémicas empleando objetos computacionales (Ramírez, R.V., 1991).

Cabe precisar que a los efectos de la presente investigación la situación problémica es interpretada como una situación conformada por objetos reales y/o matemáticos, que han sido didácticamente tratados por el profesor para hacerlos asequibles, en dependencia del nivel de profundidad requerido por los estudiantes que se inician en el aprendizaje de la resolución de problemas de programación computacional. La misma adquiere la connotación de problema de programación computacional en la medida en que el estudiante asume su resolución y encamina sus esfuerzos a la interpretación matemática y computacional de la misma.

Desde esta perspectiva de análisis, se revela la necesidad de diagnosticar las insuficiencias que manifiestan los estudiantes en el análisis y solución de situaciones problémicas durante el proceso de enseñanza–aprendizaje de la Programación en las carreras de ciencias computacionales, tales como: Licenciatura en Ciencia de la Computación, Ingeniería Informática, Ingeniería en Control Automático e Ingeniería en Telecomunicaciones. Estas fueron evidenciadas en un diagnóstico realizado en los cursos 2003-2004 al 2011-2012 en la Universidad de Oriente, el que ha sido detallado en Salgado y otros (2013.a), revelándose las siguientes insuficiencias:

- Escasas destrezas para desarrollar procesos de abstracción y decodificación de situaciones problémicas, con el objetivo de modelarlas desde la programación computacional.
- Limitaciones en la utilización de estructuras computacionales, lo que afecta el correcto diseño e implementación de los programas.
- Imprecisiones en las soluciones computacionales que se dan a las situaciones problémicas, las cuales no siempre satisfacen las exigencias originales.

Estas insuficiencias en la apropiación de los contenidos de programación han sido también confirmadas a nivel internacional por destacados investigadores como Oviedo y Ortiz (2002), Ferreira y Rojo (2005), Guibert, Guittet y Girard (2005) y Faouzia y Mostafa (2007), quienes de una forma u otra reconocen que el proceso de enseñanza–aprendizaje de la resolución de problemas de programación computacional, tiene como centro de sus dificultades la algoritmización. Sin embargo, sus propuestas investigativas se centran en la

praxis de dicha programación, sin profundizar en la fundamentación teórica de la lógica del proceso de algoritmización.

Para dar solución a esta carencia teórica se profundizó en la contradicción que se manifiesta entre la modelación matemática y su sistematización algorítmica durante el proceso de resolución de una situación problémica, la que sirvió de sustento para modelar la dinámica lógico–algorítmica de la resolución de problemas de programación computacional, según se explicita en Salgado y otros (2013.b). El modelo de la citada dinámica está conformado por cuatro dimensiones, las que son expresión de sus movimientos internos y permiten revelar la transformación de dicho proceso. Dichas dimensiones son: la construcción lógico–matemática, la orientación matemático–algorítmica, la estructuración algorítmico–generalizadora y la validación algorítmico–computacional.

Consecuentemente, a partir de la citada modelación, se elaboró un sistema de procedimientos didácticos conformado por un conjunto de acciones, lógicamente estructuradas y ordenadas, que facilitan el desarrollo de la dinámica lógico–algorítmica de la resolución de problemas de programación computacional. El sistema se construyó a partir del método Sistémico-Estructural-Funcional, instrumentando la citada dinámica mediante un conjunto de procedimientos integrados, como puede verse en Salgado y otros (2014). Es por tanto un instrumento de intervención didáctica, que tiene como objetivo la orientación intencional a los profesores de la asignatura de Programación para la conducción del proceso de enseñanza–aprendizaje de la algoritmización durante la resolución de problemas de programación computacional, todo lo cual persigue elevar a niveles cualitativamente superiores la actividad formativa del futuro egresado de las ciencias computacionales.

Sin embargo, teniendo en cuenta que un sistema de procedimientos didácticos por sí sólo, no siempre es de total ayuda para los profesores noveles, el presente trabajo tiene como objetivo realizar una ejemplificación de la solución algorítmica de dos problemas de programación computacional, a partir del citado sistema de procedimientos didácticos, con el propósito de facilitar la comprensión de la forma de conducir la dinámica lógico–algorítmica de la resolución de dichos problemas de programación.

DESARROLLO

La ejemplificación del sistema de procedimientos didácticos se hará sobre la base de dos situaciones problémicas extraídas del contexto cercano a las esferas de actuación de los futuros profesionales de las carreras de las ciencias computacionales. En ambos casos se irá simulando el posible modo de actuar del estudiante y del profesor durante el proceso de solución de dichas situaciones, con la intención de que sirva de guía para futuros procesos resolutores.

Situación problémica 1: *Con el objetivo de mejorar las telecomunicaciones en la Universidad de Oriente, la dirección de informatización se propuso incluir en el año 2014 un nuevo servicio de conexión a Internet e intranet, usando tecnología inalámbrica (WiFi). El objetivo primario fue conectar las tres sedes de la universidad, Antonio Maceo Grajales, Julio Antonio Mella y Frank País García, para lo cual se necesitó comprar AP inalámbricos para redes WiFi. Es así que los especialistas se vieron ante la tarea de determinar qué tipos de AP comprar (según su alcance) y para ello necesitaron saber qué distancia existe entre las tres sedes. Además, al tener en cuenta que los comercializadores realizan una rebaja del 5% a cada dispositivo extra del mismo tipo que se les compre, se preguntaron si sería posible adquirir los tres AP necesarios al menor precio posible. Se pide diseñar un algoritmo que dé respuesta a la problemática anterior, conociendo las coordenadas (x1, y1), (x2, y2) y (x3, y3) de las tres sedes.*

Observación: esta situación problémica muestra un escenario aparentemente sencillo y fácilmente extrapolable al rol que debe desempeñar un profesional de las ciencias computacionales en la práctica, su transposición didáctico profesional busca despertar el interés del estudiante. Además, la misma está acorde a los conocimientos que deben tener los estudiantes que reciben por primera vez la asignatura de Programación en las carreras de ciencias computacionales, con lo cual se garantiza que los mismos ganen en seguridad y puedan auto–valorar adecuadamente sus capacidades y posibilidades resolutoras.

Procedimiento de la construcción lógico–matemática

Actuar del profesor: dar tiempo a que cada estudiante, de manera individual, trate de comprender la situación problémica y luego propiciar un primer intercambio grupal en el que discutan la misma, de modo que se favorezca la interacción de lo individual con lo colectivo en el proceso de aprendizaje.

Actuar del estudiante: analizar detalladamente la situación problémica que se describe y tratar de comprender cada uno de los elementos relevantes de la misma a partir de sus conocimientos matemáticos y del contexto profesional descrito, con lo que deberá identificar los objetos (algoritmo, coordenadas, distancia), así como sus características, llegando a concluir que las coordenadas dadas forman un triángulo y que en dependencia del tipo de triángulo será el ahorro que se obtenga por concepto de precio de los AP (equilátero rebaja de un 10%, isósceles rebaja de un 5% y escaleno sin rebaja).

Posteriormente intentará expresar en lenguaje matemático cada uno de los objetos, características y relaciones relevantes que identifique en dicha situación y determinar las condiciones y exigencias, con lo que podría obtener que las condiciones del problema están dadas por las coordenadas de los tres vértices de un triángulo: Vértice_1 (x1, y1), Vértice_2 (x2, y2) y Vértice_3 (x3, y3) y que la exigencia consiste en determinar si el mismo es escaleno, isósceles o equilátero.

De lo anterior se deriva que la interpretación de la situación problémica puede llevarlos a comprender que necesitan diseñar un algoritmo que permita, dados los vértices de un triángulo (x1, y1), (x2, y2) y (x3, y3), determinar si el mismo es isósceles, equilátero o escaleno.

Observación: el estudiante deberá recuperar de su base de conocimientos y experiencias la información concerniente a la definición de cada tipo de triángulo según la longitud de sus lados, es decir, deberá identificar que un triángulo es:

Escaleno: si las longitudes de los tres lados son diferentes.

Isósceles: si las longitudes de dos lados cualesquiera son iguales.

Equilátero: si las longitudes de los tres lados son iguales.

Actuar del profesor: debe inducir a los estudiantes a que realicen una representación gráfica de la situación problémica, empleando para ello los elementos básicos que haya podido interpretar de las condiciones y teniendo en cuenta la exigencia.

Actuar del estudiante: puede representar un triángulo con sus lados debidamente identificados, lo que debe ser análogo a lo que se muestra en la Figura 1.

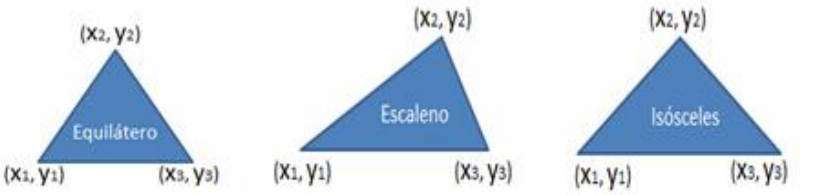

Figura .1: Representación geométrica del problema

Actuar del profesor: en este momento debe orientar a los estudiantes hacia la búsqueda de relaciones entre los datos que propician las condiciones, para obtener una visión de sistema de la situación problémica. Así, por ejemplo, puede propiciar que los estudiantes recuperen de su mente información acerca de cómo se calcula la distancia entre dos puntos, enfatizando además en la importancia de obtener las longitudes de los tres lados del triángulo, para facilitarles el establecimiento de comparaciones que les permitan determinar el tipo de triángulo.

Actuar del estudiante: se espera que trate de representar las comparaciones de las tres longitudes o distancias de los lados del triángulo. Para esto deberá definir las variables que identificarán a esas tres distancias, lo que le permitirá el perfeccionamiento de la representación matemática inicial, que será posible volviendo sobre el análisis de la situación problémica para profundizar en ésta y enriquecer la interpretación que de ella se había hecho. La representación perfeccionada al considerar las tres distancias podría quedar como se muestra en la Figura 2.

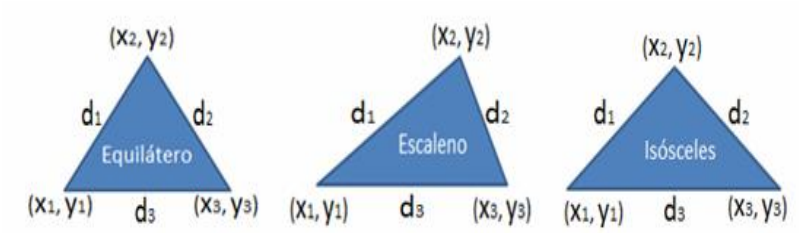

Figura 2: Representación geométrica transformada del problema

Al avanzar en la dinámica comparativa, animado por las preguntas e incentivos del profesor, podrá llegar a una representación más esencial, que es aquella que puede obtener a partir de objetos algebraicos, como se muestra en la Figura 3.

1. $d_1 = d_2 = d_3 \rightarrow$ Equilátero	2. $d_1 = d_2 \neq d_3 \rightarrow$ Isósceles
3. $d_1 \neq d_2 \neq d_3 \rightarrow$ Escaleno	4. $d_1 = d_3 \neq d_2 \rightarrow$ Isósceles
	5. $d_2 = d_3 \neq d_1 \rightarrow$ Isósceles

Figura 3: Representación algebraica del problema transformado

Actuar del profesor: a partir de aquí debe propiciar que el estudiante represente el problema de diversas formas, es decir, empleando objetos matemáticos de naturaleza geométrica y algebraica, entre otras. Para que pueda comparar dichas representaciones y seleccionar la que le resulte más fácil de resolver, de acuerdo a las condiciones de la situación problémica y a sus conocimientos matemáticos. Así deberá aprovechar este análisis para inducir la necesidad de tener en cuenta que mientras más objetos y relaciones matemáticas contenga la representación seleccionada, mayor cantidad de estructuras lógico–computacionales serán necesarias para crear el pseudocódigo.

Actuar del estudiante: debe llegar a plantear la fórmula de la distancia entre dos puntos (distancia euclídea): $d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$; $i < j$, para posteriormente relacionar este conocimiento matemático con la representación anterior y llegar a una representación algebraica como la siguiente:

$$d_1 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2},\ d_2 = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2},\ d_3 = \sqrt{(x_1 - x_3)^2 + (y_1 - y_{23})^2}$$

Observación: la relación entre este conocimiento matemático y la representación algebraica del problema, ilustrada en la Figura 3, puede ser considerada como la concreción del primer procedimiento y a la vez la base para iniciar el segundo.

El procedimiento de la orientación matemático–algorítmica

Actuar del profesor: propiciar el debate sobre las estructuras computacionales que pueden ser empleadas para representar los objetos y relaciones matemáticas contenidas en la Figura 3, llevando a que comprendan que la representación lograda plantea tres situaciones a analizar: cuándo el triángulo

es equilátero, cuando es escaleno y en el caso de que éste sea isósceles, para lo cual será necesario el cálculo de las distancias d1, d2 y d3.

También será importante que precise el papel que juegan los vértices del triángulo para la introducción de las estructuras lógico–computacionales, cuyas coordenadas se constituyen en condiciones o datos de entrada. Deberá enfatizar en la necesidad del uso de pseudocódigos como herramientas para la definición de las citadas estructuras.

Actuar del estudiante: tendrá que ir recuperando de su mente las estructuras algorítmicas necesarias para transformar los objetos y relaciones que aparecen en la representación matemática de la situación problémica (Figura 3). Se espera que identifique como una de las estructura el *"Si" (if),* teniendo en cuenta que tiene varias comparaciones que realizar entre las distancias de los lados del triángulo. No obstante, alguno podría identificar como estructuras a usar: *"Mientras" (while)* o *"Desde" (for).* Sin embargo, en el transcurso de la dinámica deberá llegar al convencimiento de que lo correcto es usar el *"Si"* pues, debido a las exigencias del problema y la representación realizada, no es necesario usar estructuras iterativas complejas como *"Mientras"* y *"Desde",* sino que solamente hay que realizar tres acciones fundamentales de comparación, las que por demás son distintas.

A continuación se supone que analice y concatene las estructuras computacionales de manera adecuada, en aras de formar una estructura funcional más esencial, a partir de lo cual podría obtener una estructura básica como se muestra en la Figura 4.

Entrar los vértices

Calcular d_1, d_2, d_3

Comparar usando el *Si,* las tres distancias

Si es escaleno *entonces* devolver **"Escaleno"**

Si es isósceles *entonces* devolver **"Isósceles"**

Si es equilátero *entonces* devolver **"Equilátero"**

Figura 4 Integración básica de estructuras lógico-computacionales

Observación: la integración básica de estructuras lógico–computacionales, ilustrada en la Figura 4, puede ser considerada como la concreción del segundo procedimiento y a la vez la base para iniciar el tercero.

El procedimiento de la estructuración algorítmico–generalizadora

Actuar del profesor: se asegurará de que los estudiantes realicen estructuraciones del algoritmo basadas en la integración obtenida, usando pseudocódigo y sistematizando las ventajas y desventajas de cada estructuración realizada. Para tales propósitos puede resultar provechoso el empleo de un software como el PSeInt, FreeDFP, RAPTOR, el Software para la enseñanza–aprendizaje de algoritmos estructurados o cualquier otro afín, que permita apoyar la dinámica resolutora.

Actuar del estudiante: se espera que el estudiante perfeccione la integración ilustrada en la Figura 4, detallando cada paso descrito en la misma y analizando las posibles estructuraciones que conlleven a la solución. En estos

momentos la estructuración algorítmica creada por el estudiante podría quedar como se ilustra en la Figura 5.

Si bien la estructuración mostrada en la Figura 5 es válida, podrían proponer otra como la que se ilustra en la Figura 6, utilizando los operadores lógicos (y/o).

Entrar los vértices (x_1, y_1) (x_2, y_2) (x_3, y_3)
Calcular d_1
$a_1 = (x_1 - x_2)\ (x_1 - x_2)$
$b_1 = (y_1 - y_2)\ (y_1 - y_2)$
$c_1 = a_1 + b_1$
$d_1 = \sqrt{c_1}$
Calcular d_2
$a_2 = (x_2 - x_3)\ (x_2 - x_3)$
$b_2 = (y_2 - y_3)\ (y_2 - y_3)$
$c_2 = a_2 + b_2$
$d_2 = \sqrt{c_2}$
Calcular d_3
$a_3 = (x_1 - x_3)\ (x_1 - x_3)$
$b_3 = (y_1 - y_3)\ (y_1 - y_3)$
$c_3 = a_3 + b_3$
$d_3 = \sqrt{c_3}$
Si $(d_1 \neq d_2)$ ***entonces***
{
Si $(d_1 \neq d_3)$ ***entonces***
{
Si $(d_2 \neq d_3)$ ***entonces***
{
"Escaleno**"**
}
}
Sino
{
"Isósceles**"**
}
Sino
{
Si $(d_1 == d_3)$ ***entonces***
"Equilátero**"**
Sino
"Isósceles**"**
}

Figura 5: Posible estructuración algorítmica

Entrar los vértices (x_1, y_1) (x_2, y_2) (x_3, y_3)
Calcular d_1
$a_1 = (x_1 - x_2)\ (x_1 - x_2)$
$b_1 = (y_1 - y_2)\ (y_1 - y_2)$
$c_1 = a_1 + b_1$
$d_1 = \sqrt{c_1}$
Calcular d_2
$a_2 = (x_2 - x_3)\ (x_2 - x_3)$
$b_2 = (y_2 - y_3)\ (y_2 - y_3)$
$c_2 = a_2 + b_2$
$d_2 = \sqrt{c_2}$
Calcular d_3
$a_3 = (x_1 - x_3)\ (x_1 - x_3)$
$b_3 = (y_1 - y_3)\ (y_1 - y_3)$
$c_3 = a_3 + b_3$
$d_3 = \sqrt{c_3}$
Si $(d_1 \neq d_2)$ y $(d_1 \neq d_3)$ y $(d_2 \neq d_3)$ ***entonces***
{
"Escaleno**"**
}
Sino
{
Si $(d_1 == d_2)$ y $(d_1 == d_3)$ y $(d_2 == d_3)$ ***entonces***
{
"Equilátero**"**
}
Sino
{
"Isósceles**"**
}
}

Figura 6: Otra posible estructuración algorítmica

Observación: en ambas estructuraciones, por lo general, los estudiantes realizan la tercera comparación de las distancias, es decir escriben una sentencia como la que se muestra a continuación:

Si $(d_1 == d_2)$ o $(d_1 == d_3)$ o $(d_2 == d_3)$ ***entonces***
{
"Isósceles"
}

Este paso es innecesario, puesto que si el triángulo no es equilátero ni escaleno, necesariamente tiene que ser isósceles.

Actuar del profesor: en este momento puede hacer preguntas sobre si es necesario o no hacer la última comparación y lo que implica computacionalmente el hecho de no hacerla, todo ello con el objetivo de ir formando en los estudiantes habilidades de optimización. Además pudiera introducir una modificación al problema, como por ejemplo la que consiste suponer que además de determinar el tipo de triángulo pueda determinarse el lado base (para el caso de ser isósceles). Aquí si sería necesaria la tercera comparación, pues haría falta saber cuáles son los lados iguales para concluir que el tercer lado es el lado base.

El procedimiento de la validación algorítmico–computacional

Actuar del profesor: una vez que ya se tiene estructurado el algoritmo que da solución a la situación problémica planteada inicialmente, debe hacer énfasis en que el estudiante domine las características del pseudocódigo utilizado, en relación con la definición del funcionamiento básico de las diferentes estructuras lógico–computacionales previamente identificadas e integradas. Todo ello con el propósito de que optimicen el pseudocódigo para elevar la eficiencia y eficacia de los resultados. También es conveniente que propicie la discusión de las soluciones propuestas, con el objetivo de perfeccionarlas y eliminar errores de sintaxis, comparaciones innecesarias o errores que afecten la semántica de la solución.

Observación: en este momento del proceso resolutor se deben valorar las dos estructuraciones propuestas en las Figuras 5 y 6, llevando al estudiante a determinar cuál ofrece mayores ventajas computacionales en cuanto a optimizar la memoria y el tiempo de ejecución del computador.

Actuar del estudiante: trabajará en el refinamiento sucesivo del algoritmo, a partir de tomar en cuenta la sintaxis o escritura del pseudocódigo. Esto le permitirá regular y evaluar sistemáticamente el proceso de algoritmización computacional. Deberá realizar una ejecución manual del algoritmo con datos completos, que abarquen todo el posible rango de valores, para comprobar que la salida coincide con lo esperado en cada caso. Se espera que verifique si el algoritmo contiene, con suficiente detalle, los pasos a realizar y el conjunto de palabras propias del pseudocódigo que ha sido previamente orientado. También podría rediseñar determinadas partes del algoritmo sobre la base de valoraciones realizadas a otras estructuraciones obtenidas en la dinámica resolutora.

Actuar del profesor: hacer notar que en la estructuración que ilustra la Figura 5 no es necesaria la comparación “Si (d2 ≠ d3) entonces” y en la Figura 6 cuando se emplea “Si (d1 ≠ d2) y (d1 ≠ d3) y (d2 ≠ d3) entonces”, ésta sobra.

Actuar del estudiante: se espera que el estudiante arribe a una versión más refinada, como la que se ilustra en la Figura 7.

Actuar del profesor: inducir un nuevo refinamiento de la solución pidiendo a los estudiantes que profundicen en las sentencias dónde se determina que el triángulo es equilátero (Figura 8). Esto puede lograrlo a partir de preguntas tales como: ¿Creen ustedes que será necesario preguntar si (d2 == d3)?, ¿Se obtiene el mismo resultado si se elimina esa comparación? ¿Cuál es la implicación, en términos computacionales, de eliminar la citada sentencia?

Entrar los vértices (x_1, y_1) (x_2, y_2) (x_3, y_3)
Calcular d_1
$a_1 = (x_1 - x_2)\ (x_1 - x_2)$
$b_1 = (y_1 - y_2)\ (y_1 - y_2)$
$c_1 = a_1 + b_1$
$d_1 = \sqrt{c_1}$
Calcular d_2
$a_2 = (x_2 - x_3)\ (x_2 - x_3)$
$b_2 = (y_2 - y_3)\ (y_2 - y_3)$
$c_2 = a_2 + b_2$
$d_2 = \sqrt{c_2}$
Calcular d_3
$a_3 = (x_1 - x_3)\ (x_1 - x_3)$
$b_3 = (y_1 - y_3)\ (y_1 - y_3)$
$c_3 = a_3 + b_3$
$d_3 = \sqrt{c_3}$
Si $(d_1 \neq d_2)$ ***entonces***
{
Si $(d_1 \neq d_3)$ ***entonces***
{
"Escaleno**"**
}
}
Sino
{
"Isósceles**"**
}
Sino
{
Si $(d_1 == d_3)$ ***entonces***
"Equilátero**"**
Sino
"Isósceles**"**
}
}

Figura 7: Estructuración más esencial respecto a la representada en la Figura 6

Entrar los vértices (x_1, y_1) (x_2, y_2) (x_3, y_3)
Calcular d_1
$a_1 = (x_1 - x_2)\ (x_1 - x_2)$
$b_1 = (y_1 - y_2)\ (y_1 - y_2)$
$c_1 = a_1 + b_1$
$d_1 = \sqrt{c_1}$
Calcular d_2
$a_2 = (x_2 - x_3)\ (x_2 - x_3)$
$b_2 = (y_2 - y_3)\ (y_2 - y_3)$
$c_2 = a_2 + b_2$
$d_2 = \sqrt{c_2}$
Calcular d_3
$a_3 = (x_1 - x_3)\ (x_1 - x_3)$
$b_3 = (y_1 - y_3)\ (y_1 - y_3)$
$c_3 = a_3 + b_3$
$d_3 = \sqrt{c_3}$
Si $(d_1 \neq d_2)$ y $(d_1 \neq d_3)$ ***entonces***
{
"Escaleno**"**
}
Sino
{
Si $(d_1 == d_2)$ y $(d_1 == d_3)$ y $(d_2 == d_3)$
entonces
{
"Equilátero**"**
}
Sino
{
"Isósceles**"**
}
}

Figura 8: Estructuración más esencial respecto a la representada en la Figura 7.

Ahora bien, para concluir la solución de la situación problémica planteada debe inducir el análisis de la rebaja del precio de los AP, a partir del tipo de triángulo obtenido con el algoritmo.

Observaciones finales: la situación problémica puede explotarse más introduciendo elementos de análisis que complejicen su algoritmización, tales como pedir a los estudiantes que:

- Contemplen en el algoritmo la verificación de que las coordenadas dadas forman un triángulo.
- Incluyan en el algoritmo la determinación del porciento de rebaja del precio de los AP, según el tipo de triángulo.
- Trabajar con los valores de los precios de los AP según su alcance para determinar el importe total.

Situación problémica 2: *Considere una estación de servicios de correos electrónicos mediante acceso remoto con un único servidor. A la misma llegan peticiones provenientes de cuatro estaciones de trabajo. El servidor sólo atiende un cliente en cada momento y cuando termina se ocupa del que ha estado esperando más tiempo. Los clientes de cada estación pueden solicitar el acceso simultáneamente. Se requiere determinar todas las formas posibles de organizar la atención a los clientes cuando hay peticiones simultáneas, de manera tal que los mismos disfruten de iguales privilegios. Se pide crear un algoritmo para dar solución a esta situación.*

Observación: al igual que en la primer situación, la presente está relacionada con problemas inherentes al quehacer de los profesionales de las ciencias computacionales y su solución requiere de conocimientos que son asequibles a los estudiantes a los que va dirigida esta práctica docente.

El procedimiento de la construcción lógico–matemática

Actuar del profesor: esperar que cada estudiante, de manera individual, se esfuerce por comprender la situación problémica y luego propiciar un primer intercambio grupal en el que discutan la misma, de modo que se favorezca la interacción de lo individual con lo colectivo en este proceso de aprendizaje.

A tales fines deberá hacer preguntas que lleven a que los estudiantes exploren la situación problémica, analizando cuidadosamente sus elementos y componentes, con la intención de crear patrones de búsqueda de una vía de solución. Debe conducir el proceso a que los estudiantes concluyan que en este caso los objetos dados son las estaciones de trabajo, el servidor y los clientes.

Actuar del estudiante: debe analizar detalladamente el texto de la situación problémica y tratar de comprender cada uno de los elementos relevantes de la misma, pues de esa comprensión dependerá que tenga éxito su resolución. Para la citada comprensión necesitará de conocimientos computacionales y del contexto, para interpretar la información contenida en el mismo. Se espera que los estudiantes imaginen las cuatro estaciones de trabajo como cuatro ordenadores conectados a un servidor principal. Además de que consideren que se hace una cola en espera de ser atendido, por lo que puede suceder que el mismo cliente siempre se atienda primero o al mismo cliente siempre le toque esperar. Llegando a la conclusión que lo que se necesita es hallar todas las

posibles formas de organizar o permutar los clientes para que todos puedan ser atendidos con igual prioridad.

Posteriormente, deberá tratar de expresar en lenguaje matemático cada uno de los objetos y relaciones que identifique en dicho texto, tratando de asociar una variable matemática a cada una de las actividades que conforman la situación, identificando las condiciones y las exigencias, con lo que podría obtener que las condiciones del problema están dadas por los cuatro objetos (en este caso clientes). Además de que la exigencia del problema consiste calcular el factorial de 4.

Actuar del profesor: en este momento el profesor debe conducir a los estudiantes hacia la búsqueda de las relaciones entre los datos, para obtener una visión de sistema de la situación problémica, que incida en el desarrollo del pensamiento algorítmico, permitiendo a los mismos encontrar mayor cantidad de implicaciones, nexos y relaciones dentro de la información, favoreciendo su visión sobre ésta. Así mismo, debe propiciar que recuperen de su mente vías para el cálculo del factorial de un número.

Lo anterior permitirá al estudiante ir identificando las características de los objetos (matemáticos, reales, computacionales, entre otros) que están o podrían estar presentes en el proceso de resolución de la situación problémica. Ello favorecerá el desarrollo de habilidades para seleccionar correctamente los objetos a utilizar, de acuerdo a las características identificadas.

Actuar del estudiante: debe tratar de recuperar de su mente los conocimientos previos sobre el concepto de factorial de un número y concluir que es la productoria de todos los números menores que el número dado y que además el factorial de 0 es igual 1 por definición.

Actuar del profesor: puede hacer énfasis en la importancia de representar u obtener los números menores que el número dado, puesto que harán falta para el cálculo del citado factorial.

Actuar del estudiante: tratar de representar matemáticamente el cálculo del factorial de un número n cualquiera (en este caso 4). Para esto, en primer lugar deberá definir la variable que identificará el factorial y la definición del factorial de 0 como caso particular. También será necesario contemplar una variable para ir multiplicando en cada paso los números menores que n. Esto permitirá el perfeccionamiento de la representación matemática inicial.

Observación: el citado perfeccionamiento será posible volviendo sobre el análisis de la situación problémica, lo que permitirá profundizar en ésta y enriquecer la interpretación que de ella se había hecho. Esto podrá hacerse colectivamente, para que se transmitan patrones, estrategias y métodos. Finalmente podría quedar así:

1. *n:* variable que representa el número al que se le calculará el factorial.
2. *n!:* Forma de representar matemáticamente el factorial de un número.

3. Por definición, *0! = 1*.
4. Multiplicador: variable que tomará el valor de cada número menor que n, que se necesite multiplicar.

Actuar del profesor: una vez obtenidas esas diversas representaciones matemáticas debe facilitar que el estudiante analice cada una de ellas y las compare, para seleccionar la que le resulte más fácil de resolver, de acuerdo a las condiciones de la situación problémica y a sus conocimientos matemáticos. Aquí se debe aprovechar este análisis para insistir en que, mientras más objetos y relaciones matemáticas contenga la representación seleccionada, mayor cantidad de estructuras lógico–computacionales serán necesarias para crear el pseudocódigo.

Actuar del estudiante: se confía en que el estudiante plantee la fórmula para determinar el factorial de n. Posteriormente, deberá tratar de expresar en lenguaje matemático cada uno de los objetos y relaciones que identifique en dicho texto, tratando de asociar una variable matemática a cada una de las actividades que conforman la situación problémica, además de la concepción de abstracciones sustentadas en objetos y relaciones matemáticas para obtener una interpretación matemática de la situación problémica como se muestra a continuación. Esto podrá lograrlo a partir de asociar a cada parte del objeto simplificado, objetos y relaciones matemáticas que lo identifiquen, de manera que obtenga una primera representación matemática de la situación problémica como la que se muestra a continuación:

1. *n! = n * (n–1) * (n–2) *…* 1* (se calcula el factorial como el producto) (representación 1)
2. *n! = 1 * 2 *…* (n–1) * n* (se calcula el factorial como el producto) (representación 2)

Observación: aquí el estudiante podría realizar varias representaciones matemáticas de la situación problémica, lo que le permitirá establecer comparaciones entre el número de objetos utilizados, las relaciones establecidas y la intencionalidad deseada. Esto favorece la sistematización de los contenidos estudiados.

El procedimiento de la orientación matemático–algorítmica

Actuar del profesor: propiciar el debate sobre las estructuras computacionales desde una perspectiva que permita al estudiante reconocer las ventajas y desventajas de cada una, haciendo énfasis en que la representación lograda evidencia que el cálculo del producto requiere acumular las multiplicaciones de los valores *n, n–1, n–2,* etc. Debe hacer énfasis en la necesidad de leer primero el valor del número al que se le desea calcular el factorial, es decir, debe tener dicho número como entrada. Otro elemento importante es insistir en el uso del pseudocódigo como herramienta para la definición de las estructuras lógico–computacionales.

Actuar del estudiante: ir reconociendo las estructuras algorítmicas necesarias para transformar los objetos y relaciones que aparecen en la representación matemática de la situación problémica. Convencerse de la necesidad de usar estructuras iterativas como el *while*, *do while* o el *for*, debido a que hay que realizar la misma acción de multiplicación varias veces, de acuerdo a la representación matemática realizada y que por estas mismas razones no debe usar el *Si (if).*

Además deberá analizar y concatenar las estructuras adecuadamente para lograr niveles cada vez más esenciales y precisos de la representación matemática inicial desde de una perspectiva algorítmico–computacional. A partir de lo anterior se podría obtener una representación como la que se muestra en la Figura 9.

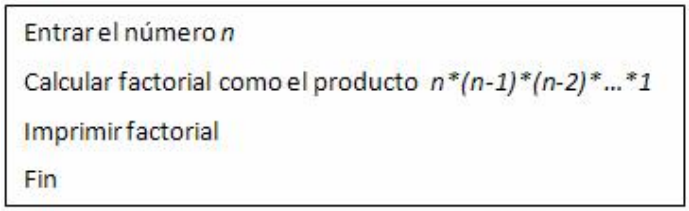

Figura 9 Integración básica de estructuras lógico-computacionales

Actuar del profesor: hacer énfasis en las ventajas y desventajas de cada estructura lógico–computacional. Pudiendo formular preguntas tales como: ¿Cuáles son las características de la instrucción *while*, *do while*, *for*? ¿Cuáles son sus ventajas? ¿Cuándo es mejor usarla? ¿Por qué? Explicar que se podrá desarrollar la solución usando el *while* o *do while* o el *for*.

El procedimiento de la estructuración algorítmico–generalizadora

Actuar del profesor: debe tener claro que en este momento ya el estudiante tiene suficientes elementos para representar la solución algorítmica correcta y detallada usando pseudocódigo, para lo cual, al igual que en el caso anterior, puede sugerir el empleo de un software como el PSeInt, FreeDFP, RAPTOR o el Software para la enseñanza–aprendizaje de algoritmos estructurados. El uso de estas herramientas favorecerá la creación y análisis de nuevas generalizaciones algorítmicas, además de desarrollar habilidades computacionales.

Actuar del estudiante: debe perfeccionar la representación anterior, detallando cada paso descrito en la Figura 9 y analizando las posibles estructuraciones que conlleven a la solución. Así podría obtener una estructuración algorítmica usando el *while* como se muestra en la Figura 10.

```
Estructuración usando while (Mientras Hacer)
Entrar el número n
factorial = 1 y multiplicador  = n
mientras (multiplicador >1)
        {
            factorial = factorial * multiplicador
            multiplicador  = multiplicador  – 1
        }
Imprimir factorial
Fin
```

Figura 10: Posible estructuración algorítmica usando el ***while***.

Actuar del profesor: a partir de la Figura 10, podría realizar algunas preguntas al estudiante sobre la estructuración, como por ejemplo: ¿Qué pasaría si se decrementa el multiplicador antes de multiplicar por el factorial? ¿Se obtendría el mismo resultado?

Es preciso explicar porqué no se debe decrementar el multiplicador antes de multiplicar por el factorial, ya que se estaría omitiendo la multiplicación por el número *n* dado. Esto suele ser un error común en este tipo de ejercicios.

Si bien, esta podría ser una estructuración válida, los alumnos podrían proponer otra utilizando el *do while* como se muestra en la Figura 11. Hacer énfasis en que esta estructuración puede traer inconvenientes computacionales, pues primero se realiza la acción y luego se verifica la condición, que aunque en este caso no ocurre, como regla general sólo debe usarse cuando se pueda asegurar que las acciones a realizar se ejecutaran al menos una vez. Con respecto a la estructuración anterior esto es una desventaja.

```
Estructuración usando do while (Hacer mientras)
Entrar el número n
factorial = 1
Si (n >1)
  {
     multiplicador = n
     Hacer
          {
             factorial = factorial * multiplicador
             multiplicador = multiplicador – 1
          }
     mientras (multiplicador >1)
  }
Imprimir factorial
Fin
```

Figura 11: Posible estructuración algorítmica usando el ***do while***.

Actuar del estudiante: puede obtener otra posible estructuración usando el *for*, como se muestra en la Figura 12. Por ejemplo, el cálculo del factorial de un número *n* es *1*2*…*n.*

Observación: es evidente que hay que hacer *n* multiplicaciones para obtener el factorial (o ninguna, teniendo en cuenta que *0!=1*).

Actuar del profesor: puede realizar preguntas tales como: ¿Qué pasaría si se inicializara el multiplicador en 0? ¿Cambiaría el resultado? ¿Qué se podría hacer para que no cambie el resultado? ¿Cuántas iteraciones se van a realizar ahora? Precisar que la variable de control, que es multiplicador, debe comenzar con el valor 1. Mientras ese controlador sea menor o igual a *n* (la cantidad de veces que el lazo debe iterar) se ejecutará el cuerpo del ciclo y se incrementará la variable de control. Hacer notar lo simple que es ahora la forma expresiva de la estructura para iterar. Finalmente, precisar que como el multiplicador comienza con valor 1 y termina con valor *n*, entonces el lazo se ejecuta *n* veces (si comenzara en 0 y debería terminar en el valor *n–1*, así se ejecutaría *n* veces).

```
Estructuración usando for (Desde Hasta)
Entrar el número n
factorial = 1
     Desde multiplicador = 1 Hasta n multiplicador + 1
        {
           factorial = factorial * multiplicador
        }
Imprimir factorial
Fin
```

Figura 12: Posible estructuración algorítmica usando el ***for***.

Sistematizar cuando es recomendable usar cada una de las estructuraciones. Por ejemplo, la última con el *for* es adecuada siempre que se sepa la cantidad de veces que se va a repetir una acción, mientras que las dos primeras son más eficaces cuando no se tiene conocimiento del número de iteraciones a realizar y sólo se conoce una condición de parada o finalización.

El procedimiento de la validación algorítmico–computacional

Actuar del profesor: observar que en este momento ya se tiene estructurado el algoritmo que da solución al problema planteado inicialmente, pero se debe hacer énfasis en que el estudiante domine las características del pseudocódigo utilizado, en relación con la definición del funcionamiento básico de las diferentes estructuras lógico–computacionales identificadas e integradas previamente. También debe retomar la importancia de optimizar el pseudocódigo para elevar la eficiencia y eficacia de los resultados. Se deben discutir en el aula las soluciones propuestas con el objetivo de perfeccionarlas y eliminar errores de sintaxis, comparaciones innecesarias o errores que afecten la semántica de la solución.

Observación: en este momento se deben valorar las tres soluciones propuestas buscando cuál ofrece más ventajas computacionales en cuanto a optimizar la memoria y el tiempo de ejecución del computador. Es conveniente que la solución final se ejercite en un laboratorio con el software seleccionado.

Actuar del estudiante: trabajar en el refinamiento del algoritmo, a partir de tomar en cuenta la sintaxis o escritura del pseudocódigo. Esto permitirá regular y evaluar sistemáticamente el proceso de algoritmización computacional. Comprender que la exactitud y precisión de un algoritmo determina la calidad de su ejecución, una vez implementado en un lenguaje de programación. Realizar una ejecución manual del algoritmo con datos significativos que abarquen todo el posible rango de valores para comprobar que la salida coincide con lo esperado en cada caso. Verificar que el algoritmo contenga los pasos a realizar, suficientemente detallados. Rediseñar determinadas partes del algoritmo.

Observación: en la estructuración usando el *do while*, por lo general los estudiantes no consideran dentro de la estructura del *Si* los elementos del *do while* y lo ponen al mismo nivel de identación o sangría.

Actuar del profesor: explicar la importancia de respetar la sintaxis aunque se esté trabajado en pseudocódigo y escribir las estructuras que estén contenidas dentro de otras con mayor nivel de identación.

Para concluir la solución de la situación problémica planteada, debe inducir el análisis del resultado obtenido con el algoritmo que es 24 = 4 * 3 * 2 * 1, es decir, debe pedir a los estudiantes que una vez determinado el número de permutaciones o maneras distintas de organizar la atención de los cuatro clientes, propongan como sería esta disposición.

Actuar del estudiante: se espera que en primera instancia se dé cuenta que al ser 24 permutaciones posibles, a cada cliente le corresponde ser el primero en atenderse 6 veces. Por lo que debe ir combinando todos los clientes hasta lograr el objetivo. En este momento se debe alcanzar una respuesta como:

Cliente 1 con prioridad: C1 C2 C3 C4; C1 C2 C4 C3; C1 C3 C2 C4; C1 C3 C4 C2; C1 C4 C2 C3; C1 C4 C3 C2

Cliente 2 con prioridad: C2 C1 C3 C4; C2 C1 C4 C3; C2 C3 C1 C4; C2 C3 C4 C1; C2 C4 C1 C3; C2 C4 C3 C1

Cliente 3 con prioridad: C3 C2 C1 C4; C3 C2 C4 C1; C3 C1 C2 C4; C3 C1 C4 C2; C3 C4 C2 C1; C3 C4 C3 C1

Cliente 4 con prioridad: C4 C2 C3 C1; C4 C2 C1 C3; C4 C3 C2 C1; C4 C3 C1 C2; C4 C1 C2 C3; C4 C1 C3 C2

Pertinencia y viabilidad del sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas

Para corroborar esta pertinencia y viabilidad se realizó un experimento pedagógico con dos grupos (experimental y control) pertenecientes a la carrera de Ingeniería en Telecomunicaciones y Electrónica, de la Universidad de Oriente, Cuba, que es una de las cuatro carreras de ciencias computacionales del centro.

Una vez seleccionada la carrera y el año, se diseñó y ejecutó un *experimento con preprueba–postprueba y grupo control* (ver Hernández, R., Fernández, C., Baptista, P., 1998, pp. 124–185). Este diseño incorporó la aplicación de prepruebas a los dos grupos que formaron parte el experimento (grupo control y experimental), ambos pertenecientes al primer año de la citada carrera. Cabe precisar que para conformar ambos grupos se utilizó el emparejamiento como técnica que permite lograr una equivalencia entre los mismos, pues los dos grupos ya estaban conformados a la hora de llevar a cabo el experimento y debían funcionar como tales.

Inicialmente a los estudiantes de ambos grupos se les aplicó simultáneamente la preprueba. Luego un grupo recibió el tratamiento experimental (grupo experimental) y otro no lo recibió (grupo control). Finalmente se les aplicó a ambos grupos una postprueba, también simultáneamente. El diseño empleado puede diagramarse como sigue:

EG_1	O_1	X Sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas	O_2
EG_2	O_3	--- Método Tradicional	O_4

E: emparejamiento o técnica de apareo (en inglés matching).
G_1, G_2: Grupo experimental y grupo control respectivamente.
O_i (i=1, 2, 3, 4): Una medición a los estudiantes de un grupo (prueba pedagógica). Si aparece antes del estímulo se trata de una preprueba, si es después una postprueba.
X: Condición experimental (aplicación del sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas). Presencia de algún nivel de la variable independiente.
--- Ausencia de estímulo (nivel cero en la variable independiente, lo que equivale a desarrollar la dinámica del proceso de enseñanza–aprendizaje por la vía tradicional). Indica que se trata de un grupo control.

El empleo de la preprueba ofreció dos ventajas:

- Sus puntuaciones se usaron para fines de control en el experimento. La comparación de las prepruebas de los dos grupos permitió evaluar qué tan adecuado fue el emparejamiento.
- Se pudo analizar el puntaje ganancia de cada grupo (la diferencia entre las puntuaciones de la preprueba y la postprueba).

Este diseño tuvo en cuenta las principales fuentes de invalidación interna, por lo que la administración de la prueba quedó controlada. Si la preprueba hubiese afectado las puntuaciones de la postprueba, los resultados de dicha afectación hubiesen sido similares en ambos grupos, de aquí que se siguiera cumpliendo con la esencia del control experimental. En general, durante la duración del experimento las variables que influyeron en un grupo también lo hicieron de la misma manera en el otro, salvo para el caso de la variable experimental, esto permitió mantener la equivalencia de los dos grupos.

Cabe señalar que durante la realización del experimento se controló:

- La historia como fuente de invalidación interna, pues no ocurrió ningún acontecimiento significativo que afectara a los grupos (control y experimental).
- La maduración, porque al existir un emparejamiento de los estudiantes de ambos grupos, la maduración esperada fue similar en los mismos.
- La inestabilidad en la aplicación de los instrumentos y en su medición, pues se aplicaron en iguales condiciones y tiempo y fueron calificados por un único profesor.
- La administración de pruebas, pues la preprueba no afectó las puntuaciones de la postprueba, ya que hubo suficiente tiempo entre la aplicación de las mismas.

A continuación se muestran los resultados obtenidos en cada una de las fases del experimento pedagógico desarrollado.

Emparejamiento y aplicación de la preprueba

Se aplicó la preprueba y se calificó en base a la escala ordinal (2, 3, 4, 5). Posteriormente, para cada uno de los 23 estudiantes de cada grupo (control y experimental) se ordenaron las calificaciones de menor a mayor y se procedió a realizar el emparejamiento (ver Tabla 13).

Est.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	PT
GExp.	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	4	5	**59**
GCon.	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	**59**

Tabla 13. Resultados de la aplicación de la preprueba y emparejamiento en los dos grupos (experimental y control).

Luego se pasó a analizar si existían diferencias significativas entre el grupo experimental y el grupo control respecto a su tendencia central, tomando como base las calificaciones obtenidas. A tales efectos se utilizó la Prueba no paramétrica U de Mann–Whitney (Siegel, S., 1972, pp. 143–155), que puede ser utilizada para al menos una escala de tipo ordinal y permite docimar que las muestras provienen de igual población o poblaciones diferentes. Las hipótesis formuladas fueron:

- H_o: las calificaciones de la preprueba del grupo control y experimental no difieren significativamente respecto a su tendencia central (mediana).
- H_A: las calificaciones de la preprueba del grupo control y experimental difieren significativamente respecto a su tendencia central (mediana).

El nivel de significación que se utilizó fue $\alpha = 0.05$ y se docimó mediante la aproximación a la normal (prueba de dos colas) para valores mayores que 20, obteniéndose que $Z = -0.21$ por lo que no hay evidencia para rechazar H_o, pudiéndose concluir que ambas muestran pertenecen a una misma población, o bien, que el proceso de emparejamiento llevado a cabo fue correcto.

Aplicación del tratamiento (variable independiente X)

El sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas se consideró como la variable independiente X. El mismos se aplicó a un subgrupo de 23 alumnos del primer año de la carrera de Ingeniería en Telecomunicaciones y Electrónica (grupo experimental), durante un período de tiempo de 18 semanas del año 2013. La frecuencia semanal fue de dos encuentros de dos horas cada uno, es decir, de 4 horas semanales, para un total de 72 horas presenciales, a las que se adicionaron otras 72 horas no presenciales. Cabe precisar que tanto el grupo control como el experimental recibieron los mismos contenidos de la asignatura de Programación y que el segundo grupo (experimental) recibió además contenidos de algoritmización, en correspondencia con el sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas que se propone.

Para el experimento se tomó como variable dependiente: Y → aprendizaje de la algoritmización para la resolución de problemas de programación computacional. Considerando que dicho aprendizaje está en correspondencia con los cuatro momentos del sistema de procedimientos didácticos y con los criterios evaluativos y patrones de logro que aporta el mismo.

Aplicación de la postprueba

Se aplicó la postprueba pedagógica para evaluar la efectividad del sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas (ver Tabla 14).

Est.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	PT
GExp.	4	5	4	2	4	4	5	4	5	2	3	2	3	4	5	4	3	4	3	3	5	3	5	86
GCon.	5	4	3	2	4	2	2	3	3	2	4	2	3	4	3	3	3	4	2	3	2	3	4	73

Tabla 14. Resultados de la postprueba.

Con el propósito de conocer si había diferencias significativas entre las calificaciones obtenidas por el grupo experimental y el grupo control se utilizó la Prueba U de Mann–Whitney para muestras no relacionadas, ideal para el caso de una escala de medición de tipo ordinal. A tales efectos se plantearon las siguientes hipótesis:

- H_o: Las calificaciones de los estudiantes del grupo experimental son menores o iguales que las del grupo control en la postprueba.
- H_A: Las calificaciones de los estudiantes del grupo experimental son mayores que las del grupo control en la postprueba.

El nivel de significación que se utilizó fue $\alpha = 0.05$ y se docimó (para una prueba de una cola) mediante la aproximación a la normal, obteniéndose que $Z = -2.25$, por lo que se rechazó Ho.

De lo anterior se pudo concluir que hay suficientes evidencias en los datos obtenidos para plantear que existen diferencias significativas entre las calificaciones de los estudiantes del grupo control y experimental en la postprueba, siendo estas últimas significativamente mayores, lo que implica un mejor aprovechamiento académico. De aquí que se pueda concluir desde el experimento pedagógico realizado (centrado en el perfeccionamiento de la dinámica del proceso de algoritmización), que el sistema de procedimientos didácticos apoyado en la ejemplificación de soluciones algorítmicas que se aporta, brinda suficientes evidencias sobre su influencia positiva en el perfeccionamiento de la variable dependiente: aprendizaje de la algoritmización para la resolución de problemas de programación computacional.

La concreción de ese aprendizaje se evidenció en un incremento de habilidades para concebir representaciones matemáticas de las situaciones problémicas que el profesor propuso, así como para identificar, seleccionar e integrar las estructuras algorítmicas necesarias para transformar estas representaciones, obteniendo generalizaciones basadas en pseudocódigos. También se observó un notable avance en el refinamiento de los algoritmos, a partir de la evaluación de su sintaxis y su semántica y en el desarrollo de corridas de los mismos.

CONCLUSIONES

La ejemplificación presentada facilita y amplía la comprensión del sistema de procedimientos didácticos creado a tales efectos y la forma de llevarlo a la práctica durante la dinámica del proceso de enseñanza-aprendizaje de la Programación en las carreras de ciencias computacionales.

El sistema de procedimientos didácticos que se aporta, apoyado en la ejemplificación de soluciones algorítmicas, brinda suficientes evidencias sobre su influencia positiva en el perfeccionamiento de la variable dependiente: aprendizaje de la algoritmización para la resolución de problemas de programación computacional.

BIBLIOGRAFÍA

Chesñevar, C. I (2001). Utilización de los mapas conceptuales en la enseñanza de la programación. Disponible en: http://cs.uns.edu.ar/~cic/2000/2000-jornadas-mapas/2000-jornadas-mapas.pdf [Consultado el 10 de diciembre de 2011].

De Lobos, M. E. (2010). Aprende a programar. Disponible en: http://www.mailxmail.com/curso-aprende-programar/tipos-estructuras-programacion-estructuras-basicas-secuencial [Consultado: febrero, 24 de 2011].

Faouzia, B y Mostafa, H. (2007). Utilisation des NTICs pour l'apprentissage et l'autoévaluation de l'algorithmique. SETIT 2007, 4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications. TUNISIA, Marzo 25-29, 2007.

Ferreira, A. y Rojo, G. (2005). Enseñanza de la programación. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Disponible en: http://teyet-revista.info.unlp.edu.ar/numero-1.htm [Consultado el 25 de junio de 2011].

Guibert, N., Guittet, L. y Girard, P. (2005). A study of the efficiency of an alternative programming paradigm to teach the basics of programming. Disponible en: http://www.lisi.ensma.fr/fr/equipes/idd/publications.html [Consultado el 10 de enero de 2012].

Hernández, S., Fernández, C. y Baptista, P. (1998). Metodología de la investigación social. McGraw-HILL Interamericana Edit., S. A.

Martinelli, O. (2006). "Elementos básicos de programación en C". Ediciones AKAL. S.A., Madrid.

Microsoft Student, (2009). Informatics programming.

Oviedo, M. y Ortiz, F.G. (2002). La enseñanza de la programación. Disponible en: http://bibliotecadigital.conevyt.org.mx/colecciones/documentos/somece2002/Grupo4/Oviedo.pdf [Consultado el 25 de abril de 2012]

Ramírez, R. V. (1991). NEWT, una herramienta de programación gráfica para la enseñanza del pensamiento algorítmico. IX Reunión de Intercambio de Experiencias en Estudios sobre Educación. Monterrey, N.L., México, Agosto de 1991.

Salgado, A., Alonso, I., Gorina, A. y Tardo, Y. (2013.a). Lógica algorítmica para la resolución de problemas de programación computacional: una propuesta didáctica. Vol. IV. Año 2013. Número 1, Enero-Marzo, pp. 57-76.

Salgado, A., Gorina, A. y Alonso, I. (2013.b). Modelo de la Dinámica Lógico–Algorítmica para la Resolución de Problemas de Programación Computacional. Revista EDUCARE. Volumen 17 Nº 1, Enero-Abril, pp. 27-51.

Salgado y otros (2014). Sistema de Procedimientos Didácticos para perfeccionar la Algoritmización Computacional. I Conferencia Científica Internacional UCIENCIA 2014, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Siegel, Sidney (1972). "Diseño experimental no paramétrico aplicado a las ciencias de la conducta", Cuba, Edit. Revolucionaria.