UNA PROPUESTA DIDÁCTICA PARA PERFECCIONAR LA ALGORITMIZACIÓN COMPUTACIONAL

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

DIRECCIÓN PARA CORRESPONDENCIA: E-mail: asalgadocastillo@gmail.com

Fecha de recepción: 22 - 07 - 2015 Fecha de aceptación: 10 - 09 - 2015

RESUMEN

En el presente artículo se propone un sistema de procedimientos didácticos para perfeccionar la algoritmización computacional, el que permite potenciar la doble modelación (matemática y computacional) que caracteriza la resolución de un problema de programación computacional. El citado sistema fue estructurado en cuatro procedimientos: construcción lógico – matemática, orientación matemático – algorítmica, estructuración algorítmico – generalizadora y validación algorítmico – computacional, los que favorecen el desarrollo de un pensamiento algorítmico – computacional. La factibilidad y pertinencia del sistema fue corroborada mediante la realización de un experimento pedagógico. El análisis estadístico realizado en dicho experimento permitió concluir que el sistema que se aporta brinda suficientes evidencias sobre sus posibilidades de perfeccionar el proceso de algoritmización y desarrollar un pensamiento algorítmico – computacional en los estudiantes que se inician en la programación.

PALABRAS CLAVE: Algoritmización; enseñanza de la programación; programación computacional; resolución de problemas.

A DIDACTIC PROPOSAL TO PERFECT THE COMPUTATIONAL ALGORITHMIZATION

ABSTRACT

In this article is proposed a didactics procedures system for improving computational algorithmization, which enhances a dual modelling (mathematical and computational) that characterizes the solving computer programming problems. The system was structured in four procedures: logical-mathematical construction, mathematical-algorithmic orientation, algorithmic – generalizing structure and algorithmic-computational validation, which favour the development of an algorithmic-computational thinking. The feasibility and

[1] Licenciado en Ciencia de la Computación. Máster en Neurociencias. Doctorante en Ciencias Pedagógicas. Hospital General Dr. Juan Bruno Zayas Alfonso. 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@uo.edu.cu
[3] Licenciado en Matemática. Doctor en Ciencias Pedagógicas. Profesor Titular. Departamento de Contabilidad y Finanzas. Filial Universitaria Contramaestre. Universidad de Oriente. Cuba. E-mail: gorina@contre.sum.uo.edu.cu

relevance of the system was confirmed by performing a pedagogical experiment. The statistical analysis in this experiment showed that this system provides sufficient evidence about their ability to improve the algorithmization process and develop an algorithmic-computational thinking in students who are new to programming.

KEYWORDS: Algorithmization; problems solving; computational programming; teaching programming.

INTRODUCCIÓN

La Programación es una disciplina de las Ciencias de la Computación con aplicaciones en todas las esferas de la sociedad, incluyendo problemas no triviales cuya resolución constituye un desafío intelectual. La verdadera dificultad no reside en expresar la solución del problema en términos de instrucciones elementales de un lenguaje de programación específico, sino en la resolución del problema propiamente dicha. El proceso de encontrar una solución adecuada a un problema provoca en el alumno un conflicto cognitivo pues no dispone de un sistema de estrategias que le permitan responder de manera satisfactoria (Arellano, N. y otros., 2014).

La adquisición de un estilo correcto de programación es una tarea fundamental en las carreras de ciencias computacionales, sin embargo la enseñanza de la programación es una tarea compleja que debe sustentarse en instrumentos y estrategias adecuadas si se quieren obtener resultados satisfactorios (Luna, C. y otros, 2007; Tan, J. y otros., 2014). Una alternativa que ha dado buenos resultados ha sido comenzar a enseñar programación utilizando algoritmos en pseudocódigos o diagramas de flujo para plasmar el modelo de la resolución de un problema (Kordakia, M., Miatidisb M. y Kapsampelisa, G., 2008; Martínez, S. y Fariñas, J. L., 2012; Salgado, A. y otros., 2013).

Hoy en día, los estudiantes de nivel superior que cursan carreras de ingeniería y especialmente aquellos inscritos en carreras afines a las ciencias computacionales, necesitan aprender a programar soluciones algorítmicas a problemas computables, utilizando diferentes paradigmas con diversos lenguajes y herramientas de programación (Arellano, N. y otros., 2012). Lo anterior demanda que se perfeccione el proceso de enseñanza-aprendizaje de la resolución de problemas de programación computacional en las carreras de ciencias computacionales: Licenciatura en Ciencia de la Computación e Ingenierías en Telecomunicaciones y Electrónica, Informática y Automática. La incidencia de estas carreras para potenciar el desarrollo económico político y social de un país es innegable, razón que justifica a priori la presente investigación, la cual está encaminada a solventar las insuficiencias que existen actualmente en los estudiantes al momento de resolver un problema computacional a través de la programación, dotándolos de un pensamiento algorítmico-computacional que les permita crear, modelar y refinar sus programas de forma eficiente y eficaz, es decir, de forma que alcancen los objetivos o efectos que se desean a partir de los mismos.

En tal sentido, las perspectivas de análisis para abordar la Didáctica de la Programación se fundamentan en una variedad de propuestas, a partir de enfoques orientados a la especificidad de un lenguaje particular (Wang, J., Mendori, T y Xiong, J., 2014; Davis, H. C., Hugh, C. y White, S., 2011; Pérez, R., 2009; Soler, Y. y otros., 2008; Al-Imamy, S., Alizadeh, J. y Nour, M. A., 2006; González, W., Estrada, V. y Martínez, M., 2006); mientras otros emplean un lenguaje algorítmico a través de pseudocódigos que facilite su posterior implementación (Arellano y otros. 2014; Arellano y otros. 2012; Martínez, S. y Fariñas, J. L., 2012; Novara, P., 2012; Kordakia, M., Miatidisb, M. y Kapsampelisa, G., 2008; Faouzia, B. y Mostafa, H., 2007; Ferreira, A. y Rojo, G., 2006; Guibert, N., Guittet, L. y Girard, P., 2006).

Los partidarios del primer enfoque proponen el uso de mapas conceptuales, multimedias , tutoriales y libros electrónicos en lenguajes de programación de alto nivel, que faciliten la representación gráfica de los elementos estáticos y dinámicos de programas implementados en un lenguaje, así como su diseño, visualización y prueba. Estas herramientas aumentan la comprensión de conceptos de un alto nivel de abstracción y complejidad, pero enfatizan en el dominio de la sintaxis del lenguaje, en detrimento de las habilidades de algoritmización.

En este enfoque se reconoce como válida la propuesta hecha por R. Pérez (2009) de la Universidad Politécnica del Valle, en México, consistente en un sistema informático- educativo con ambiente visual denominado “Nuevo Lenguaje de Programación (NLP)”, que usa seis lenguajes diferentes con el objetivo de potenciar el desarrollo de algoritmos; obteniéndose resultados alentadores con la herramienta, ya que los alumnos empiezan a descubrir la sintaxis de los lenguajes rápidamente y si no tienen experiencia previa en la programación, se adquiere a partir de que el software indica las correcciones sintácticas.

Sin embargo, aunque este parecería ser un proceder ideal, puede conducir al estudiante a una mecanización del pensamiento, ya que el mismo emplea mayor tiempo en aprender la sintaxis de los lenguajes, que en el diseño lógico del programa. Además, no tiene en cuenta las imprecisiones semánticas, siendo estas un indicador del cumplimiento de la intencionalidad del algoritmo. En resumen, aunque se reconoce la pertinencia de esta contribución, se considera que no logra potenciar, desde una óptica integradora, las relaciones existentes entre la algoritmización y su posterior codificación en un lenguaje específico, al no desarrollar habilidades esenciales para la programación, tales como la modelación y la representación.

Como parte de este mismo enfoque se destaca la propuesta de los investigadores árabes S. Al-Imamy, J. Alizadeh y M. A. Nour (2006), los que proponen que la enseñanza de la programación se sustente en el uso de plantillas en lenguaje C, afirmando que los estudiantes logran apropiarse de los conceptos del lenguaje en poco tiempo. No obstante, es criticable que se

potencie el aprendizaje de la sintaxis del lenguaje, desdeñando la formación de habilidades para el diseño lógico que deben tener los programas.

Así mismo, los investigadores cubanos W. González, V. Estrada y M. Martínez (2006) proponen un modelo que se basa en la Programación Orientada a Objetos (POO), necesitando un lenguaje de alto nivel que requiere de la formación de estrategias heurísticas para la resolución de un problema. En esta propuesta es censurable que se dirija a las generalidades de la programación y no tenga en cuenta las esencialidades de la algoritmización. Además, soporta su estrategia en la enseñanza de la POO, sin tener en cuenta que la propia naturaleza de lo heurístico requiere de un pensamiento algorítmico. Por lo que comenzar a enseñar con la POO implicaría un tiempo considerable para su estudio, en detrimento del aprendizaje y desarrollo de habilidades inherentes a la algoritmización.

Por su parte los investigadores brasileños F. Ramos de Melo y otros. (2014) proponen un sistema tutorial inteligente como modelo computacional basado en redes neuronales artificiales, para que los estudiantes puedan organizar y personalizar adecuadamente los contenidos didácticos para el estudio independiente, asegurando que esto permite un mejor uso del contenido y el conocimiento. Sin embargo esta propuesta aunque novedosa, se dirige a potenciar una estructuración del aprendizaje guiada por el estudiante, y a criterio de los autores del presente trabajo esto no es adecuado, pues la utilización de las herramientas tecnológicas para la enseñanza de la programación deben sustentarse en instrumentos didácticos bien estructurados, que permitan guiar y sistematizar el proceso de enseñanza-aprendizaje, donde medie el profesor como su conductor principal.

Ahora bien, dentro de los partidarios del segundo enfoque didáctico se destaca la metodología de A. Ferreira y G. Rojo (2006), sustentada en la ejecución de varias fases (análisis, diseño, implementación y prueba) para resolver un problema en un lenguaje algorítmico, el que luego se traduce a un lenguaje de alto nivel. Sin embargo, aun cuando esta propuesta resulta válida, pues se reconoce la algoritmización como parte esencial de la programación, no se profundiza en las particularidades de los procesos que intervienen en la didáctica de la misma. Esto se debe, fundamentalmente, a que no se favorece el componente lógico-matemático necesario en un lenguaje algorítmico estructurado, que posibilite realizar una comprensión y modelación matemática previa al proceso de implementación en dicho lenguaje.

En este mismo enfoque se destaca la propuesta de los investigadores franceses N. Guibert, L. Guittet y P. Girard (2006), consistente en una metodología que describe tres momentos principales del proceso de resolución de un problema de programación utilizando pseudocódigos en un software denominado “MELBA” y la de B. Faouzia y H. Mostafa (2007) relativa a un software llamado “Easy Algo”, el que permite algoritmizar usando pseudocódigos. Si bien estas dos propuestas son más avanzadas, porque con estos instrumentos didácticos se potencia la algoritmización como proceso esencial de la programación, las

mismas no brindan elementos didácticos para orientar la enseñanza en cada etapa de éste.

Las investigadoras N. Arellano y otros. (2014) de la Universidad de San Luis en Argentina proponen una metodología para la enseñanza de la programación a estudiantes noveles que parte de la inclusión de horarios extras en los laboratorios usando varios software de apoyo, comenzando con la herramienta "TIMBA" (Terribly Imbecile Machine for Boring Algorithms), la misma fue diseñada en la propia Universidad, y permite introducir al estudiante en la noción de algoritmo, luego se utiliza una aplicación denominada DIA para complementar de forma gráfica la resolución del algoritmo y por último el software "PseIn" (Novara, P., 2012) que permite modificar, ejecutar y depurar los algoritmos. Esta estrategia metodológica, si bien es acertada, en cuanto a que favorece la creación de un pensamiento computacional, no propicia los resultados esperados pues se centra en el uso de las mencionadas herramientas tecnológicas, sin sustentarse en un instrumento didáctico que oriente y estructure el proceso de enseñanza-aprendizaje de la programación computacional.

Asimismo los investigadores mejicanos J. J. Arellano y otros. (2012) proponen un "Software para la enseñanza-aprendizaje de algoritmos estructurados" con base en la heurística de resolución de problemas de G. Polya (2004), que da soporte a las fases de análisis y planteamiento del problema, además del diseño y traza completa de la prueba. La estructura y funcionalidad del software propuesto contribuye a que el estudiante adquiera, practique y ejercite la capacidad de resolver problemas de forma metódica a través de soluciones algorítmicas estructuradas.

Se debe reconocer que esta propuesta tiene en cuenta la algoritmización como eje central del proceso de resolución de problemas de programación computacional, además contiene las principales etapas del mismo, sin embargo al igual que las propuestas anteriores no se sustenta en instrumentos didácticos que permitan conducir de manera eficiente y eficaz su aplicación durante el proceso de programación.

Teniendo en cuenta la insuficiente instrumentación didáctica del proceso de algoritmización computacional y lo complejo que resulta para los profesores poder concretar estos enfoques e instrumentos, de manera efectiva en sus clases, surge la necesidad de proponer nuevos instrumentos didácticos que respondan a enfoques más integrales de la enseñanza de la programación y que potencien la formación de un pensamiento algorítmico – computacional.

Consecuentemente, la presente investigación tiene como objetivo, la elaboración de un sistema de procedimientos didácticos para perfeccionar el proceso de algoritmización computacional, sustentado en el modelo de la dinámica lógico-algorítmica de la resolución de problemas de programación computacional propuesto por A. Salgado, A. Gorina e I. Alonso (2013), el que está encaminado a elevar a niveles cualitativamente superiores la actividad formativa del futuro

egresado de las ciencias computacionales. El sistema ha sido ejemplificado por A. Salgado, I. Alonso y A. Gorina (2014) mediante la resolución de varios problemas de programación computacional, lo que ha permitido a los profesores que imparten la asignatura una mejor comprensión del mismo y así confirmar su pertinencia didáctica y metodológica.

DESARROLLO

El sistema de procedimientos didácticos que se propone para la algoritmización computacional está formado por un conjunto de acciones, lógicamente estructuradas, que posibilitan el desarrollo de un pensamiento algorítmico. El mismo se construyó a partir del método Sistémico Estructural Funcional, dada la necesidad didáctico – metodológica de secuenciar sus procedimientos y acciones de forma integrada. Es por tanto un instrumento de intervención didáctica, que tiene como objetivo general la orientación intencional a los profesores de programación para la conducción del proceso de enseñanza-aprendizaje de la algoritmización, durante el proceso de resolución de problemas de programación computacional.

Cabe precisar que el mismo está diseñado para potenciar la formación de aquellos estudiantes que se inician en el estudio de la Programación, los que deben priorizar el aprendizaje de contenidos de algoritmización computacional, que pueden estar contenidos en dicha asignatura o constituirse en cursos introductorios para las carreras de ciencias computacionales.

La intervención didáctica, que se podrá llevar a cabo mediante este sistema de procedimientos, favorecerá el desarrollo de acercamientos cada vez más profundos y esenciales a la algoritmización computacional, como elemento dinamizador del proceso de resolución de problemas de programación, con lo que se estará contribuyendo al desarrollo de un pensamiento algorítmico en el estudiante, independiente del lenguaje de programación, en virtud del cual se deberán ir obteniendo niveles cualitativamente superiores de formación computacional en dichos estudiantes. Es decir, que la lógica del sistema de procedimientos promueve transformaciones cada vez más relevantes, que contribuyen al perfeccionamiento de la algoritmización en la programación, las cuales se convertirán en una guía para el logro de una autonomía resolutora, a partir de un trabajo más consciente y estable, que viabilice el auto-desarrollo formativo de los mismos, de aquí su carácter didáctico.

Otras cualidades que lo distinguen desde su condición de sistema, son las siguientes:

- Es un sistema abierto, dando cuenta de ello el hecho de que está sometido a múltiples influencias externas y la dinámica que promueve permite su remodelación y mejora constante. Esto posibilita su continuo perfeccionamiento a partir de las nuevas tecnologías computacionales y el desarrollo de la propia didáctica de la computación.

- Es recursivo ya que adquiere sentido de los procedimientos que lo conforman (sus partes) y dichos procedimientos adquieren significado a través de su integración sistémica (todo), lo que da cuenta de su coherencia.
- Manifiesta su sinergia en el pensamiento algorítmico computacional que se configura en la dinámica de la resolución de problemas de Programación, como nueva cualidad totalizadora alcanzada en su implementación.
- Su entropía puede ser provocada por las insuficiencias que presentan los estudiantes para concebir un algoritmo coherente a la hora de resolver computacionalmente una situación problémica, así como por la resistencia de los actores del proceso para aceptar los cambios que en el mismo implica la introducción de la nueva dinámica de algoritmización, la que exige de una mayor preparación matemática y computacional, que requiere de una profundización en el trabajo con representaciones que involucran objetos, relaciones matemáticas y pseudocódigos. Otra fuente de entropía es la asociada al proceso de comunicación, el que no siempre logra la eficiencia necesaria.
- La homeostasis es favorecida cuando el profesor adquiere conciencia de la necesidad de realizar una doble modelación (matemática y computacional) para la resolución de los problemas de programación computacional y es capaz de formar habilidades en sus estudiantes para realizar las acciones previstas en el sistema de procedimientos didácticos. Además se puede potenciar valiéndose de las posibilidades que ofrecen los programas analíticos de Programación, de las Prácticas Laborales y de los Trabajos de Curso, en función de adiestrar a los estudiantes en la dinámica lógico-algorítmica que se propone. También se puede fomentar dicho equilibrio mediante el empleo de software educativos, que permitan profundizar en la interpretación de la algoritmización de diversas situaciones problémicas.
- Su autodesarrollo se expresa en el carácter flexible, abierto, dinámico que posee, el que facilita al profesor su sistemática adecuación a determinadas circunstancias contextuales, que permiten su progresivo perfeccionamiento y desarrollo.
- La estructura se conforma a partir de cuatro procedimientos didácticos que mantienen una sistemática interacción durante la dinámica de la algoritmización computacional, los que son denominados: construcción lógico-matemática, orientación matemático-algorítmica, estructuración algorítmico-generalizadora y validación algorítmico-computacional. Cada uno de estos procedimientos consta de dos tipos de acciones, uno dedicado a los profesores y el otro a los estudiantes. El sistema cuenta, además, con criterios evaluativos y patrones de logro para profesores y para estudiantes.

El procedimiento de la construcción lógico-matemática tiene como objetivo: la orientación a profesores y estudiantes sobre la forma de concretar, en el proceso de algoritmización computacional, las relaciones que se establecen

entre la comprensión, interpretación y representación matemática de la situación problémica.

Acciones a realizar por el profesor: Durante su ejecución el docente debe propiciar la motivación de los estudiantes por la resolución de los problemas de programación, a partir de:

- Analizar en el aula situaciones problémicas que requieran de una primera representación matemática, para motivar y transmitir patrones de actuación. En el proceso de descubrimiento de las relaciones entre los datos se obtiene una visión de sistema de la situación problémica, que incide en el desarrollo del pensamiento algorítmico, permitiendo al sujeto encontrar mayor cantidad de implicaciones, nexos y relaciones dentro de la información, favoreciendo la visión del estudiante sobre esta.
- Considerar los conocimientos matemáticos previos en la selección de las situaciones problémicas a utilizar en la asignatura. Los estudiantes, al abordar situaciones acordes a sus conocimientos previos, ganan en seguridad y pueden auto-valorar adecuadamente sus capacidades y posibilidades de resolverlas.
- Discutir numerosas situaciones problémicas sencillas, que vayan aumentando en su complejidad intrínseca y algorítmica. Esto permitirá que el estudiante cree la habilidad de analizar minuciosamente cada situación problémica, para comprenderla al máximo antes de pasar a su resolución y lograr una valoración positiva de la importancia que tiene darle solución.
- Ejercitar la recuperación mental de conocimientos matemáticos básicos relacionados con la situación problémica bajo análisis. 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. Esto favorecerá el desarrollo de habilidades para seleccionar correctamente los objetos a utilizar, de acuerdo a sus características.
- Demostrar cómo representar, a partir del uso de objetos y relaciones matemáticas conocidas, varias situaciones problémicas que hayan sido discutidas en el aula. El estudiante debe comprender como integrar todos los elementos analizados en una primera representación matemática del problema que se trata de resolver.
- Propiciar la confección, por parte de los estudiantes, de diversas representaciones matemáticas para un mismo problema. Esta acción se relaciona con la anterior, pero su objetivo es que una vez obtenidas esas diversas representaciones matemáticas, el estudiante pueda analizar cada una de ellas y compararlas 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 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.

- Representar situaciones problémicas integradoras, que permitan la sistematización de la comprensión y la interpretación, desde los conocimientos y habilidades matemáticas. Puede hacerse empleando el método de elaboración conjunta o algún método problémico.
- Demostrar, a través de las situaciones problémicas, la utilidad de la Programación, que permite automatizar importantes procesos productivos, de servicio, etc. Se puede lograr seleccionando situaciones problémicas provenientes de la esfera social, productiva y técnica del territorio y del país.
- Acciones a realizar por el estudiante: Los estudiantes deben avanzar en la realización de acciones dinamizadoras del razonamiento lógico-matemático propuesto, es decir, dirigir sus acciones hacia:
- Activar conocimientos matemáticos y del contexto en que se desarrolla la situación problémica que se presenta. Esto les permitirá ir imaginando objetos y relaciones, en correspondencia con los objetos y relaciones externas que presenta la citada situación.
- Interactuar con la situación problémica para obtener una primera comprensión del significado de la misma, que le posibilite la concepción de una vía de solución, para lo cual será necesario:
- Realizar un análisis detallado de la situación problémica, mediante el cual se fragmente la misma en sus principales partes, para luego examinar minuciosamente cada una de ellas con el propósito de identificar los objetos que las conforman, sus características, funciones y las relaciones que existen entre ellos.
- Vincular los resultados del análisis y establecer relaciones con la información obtenida. Es decir; relacionar los datos similares y localizar las diferencias y semejanzas, para que, una vez hecha la comparación, se puedan establecer los vínculos entre los componentes de la información.
- Determinar las partes más importantes o esenciales del objeto bajo estudio, a partir del análisis hecho.
- Integrar las partes seleccionadas en un objeto más simplificado y esencial que el original, en cuanto a estructura y funciones.
- Concebir abstracciones sustentadas en objetos y relaciones matemáticas para obtener una interpretación matemática de la situación problémica. 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.
- Perfeccionar la representación matemática inicial. Esto será posible volviendo sobre el análisis de la situación problémica, lo que le permitirá

profundizar en esta y enriquecer la interpretación que de ella se había hecho. Esto podrán hacerlo colectivamente, para que se transmitan patrones estrategias y métodos.

- Realizar varias representaciones matemáticas de un mismo problema. Esta forma de actuación le permitirá establecer comparaciones entre: el número de objetos utilizados, las relaciones establecidas y la intencionalidad deseada, lo que favorecerá la sistematización de los contenidos estudiados por los estudiantes.

El procedimiento de la orientación matemático-algorítmica tiene como objetivo: la orientación a profesores y estudiantes sobre cómo llevar a la práctica, en el proceso de algoritmización computacional, las relaciones que se establecen entre la representación matemática de la situación problémica y la identificación e integración jerárquica de estructuras lógico-computacionales.

Acciones a realizar por el profesor: Para tener éxito en la ejecución de este procedimiento deberá favorecerse:

- Enseñar el pseudocódigo como herramienta para la definición de las estructuras lógico-computacionales. Hasta este momento el estudiante debe conocer y haber sistematizado las diferentes estructuras computacionales, sobre la base del lenguaje natural, por lo que se hace necesario que el profesor seleccione o diseñe el pseudocódigo a emplear, explicándolo minuciosamente, basándose en numerosas ejemplificaciones.
- Trabajar en aras de la apropiación de conocimientos acerca de las propiedades y funciones de las estructuras lógico-computacionales. Puede lograrlo explicándoles dichas estructuras, sus funciones, relaciones, ventajas y limitaciones. Además de realizar una amplia ejemplificación de estos aspectos para que se fijen en la mente del estudiante y facilite así su posterior identificación. Debe utilizar fundamentalmente el lenguaje natural, lo que facilitará al estudiante captar la esencia de las estructuras sin detenerse en los aspectos técnicos de la sintaxis.
- Inducir a la utilización de diversas estructuras lógico-computacionales para un mismo problema. Se hace con el objetivo de analizar cada una de ellas y realizar comparaciones, teniendo en cuenta los objetos utilizados, la intencionalidad del uso de cada una, sus ventajas y limitaciones.
- Emplear métodos de enseñanza participativos para propiciar la reflexión y discusión de las representaciones de las situaciones problémicas. Con esta acción se promueve la recuperación y reconstrucción de los conocimientos matemáticos y computacionales de los estudiantes.
- Incidir en que los estudiantes resuman los aspectos relevantes de la representación matemática bajo estudio. Esto tiene el propósito de suscitar la profundización en los objetos y relaciones que conforman dicha representación y facilitar que puedan recuperar de su mente estructuras

lógico-computacionales adecuadas para obtener representaciones más esenciales de la misma.

- Analizar diferentes formas de ordenar y concatenar las estructuras computacionales, de acuerdo con las representaciones matemáticas obtenidas. Se hace con la intención de contribuir a formar en el estudiante patrones a seguir en la resolución de las situaciones problémicas, de manera que este pueda valorar la pertinencia de cada concatenación e integración teniendo en cuenta las ventajas, desventajas, relaciones y características de cada estructura, explotando el uso de casos particulares para reforzar el aprendizaje de la representación matemática-computacional mediante estructuras algorítmicas, transitando de lo simple a lo complejo.

Acciones a realizar por el estudiante: Deberán dirigir su trabajo hacia:

- Analizar la representación matemática de la situación problémica. Este análisis deberá irlo complementando con la interpretación de los objetos y relaciones que conforman dicha situación a partir de sus conocimientos computacionales.
- Identificar las estructuras algorítmicas necesarias para transformar los objetos y relaciones que aparecen en la representación matemática obtenida. Por ejemplo, si está ante una representación matemática en términos de una sumatoria debe identificar como estructuras computacionales posibles a emplear, las iterativas.
- Comparar las estructuras lógico-computacionales identificadas. Aquí, deberá ser capaz de seleccionar, analizar y concatenar adecuadamente las mencionadas estructuras, en aras de formar una estructura funcional del proceso de programación y obtener niveles cada vez más esenciales y precisos de la representación matemática inicial, a partir de una perspectiva algorítmico-computacional.
- Sistematizar la representación de las estructuras lógico-computacionales identificadas e integradas en el pseudocódigo propuesto, a partir de representaciones matemáticas de situaciones problémicas más complejas. Esto puede hacerlo mediante el empleo de numerosas estructuras lógico-computacionales, para remodelar diversas representaciones matemáticas, lo que les permitirá apropiarse de las principales vías de integración de dichas estructuras y alcanzar una adecuada familiarización con el pseudocódigo, antes de pasar al diseño del algoritmo. Cabe señalar que en algunas situaciones la representación matemática obtenida conduce a una modelación computacional muy compleja, lo que demanda una revisión y perfeccionamiento de la citada representación.

El procedimiento de la estructuración algorítmico-generalizadora tiene como objetivo la orientación a profesores y estudiantes sobre la forma de implementar las relaciones que se establecen entre la generalización de la representación

pseudocodificada y la identificación e integración jerárquica de estructuras lógico-computacionales.

Acciones a realizar por el profesor: Para conseguir que dicha generalización sea eficaz deberá trabajar por:

- Lograr un compromiso, por parte del estudiante, hacia la construcción y sistematización de un amplio conocimiento sobre las formas de concatenar estructuras lógico-computacionales para formar un algoritmo. Esto se puede lograr con la ejercitación de numerosas situaciones problémicas donde el estudiante deba identificar y concatenar adecuadamente dichas estructuras. Aquí se le debe exigir que establezca comparaciones para justificar el uso de determinadas estructuras, así como su integración mediante un pseudocódigo en un algoritmo.
- Elaborar diferentes formas de estructurar una misma representación matemática de una situación problémica, usando pseudocódigos. Aquí se debe sistematizar el uso del pseudocódigo mediante la realización de numerosos ejemplos en los que se enfaticen las ventajas y desventajas de cada estructuración realizada.
- Desarrollar un convencimiento sobre lo imprescindible de aprender a construir algoritmos antes de introducirse en el conocimiento de un determinado lenguaje de programación. Se pueden mostrar ejemplos de las limitaciones e inconvenientes de hacer una estructuración computacional en un lenguaje de alto nivel (como C, C++, Java, C#, etc.) sin concebir un algoritmo previo. Luego comparar con la estructuración que se puede lograr usando un algoritmo en pseudocódigo, enfatizando en las ventajas de este proceder. Todo lo anterior permitirá explicitar los errores (estructurales y/o semánticos) que se pueden cometer cuando se pasa a resolver una situación problémica empleando directamente un lenguaje de programación.
- Reconocer y aplicar el pseudocódigo como herramienta base para la escritura del algoritmo y como vía de alcanzar la generalización en la resolución de problemas de programación computacional. Con independencia de lo estrechamente relacionada que está esta acción con la anterior, es importante precisarla para puntualizar la necesidad de mostrar ejemplos de estructuración de algoritmos usando pseudocódigo. En este punto deberán enfatizarse las ventajas del mismo, como herramienta que permite escribir algoritmos usando un lenguaje similar al lenguaje natural, lo que facilitará que el estudiante se identifique con el pseudocódigo de una forma más amena.
- Confrontar puntos de vista contrarios para contribuir a la elaboración de nuevas propuestas de estructuración que conduzcan a algoritmos más eficientes y generales. Esto conllevará a la modificación positiva de algunos patrones preestablecidos, además de fomentar en el estudiante hábitos de

socialización de sus ideas y de aceptación del trabajo colectivo para perfeccionar sus propias ideas.

- Efectuar corridas del algoritmo que ha sido diseñado usando pseudocódigos; estas pueden realizarse de manera manual o mediante un software de apoyo como el "PSeInt" de P. Novara (2012), Software para la enseñanza-aprendizaje de algoritmos estructurados de J.J. Arellano y otros. (2012) o cualquier otro que permita cumplir con el citado propósito. Esto favorecerá la creación y análisis de nuevas generalizaciones algorítmicas, además de motivar al estudiante y desarrollar en él habilidades computacionales.

Acciones a realizar por el estudiante: Será preciso que encamine el proceso de aprendizaje hacia:

- Realizar una resignificación y reestructuración lógico-computacional de la representación matemática de la situación problémica bajo estudio. Debe sistematizarse la identificación, selección y concatenación de las estructuras lógico-computacionales, con el objetivo de afianzar los conocimientos adquiridos sobre las mismas.
- Obtener una generalización de la representación matemática, expresándola en pseudocódigos. Debe sistematizarse la transformación de la representación matemática obtenida, en una estructuración algorítmica usando pseudocódigos. Esto con vista a obtener una primera aproximación de la solución algorítmica deseada.
- Ejercitar, manual o computacionalmente, la estructuración algorítmica de la representación matemática realizada usando pseudocódigos. Puede realizarse empleando el software indicado por el profesor, con el objetivo de desarrollar habilidades computacionales, a la vez que elevar la motivación por el estudio de la estructuración algorítmica mediante pseudocódigos.
- Aprovechar puntos de vista divergentes, emitidos por otros estudiantes, para perfeccionar sus propuestas de estructuración, de manera que sean más generales y eficientes. Esto conllevará a la modificación positiva de algunos de sus patrones de abordaje de la estructuración algorítmica, además de que potenciará sus hábitos de socialización de ideas y aceptación del trabajo en grupos.

El procedimiento de la validación algorítmico-computacional tiene como objetivo la orientación a profesores y estudiantes sobre la conducción del proceso de instrucción de las relaciones que se establecen entre la generalización de la representación pseudocodificada, el control sintáctico computacional y la confirmación semántica de la representación computacional.

Acciones a realizar por el profesor: Para llevar a cabo este procedimiento será preciso:

- Considerar la importancia de 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. Este conocimiento básico se debe formar en el estudiante a partir de la sistematización de las características del pseudocódigo y su aplicación a la solución de numerosos ejemplos.
- Convencer al estudiante de que debe considerar como premisa esencial que el empleo de pseudocódigos no significa la ausencia de errores sintácticos. Mostrar al estudiante ejemplos concretos donde se evidencien errores sintácticos comunes al emplear pseudocódigos. Esto permitirá ir creando la necesidad de revisar minuciosamente el pseudocódigo del algoritmo una vez escrito, lo que contribuirá al desarrollo de habilidades para implementar algoritmos sintácticamente correctos en un lenguaje de programación. Trabajar en grupos para establecer discusiones que den la oportunidad de exponer y defender ideas sobre la intencionalidad de la estructuración realizada. Así se potenciará que los estudiantes se apropien de patrones de análisis y perfeccionamiento constante de los procesos de control sintáctico y de confirmación semántica.
- Evidenciar la importancia de optimizar el pseudocódigo para elevar la eficiencia y eficacia de los resultados. Esto puede lograrse a través de ejemplos concretos en los que se creen conflictos cognitivos que estimulen cambios en las estructuraciones algorítmicas realizadas, lo que favorecerá la formación de habilidades para diseñar algoritmos que permitan optimizar la memoria y el tiempo de ejecución del computador.
- Promover la ejercitación de la estructuración algorítmica realizada, tanto manual como empleando algún software que puede ser el Software para la enseñanza – aprendizaje de algoritmos estructurados", el "PseInt", u otro afín, con el objetivo de corroborar si el algoritmo cumple con la intencionalidad deseada. La ejercitación debe permitir corroborar si los datos de salida del algoritmo presentan las características previamente estimadas para la solución de la situación problémica.
- Resolver ejemplos integradores que permitan sistematizar las acciones anteriores, potenciando la emergencia de un pensamiento algorítmico-computacional. Esto facilitará que el estudiante observe el proceso resolutor completo y se apropie del modo de abordar una situación problémica, para lo cual debe de alcanzar niveles más esenciales de síntesis reflexivas, imprescindibles para conformar algorítmos que brinden soluciones eficientes y eficaces.

Acciones a realizar por el estudiante: En la dinámica del aula deberá encaminar su proceso de aprendizaje de la resolución de problemas de programación computacional hacia:

- Refinar el algoritmo, a partir de tomar en cuenta la escritura del pseudocódigo. Esto le permitirá regular y evaluar sistemáticamente el

proceso de algoritmización computacional con el objetivo de adquirir hábitos de revisión sintáctica de los algoritmos.

- Desarrollar habilidades de exactitud y precisión para el logro y control eficiente de los algoritmos. Será importante llegar al convencimiento de que la exactitud y precisión de un algoritmo determina la calidad de su ejecución una vez implementado en un lenguaje de programación, dando respuesta a la solución de la situación problémica bajo estudio.
- Comprobar que el conjunto de sentencias es completo. Debe verificar que el algoritmo contenga los pasos a realizar, suficientemente detallados, además de un conjunto de palabras reservadas del pseudocódigo concebido.
- Confirmar el flujo de control del algoritmo. Se debe confirmar que el orden temporal en el cual se ejecutan los pasos individuales del algoritmo es correcto, es decir, que no se viola la disposición lógica de la secuencia de operaciones establecida.
- Rediseñar determinadas partes del algoritmo, en caso de que no funcionen bien. Es importante repetir el proceso de localización de errores, definiendo nuevos casos de prueba y corriendo de nuevo el algoritmo con dichos datos.
- Verificar la validez de la representación computacional. La realización de 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. Esto facilitará ratificar si la estructuración del pseudocódigo es correcta en términos del contenido.
- Resolver problemas integradores, que estimulen la formación de un pensamiento algorítmico computacional. El estudiante transitará por el proceso resolutor completo y se apropiará de los patrones más relevantes que permiten abordar una situación problémica, conformando algorítmos que brinden soluciones eficientes y eficaces.

Para medir la efectividad del sistema de procedimientos se proponen los siguientes criterios evaluativos y patrones de logros para profesores y estudiantes involucrados en el proceso.

Criterio evaluativo para los profesores: Nivel de pertinencia didáctica desarrollado en la enseñanza de la algoritmización, en el proceso de resolución de problemas de programación computacional.

Patrones de logro para los profesores: Evidenciar un adecuado dominio del proceso de resolución de problemas de programación computacional desde la estructuración del sistema de relaciones esenciales de la dinámica lógico-algorítmica que lo sustenta, así como una acertada valoración didáctica de la algoritmización como alternativa viable para la enseñanza de la resolución de problemas de programación computacional y una apropiada sistematización del sistema de procedimientos didácticos que se propone, expresado en el desarrollo de habilidades algorítmicas en sus estudiantes, que les permitan

representar matemáticamente las situaciones problémicas y generalizarlas mediante pseudocódigos, además de una satisfactoria motivación y desempeño de los estudiantes para estructurar algorítmicamente representaciones matemáticas de manera eficiente y eficaz.

Criterio evaluativo para los estudiantes: Nivel de desarrollo algorítmico alcanzado para la resolución eficiente y eficaz de problemas de programación computacional.

Patrones de logro para los estudiantes: Evidenciar la apropiación de habilidades para la concepción de representaciones matemáticas de las situaciones problémicas que se les presenten, el reconocimiento de las estructuras algorítmicas necesarias para transformar las representaciones matemáticas, la integración de las estructuras computacionales, previamente seleccionadas, en una generalización de la representación matemática pseudocodificada, el refinamiento de los algoritmos a partir de la evaluación de su sintaxis y su semántica y la concepción de algoritmos mediante pseudocódigos, realizando corridas manuales y/o empleando algún software.

Corroboración de la viabilidad y pertinencia del sistema de procedimientos didácticos

Para llevar a cabo la citada corroboración se diseñó y ejecutó un experimento pedagógico, clasificado como experimento con preprueba – postprueba y grupo control. Este diseño incorporó la aplicación de prepruebas y postpruebas a los dos grupos que formaron parte el experimento (el grupo control y el experimental) ambos pertenecientes al primer año de la carrera de Ingeniería en Telecomunicaciones y Electrónica de la Universidad de Oriente.

Cabe precisar que para conformar los citados grupos, en lugar de que los estudiantes fueran asignados al azar a estos (que es el método más preciso), 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 tal.

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 se muestra en la tabla 1:

Tabla 1. Diseño del emparejamiento con los dos grupos (experimental y control).

EG_1	**O_1**	**X:** Sistema de procedimientos didácticos	**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). 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.	

Nota: *Elaboración de los autores.*

El empleo de la preprueba ofreció dos ventajas:

- Las puntuaciones de las prepruebas se usaron para fines de control en el experimento. Al compararse las prepruebas de los dos grupos se pudo 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 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 una preprueba donde se tuvieron en cuenta los elementos cognoscitivos que se desean formar en lo referido al pensamiento algorítmico – computacional. Se calificó la prueba en base a la escala ordinal (2, 3, 4, 5) y luego se ordenaron las calificaciones de cada grupo de menor a mayor y se procedió con el emparejamiento (ver tabla 2).

Tabla 2. Resultados de la aplicación de la preprueba y emparejamiento en los dos grupos (experimental y control).

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**
G_1	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	4	5	**59**
G_2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	**59**

Nota: Elaboración de los autores.

A tales efectos se utilizó la Prueba no paramétrica U de Mann – Whitney (Siegel, S., 1972, p.143-155) que permite docimar que las muestras provienen de igual población o poblaciones diferentes. Las hipótesis fueron:

- H_0: 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 $\alpha = 0.05$, obteniéndose que $Z = -0.21$ por lo que no hay evidencia para rechazar H0, 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: El sistema de procedimientos didácticos (variable independiente X) se aplicó en un subgrupo de 23 alumnos durante un periodo de tiempo de 18 semanas del año 2013. La frecuencia semanal fue de dos encuentros de dos horas cada uno, para un total de 72 horas presenciales, a las que se adicionaron otras 72 horas no presenciales. Para el experimento se tomó como variable dependiente: Y $\rightarrow$ aprendizaje de la algoritmización para la resolución de problemas de programación computacional. Considerando que dicho aprendizaje está en correspondencia con los cuatro procedimientos del sistema y con los criterios evaluativos y patrones de logro que aporta el mismo.

Aplicación de la postprueba: Se realizó para evaluar la efectividad del sistema de procedimientos didácticos. Al igual que para la preprueba se calificó en base a la escala ordinal (2, 3, 4, 5) y luego se ordenaron las calificaciones de cada grupo de menor a mayor (ver tabla 3). 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 no paramétrica U de Mann – Whitney. A tales efectos se plantearon las siguientes hipótesis:

- H_0: 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.

Tabla 3. Resultados de la postprueba.

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**
G_1	4	5	4	2	4	4	5	4	5	2	3	2	3	4	5	4	3	4	3	3	5	3	5	**86**
G_2	5	4	3	2	4	2	2	3	3	2	4	2	3	4	3	3	3	4	2	3	2	3	4	**73**

Nota: *Elaboración de los autores.*

El nivel de significación que se utilizó fue $\alpha = 0.05$, obteniéndose que $Z = -2.25$, por lo que se rechaza H_0.

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 que el sistema de procedimientos didácticos que se aporta brinda suficientes evidencias sobre su influencia positiva en el perfeccionamiento del 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

Al profundizar en el proceso de enseñanza-aprendizaje de la resolución de problemas de programación se evidenciaron insuficiencias relativas a la instrumentación didáctica de dicho proceso. Situación expresada en el predominio del uso de la ejercitación de los lenguajes de programación en las computadoras y el uso de software educativos, mapas conceptuales entre otros como estrategia principal de enseñanza, que si bien sirven como instrumentos didácticos, adolecen de un instrumento base que guie su utilización para formar en el estudiante un pensamiento algorítmico-computacional.

El sistema de procedimientos didácticos propuesto consta de cuatro procedimientos que permiten instrumentar el proceder didáctico – metodológico

para la algoritmización computacional, en aras de potenciar la formación de un pensamiento algorítmico-computacional, imprescindible para los profesionales de las carreras de ciencias computacionales.

La corroboración y aplicación del sistema de procedimientos didácticos fueron exitosas, tomando en consideración el adecuado nivel de pertinencia, factibilidad y coherencia logrado en el experimento pedagógico realizado. Al respecto se reconocieron las potencialidades del mencionado sistema para formar habilidades y desarrollar el pensamiento algorítmico-computacional en los estudiantes que se inician en el mundo de la programación.

BIBLIOGRAFÍA

Al-Imamy, S., Alizadeh, J. y Nour, M. A. (2006). On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. In Journal of Information Technology Education, 2006, 5: pp.271 – 283.

Arellano, J. J., Nieva, O. S., Solar, R. y Arista, G. (2012). Software para la enseñanza-aprendizaje de algoritmos estructurados. Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación, 2012, (8): pp.23 – 33.

Arellano, N., Fernandez, J., Rosas, M. V. y Zuñiga, M. E. (2014). Estrategia metodológica de la enseñanza de la programación para la permanencia de los alumnos de primer año de Ingeniería Electrónica. Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación, 2014, (13): pp.55 – 60.

Davis, H. C., Hugh, C. y White, S. (2011). The personalization of a learning environment: student-led connections online and offline. At HEA Enhancement Academy Team Leaders Meeting in May 2011, University of Southampton, 25 – 26 May 2011. Recuperado de: http://eprints.soton.ac.uk/272349/

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

Ferreira, A. y Rojo, G. (2005). Enseñanza de la programación. En Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología, 2006, 1(1): pp.1 – 8. Recuperado de: http://teyet-revista.info.unlp.edu.ar/numero-1.htm

González, W., Estrada, V. y Martínez, M. (2006). Contribución al desarrollo de la creatividad a través de la enseñanza de la programación. En Revista Pedagogía Universitaria, 2006, 9 (3) Recuperado de: http://169.158.24.166/texts/pd/1894/04/3/189404308.pdf

Guibert, N., Guittet, L. y Girard, P. (2006) Performances et usages d' un environnement d'apprentisage de la programmation basé sur exemple. ERGO'IA, 2006, pp.103–110. Recuperado de: http://www.lisi.ensma.fr/fr/equipes/idd/publications.html

Kordakia, M., Miatidisb, M. y Kapsampelisa, G. (2008). A computer environment for beginners' learning of sorting algorithms: Design and pilot evaluation. En Revista Computers & Education, Volumen 51, No. 2, Septiembre 2008, pp. 708 – 723.

Luna, C., Pedemonte, M., Viera, M. y Fraschini, E. (2007). Organización para un curso de programación en un contexto de masividad. Resultados tras experiencia de 4 años. Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación, 2007, (2): pp.83 – 91.

Martínez, S. y Fariñas, J. L. (2012). La competencia elaborar programas informáticos desde el proceso de enseñanza – aprendizaje de la disciplina lenguaje y técnicas de programación. En Revista Didasc@lia, 2012, 3(2): pp.125 – 144.

Novara, P. (2012). PseInt. Recuperado de: http://pseint.sourceforge.net

Pérez, R. (2009). Una herramienta y técnica para la enseñanza de la programación. Recuperado de: http://campusv.uaem.mx/cicos/imagenes/memorias/6tocicos2008/Articulos/Cartel%206.pdf

Polya, G. (2004). How to Solve It. Princeton Science Library Edition, 2004.

Ramos de Melo, F., Flôres, E. L., Diniz de Carvalho, S., Gonçalves de Teixeira, R. A., Batista, L. F. y Renato de Sousa, G. (2014). Computational organization of didactic contents for personalized virtual learning environments. En Revista Computers & Education. Volumen 79, Octubre 2014, pp. 126 – 137.

Salgado, A., Alonso, I., Gorina, A. (2014). Ejemplificación de la solución algorítmica de problemas de programación computacional. En Revista Didasc@lia, 2014, 5 (4): pp.15–36. Recuperado de: http://ojs.uo.edu.cu/index.php/Didascalia/article/download/4499/3787

Salgado, A., Alonso, I., Gorina, A. y Tardo, Y. (2013). Lógica algorítmica para la resolución de problemas de programación computacional: una propuesta didáctica. En Revista Didasc@lia, 2013, 4 (1): pp. 57 – 76. Recuperado de: http://revistas.ojs.es/index.php/didascalia

Salgado, A., Gorina, A. y Alonso, I. (2013). Modelo de la dinámica lógico-algorítmica para la resolución de problemas de programación computacional. En Revista Educare, 2013, 17 (1): pp. 27-51. Recuperado de: http://revistas.upel.edu.ve/index.php/educare/article/view/1071/384

Siegel, S. (1972). Diseño experimental no paramétrico aplicado a las ciencias de la conducta. Cuba, Editorial Revolucionaria, 1972. pp. 143 – 155.

Soler, Y., Frías, I., Linares, M. J., Rodríguez E. A. y Lezcano, M. (2008). Mapa conceptual tipos abstractos de datos y sistema de visualización de programas SVP – SUBC: herramientas eficaces en la formación virtual del ingeniero informático. Congreso Virtual Iberoamericano de Calidad en Educación a Distancia. Recuperado de: http://es.scribd.com/doc/21739903/RD14.Pág.1 – 13.

Tan, J., Guo, X., Zheng, W. y Zhong, M. (2014). Case – based teaching using the Laboratory Animal System for learning C/C++ programming. En Revista Computers & Education. Volumen 77, Agosto 2014, pp. 39 – 49.

Wang, J., Mendori, T y Xiong, J. (2014). A language learning support system using course – centered ontology and its evaluation. En Revista Computers & Education. Volumen 78, Septiembre 2014, pp. 278 – 293.