Rev. innova educ. (2023). Vol. 5 Núm. 6 págs. 105-120

Revista Innova Educación

www.revistainnovaeducacion.com

ISSN: 2664-1496 ISSN-L: 2664-1488

Editada por: Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú

ARTÍCULO ORIGINAL

El impacto del uso de Scratch para favorecer el pensamiento algorítmico en estudiantes de educación superior

The impact of using Scratch to promote algorithmic thinking in higher education students

O impacto do uso do Scratch na promoção do pensamento algorítmico em estudantes de ensino superior

María Velasco-Ramírez[1]
Universidad Veracruzana, Xalapa – Veracruz, México
https://orcid.org/0000-0003-1491-7429
lvelasco@uv.mx (correspondencia)

Alma Otero-Escobar
Universidad Veracruzana, Xalapa – Veracruz, México
https://orcid.org/0000-0001-9266-6587
aotero@uv.mx

DOI: https://doi.org/10.35622/j.rie.2023.05v.006

Recibido: 11/08/2023 Aceptado: 5/11/2023 Publicado: 12/11/2023

PALABRAS CLAVE

educación superior, estrategia tecnopedagógica, pensamiento algorítmico, Scratch.

RESUMEN. En este trabajo de investigación se presenta la propuesta de una estrategia tecnopedagógica de intervención, orientada a favorecer el pensamiento algorítmico. Se define el constructo pensamiento algorítmico, se describe la estrategia tecnopedagógica seleccionada, la cual consiste en el uso del ambiente de programación Scratch para desarrollar el pensamiento algorítmico a través de una plataforma electrónica. La investigación se ha realizado bajo un enfoque cuantitativo, transversal y descriptivo, con un diseño preTest-posTest para un grupo. Las pruebas se aplicaron a 48 estudiantes de Licenciatura en Sistemas Computacionales Administrativos de la Universidad Veracruzana que cursaban la experiencia educativa de Diseño e Implementación de Algoritmos. Se utilizó la prueba Wilcoxon para muestras relacionadas en SPSS, con la finalidad de aceptar o rechazar la hipótesis de investigación planteada, lo que permitió observar resultados favorables en el postTest con respecto al preTest, concluyendo que el uso de Scratch contribuye a las habilidades del pensamiento algorítmico.

KEYWORDS

higher education, technopedagogical

ABSTRACT. This research work presents the proposal of a technopedagogical intervention strategy aimed at promoting algorithmic thinking. The construct of algorithmic thinking is defined, and the selected technopedagogical strategy is described, which involves the use of the Scratch programming environment to develop algorithmic thinking through an electronic platform. The research was conducted using a quantitative, cross-sectional, and descriptive approach with a pretest-posttest design for a group of 48 students majoring in Computer Systems Administration

[1] Docente en la Universidad Veracruzana, México. Máster en Tecnología Educativa por la Universidad Atenas Veracruzana, México.

strategy, algorithmic thinking, Scratch.

at the University of Veracruz who were taking the educational experience in Algorithm Design and Implementation. The Wilcoxon test for related samples in SPSS was used to accept or reject the research hypothesis, which allowed for the observation of favorable results in the posttest compared to the pretest, concluding that the use of Scratch contributes to algorithmic thinking skills.

PALAVRAS-CHAVE

educação superior, estratégia tecnopedagógica, pensamento algorítmico, Scratch.

RESUMO. Neste trabalho de pesquisa, apresenta-se a proposta de uma estratégia tecnopedagógica de intervenção destinada a promover o pensamento algorítmico. O construto do pensamento algorítmico é definido, e descreve-se a estratégia tecnopedagógica selecionada, que envolve o uso do ambiente de programação Scratch para desenvolver o pensamento algorítmico por meio de uma plataforma eletrônica. A pesquisa foi conduzida com uma abordagem quantitativa, transversal e descritiva, com um design pré-teste e pós-teste para um grupo de 48 estudantes de Licenciatura em Administração de Sistemas de Computação na Universidade Veracruz que estavam cursando a experiência educacional em Design e Implementação de Algoritmos. Foi utilizado o teste de Wilcoxon para amostras relacionadas no SPSS para aceitar ou rejeitar a hipótese de pesquisa, o que permitiu observar resultados favoráveis no pós-teste em relação ao pré-teste, concluindo que o uso do Scratch contribui para as habilidades de pensamento algorítmico.

1. INTRODUCCIÓN

La resolución de problemas algorítmicos es probablemente una de las habilidades más importantes que debe poseer un estudiante de ciencias de la computación, así como la de pensar de manera creativa (Biju, 2019). Es una labor desafiante tanto para los profesores el fomentarla como para los estudiantes adquirirla.

El nivel abstracto del tema y la falta de vinculación con problemas prácticos y significativos son factores que afectan la motivación en los estudiantes al enfrentar la resolución de problemas en el tema de desarrollo de algoritmos (Chezzi et al., 2017).

Otros factores que influyen en la dificultad para resolver problemas algorítmicos son, la falta de comprensión de diversos conceptos con frecuencia considerados como difusos, una sobrecarga intrínseca debido al flujo de información que debe ser posible de entender, así como una sobrecarga extrínseca por la forma en que se presenta la información (Babori et al., 2016) y tal como señalan Blanco-Hamad et al. (2016) involucra las habilidades de analizar, interpretar, abstraer, modelar, identificar y validar la información que ofrece la situación problemática.

Generalmente se espera que los estudiantes de un curso de Diseño e Implementación de Algoritmos sean capaces de resolver problemas eligiendo uno de los métodos de diseño adecuados. Por ello se apuesta por el uso de estrategias en el contexto del constructivismo que permitan fomentar el pensamiento algorítmico en los estudiantes, que les permita ser capaces de describir y abstraer el problema, diseñar el algoritmo y probar la solución (Ritter & Standl, 2023).

Ülker (2020) señala que, durante el diseño del algoritmo, una de las principales preocupaciones es favorecer el pensamiento algorítmico. Mientras que Byrka et al. (2021), basados en un análisis científico concluyeron que el pensamiento algorítmico (PA) es esencial no solo en el área de las TIC, sino que tiene una mayor importancia en cualquier materia de educación superior por su valor sustancial en el contexto de la futura profesión y en la vida cotidiana en condiciones de la sociedad del conocimiento.

Por otro lado, una estrategia tecno didáctica es una serie de actividades estructuradas en contenidos específicos de conocimiento que son aplicadas en contextos virtuales para movilizar y transferir competencias digitales hacia el logro de intenciones pedagógicas (Martín & Calvillo, 2017).

Los autores señalan que, al emplear una estrategia, se deben enfocar un conjunto de actividades secuenciales en un marco de acción para el logro de una intención pedagógica, lo que implica integración, situando un contenido específico, desde un enfoque pedagógico, en un ambiente o herramienta virtual.

En el contexto de esta investigación, una estrategia tecnopedagógica se define como una combinación de enfoques pedagógicos sobre el proceso de enseñanza aprendizaje y recursos tecnológicos utilizados en el diseño actividades de aprendizaje en ambientes semipresenciales o virtuales.

Existe la creencia general de que algunas personas son mejores solucionadores de problemas que otras porque usan estrategias más efectivas para resolverlos.

En este sentido, se han desarrollado varios sistemas para ayudar a los usuarios a aprender el diseño de algoritmos (Scratch, AlgoBox, entre otros) que proporcionan una visión de la programación más cercana al lenguaje natural, así como los proyectos de Codewitz como un ejemplo que apunta a desarrollar una visualización basada en la web de los conceptos de programación (Babori et al., 2016).

Principales características del pensamiento algorítmico

Byrka et al. (2021), nos dicen que el concepto de "pensamiento algorítmico" en estudios psicológicos y pedagógicos modernos se interpreta de diferentes maneras que reflejan varios aspectos de su visión por parte de ellos científicos, pero común a todos los puntos de vista es la determinación de un algoritmo como resultado de lo pensado. Según Gubina (2016, citado en Byrka et al., 2021), el pensamiento algorítmico es un sistema de técnicas, construcciones, un conjunto de métodos de actividad, necesarios para resolver un problema particular.

Este tipo de pensamiento se realiza al identificar subtareas separadas para resolver un problema, construir un modelo de información, organizando la búsqueda de la información necesaria, y obteniendo el algoritmo apropiado.

Güler (2021), menciona que, para la mayoría de las personas, la palabra algoritmo puede sugerir algo sobre ciencias de la computación o programación. Sin embargo, el PA se aplica solamente a la ciencia programación sino a las actividades diarias, ya que se desarrolla a partir de las experiencias de la vida cotidiana (Hubalovsky & Korinek, 2015).

Lockwood et al. (2016) definen el PA como, una lógica manera organizada de pensar que se utiliza para dividir un objetivo complicado en una serie de pasos (ordenados) utilizando las herramientas disponibles.

La habilidad "implementar algoritmos" es elemental para programadores, así como para diseñadores y administradores de bases de datos. Aunque existen estudios sobre la complejidad e insuficiencia en la comprensión de los contenidos de programación, aún faltan investigaciones que reflejen los fundamentos teóricos que permiten sustentar cómo es la estructura interna de esta habilidad (Saez & Menéndez, 2015).

Se ha observado que los estudiantes se ven desmotivados ante el desafío que representa la resolución de problemas en el tema del desarrollo de algoritmos, debido al nivel abstracto del mismo y la falta de vinculación con problemas prácticos y significativos relacionados con temáticas de sus carreras (Chezzi et al., 2017).

En el ámbito de la investigación de Guerrero y García (2016) el problema que plantean es que los estudiantes de primer semestre de la carrera de Ingeniería en Sistemas Computacionales no manifiestan el desarrollo pleno de la competencia del pensamiento algorítmico, lo que representa un mayor índice de reprobación.

Mac Gaul de Jorge et al. (2008) identificaron al proceso de resolución de problemas como una situación que crea en el alumno un conflicto cognitivo en la medida que no dispone de un sistema de estrategias ampliamente constituido; de igual manera Salgado et al. (2013), en su investigación revelaron como problema la insuficiente aprehensión, por parte de los estudiantes de la Ciencia de la Computación, de los contenidos de programación, principalmente aquellos vinculados con la interpretación de situaciones problemáticas.

Sleeman nos dice que los problemas con la enseñanza y el aprendizaje de introducción a la programación se han documentado desde la década de 1980 y de acuerdo con Berking y Reilly: "Es bien sabido en la comunidad de Educación en Ciencias de la Computación (CSE) que los estudiantes tienen dificultades con los cursos de programación y esto puede ser el resultado de una alta tasa deserción y fracaso" (como se citó en Nikula et al., 2011, p. 24); sin embargo, como se enuncia en recientes investigaciones las dificultades persisten, así como la búsqueda de estrategias que permitan favorecer el aprendizaje de los algoritmos.

En el contexto de esta investigación el problema que se plantea es que estudiantes de segundo semestre de la carrera en Sistemas Computacionales Administrativos no demuestran el desarrollo pleno de las habilidades del pensamiento algorítmico, al mostrar dificultad en la resolución de problemas algorítmicos por el nivel de abstracción que se necesita en la comprensión de conceptos y en los pasos necesarios para alcanzar el objetivo deseado en un curso de Diseño e Implementación de Algoritmos.

De ahí la necesidad de realizar un estudio que permita a los estudiantes universitarios favorecer el desarrollo del pensamiento algorítmico a través del desarrollo de una estrategia tecnopedagógica.

Scratch como ambiente de programación

De acuerdo con Hermans (2020) los estudiantes aprenden a escribir programas usando programación textual en entornos como Java, Perl, lenguajes no fáciles de aprender; sin embargo, aprender en un lenguaje más simple como Phyton favorece el paso a un lenguaje más complejo como C++. La carga cognitiva para aprender sintaxis, recordar los comandos correctos y desarrollar programas llega a ser tan alto que un número significativo de estos estudiantes Ciencias de la Computación abandonan el curso, por lo que sugieren empezar a programas con una cantidad mínima de sintaxis, más fáciles para principiantes como Scrath, como señala la investigación de Resnick et al. (2009).

Autores como Tijani et al. (2020) buscaron obtener una atractiva experiencia en programación a través de Scratch, como base para el aprendizaje del lenguaje QBASIC, teniendo como resultado la comprensión de conceptos relevantes sin alcanzar la resolución de problemas más complejos. Scratch se basa en la teoría constructivista de Seymour Papert, la cual sostiene que los jóvenes aprenden mejor cuando están provistos de un ambiente donde puedan expresar su creatividad, compartiendo intereses con sus compañeros (Alanazi, 2019 citado en Campbell & Atagana, 2022).

Para Pérez-Narvaés et al. (2020), Scratch permite aprender a programar mediante la experimentación de forma creativa, aunado al desarrollo de habilidades cognitivas que favorecen el aprendizaje de los fundamentos de programación.

No ha sido hasta la última década que se han realizado investigaciones sobre la adquisición de habilidades de programación y pensamiento computacional en los estudiantes, utilizando Scratch como una herramienta útil para ello (Çatlak et al., 2015)

De acuerdo con Vidal et al. (2015), Scratch presenta un ambiente en el cual los estudiantes se motivan, sin temor al error, facilita el análisis de problemas y la propuesta y desarrollo de soluciones algorítmicas factibles de mejorar.

Partiendo de lo anterior y con base a la problemática planteada en párrafos anteriores, se establece como objetivo de la investigación evaluar el impacto del uso de Scratch para favorecer el desarrollo del pensamiento algorítmico en estudiantes de educación superior, lo que permitirá impulsar su uso, como una manera creativa y fácil de aprender que impulse el pensamiento algorítmico como competencia indispensable en la resolución de problemas algorítmicos.

2. MÉTODO

Tipo de investigación

Esta investigación tiene un enfoque cuantitativo de tipo experimental, puesto que se busca someter a un grupo de estudiantes a un tratamiento (estrategia tecnopedagógica) con el objetivo de observar los efectos producidos en el desarrollo de la competencia de resolución de problemas algorítmicos. En un enfoque cuantitativo se utiliza la recolección de datos para probar hipótesis con base en la medición numérica y el análisis estadístico, con el fin establecer pautas de comportamiento y probar teorías (Hernández et al., 2014).

Diseño de la investigación

Es un diseño pre-experimental, en el que los estudiantes están constituidos previo al experimento, en este estudio, los estudiantes cursaban en el segundo semestre la experiencia educativa de Diseño e Implementación de Algoritmos.

Se contempla un solo grupo con pretest-posTest (Cohen & Manion, 2002), de tipo cuasiexperimental con un preTest-posTest.

Actividades de aprendizaje

Las actividades de aprendizaje se implementaron a través de la plataforma de enseñanza distribuida EMINUS 4 de la Universidad Veracruzana.

En clases se habló de la aplicación de Scratch, como una herramienta muy eficaz para aprender a programar para principiantes. Luego se abordaron los componentes básicos del software como son los bloques de código, el área de escritura para estos bloques y el escenario. Durante el proceso se tuvo un enfoque constructivista, en el que a lo largo del proceso se proporcionó retroalimentación de manera individual y grupal. Los estudiantes diseñaron pequeños proyectos en Scratch, para ejemplificar el uso de cláusulas si- entonces y bucles, como se muestra en las figuras 1, 2 y 3.

Figura 1

Una captura de pantalla de una actividad estudiantil con bloques de código

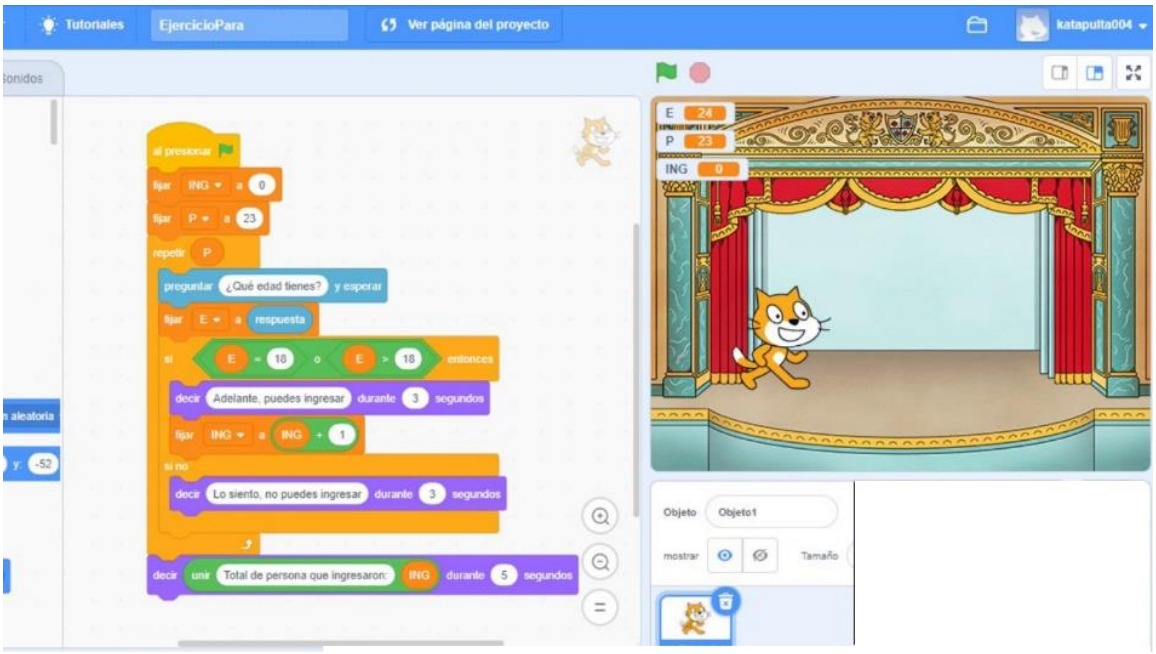


Figura 2

Una captura de pantalla de una actividad estudiantil

Figura 3

Una captura de pantalla de una actividad estudiantil

Grupo de estudio

Este estudio se llevó a cabo en la Universidad Veracruzana, en la Facultad de Contaduría y Administración, campus Xalapa, Veracruz, México, durante el período Febrero-junio de 2023; específicamente en el programa educativo de Licenciado en Sistemas Computacionales Administrativos. Se seleccionó un grupo por conveniencia de 48 estudiantes de segundo semestre, que cursaban la experiencia educativa de Diseño e implementación de algoritmos. Los estudiantes contaban ya con conocimientos básicos de Fundamentos de Álgebra, Pensamiento Crítico para la Resolución de Problemas; así como Fundamentos de las Tecnologías de Información. El grupo estuvo conformado por un 21% de mujeres y 79% de hombres, de edades comprendidas entre 18 y más años, sin considerar un límite de edad

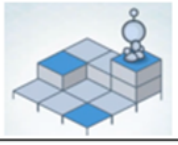
Desarrollo del instrumento

Se seleccionó una prueba de pensamiento algorítmico diseñado por Posso y Murcia (2022), quienes elaboraron y validaron la prueba, teniendo en cuenta algunos aspectos del pensamiento algoritmo relacionados con la programación de computadoras como fueron las secuencias, las condiciones y los ciclos. El instrumento también maneja diagramas de flujo de datos, este se presenta en la figura 4.

Figura 4

Test de Pensamiento Algorítmico

#	PROBLEMA		
1	Las instrucciones que hacen que el robot llegue al cuadrado azul son: 	a. 	b. 
		c. Cualquiera de la dos instrucciones hacen que el robot llegue al cuadrado azul.	d. Ninguna de las dos opciones permite darle solución al problema.

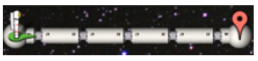
2	PROBLEMA	a.	b.
	Las instrucciones que hacen que el robot pase por los 3 cuadrados azules son: 		
		c. Cualquiera de la dos instrucciones hacen que el robot llegue a los cuadrados	d. Ninguna de las dos opciones permite darle solución al problema.

3	PROBLEMA	a.	b.
	 Cuáles son las instrucciones para que el personaje llegue a su destino, es decir, al globito rojo.		
		c. 	d. 

Figura 4A

CICLOS O REPETICIONES

Cuando se necesita repetir una instrucción se puede simplificar las instrucciones. Esto se puede hacer en programación con la instrucción REPETIR, gráficamente puede ser un bucle.



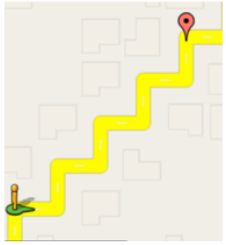
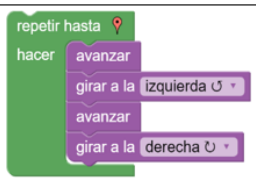
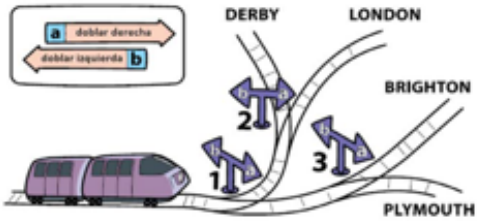
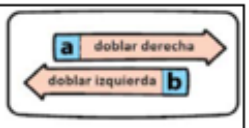
4	PROBLEMA	OPCIONES DE RESPUESTA	
		a.	b.
	Las instrucciones para que el personaje llegue a su destino son: 		
		c. 	d. 

Figura 4B

5	PROBLEMA	a.	b.
	Las instrucciones que **NO** permitirían que el personaje llegue a su destino son: 		
		c.	d.
			

Figura 4C

#	PROBLEMA				
7	 *Imagen tomada de documento Brebras.	La flechas de tráfico controlan los movimientos del tren en cada cruce.  ¿Qué par de instrucciones permitirá que el tren llegue a Londres(London)?			
		a.	b.	c.	d.
		1a y 3a	1b y 2a	1a y 2b	1b y 3a

DIAGRAMAS DE FLUJO

Observa bien el siguiente algoritmo computacional que controla el ingreso de los niños a una Montaña Rusa:

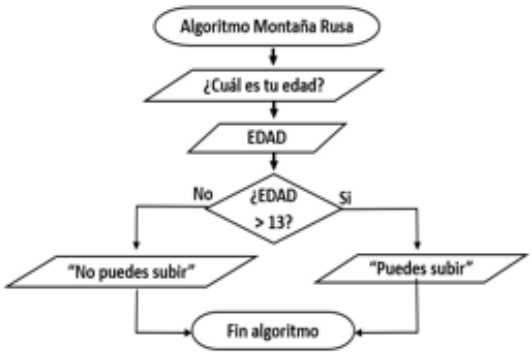
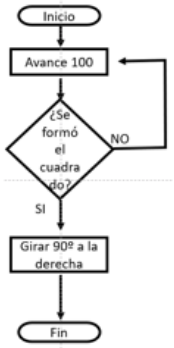
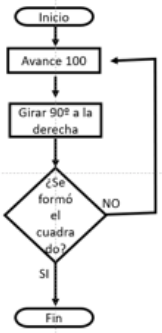


Figura 4D

8	PROBLEMA	OPCIONES DE RESPUESTA			
		a.	b.	c.	d.
	Teniendo en cuenta el diagrama anterior, el cual controla con computadora la subida de los niños(as) a la Montaña Rusa, ¿Qué sucedería si la edad del niño es igual a 13?	Puede subir a la Montaña Rusa	No puede subir a la Montaña Rusa	Puede subir, pero acompañado de un adulto	Puede subir mientras tenga en cuenta las recomendaciones.

9	PROBLEMA	a.	b.
	El diagrama de flujo que permitiría dibujar un cuadrado cuyo lado mide 100 sería: (el triángulo índica el punto de inicio). 		
		c.	d.
		Cualquiera de los dos diagramas permite hacer el cuadrado	Ninguno de los dos diagramas permite hacer el cuadrado

#	PROBLEMA	El diagrama de flujo nos representa el proceso de compra de un producto en un supermercado. Las subtareas ordenadas del proceso denominado **PAGAR PRODUCTO** son:			
10	 Figura 4E	a.	b.	c.	d.
		Ir a caja, pasar producto, leer código prodcuto, mostrar valor en pantalla, entregar dinero, dar cambio, entregar factura, verificar el cambio, salir del supermercado	Pasar producto, leer código, entregar factura, entregar dinero, dar cambio, salir del supermercado	Pasar producto a cajero, leer código producto, mostrar valor en pantalla, entregar dinero, dar cambio, entregar factura, verificar cambio.	Entrar al supermercado, dirigirse a coger el producto, ir a caja, pasar dinero, esperar cambio, salir del supermercado.

Nota. Extraído de Posso y Murcia (2022).

3. RESULTADOS

El desarrollo del pensamiento algorítmico puede ser poco efectivo por una serie de factores, tales como la concepción de conceptos erróneos: conocimiento o ideas que debilitan el proceso real de resolución de problemas, o un inconsistente conocimiento; conceptos que la persona distingue, pero no aplica de forma correcta y efectiva.

Los datos obtenidos a partir de la administración de la prueba de pensamiento algorítmico fueron analizados y procesados en Excel para su codificación y posterior uso en el programa estadístico SPSS.

La prueba estadística seleccionada inicialmente para probar la hipótesis de investigación fue t de Student; una de las premisas en las que se basa esta prueba es en la distribución normal de los datos (Turcios, 2015), por lo tanto, se validó la distribución normal de estos a través de la prueba Shapiro-Wilk que se utiliza para muestras menores a 50 (Novales, 2010).

Para probar si los datos tienen una distribución normal, se tiene que:

H0: Los datos del PreTest y los datos del PosTest tienen una distribución normal

H1: Los datos del PreTest y los datos del PosTest no tienen una distribución normal

Los resultados que se obtienen se pueden observar en la tabla 1 Pruebas de normalidad PreTest y PosTest

Tabla 1

Pruebas de normalidad PreTest y Postest

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	Gl	Sig.	Estadístico	Gl	Sig.
PreTest	.193	48	.000	.940	48	.016
PosTest	.171	48	.001	.939	48	.014

Nota. Corrección de significación de Lilliefors

Como se puede percibir el valor de la significación estadística p tiene los valores .016 con los datos del PreTest y .014 con los datos obtenidos en el PosTest, los cuales son menores a 0.05 por lo que se rechaza la hipótesis nula, lo que significa que la distribución de los datos no es normal (López, 2016); en consecuencia, se procede a obtener la prueba Wilcoxon, que es una prueba no paramétrica que sustituye a la prueba t de Student para muestras relacionadas cuando la distribución normal de los datos no se cumple (Gómez-Gómez et al., 2003).

Con una hipótesis nula:

> H0: El uso de Scratch no favorece el desarrollo del pensamiento algorítmico de los estudiantes de la Licenciatura en Sistemas Computacionales Administrativos de la Facultad de Contaduría y Administración de la Universidad Veracruzana.

Y la hipótesis de investigación:

> H1: El uso de Scratch favorece el desarrollo del pensamiento algorítmico de los estudiantes en la Licenciatura en Sistemas Computacionales Administrativos en la Facultad de Contaduría y Administración.

La tabla 2 muestra el estadístico Wilcoxon(Z) y su nivel crítico bilateral (Sig. Asint.Bilateral).

Tabla 2

Prueba de Wilcoxon

Z	-3.731
Sig. asintótica(bilateral)	.000

Se obtiene un valor de significación estadística de .000, que sería el valor de p, el cual es menor a 0.05, demostrando una diferencia estadísticamente significativa entre los datos obtenidos, tal como señalan Berlanga y Rubio (2012), lo que permite concluir que el nivel de pensamiento algorítmico es diferente entre la primera y segunda medición, rechazando la hipótesis nula, aceptando la hipótesis de nuestra investigación, la cual señala que el uso de Scratch favorece el desarrollo del pensamiento algorítmico de los estudiantes en la Licenciatura en Sistemas Computacionales Administrativos en la Facultad de Contaduría y Administración.

4. DISCUSIÓN

El aprendizaje de la disciplina Algorítmica representa un gran reto para los estudiantes universitarios en el área de Ciencias de la Computación. El principal obstáculo que afrontan los estudiantes es la falta de capacidad y flexibilidad en la resolución de problemas, lo que involucra establecer una secuencia de pasos elementales, cada uno de los cuales genera un nuevo conocimiento, el cual resulta de la inferencia lógica a partir de los conocimientos y experiencias del estudiante y de las condiciones del problema (Salgado et al., 2013).

Al aplicar la estrategia tecnopedagógica con problemas cercanos al contexto profesional del estudiante, de una manera divertida y contando con la colaboración entre sus compañeros se buscó fortalecer el pensamiento algorítmico, ya que como señala Biju (2019) la enseñanza en el aula no fomenta las habilidades de pensamiento crítico para la resolución de problemas. Un maestro tiene muchas cosas que enseñar, dentro de un tiempo asignado, con poca o ninguna interacción del lado de los estudiantes. También hay otras deficiencias de la enseñanza en el aula, muchos estudiantes pueden quedarse atascados mientras realizan conjuntos de problemas en casa, por lo que la colaboración entre pares o en equipo es una opción factible a través de la cual los estudiantes más débiles pueden desempeñarse mejor si se les proporciona una interacción uno a uno.

Los hallazgos contenidos en esta investigación permitieron determinar que el uso de Scratch fue efectivo en las habilidades del pensamiento algorítmico, siendo consistente con estudios anteriores (Campbell & Atagana, 2022; Buyukkarci & Taslidere, 2021; İlic, 2021; Altun & Kasalak, 2018; Pérez-Narvaés et al., 2020), a través de los cuales se observó que Scratch contribuía a la resolución de problemas y a las habilidades del pensamiento algorítmico.

Elegir Scratch como estrategia para facilitar el desarrollo del pensamiento algorítmico reafirmó lo que señala la revisión de la literatura, tal como lo exponen Ritter y Standl (2022), que en la enseñanza de programación la búsqueda de estrategias para enseñar habilidades en la resolución de problemas ha recibido principal atención sobre todo aquellas que se centran en el constructivismo.

Aunque algunos estudios sobre el uso de Scratch en la enseñanza de programación señalan resultados positivos, sin embargo, otros advierten que Scratch no tiene efectos significativos sobre todo en aquellos estudiantes con alguna experiencia en programación, de acuerdo con un estudio realizado en la Universidad de Harvard por Malan y Leitner (2007), lo que expone la importancia de continuar con estudios sobre la aceptación del uso Scratch en estudiantes de introducción a la programación.

La presente investigación es un primer acercamiento al uso de Scratch como estrategia tecnopedagógica para favorecer el pensamiento algorítmico de los estudiantes de la licenciatura en Sistemas Computacionales Administrativos, puesto que el diseño preexperimental de la investigación permite tener una aproximación del fenómeno de estudio, y medir los efectos de una variable en un grupo pequeño de estudiantes (Ramos-Galarza, 2021). El diseño preexperimental es utilizado con frecuencia en la investigación en educación, ya que, aunque con limitaciones como la falta de control de validez interna y solo se aplican cuando no es posible manipular más de una condición de la variable independiente; si son bien utilizados pueden ser de gran utilidad principalmente en la investigación aplicada (Salas, 2013).

Continuando con la investigación de Salas (2013), la autora también menciona que los diseños preexperimentales son considerados importantes y de utilidad por investigadores médicos, en el campo educativo y en el de la psicología, debido a que los participantes de la investigación casi nunca son elegidos al

azar ni son representativos de la población, además de que cuentan con una característica importante: cumplen con la mínima condición de un experimento, la manipulación de una variable independiente.

Lo anterior permite afirmar que esta investigación significa un primer paso para establecer la evidencia a favor o en contra de una intervención, lo que permitirá justificar un estudio a mayor escala.

Es por ello importante señalar que el diseño de una investigación constituye un aspecto importante en todo proceso científico para dar respuestas adecuadas a las preguntas planteadas por un investigador, ayudando a resolver un problema planteado especificado en una hipótesis, de ahí la importancia de los resultados obtenidos en el presente trabajo.

5. CONCLUSIONES

Se logró fortalecer el desarrollo del pensamiento algorítmico en estudiantes de segundo semestre de la carrera en Sistemas Computacionales Administrativos inscritos en el curso de Diseño e Implementación de Algoritmos; a través del uso de Scratch se facilitaron los temas del contenido del curso, logrando la atención y colaboración de los estudiantes.

Aunque existen otras herramientas utilizadas para apoyar a estudiantes que se inician en el desarrollo de programas o en el diseño de algoritmos computacionales, como el software PseInt, también utilizado en el mejoramiento de los niveles cognitivos de aprendizaje de un curso de Algoritmos (Estrada, 2016). Este trabajo acredita que Scratch a través de los recursos multimedia como sonidos, objetos gráficos y movimiento es una herramienta adecuada en el fortalecimiento del pensamiento algorítmico y que es posible utilizarlo como apoyo en una clase de Diseño e Implementación de Algoritmos.

Resolver problemas es esencial en la disciplina de programación, pero que demanda la capacidad de integrar y aplicar una serie de conceptos básicos, habilidades cognitivas y estilos de pensamiento. Desarrollar habilidades del pensamiento algorítmico para resolver problemas de manera metodológica es trascendental para el desarrollo de algoritmos como un paso previo a la creación de programas de computadora.

Es por ello por lo que se ha evidenciado la necesidad de cursos para iniciar a programar alineados en la resolución de problemas algorítmicos que introduzcan el pensamiento creativo, el razonamiento lógico y las capacidades de resolución de problemas en los estudiantes como habilidades imprescindibles para aprender a programar.

Como trabajo futuro se plantea poner en práctica la estrategia tecnopedagógica propuesta en un diseño de investigación cuasiexperimental en el que se tenga un grupo de control y otro en el que se aplique la estrategia, de modo que se pueda comparar los resultados de aprendizaje entre un grupo y otro, para continuar evaluando la eficacia de la estrategia propuesta.

Conflicto de intereses / Competing interests:

Los autores declaran que el presente proyecto no representa ningún conflicto de interés real, potencial o evidente, de carácter personal, con la revista, la entidad editora y las entidades financiadoras.

Rol de los autores / Authors Roles:

María Velasco-Ramírez: Conceptualización, curación de datos, análisis formal, investigación, metodología, recursos, software, supervisión, validación, visualización, administración del proyecto, escritura-preparación del borrador original, escritura-revisión y edición.

Alma Otero-Escobar: Conceptualización, análisis formal, investigación, metodología, supervisión, validación, visualización, administración del proyecto, escritura-revisión y edición.

Aspectos éticos/legales:

Los autores declaran no haber incurrido en aspectos antiéticos, ni haber omitido aspectos legales en la realización de la investigación.

Fuentes de financiamiento / Funding:

Las fuentes de financiación que dieron lugar a la investigación son de carácter personal, motivación profesional y académica.

REFERENCIAS

Altun, A., & Kasalak, İ. (2018). Perceived self-efficacy scale development study related to block-based programming: Scratch case. *Educational Technology Theory and Practice, 8*(1), 209-225. https://doi.org/10.17943/etku.335916

Babori, A., Fassi, H., F., Hariri, A. & Bideq, M. (2016). An e-Learning environment for algorithmic: Toward an active construction of skills. *World Journal on Eductional Technology: Current Issues. 8*(2), 82-90.

Berlanga Silvestre, V., y Rubio Hurtado, M. J. (2012). Clasificación de pruebas no paramétricas. Cómo aplicarlas en SPSS. *Revista d'Innovació i Recerca En Educació, 5*(2), 101–113. https://doi.org/10.1344/reire2012.5.2528

Biju, S. M. (2019). Benefits of Working in Pairs in Problem Solving and Algorithms - Action Research. *Athens Journal of Education, 6*(3), 223–236. https://doi.org/10.30958/aje.6-3-4

Blanco-Hamad, A., Salgado, A. & Alonso, I. (2016). Habilidades para la algoritmización computacional en la Licenciatura en Educación: Especialidad Educación Laboral-Informática. *Revista Maestro y Sociedad, 13*(1), pp. 16-28.

Buyukkarci, A., & Taslidere, E. (2021). The Effect of Coding Education on Students' Efficiency and Scratch Achievements. *i-manager's. Journal of Educational Technology, 18*(2), 63-74. https://doi.org/10.26634/jet.18.2.17970

Byrka, M.F., Sushchenko, A.V., Svatiev, A.V. Mazin, V.M., & Veritov, O.I. (2021). A New Dimension of Learning in Higher Education: Algorithmic Thinking. *Propósitos y Representaciones, 9*(SPE2), e990. http://dx.doi.org/10.20511/pyr2021.v9nSPE2.990

Campbell, O. O., & Atagana, H. I. (2022). Impact of a Scratch programming intervention on student engagement in a Nigerian polytechnic first-year class: verdict from the observers. *Heliyon, 8*(3). https://doi.org/10.1016/j.heliyon.2022.e09191

Çatlak, S., Tekdal, M., & Baz, F. Ç. (2015). The status of teaching programming with scratch: a document review work. *Journal of Instructional Technologies & Teacher Education, 4*(3), 13-25.

Cohen, L., & Manion, L. (1990). *Métodos de Investigación educativa*. La Muralla, S.A.

Chezzi, C. M., Salvarredi, M., Casañas, F. A., Giupponi, D. M. M., & Anzardi, A. I. (2017). *Estrategia de motivación para el razonamiento de algoritmos computacionales mediante juegos*. Repositorio Institucional Abierto. http://hdl.handle.net/20.500.12272/2266

Estrada Aro, W. M. (2016). *Software Pseint en los niveles cognitivos en estudiantes del curso Principios de Algoritmos de la Universidad Tecnológica del Perú – Lima*. [Tesis Doctoral, Universidad Tecnológica del Perú]. Repositorio de la Universidad Tecnológica del Perú https://hdl.handle.net/20.500.12692/4198

Gómez-Gómez, M, Danglot-Banck, C., & Vega-Franco, L. (2003). Sinopsis de pruebas estadísticas no paramétricas. Cuando usarlas. *Revista Mexicana de Pediatría. 70*(2),91-99.

Güler, Ç. (2021). Algorithmic Thinking Skills without Computers for Prospective Computer Science Teachers. *Journal of Theoretical Educational Science, 14*(4), 570-585. https://doi.org/10.30831/akukeg.892869

Guerrero Posadas, M., & García Orozco, J. (2016). Desarrollo Del Pensamiento Algorítmico Con El Apoyo De Objetos De Aprendizaje Generativos. *Píxel-Bit, Revista de Medios y Educación, 49*, 163–175. https://doi.org/10.13140/RG.2.1.4709.5921

Hermans, F. (2020). Hedy: A Gradual Language for Programming Education. *ICER 2020 - Proceedings of the 2020 ACM Conference on International Computing Education Research*, 259–270. https://doi.org/10.1145/3372782.3406262

Hernández, R., Fernández, C. y Baptista, P. (2014). *Metodología de la investigación*. MacGrawHill.

Hubalovsky, S., & Korinek, O. (2015). Evaluation of Algorithmic Thinking of Students Using Control Testing Environment. *International Journal of Education and Information Technologies, 9*.

İlic, U. (2021). The impact of Scratch-assisted instruction on computational thinking (CT) skills of pre-service teachers. *International Journal of Research in Education and Science (IJRES), 7*(2), 426-444. https://doi.org/10.46328/ijres.1075

Lockwood, E., Asay, A., DeJarnette, A. F., & Thomas, M. (2016). Algorithmic thinking: An initial characterization of computational thinking in mathematics. *38th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, 1588–1595.

López, P. (2016). Prueba de bondad de ajuste a una distribución normal. *Revista Enfermería Del Trabajo, 6*(3), 105–114.

Mac Gaul de Jorge, M. I., López, M. F., & Olmo, A. P. D. (2008, 12-13 de junio). Resolución de problemas computacionales: análisis del proceso de aprendizaje [Sesión de Conferencia]. *Congreso de Tecnología en Educación y Educación en Tecnología*. Argentina, Buenos Aires. http://sedici.unlp.edu.ar/handle/10915/19049

Malan, D. J., & Leitner, H. H. (2007). Scratch for Budding Computer Scientists Terms of Use Scratch for Budding Computer Scientists. *ACM SIGCSE Bulletin 39*(1), 223–227.

Martín, D., & Calvillo, A. J. (2017). *The Flipped Learning: Guía "Gamificada" para novatos y no tan novatos.* Universidad Internacional de La Rioja.

Nikula, U., Gotel, O., & Kasurinen, J. (2011). A Motivation Guided Holistic Rehabilitation of the First. *ACM Transactions on Computing Education, 11*(4), 1- 38. https://doi.org/10.1145/2048931.2048935

Novales, A. (2010). *Análisis de regresión*. Universidad Complutense de Madrid.

Pérez-Narváez, H. O., Roig Vila, R., y Jaramillo Naranjo, L. (2020). Uso de SCRATCH en el aprendizaje de Programación en Educación Superior. *Cátedra, 3*(1), 28–45. https://doi.org/10.29166/10.29166/catedra.v3i1.2006

Posso, M.E. & Murcia, E. (2022). *Las "actividades desconectadas" y el desarrollo del pensamiento algorítmico* [Tesis de Maestría, Universidad Católica de Pereira]. Repositorio de la Universidad Católica de Pereira http://hdl.handle.net/10785/9635.

Ramos-Galarza, C. (2021). Editorial: Diseños de investigación experimental. *CienciAmérica, 10*(1), 1–7. https://doi.org/10.33210/ca.v10i1.356

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Ritter, F., & Standl, B. (2023). Promoting Student Competencies in Informatics Education by Combining Semantic Waves and Algorithmic Thinking. *Informatics in Education, 22*(1), 141–160. https://doi.org/10.15388/infedu.2023.07

Saez, A., Febe, C., Puentes, U., & Menéndez, J. (2015). El desarrollo de la habilidad: implementar algoritmos. Teoría para su operacionalización. *Revista Cubana de Ciencias Informáticas, 9*(3), 99–112.

Salas, E. (2013). Diseños preexperimentales en psicología y educación: Una revisión conceptual. *Liberabit revista de psicología, 19*(1), 133-141.

Salgado, A., Alonso, I., Sánchez, G., & Tardo, Y. (2013). Didáctica de la resolución de problemas de Programación Computacional. *Pedagogía Universitaria, 18*(4). 62-74.

Tijani, F., Callaghan, R., & de Villers, R. (2020). An Investigation into Pre-service Teachers' Experiences While Transitioning from Scratch Programming to Procedural Programming. *African Journal of Research in Mathematics, Science and Technology Education*, *24*(2), 1–13. https://doi.org/10.1080/18117295.2020.1820798

Turcios, R. A. S. (2015). T-Student: Usos y abusos. *Revista Mexicana de Cardiologia, 26*(1), 59–61.

Ülker, E. D. (2020). The effect of applying 4-stages on learning analysis and design of algorithms. *Cypriot Journal of Educational Sciences, 15*(5), 1238–11248. https://doi.org/10.18844/CJES.V15I5.4621

Vidal, C. L., Cabezas, C., Parra, J. H., & López, L. P. (2015). Experiencias prácticas con el uso del lenguaje de programación scratch para desarrollar el pensamiento algorítmico de estudiantes en Chile. *Formación Universitaria, 8*(4), 23–32. https://doi.org/10.4067/S0718-50062015000400004