PONTIFICIA UNIVERSIDAD

CATÓLICA DEL PERÚ

Escuela de Posgrado

ANÁLISIS DE LAS PRAXEOLOGÍAS MIXTAS QUE SE PRESENTAN EN LAS TAREAS MATEMÁTICAS CON SCRATCH ASOCIADO A LA SECUENCIA FIGURAL

Tesis para obtener el grado académico de Maestra en Enseñanza de las Matemáticas que presenta:

Meliza Inés Malpartida Rodríguez

Asesor:

Cintya Sherley Gonzales Hernández

Lima, 2024

Informe de Similitud

Yo, Cintya Sherley Gonzales Hernández, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesora de la tesis titulada: Análisis de las praxeologías mixtas que se presentan en las tareas matemáticas con scratch asociado a la secuencia figural, de la autora Meliza Malpartida, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 12%. Así lo consigna el reporte de similitud emitido por el software Turnitin el 21/01/2025.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:

21 de enero del 2025

<table>
<tr><td colspan="2">Apellidos y nombres de la asesora: Gonzales Hernández Cintya Sherley</td></tr>
<tr><td>DNI:42741882</td><td rowspan="2">Firma:
  </td></tr>
<tr><td>ORCID: 0000-0003-2130-1710</td></tr>
</table>

Dedicatoria

A mi esposo Jimmy Tamara por darme siempre fuerza

y valor para terminar este trabajo.

A mis hijos Caleb y Abigail por ayudarme a ser mejor persona cada día

.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi asesora, Mg. Cintya Gonzales, por su incondicional apoyo, paciencia y dedicación a lo largo de todo el proceso de desarrollo y culminación de esta tesis. Su orientación y compromiso han sido fundamentales para llevar a cabo este trabajo.

A la Dra. Noemí Ruiz, por sus acertadas sugerencias y observaciones, las cuales fueron esenciales para mejorar y profundizar la investigación, permitiéndome alcanzar una versión más sólida de este trabajo.

A la Dra. Cecilia Gaita, por su tiempo, dedicación y valiosas observaciones durante el proceso de evaluación de este trabajo. Agradezco profundamente sus sugerencias y comentarios, los cuales han sido de gran ayuda para enriquecer y profundizar mi investigación.

A mi familia, por su constante acompañamiento, comprensión y paciencia frente a mis ausencias, su apoyo incondicional ha sido una fuente de fortaleza durante este tiempo.

Finalmente, quiero agradecer a la línea de investigación de Epistemología de las Matemáticas en la Didáctica de las Matemáticas, en particular al área de la Antropología del Conocimiento Matemático y el Diseño de Secuencias Didácticas, por haber aportado conocimientos clave a mi formación profesional y haber impulsado mi desarrollo académico.

Resumen

La presente investigación analiza las praxeologías mixtas presentes en las tareas matemáticas realizadas con Scratch, enfocadas en la secuencia figural. Para ello, se utilizaron materiales y recursos extraídos de investigaciones previas que presentaban ejercicios implementados con Scratch, centrados en nuestro objeto matemático. La problemática que originó a este estudio se basa en determinar qué dominios de conocimiento están presentes en las nuevas praxeologías del pensamiento computacional cuando se relaciona con las matemáticas, en particular, la generalización de patrones, principalmente debido al predominio de estrategias aritméticas y recursivas, con enfoques visuales o explícitos. Para responder a esta problemática se adoptó la Teoría Antropológica de lo Didáctico y el modelo praxeológico T4TEL como marco teórico y metodológico. A través de este modelo teórico se analizó la organización matemática de la secuencia figural, que resultó clave para comprender y describir la organización matemática mixta. Esta praxeología de referencia facilitó el análisis praxeológico asociado a la secuencia figural implementada en el entorno de Scratch. Posteriormente, con los elementos teóricos proporcionados por el marco, se analizaron diversos ejercicios con Scratch relacionados con el objeto matemático en estudio. Se lograron identificar 10 tipos de tareas, que clasificamos en 8 subtipos dentro del tipo de tarea $\boldsymbol{T_{prog}}[\boldsymbol{T_{SF}}]$ "Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_{SF}}$", siendo $\boldsymbol{T_{SF}}$ un tipo de tarea de la secuencia figural. Para esto, se basó en las variables previamente identificadas. Asimismo, se reconocieron ingredientes técnicos clave en esta praxeología mixta, por ejemplo, el más usado es ${T^*}_m[unid.\,de\,patrón]$: identificar la unidad de patrón en el programa de Scratch.

Palabras clave: Secuencias figurales, Scratch, praxeologías mixtas, TAD

Abstract

This This research analyzes the mixed praxeologies present in mathematical tasks performed with Scratch, focusing on figural sequences. For this, materials and resources extracted from previous research presenting exercises implemented with Scratch, centered on the mathematical object, were used. The problem that originated this study is based on determining which knowledge domains are present in the new praxeologies of computational thinking when related to mathematics, in particular, the generalization of patterns, mainly due to the predominance of arithmetic and recursive strategies, with visual or explicit approaches. Additionally, the Anthropological Theory of the Didactic and the praxeological model T4TEL were adopted as the theoretical and methodological framework. Through this theoretical model, the mathematical organization of the figural sequence was analyzed, which proved key in understanding and describing the mixed mathematical organization. This reference praxeology facilitated the analysis of the didactic sequence associated with the figural sequence implemented in the Scratch environment. Subsequently, with the theoretical elements provided by the framework, various articles presenting Scratch exercises related to the mathematical object under study were analyzed. Ten types of tasks were identified, which we classified into eight subtypes within the task type $\boldsymbol{T_{prog}}[\boldsymbol{T_{SF}}]$ "Develop a program in Scratch that executes T_SF," with $\boldsymbol{T_{SF}}$ being a type of task for the figural sequence. For this, the previously identified variables were used. Furthermore, seven key technical ingredients in this mixed praxeology were recognized, for example, the most used is $T^*{}_m[\text{pattern unit}]$: identify the pattern unit in the Scratch program.

Keywords: Figural sequences, Scratch, mixed praxeologies, ATD.

Índice

Lista de tablas

Lista de figuras

Introducción

En la actualidad, el pensamiento computacional (PC) se ha consolidado como una habilidad fundamental en la educación, permitiendo abordar problemas complejos mediante la descomposición en partes más simples, el reconocimiento de patrones y la formulación de soluciones lógicas. Este enfoque no solo es esencial en la programación, sino que también se aplica en diversas disciplinas, como las matemáticas.

Dentro de este marco, la programación por bloques se presenta como una herramienta pedagógica clave, especialmente cuando se trabaja con secuencias figurales**,** en las que se requieren habilidades para analizar, extender y modificar patrones geométricos. Herramientas como Scratch facilitan este proceso, al ofrecer un entorno visual e interactivo que permite representar y manipular estos patrones de manera intuitiva.

La búsqueda de la sinergia entre el pensamiento computacional y las prácticas matemáticas está siendo continuamente investigada (Bråting y Kilhamn, 2021; Broley et al., 2023; Chaachoua et al., 2020). Esta necesidad ha hecho emerger una diversidad de recursos curriculares de matemáticas que incluye el PC; sin embargo, continua el interés en determinar esa integración en lugar de analizar cómo se presentan las propuestas en los recursos didácticos que están disponibles (Elicer et al., 2023).

Es importante comprender que se combinan conocimientos teóricos y habilidades prácticas en la resolución de problemas. Sin embargo, los estudiantes no pueden ver dichas conexiones, es necesario que los profesores sean mediadores de la buena elección de tareas donde el propósito es aprender matemática (Nyman et al., 2024).

En este estudio, se explora cuáles son las praxeologías mixtas a través de la programación con Scratch para abordar problemas relacionados con secuencias figurales tomados de ejercicios propuestos de artículos de investigación.

El objetivo de este trabajo es describir y analizar las praxeologías mixtas que se presentan en las tareas matemáticas con Scratch asociadas a la secuencia figural en

Educación Primaria. Para ello, se cumplieron tres objetivos específicos: (a) Determinar una organización matemática de referencia para la secuencia figural, (b) Identificar las organizaciones matemáticas mixtas asociadas a la secuencia figural en tareas matemáticas realizadas en Scratch, y (c) Construir un modelo praxeológico de referencia relacionado a las tareas matemáticas realizadas en Scratch asociado a la secuencia figural.

La presente investigación se estructura en seis capítulos. El primer capítulo se centra en exponer la problemática de la investigación. Por ende, se aborda los antecedentes relevantes, que incluye estudios sobre el pensamiento computacional, las matemáticas y el objeto matemático: secuencia figural. Además, se presenta la justificación del estudio, se formula la pregunta de investigación y se definen los objetivos planteados para el trabajo.

En el segundo capítulo, se describen los fundamentos teóricos de la Teoría Antropológica de lo Didáctico (TAD) y el modelo praxeológico T4TEL propuesto por Chaachoua (2020). También, se aborda la definición de Organización Matemática Mixta y se explican los procedimientos metodológicos seguidos para alcanzar los objetivos de la investigación.

En el tercer capítulo, se presentan algunos aspectos de la programación con especial énfasis en el lenguaje de programación por bloques como el Scratch, que se utilizará en nuestro estudio.

En el cuarto capítulo, se realiza un análisis del objeto matemático: secuencia figural que permite la construcción de una praxeología de referencia de este objeto.

En el quinto capítulo, se aborda un análisis praxeológico de la secuencia figural presente en artículos de investigación y textos. Además, se describe el modelo de Organización Matemática Mixta, definido por la praxeología de referencia y por las praxeologías algorítmicas identificadas durante el análisis de la secuencia didáctica.

En el sexto capítulo, se realiza un análisis detallado de los ejercicios extraídos de dos artículos clave: el primero, de Alayrangues et al. (2019), y el segundo, de Ye et al. (2024). En estos estudios se describe la organización de los elementos relacionados con la secuencia figural, y dicho análisis se enmarca dentro de la teoría T4TEL.

Finalmente, en esta misma sección, se presentan los resultados obtenidos, se ofrecen consideraciones finales y se proponen recomendaciones para futuras investigaciones sobre el estudio de la secuencia figural.

CAPÍTULO I

MARCO CONTEXTUAL: PROBLEMÁTICA

En el presente capítulo, se aborda una revisión de las investigaciones más importantes sobre el pensamiento computacional y el objeto matemático de la secuencia figural. Asimismo, en este presente capítulo, se formula la pregunta de investigación y se plantean los objetivos que este estudio pretende alcanzar.

1.1 Investigaciones de referencia

Investigaciones relacionadas al pensamiento computacional y la Matemática

En el trabajo de Miller (2019), se investiga cómo la codificación informática facilita la identificación de patrones y estructuras matemáticas como descomposición, abstracción y creación de algoritmos. El trabajo incluyó el desarrollo de dos tareas que utiliza el programa visual Scratch: dibujar un cuadrado y un espirolateral.

En el análisis de la investigación, se observó que los estudiantes, en la primera tarea, identificaron los patrones repetitivos en los códigos. Luego, dedujeron una regla general para calcular el perímetro de un cuadrado. Con respecto a la segunda tarea, lograron identificar la estructura del espirolateral que presenta patrones crecientes y repetitivos en el código.

Los resultados del estudio indican que hay una conexión entre la capacidad de identificar la unidad de repetición de un patrón y la capacidad para aplicar este entendimiento en contextos matemáticos, que está vinculada al pensamiento computacional.

Sobre las conclusiones, el autor señala que la codificación informática potencia la habilidad de los estudiantes para identificar patrones, reconocer las unidades de repetición, deducir patrones y abstraer estructuras generales. Estos son componentes claves del pensamiento computacional. También, concluye que la codificación informática representa un nuevo medio para que los estudiantes puedan reconocer, desarrollar, y aplicar patrones y estructuras que facilitan la generalización en estructuras matemáticas.

Otra investigación relevante es la Haspekian et al. (2023) quienes investigan cómo la tecnología digital (Scratch, Excel y GeoGebra) transforma el aprendizaje de los conceptos algebraicos. Ellos utilizan como marco teórico la noción de distancia instrumental entre entornos, y destacan la diversidad de técnicas y conceptos generales que emergen al resolver tareas en diferentes plataformas.

Los autores manifiestan que los entornos digitales (Scratch, Excel y GeoGebra) para algunos usuarios podrían simplificarse la resolución de los ejercicios matemáticos y para otros, convertirse de forma compleja. Ello dependerá qué tan familiarizado esté el usuario con los programas. Asimismo, los investigadores indican que al usar la tecnología digital existen probabilidades que no se pueda entender el proceso de las resoluciones de los problemas, pues simplifican pasos. Sin embargo, esto no sucede cuando se desarrolla en un entorno de papel y lápiz. Por lo tanto, ellos mencionan que es necesario que la distancia instrumental (desfamiliarización digital) sea mínima.

Así mismo, muestran los objetos y las técnicas del Álgebra desde el punto de vista algoritmico y de la aritmética generalizada. Los objetos que se presentan son las variables, las expresiones y las ecuaciones. En cambio, las técnicas son aquellas relacionadas con la resolución de ecuaciones, todo ello en el entorno de papel y lápiz. Los autores definen a las variables como incógnita, cantidad variable, número generalizado y parámetro. Por otra parte, las expresiones algebraicas son aquellas que constan de variables, coeficientes y constantes, signos de operación de suma, resta, multiplicación, división y exponenciación. Con respecto a las ecuaciones algebraicas, constan de dos expresiones algebraicas con un signo igual entre ellas.

Los autores afirman que los entornos digitales pueden cambiar el significado de los objetos, influir en las técnicas, y alterar la conexión y el equilibrio entre los objetos, las técnicas y las representaciones. Además, manifiestan que dentro de un entorno digital se introducen nuevas formas de resolver tareas similares, que se suman a las habituales o las sustituyen.

Además, sostienen que introducir el concepto variable en los primeros grados se puede resolver algunas tareas en Scratch. No obstante, los investigadores indican que existe una distancia entre el concepto de la variable en matemáticas con respecto a los

comandos del Scratch. Por ende, estas diferencias les resultarían difícil al docente al momento de enseñar.

Uno de los resultados de esta investigación señala que los profesores tienen ciertas dificultades cuando quieren integrar las herramientas digitales para la enseñanza y el aprendizaje del álgebra. Esto se debe a la distancia instrumental que presentan estos entornos digitales.

Con respecto a las conclusiones, los autores manifiestan que se requiere que los docentes desarrollen nuevas organizaciones didácticas para enseñar, evaluar, etc., y para poder modificar los contenidos y coherencias matemáticas habituales.

Investigaciones relacionadas al pensamiento computacional y TAD

En la investigación de Chaachoua et al. (2022), se modelizó una organización matemática mixta en torno a la noción de división euclidiana en dos libros de texto de nivel primaria. Se empleó como marco teórico la Teoría Antropológica de lo Didáctico (TAD), específicamente, el marco T4TEL.

Los investigadores se apoyan en el trabajo de Couderette (2016), quien analizó las praxeologías algorítmicas en la intersección de las matemáticas y la informática. Esto se debe a que identificó tres tipos de praxeologías: matemática, algorítmica e informática, todas interconectadas entre sí. Asimismo, consideran la investigación de Strock y Artaud (2019), ya que examinaron las dificultades para destacar los elementos tecnológicos y teóricos propios de los algoritmos. La idea era reconocer las causas de estas dificultades. Ellos identificaron siete tipos de tareas que pueden ser reconocidas y trabajadas, las cuales están relacionadas entre sí, que forman una red de tipos de tareas. Estos dos estudios contribuyeron a la creación de un modelo praxeológico de referencia, que facilita la descripción y análisis de los saberes relacionados con las matemáticas y la algorítmica.

Por otra parte, Chaachoua et al. (2022) desarrollaron una secuencia didáctica centrada en la división euclidiana con el objetivo de explorar un tipo de tarea matemática, $\boldsymbol{T_{DE}}$, . Esta consiste en determinar uno o dos términos desconocidos de la

expresión $a = bq + r$, donde a es conocido y $0 \leq r < b$. A través del entorno de programación visual Scratch, buscan calcular el cociente de la división euclidiana de a entre b y determinar el resto.

Los investigadores se centran en describir los tipos de tareas, $\boldsymbol{T_{prog}[T]}$. Para esto, se crea un programa en Scratch que realice un tipo de tarea $\boldsymbol{T}$, donde $\boldsymbol{T}$ es un tipo de tarea matemática. Además, se identifica cuatro tipos de tareas para $\boldsymbol{T_{DE}}$: (1) $\boldsymbol{T_{\text{Descomponer}}}$, que implica descomponer un número natural a como un producto de dos números naturales; (2) $\boldsymbol{T_{\text{Cociente}}}$, que consiste en determinar el cociente de la división euclidiana de dos números naturales *a* y *b*; (3) $\boldsymbol{T_{Resto}}$, que se centra en determinar el resto de la división euclidiana de dos números naturales a y b; y (4) $\boldsymbol{T_{\text{Reprersenta}}}$ que se define como la representación de la división euclidiana de dos números naturales a y b.

Además, ellos mencionan que en el bloque "saber hacer" de esta organización matemática, se basa principalmente en la noción de ingrediente de una técnica. Para ello, describieron los ingredientes de las técnicas asociadas a estos tipos de tareas mencionados en el anterior párrafo.

Los investigadores afirman que la combinación de las praxeologías relacionadas con el tipo de tareas $\boldsymbol{T_{DE}}$ y los elementos algorítmicos crearán un modelo praxeológico de referencia para el tipo de tarea $\boldsymbol{T_{prog}[T_{DE}]}$. Para esto, unen las técnicas del tipo de tarea $\boldsymbol{T_{prog}[T_{DE}]}$ con las del tipo de tarea $\boldsymbol{T_{DE}}$. En el nivel praxeológico de la programación, se identificaron los siguientes ingredientes técnicos: $T_{Repetir\ q\ veces}$, que consiste en ingresar el número q en el bloque de repetición; $T_{asociar}$, que asocia dos bloques de instrucción entre sí; $T_{duplicar}$, que se refiere a duplicar un bloque de instrucción; y $T_{ejecutar}$, que consiste en ejecutar el programa creado en Scratch como tipos de tareas elementales.

Entre las conclusiones de este trabajo, señalan que la modelización de los saberes de una organización matemática mixta se concibe a través de un modelo praxeológico de referencia sobre un objeto matemático, así como en la estructuración de

las praxeologías algorítmicas. Además, este el modelo praxeológico de referencia puede generalizarse a los tipos de tareas $\boldsymbol{T_{prog}[T]}$, donde T es un tipo de tarea matemática.

La investigación de Chaachoua et al. (2022) es relevante para nuestro estudio, ya que nos ofrece una guía para la construcción del modelo praxeológico mixto sobre secuencias figurales fundamentado en el marco T4TEL.

Asimismo, en el estudio de Kilhamn et al. (2022), se investiga acerca de la incorporación de la programación en la enseñanza de las matemáticas en las escuelas de Suecia. Se emplea el marco teórico de la Teoría Antropológica de lo Didáctico de Yves Chevallard (2006), sobre la transposición de saberes de la programación a las matemáticas, desde el "saber a enseñar" hasta el "saber enseñado".

Ellos analizan tres tareas de programación en clases de matemáticas con el fin de reconocer la praxis (qué y cómo) y el logos (razones y justificaciones para la elección de qué y cómo, así como los objetivos de aprendizaje) en la ejecución de la tarea, que se centra en el uso de variables.

La metodología, para analizar el uso de las variables presentes en las tareas de programación, se basa en la investigación de Radford (2014). Esto se debe a que él identifica tres condiciones fundamentales del pensamiento algebraico: la indeterminación, la denotación y la analiticidad. Además, ellos se apoyan en el trabajo de Mason et al. (2005), quienes definen el pensamiento algebraico como la capacidad de articular y reconocer la generalidad, es decir, la habilidad para pasar de lo particular a lo general y viceversa.

Durante la ejecución de las tres tareas, utilizaron el programa Scratch. La primera tarea consistió en dibujar un círculo, un semicírculo y segmentos más pequeños de un círculo como un cuarto o un tercio. También, se solicitó dibujar un círculo en el que una mitad fuera de color rojo y la otra, de color azul. El objetivo consistió experimentar los grados del círculo sin hacer distinción entre el disco y el contorno.

Los investigadores indican que el código original de Scratch utilizado en esta tarea no incluye bloques para definir variables y no las presenta de manera clara o explícita como se haría en matemáticas. Aunque la posición del *sprite* se determina por sus coordenadas x e y, estas no se utilizan directamente en el código, salvo al iniciar en 0.

En su lugar, se emplea comandos de movimiento que modifican las coordenadas de forma indirecta. Esto sugiere la existencia de varias variables implícitas en el programa, que no son evidentes por la forma en que se ha escrito el código.

Algunas de estas variables implícitas, como la posición del sprite (x, y), indican el centro del círculo, así como la rotación del sprite, el tamaño, el color y la ubicación del lápiz. Con respecto al número de rayos y el número de pasos que se mueven, se pueden hacer visibles al crear bloques en Scratch y nombrarlos. Esto facilita la exploración de cómo su variación afecta el resultado.

Los investigadores destacan que, aunque Scratch permite a los usuarios identificar variables, se pierde una valiosa oportunidad de aprendizaje cuando estas están ocultas dentro de otros tipos de bloques. Esto dificulta que los estudiantes comprendan y aprendan sobre las variables de manera efectiva, lo que limita así su desarrollo de habilidades en programación.

En la segunda tarea, ellos utilizan variables en un código de Scratch para realizar multiplicaciones mediante sumas repetidas. Introducen dos variables, count y sum, que se inicia en 0 y las otras dos, var1 y var2, que representan el número a repetir y la cantidad de repeticiones respectivamente. A lo largo del programa, se crea un bucle donde la variable count se incrementa hasta que alcanza el valor de var2, que muestra en cada iteración la multiplicación correspondiente. Además, se menciona que var1 y var2 mantienen sus valores fijos durante todo el programa, mientras que count y sum cambian. La variable count se usa para llevar el registro de las sumas y sum acumula el resultado final de esas sumas, lo que se convierte en la salida del programa.

Los investigadores destacan que la tarea implica varias variables, lo que genera dificultades al pasar de variables sin nombre claro a aquellas que están definidas, así como el uso de una variable contador que registra repeticiones. También, se abordan problemas de sintaxis en Scratch como el uso de bloques de diferentes colores, la colocación dentro y fuera del bucle, y la diferenciación entre entradas y salidas. Se menciona que el bloque de respuesta es una variable temporal que puede causar confusión sobre el manejo de variables con el mismo nombre. Finalmente, sugieren que practicar el conteo puede ayudar a los estudiantes a entender mejor los conceptos de contador y de variables.

Los investigadores modifican la tarea al incorporar un enfoque algebraico, pues proponen una alternativa en el uso de un contador para repetir acciones complejas. Se sugiere utilizar números más grandes como 147 y presentar una suma de números consecutivos, lo que requiere un mayor esfuerzo y despierta el interés de los estudiantes. La tarea se diseñó para estar ligeramente fuera del alcance inmediato de los alumnos, pero dentro de acciones familiares, lo que permite al profesor guiarlos mediante ejemplos más simples.

Además, señalan que es fundamental que los estudiantes comprendan el papel de cada instrucción y discutan la nomenclatura utilizada. Para esto, ellos deben diferenciar entre letras individuales en álgebra y palabras descriptivas en programación. Por último, se destaca que las variables de contador son herramientas valiosas para explorar patrones.

En la tercera tarea, se introduce el concepto de funciones a través de la programación, destacando los parámetros como un tipo específico de variable. Esta actividad permite a los estudiantes identificar las diferencias entre variables y parámetros, ya que, mediante el código en Scratch, pueden observar cómo desempeñan diferentes funciones y cómo se desarrollan a medida que se ejecuta el programa.

Uno de los resultados de esta investigación indica que, al centrarse en los conceptos algebraicos, el entorno de programación puede actuar como un nuevo medio para reflejar y explorar diversos aspectos de las variables en el contexto del álgebra temprana, específicamente con estudiantes de entre 10 y 12 años de edad.

En cuanto a las conclusiones, los autores sugieren la necesidad de un tipo de logos orientado a las matemáticas, particularmente en el ámbito del Álgebra. El fin es mejorar el aprendizaje de esta disciplina durante las actividades de programación. También, resaltan la importancia de desarrollar una praxeología que vincule las tareas y técnicas con un logos que justifique el uso de la programación.

Investigaciones relacionadas a la secuencia figural

Zapatera (2022), explora cómo la generalización de patrones puede ser utilizada como una herramienta efectiva para introducir el pensamiento algebraico en educación

primaria. El autor argumenta que la identificación y generalización de patrones es un punto de partida valioso para el desarrollo de habilidades algebraicas en los estudiantes jóvenes, lo que facilita su transición de la aritmética a conceptos más abstractos del álgebra.

El estudio sugiere que trabajar con patrones permite a los estudiantes experimentar con ideas fundamentales del álgebra como la generalización y la estructuración de relaciones. Al identificar y extender patrones, los estudiantes comienzan a comprender conceptos algebraicos básicos como las secuencias y las variables de manera concreta y accesible.

El investigador describe diversas metodologías y estrategias para enseñar patrones en el aula. Estas incluyen actividades prácticas y visuales que ayudan a los estudiantes a observar, describir, y generalizar patrones numéricos y geométricos. El enfoque está en desarrollar la capacidad de los estudiantes para reconocer regularidades y formular generalizaciones a partir de ellas.

Los resultados indican que la práctica con patrones contribuye significativamente al desarrollo del pensamiento algebraico. La capacidad de generalizar patrones ayuda a los estudiantes a construir una base sólida para entender conceptos algebraicos más complejos en el futuro. Además, la generalización fomenta habilidades de razonamiento lógico y resolución de problemas.

El autor concluye que la generalización de patrones es una herramienta poderosa para la enseñanza del álgebra en la educación primaria. Al centrarse en patrones, los estudiantes no solo aprenden a reconocer y extender secuencias, sino que también desarrollan habilidades algebraicas fundamentales que servirán como base para estudios matemáticos más avanzados.

Asimismo, en la investigación realizada por Uicab et al. (2022), se analiza cómo los estudiantes de 10 a 12 años generalizan patrones cuando resuelven problemas de secuencias figurales y numéricas. Además, se interesan sobre qué tipos de expresiones emplean al resolver diversas tareas matemáticas relacionadas con la generalización. Para ello, adoptan la perspectiva teórica de Mason (1996). Él describe la generalización como una espiral continua de acciones denominadas manipular, obtener sentido de y

articular, y aboga porque los estudiantes desarrollen su propia simbología en lugar de utilizar de inmediato una notación convencional.

Los investigadores realizaron un análisis de las respuestas de los estudiantes a tareas específicas relacionadas con secuencias, y observaron cómo los estudiantes identificaron patrones y formularon expresiones generales para resolver los problemas.

Asimismo, el estudio revela las formas en que los estudiantes expresan y formalizan la generalización de patrones. También, examina las dificultades que enfrentan y cómo estas dificultades pueden ser abordadas para mejorar la comprensión matemática.

Los hallazgos ofrecen recomendaciones para diseñar tareas y actividades que ayuden a los estudiantes a desarrollar habilidades de generalización más efectivas.

Se considera que este tipo de estudio es valioso, porque ayuda a entender cómo los estudiantes piensan y razonan sobre las matemáticas, y puede informar el diseño de intervenciones educativas más efectivas.

1.2 Justificación

Con base en las investigaciones de referencias presentadas, se evidencia la importancia de adquirir las habilidades del pensamiento computacional para la enseñanza matemática y la resolución de problemas, particularmente en el reconocimiento y generalización de patrones, que tanto los profesores como los estudiantes deben poseer.

En particular, en el contexto del sistema educativo peruano, se demanda explícita e implícitamente que el docente desarrolle y aplique el pensamiento computacional para analizar problemas. Esto lo vemos reflejado en el Diseño Curricular Básico Nacional (DCBN) de la Formación Inicial Docente (2020). Este es un documento de política educativa que presenta tanto el perfil de egreso y las competencias de un estudiante de Formación Inicial Docente (FID) como los niveles de desarrollo de dichas competencias. También, presenta el modelo curricular, las descripciones de los cursos y módulos e incluso orientaciones pedagógicas generales para el desarrollo integral del profesional docente.

El DCBN de la FID afirma que, en la actualidad los estudiantes de la Educación Básica Regular necesitan de habilidades asociadas al manejo de entornos visuales. Por ello, una de las demandas de la educación secundaria que debe encargarse el Diseño Curricular Básico Nacional en la especialidad de matemática es potenciar el pensamiento computacional.

Con respecto al perfil de egreso de un estudiante de la FID, se evidencia competencias guiados al reforzamiento del desarrollo personal, al empleo de habilidades investigativas y al manejo de entornos digitales. Está organizado en cuatro dominios:

Dominio1: Preparación para el aprendizaje de los estudiantes

Dominio 2: Enseñanza en el aprendizaje de los estudiantes

Dominio 3: Participación en la gestión de la escuela articulada a la comunidad

Dominio 4: Desarrollo personal y de la profesionalidad e identidad del docente

Nuestra investigación se enfoca en el dominio 4, particularmente en la competencia 11. Esto se debe a que el estudiante de la FID desarrolla su formación profesional con ayuda de las tecnologías digitales. Asimismo, crea y/o motiva oportunidades de aprendizaje en los alumnos. Además, aprovecha estos entornos digitales para su práctica pedagógica y de esta manera ayuda a los estudiantes a desarrollar su pensamiento computacional. En la tabla 1, se presenta las capacidades de la competencia 11 del dominio 4.

Tabla 1

Capacidades de la competencia 11

DOMINIO 4: DESARROLLO PERSONAL Y DE LA PROFESIONALIDAD E IDENTIDAD DEL DOCENTE	
COMPETENCIA 11	**CAPACIDADES**
Gestiona los entornos digitales y los aprovecha para su desarrollo profesional y práctica pedagógica, respondiendo a las necesidades e intereses de aprendizaje de los estudiantes y los contextos socioculturales, permitiendo el desarrollo de la ciudadanía, creatividad y emprendimiento digital en la comunidad educativa.	Ejerce su ciudadanía digital con responsabilidad. Gestiona información en entornos digitales con sentido crítico, responsable y ético. Gestiona herramientas y recursos educativos en los **entornos digitales** para mediar el aprendizaje y desarrollar habilidades digitales en sus estudiantes. Se comunica y establece redes de colaboración a través de entornos digitales con sus pares y los miembros de su comunidad educativa. Resuelve diversos problemas de su entorno mediante el **pensamiento computacional**.

Fuente: Adaptado de DCBN (2020)

Se observa que una de las capacidades de la competencia 11 tiene la necesidad de fomentar el pensamiento computacional a través de los entornos digitales para propiciar el aprendizaje en los estudiantes. Al respecto, De la Hoz & Neira (2022) manifiestan que es necesario que los docentes reciban capacitación en el uso de Scratch para poder integrarlo eficazmente en su enseñanza.

Además, en los estándares de la FID, se evalúa la progresión de las competencias del estudiante de la FID con la finalidad de conocer qué conocimientos y habilidades deben saber y desarrollar para su formación docente. Los estándares presentan tres niveles. El primero y segundo nivel son expectativas que se espera que el profesional en formación pueda lograr al finalizar V ciclo y en X ciclo respectivamente. El tercer nivel es el sobresaliente, pues supera las expectativas antes de concluir su formación profesional. Este último nivel está en relación con el décimo ciclo y con la formación docente en servicio en una lógica de mejora continua a lo largo de la trayectoria profesional. En la

tabla 2, se puede apreciar algunos estándares por niveles de la competencia 11 que se muestran en el DCBN de la FID.

Tabla 2

Estándares de la competencia 11

COMPETENCIA 11	
Nivel 1 de desarrollo de la competencia Expectativa hacia el V ciclo	Identifica las oportunidades que ofrecen las tecnologías digitales en términos de acceso a la información y su valor como herramientas para mediar el aprendizaje. Explica y justifica cómo facilitan su propio proceso de aprendizaje y reconoce la importancia de utilizarlas con responsabilidad, ética y sentido crítico. Valora el papel de las tecnologías para la comunicación y la generación de espacios de colaboración entre los miembros de su comunidad educativa y para el desarrollo del **pensamiento computacional.**
Nivel 2 de desarrollo de la competencia Expectativa hacia el X ciclo	Aprovecha las tecnologías digitales de manera responsable y ética, tanto en su vida privada como profesional. Incorpora medidas de seguridad en la red y cuida de su bienestar físico y psicológico en el mundo digital. Asimismo, discrimina e incorpora en el proceso de enseñanza y aprendizaje información proveniente de internet y de diferentes formatos (textos, videos, sonidos, animaciones, etc.). Explica y justifica las posibilidades que ofrecen las tecnologías digitales para el quehacer docente y la importancia de utilizarlas con sentido crítico. Además, las utiliza eficientemente para comunicarse con sus pares y otros miembros de la comunidad educativa. Accede a plataformas donde los docentes intercambian contenidos y opiniones. Resuelve problemas digitales, transfiere su competencia digital a nuevas situaciones y valora el papel de las tecnologías en el desarrollo del **pensamiento computacional.**
Destacado - Articulación con la Formación Docente en Servicio	Aprovecha las tecnologías digitales de manera responsable y ética tanto en su vida privada como profesional. Incorpora medidas de seguridad en la red y cuida su bienestar físico y psicológico en el mundo digital. Asimismo, discrimina, organiza convenientemente e incorpora en el proceso de enseñanza y aprendizaje información proveniente de internet y de diferentes formatos (textos, videos, sonidos, animaciones, etc.), combinando pertinentemente las tecnologías digitales de las que dispone. Además, las utiliza eficientemente para comunicarse, colaborar e intercambiar información con sus pares y otros miembros de la comunidad educativa. Resuelve problemas digitales, transfiere su competencia digital a nuevas situaciones y sabe cómo aplicar el **pensamiento computacional** para analizar problemas.

Fuente: Adaptado de DCBN (2020)

Entre las expectativas de la competencia 11, se encuentra la necesidad de usar entornos digitales para el desarrollo del pensamiento computacional, y de esta manera analizar y resolver problemas de su entorno. Al respecto, Miller (2019) manifiesta que los sistemas educativos deben proporcionar la posibilidad de desarrollar y ayudar a los profesores en la ejecución de nuevas tecnologías y habilidades como el pensamiento computacional.

Así mismo, el plan de estudio del Diseño Curricular Básico Nacional (DCBN) de la FID en la especialidad de matemática para la Educación Básica Regular (EBR) consta de diez (10) ciclos académicos. También, presenta cursos y módulos que están articulados al perfil de egreso y organizados por tres componentes curriculares: formación general, formación en la práctica e investigación y formación específica. A continuación, se presenta algunos cursos que tienen relación con pensamiento Computacional y sucesiones.

En la componente curricular de la formación general en el ciclo II, se encuentra el curso Resolución de Problemas Matemáticos II correspondiente al primer año de la formación inicial docentes como se muestra en la tabla 3.

Tabla 3

Características del curso Resolución de Problemas II de la Formación General

Componente Curricular	**Formación General**		
Curso	**RESOLUCIÓN DE PROBLEMAS MATEMÁTICOS II**		
Ciclo	**II**	**Competencia**	**4,8,11**

El curso está diseñado para que el estudiante de FID tenga oportunidades de visualizar, modelar y transformar las formas bidimensionales y tridimensionales, medir y estimar objetos, y describir su ubicación mediante sistemas de referencia, así como de interpretar y generalizar patrones. El curso propicio que el estudiante de FID reflexione sobre las ideas centrales abordadas, **reconozca los alcances de las técnicas desarrolladas** y establezca relaciones cada vez más generales entre las nociones matemáticas estudiadas. Para llevar a cabo todo lo anterior, puede hacer **uso de su pensamiento computacional** y diversos recursos informáticos.

Algunos de los desempeños específicos que se esperan alcanzar al final del curso son los siguientes:

- Justifica su proceso de resolución de situaciones problemáticas del entorno asociadas a las formas bidimensionales y tridimensionales; al movimiento y localización de objetos; y a relaciones de regularidad, equivalencia y cambio.
- Identifica cuáles son sus fortalezas y qué aspectos debe mejorar al usar sus conocimientos matemáticos para resolver, evaluar y tomar decisiones sobre situaciones problemáticas del entorno.
- **Utiliza recursos informáticos para interpretar y generalizar patrones**, establecer relaciones de equivalencia y analizar situaciones de cambio, y justifica cómo estas tecnologías facilitan su aprendizaje.

Fuente: Adaptado de DCBN (2020)

Entonces, según las descripciones del curso en la tabla 3, se espera que el estudiante de la FID utilice el pensamiento computacional con el propósito de generalizar patrones e identificar los alcances de las técnicas realizadas a través de los recursos informáticos. De esta manera, favorece su enseñanza y aprendizaje. En este aspecto, Miller (2019) manifiesta que la codificación tiene un potencial en la capacidad de los estudiantes para ver patrones, reconocer la unidad de repetición, deducir el patrón y abstraer la estructura general; estos son componentes claves del pensamiento computacional.

De igual manera, en la componente curricular de la formación específica, se encuentra el curso Matemática y Pensamiento Computacional correspondiente al tercer año de formación inicial docentes como se observa en la tabla 4.

Tabla 4

Características del curso Matemática y pensamiento computacional de la Formación Especifica

Componente Curricular	**Formación Específica**		
Curso	**MATEMÁTICA Y PENSAMIENTO COMPUTACIONAL**		
Ciclo	**VI**	**Competencia**	**4,8,11**

El curso tiene por propósito que los estudiantes de FID comprendan la importancia del desarrollo del pensamiento computacional para el desarrollo de las competencias matemáticas desde una perspectiva transversal e integradora con otros aprendizajes. Reconoce el valor de la ciencia de la computación en la formación de ciudadanos preparados para afrontar un mundo cada vez más digital e identifica el pensamiento computacional como un aprendizaje importante para resolver problemas complejos del entorno con y sin recursos informáticos. El estudiante de FID diseña e implementa experiencias de aprendizaje que conectan el pensamiento computacional con la matemática para la resolución de problemas que favorezcan la organización lógica y análisis de datos, la modelización, la algoritmización, la generalización, entre otros. Además, hace uso de la programación de computadoras con lenguajes de **programación iconográficos.**

Algunos de los desempeños específicos que se esperan alcanzar al final del curso son los siguientes:

- Ejecuta acciones pedagógicas para desarrollar aprendizajes que conecten la matemática con el pensamiento computacional lo que brinda oportunidades para que los estudiantes de EB elaboren sus propias ideas y exploren soluciones.
- Evalúa cómo su práctica promueve una visión interdisciplinaria y compleja del conocimiento a partir del desarrollo del pensamiento computacional ligado a las actividades de aprendizaje de matemática, y del diálogo con sus pares y docentes formadores.
- Utiliza la programación de computadoras para la resolución de problemas matemáticos; ello implica el pensamiento computacional.

Fuente: Adaptado de DCBN (2020)

Entonces, según la tabla 4, se requiere desarrollar el pensamiento computacional en los futuros profesionales docentes para que en su práctica pedagógica usen programas iconográficos con el fin de resolver problemas matemáticos

Por otro lado, en la componente curricular de la formación específica, se encuentra los cursos Variación y sus Fundamentos para el Aprendizaje y la Enseñanza I y Álgebra como Herramienta Modelizadora I . Esto corresponde al segundo año de

formación inicial docentes que tienen relación con el objeto matemático sucesiones como se observa en la tabla 5 y la tabla 6 respectivamente.

Tabla 5

Características del curso de Variación y sus Fundamentos para el Aprendizaje y la Enseñanza I

Componente Curricular	**Formación Específica**		
Curso	**VARIACIÓN Y SUS FUNDAMENTOS PARA EL APRENDIZAJE Y LA ENSEÑANZA I**		
Ciclo	**III**	**Competencia**	**1, 2, 11**

El curso tiene como propósito que los alumnos de FID comprendan la variación y el cambio entre magnitudes y cómo desarrollan los estudiantes de EB su pensamiento variacional. Se examinan los procesos matemáticos que se llevan a cabo cuando los estudiantes de EB interpretan y representan relaciones de dependencia presentes en diferentes contextos, especialmente las que aparecen en las **sucesiones**; analizan la naturaleza del cambio; modelan situaciones o fenómenos del mundo real mediante funciones lineales, afines, cuadráticas, exponenciales, logarítmicas y trigonométricas; y formulan y argumentan afirmaciones a partir de las relaciones encontradas. En ese proceso de aprendizaje, los estudiantes de FID profundizan en los conocimientos disciplinares asociados a las funciones señaladas y establecen conexiones entre dichos conceptos. Ellos estudian las principales dificultades que pueden presentarse durante el desarrollo del pensamiento variacional a partir de la lectura de investigaciones, y cómo se debe gestionar el error para favorecer el aprendizaje de los estudiantes, especialmente cuando realizan transformaciones entre las distintas representaciones de una función. Se identifican fenómenos para ser estudiados mediante actividades o problemas que requieren la caracterización de la variación en diferentes contextos y se discute cómo planificar actividades y crear problemas que demanden la evolución del pensamiento variacional.

Algunos de los desempeños específicos que se esperan alcanzar al final del curso son los siguientes:

- Resuelve situaciones problemáticas de la vida diaria asociadas a **patrones, sucesiones**, relaciones y funciones; identifica los conocimientos y procesos matemáticos involucrados en la solución; y analiza si es posible adaptar la situación a la EB.
- Diseña experiencias de aprendizaje relacionadas con funciones lineales, afines, cuadráticas, entre otras; y propone una secuencia de actividades en las que los estudiantes de educación básica exploran soluciones o confrontan sus puntos de vista.
- Utiliza tecnologías digitales para analizar la naturaleza del cambio y representar funciones lineales, afines, cuadráticas, exponenciales, logarítmicas y trigonométricas; y justifica cómo dicha elección favorece el aprendizaje.

Fuente: Adaptado de DCBN (2020)

Tabla 6

Características del curso Álgebra como Herramienta Modelizadora I

Componente Curricular	**Formación Específica**		
Curso	**ÁLGEBRA COMO HERRAMIENTA MODELIZADORA I**		
Ciclo	**IV**	**Competencia**	**1, 4, 5**

El curso tiene por propósito que los estudiantes de FID desarrollen en sus estudiantes de EB capacidades que permitan encontrar y analizar regularidades para que en forma progresiva identifiquen modelos y los formalicen. Para ello, propone actividades de aprendizaje, crea problemas y analiza evidencias de aprendizaje que estos recogen. Además, estudia los procesos matemáticos que se manifiestan cuando los estudiantes de EB interpretan y representan la regularidad existente en las **sucesiones**, establecen equivalencias entre expresiones algebraicas y las operan, así como cuando transforman las condiciones de una situación problemática en ecuaciones e inecuaciones lineales y cuadráticas o en sistemas de dos ecuaciones lineales con dos incógnitas y argumentan sus procedimientos. Profundiza en los contenidos disciplinares asociados a los conocimientos abordados y establece conexiones entre ellos e identifica los fenómenos que pueden ser estudiados a través de problemas que requieren la construcción de nociones algebraicas. Investiga sobre la evolución del razonamiento algebraico y sobre las principales dificultades que pueden presentar los estudiantes de EB durante el aprendizaje de igualdades y desigualdades a partir del estudio de investigaciones, que implica un análisis de evidencias de aprendizaje del estudiante de EB. Finalmente, reflexiona sobre la pertinencia de introducir el álgebra mediante actividades de modelización matemática para el desarrollo de las competencias.

Algunos de los desempeños específicos que se esperan alcanzar al final del curso son los siguientes:

- Resuelve situaciones problemáticas de la vida diaria asociadas al uso de diversas técnicas algebraicas con ecuaciones e inecuaciones, identifica los conocimientos y procesos matemáticos involucrados en la solución, y analiza si es posible adaptar la situación a la EB.
- Desarrolla actividades de aprendizaje para la construcción de modelos que implica el uso de expresiones algebraicas y brinda oportunidades para que los estudiantes elaboren sus propias ideas y exploren soluciones.
- Describe evidencias de aprendizaje de los estudiantes de EB sobre **sucesiones**, ecuaciones e inecuaciones lineales y cuadráticas o sistemas de dos ecuaciones lineales con dos incógnitas a partir de la información recogida al aplicar instrumentos de evaluación en espacios de práctica reales.

Fuente: Adaptado de DCBN (2020)

Por otro lado, el DCBN brinda las orientaciones pedagógicas generales con el fin de que el estudiante de FID tenga un desarrollo universal de las competencias establecidas en el perfil de egreso. Particularmente, en la orientación para el desarrollo y tratamiento de la competencia digital, se establece que las Escuelas de Educación Superior (EES) deben desarrollar en sus estudiantes el pensamiento computacional para que sean capaces de resolver problemas de su entorno.

1.3 Pregunta y objetivo de la investigación

Por todo lo descrito anteriormente, se considera fundamental estudiar las praxeologías mixtas que se presentan en las tareas matemáticas con Scratch asociada a la secuencia figural. Con base en esto, se formula la siguiente pregunta de investigación:

¿Cuáles son las praxeologías mixtas que se presentan en las tareas matemáticas con Scratch asociadas a la secuencia figural en Educación Primaria?

Objetivo general

Describir y analizar las praxeologías mixta que se presentan en las tareas matemáticas con Scratch asociadas a la secuencia figural en Educación Primaria

Objetivos específicos

- Determinar una organización matemática de referencia para la secuencia figural
- Identificar las organizaciones matemáticas mixtas asociadas a la secuencia figural en tareas matemáticas realizadas en Scratch
- Construir un modelo praxeológico de referencia relacionando a las tareas matemáticas realizadas en Scratch asociado a la secuencia figural

En el siguiente capítulo, se presentan los elementos del marco teórico y la metodología, que fundamentan y guían la investigación realizada.

CAPÍTULO II

MARCO CONCEPTUAL: ASPECTOS TEÓRICOS Y METODOLÓGICOS

En este capítulo, se presenta el marco teórico que utilizaremos en nuestra investigación: la Teoría Antropológica de lo Didáctico (TAD). Esta teoría nos permitirá describir una organización matemática denominada también como praxeología, que nos ayudará a identificar los tipos de tareas, técnicas, tecnologías y teorías involucradas. Además, se detallarán los procedimientos metodológicos que facilitarán el abordaje de los objetivos de la investigación.

2.1 Teoría Antropológica de lo Didáctico (TAD)

La Teoría Antropológica de lo Didáctico desarrolla un modelo para describir cualquier actividad humana a través del concepto de praxeología (Chevallard, 1999). Además, se indica que, en una praxeología u organización matemática, el saber se organiza en dos bloques. El primero es el bloque de la praxis o del saber- hacer $[\boldsymbol{T}/\boldsymbol{\tau}]$, constituido por los tipos de tareas ($\boldsymbol{T}$) y las técnicas ($\boldsymbol{\tau}$) que se usan para resolver las tareas. El segundo es el bloque del logo o del saber $[\boldsymbol{\theta}/\boldsymbol{\Theta}]$, constituido por las tecnologías ($\boldsymbol{\theta}$) , que son los discursos que describen, explican y argumentan las técnicas que se emplean; y la teoría ($\boldsymbol{\Theta}$) el cual presenta un rol parecido que la tecnología hace para las técnicas, pero de manera general y abstracta.

2.1.1 Noción de praxeología u organización matemática

Según Chevallard (1999), uno de los conceptos fundamentales de la TAD es la noción de praxeología u Organización Matemática (OM) que permite describir y modelizar la actividad matemática, considerada como una actividad humana.

La unidad mínima de análisis de las organizaciones matemáticas (OM) está formado por los *tipos de tareas, técnicas, tecnologías y teorías*. Estos elementos de la OM se representan simbólicamente como $\boldsymbol{P} = [\boldsymbol{T}/\boldsymbol{\tau}/\,\boldsymbol{\theta}/\boldsymbol{\Theta}]$, que están fuertemente interrelacionados entre sí.

Elementos de una praxeología u organización matemática

Para Chevallard (1999), una praxeología está integrada por cuatro elementos: Un tipo de tarea $\boldsymbol{T}$; una técnica $\boldsymbol{\tau}$, que consiste en la manera de resolver el tipo de tarea $\boldsymbol{T}$; la justificación por una tecnología $\boldsymbol{\theta}$; y , por último, una teoría $\boldsymbol{\Theta}$ que legitima y justifica la tecnología. A continuación, se desarrollará cada uno de los elementos.

Tipos de tareas ($\boldsymbol{T}$)

Dentro del bloque saber- hacer se encuentra las nociones de tareas ($\boldsymbol{t}$), el tipo de tareas ($\boldsymbol{T}$) y el género de tareas. Una tarea ($\boldsymbol{t}$) se define como una acción específica sobre un objeto particular, que generalmente se expresa mediante un verbo conocido como el género de la tarea. Además, un tipo de tarea se refiere a una acción puntual que puede estar compuesta por una o más tareas como problemas o ejercicios.

Según Chevallard (1999), una tarea t es un elemento del tipo de tarea $\boldsymbol{T}$ y se denota como $t \in \boldsymbol{T}$. También, señala que las tareas, tipos de tareas, géneros de tareas no son datos de la naturaleza, sino que son “artefactos” u “obras” derivadas de construcciones institucionales específicas. La manera en que estas construcciones se reconstruyen en una institución es un objeto de estudio de la didáctica.

Técnicas ($\boldsymbol{\tau}$)

Una técnica es la manera de resolver una tarea ($\boldsymbol{t}$) específica dentro de una institución. El bloque[$\boldsymbol{T/\tau}$]**,** denominado práctico-técnico (saber hacer) está compuesto por un conjunto de tipos de tarea ($\boldsymbol{T}$) y un conjunto de técnicas ($\boldsymbol{\tau}$). Según Chevallard (1999), las técnicas ($\boldsymbol{\tau}$) no necesariamente tienen un carácter algorítmico y solo serán efectivas para una parte de las tareas, lo que se conoce como *alcance de la técnica*. Además, Chevallard señala que una técnica ($\boldsymbol{\tau}$) puede ser superior a otra si es capaz de resolver de manera más efectiva las tareas del tipo de tareas en cuestión.

Tecnología ($\boldsymbol{\theta}$)

Chevallard (1999) sostiene que la tecnología ($\boldsymbol{\theta}$) es un discurso racional sobre la técnica y se define mediante tres funciones. La primera función es justificar la técnica($\boldsymbol{\tau}$) de manera racional para garantizar que resuelva un tipo de tareas $\boldsymbol{T}$. La segunda función es explicar la técnica ($\boldsymbol{\tau}$) para que sea comprensible. Finalmente, la tercera función de la tecnología (θ) es desarrollar nuevas técnicas (τ) que sean más eficientes y adecuadas para la resolución de tareas específicas.

Teoría ($\boldsymbol{\theta}$)

Otro elemento de la praxeología es la teoría. Esta cumple un rol similar al de la tecnología ($\boldsymbol{\theta}$) en relación con las técnicas, pero de manera más general y abstracta. Por lo tanto, la teoría ($\boldsymbol{\theta}$) proporciona un nivel superior de justificación, explicación y desarrollo de la tecnología ($\boldsymbol{\theta}$) **.**

2.1.2 El Marco T4TEL

El marco T4TEL es una extensión y formalización del modelo praxeológico de referencia de la teoría antropológica de la didáctica, desarrollado por Chevallard (1999). En este marco, T4 se refiere al cuádruple clásico de la praxeología: tareas, técnicas, tecnología y teoría. Por otro lado, TEL alude al Aprendizaje Mejorado por la Tecnología. Este marco fue elaborado por el equipo de Modelos y Tecnologías para el Aprendizaje Humano (MeTAH).

Chaachoua (2020) define las bases del marco T4TEL de la siguiente manera:

Tipo de tarea y subtipo de tarea.- Sea $\boldsymbol{T}$ el conjunto de tareas. Un elemento del conjunto $\boldsymbol{T}$, se llamará tarea y será denotado por $\boldsymbol{t}$. Una tarea $\boldsymbol{t}$ es descrita mediante un verbo de acción como determinar, hallar, etc. Asimismo, presenta complementos fijos como la regla general de una secuencia, la unidad de repetición de una secuencia figural, etc. Para todo conjunto $\boldsymbol{T}$ existe al menos una técnica $\boldsymbol{\tau}$, que resuelva una tarea $\boldsymbol{t}$ tal que el alcance de la técnica $\boldsymbol{\tau}$ escrito por

$\boldsymbol{P}(\boldsymbol{\tau})$ es un subconjunto de $\boldsymbol{T}$ o $\boldsymbol{T}$ es un subconjunto de $\boldsymbol{P}(\boldsymbol{\tau})$. Entonces, el conjunto de tareas $\boldsymbol{T}$ es un tipo de tarea.

Además, si $\boldsymbol{T}$ y $\boldsymbol{T'}$ son dos tipos de tareas tal que $\boldsymbol{T'} \subset \boldsymbol{T}$, entonces $\boldsymbol{T'}$ es un subtipo de tarea del tipo de tarea $\boldsymbol{T}$.

Generador de tipos de tareas y sistemas de variables. - Un sistema de variables es una lista de valores específicos que se pueden tomar dentro de un dominio matemático. Los distintos valores de las variables permiten generar tipos de tareas de manera específica. Un generador de tipos de tareas ($\boldsymbol{GT}$) está definido por un tipo de tarea ($\boldsymbol{T}$) y un sistema de variables.

Se debe tener en cuenta que un generador de tipos de tareas ($\boldsymbol{GT}$) no es un tipo de tarea ($\boldsymbol{T}$)**;** sin embargo, ayuda a crear tipos de tareas según su nivel de clasificación. Además, la creación de generadores de tipos de tareas precisa de las preguntas de investigación y de la institución de destino.

Asimismo, Chaachoua y Bessot (2019) señalan que las variables tienen funciones: (1) generar subtipos de tareas que cambian los valores de una o más variables, (2) caracterizar los alcances de las técnicas y sus dominios de pertenencia y (3) describir las praxeologías implementadas por los estudiantes.

Descripción de la técnica.- Una técnica $\boldsymbol{\tau}$ se describe mediante un conjunto de tipos de tareas $\{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \ldots \ldots \boldsymbol{T_i}\}$ tal que $\boldsymbol{T_i}$ puede ser de dos tipos. Por un lado, están las tareas intrínsecas. Están constituidas por los tipos de tareas que solo existen mediante la aplicación de las técnicas de otros tipos de tareas determinados. Por otro lado, están las tareas extrínsecas. Están formadas por los tipos de tareas que pueden prescribir institucionalmente a los estudiantes, es decir, los tipos de tareas que se pueden resolver de una cierta manera en una institución.

Ingrediente de una técnica

Cada tipo de tarea se define como un ingrediente de una técnica y estas por su parte tienen una o varias técnicas que se expresan mediante un conjunto de tipos de tarea (Chaachoua, 2020).

De aquí en adelante, a lo largo de la tesis, se utilizará la notación " $T' \longrightarrow T$ " y " $T^* \multimap T'$ " para indicar que T' es un subtipo de la tarea T, y que T^* es un ingrediente de al menos una técnica de T'.

Alcances de las técnicas.- Una técnica -una "manera de hacer las cosas"-sólo tiene éxito en una parte $\boldsymbol{P}(\boldsymbol{\tau})$ de las tareas del tipo $\boldsymbol{T}$ a las que es relativa. Esta parte se denomina alcance de la técnica. Esta tiende a fallar en $\boldsymbol{T}\backslash\boldsymbol{P}(\boldsymbol{\tau})$, por lo que podemos decir que "no se sabe, en general, cómo realizar tareas de tipo $\boldsymbol{T}$ ". Se distinguen dos tipos de alcance. Por un lado, está el *alcance teórico de una técnica que* es el conjunto de tareas donde la técnica es aplicable sin tomar en cuenta las condiciones de su ejecución. Por otro lado, está el *alcance pragmático de una técnica* que es el conjunto de tareas donde la técnica es confiable en el sentido de que permite realizar la tarea con poco riesgo de falla y un costo razonable (poco riesgo de error). Es obvio que el ámbito pragmático está incluido en el ámbito teórico. La técnica tiende a tener éxito en este alcance y tiende a fallar fuera de él. Es evidente que el ámbito pragmático está incluido en el ámbito teórico.

Otro elemento teórico a tomar en cuenta en la Teoría Antropológica de lo Didáctico (TAD) es la institución.

Institución. - Es una organización social estable en donde se llevan a cabo ciertas actividades específicas dentro de ciertos límites. Además, establece un contexto normativo para las acciones que se llevan a cabo en su interior. Al mismo tiempo, proporciona los recursos necesarios como materiales, organizacionales y cognitivos con el fin de que dichas actividades sean posibles realizarlas (Castela, 2016).

2.1.3 Ostensivos en la TAD

De acuerdo con Bosh (1994, citado en Najarro, 2018), el término "ostensivo" se refiere a cualquier objeto que tenga una naturaleza sensible y material, lo que permite que el sujeto lo perciba y lo manipule como una realidad concreta. Ejemplos de elementos ostensivos incluyen los sonidos (morfemas

lingüísticos), los grafismos (grafemas en la escritura de lenguas naturales y formales), los gestos, entre otros.

Bosh (1994) también sostiene que los objetos ostensivos tienen dos valencias: la instrumental y la semiótica. La primera considera el objeto ostensivo como una herramienta para llevar a cabo ciertas tareas o trabajos, es decir, como un elemento que facilita la ejecución de actividades. Esta valencia se aplica tanto a los símbolos escritos en matemáticas como a las palabras pronunciadas o los gestos realizados. La segunda que es la valencia semiótica ve al objeto ostensivo como un signo que representa o se refiere a otros objetos.

2.1.4 Modelo Epistemológico de Referencia (MER)

Un Modelo Epistemológico de Referencia es un concepto fundamental, ya que describe el conjunto de conocimientos y prácticas matemáticas que se consideran válidos y relevantes en un determinado contexto educativo. Este modelo proporciona un marco de referencia para la enseñanza y el aprendizaje de las matemáticas, lo que influye en la selección de contenidos, enfoques pedagógicos y evaluaciones.

En el contexto de la TAD, el MER se utiliza para analizar cómo se transmite y se enseña el conocimiento en el aula, así como para entender las dificultades y obstáculos que enfrentan los estudiantes al intentar aprender ciertos conceptos.

Es importante destacar que el MER presenta ciertas características. En primer lugar, es un modelo provisional, es decir, una suposición que será sometida a prueba con datos experimentales y puede ser modificada de manera permanente. En segundo lugar, es un modelo relativo que es elaborado por el investigador en educación con propósitos específicos y con límites definidos (Rojas y Sierra, 2022).

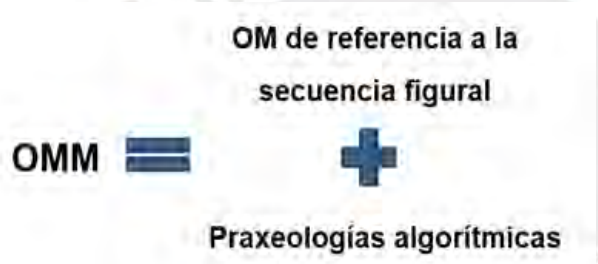
2.1.5 Organización matemática mixta

Según Chevallard (2001, citado en Chaachoua et al., 2022) afirma que una organización matemática mixta consiste en un conjunto de praxeologías que

integran las matemáticas con otras áreas del conocimiento, ya sean disciplinas escolares u otros saberes fuera del ámbito escolar.

Asimismo, Macias y Romo (2013) afirman que una praxeología mixta puede incluir tanto componentes matemáticos como no matemáticos. Por ejemplo, si una tarea que no es matemática se resuelve utilizando una técnica matemática, requerirá una validación matemática, así como validaciones no matemáticas relacionadas con la naturaleza de la tarea. Además, este tipo de praxeología exigirá validaciones no matemáticas como pruebas experimentales o validaciones vinculadas al contexto en el que se desarrolla la tarea.

Figura 1
Organización matemática mixta



Fuente. Adaptado de Crisci, 2020, p.88

2.1.6 Transposición didáctica

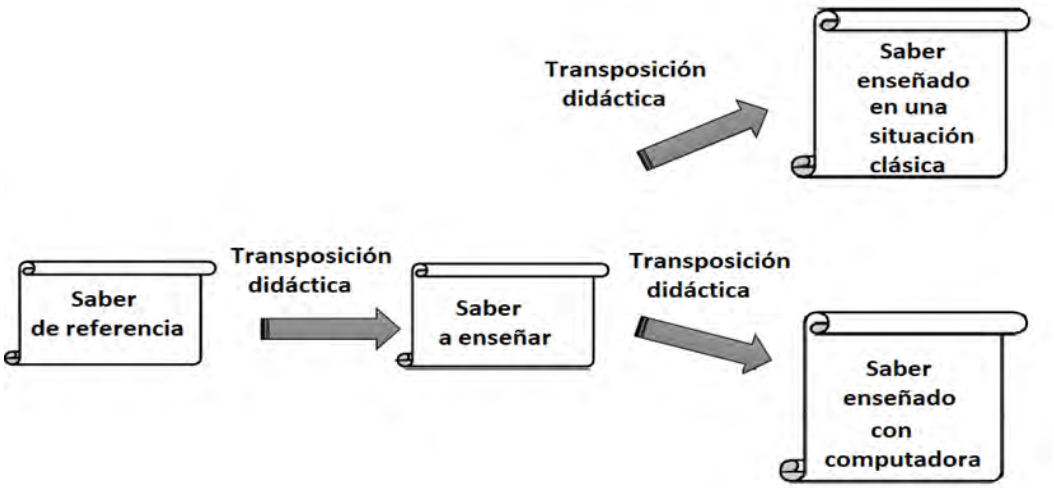
Según Bosch y Gascón (2014), las transformaciones, que sufre un contenido desde su producción y uso hasta su enseñanza y aprendizaje en una institución educativa, se conocen como el proceso de transposición didáctica. En otras palabras, se refiere al conjunto de mecanismos a través de los cuales se conceptualiza el saber que se enseña.

Figura 2
Diagrama del proceso de transposición didáctica

Fuente. Adaptado de Bosch y Gascón, 2014, p.70

Por otro lado, Balacheff (1994; como se cita en Briant & Bronner, 2015) aborda y reelabora el concepto de transposición didáctica de Chevallard. Para esto, introduce el término "transposición informática" para referirse a los obstáculos asociados al aprendizaje de saberes en un entorno informático.

Figura 3
Transposiciones didácticas e informáticas



***Fuente*.** Adaptado de Briant & Bronner, 2015, p.235

Este investigador identifica dos tipos de obstáculos relacionados con la transposición informática: los obstáculos de modelización computable, y los obstáculos vinculados al software y hardware de los medios informáticos. El primer tipo de obstáculo se refiere a la representación y el procesamiento interno del saber en la computadora; el segundo se relaciona con la representación y el procesamiento a nivel de la interfaz, es decir, lo que resulta “visible” para el usuario (Balacheff,1994, como se cita en Briant & Bronner, 2015).

Briant & Bronner (2015) adopta el concepto de transposición informática de Balacheff. En su investigación, aborda sobre la integración de algoritmos en la enseñanza de las matemáticas. El autor señala que se produce una doble transposición al plantear una tarea como el diseño de un programa para resolver un problema. Esta tarea está asociada a diferentes técnicas, justificadas por

tecnologías que pueden provenir del ámbito matemático, del ámbito informático o de ambos simultáneamente.

Para nuestra investigación, se utilizará las herramientas proporcionadas por la Teoría Antropológica de lo Didáctico (TAD) y el modelo praxeológico T4TEL, ambas fundamentales en el ámbito de la educación matemática. En particular, el modelo T4TEL resulta esencial para identificar y analizar praxeologías mixtas, es decir, prácticas educativas que combinan distintos enfoques y metodologías. Este aspecto es especialmente relevante en el contexto de la enseñanza de matemáticas con herramientas como Scratch, ya que facilita el estudio de cómo se integran las tecnologías y enfoques didácticos en la práctica educativa.

2.2 Procedimientos Metodológicos

Este trabajo de investigación se enmarca en el enfoque cualitativo, ya que se enfoca en el análisis constante de los datos recopilados. Según Bogdan y Bliken (1994; como se cita en Oliveira, 2019), la investigación cualitativa es esencialmente descriptiva y, en este contexto, las palabras y las imágenes son más adecuadas para la descripción que los números. En la presentación de los resultados, es común sintetizar los datos para 'ilustrar y fundamentar la presentación', respetando así la forma en que se obtuvieron. De este modo, los informes resultantes pueden ser elaborados de manera detallada. De esta forma, se reconoce que ninguna perspectiva puede ser reducida a lo trivial y que cada detalle tiene un significado propio.

Se tomará como marco teórico la Teoría Antropológica de lo Didáctico (TAD) y se empleará la formalización del modelo praxeológico T4TEL. Este enfoque nos proporciona herramientas para modelar los contenidos de una organización matemática mixta. El objetivo es estudiar e identificar las praxeologías mixtas presentes en las tareas matemáticas realizadas con Scratch en relación con la secuencia figural.

Para el desarrollo de la investigación, se seguirán los siguientes pasos:

- Realizar un estudio epistemológico de la secuencia figural y de la programación por bloques en Scratch

- Revisión de investigaciones sobre la secuencia figural
- Análisis de las actividades, relacionados al objeto matemático secuencia figural en investigaciones, artículos y libros de texto
- Construcción de un modelo praxeológico de la secuencia figural
- Identificar las praxeologías algorítmicas derivadas de la secuencia didáctica analizada
- Construir un modelo praxeológico de referencia mixta
- Llevar a cabo un análisis praxeológico de los materiales y recursos enfocados en la enseñanza de secuencia figural mediadas por Scratch

En el siguiente capítulo se presenta Scratch, el lenguaje de programación por bloques utilizado en esta investigación, explorando su relación con el pensamiento computacional y su aplicación en el aprendizaje de las matemáticas.

CAPÍTULO III

MARCO CONCEPTUAL: ASPECTOS DE LA PROGRAMACIÓN

En este capítulo, se presenta los elementos de la programación que utilizaremos en nuestra investigación. Se iniciará explorando el pensamiento computacional y su importancia en el aprendizaje de las matemáticas. Este enfoque fomenta habilidades esenciales como la descomposición de problemas, la identificación de patrones y el razonamiento lógico, que son fundamentales para comprender conceptos matemáticos, especialmente en el contexto de las secuencias figurales.

Además, se abordará el concepto de algoritmos y su relación con la programación. Asimismo, a lo largo del capítulo, se introducirá el Scratch. El fin de ello es analizar su interfaz, características y funcionalidades clave, lo que lo convierte en una herramienta accesible y eficaz para el aprendizaje de la programación.

3.1 El pensamiento computacional en la matemática

La tecnología desempeña un papel cada vez más importante en la vida de los estudiantes. Por ello, es que la *Organisation for Economic Co-operation and Development* [OECD] (2018) incluye algunos aspectos del pensamiento computacional (PC) en la prueba PISA sobre la competencia matemática. Entonces, ¿qué es el pensamiento computación? Según Wing (2006), el pensamiento computacional es un conjunto de habilidades útiles e importantes que todo estudiante debe desarrollar en su etapa educativa. Este no debe ser considerado como una competencia exclusiva de un informático o un programador.

Así mismo, la Sociedad Internacional para la Tecnología en la Educación (ISTE) (2016) manifiesta que el pensamiento computacional es un proceso de resolución de problema que presenta las siguientes características: (a) la formulación de problemas, de tal manera que podemos usar la computadora y otras herramientas y poder resolverlos; (b) el análisis y la representación datos, a través de abstracciones como modelos y simulaciones; y (c) el pensamiento algorítmico para automatizar soluciones.

En términos generales, el pensamiento computacional plantea cuatro habilidades: la descomposición, la abstracción, reconocimiento de patrones y el diseño de algoritmos (Wing, 2006; ISTE 2016; Hoyles y Noss, 2015).

De acuerdo con esto, la presente investigación se centrará en los lenguajes de programación por bloques como una estrategia para promover el pensamiento computacional lo que evita la complejidad de la programación convencional (Sáez-López et al., 2019).

Popat y Starkey (2019; como se cita en Diago et al., 2022) sostienen que, en contextos educativos, resulta evidente que la utilización de entornos tecnológicos que emplean programación en bloques tiene un impacto significativo en el desarrollo de habilidades vinculadas al pensamiento crítico, la resolución de problemas, las habilidades sociales y de autocontrol, así como en la adquisición de otros conocimientos académicos.

3.2 Algoritmos y programación en la enseñanza de las matemáticas

Antes de introducir el programa que utilizaremos en nuestra investigación, es fundamental conocer el concepto de algoritmo y programación. Un algoritmo se define como una serie sistemática de pasos ordenados, donde cada uno produce un resultado específico y al ser ejecutados en su totalidad, solucionan un problema o realizan una tarea determinada (Tchounikine, 2017).

De manera general, se puede definir un programa de computadora como "un algoritmo expresado en un lenguaje de programación", lo que implica que puede ser ejecutado directamente en una máquina (Crisci, 2020).

3.3 Lenguaje de programación por bloque

Un lenguaje de programación por bloques es una forma de programación visual en la que se construyen programas mediante la manipulación de bloques gráficos en lugar de escribir código de programación tradicional. Estos bloques representan diferentes funciones, operaciones o acciones que se pueden arrastrar y soltar en un entorno de programación para crear programas. El propósito principal de estos

dispositivos es eliminar errores relacionados con la sintaxis y la formación de estructuras lógicas, lo que simplifica la creación de algoritmos. En lugar de redactar el código manualmente, el usuario emplea diferentes bloques. Cada uno representa un elemento de programación como una estructura de control, una variable, una función, una condición, entre otros (Molina, 2022).

Los bloques de programación suelen presentar una forma y color específico, similar a las piezas de un rompecabezas. Esta característica limita la forma en que se pueden utilizar las piezas al construir un algoritmo, lo que simplifica significativamente la tarea de crear programas con una sintaxis precisa (Utreras y Pontelli, 2020).

Entre todos los entornos de programación por bloques disponibles en la actualidad, Scratch es el que se empleará en nuestro estudio de investigación.

3.4 Lenguaje de programación por bloque: Scratch

Scratch es un sistema de codificación visual con bloques diseñado principalmente para niños. Asimismo, los usuarios pueden programar acciones para objetos o personajes. Para esto, usan una interfaz basada en bloques (Haspekian et al., 2023). Fue creada en el Instituto Tecnológico de Massachusetts (MIT).

Según Chaachoua et al. (2022), los entornos digitales más adecuados para los niños son aquellos que permiten programar sin la complejidad inherente a un lenguaje de programación convencional. Un ejemplo de ello es el Scratch. En este se emplean estructuras visuales como bloques de instrucciones que son manipulables, pues se pueden arrastar y soltar. Además, permite la verificación automática de la sintaxis. Esto posibilita la superposición de las etapas algorítmica y de programación. Al respecto, los alumnos pueden elaborar su algoritmo que consiste en ordenar los bloques de Scratch, lo que les permite ejecutarlo directamente. Asimismo, sostiene que este ambiente tiene la ventaja de estar en constante expansión, y uso en las escuelas de primaria y secundaria.

En la misma línea, Molina (2022) manifiesta que Scratch es una plataforma de programación por bloque que se ofrece de forma gratuita y que tiene como objetivo principal acercar la programación a los estudiantes desde una edad temprana. Es

ampliamente reconocido como uno de los recursos más populares para introducir el pensamiento computacional a estudiantes de todos los niveles educativos.

Tchounikine (2017) sostiene que Scratch se presenta como un lenguaje y entorno de programación que sirve como herramienta para llevar a cabo diversas tareas. En el contexto educativo, su uso implica asignar a los estudiantes una tarea específica para realizar. Es este punto, Scratch se presenta como el medio principal para llevar a cabo total o parcialmente dicha tarea de manera coherente con los objetivos educativos establecidos.

Además, los autores Benton et al. (2017) señalan que, en los primeros niveles educativos, se pueden emplear lenguajes de programación basados en bloques como Scratch para promover tanto el pensamiento computacional como el matemático, dado que estos lenguajes involucran conceptos matemáticos específicos.

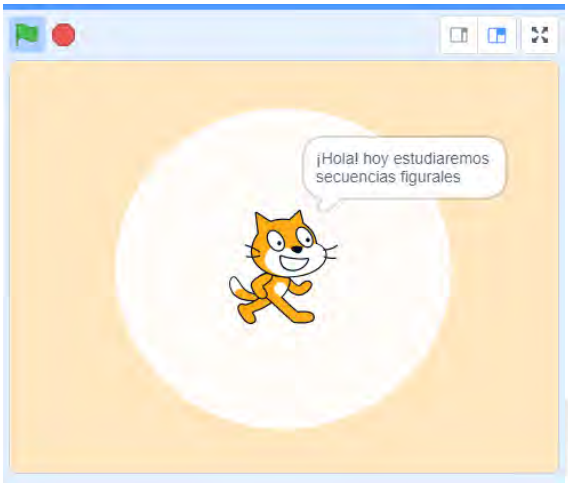
Antes de explorar el entorno de programación de Scratch, es fundamental entender el concepto de código. En el ámbito de la programación, el "código" se refiere a un conjunto de instrucciones escritas en un lenguaje de programación que indican a una computadora o programa qué acciones realizar. Estas instrucciones pueden incluir tareas como ejecutar operaciones matemáticas, crear, modificar o eliminar información almacenada, entre otras.

En Scratch, el código se presenta de forma visual mediante bloques que se ensamblan, lo que simplifica la programación sin la necesidad de escribir texto. Esta metodología permite a los usuarios, especialmente a los principiantes, captar la lógica de la programación de manera más intuitiva.

3.4.1 Presentación de Scratch 3

En Scratch, un script o programa es una secuencia de acciones asignadas a un personaje (sprite) que interactúa en un entorno virtual. Por ejemplo, en este caso, el sprite realiza la acción de decir "¡Hola! Hoy estudiaremos secuencias figurales". Este programa puede activarse al hacer clic en la bandera verde o directamente en el propio personaje. En la figura 4 se presenta uno de los personajes de Scratch.

Figura 4
Personaje o sprite de Scratch



La unidad básica en Scratch es un bloque de código que contiene instrucciones específicas que un sprite puede ejecutar. Cada bloque tiene una función específica como mover un sprite, cambiar su apariencia, emitir sonidos o responder a eventos. Al ensamblar diferentes bloques de código, los usuarios crean scripts que determinan el comportamiento de los sprites en el entorno de programación. En la figura 5 se muestra algunos bloques de código de Scratch.

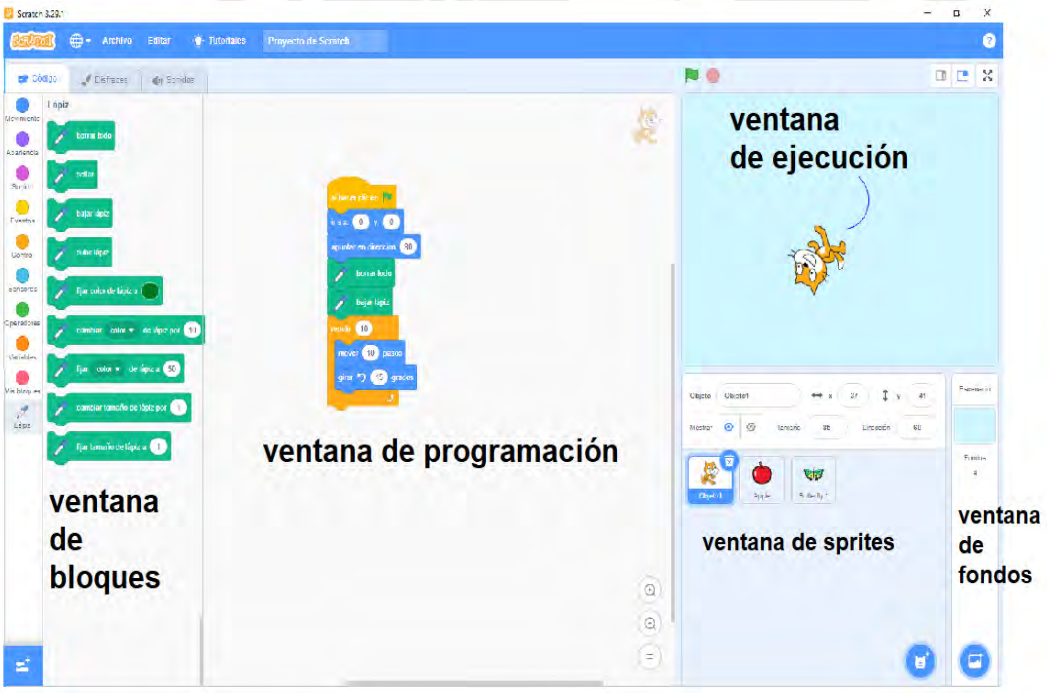
Figura 5
Bloques de código

En el contexto de Scratch, un "proyecto" se refiere a una creación u obra completa que incluye diferentes elementos como sprites, fondos, sonidos y scripts. Puede ser un juego, una animación, una historia interactiva o cualquier otra actividad que se combina en la plataforma. Los proyectos se desarrollan ensamblando bloques de código para definir el comportamiento de los personajes y los escenarios. Esto permite que los usuarios expresen su creatividad y habilidades de programación.

3.4.2 Ventanas de Scratch 3

El entorno de trabajo de Scratch 3 cuenta con varias ventanas principales. Cada una presentan sus características específicas, que facilitan la creación de proyectos. En la figura 6 se pueden observar las más destacadas, que son las siguientes:

Figura 6
Ventanas de Scratch



- **Ventana de ejecución.-** Es el espacio donde se pueden observar los efectos de un programa al ejecutarlo, así como la interacción de los sprites.
- **Ventana de programación. -** Es el lugar donde se ensamblan los bloques de código para crear un script o programa.

o **Ventana de bloques.-** Muestra los diferentes bloques de código disponibles para programar. Estos bloques están organizados en categorías, cada una representada por un color específico, lo que facilita su selección y uso.

o **Ventana de sprites.-** Se presentan todos los personajes (o sprites) que se puede utilizar en el proyecto.

o **Ventana de fondos.-** En esta ventana, se puede visualizar y seleccionar los diferentes fondos que se desea usar en el proyecto.

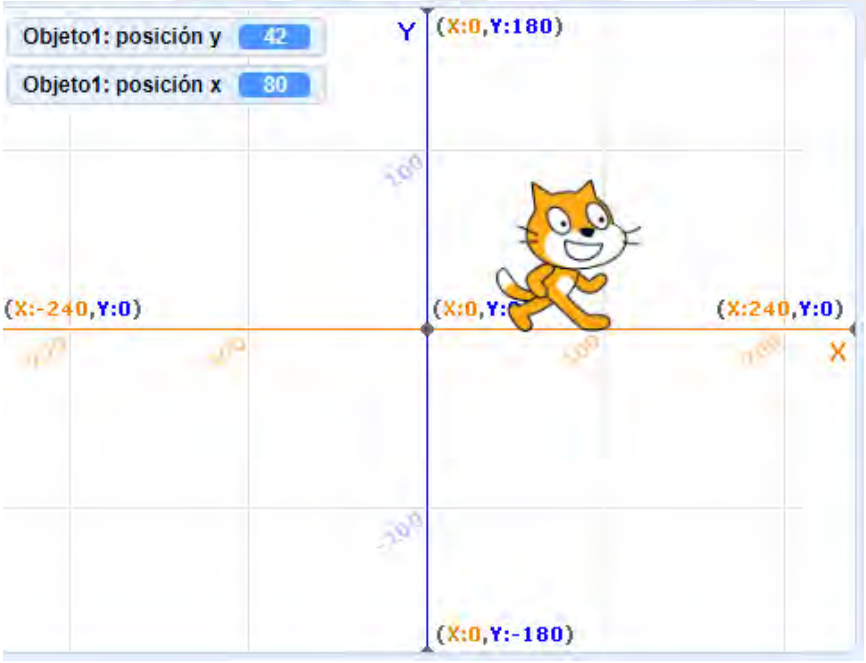
3.4.3 Sistema de coordenadas en Scratch

La ventana de ejecución funciona como un plano de coordenadas con el origen situado en el centro de la misma. Los rangos de coordenadas son fijos.

- Las abscisas van de -240 a 240.
- Las ordenadas van de -180 a 180.

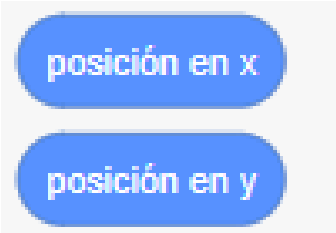
En la figura 7 se muestra la ventana de ejecución, representada como un plano de coordenadas.

Figura 7
Sistema de coordenadas de Scracth



Para cada sprite en Scratch, hay dos variables específicas que se utilizan para almacenar su ubicación en el plano de coordenadas, tal como se muestra en la figura 8.

Figura 8
Variables de ubicación en Scratch



- **Abscisa x.-** Representa la posición horizontal del sprite.
- **Ordenada y.-** representa la posición vertical del sprite.

Estas variables permiten saber dónde se encuentra cada sprite en el entorno de ejecución, lo que facilita su movimiento y acciones en el programa.

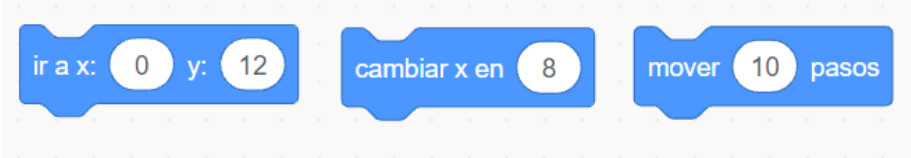
Además, en Scratch, hay bloques de código que permiten cambiar la posición de un sprite en el plano de coordenadas de dos maneras:

Desde una perspectiva absoluta.- Esto implica establecer directamente la posición del sprite en coordenadas específicas. Por ejemplo, "ir a $x: 0$ $y: 0$ " coloca el sprite en el centro del plano.

Desde una perspectiva relativa.- Esto significa modificar la posición del sprite en relación a su ubicación actual. Por ejemplo, "sumar 10 a x " movería el sprite 10 unidades a la derecha desde su posición actual. En la figura 9 se pueden observar algunos bloques que determinar la posición del sprite.

Figura 9

Posiciones de un sprite en la ventana de ejecución



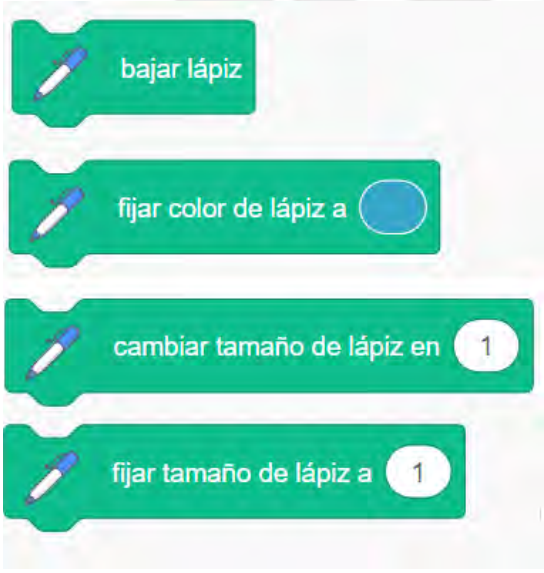
Estos bloques permiten tanto fijar la posición exacta de un sprite como moverlo en función de su ubicación actual.

3.4.4 Lápiz en Scratch

Cada sprite cuenta con un lápiz que le permite dibujar en la ventana de ejecución. La ubicación de la punta del lápiz coincide con las coordenadas del sprite. A este lápiz se le pueden asociar diversos parámetros como la posición de escritura, el color, el tamaño y la intensidad. Por ejemplo, en la figura 10, se pueden observar algunos de estos parámetros.

Figura 10

Parámetros del lápiz en Scratch

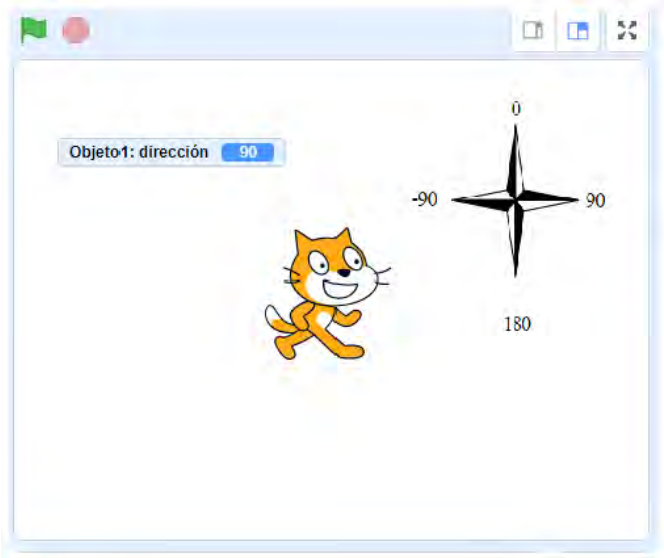


3.4.5 Orientación del plano en Scratch

El plano de referencia de la ventana de ejecución en Scratch tiene una orientación específica. Para cada sprite, hay una variable reservada que guarda la orientación tal como se muestra en la figura 11.

Figura 11

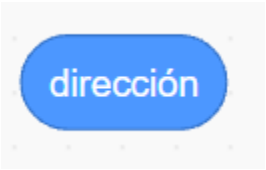
Orientación del plano en Scratch



Variable de orientación.- Cada sprite tiene una variable que almacena su orientación representada por un valor en grados. Esto se ilustra en la figura 12.

Figura 12

Variable de orientación en Scratch



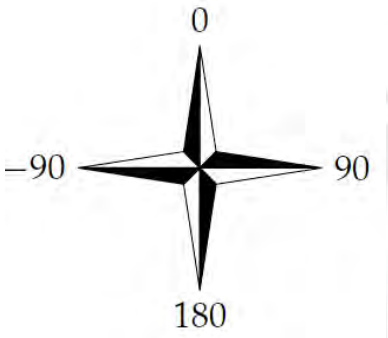
Por otra parte, los valores -90, 0, 90 y 180 representan las direcciones en las que puede orientarse el sprite.

- **0 grados.-** El sprite está mirando hacia arriba.
- **90 grados.-** El sprite está mirando hacia la derecha.
- **-90 grados.-** El sprite está mirando hacia la izquierda.
- **180 grados.-** El sprite está mirando hacia abajo.

Esto se ilustra en la figura 13.

Figura 13

Los valores de la variable orientación

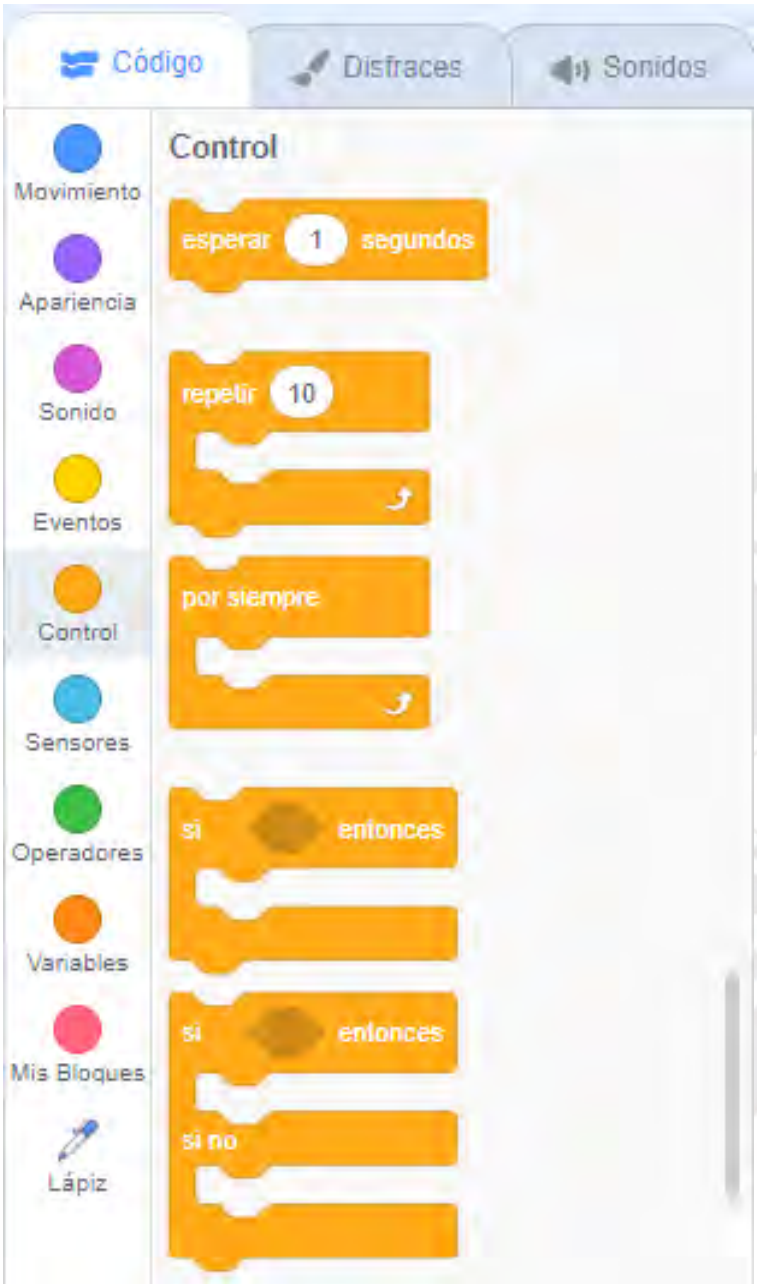


3.4.6 Bloques de código

Los bloques de código en Scratch son piezas gráficas que representan acciones que los sprites pueden realizar como "decir", "moverse" o "reproducir un sonido". Estos bloques se ensamblan visualmente como piezas de un rompecabezas para crear programas o scripts. Cada bloque cumple una función específica como cambiar la apariencia de un sprite o hacer que responda a eventos.

Los bloques están organizados en categorías. Cada una está representada por un color diferente, lo que facilita su identificación y uso. A continuación, se muestra en la figura 14 cómo está organizada.

Figura 14
Bloques de código



Al combinar diversos bloques, los usuarios pueden definir el comportamiento de los sprites y crear animaciones, juegos y proyectos interactivos. Scratch ofrece diversos tipos de bloques de código. Estos están organizados en las siguientes categorías.

Movimiento: Permite mover los sprites.

Apariencia: Sirve para cambiar cómo se ven los sprites.

Sonido: Permite añadir y controlar sonidos.

Eventos: Se encarga de iniciar acciones basadas en eventos como clics o teclas.

Control: Maneja la lógica y el flujo del programa.

Sensores: Detecta interacciones con el entorno.

Operadores: Realiza cálculos y comparaciones.

Variables: Permite crear y usar variables personalizadas.

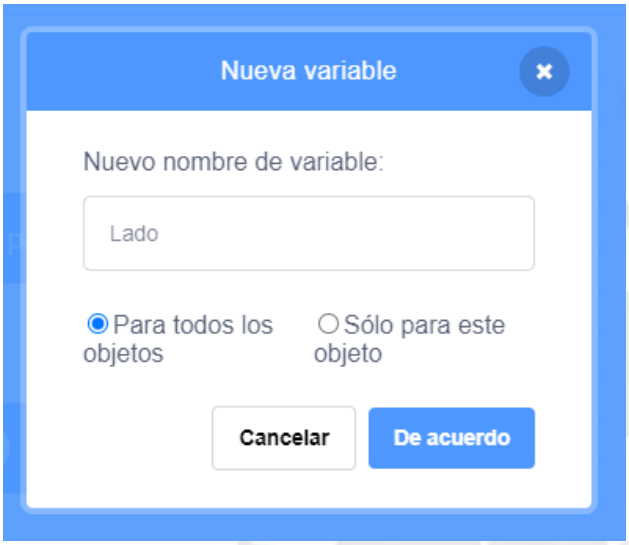
Cada categoría incluye distintos bloques que permiten a los usuarios programar de manera creativa y divertida. Por ejemplo, en la categoría de control, Scratch emplea la estructura de bucle (o "estructura iterativa"), que permite repetir un conjunto de acciones un número específico de veces. Scratch presenta tres variantes de esta estructura como se ilustra en la figura 15.

Figura 15
Bloques de control

En la categoría de variables, los usuarios tienen la opción de crear sus propias variables. Esto les permite almacenar información relevante como puntajes, niveles o cualquier dato que quieran rastrear. Además, pueden decidir si una variable será global

(accesible en todo el programa) o local (solo disponible en un contexto específico), lo que proporciona una mayor flexibilidad en la programación como se muestra en la figura 16.

Figura 16
Creación de una nueva variable en Scratch



Una vez creadas, los usuarios pueden modificar los valores de las variables en cualquier momento, lo que permite que el programa responda a las interacciones del usuario y a los cambios en el entorno. Además, Scratch permite mostrar las variables en la ventana de ejecución, lo que facilita a los usuarios visualizar y seguir el estado de sus programas como se muestra en la siguiente figura 17.

Figura 17
Variable nueva: Lado

En la categoría "Mis bloques", los usuarios pueden crear **bloques personalizados** que agrupan varias instrucciones, lo que simplifica programas complejos y mejora la organización del código. En la figura 18, se muestra un ejemplo de cómo crear un bloque personalizado.

Figura 18
Creación de un bloque personalizado

Estos bloques pueden reutilizarse en diversas partes del proyecto, lo que ahorra tiempo y evita la redundancia. Al encapsular funciones, los usuarios pueden centrarse en la lógica del programa. Además, los nombres descriptivos facilitan la comprensión de la función de cada bloque, lo que resulta especialmente útil en proyectos de mayor envergadura. En la figura 19, se presenta un ejemplo de esto.

Figura 19
Ejemplo de un bloque personalizado

A continuación, se presenta el estudio de la secuencia figural, que servirá como base para realizar el análisis praxeológico de la misma

CAPÍTULO IV

MARCO CONCEPTUAL: ESTUDIO DE LA SECUENCIA FIGURAL

En este capítulo, se examinará la transformación y desarrollo de los conceptos vinculados a la secuencia figural. Para ello, se abordará el estudio de los significados y clasificaciones de objeto matemático en el nivel básico.

4.1 Aspecto epistemológico del objeto matemático secuencia figural

Es fundamental empezar por definir un patrón de manera adecuada para poder establecer una estructura matemática, dado que la secuencia figural está determinada por dicho patrón. En la revisión del Programa curricular de Educación Primaria (2016), se ofrece una definición de patrón que lo describe como una secuencia de símbolos (que pueden ser hablados, gestuales, visuales, geométricos, numéricos, entre otros) que se generan de acuerdo con una regla o procedimiento específico (Bressan y Bogisic, 1996, como se cita en el Programa curricular de Educación Primaria, 2016).

En el ámbito matemático, Mulligan y Michelmore (2009; como se cita en Cetina-Vázquez & Cabañas-Sánchez, 2022) definen un patrón como una regularidad que, en la mayoría de los casos, involucra relaciones numéricas, espaciales o lógicas. De manera similar, Castro et al. (2010; como se cita en Zapatera, 2018) describen un patrón como una regularidad observada en diversas situaciones o hechos que se espera que se repita en el futuro.

Por otro lado, en la investigación realizada por Bustamante (2017), se explica que un patrón puede ser definido como una regla o conjunto de reglas que los objetos (como números, figuras, letras, etc.) siguen para generar otros objetos del mismo tipo. Además, se destaca que no necesariamente hay un único patrón para un conjunto de elementos, ya que pueden existir varios patrones que, aunque diferentes, generen los mismos elementos. Sin embargo, en este caso, la definición de patrón no requiere que los elementos del conjunto sigan un orden específico.

Por lo tanto, en esta investigación, se tomará en cuenta un tipo particular de patrón: aquellos que generan objetos en un orden específico. Este tipo de patrón será clave para la definición de secuencia figural que se propone. En este contexto, un patrón

se refiere a una secuencia o disposición de figuras que sigue una regla o relación específica, la cual se repite a lo largo de la secuencia. Al respecto, Zapatera (2018) señala que un patrón implica secuencias de elementos que se construyen siguiendo una norma determinada y los estudiantes, a partir de ejemplos particulares, deben deducir esa regla para generalizar el patrón y continuar la secuencia.

Por otra parte, una secuencia figural es un conjunto de elementos visuales como formas geométricas o dibujos, que siguen un orden determinado y preestablecido, generados por un patrón específico. Los elementos de esta secuencia se etiquetarán utilizando números cardinales. En lugar de utilizar términos como "1° término", "2° término", "3° término", etc., se emplearán etiquetas como "figura 1", "figura 2", "figura 3" y así sucesivamente. Esto se debe a que estos elementos están ordenados de manera secuencial y es posible asignarles una posición dentro del conjunto. De esta manera, un elemento en la secuencia en la posición *n* será denominado "figura n", representado como $fig.n$.

Estas secuencias se caracterizan por seguir una estructura repetitiva o creciente que puede involucrar cambios en diversas características de las figuras como su forma, tamaño, número, orientación o color.

4.2 Clases de secuencias figurales

Según Warren (2005), Warren y Cooper (2006), y Rivera (2013), en edades tempranas, se exploran dos tipos de secuencias que favorecen la generalización: aquellas que se componen de patrones de repetición y aquellas basadas en patrones recurrentes (como se cita en Uicab et al., 2022).

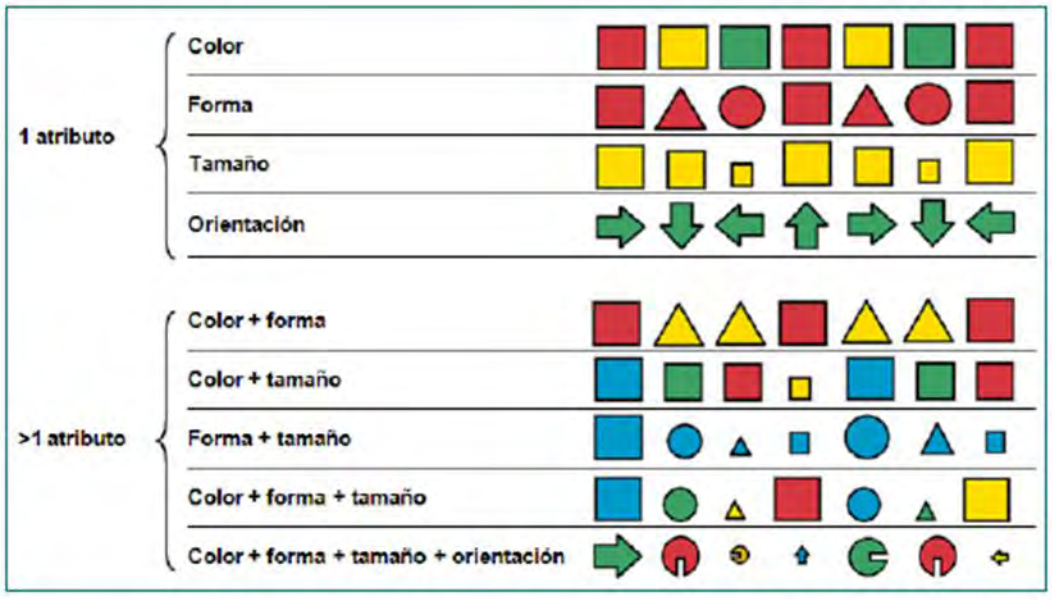
Los **patrones de repetición** son secuencias que contienen un conjunto constante de elementos que se repiten a lo largo de la secuencia, siendo este conjunto conocido como la **unidad de repetición** (Papic y Mulligan, 2007; Zazkis y Liljedahl, 2002, como se cita en Wijns et al., 2019). En otras palabras, un patrón de repetición puede entenderse como una unidad claramente identificable que se repite de manera continua. Esta repetición forma la secuencia, la cual presenta una estructura cíclica generada por

la iteración constante del patrón (Uicab et al., 2022). Por ejemplo, en la figura 20, la unidad de repetición es tres triángulos y dos cuadrados.

Figura 20
Secuencia conformada por un patrón de repetición geométrico

En los patrones de repetición, los elementos se repiten según uno o más atributos como el color, la forma, el tamaño o la orientación. Estos patrones son los más sencillos de identificar por lo que se emplean en la Educación Inicial y en los primeros años de la Educación Primaria**.** La dificultad de los patrones de repetición aumenta cuando se incorporan más de un atributo o cuando se utilizan atributos menos evidentes (Zapatera, 2018). Por ejemplo, en la figura 21 podemos observar algunos ejemplos de las secuencias de repetición con 1 o más atributos.

Figura 21
Secuencias de patrones de repetición con uno o más atributos

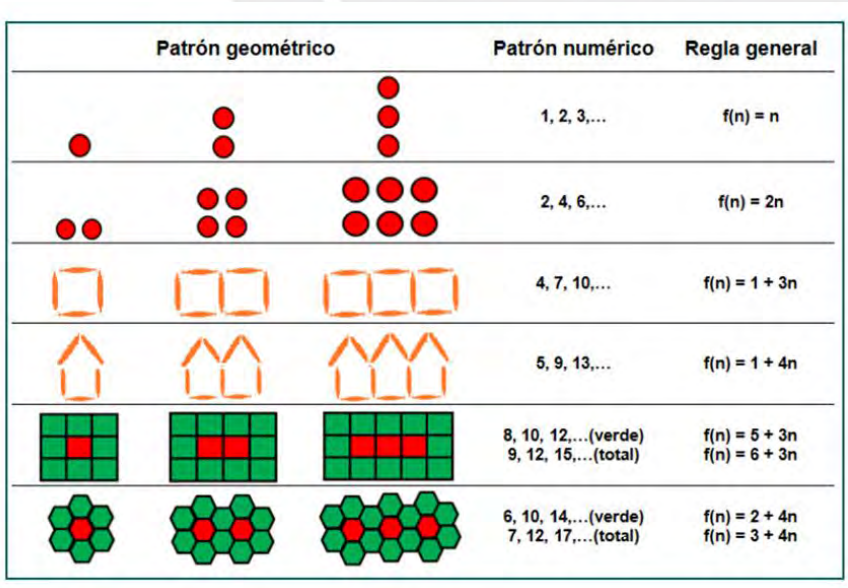


*Fuente. T*omado de Zapatera, 2018, p.56

Este tipo de secuencias permite a los estudiantes trabajar dentro de un conjunto específico de elementos. También, les ayuda a identificar la unidad de repetición (el patrón) y a generar nuevas secuencias que sigan la misma regla estructural. Además, los estudiantes pueden extender las secuencias existentes. Para esto, añade una o más unidades de repetición o bien completar una secuencia incompleta colocando las piezas faltantes de acuerdo con el patrón (Uicab et al., 2022).

A diferencia de los patrones repetitivos, los **patrones recurrentes** son secuencias en las que la cantidad de elementos varía de un término a otro. Es decir, el número de elementos aumenta o disminuye progresivamente a lo largo de la secuencia, siguiendo una regla de formación específica. Estos patrones pueden ser de tipo **geométrico** o **numérico** y presentan un nivel de dificultad superior al de los patrones repetitivos. La complejidad de los patrones recurrentes crece a medida que aumenta el número de elementos en cada término y la dificultad de la regla que rige su formación (Zapatera, 2018). A continuación, en la figura 22 se presenta algunos ejemplos de secuencias con este tipo de patrón.

Figura 22
Secuencias de patrones recurrentes



Fuente. Tomado de Zapatera, 2018, p.57

Generalización de patrones

La generalización de patrones se puede definir como el proceso cognitivo de identificar y extender una regla o estructura subyacente que describe cómo cambian o se relacionan las figuras en una secuencia. Este proceso implica reconocer regularidades o tendencias en una serie de elementos (formas, colores, tamaños, posiciones, etc.) y usar esa comprensión para predecir los elementos que seguirán o para formular una descripción general de cómo se desarrollará la secuencia. Es decir, la generalización de patrones involucra tres pasos: reconocer una propiedad común, generalizar esa propiedad a todos los términos de la secuencia y usar la propiedad común para establecer una regla que permita determinar cualquier término dentro de la secuencia (Dreyfus, 1991, como se cita en Zapatera, 2018).

A continuación, se expone la organización matemática mixta, la cual facilitará el análisis de las diversas tareas que serán tomadas de dos artículos fundamentales: el primero, de Alayrangues et al. (2019), y el segundo, de Ye et al. (2024).

CAPÍTULO V

MODELO PRAXEOLÓGICO DE REFERENCIA

En este capítulo, se realiza la reconstrucción de la praxeología mixta basada en la investigación de Crisci (2020). A continuación, se presenta el análisis de la organización matemática de la secuencia figural, que es fundamental para comprender y caracterizar la organización matemática mixta. Esta praxeología de referencia nos permitirá analizar la secuencia didáctica acerca de la secuencia figural implementado en el entorno de Scratch.

5.1 Análisis Praxeológico de la secuencia figural

En esta sección, se expone la reconstrucción del Modelo Praxeológico de Referencia (MPR) basada en los conceptos desarrollados en el capítulo anterior. La revisión de otras investigaciones como las de Uicab et al. (2022); y Zapatera (2018) fuentes claves para este proceso. Además, se realiza un análisis de las actividades presentes en el cuaderno de trabajo de segundo grado de primaria, que complementa la comprensión de la praxeología de una institución dentro del MPR.

Se considera las siguientes notaciones para presentar los elementos de la praxeología.

- $\boldsymbol{T_i}$: Tipo de tarea
- $\boldsymbol{T'_i}$: Subtipo de tarea del tipo de tarea i
- $\boldsymbol{T^*_k}$: Ingrediente de una técnica
- $\boldsymbol{GT}_i$: Generador del tipo de tarea i
- $\boldsymbol{V_i}$: Variable asociadas al tipo de tarea i
- $\boldsymbol{t_{i,j}}$: Tareas, donde j representa la tarea j dentro del tipo de tarea i
- $\boldsymbol{\tau}$: Técnica
- $\boldsymbol{\vartheta}$: Tecnología
- $\boldsymbol{\Theta}$: Teoría

Para iniciar el análisis de las tareas del Tipo de tarea $\boldsymbol{T_{SF}}$: Generalizar el patrón en una secuencia figural, es fundamental identificar las variables que intervienen en el proceso. En la tabla 7, se detallan los valores de las variables, donde **"n"** es un número entero que representa la cantidad de elementos en una secuencia

Tabla 7
Variables y valores de la Variable

Variables	**Valores de la Variable**	
$\boldsymbol{V_{fig}}$**: tipo de figura**	$\boldsymbol{V_{fig,1}}$: poligonal (figura simple)	$\boldsymbol{V_{fig,1,a}}$:: abierta
		$\boldsymbol{V_{fig,1,b}}$:: cerrada
	$\boldsymbol{V_{fig,2}}$: figura compuesta	
$\boldsymbol{V_{patrón}}$**: tipo de patrón**	$\boldsymbol{V_{patrón,1}}$: repetición	$\boldsymbol{V_{patrón,1,a}}$: un atributo
		$\boldsymbol{V_{patrón,1,b}}$: con 2 o más atributos
	$\boldsymbol{V_{patrón,2}}$: recurrente	$\boldsymbol{V_{patrón,2,a}}$: creciente
		$\boldsymbol{V_{patrón,2,b}}$: decreciente
$\boldsymbol{V_n}$**: tamaño de n**	$\boldsymbol{V_{n,1}}$: pequeño ($n < 5$)	
	$\boldsymbol{V_{n,2}}$: grande ($n \geq 5$)	

El tipo de tarea y los subtipos de tareas reconocidos en la praxeología de la secuencia figural son los siguientes:

- $\boldsymbol{T'_{und.patrón}}$**:** Identificar la unidad de patrón presente en una secuencia figural
- $\boldsymbol{T'_{Completar}}$**:** Completar la figura n de una secuencia figural
- $\boldsymbol{T'_{valor\ fig.n}}$**:** Calcular el valor de la $fig.n$ en la secuencia

- $T_{cant.de\ elementos}$: Determinar la cantidad de elementos que componen la figura n en una secuencia figural

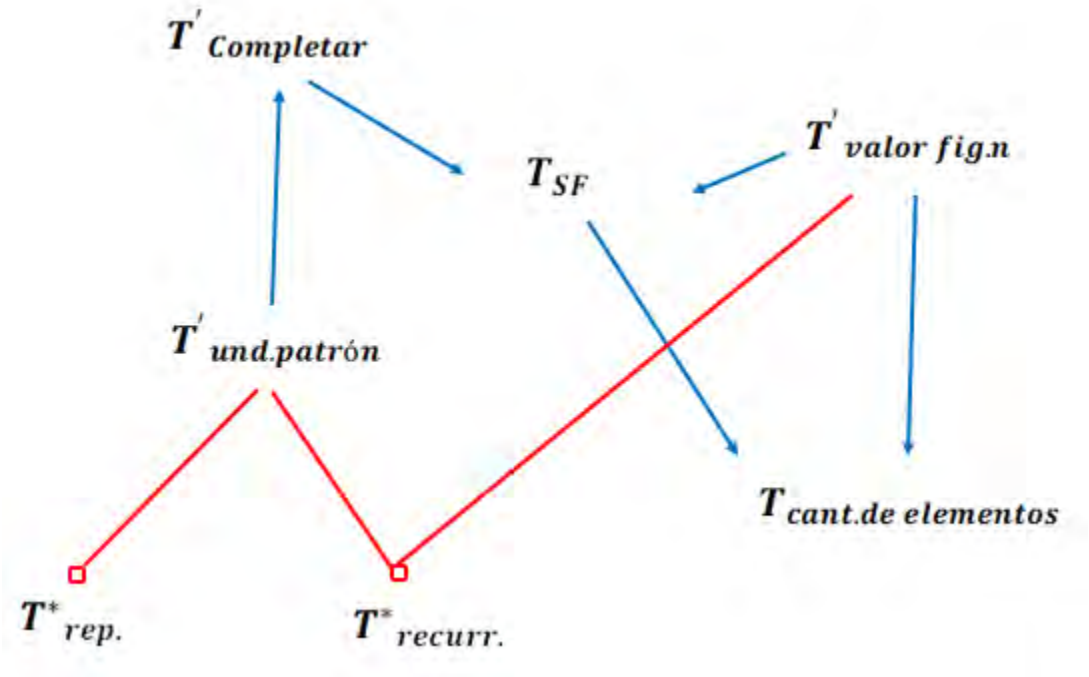
Relación entre los tipos de tareas

En nuestro análisis, consideramos que:

- $T'_{und.patrón}$ es un requisito previo para $T'_{valor\ fig.n}$ debido a que la técnica para lograr $T'_{und.patrón}$ tiene como ingrediente a T^*_{recurr} .
- $T'_{und.patrón}$ constituye un subtipo de tarea de $T'_{Completar}$.
- Los tipos de tareas $T'_{Completar}$ y $T'_{valor\ fig.n}$, son subtipos de tarea del Tipo de tarea T_{SF}: Generalizar el patrón en una secuencia figural.
- $T_{cant.de\ elementos}$ tiene como subtipo de tarea a T_{SF}

En la figura 23 se presenta la relación entre los diferentes tipos de tareas de la organización matemática de la secuencia figural abordados en esta sección.

Figura 23
Organización matemática de la secuencia figural



En la Figura 23, se ilustra la conexión entre los diferentes tipos de tareas de OM que se han presentado en esta sección. De aquí en adelante, a lo largo de la tesis, se

$T^* \circ\!- T'$

utilizará la notación $T' \longrightarrow T$ y para indicar que T' es un subtipo de la tarea T, y que T^* es un ingrediente de al menos una técnica de T'.

Para comenzar el análisis se describen las tareas de los subtipos de tareas $\boldsymbol{T_{SF}}$: Generalizar el patrón en una secuencia figural. Además, se utilizará las variables $\boldsymbol{V_{fig}}$, $\boldsymbol{V_{patrón}}$ y $\boldsymbol{V_n}$ tal como se detalla en la Tabla 7.

A continuación, se describirá las técnicas de los subtipos de tareas $\boldsymbol{T'_{und.patrón}}$ y $\boldsymbol{T'_{valor\ fig.n}}$

Técnicas de $\boldsymbol{T'_{und.patrón}}$

En la tabla 8, se describirá las técnicas de los ingredientes $\boldsymbol{T^*_{rep}}$ y $\boldsymbol{T^*_{recurr}}$

Tabla 8
Técnicas del subtipo $\boldsymbol{T'_{und.patrón}}$

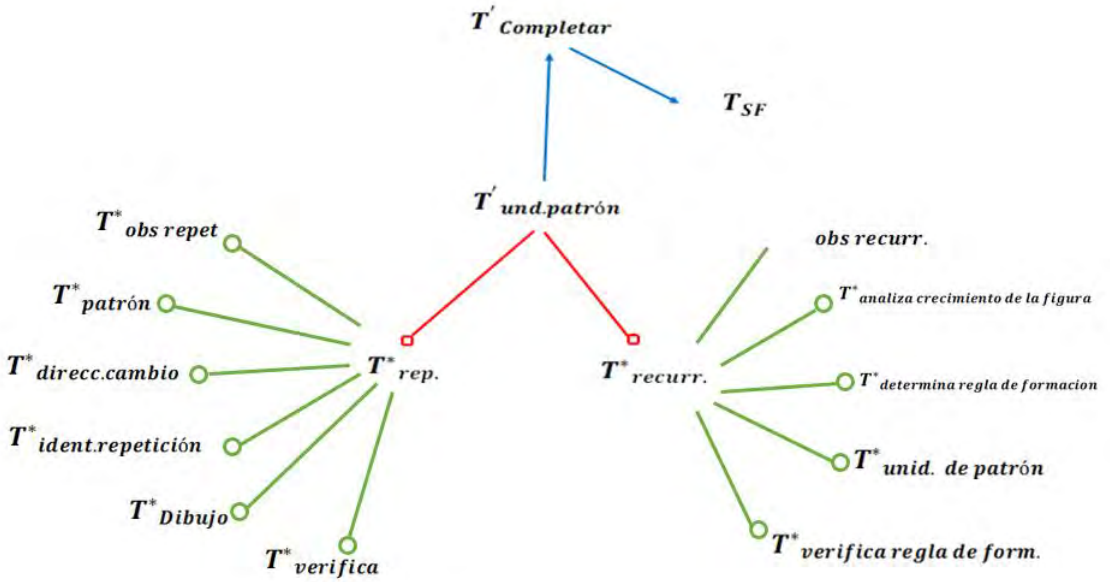
Técnica	Ingredientes de la técnica
$\boldsymbol{\tau_{repetición}}$	$\boldsymbol{T^*_{obs\ repet}}$**:** Observa la secuencia figural con la finalidad de buscar atributos para establecer relaciones. $\boldsymbol{T^*_{busca\ atributos}}$ **:** Busca atributos simples; es decir, observa si hay elementos que se repiten de manera consistente como formas geométricas, colores o direcciones de movimiento. $\boldsymbol{T^*_{direcc.cambio}}$**:** Determina si los elementos cambian de una figura a otra de una manera específica como rotación, reflexión, traslación o cambio en el tamaño. $\boldsymbol{T^*_{unid.de\ repetición}}$**:** Identifica la unidad de repetición. $\boldsymbol{T^*_{Dibujo}}$**:** Dibuja la figura del término pedido.

	$T^*_{verifica}$: Verifica si la respuesta sigue el patrón observado en la secuencia figural y si se ajusta a las reglas establecidas por la unidad de patrón identificada.
$\tau_{recurrente}$	$T^*_{obs\ recurr.}$: Observa la secuencia figural para comprender cómo cambian o crecen en cada paso la secuencia. $T^*_{analiza\ crecimiento\ de\ la\ figura}$: Analiza cómo cambian las características de las figuras a medida que avanza la secuencia. Esto podría incluir cambios en el número de lados, el tamaño de los ángulos, la longitud de los lados, etc. $T^*_{determina\ regla\ de\ formacion}$: Determina la regla de formación que gobierna cómo cambian las figuras a medida que progresa la secuencia. Esto podría ser un aumento constante en el número de lados, un cambio en la orientación de los ángulos, una duplicación del tamaño, etc. $T^*_{unid.\ patrón\ recurrente}$: Una vez que hayas identificado la regla de formación busca la unidad más pequeña que se repite en cada paso de la secuencia. Esta unidad representará el cambio mínimo que ocurre de una figura a otra. $T^*_{verifica\ regla\ de\ form.}$: Verifica si la regla identificada se mantiene constante a lo largo de la secuencia. Cada figura debe seguir el mismo patrón recurrente. T^*_{Dibujo}: Dibuja la figura del término pedido.

En la figura 24, se resumen los tipos de tarea identificados de los ingredientes T^*_{rep} y T^*_{recurr}

Figura 24

Diagrama de tipos de tareas alrededor T^*_{rep} **y** T^*_{recurr}



Técnica de $T'_{valor\ fig.n}$

En la tabla 9, se describe la técnica del subtipo $T'_{valor\ fig.n}$ del tipo de tarea T_{SF}

Tabla 9

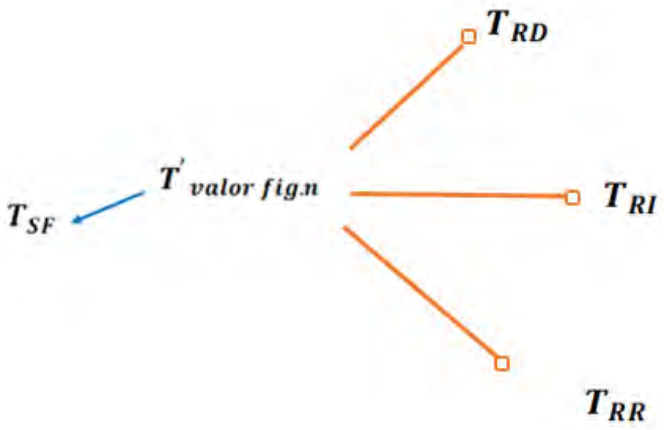
Técnica del subtipo $T'_{valor\ fig.n}$

Técnica	Ingrediente de la técnica
$\tau_{recuento}$	T^*_{RD}: Recuento sobre el dibujo, dibuja la figura y cuenta sus elementos. T^*_{RI}: Recuento iterativo. Se parte de la primera figura y suma el patrón de recurrencia de forma iterada hasta el término requerido. T^*_{RR}: Recuento recursivo, se parte de una figura cualquiera y suma el patrón de recurrencia de forma iterada hasta el término requerido.

En la figura 25 ilustra la conexión entre los distintos tipos de tareas en torno a $T_{valor\ fig.n}$

Figura 25

Diagrama de tipos de tareas alrededor de $T_{valor\ fig.n}$



Técnicas de $T_{cant.de\ elementos}$

En el tipo de tarea $T_{cant.de\ elementos}$ se han identificado dos técnicas: $\tau_{\text{Funcional}}$ y $\tau_{\text{proporcional}}$. La primera técnica está vinculada a la relación entre dos variables: la posición de la figura y su número de elementos. Para ello, se utiliza una función afín $f(n) = an + b, b \neq 0$, donde a representa el patrón de recurrencia y b es el término independiente, que permanece constante.

En la tabla 10, se explica las técnicas de $T_{cant.de\ elementos}$

Tabla 10

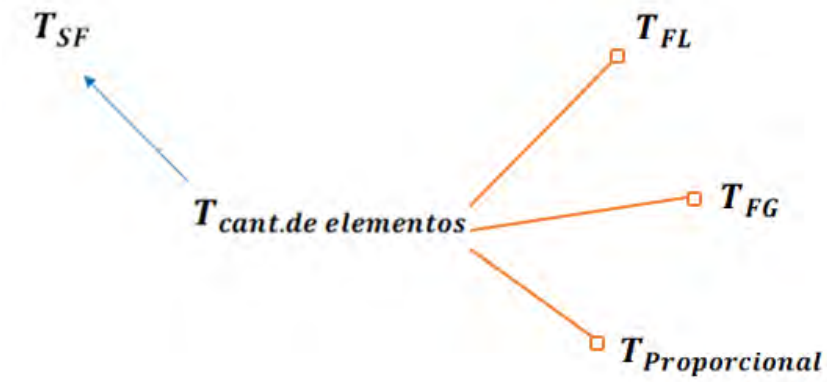
Técnicas de $T'_{cant.de\ elementos}$

Técnica	Ingrediente de la técnica
$\tau_{\text{Funcional}}$	**T^*_{FL}:** Función local, utiliza una función para hallar los elementos de un determinada figura. **T^*_{FG}:** Función global, utiliza una función para hallar los elementos de un figura cualquiera.

$\tau_{\text{proporcional}}$	$T^*_{Proporcional}$: **H**alla, erróneamente, el número de elementos mediante razonamientos proporcionales con multiplicaciones o reglas de tres con base en una función lineal $f(n) = an$, donde no se considera el término independiente que se mantiene constante.

La figura 26 muestra cómo se interrelacionan los distintos tipos de tareas alrededor de $T_{cant.de\ elementos}$

Figura 26
Diagrama de tipos de tareas alrededor de $T_{cant.de\ elementos}$



A continuación, se analizan cada una de las tareas correspondientes a los subtipos de tarea $T'_{Completar}$, $T'_{und.patrón}$ y $T'_{valor\ fig.n}$ en los siguientes párrafos.

Figura 27
Ejercicio 2a - *Cuaderno de trabajo 2 do. Grado de primaria*

Fuente. Matemática 2. Cuaderno de trabajo de segundo grado de primaria, 2024, p.33

Tipo de tarea T_{SF}: Generalizar el patrón en una secuencia figural,

GT_{SF}: [Generalizar el patrón en una secuencia figural, V_{fig} , $V_{patrón}$, V_n]

Subtipo de tarea: $T'_{Completar}$**:** Completar la figura n de una secuencia figural

Tarea

- $t_{Completar,1}$**:** Completar la figura n presente en una secuencia figural compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón de recurrencia creciente y el valor de n pequeño.

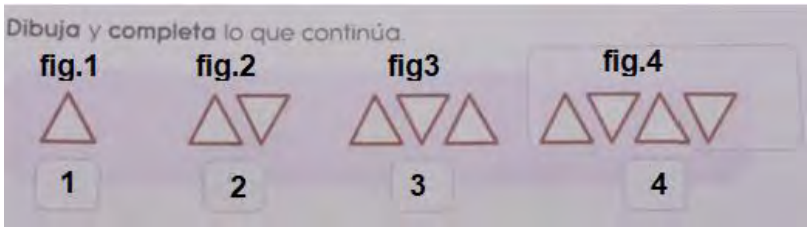
Técnica:

- ✓ $T^*_{obs\ recurr.}$**:** Se observa que la secuencia de figuras está compuesta por triángulos que van aumentando de uno en uno.
- ✓ $T^*_{analiza\ crecimiento\ de\ la\ figura}$: Analizando la secuencia se tiene que en la figura 1, se encuentra un triángulo. En la figura 2, además del triángulo original, se añade otro triángulo de igual tamaño y forma, pero invertido con respecto al primero. Finalmente, en la figura 3, se agrega un tercer triángulo idéntico al de la figura 1.
- ✓ $T^*_{determina\ regla\ de\ formacion}$**:** La secuencia figural es creciente (aumenta de forma progresiva el número de triángulos) y lineal (porque la regla que lo rige es n).
- ✓ $T^*_{unid.\ patrón\ recurrente}$**:** Se busca la unidad más pequeña que se repite en cada paso de la secuencia. Esta unidad representará el cambio mínimo que ocurre de una figura a otra.

  La unidad de patrón recurrente es agregar un triángulo.
- ✓ T^*_{Dibujo}**:** Dibuja la figura correspondiente al término solicitado, tal como se muestra en la figura 28.

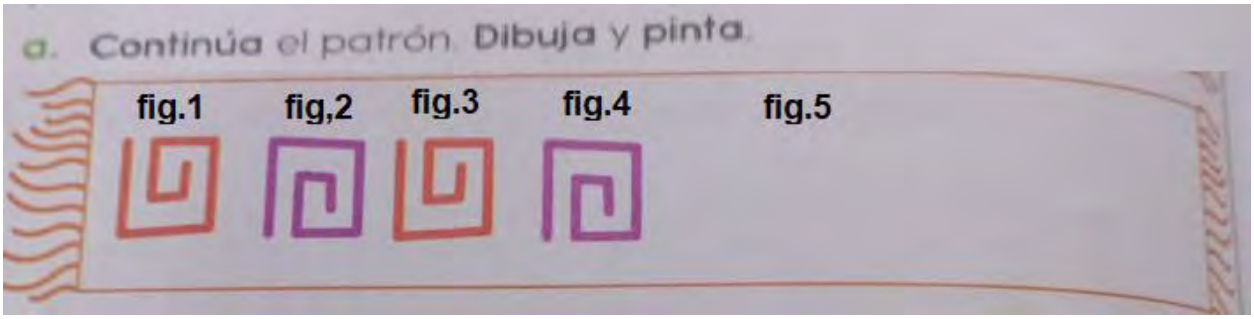
Figura 28

Solución de la secuencia figural formada por triángulos



Fuente. Matemática 2. Cuaderno de trabajo de segundo grado de primaria, 2024, p.33

Figura 29
Ejercicio 2a - Cuaderno de trabajo 2 do. Grado de primaria



Fuente. Matemática 2. Cuaderno de trabajo de segundo grado de primaria, 2024, p.12

Tipo de tarea T_{SF}: Generalizar el patrón en una secuencia figural,

GT_{SF}: [Generalizar el patrón en una secuencia figural, V_{fig} , $V_{patrón}$, V_n]

Subtipo de tarea: $T'_{Completar}$**:** Completar la figura n de una secuencia figural.

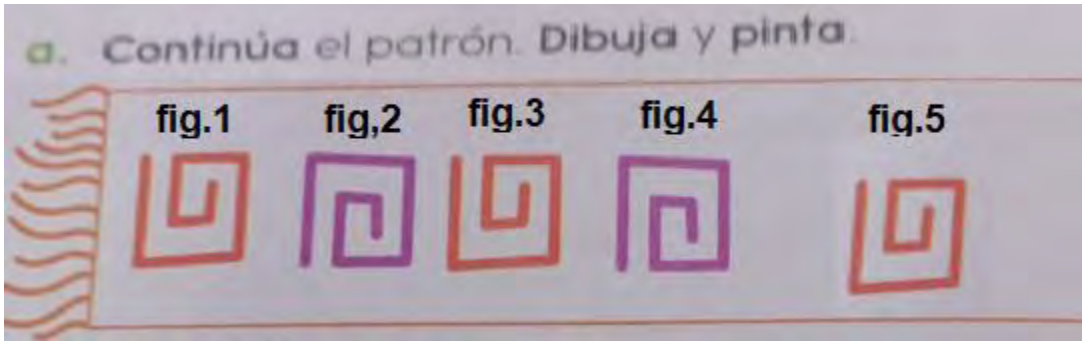
Tarea

- $t_{Completar,2}$**:** Completar la figura n presente en una secuencia figural compuesta por figuras poligonales abiertas, donde cada figura sigue un patrón de repetición y el valor de n grande.

Técnica $\tau_{repetición}$

- ✓ $T^*_{obs\ repet}$**:** Se observa que la secuencia consiste en formas rectangulares en espiral, las cuales experimentan cambios en color y dirección de movimiento entre una figura y otra.
- ✓ $T^*_{busca\ atributos}$**:** Al analizar la secuencia, se observa que los colores que se repiten siguen un patrón de anaranjado seguido de morado.
- ✓ $T^*_{direcc.cambio}$: En cuanto a la dirección de movimiento, se observa una rotación de 180° en sentido antihorario y una simetría de la figura para pasar de una figura a la siguiente.
- ✓ $T^*_{unid.de\ repetición}$**:** Por lo tanto, se concluye que la unidad de repetición es espiral 1 anaranjado, espiral 2 morado. Entonces, la figura 30 es un espiral 1 anaranjado.
- ✓ T^*_{Dibujo}**:** Dibuja la figura del término pedido, tal como se muestra en la figura 30.

Figura 30
Solución de la secuencia figural formada por espirales



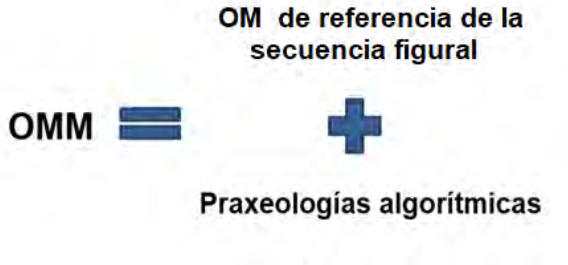
Fuente. Matemática 2. Cuaderno de trabajo de segundo grado de primaria, 2024, p.12

5.2 Caracterización de la Organización Matemática Mixta

Debido a la praxeología de referencia de la secuencia figural establecida anteriormente, se procede analizar la secuencia didáctica relacionada con este objeto matemático, que se ha implementado en el entorno de Scratch.

En esta sección, se ha desarrollado un modelo praxeológico para describir la praxeología mixta elaborada. Para ello, se ha abordado la pregunta de investigación: ¿Cuáles son las praxeologías mixtas que se presentan en las tareas matemáticas con Scratch asociadas a la secuencia figural? En particular, la figura 31 muestra que esta praxeología mixta elaborada se define por la praxeología de referencia de la secuencia figural y por praxeologías algorítmicas que identificaremos a lo largo de esta sección, que son derivadas del análisis de la secuencia didáctica.

Figura 31
Relación entre OMM elaborada y OM de referencia



Fuente. Adaptado de Crisci, 2020, p.88

Se consideran las siguientes notaciones para exponer los elementos de la praxeología algorítmica.

- $\boldsymbol{T_{prog}[T_i]}$ **:** Tipo de tarea en el programa Scratch
- $\boldsymbol{T'_{prog}[T_i]}$ **:** Subtipo de tarea en el programa Scratch
- $\boldsymbol{T^*_{prog}[\,t]}$**:** Ingrediente de una técnica
- $\boldsymbol{t_{prog}}$: Tarea
- $\boldsymbol{GT_{prog}[T_i]}$: Generador de tipos de tarea en el programa Scratch
- $\boldsymbol{V_{prog}}[variable]$: Variable en el programa Scratch
- $\boldsymbol{\tau_{prog}[T_i, i]}$**:** Técnica en el programa Scratch

Por otro lado, el tipo de tarea que se espera que los estudiantes realicen en la praxeología mixta elaborada utilizando el entorno de Scratch es el siguiente:

Tipo de tarea $\boldsymbol{T_{prog}[T_{SF}]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_{SF}}$**.**

Variables introducidas en el programa Scratch

Utilizando las variables $\boldsymbol{V_{fig}}$, $\boldsymbol{V_{patrón}}$ y $\boldsymbol{V_n}$ que hemos identificado dentro del marco de la praxeología de referencia de la secuencia figural para el tipo de tarea $\boldsymbol{T_{SF}}$, se pueden establecer conexiones entre los ejercicios propuestos en la secuencia y las tareas creadas mediante el tipo de tarea $\boldsymbol{T_{prog}[T_{SF}]}$**.** Las variables que intervienen en este proceso se detallan en la tabla 11.

Tabla 11
Variables introducidas en el programa Scratch.

Variables	**Valores de la Variable**	
$\boldsymbol{V_{prog}}[\textbf{tipo de figura}]$	$\boldsymbol{V_{prog}}$[tipo de figura1]: poligonal	$V_{prog}[\textbf{t}$ipo de figura $1, \mathrm{a}]$: abierta
		$V_{prog}[\textbf{t}$ipo de figura $1, \mathrm{b}]$

		: cerrada
	V_{prog}[tipo de figura, 2]: figuras compuesta	
$\boldsymbol{V_{prog}}$[**tipo de patrón**]	V_{prog}[tipo de patrón 1] : repetición	V_{prog}[tipo de patrón 1] **:** un atributo
		V_{prog}[tipo de patrón 1] **:** con 2 o más atributos
	V_{prog}[tipo de patrón **2**] : recurrente	V_{prog}[**t**ipo de patrón **2**, **a**] : creciente
		V_{prog}[**t**ipo de patrón **2**, **b**] : decreciente
$\boldsymbol{V_{prog}}$[**cantidad de pasos**] **(longitud)**	$\boldsymbol{V_{prog}}$[cantidad de pasos, 1]: creciente	
	$\boldsymbol{V_{prog}}$[cantidad de pasos, 2]: decreciente	
	$\boldsymbol{V_{prog}}$[cantidad de pasos, 3]: constante	
$\boldsymbol{V_{prog}}$[á**ngulo de giro**]	V_{prog}[ángulo de giro, 1]: $\alpha < 90°$	V_{prog}[ángulo de giro, 1, a] : sentido horario V_{prog}[ángulo de giro, 1, b] : Sentido antihorario
	V_{prog}[ángulo de giro, 2]: $\alpha = 90°$	V_{prog}[ángulo de giro, 2, a] : sentido horario V_{prog}[ángulo de giro, 2, b]

		: Sentido antihorario
	V_{prog}[ángulo de giro, 3]:	V_{prog}[ángulo de giro, 3, a]
	$\alpha > 90°$	: sentido horario
		V_{prog}[ángulo de giro, 3, b]
		: Sentido antihorario

Además, los tipos de tareas de $\boldsymbol{T_{prog}[T_{SF}]}$ reconocidos en la praxeología mixta son los siguientes:

- $\boldsymbol{T_1}$**:** Identificar la unidad de patrón presente en una secuencia figural
- $\boldsymbol{T_2}$**:** Traduce patrones en nuevos patrones con la misma regla estructural
- $\boldsymbol{T_3}$**:** Crear una secuencia a partir de una secuencia dada
- $\boldsymbol{T_4}$**:** Extiende patrones al menos una unidad de patrón,
  por ejemplo, $\boldsymbol{T_{prog}[T_1]}$ es una tarea de la praxeología mixta.

5.3 Modelo de la Organización Matemática Mixta (OMM)

En esta sección, se emplea la organización matemática de referencia para describir las tareas de OMM que se basa en el análisis de formatos tanto en papel como digital que se centra en las secuencias figurales.

5.3.1 Tareas de $\boldsymbol{T_{prog}[T_{SF}]}$ involucradas en la OMM

En la OMM elaborada, se han identificado diez tareas, las cuales hemos clasificado en ocho subtipos diferentes de los tipos de tarea de $\boldsymbol{T_{prog}[T_{SF}]}$. Para esto, se basa en las variables identificadas. En la tabla 12, se clasifica y se caracteriza las tareas de $\boldsymbol{T_{prog}[T_{SF}]}$ en función de las diversas formas en que se manifiestan las variables.

Tabla 12
Tareas de $\boldsymbol{T_{prog}[T_{SF}]}$

Tipo de tarea	Subtipo de tarea	tarea	$\boldsymbol{V_{prog}[tipo\ de\ figura]}$	$\boldsymbol{V_{prog}[tipo\ de\ patrón]}$	$\boldsymbol{V_{prog}[cant.de\ pasos]}$	$\boldsymbol{V_{prog}[\measuredangle\ de\ giro]}$
T_1	$\boldsymbol{T'_{prog}[T_1,1]}$	$t_{1,1,1}$	Poligonal cerrada	Patrón de repetición	constante	$\alpha < 90º$ antihorario
		$t_{1,1,2}$				$\alpha > 90º$ antihorario
		$t_{1,1,3}$				
	$\boldsymbol{T'_{prog}[T_1,2]}$	$t_{1,2}$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario
T_2	$\boldsymbol{T'_{prog}[T_2,1]}$	$t_{2,1}$	Poligonal abierta	Patrón recurrente creciente	creciente	$\alpha = 90º$ antihorario
T_3	$\boldsymbol{T'_{prog}[T_3,1]}$	$t_{3,1}$	Poligonal cerrada	Patrón recurrente creciente	constante	$\alpha < 90º$ $\alpha > 90º$ antihorario
	$\boldsymbol{T'_{prog}[T_3,2]}$	$t_{3,2}$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario
	$\boldsymbol{T'_{prog}[T_3,3]}$	$t_{3,3}$	Poligonal cerrada	Patrón de repetición	constante	$\alpha > 90º$ horario
T_4	$\boldsymbol{T'_{prog}[T_4,1]}$	$t_{4,1}$	Poligonal cerrada	Patrón de repetición	constante	$\alpha < 90º$ $\alpha > 90º$ antihorario
	$\boldsymbol{T'_{prog}[T_4,2]}$	$t_{4,2}$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario

5.3.2 $\boldsymbol{T_{prog}[T_{SF}]}$

Dentro de las características o descripción del tipo de tarea $\boldsymbol{T_{prog}[T_{SF}]}$, se encuentra una referencia clara y directa al tipo de tarea $\boldsymbol{T_{SF}}$ en sí mismo. Es decir,

$\boldsymbol{T_{prog}[T_{SF}]}$ hace mención explícita de $\boldsymbol{T_{SF}}$ en algún aspecto relacionado con sus propias características o funciones. Esta cualidad se extiende a las técnicas utilizadas en estos tipos de tareas, donde se puede identificar una relación o similitud en la forma en que se abordan o ejecutan.

Sea $\boldsymbol{\tau}$ la técnica de $\boldsymbol{T_{SF}}$ y $\boldsymbol{\tau_{prog}}$ la técnica de $\boldsymbol{T_{prog}[T_{SF}]}$**.** Si $\boldsymbol{T^*}$ es un ingrediente de la técnica $\boldsymbol{\tau}$ entonces $\boldsymbol{T^*}$ también podría utilizarse en la técnica $\boldsymbol{\tau_{prog}}$**.** Sin embargo, cuando utilizas el ingrediente $\boldsymbol{T^*}$ en la técnica $\boldsymbol{\tau_{prog}}$, es necesario combinarlo con otros ingredientes que son específicos de $\boldsymbol{\tau_{prog}}$ y que no están incluidos en $\boldsymbol{\tau}$ .En otras palabras, aunque $\boldsymbol{T^*}$ pueda ser compartido entre $\boldsymbol{\tau}$ y $\boldsymbol{\tau_{prog}}$, la técnica $\boldsymbol{\tau_{prog}}$ tiene requisitos adicionales (otros ingredientes de $\boldsymbol{\tau_{prog} \setminus \tau}$) que deben ser considerados y utilizados junto con $\boldsymbol{T^*}$ para esa técnica específica.

A continuación, se describen los ingredientes técnicos del plano praxeológico de la programación. Entre ellos se encuentra una variable "x", cuyo significado varía según las características específicas del proyecto. Por ejemplo, "x" puede representar el tamaño de un lápiz, lo que asume valores como 1, 2, etc. Además, puede utilizarse como un valor para el bloque de movimiento "mover" o "girar", así como para representar el ángulo de orientación o el nombre de un bloque personalizado. En este sentido, "x" podría corresponder a términos como lado o longitud si se refiere a una variable o triángulo o rombo si se relaciona con un bloque personalizado.

En el plano praxeológico de la programación, se identifican los siguientes ingredientes técnicos:

- ✓ ${T^*}_{prog}[comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ ${T^*}_{prog}[borrar\ todo]$: Limpiar la ventana de ejecución
- ✓ ${T^*}_{prog}[bajar\ lápiz]$: Activar el lápiz para empezar a dibujar
- ✓ ${T^*}_{prog}[subir\ lápiz]$: Levantar el lápiz para dejar de dibujar
- ✓ ${T^*}_{prog}[tamaño\ lápiz\ ]$: Fijar el tamaño del lápiz en "x"
- ✓ ${T^*}_{prog}[esconder]$: Elegir el bloque de apariencia "esconder"

- ✓ $T^*_{prog}[coordenadas]$: Posicionar el sprite en las coordenadas (x, y).
- ✓ $T^*_{prog}[orientación]$: Orientar hacia la derecha o izquierda “x” grados el cursor
- ✓ $T^*_{prog}[crear\ bloque]$: Crear un bloque personalizado “nombre del tipo de figura”
- ✓ $T^*_{prog}[usar\ bloque\ ]$: Usar el bloque personalizado "x”
- ✓ $T^*_{prog}[crear\ variable]$: Crear la variable “$x$”
- ✓ $T^*_{prog}[fijar\ variable]$: Dar a la variable “$x$” el valor de “$k$”.
- ✓ $T^*_{prog}[cambiar\ valor\ de\ la\ variable]$: Cambiar el valor de la variable “x” en “j”
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ variable]$: Insertar la variable “x” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor “x” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento “girar”
- ✓ $T^*_{prog}[girar\ n]$: Insertar “n” dentro del bloque “girar” en sentido antihorario u horario
- ✓ $T^*_{prog}[repetir]$: Usar el bloque de control “repetir” en un programa Scratch
- ✓ $T^*_{prog}[Repetir\ n\ veces]$: : Insertar el valor de "n" dentro del bloque de control “repetir”.
- ✓ $T^*_{prog}[Repetir\ hasta\ que\ ]$: Usar el bloque de control “repetir hasta que”
- ✓ $T^*_{prog}[Repetir\ hasta\ que$ "operador" $veces\ ]$: Insertar el operador “x” dentro del bloque de control “repetir hasta que”
- ✓ $T^*_{prog}[verificar]$: Verificar el programa desarrollado en Scratch.

La combinación de estos ingredientes conduce a diversas técnicas y programas diferentes en Scratch. Los ingredientes T_{mover} , T_{girar} $T_{Repetir}$, $T_{mover\ longitud}$, $T_{fijar\ longitud}$, $T_{Repetir\ n\ veces}$ y $T_{comenzar}$ se consideran tipos de tareas elementales. Por lo tanto, no se profundizará en sus técnicas específicas. En Scratch, $T_{comenzar}$ se realiza al hacer clic en la bandera verde en la ventana de programación o al hacer doble clic en el programa. Por otro lado, $T_{Verificar}$ implica comprobar que, tras la ejecución del programa, se hayan cumplido todas las restricciones establecidas para la tarea propuesta que el programa debe realizar (Crisci, 2020).

En el contexto de la presente investigación, se ha identificado ejercicios que emplean Scratch como herramienta para modelar tareas que requieren la creación de algoritmos. Estos ejercicios no solo movilizan conocimientos matemáticos, sino que también integran habilidades de programación y pensamiento computacional, lo que genera así praxeologías mixtas.

Durante el desarrollo de la investigación, se ha identificado que el concepto de unidad de patrón (o repetición) desempeña un papel fundamental tanto en la programación como en las matemáticas, especialmente al trabajar con patrones o secuencias repetitivas. Dependiendo del contexto, la unidad de patrón puede entenderse de diferentes maneras:

- **En programación**. - En el entorno de Scratch, una unidad de patrón se refiere a un bloque de código o una secuencia de acciones que se repiten o se utilizan como base para crear un patrón más complejo. Por ejemplo, en la figura 32 el uso del bloque de control "repetir" para ejecutar una figura y el bloque de movimiento “mover” varias veces puede considerarse una unidad de patrón.

  Ejemplo de una unidad de patrón en Scratch.

  Figura 32

- **En matemáticas**. - En este ámbito, una unidad de patrón es el componente básico que forma parte de un patrón más grande. Por ejemplo, al trabajar con patrones geométricos (como una roseta), una unidad de patrón puede ser el diseño

repetitivo más pequeño que se replica para generar el patrón completo. Esto puede incluir figuras geométricas, como triángulos o cuadrados, que se repiten en posiciones o ángulos específicos para crear un diseño más complejo.

En ambos casos, la **unidad de patrón** representa un bloque fundamental que, al repetirse o modificarse de manera controlada, contribuye a la formación de una estructura más compleja o de un patrón repetitivo que se expande a lo largo de una secuencia o figura.

A continuación, se presentan los ingredientes técnicos que se encuentran en esta praxeología:

- ✓ ${T^*}_m[unid.\,de\ patrón]$: identificar la unidad de patrón en el programa de Scratch
- ✓ ${T^*}_m[analizar\ bloque\ ]$: analizar el bloque personalizado “x” u otro bloque de Scratch
- ✓ ${T^*}_m[ident.\,patrón\ recurrente]$: Identifica la unidad de cambio que ocurre de una figura a otra en Scratch
- ✓ ${T^*}_m[analizar\ el\ programa]$: examinar la secuencia y la funcionalidad de los bloques.
- ✓ ${T^*}_m[interpretar\ el\ programa]$: comprender cómo funcionan sus diferentes componentes y cómo interactúan entre sí para lograr un objetivo específico.
- ✓ ${T^*}_m[interpretar\ el\ bloque]$: entender cómo funciona un bloque específico dentro de un programa en Scratch. Esto implica comprender qué instrucciones o acciones realiza el bloque cuando se ejecuta en el contexto del script.
- ✓ ${T^*}_m[Dibujo]$: interpreta geométricamente el bloque personalizado “x” o el programa.

A continuación, se desarrollará los elementos praxeológico encontrados en los artículos de Alayrangues et al. (2019) y de Ye et al. (2024).

CAPÍTULO VI

ANÁLISIS DE LA ORGANIZACIÓN MATEMÁTICA MIXTA

En este capítulo se analizan las tareas extraídas de dos artículos clave. El primero, de Alayrangues et al. (2019), reflexiona sobre cómo los ejercicios del examen de matemáticas de 2017 del *brevet* (examen nacional de Francia al final de la educación secundaria) podrían integrarse de manera más efectiva con el enfoque de la programación y la informática, para hacer su aplicación en el aprendizaje de las matemáticas más relevante y enriquecedora. El segundo artículo, de Ye et al. (2024), se centra en cómo los docentes de matemáticas en servicio (aquellos que ya están trabajando en el aula) desarrollan conceptos matemáticos y habilidades de programación a través de acciones creativas en un entorno de programación basado en bloques como Scratch. Mediante entrevistas y análisis, el estudio identifica dos prácticas creativas clave: prueba e iteración, y reutilización y remix. Estas prácticas ofrecen nuevas formas de hacer y entender las matemáticas desde una perspectiva computacional, sugiriendo que pueden enriquecer la enseñanza y mejorar la formación y desarrollo profesional de los docentes, generando nuevas oportunidades para el aprendizaje.

A continuación, se presentará una descripción y análisis de la Organización Matemática Mixta relacionada con el concepto de "Secuencia Figural". Además, se discutirán los resultados obtenidos a partir de este análisis.

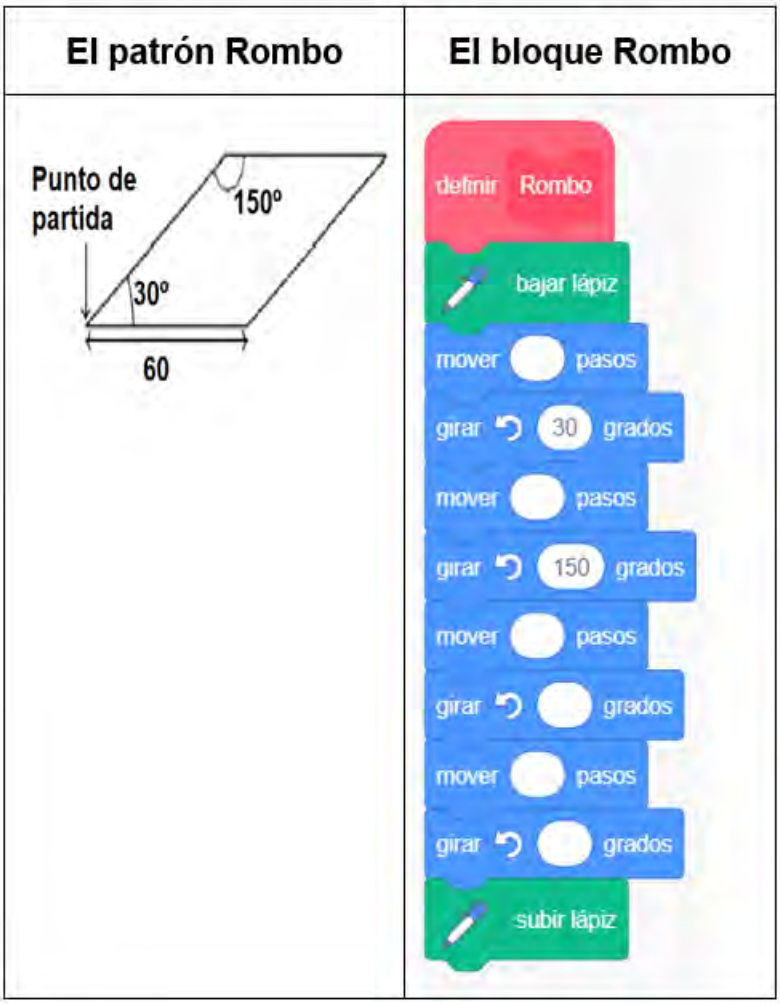
6.1 Análisis de los tipos de tareas

En esta sección, se describen y analizan las tareas correspondientes a los tipos de tareas $\boldsymbol{T_1}$, $\boldsymbol{T_2}$, $\boldsymbol{T_3}$ y $\boldsymbol{T_4}$ identificados a través del modelo praxeológico mixto presentado en la sección anterior.

Figura 33

Alayrangues, et al. Ejercicio 4- (p.10)

1. Se desea trazar el patrón a continuación en forma de rombo. Completar en el anexo 1, el script del bloque Rombo para obtener este patrón.



Fuente. Tomado y traducido de Alayrangues, et al., 2019, pag.10

Figura 34
Alayrangues, et al. Ejercicio 4- (p.11)

Fuente. Tomado y traducido de Alayrangues, et al., 2019, pag.11

El ejercicio 4 se centra en completar un bloque personalizado y la programación de un script para trazar un “patrón” en forma de rombo en el entorno de programación visual Scratch. A continuación, se presenta un análisis detallado de cada parte del ejercicio:

Este ejercicio consta de dos preguntas. La primera pregunta incluye una tarea $t_{1,1,1}$ del tipo de tarea $\boldsymbol{T_1}$, en la cual se presenta la figura de un rombo y se solicita completar un bloque personalizado para obtener dicha figura. La segunda pregunta presenta una tarea $t_{4,1}$ tipo de tarea $\boldsymbol{T_4}$ en la cual se pide completar el script para la producción de una figura

En el bloque personalizado "Rombo" se tiene los siguientes ingredientes:

- ✓ $T^*_{prog}[Crear\ bloque]$: Crear un bloque personalizado "Rombo"
- ✓ $T^*_{prog}[bajar\ lápiz]$: Se activa el lápiz para empezar a dibujar
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento "mover"
- ✓ $T^*_{prog}[\ mover\ valor]$: Insertar el valor "60" dentro del bloque de movimiento "mover"
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento "girar" en sentido antihorario
- ✓ $T^*_{prog}[\ girar\ n]$: Insertar "30" dentro del bloque "girar" en sentido antihorario
- ✓ $T^*_{prog}[mover\ ]$: Elegir el bloque de movimiento "mover"
- ✓ $T^*_{prog}[\ mover\ valor]$: Insertar el valor "60" dentro del bloque de movimiento "mover"
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento "girar" en sentido antihorario
- ✓ $T^*_{prog}[\ girar\ n]$: Insertar 150 dentro del bloque "girar" en sentido antihorario
- ✓ $T^*_{prog}[\ mover]$: Elegir el bloque de movimiento "mover"
- ✓ $T^*_{prog}[mover\ valor\ ]$: Insertar el valor "60" dentro del bloque de movimiento "mover"
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento "girar" en sentido antihorario
- ✓ $T^*_{prog}[girar\ n\ ]$: Insertar 30º dentro del bloque "girar" en sentido antihorario
- ✓ $T^*_{prog}[mover\ ]$: Elegir el bloque de movimiento "mover"
- ✓ $T^*_{prog}[mover\ valor\ ]$: Insertar el valor 60 dentro del bloque de movimiento "mover"
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento "girar" en sentido antihorario
- ✓ $T^*_{prog}[girar\ n\ ]$: Insertar 150 dentro del bloque "girar" en sentido antihorario
- ✓ $T^*_{prog}[\ subir\ lápiz]$: levantar el lápiz para dejar de dibujar

En el Script: se observa los siguientes ingredientes:

- ✓ $T^*_{prog}[\ comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{prog}[\ borrar\ todo]$: se limpia la ventana de ejecución
- ✓ $T^*_{prog}[coordenadas\ ]$: el cursor se posiciona en las coordenadas (-200, -100).
- ✓ $T^*_{prog}[orientación\ ]$: El cursor se orienta hacia la derecha 90”
- ✓ $T^*_{prog}[Repetir\ ]$: Usar el bloque de control “repetir”
- ✓ $T^*_{prog}[Repetir\ n\ veces\ ]$: Insertar el valor de "5" dentro del bloque de control “repetir”.
- ✓ $T^*_{prog}[Verificar\ ]$: Verificar el programa desarrollado en Scratch.

Pregunta 1: Se desea trazar el patrón a continuación en forma de rombo.
Completar en el anexo 1, el script del bloque Rombo para obtener este patrón.

Esta pregunta presenta una tarea $t_{1,1,1}$ del tipo de tarea $\boldsymbol{T_1}$

Tipo de tarea $\boldsymbol{T_1}$: Identificar la unidad de patrón presente en una secuencia figural.
Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_1]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_1}$

$\boldsymbol{GT_{prog}[T_1]}$: [Identificar la unidad de patrón presente en una secuencia figural, $\boldsymbol{V_{prog}}[\mathbf{tipo\ de\ figura}]$, $\boldsymbol{V_{prog}}[\mathbf{tipo\ de\ patrón}]$, $\boldsymbol{V_{prog}}[\ \mathbf{cantidad\ de\ pasos}]$,

$\boldsymbol{V_{prog}}[\ \mathbf{ángulo\ de\ giro}]$]

Sub tipo de tarea

- $\boldsymbol{T'_{prog}[T_1, 1]}$**:** Identificar la unidad de patrón presente en una figura poligonal cerrada que presenta un patrón de repetición. La cantidad de pasos es constante y el ángulo de giro es tanto menor como mayor de 90º, en sentido antihorario.
- **Técnica** $\boldsymbol{[T_1, 1], 1}$
  En esta descripción, se considerará el rombo como una secuencia de segmentos consecutivos.
  - ✓ $T^*_{obs\ repet}$: Observa la secuencia figural con la finalidad de buscar atributos para establecer relaciones.

Observa la figura con la finalidad de identificar las propiedades del rombo

- ✓ $T^*_{busca\ atributos}$: Busca atributos simples

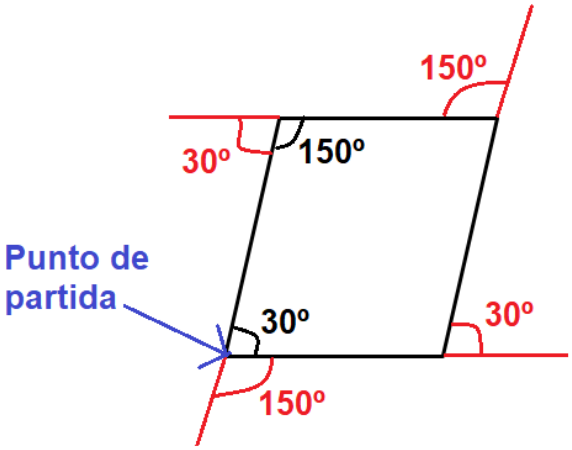
  El lado de la figura es 60 y sus ángulos interiores es 30º y 150º

- ✓ $T^*_{direcc.cambio}$: Determina como varían los ángulos exteriores de la figura (rombo).

  En la figura 35 se observan estos ángulos.

Figura 35

Ángulos exteriores del rombo de la tarea $\boldsymbol{t_{1,1,1}}$



- ✓ $T^*_{unid.\ de\ repetición}$: Identifica la unidad de repetición en el rombo

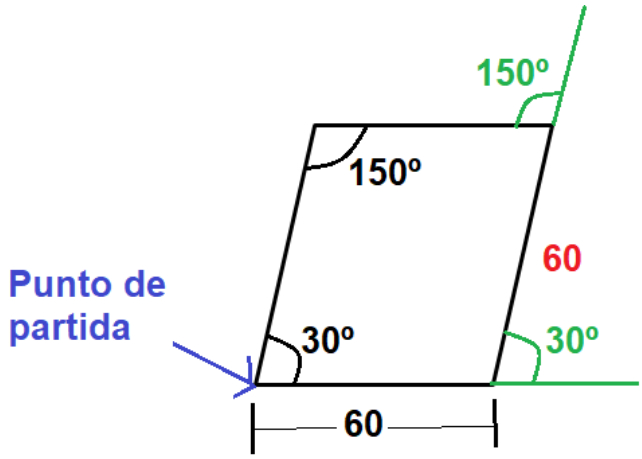
  La unidad de repetición tiene dos atributos: longitud del segmento y ángulo de rotación

  La unidad de repetición: 60 pasos; 30º en sentido antihorario; 60 pasos; 150º en sentido antihorario.

- ✓ T^*_{Dibujo}: Dibuja el rombo, destacando la unidad de repetición tal como se muestra en la Figura 36.

Figura 36

Identificación de la unidad de repetición del rombo de la tarea $\boldsymbol{t_{1,1,1}}$



✓ $T^*{}_m[unid.\,de\,patrón]$: identifica la unidad de patrón del rombo de la tarea $t_{1,1,1}$ en el programa de Scratch, tal como se muestra en la Figura 37.

Figura 37

Unidad de patrón del rombo en el bloque personalizado(tarea $\boldsymbol{t_{1,1,1}}$)

✓ $T^*{}_m[analizar\ bloque\ ]$: Analiza el bloque personalizado “Rombo”

Analiza si el patrón observado en la secuencia de segmentos consecutivos se ajusta a las reglas establecidas por la unidad de patrón identificada en programa Scratch. Para ello completa el script del bloque personalizado “Rombo” tal como se muestra en la figura 38

Figura 38

Bloque personalizado "Rombo" de la tarea $\boldsymbol{t_{1,1,1}}$

✓ $T^*_{prog}[comenzar]$: Comenzar el programa desarrollado en Scratch. Para ello se inserta en la ventana de programación el bloque de evento "al hacer clic en " seguido del bloque lápiz y del bloque personalizado" Rombo", tal como se muestra en la figura 39.

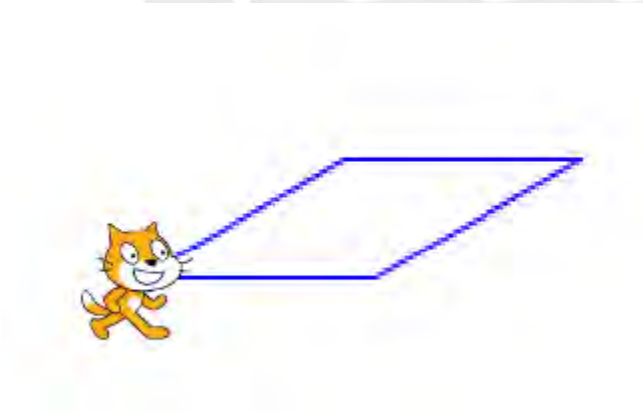
Figura 39

Script del "Rombo" de la tarea $\boldsymbol{t_{1,1,1}}$

- ✓ $T^*_{prog}[Verificar]$: Verifica el programa desarrollado en Scratch observando en la ventana de ejecución si la figura generada coincide con lo especificado en la tarea $t_{1,1,1}$. En la figura 40 muestra la figura creada por Scratch.

Figura 40

El rombo de la tarea $\boldsymbol{t_{1,1,1}}$ *creado en el programa Scratch*



Pregunta 2: Se desea realizar la figura a continuación, construida a partir del bloque Rombo completado en la pregunta 1

Esta pregunta presenta una tarea $t_{4,1}$ del tipo de tarea $\boldsymbol{T_4}$

Tipo de tarea $\boldsymbol{T_4}$: Extiende patrones al menos una unidad de patrón

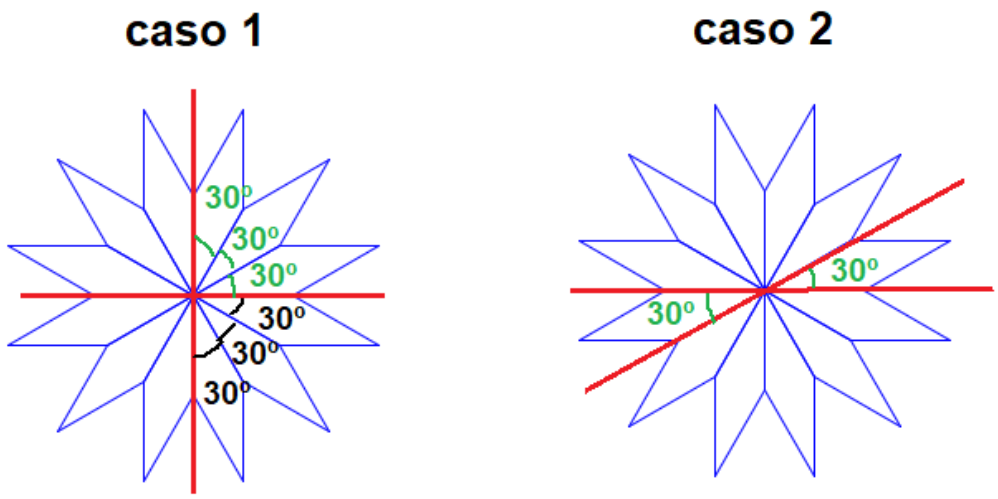
Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_4]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_4}$,

$\boldsymbol{GT_{prog}[T_4]}$: [Extiende patrones, al menos una unidad de patrón, $\boldsymbol{V_{prog}[\text{tipo de figura}]}$, $\boldsymbol{V_{prog}[\text{tipo de patrón}]}$, $\boldsymbol{V_{prog}[\text{cantidad de pasos}]}$, $\boldsymbol{V_{prog}[\text{ángulo de giro}]}$]

Sub tipo de tarea

- $\boldsymbol{T'_{prog}[T_4, 1]}$**:** Extiende patrones, al menos una unidad de patrón presente en figura poligonal cerrada que presenta un patrón de repetición. La cantidad de pasos es constante y el ángulo de giro es tanto menor como mayor de 90º, en sentido antihorario.
- **Técnica:** $\boldsymbol{[T_4], 1}$
  - ✓ $T^*_{obs\ repet}$: Observa la roseta con el objetivo de identificar atributos que permitan establecer relaciones. Tal como se muestra en la figura 41, se trazan líneas sobre la figura para analizar una parte específica de la roseta, con el fin de identificar características que faciliten la determinación de los ángulos de rotación.

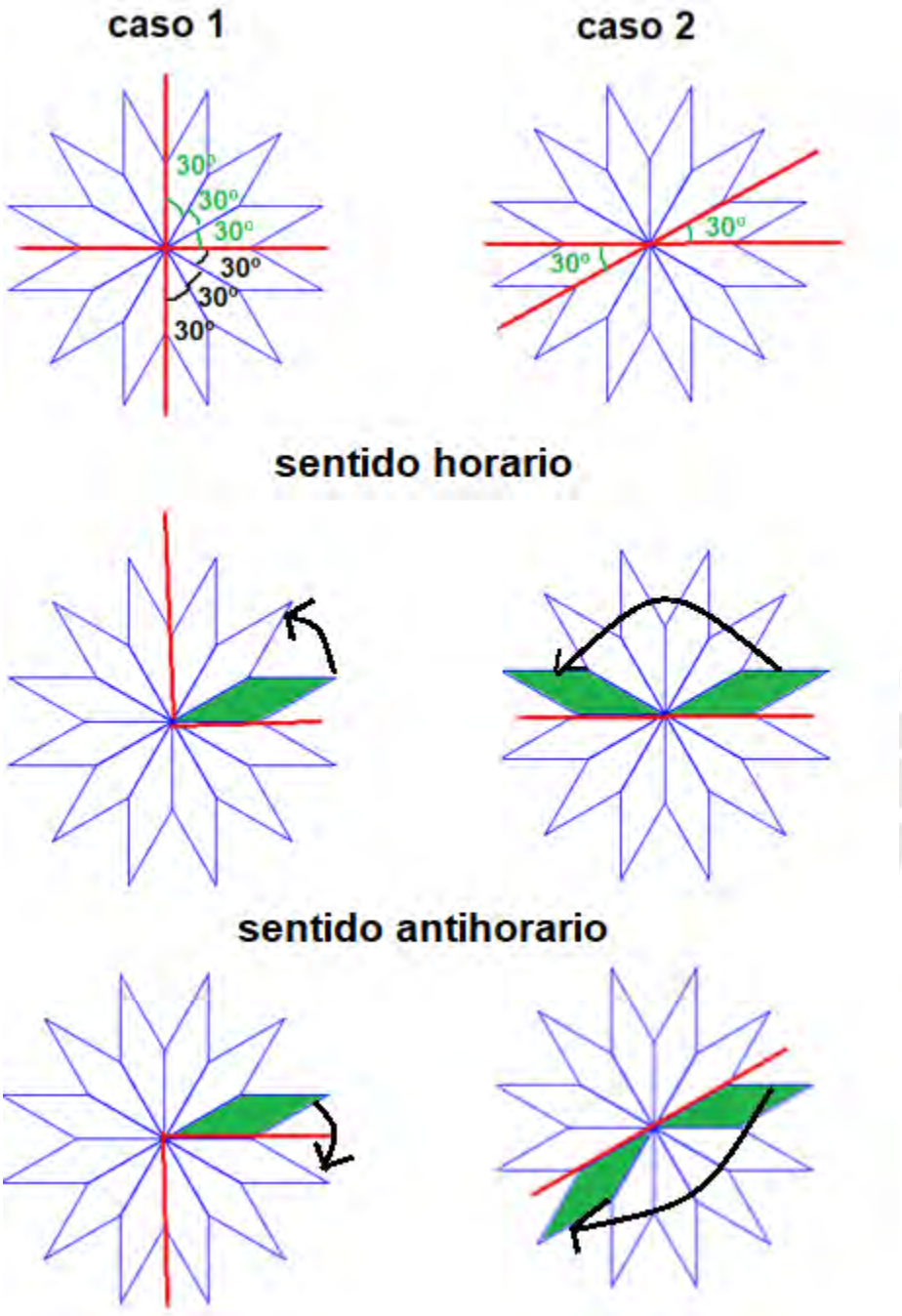
Figura 41
Análisis de la roseta en la tarea $\boldsymbol{t_{4,1}}$



- ✓ $T^*_{busca\ atributos}$: Busca atributos simples, es decir, identifica elementos que se repiten de manera consistente. Por ejemplo, en la figura 42 se observa que, en el caso 1, donde las líneas trazadas forman una cruz y dividen la figura en cuatro cuadrantes, el ángulo de rotación puede ser en sentido horario o antihorario. En el caso 2, donde las líneas son diagonales, el ángulo de rotación también puede ser en ambos sentidos, horario o antihorario.

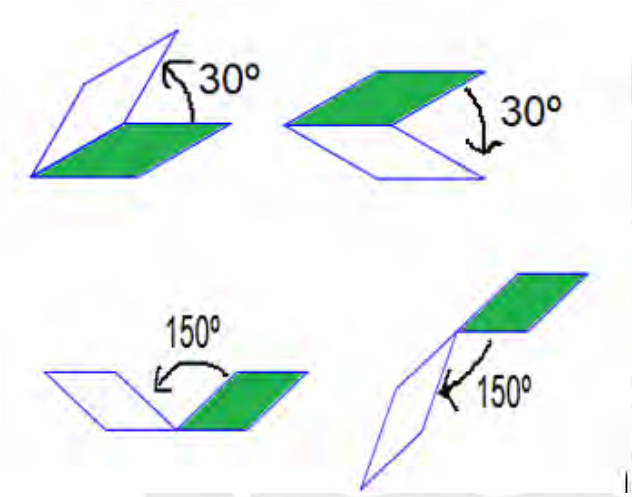
Figura 42

Atributos de la roseta en la tarea $t_{4,1}$



- ✓ $T^*_{direcc.cambio}$: Determina como cambia el rombo (pétalo de la roseta), como se muestra en la figura 43 el rombo puede rotarse de diversas maneras para formar la roseta, identificándose cuatro rotaciones posibles: 30º en sentido horario, 30º en sentido antihorario, 150º en sentido horario y 150º en sentido antihorario, siendo estos ángulos los ángulos interiores del rombo.

Figura 43
Rotaciones posibles del rombo para Formar la roseta en la tarea $\boldsymbol{t_{4,1}}$.

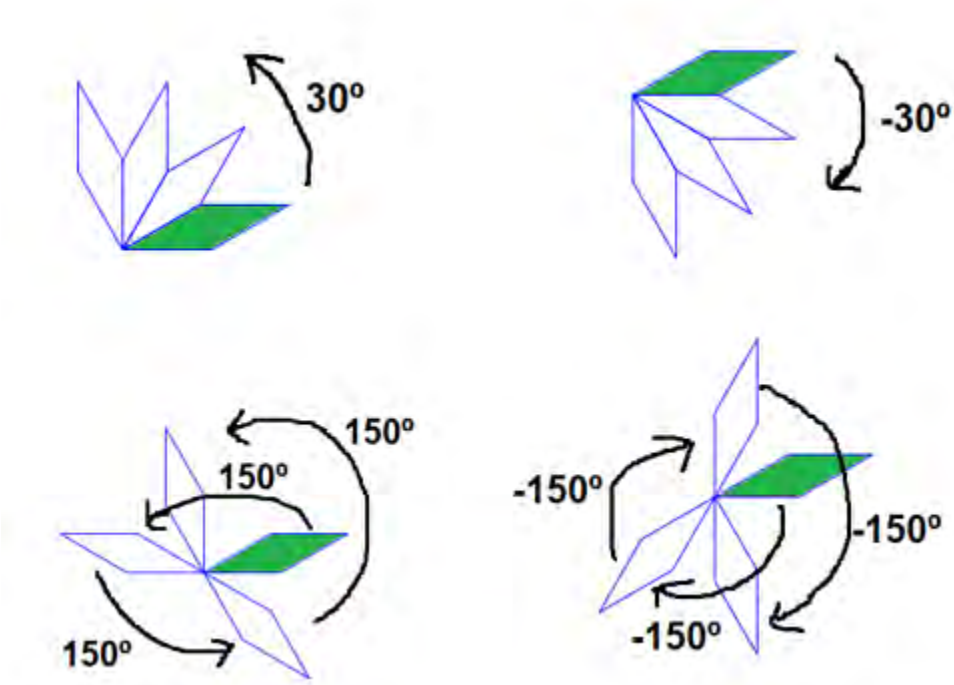


- ✓ $T^*_{unid.\ de\ repetición}$: Identifica la unidad de repetición de la roseta, la cual posee dos atributos.
  - o Ángulo de rotación de 30º en sentido horario.
  - o Ángulo de rotación de 30º en sentido antihorario.
  - o Ángulo de rotación de 150º en sentido horario.
  - o Ángulo de rotación de 150º en sentido antihorario.
- ✓ T^*_{Dibujo}: Dibuja la roseta

  El rombo de color verde es el rombo inicial, que rotará para generar la figura solicitada. Se pueden identificar cuatro giros posibles: 30º, -30º, 150º y -150º en Scratch, utilizando el bloque de movimiento 'girar' en sentido antihorario, tal como se muestra en la figura 44.

Figura 44

Giros posibles en Scratch del rombo para formar la roseta en la tarea $t_{4,1}$.

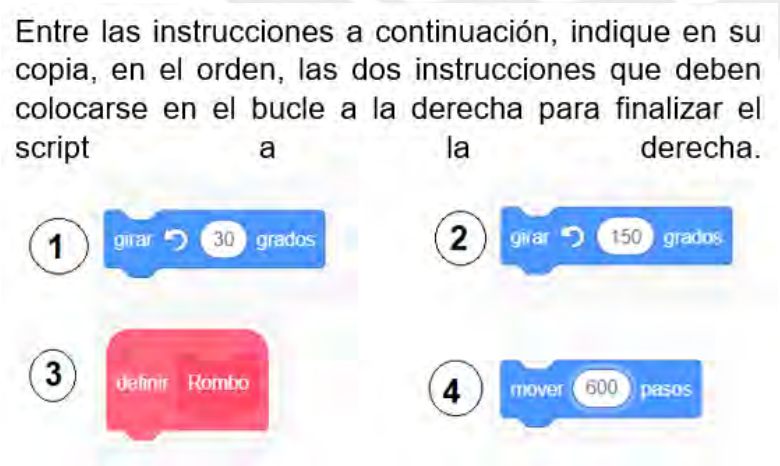


✓ $T^*{}_m[analizar\ bloque\ ]$: analiza el bloque

Analiza si el patrón observado en la secuencia de rombos consecutivos se ajusta a las reglas establecidas por el ejercicio. Ver figura 45.

Figura 45

Instrucciones de la tarea $t_{4,1}$



✓ ${T^*}_m[unid.\,de\,patrón]$: Identifica la unidad de patrón en la roseta de la tarea $t_{4,1}$, en el programa Scratch, como se muestra en la Figura 46.

Figura 46
Unidad de patrón de la roseta en Scratch (tarea $\boldsymbol{t_{4,1}}$)

${T^*}_{prog}[comenzar]$: Comenzar el programa en Scratch. Para ello se requiere insertar en el script el bloque personalizado "Rombo", seguido del bloque de movimiento "Girar" en sentido antihorario. En el **primer caso**, el bloque "Girar" debe configurarse para girar 30º, lo que corresponde a la instrucción número 1. En el **segundo caso**, el bloque "Girar" se ajusta para girar 150º, lo que corresponde a la instrucción número 2. En ambos casos, el bloque "Rombo" (que corresponde a la instrucción numero 3) se mantiene como parte fundamental del script, tal como se ilustra en la figura 47.

Figura 47

Script de la roseta correspondiente a la tarea $\boldsymbol{t_{4,1}}$

- ✓ $T^*_{prog}[verificar]$: Verifica el programa desarrollado en Scratch mediante la ventana de ejecución, donde se observa que, tanto en el primer caso como en el segundo caso , se genera la misma figura, es decir, la roseta de la tarea $\boldsymbol{t_{4,1}}$. En la figura 48 se muestra la figura generada por Scratch para ambos casos.

Figura 48

La roseta de la tarea $t_{4,1}$ creado en el programa Scratch

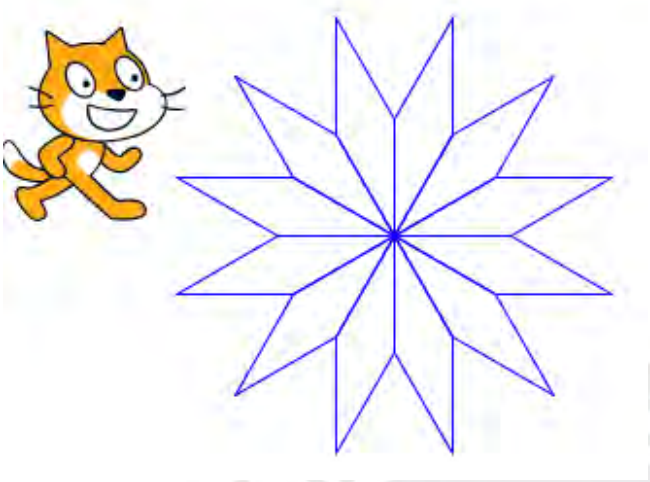
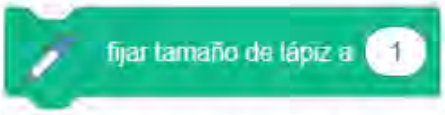


Figura 49
Alayrangues, et al. Ejercicio 4- (p.12)

1. Para realizar la figura anterior, se definió un patrón en forma de rombo y se utilizó uno de los dos programas A y B a continuación. Determina cuál de ellos y señala con un dibujo a mano alzada el resultado que se obtendría con el otro programa.

2. ¿Cuánto mide el espacio entre dos patrones sucesivos?
3. Se desea realizar la figura a continuación:

Para ello, se plantea insertar la instrucción 

en el programa utilizado en la pregunta 1. ¿Dónde se debe insertar esta instrucción?

Fuente. Tomado y traducido de Alayrangues, et al.,2019, p.12

El ejercicio consta de tres preguntas. La primera pregunta plantea una tarea $t_{1,1,2}$ del tipo de tarea $\boldsymbol{T_1}$, que consiste en identificar cuál de los dos programas (A o B) se utiliza para crear una secuencia figural en forma de rombo, así como imaginar y dibujar el resultado que se obtendría si se usara el otro programa. La segunda pregunta presenta una tarea $t_{1,1,3}$ del tipo de tarea $\boldsymbol{T_1}$, y se centra en determinar la distancia o separación entre dos patrones sucesivos en forma de rombo cuando se repiten en una figura. Por último, la tercera pregunta aborda una tarea $t_{3,1}$ del tipo de tarea $\boldsymbol{T_3}$, que se enfoca en la modificación del programa para realizar una secuencia figural específica.

En el bloque personalizado "Rombo" se tiene los siguientes ingredientes:

- ✓ $T^*_{prog}[Crear\ bloque]$: Crear un bloque personalizado "Rombo"
- ✓ $T^*_{prog}[bajar\ lápiz]$: Se activa el lápiz para empezar a dibujar
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento "mover"
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor "40" dentro del bloque de movimiento "mover"
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento "girar" en sentido antihorario
- ✓ $T^*_{prog}[girar\ n]$: Insertar 45º dentro del bloque "girar" en sentido antihorario

- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor “40” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n]$: Insertar 135º dentro del bloque “girar” en sentido antihorario
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor “40” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n]$: Insertar 45º dentro del bloque “girar” en sentido antihorario
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor 40 dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar]$Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n]$: Insertar 150 dentro del bloque “girar” en sentido antihorario

En el Script: se observa los siguientes ingredientes:

Programa A

- ✓ $T^*_{prog}[\ comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{prog}[esconder]$: Elegir el bloque de apariencia “esconder”
- ✓ $T^*_{prog}[\ borrar\ todo]$:se limpia la ventana de ejecución
- ✓ $T^*_{prog}[tamaño\ lapiz]$: fijar el tamaño del lápiz en 1
- ✓ $T^*_{prog}[coordenadas\ ]$: el cursor se posiciona en las coordenadas (-230, 0).
- ✓ $T^*_{prog}[orientación\ ]$: El cursor se orienta hacia la derecha 90”
- ✓ $T^*_{prog}[Bloque\ personalizado$: usar el bloque personalizado “Rombo”
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor]$: Insertar el valor “55” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[Repetir]$: Usar el bloque de repetición
- ✓ $T^*_{prog}[Repetir\ n\ veces]$: Insertar el valor de "8" dentro del bloque de repetición.

- ✓ $T^*_{prog}[verificar]$: Verificar el programa desarrollado en Scratch.

Programa B

- ✓ $T^*_{prog}[\ comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{prog}[esconder]$:Elegir el bloque de apariencia “esconder”
- ✓ $T^*_{prog}[\ borrar\ todo]$:se limpia la ventana de ejecución
- ✓ $T^*_{prog}[tamaño\ lápiz\ ]$: fijar el tamaño del lápiz en 1
- ✓ $T^*_{prog}[coordenadas\ ]$: el cursor se posiciona en las coordenadas (0, 0).
- ✓ $T^*_{prog}[orientación\ ]$: El cursor se orienta hacia la derecha 90”
- ✓ $T^*_{prog}[Bloque\ personalizado]$: usar el bloque personalizado “Rombo”
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n]$: Insertar “45” dentro del bloque “girar” en sentido antihorario
- ✓ $T^*_{prog}[Repetir\ ]$: Usar el bloque de repetición
- ✓ $T^*_{prog}[Repetir\ n\ veces\ ]$: Insertar el valor de "8" dentro del bloque de repetición.
- ✓ $T^*_{prog}[verificar]$: Verificar el programa desarrollado en Scratch.

Pregunta 1: Para realizar la figura anterior, se definió un patrón en forma de rombo y se utilizó uno de los dos programas A y B a continuación. Determina cuál de ellos y señala con un dibujo a mano alzada el resultado que se obtendría con el otro programa.

Esta pregunta presenta una tarea $\boldsymbol{t_{1,1,2}}$ del tipo de tarea $\boldsymbol{T_1}$

Tipo de tarea $\boldsymbol{T_1}$: Identificar la unidad de patrón presente en una secuencia figural.
Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_1]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_1}$

$\boldsymbol{GT_{prog}[T_1]}$: [Identificar la unidad de patrón presente en una secuencia figural, $\boldsymbol{V_{prog}[\text{tipo de figura}]}$, $\boldsymbol{V_{prog}[\text{tipo de patrón}]}$, $\boldsymbol{V_{prog}[\text{ cantidad de pasos}]}$, $\boldsymbol{V_{prog}[\text{ ángulo de giro}]}$]

Subtipo de tarea

- $\boldsymbol{T'_{prog}[T_1, 1]}$**:** Identificar la unidad de patrón presente en una secuencia figural compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón de

repetición, la cantidad de pasos es constante y el ángulo de giro es tanto menor como mayor de 90º, en sentido antihorario.

- **Técnica $[T_1, 1], 2$**
  - ✓ $T^*{}_{obs\ repet}$: observa la secuencia figural con la finalidad de buscar atributos para establecer relaciones

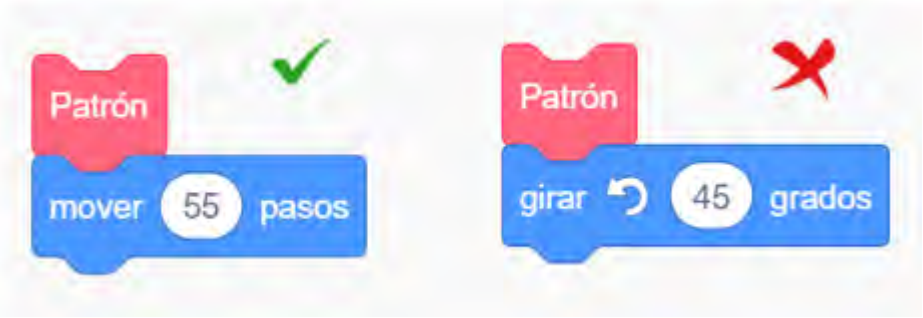
    La secuencia figural está formada por rombos
  - ✓ $T^*{}_{busca\ atributos}$: busca atributos simples

    Los rombos de la secuencia figural son del mismo tamaño y forma, y están equidistantes entre sí.
  - ✓ $T^*{}_{unid.patrón\ de\ repetición.}$: identifica la unidad de repetición

    La unidad de repetición tiene dos atributos: rombo del mismo tamaño y forma, separados por una distancia uniforme.
  - ✓ $T^*{}_m[unid.\ de\ patrón]$: identificamos la unidad de patrón en el programa A.

    Después de reconocer la unidad de repetición, descartamos la unidad de patrón del programa B, tal como se muestra en la figura 50.

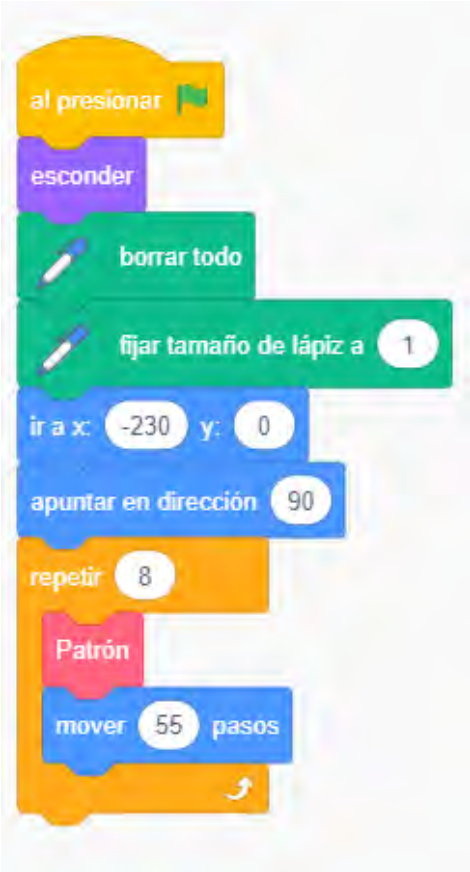
    Figura 50
    Unidad de patrón del programa A (tarea $\mathbf{t_{1,1,2}}$)



  - ✓ $T^*{}_{prog}[comenzar]$: se presenta a continuación el inicio del programa A, el cual ha sido desarrollado utilizando Scratch. En la figura 51 se muestra el código completo del programa A implementado en esta plataforma.

Figura 51

Script de la secuencia de rombos correspondiente a la tarea $\boldsymbol{t_{1,1,2}}$



- ✓ $T^*_{Verificar}$: al revisar en la ventana de ejecución de Scratch, confirmamos que la unidad de patrón utilizada en el programa A es la encargada de generar la secuencia de rombos, como se muestra en la Figura 52.

Figura 52

Secuencia de rombos de la tarea $\boldsymbol{t_{1,1,2}}$ *creado en Scratch.*

Pregunta 2: ¿Cuánto mide el espacio entre dos patrones sucesivos?

Esta pregunta presenta una tarea $\boldsymbol{t_{1,1,3}}$ del tipo de tarea $\boldsymbol{T_1}$

Tipo de tarea T_1: Identificar la unidad de patrón presente en una secuencia figural

Tipo de tarea en Scratch: $T_{prog}[T_1]$: Desarrollar un programa en Scratch que ejecute T_1

$GT_{prog}[T_1]$: [Identificar la unidad de patrón presente en una secuencia figural, , $V_{prog}[\textbf{tipo de figura}]$, $V_{prog}[\textbf{tipo de patrón}]$, $V_{prog}[\textbf{cantidad de pasos}]$, $V_{prog}[\textbf{ángulo de giro}]$]

Subtipo de tarea

- $T'_{prog}[T_1, 1]$**:** Identificar la unidad de patrón presente en una secuencia figural, compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón de repetición, la cantidad de pasos es constante y el ángulo de giro es tanto menor como mayor de 90º, en sentido antihorario.
- **Técnica** $[T_1, 1], 3$
  - ✓ $T^*{}_m[analizar\ bloque\ ]$: analiza el bloque personalizado “Patrón”

bajar lápiz: Esto indica que el "lápiz" de Scratch se coloca en modo de escritura, lo que significa que comenzará a dibujar cuando el sprite (gato) se mueva.

mover 40 pasos: El sprite se moverá hacia adelante (por el signo positivo) 40 unidades (píxeles), creando la primera línea del rombo.

girar 45 grados: Después de avanzar, el sprite gira 45 grados a la derecha. Este giro establece la dirección para la siguiente línea del dibujo.

mover 40 pasos: Nuevamente, el sprite avanza 40 unidades, creando otra línea del rombo.

girar 135 grados: El sprite gira 135 grados a la derecha, orientándose hacia la dirección adecuada para continuar el rombo.

mover 40 pasos: Se repite el movimiento hacia adelante de 40 unidades para dibujar la siguiente línea.

girar 45 grados: Otra vez, el sprite gira 45 grados, preparando el ángulo para la siguiente línea.

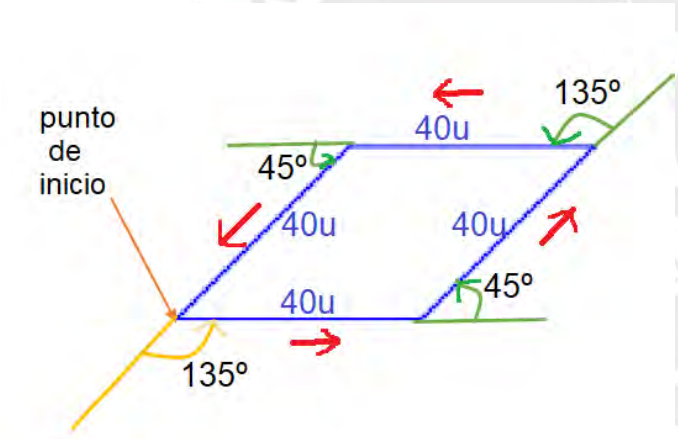
mover 40 pasos: El sprite avanza nuevamente 40 unidades, completando el lado del rombo.

girar 135 grados: Finalmente, el sprite realiza un giro de 135 grados para cerrar la figura.

Subir lápiz: Esto indica que el lápiz se levanta, lo que significa que el sprite dejará de dibujar al moverse, finalizando el dibujo.

✓ $T^*{}_m[Dibujo]$: realiza una interpretación geométrica del bloque personalizado denominado 'Rombo', como se observa en la Figura 53.

Figura 53
Interpretación geométrica del bloque personalizado 'Rombo de la tarea $\boldsymbol{t_{1,1,3}}$



✓ $T^*{}_m[analizar\ el\ programa]$: examinar la secuencia y la funcionalidad de los bloques.

- **Al presionar la bandera verde:** Este es el bloque que inicia el programa. Cuando el usuario hace clic en la bandera verde, el resto de las instrucciones se ejecutan.
- **Esconder**: Esta instrucción hace que el sprite (gato) se oculte durante el dibujo
- **Borrar todo**: Esta acción borra todo lo que se ha dibujado en la pantalla hasta ese momento.
- **Fijar tamaño de lápiz**: Esta instrucción establece el grosor del lápiz en 1. Esto significa que las líneas que dibuje el sprite serán finas.

- **Ir a x: -230 y: 0**: Aquí se establece la posición inicial del sprite en las coordenadas (x: -230, y: 0). Esto coloca el sprite en el lado izquierdo de la pantalla.
- **Apuntar en dirección 90**: Esta instrucción orienta el sprite a 90 grados, lo que significa que estará apuntando hacia la hacia la derecha. Esto es importante para la dirección del movimiento.
- **Repetir**: Esta instrucción indica que el bloque de instrucciones que sigue se repetirá 8 veces. Esto permite crear múltiples repeticiones de un patrón.
- **Bloque personalizado "Patrón"**: Este bloque contiene un conjunto de movimientos y giros que describen cómo dibujar el patrón (Rombo).
- **Mover 55 pasos**: Esta instrucción indica que el sprite se moverá hacia adelante 55 unidades (píxeles). Esto es parte del patrón que se está dibujando.

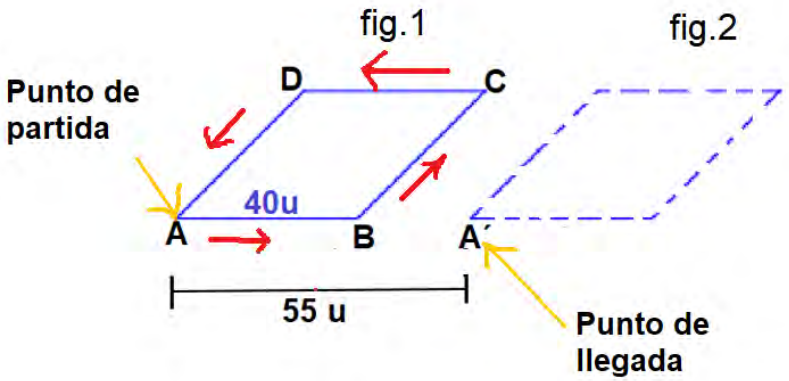
✓ $T^*{}_m[unid.\,de\,patrón]$: identifica la unidad de patrón en el programa de Scratch. Después de analizar el programa, el siguiente paso es reconocer la unidad de patrón, la cual se muestra en la Figura 54.

Figura 54
Unidad de patrón de la secuencia de rombos (tarea $\boldsymbol{t_{1,1,3}}$ *)*

$T^*{}_{Dibujo}$: Dibujaremos $fig.1$. El siguiente paso consiste en dibujar la primera figura de la secuencia el cual. Tras completar el primer patrón (fig. 1), el sprite se detiene y avanza 55 unidades antes de repetir el patrón nuevamente (fig.2), tal como se muestra en la figura 55.

Figura 55

$fig.1$. de la secuencia de rombos correspondiente a la tarea $t_{1,1,3}$



✓ Analizamos la figura

El lado del rombo mide 40 unidades (píxeles), es decir, $\overline{\overline{AB}}$ es 40 u, y la distancia entre los dos patrones es de 55 unidades, por lo que $\overline{\overline{AA'}} = 55$.

Definamos x como la distancia del espacio entre los patrones, es decir, $\overline{\overline{BA'}} = x$. Entonces, podemos establecer la relación:

$$\overline{\overline{AA'}} = \overline{\overline{AB}} + \overline{\overline{BA'}}$$

Sustituyendo los valores, tenemos: $55 = 40 + x$

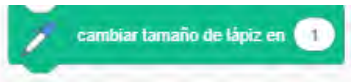
Resolviendo para x

$$55 - 40 = x$$

$$15 = x$$

Por lo tanto, el espacio entre dos patrones sucesivos mide 15 unidades (pixeles)

Pregunta 3: Se desea realizar la figura a continuación:

Para ello, se plantea insertar la instrucción 

en el programa utilizado en la pregunta 1. ¿Dónde se debe insertar esta instrucción?

Esta pregunta presenta una tarea $t_{3,1}$ del tipo de tarea $\boldsymbol{T_3}$

Tipo de tarea $\boldsymbol{T_3}$: Crear una secuencia a partir de una secuencia dada.

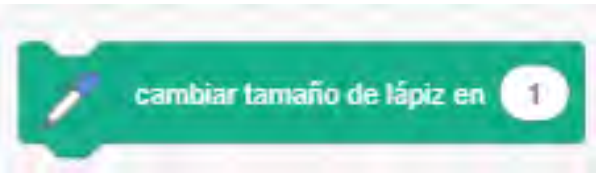
Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_3]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_3}$. Donde $\boldsymbol{T_3}$**:** Crear una secuencia partir de una secuencia dada.

$\boldsymbol{GT_{prog}[T_3]}$: [Crear una secuencia partir de una secuencia dada, $\boldsymbol{V_{prog}}$[**tipo de figura**], $\boldsymbol{V_{prog}}$[**tipo de patrón**], $\boldsymbol{V_{prog}}$[**cantidad de pasos**], $\boldsymbol{V_{prog}}$[**ángulo de giro**]]

Subtipo de tarea

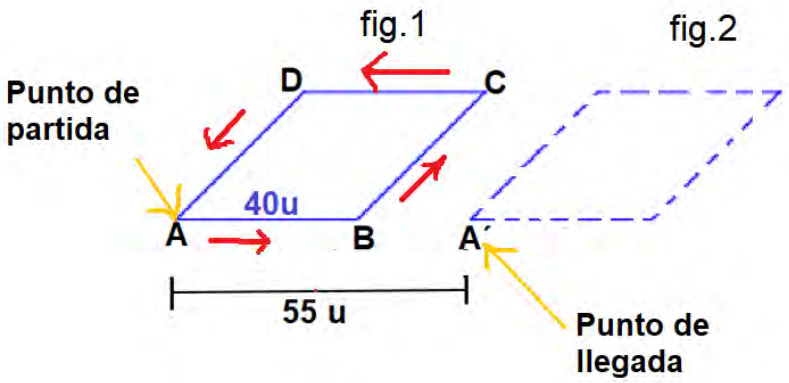
- $\boldsymbol{T'_{prog}[T_3, 1]}$**:** Crear una secuencia partir de una secuencia dada, compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón recurrente creciente, la cantidad de pasos es constante y el ángulo de giro es tanto menor como mayor de 90º, en sentido antihorario.
- **Técnica** $\boldsymbol{[T_3], 1}$
  - ✓ $T^*_{obs\ recurr.}$: Observa la secuencia figural para comprender cómo cambian o crecen en cada paso la secuencia.
  - ✓ $T^*_{analiza\ crecimiento\ de\ la\ figura}$: Observa cómo cambian las características de las figuras a medida que avanza la secuencia.
    El grosor del perímetro va aumentando.
  - ✓ $T^*_m[ident.patrón\ recurrente]$: la unidad de cambio que se observa en la secuencia figural consiste en la variación en el grosor del perímetro de cada rombo, lo cual está vinculado al bloque "tamaño de lápiz" en Scratch. Este bloque se muestra en la Figura 56.

Figura 56
Bloque de código " cambiar tamaño de lápiz"



✓ T^*_{Dibujo}: dibujaremos $fig.1$. Después de completar el primer rombo (fig. 1), el sprite se detiene en el punto A', y a partir de allí, el siguiente rombo experimenta un cambio en el grosor de su perímetro (fig. 2), tal como se muestra en la figura 57.

Figura 57
***fig*. 1**. *de la secuencia de rombos correspondiente a la tarea* $\boldsymbol{t_{3,1}}$



✓ $T^*_m[unid.de\ patrón]$: para identificar la unidad de patrón en el script, se analizó el código del bloque personalizado "patrón", que dibuja un rombo en la ventana de ejecución. Se observó que, después de dibujar el primer rombo, el lápiz se levanta, interrumpiendo el trazado durante los siguientes 55 pasos, es decir, hasta llegar al punto A'. A partir de este punto, el lápiz se vuelve a bajar para dibujar el siguiente rombo, y es en este momento cuando se ajusta el tamaño del lápiz. Así, se pueden identificar dos posibles unidades de patrón: primero, el bloque personalizado "patrón", seguido por el bloque de movimiento "mover 55 pasos" y luego el bloque "cambiar tamaño de lápiz"; o alternativamente, el bloque "patrón", seguido por el bloque "cambiar tamaño de lápiz" y finalmente el bloque de movimiento "mover 55 pasos", como se ilustra en la figura 58.

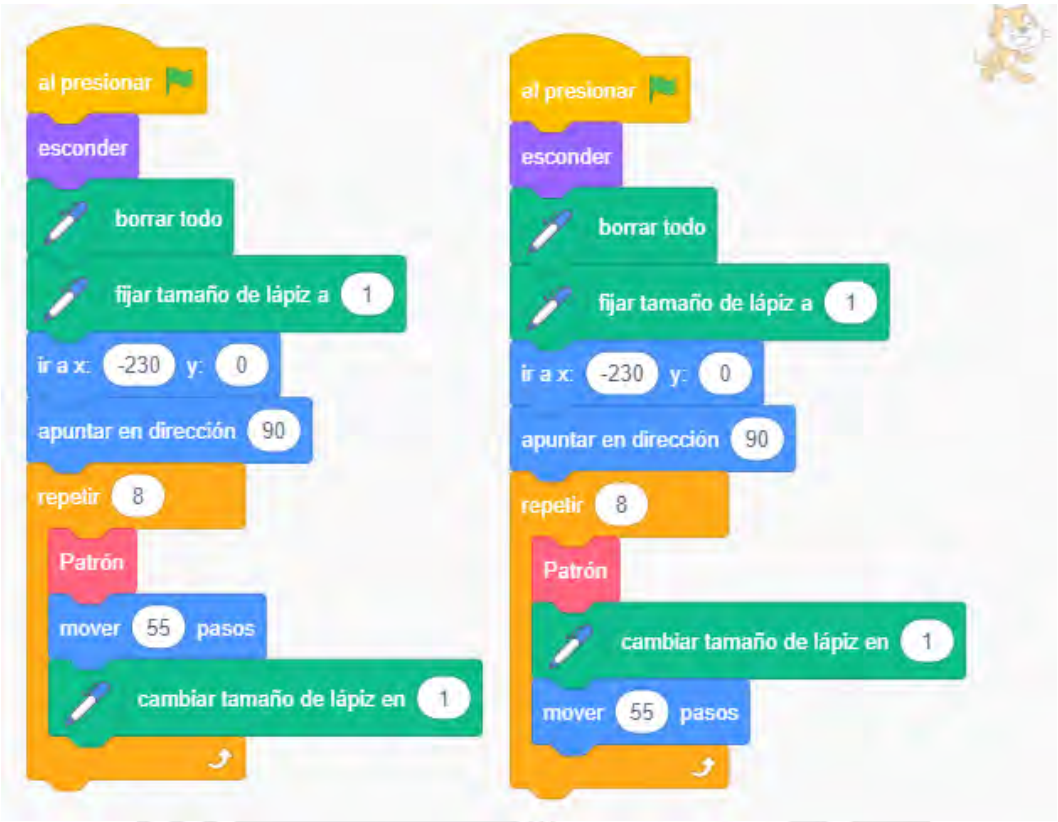
Figura 58

Unidades de patron para la tarea $\boldsymbol{t_{3,1}}$

- ✓ $T^*_{prog}[comenzar]$: una vez identificada la unidad de patrón, en la ventana de programación se inserta el bloque "cambiar tamaño de lápiz" y, posteriormente, se inicia el programa, tal como se muestra en la figura 59.

Figura 59

Script de la nueva secuencia de rombos (tarea $t_{3,1}$)



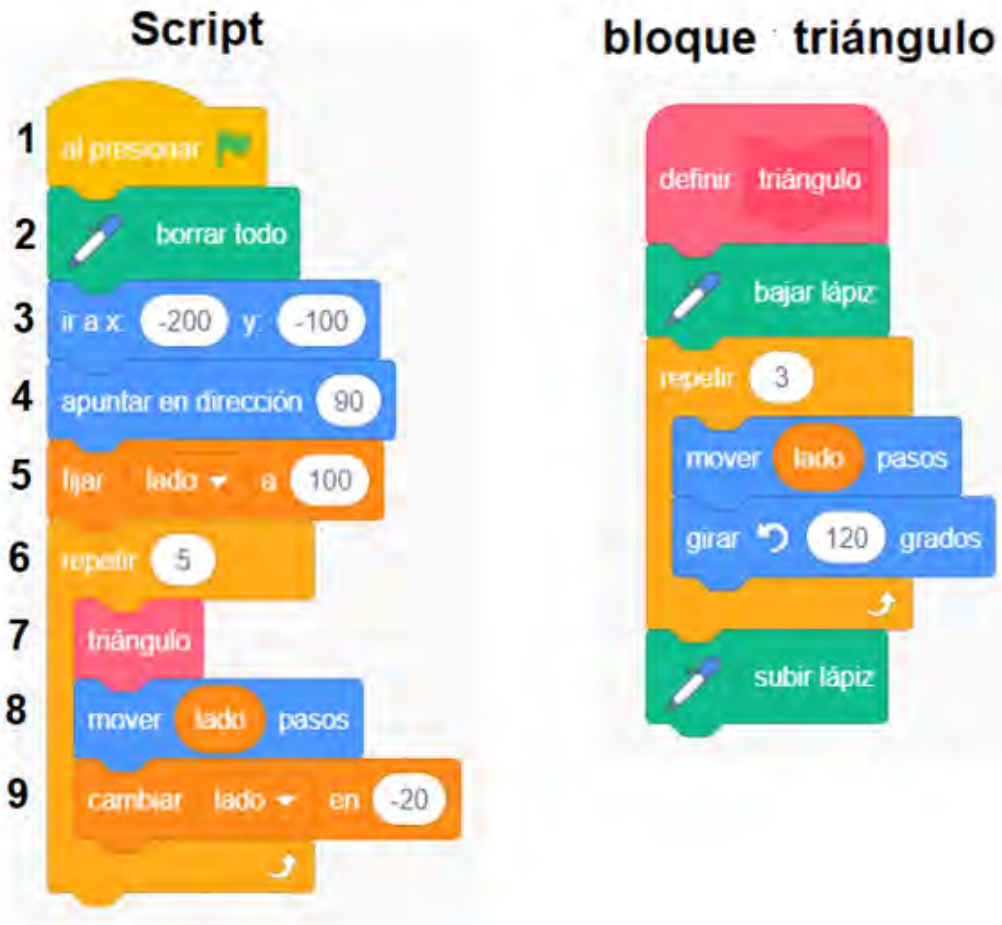
✓ $T^*_{prog}[Verificar]$: En la ventana de ejecución se observa que ambos programas mostrados en la figura 60 generan la secuencia de rombos correspondiente a la tarea.$t_{3,1}$

Figura 60

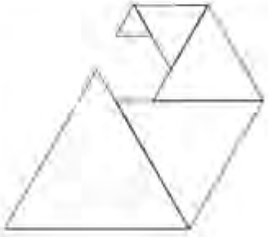
Programas que generan la secuencia de rombos para la tarea t_3,1

Figura 61
Alayrangues, et al. Ejercicio 2- (p.14)

Se da el siguiente programa que permite trazar varios triángulos equiláteros de tamaños diferentes. Este programa contiene una variable llamada "lado". Las longitudes se dan en píxeles. Se recuerda que la instrucción "orientarse a 90" significa que se dirige hacia la derecha.



1. ¿Cuáles son las coordenadas del punto de partida del trazado?
2. ¿Cuántos triángulos son dibujados por el script?
3.
   (a) ¿Cuál es la longitud (en pixeles) del lado del segundo triángulo trazado?
   (b) Dibuja a mano alzada cómo sería la figura obtenida al ejecutar este script.
4. Modificamos el script inicial para obtener la figura de al lado. Indica el número de una instrucción del script después de la cual se puede colocar la instrucción  para obtener esta nueva figura.



Fuente. Tomado y traducido de Alayrangues, et al., 2019, pag.14

Este ejercicio consiste en un programa diseñado para trazar múltiples triángulos equiláteros de diferentes tamaños utilizando el lenguaje de programación visual Scratch. El script está estructurado de tal manera que permite la variación de la longitud de los lados de los triángulos a través de una variable llamada “lado”, donde las medidas se expresan en píxeles.

El ejercicio 2 se compone de cuatro preguntas. En particular, la pregunta 3 incluye dos ítems: el ítem (a) presenta una tarea $t_{1,4}$ que corresponde al tipo de tarea $\boldsymbol{T_1}$, mientras que el ítem (b) presenta una tarea $t_{4,2}$ que pertenece al tipo de tarea $\boldsymbol{T_4}$. En la pregunta 4 presenta una tarea $t_{3,2}$ que corresponde al tipo de tarea $\boldsymbol{T_3}$

En el bloque personalizado “triángulo” se tiene los siguientes ingredientes:

- ✓ $T^*_{prog}[Crear\ bloque]$: Crear un bloque personalizado “triángulo”
- ✓ $T^*_{prog}[bajar\ lápiz]$: Se activa el lápiz para empezar a dibujar
- ✓ $T^*_{prog}[Repetir]$: Usar del bloque de repetición
- ✓ $T^*_{prog}[Repetir\ n\ veces]$: Insertar el valor de "3" dentro del bloque de repetición
- ✓ $T^*_{prog}[Crear\ variable]$: Crear la variable “lado”
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”

- ✓ $T^*_{prog}[mover\ variable]$: Insertar la variable “lado” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ \ n]$: Insertar “120” dentro del bloque “girar” en sentido antihorario.
- ✓ $T^*_{prog}[subir\ lápiz]$: levantar el lápiz para dejar de dibujar

En el Script: se observa los siguientes ingredientes:

- ✓ $T^*_{prog}[comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{prog}[borrar\ todo]$: se limpia la ventana de ejecución
- ✓ $T^*_{prog}[coordenadas]$: el cursor se posiciona en las coordenadas (-200, -100).
- ✓ $T^*_{prog}[orientación]$: El cursor se orienta hacia la derecha 90”
- ✓ $T^*_{prog}[fijar\ variable]$: Dar a la variable “lado” el valor de "100"
- ✓ $T^*_{prog}[repetir]$: Usar el bloque de control “repetir” en un programa Scratch
- ✓ $T^*_{prog}[Repetir\ n\ veces]$: Insertar el valor de "5" dentro del bloque de control “repetir”.
- ✓ $T^*_{prog}[usar\ bloque\ ]$: Usar el bloque personalizado “triángulo”
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ variable]$: Insertar la variable “lado” dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[cambiar\ valor\ de\ la\ variable]$: Cambiar el valor de la variable “lado” en -20
- ✓ $T^*_{prog}[verificar]$: Verificar el programa desarrollado en Scratch.

Pregunta 1: ¿Cuáles son las coordenadas del punto de partida del trazado?

Esta pregunta presenta una tarea $\boldsymbol{T^*_{prog}[coordenadas]}$: Posicionar el sprite en las coordenadas (x, y).

Técnica:

- ✓ $T^*_{m}[interpretar\ el\ bloque]$: Entender cómo funciona un bloque específico dentro de un programa en Scratch.

El bloque "Ir a x: -200 y: -100" establece que las coordenadas del punto de partida del trazado son (-200, -100). Esto indica que el sprite se desplazará a esa posición en la ventana de ejecución antes de iniciar el dibujo del triángulo.

Pregunta 2: ¿Cuántos triángulos son dibujados por el script?

Esta pregunta presenta una tarea $\boldsymbol{T^*_{prog}[Repetir\ n\ veces]}$: Insertar el valor de "$n$" dentro del bloque de control “repetir”.

Técnica:

- ✓ $T^*_{prog}[interpretar\ el\ bloque]$: Entender cómo funciona un bloque específico dentro de un programa en Scratch.
  Dentro del bloque de control "repetir", se ha insertado el valor de "5", lo que significa que el script se ejecutará cinco veces. Esto sugiere que se dibujarán un total de cinco triángulos, cada uno con una longitud de lado que se reducirá en cada iteración.

Pregunta 3

Ítem(a): ¿Cuál es la longitud (en pixeles) del lado del segundo triángulo dibujado?

Esta pregunta presenta una tarea $t_{1,2}$ del tipo de tarea $\boldsymbol{T_1}$

Tipo de tarea $\boldsymbol{T_1}$: Identificar la unidad de patrón presente en una secuencia figural.

Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_1]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_1}$

$\boldsymbol{GT_{prog}[T_1]}$: [Identificar la unidad de patrón presente en una secuencia figural, $\boldsymbol{V_{prog}}$[**tipo de figura**], $\boldsymbol{V_{prog}}$[**tipo de patrón**], $\boldsymbol{V_{prog}}$[**cantidad de pasos**],

$\boldsymbol{V_{prog}}$[á**ngulo de giro**]]

Subtipo de tarea

- $\boldsymbol{T'_{prog}[T_1, 2]}$**;** Identificar la unidad de patrón presente en una secuencia figural compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón recurrente decreciente, la cantidad de pasos es constante y el ángulo de giro es mayor a 90º, en sentido antihorario.

- **Técnica** $[\boldsymbol{T_1}, \mathbf{2}], \mathbf{1}$
  - ✓ ${T^*}_m[interpretar\ el\ programa]$: Comprender cómo funcionan sus diferentes componentes y cómo interactúan entre sí para lograr un objetivo específico.
  - **bloque personalizado "triángulo":** Se identifica que la figura es un triángulo equilátero.
  - **fijar variable "lado":** Dar a la variable "lado" el valor de "100".
    La longitud inicial del triángulo equilátero es de 100 pixeles
  - **cambiar valor de la variable "lado":** Cambiar el valor de la variable "lado" en " -20".
    La longitud del lado del triángulo equilátero se reduce en 20 unidades en cada iteración.
  - ✓ ${T^*}_{unid.\ de\ patrón\ recurrente}$**:** Esta unidad representará el cambio mínimo que ocurre de una figura a otra.

    Se identifica el patrón recurrente es -20
  - ✓ ${T^*}_{verifica\ regla\ de\ form.}$: Cada figura debería seguir el mismo patrón recurrente.
    La longitud del lado del segundo triángulo sería $100 - 20 = 80$ pixeles

Ítem(b): ¿Dibuja a mano alzada la forma de la figura obtenida al ejecutar este Script?
Esta pregunta presenta una tarea $t_{4,2}$ del tipo de tarea $\boldsymbol{T_4}$

Tipo de tarea: $\boldsymbol{T_4}$**:** Extiende patrones al menos una unidad de patrón

Tipo de tarea en Scratch: $\boldsymbol{T_{prog}}[\boldsymbol{T_4}]$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_4}$,

$\boldsymbol{GT_{prog}}[\boldsymbol{T_4}]$ *:* [Extiende patrones al menos una unidad de patrón. $\boldsymbol{V_{prog}}[\mathbf{tipo\ de\ figura}]$, $\boldsymbol{V_{prog}}[\mathbf{tipo\ de\ patrón}]$, $\boldsymbol{V_{prog}}[\mathbf{\ cantidad\ de\ pasos}]$, $\boldsymbol{V_{prog}}[\ \mathbf{ángulo\ de\ giro}]$]

Subtipo de tarea

- $\boldsymbol{T'_{prog}}[\boldsymbol{T_4}, \mathbf{2}]$**:** Extiende patrones al menos una unidad de patrón presente en una secuencia figural compuesta por figuras poligonales cerradas, donde cada figura

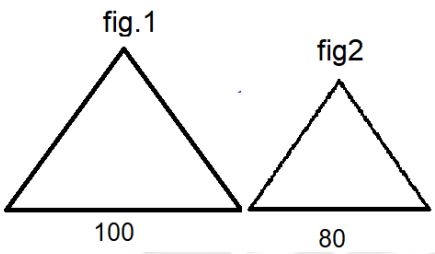
sigue un patrón recurrente decreciente, la cantidad de pasos es constante y el ángulo de giro es mayor a 90ª en sentido antihorario.

- **Técnica:**
  - ✓ $T^*_{unid.\ de\ patrón\ reccurrente}$**:** Esta unidad representará el cambio mínimo que ocurre de una figura a otra.

    se identifica el patrón recurrente es -20

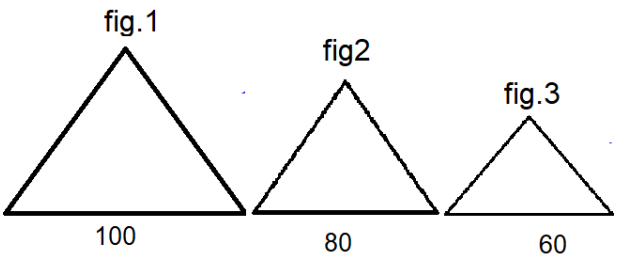
  - ✓ $T^*_{verifica\ regla\ de\ form.}$: Cada figura debería seguir el mismo patrón recurrente
    La longitud del lado del segundo triángulo sería: $100 - 20 = 80$ pixeles
  - ✓ T^*_{Dibujo}: Dibuja la figura 2, tal como ilustra en la figura 62

Figura 62
fig2: Triángulo equilátero de lado 80



  - ✓ $T^*_{verifica\ regla\ de\ form.}$: cada figura debería seguir el mismo patrón recurrente
    La longitud del lado del tercer triángulo sería: $80 - 20 = 60$ pixeles
  - ✓ T^*_{Dibujo}: Dibuja la figura 3, tal como ilustra en la figura 63

Figura 63
fig3: Triángulo equilátero de lado 60

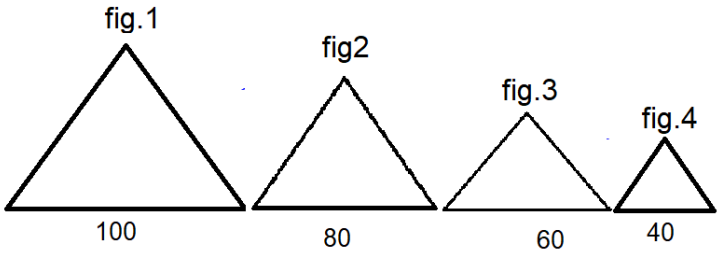


  - ✓ $T^*_{verifica\ regla\ de\ form.}$: Cada figura debería seguir el mismo patrón recurrente.

La longitud del lado del cuarto triángulo sería: $60 - 20 = 40$ pixeles

- ✓ T^*_{Dibujo}: Dibuja la figura 4, tal como ilustra en la figura 64

Figura 64
fig4: Triángulo equilátero de lado 40

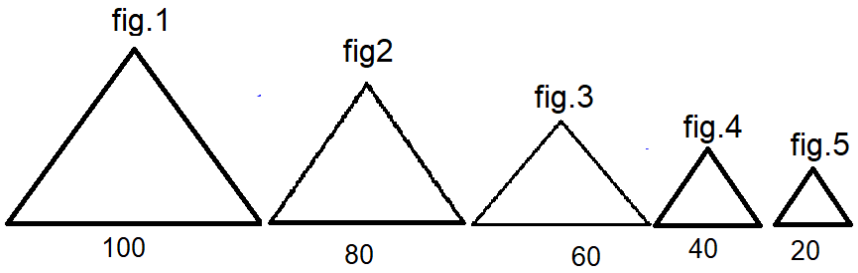


- ✓ $T^*_{verifica\ regla\ de\ form.}$: Cada figura debería seguir el mismo patrón recurrente

La longitud del lado del tercer triángulo sería: $40 - 20 = 20$ pixeles

- ✓ T^*_{Dibujo}: Dibuja la figura 5, tal como ilustra en la figura 65

Figura 65
fig5: Triángulo equilátero de lado 20



- ✓ $T^*_{prog}[comenzar]$: A continuación, se inicia el programa, desarrollado en Scratch, al hacer clic en la bandera verde. En la figura 66 se presenta el código completo del programa implementado en esta plataforma.

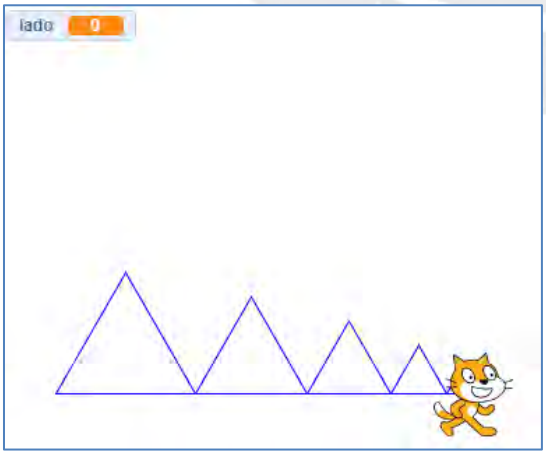
Figura 66

Script de la secuencia de triángulos correspondiente a la tarea $\boldsymbol{t_{4,2}}$

✓ $T_{Verificar}$: en la ventana de ejecución, verificamos los resultados generados por el programa y comprobamos si coinciden con lo que fue dibujado previamente en lápiz y papel, tal como se ilustra en la figura 68

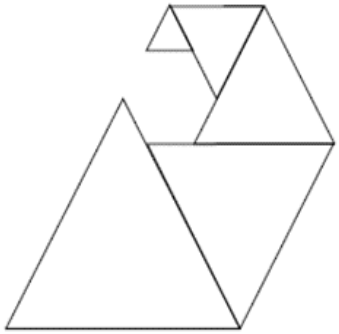
Figura 67

Secuencia de triángulos de la tarea $\boldsymbol{t_{4,2}}$ *en Scratch.*



pregunta 4: Modificamos el script inicial para obtener la figura de al lado. Indica el número de una instrucción del script. Luego, se puede colocar la instrucción 

para obtener esta nueva figura.



Esta pregunta presenta una tarea $t_{3,2}$ que corresponde al tipo de tarea $\boldsymbol{T_3}$

Tipo de tarea $\boldsymbol{T_3}$: Crear una secuencia partir de una secuencia dada.

Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_3]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_3}$

$\boldsymbol{GT_{prog}[T_3]}$: [Crear una secuencia partir de una secuencia dada, $\boldsymbol{V_{prog}}$[**tipo de figura**], $\boldsymbol{V_{prog}}$[**tipo de patrón**], $\boldsymbol{V_{prog}}$[**cantidad de pasos**], $\boldsymbol{V_{prog}}$[**ángulo de giro**]]

Sub tipo de tarea

- $\boldsymbol{T'_{prog}[T_3, 2]}$**:** Crear una secuencia a partir de una secuencia dada presente en una figura poligonal cerrada que presenta un patrón de recurrencia decreciente. La cantidad de pasos es constante y el ángulo de giro es mayor de 90º, en sentido antihorario.

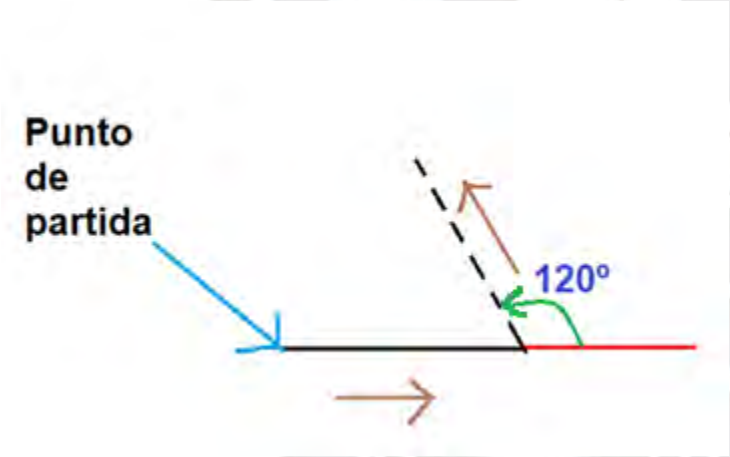
- **Técnica**

  - ✓ $T^*{}_m[analizar\ bloque\ ]$: Analiza el bloque personalizado “triángulo”

    - **Bloque personalizado llamado "triángulo",** permite encapsular las instrucciones necesarias para dibujar un triángulo en un solo lugar.
    - **Bajar lápiz:** Este bloque activa el lápiz, permitiendo que el sprite comience a dibujar en la ventana de ejecución.
    - **Repetir:** Se establece que el bloque de control “repetir”, debe ejecutarse 3 veces, que es la cantidad de lados en un triángulo.

- **Crear variable "lado":** Se define una variable llamada "lado" que se usará para controlar la longitud de cada lado del triángulo.
- **Mover variable “lado”:** Se inserta la variable "lado" en el bloque de movimiento “mover”, lo que significa que el sprite se moverá una distancia igual al valor de la variable "lado".
- **Girar 120:** Se establece que el sprite debe girar 120 grados, que es el ángulo externo de un triángulo equilátero.
- **Subir lápiz:** Este bloque levanta el lápiz, deteniendo el dibujo al final del proceso T^*_{Dibujo}: para realizar el dibujo geométrico del bloque personalizado "triángulo", se parte desde el punto de inicio, que corresponde a la coordenada (-200, -100), como se muestra en la figura 69.

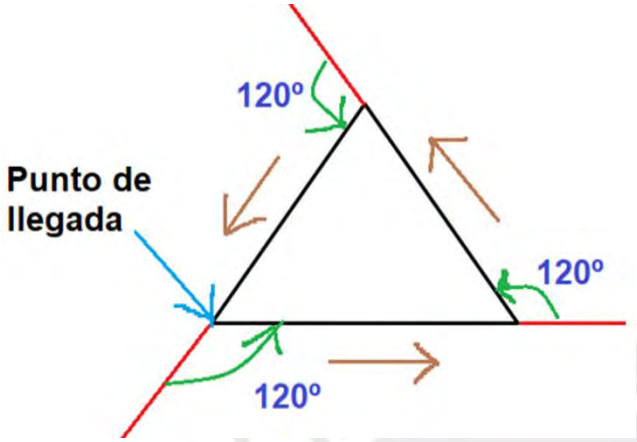
Figura 68
Punto de inicio del triángulo en la coordenada (-200, -100).



El punto de llegada es la coordenada (-200, -100), que coincide con el punto de inicio, cerrando así el polígono de tres lados, es decir, el triángulo, como se muestra en la figura 70.

Figura 69

Cierre del triángulo en el punto de inicio (-200, -100)



- ✓ $T^*{}_m[analizar\ el\ programa]$: Examinar la secuencia y la funcionalidad de los bloques.

  Script Principal:

- **Al presionar la bandera verde**: Se inicia el programa, lo que activa todo el script.
- **Borrar todo:** Se limpia la ventana de ejecución para asegurar que el dibujo anterior no interfiera con el nuevo.
- **Ir a x: -200 y: -100:** Se posiciona el sprite en las coordenadas (-200, -100), lo que establece el punto de inicio para el dibujo.
- **Apuntar en dirección 90:** Se orienta el cursor hacia la derecha, a 90 grados, asegurando que el triángulo se dibuje en la dirección correcta.
- **Fijar variable "lado":** Se asigna el valor 100 a la variable "lado", determinando la longitud de los lados del triángulo.
- **Repetir n veces:** Se usa el bloque de control “repetir” en cual se ejecutará 5 veces. Dentro de este bucle están los siguientes bloques:

  bloque personalizado "triángulo", que realiza el dibujo del triángulo con el valor actual de la variable "lado".

  Mover variable “lado”: Se mueve el sprite una distancia igual al valor de "lado" después de cada triángulo.

Cambiar valor de la variable “lado”: Se disminuye el valor de la variable "lado" en 20, lo que significa que el siguiente triángulo será más pequeño.

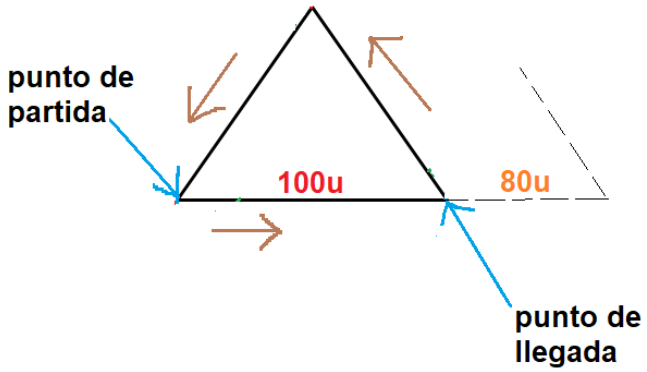
- ✓ $T^*{}_m[unid.\,de\,patrón]$: identifica la unidad de patrón en el programa, tal como se muestra en la figura 71.

Figura 70
Unidad de patrón de la secuencia de triangulos (tarea $\boldsymbol{t_{3,2}}$)

- ✓ $T^*{}_m[interpretar\,el\,programa]$: comprender cómo funcionan sus diferentes componentes y cómo interactúan entre sí para lograr un objetivo específico.
  Este programa generará cinco triángulos equiláteros, cada uno con lados de longitud decreciente: 100, 80, 60, 40 y 20 unidades. Esto se logra mediante el uso del bloque de control que repite la acción cinco veces.

  El uso del bloque personalizado "triángulo" permite que, cada vez que se invoca, se dibuje un triángulo equilátero. Tras cada dibujo, el sprite se mueve una distancia igual a la longitud del lado del triángulo dibujado y, al mismo tiempo, se reduce la longitud del lado para el siguiente triángulo, como se puede observar en la figura 72.

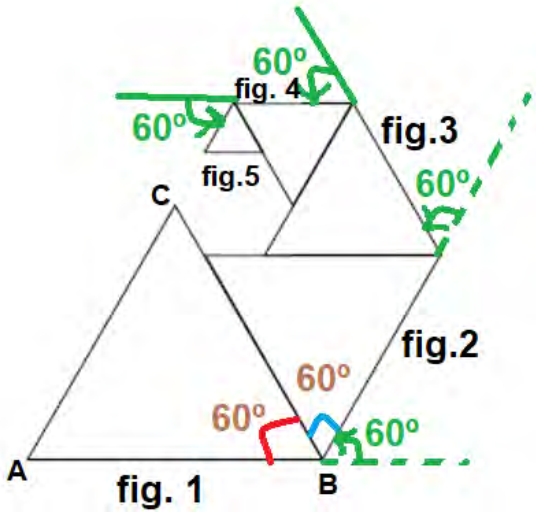
Figura 71.
Triángulo equilátero en Scratch



✓ $T^*_{analiza\ crecimiento\ de\ la\ figura}$: Analiza cómo cambian las características de las figuras a medida que avanza la secuencia.

Es crucial identificar en qué parte del script se debe insertar el bloque de movimiento "girar 60 grados" para obtener la figura deseada. Para ello debemos analizar detenidamente la figura y observar cómo se disponen los triángulos. Esto nos permitirá determinar el momento adecuado para agregar el giro y así lograr el patrón deseado., como se muestra en la figura 73.

Figura 72
Secuencia de triángulos correspondiente a la tarea $\boldsymbol{t_{3,2}}$



- ✓ $T^*{}_m[unid.\,de\,patrón]$: Se identifica la unidad de patrón después de que el sprite dibuje el triángulo (fig. 1) y se desplace a la posición donde dibujará el siguiente triángulo (punto B). Al girar 60 grados, la orientación del sprite cambiará para el próximo triángulo (fig. 2), lo que permite crear el patrón deseado. A partir de este análisis, se obtienen dos unidades de patrón, como se muestra en la figura 74.

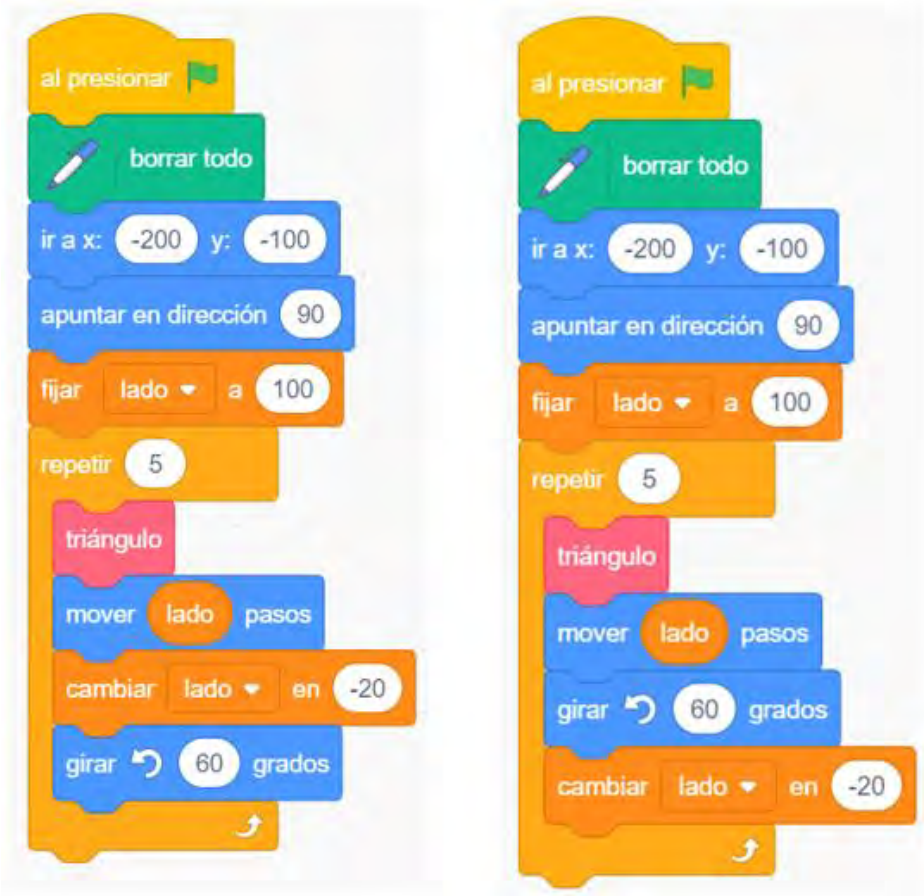
Figura 73

Unidades de patrón correspondiente a la tarea $\boldsymbol{t_{3,2}}$

- ✓ $T^*{}_{prog}[comenzar]$: iniciaremos el programa desarrollado en Scratch haciendo clic en la bandera verde. Dado que se encontraron dos unidades de patrón, se

ejecutarán dos programas. En la figura 75 se presentan los códigos correspondientes a ambos programas.

Figura 74
Script de la secuencia de triángulos correspondiente a la tarea $t_{3,2}$



✓ $T^*_{prog}[Verificar]$: Verificar el programa desarrollado en Scratch.

Este paso final permite revisar el funcionamiento del programa, asegurando que todo opere correctamente. En la figura 76 podemos ver el resultado de ejecutar el programa.

Figura 75

Secuencia de triangulos de la tarea $t_{3,2}$ creado en el programa Scratch

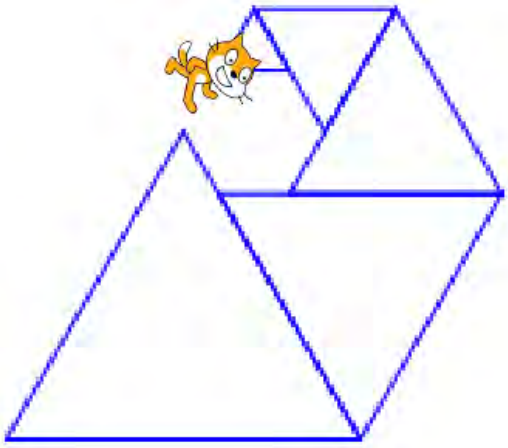


Figura 76. *Alayrangues, et al. Ejercicio 3- (p.15)*

Aquí hay tres figuras diferentes, ninguna está a la escala indicada en el ejercicio:

figure 1 figure 2 figure 3

El programa a continuación contiene una variable llamada "longitud".

1)

(a) Dibuja la figura obtenida con el bloque "una vuelta" dado en el marco de la derecha anterior, para una longitud inicial de 30, orientado hacia la derecha con el lápiz al inicio del trazado. Se tomará 1 cm por 30 unidades de longitud, es decir, 30 píxeles.

(b) ¿Cómo se está orientado el lápiz después de este trazado? (No se requiere justificación)

2) ¿Cuál de las figuras 1 o 3 permite obtener el programa anterior? Justifica tu respuesta.

3) ¿Qué modificación se debe realizar en el bloque "una vuelta " para obtener la figura 2 anterior?

Fuente. Tomado y traducido de Alayrangues, et al.,2019, p.15.

En esta tarea, se presenta un programa que traza figuras poligonales a partir de una variable llamada “longitud”. El script permite crear una figura a través de un bloque personalizado llamado «una vuelta», utilizando una serie de instrucciones que determinan la longitud de los lados y los giros.

Este ejercicio presenta

En el Bloque personalizado “una vuelta” se tiene los siguientes ingredientes:

- ✓ $T^*_{Crear\ bloque}$: Crear un bloque personalizado “una vuelta”
- ✓ $T^*_{Repetir}$: Usar del bloque de repetición
- ✓ $T^*_{Repetir\ 2\ veces}$: Insertar el valor de "2" dentro del bloque de repetición.
- ✓ $T^*_{Crear\ variable}$: Crear la variable “longitud”
- ✓ T^*_{mover}: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{mover\ longitud}$: Insertar la variable longitud dentro del bloque de movimiento “mover”
- ✓ T^*_{girar}: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{\ girar\ 90}$: Insertar “90” dentro del bloque “girar” en sentido antihorario
- ✓ $T^*_{cambiar\ valor\ de\ la\ variable}$: cambiar el valor de la variable “longitud” en 30
- ✓ $T^*_{Repetir}$: Usar del bloque de repetición
- ✓ $T^*_{Repetir\ 2\ veces}$: Insertar el valor de "2" dentro del bloque de repetición.
- ✓ T^*_{mover}: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{mover\ longitud}$: Insertar la variable longitud dentro del bloque de movimiento “mover”
- ✓ T^*_{girar}: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{\ girar\ 90}$: Insertar “90” dentro del bloque “girar” en sentido antihorario
- ✓ $T^*_{Repetir}$: Usar del bloque de repetición
- ✓ $T^*_{Repetir\ 2\ veces}$: Insertar el valor de "2" dentro del bloque de repetición.

En el Script: se observa los siguientes ingredientes:

- ✓ $T^*_{comenzar}$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{esconder}$: Elegir el bloque de apariencia “esconder”
- ✓ $T^*_{coordenadas}$: Posicionar en las coordenadas (0, 0) el cursor

- ✓ $T^*_{orientación}$: orientar hacia la derecha 90” el cursor
- ✓ $T^*_{fijar\ longitud}$: Dar a la variable “longitud” el valor de 30
- ✓ $T^*_{borrar\ todo}$: Limpiar la ventana de ejecución.
- ✓ $T^*_{tamaño\ lápiz}$: fijar el tamaño del lápiz en “x”
- ✓ $T^*_{bajar\ lápiz}$: Activar el lápiz para empezar a dibujar
- ✓ $T^*_{Bloque\ personalizado}$: usar el bloque personalizado “una vuelta”
- ✓ $T^*_{cambiar\ valor\ de\ la\ variable}$: cambiar el valor de la variable “longitud” en 30
- ✓ $T^*_{Repetir}$: Usar el bloque de repetición
- ✓ $T^*_{Repetir\ n\ veces}$: Insertar el valor de "2" dentro del bloque de repetición.
- ✓ $T^*_{Verificar}$: Verificar el programa desarrollado en Scratch.

Pregunta 1:

Ítem(a): Dibuja la figura obtenida con el bloque "una vuelta" dado en el marco de la derecha anterior, para una longitud inicial de 30, orientado hacia la derecha con el lápiz al inicio del trazado. Se tomará 1 cm por 30 unidades de longitud, es decir, 30 píxeles.

Esta pregunta presenta una tarea $\boldsymbol{T^*_m[Dibujo]}$**:** interpreta geométricamente el bloque personalizado “una vuelta”.

Técnica:

- ✓ $T^*_m[analizar\ bloque\ ]$: analizar el bloque personalizado “una vuelta”

Bloque personalizado “una vuelta”: se inicia el proceso al definir un bloque personalizado que encapsula la lógica de movimiento y giro del sprite.

Bloque de control “repetir”: se establece un bloque de control “repetir” que ejecutará las instrucciones dentro de él dos veces, lo que implica que las acciones se realizarán en ciclos.

Variable "longitud": se crea una variable llamada "longitud" que almacenará la distancia que el sprite avanzará.

Bloque de Movimiento “mover (longitud)”: El sprite se moverá una distancia igual al valor de la variable "longitud". Inicialmente, este valor debe establecerse antes de ejecutar el bloque.

Bloque de Movimiento "girar 90 grados (antihorario): después de avanzar, el sprite gira 90 grados en sentido antihorario.

Cambiar el valor de la variable "longitud" en 30: este bloque aumenta el valor de la variable "longitud" en 30, lo que hará que el siguiente movimiento sea más largo que el anterior.

Bloque de control "repetir 2 veces": Se repite nuevamente el proceso de avanzar y girar. Esto crea un patrón donde el sprite avanzará y girará en un bucle

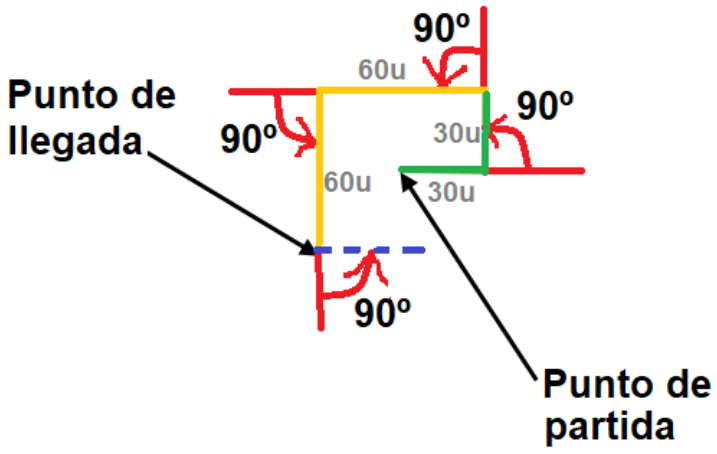
✓ $T^*{}_m[Dibujo]$: interpreta geométricamente el bloque personalizado "una vuelta".

Datos del problema:

- La longitud inicial es de 30 unidades.
- El sprite está orientado hacia la derecha, con el lápiz en la posición de inicio del trazado.
- La escala es de 1 cm por cada 30 unidades de longitud, lo que equivale a 30 píxeles.

En la figura 78 podemos ver la interpretación geométrica del boque personalizado "una vuelta"

Figura 77.
Interpretación geométrica del bloque personalizado 'Una vuelta'



Ítem(b): ¿Cómo se está orientado el lápiz después de este trazado? (No se requiere justificación)

Esta pregunta presenta una tarea $\boldsymbol{T^*_m[Dibujo]}$**:** interpreta geométricamente el bloque personalizado “una vuelta”.

Técnica:

- ✓ $T^*_{analiza\ crecimiento\ de\ la\ figura}$: Analiza cómo cambian las características de las figuras a medida que avanza la secuencia. Esto podría incluir cambios en el número de lados, el tamaño de los ángulos, la longitud de los lados, etc.

  Analicemos la figura 78 para determinar cómo está orientado el lápiz tras ejecutar el programa. Consideraremos la secuencia de movimientos y giros:

  Inicio: El sprite apunta hacia la derecha (0 grados). Primero, avanza 30 unidades en esa dirección. Luego, gira 90 grados en sentido antihorario, orientándose hacia arriba (90 grados). A continuación, avanza 30 unidades hacia arriba. Después, gira nuevamente 90 grados en sentido antihorario, apuntando hacia la izquierda (180 grados).

  La longitud se incrementa a 60 unidades (30 + 30). Luego, el sprite avanza 60 unidades hacia la izquierda. A continuación, gira 90 grados en sentido antihorario, quedando orientado hacia abajo (270 grados). El sprite avanza 60 unidades hacia abajo y, finalmente, gira 90 grados en sentido antihorario, regresando a la orientación inicial hacia la derecha (0 grados).

  Fin: Tras ejecutar todas estas instrucciones, el sprite termina orientado hacia la derecha.

  En conclusión, al final del trazado, el lápiz queda orientado hacia la derecha**.**

Pregunta 2: ¿Cuál de las figuras 1 o 3 permite obtener el programa anterior? Justifica tu respuesta.

Esta pregunta presenta una tarea $T^*_m[Dibujo]$: interpreta geométricamente el programa.

Técnica

✓ ${T^*}_m[analizar\ el\ programa]$: examinar la secuencia y la funcionalidad de los bloques.

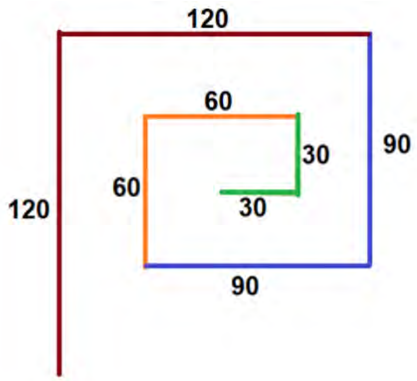
- **Al presionar la bandera verde:** Se activa cuando se hace clic en la bandera verde, iniciando la ejecución del script.
- **Esconder:** El sprite se esconde inicialmente para evitar cualquier trazo previo antes de empezar a dibujar.
- **Ir a x:0 y:0:** El sprite se posiciona en las coordenadas (0, 0), estableciendo el punto de inicio en la ventana de ejecución.
- **Apuntar en dirección 90:** Se orienta el sprite hacia la derecha (90 grados), lo que determina la dirección inicial del primer movimiento.
- **Fijar longitud a 30:** Se establece la variable "longitud" en **30 unidades**, que será la distancia inicial que avanzará el sprite.
- **Borrar todo:** Se limpia la ventana de ejecución para asegurarse de que el trazado previo no interfiera con el nuevo dibujo.
- **Fijar tamaño del Lápiz a 3:** Se ajusta el grosor del lápiz a 3.
- **Bajar lápiz:** Se coloca el lápiz en posición de escritura para comenzar a dibujar.
- **Repetir 2 veces:** Se usa el bloque de control “repetir” en cual se ejecutará 2 veces. Dentro de este bucle están los siguientes bloques:

  Bloque personalizado “una vuelta”: se utiliza el bloque "una vuelta" para ejecutar la lógica de dibujo definida anteriormente

  Cambiar longitud en 30: se incrementa el valor de la variable "longitud" en **30 unidades** para preparar el siguiente segmento de la espiral

✓ ${T^*}_m[Dibujo]$: para realizar la interpretación geométrica del programa, fue necesario analizar el código. En la figura 79 se muestra el trazo o dibujo generado a partir de este análisis.

Figura 78.
Espirolateral generada a partir del análisis del código del programa.



Por lo tanto, la figura 1 se obtiene al ejecutar el programa.

Pregunta 3: ¿Qué modificación se debe realizar en el bloque "girar" para obtener la figura 2 anterior?

Esta pregunta presenta una tarea $t_{2,1}$ que corresponde al tipo de tarea $\boldsymbol{T_2}$

Tipo de tarea $\boldsymbol{T_2}$: Traduce patrones en nuevos patrones con la misma regla estructural

Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_2]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_2}$

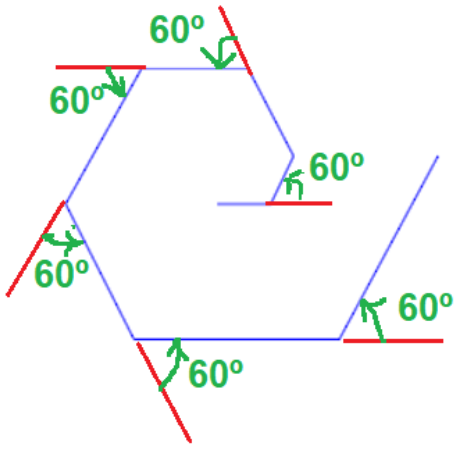
$\boldsymbol{GT_{prog}[T_2]}$: [Traduce patrones en nuevos patrones con la misma regla estructural, $\boldsymbol{V_{prog}}$**[tipo de figura]**, $\boldsymbol{V_{prog}}$**[tipo de patrón]**, $\boldsymbol{V_{prog}}$**[cantidad de pasos]**, $\boldsymbol{V_{prog}}$**[ángulo de giro]]**

Sub tipo de tarea

- $\boldsymbol{T'_{prog}[T_2, 1]}$**:** Traduce patrones en nuevos patrones con la misma regla estructural, compuesta por figuras poligonales abiertas, donde cada figura sigue un patrón recurrente creciente, la cantidad de pasos es creciente y el ángulo de giro es igual a 90ª, en sentido antihorario.
- **Técnica**
  - ✓ $T^*_{obs\ recurr.}$: Observa la secuencia figural para comprender cómo cambian o crecen en cada paso la secuencia.

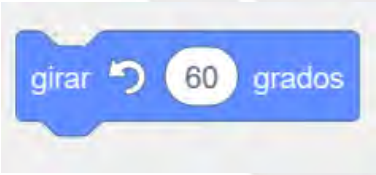
    Según la figura 2 del ejercicio propuesto, se puede observar que la espirolateral adopta la forma de un hexágono, tal como se puede observar en la figura 80.

Figura 79.
Espirolateral en forma de hexágono



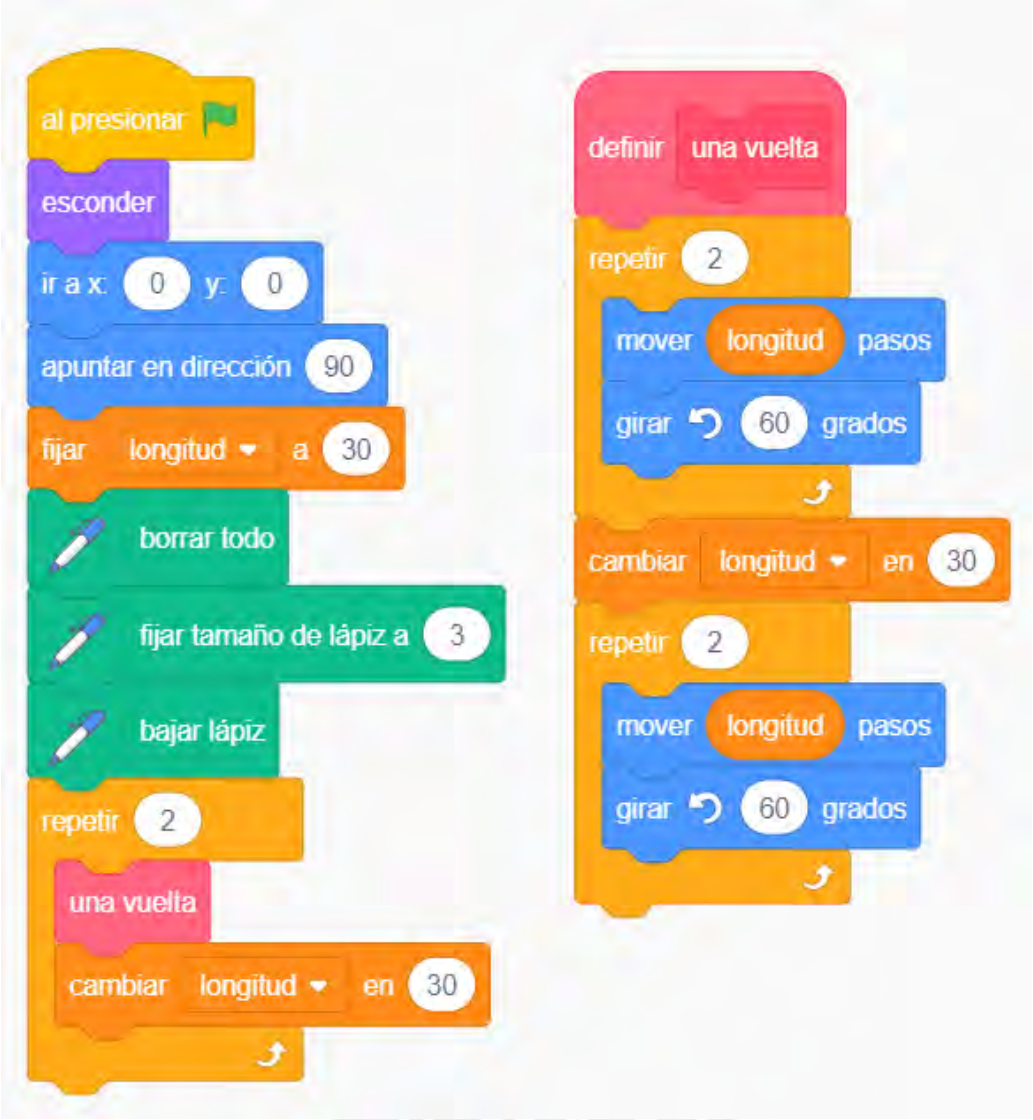
✓ $T^*_{prog}[girar\ n]$: insertar el valor "60" dentro del bloque de movimiento "girar" en sentido antihorario, ya que el ángulo exterior de un hexágono es de 60 grados, como se muestra en la figura 81.

Figura 80
Ángulo de 60 grados dentro del bloque de código: girar



✓ $T^*_{prog}[comenzar]$: Iniciar el programa desarrollado en Scratch haciendo clic en la bandera verde. En la figura 82 se muestra el bloque de movimiento “girar 60 grados” insertado en el programa.

Figura 81
Script de espirolateral correspondiente a la tarea $t_{2,1}$



✓ $T^*_{prog}[verificar]$: Verificamos el resultado obtenido al ejecutar el programa en la ventana de ejecución de Scratch, el cual se muestra en la Figura 83.

Figura 82

Espirolateral de la tarea $t_{2,1}$ creado en el programa Scratch

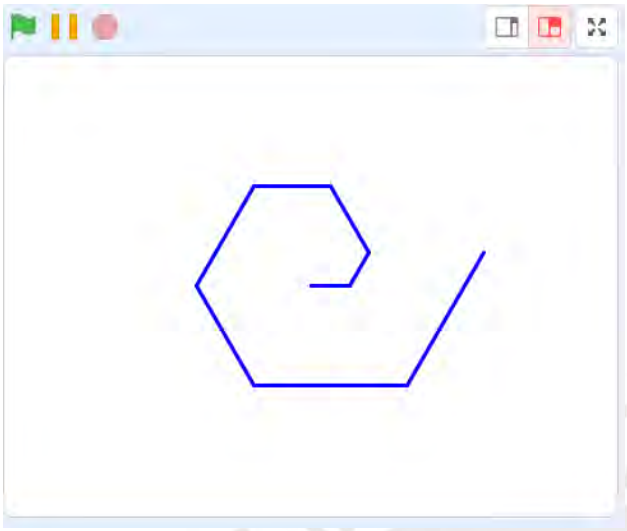
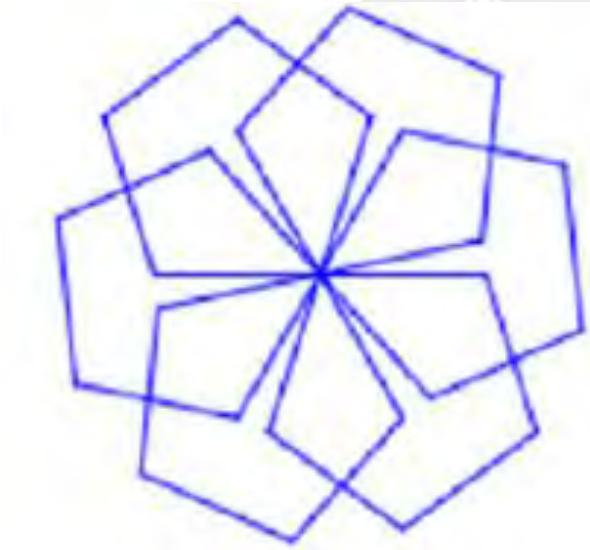


Figura 83. *Simetría rotacional de un pentágono regular*



Fuente. Tomado y traducido de Ye, et al., 2024, pág.698

Para crear la figura 84, se comenzó definiendo el bloque personalizado “pentágono regular” y, a continuación, utilizó el programa principal que se visualiza en la figura 85 para dibujar varios pentágonos rotados.

Figura 84
Script de la simetría rotacional de un pentágono regular.

Fuente. Tomado y traducido de Ye, et al., 2024, pag.700

¿Qué necesitas modificar para dibujar doce octágonos regulares alrededor de un punto y para crear su simetría rotacional?

En el bloque personalizado "Pentágono regular" se tiene los siguientes ingredientes:

- ✓ $T^*_{prog}[Crear\ bloque]$: Crear un bloque personalizado "pentágono regular"
- ✓ $T^*_{prog}[coordenadas\ ]$: posicionar el sprite en las coordenadas (0, 0).
- ✓ $T^*_{prog}[bajar\ lápiz]$: activar el lápiz para empezar a dibujar
- ✓ $T^*_{prog}[fijar\ variable]$: Dar a la variable "numero de lado" el valor de "0"
- ✓ $T^*_{prog}[Repetir\ hasta\ que\ ]$: Usar el bloque de control "repetir hasta que"
- ✓ $T^*_{prog}[Repetirhasta\ que\ "operador"\ veces\ ]$: Insertar el operador"$número\ de\ lados = 5$" dentro del bloque de control "repetir hasta que"
- ✓ $T^*_{prog}[mover]$: Elegir el bloque de movimiento "mover"

- ✓ $T^*_{prog}[mover\ ]$: Elegir el bloque de movimiento “mover”
- ✓ $T^*_{prog}[mover\ valor\ ]$: Insertar el valor 20 dentro del bloque de movimiento “mover”
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n\ ]$: Insertar 72 dentro del bloque “girar” en sentido horario
- ✓ $T^*_{prog}[cambiar\ valor\ de\ la\ variable]$: cambiar el valor de la variable “numero de lado” en "1"

En el Script: se observa los siguientes ingredientes:

- ✓ $T^*_{prog}[\ comenzar]$: Comenzar el programa desarrollado en Scratch
- ✓ $T^*_{prog}[coordenadas\ ]$: posicionar el sprite en las coordenadas (0, 0).
- ✓ $T^*_{prog}[\ borrar\ todo]$: se limpia la ventana de ejecución.
- ✓ $T^*_{prog}[fijar\ variable]$: Dar a la variable “grado total de rotación” el valor de "0"
- ✓ $T^*_{prog}[Repetir\ hasta\ que\ ]$: Usar el bloque de control “repetir hasta que”
- ✓ $T^*_{prog}[Repetir\ hasta\ que\ "operador"\ veces\ ]$: Insertar el operador "$grado\ total\ derotación = 360$" dentro del bloque de control “repetir hasta que”
- ✓ $T^*_{prog}[usar\ bloque\ ]$: usar el bloque personalizado “pentágono regular”
- ✓ $T^*_{prog}[girar\ ]$: Elegir el bloque de movimiento “girar” en sentido antihorario
- ✓ $T^*_{prog}[girar\ n\ ]$: Insertar 60 dentro del bloque “girar” en sentido horario
- ✓ $T^*_{prog}[cambiar\ valor\ de\ la\ variable]$: cambiar el valor de la variable “grado total de rotación” en "60"
- ✓ $T^*_{prog}[Verificar\ ]$: Verificar el programa desarrollado en Scratch.

Pregunta 1: ¿Qué necesitas modificar para dibujar doce octágonos regulares alrededor de un punto y para crear su simetría rotacional?

Esta pregunta presenta una tarea $t_{3,3}$ del tipo de tarea $\boldsymbol{T_3}$

Tipo de tarea $\boldsymbol{T_3}$: Crear una secuencia partir de una secuencia dada.

Tipo de tarea en Scratch: $\boldsymbol{T_{prog}[T_3]}$: Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_3}$

$\boldsymbol{GT_{prog}[T_3]}$: [Crear una secuencia a partir de una secuencia dada, $\boldsymbol{V_{prog}}$[**tipo de figura**], $\boldsymbol{V_{prog}}$[**tipo de patrón**], $\boldsymbol{V_{prog}}$[**cantidad de pasos**], $\boldsymbol{V_{prog}}$[**ángulo de giro**]]

Sub tipo de tarea

- $\boldsymbol{T'_{prog}[T_3, 3]}$**:** Crear una secuencia a partir de una secuencia dada, compuesta por figuras poligonales cerradas, donde cada figura sigue un patrón de repetición. Además, la cantidad de pasos es constante y el ángulo de giro es menor a 90º en sentido horario.
- **Técnica**
  - ✓ $T^*{}_m[analizar\ bloque\ ]$: Analizar el bloque personalizado “pentágono regular”

    Función general: Este bloque define un conjunto de instrucciones que se ejecutarán para dibujar un pentágono regular, lo que facilita su reutilización en otros lugares del programa. El nombre " pentágono regular " sugiere que este conjunto de acciones está específicamente diseñado para crear esta figura geométrica.

    Instrucciones dentro del bloque

    - **Ir a x: 0 y: 0**: Posiciona el sprite en el centro de la ventana de ejecución, lo que es útil para que el pentágono se dibuje simétricamente alrededor de este punto.
    - **Bajar lápiz:** Activa el lápiz para que comience a dibujar. Sin esta instrucción, el sprite se movería sin dejar rastro.
    - **Fijar variable "número de lado"**: Inicia la variable "número de lado" en 0. Esta variable se utilizará para contar cuántos lados del pentágono se han dibujado.
    - **Repetir hasta que**: Este bloque de control permite que el proceso de dibujo continúe hasta que se cumpla una condición específica.
    - **Condición "número de lados = 5"**: Dentro del bloque "repetir hasta que", se verifica si la variable "número de lado" ha alcanzado 5, lo que indica que se han dibujado todos los lados del pentágono.
    - **Mover**: Se elige el bloque de movimiento "mover". Este bloque es fundamental para avanzar y dibujar el lado del pentágono.

- **Mover 20 pasos**: Se inserta el valor 20, lo que significa que el sprite avanzará 20 unidades para dibujar cada lado del pentágono.
- **Girar 72 grados**: Se elige el bloque de movimiento "girar" en sentido antihorario. Este giro es crucial, porque en un pentágono regular, cada ángulo interno es de 108 grados, lo que requiere girar 72 grados para preparar el siguiente lado.
- **Cambiar valor de la variable "número de lado"**: Después de dibujar cada lado, se incrementa la variable "número de lado" en 1. Esto permite llevar un control de cuántos lados se han dibujado hasta el momento.

✓ $T^*{}_m[interpretar\ el\ bloque]$: Entender el propósito del bloque y cómo logra ese objetivo

- **Fijar variable "número de lado"**: Inicia la variable "número de lados" en 0. Esta variable servirá para contar cuántos lados de la figura se han dibujado. En este caso, se traza ocho lados, que corresponden a un octágono.
- **Repetir hasta que**: Este bloque de control garantiza que el proceso de dibujo continúe hasta que se cumpla una condición específica.
- **Condición "número de lados = 8"**: Dentro del bloque "repetir hasta que", se comprueba si la variable "número de lados" ha alcanzado el valor de 8. Esto indica que se han trazado todos los lados de la figura deseada, en este caso, un octágono.
- **Girar 45 grados**: Se elige el bloque de movimiento "girar" en sentido antihorario. En un octágono regular, cada ángulo interno es de 135 grados, lo que requiere girar es 45 grados para preparar el siguiente lado.

✓ $T^*{}_m[analizar\ el\ programa]$: **E**xaminar la secuencia y la funcionalidad de los bloques

- **al presionar la bandera verde**: Este bloque inicia la ejecución del programa. Para ello, se activa todos los procesos que se seguirán.
- **Ir a x: 0 y: 0**: Posiciona el sprite en el centro de la ventana de ejecución permitiendo que cualquier forma que se dibuje esté centrada.
- **Borrar todo**: Limpia la ventana de ejecución para asegurarse de que no haya dibujos previos. Para esto, comienza desde un lienzo en blanco.

- **Fijar variable "grado total de rotación"**: Inicia la variable "grado total de rotación" a 0. Esta variable llevará el seguimiento del total de grados que el sprite ha girado.
- **Repetir hasta que**: Este bloque de control permite que el programa continúe ejecutándose hasta que se cumpla una condición específica.
- **Condición "grado total de rotación = 360"**: Dentro del bloque "repetir hasta que", se verifica si la variable ha alcanzado 360 grados, lo que indicará que se ha completado una vuelta completa.
- **Usar bloque "pentágono regular"**: Se invoca el bloque personalizado que dibuja un pentágono regular. Cada vez que se llama, se dibuja un pentágono.
- **Girar 60 grados**: El sprite gira 60 grados en sentido antihorario después de cada pentágono. Esto permite que cada nuevo pentágono se dibuje con un ángulo de separación entre ellos.
- **Cambiar valor de la variable "grado total de rotación"**: Se incrementa la variable "grado total de rotación" en 60 grados. Esto mantiene un seguimiento del total de rotaciones realizadas.

✓ $T^*{}_m[interpretar\ el\ programa]$: Comprende cómo funcionan sus diferentes componentes y cómo interactúan entre sí para lograr un objetivo específico.

El objetivo del programa es crear un patrón de repetición geométrico. Para esto, se usa octágonos regulares dispuestos alrededor del punto central (0, 0), lo que dará lugar a un diseño circular.

Cuando se ejecuta este programa, el sprite:

Se posiciona en el centro de la ventana de ejecución.
Limpia la pantalla para empezar de nuevo.
Inicializa el contador de grados de rotación en 0.
Entra en un bucle que continúa hasta que el total de rotación alcance 360 grados.
Dentro del bucle:

- **Dibuja un octágono** al llamar al bloque personalizado "pentágono regular".

- **Gira 30 grados** en sentido antihorario para que el siguiente pentágono se dibuje en un nuevo ángulo.
- **Incrementa el contador** de grados de rotación en 30.

El proceso se repite, dibujando un total de 12 octágonos (360/30) alrededor del punto central (0, 0),

Figura 85

Script de la simetría rotacional de un octógono regular

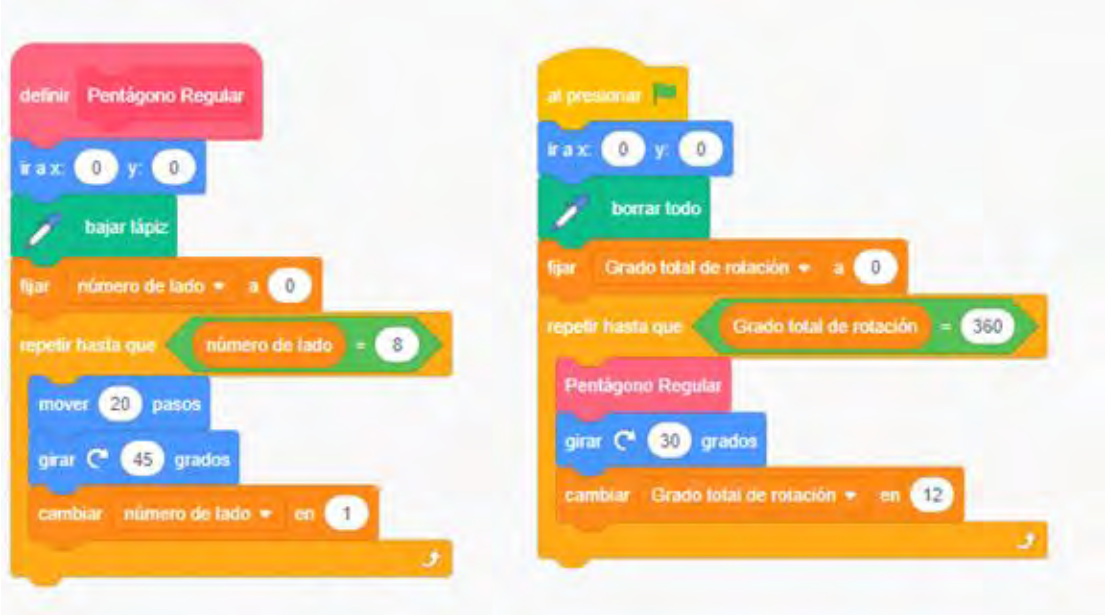
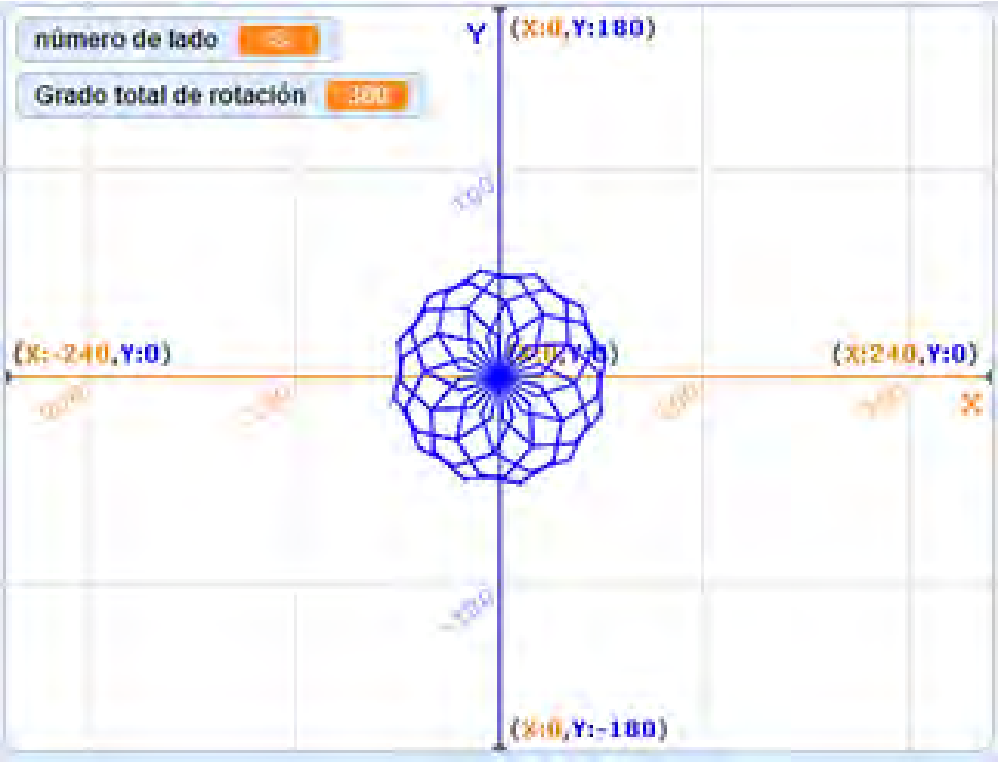


Figura 86

Visualización de la simetría rotacional de un octógono regular



6.2 Resultados

A continuación, se presenta los resultados obtenidos del análisis de las tareas realizadas.

1 Se llevó a cabo un análisis praxeológico mixto en el que se identificaron 10 tareas las cuales se clasificaron en 8 subtipos distintos dentro de los tipos de tarea de $T_{prog}[T_{SF}]$. Para el logro de ello, se ha basado en las variables previamente identificadas. Además, se identificaron 7 ingredientes técnicos clave dentro de esta praxeología. En la mayoría de las tareas se utiliza el ingrediente técnico mixto ${T^*}_m[unid.de\ patrón]$, utilizado para desarrollar el tipo de tarea T_1.

Relacionado a Scratch

2 En el análisis de las tareas $t_{1,1,1}$, $t_{1,1,2}$, $t_{2,1}$, $t_{3,1}$, $t_{3,2}$, $t_{3,3}$ y $t_{4,1}$ se identificó como técnica el uso de bloques personalizados, lo que mejoró la organización y la claridad del programa. Al usar bloques personalizados en Scratch, se evita la repetición innecesaria de código, lo que facilita la estructuración del programa. Además, el uso

de estos bloques hace que sea más sencillo realizar cambios en el programa sin tener que modificar todas las partes donde se repite el código. Esto permite que se pueda identificar la unidad de repetición en la secuencia figural y optimizar procesos.

3 En el análisis de las tareas $t_{3,2}$, $t_{2,1}$, $t_{3,3}$, se utiliza la técnica de las variables, las cuales se pueden identificar fácilmente gracias a que Scratch proporciona bloques específicos para trabajarlas. Estos bloques pueden ser nombrados (permitiendo asignarles identificadores como "lado" o "longitud") y explorados (facilitando la observación de cómo cambian o interactúan dentro del programa). Esta técnica permite almacenar y modificar valores de manera flexible. Sin el uso de variables, los programas serían considerablemente más limitados, ya que no podrían gestionar ni interactuar con la información dinámica que cambia durante su ejecución.

4 En la mayoría de las tareas se pudo identificar como técnica el uso del bloque de control “repetir” el cual permite ejecutar un conjunto de acciones múltiples veces de manera eficiente y sin necesidad de duplicar el código, facilitando la creación de patrones o movimientos repetitivos en los programas de Scratch.

5 En las tareas $t_{3,3}$ y $t_{4,1}$se observa la aparición de un nuevo tipo de patrón, no descrito previamente en la literatura: el patrón circular. Según Benton et al. (2017), para crear un patrón circular completo es necesario aplicar el conocimiento matemático sobre un giro total de 360°, lo que permite completar la figura de manera adecuada. En el análisis de estas tareas se observó que una técnica resulta más eficiente que la otra. La tarea $t_{4,1}$ emplea la técnica de utilizar el bloque de control "repetir" para completar la figura, mientras que en la tarea $t_{3,3}$ utiliza la técnica de la variable "grado total de rotación" junto con el bloque de control "repetir hasta que", lo cual facilita el seguimiento del número total de rotaciones realizadas para completar la figura. Esta última técnica resulta ser más eficiente que la anterior.

Relacionada a la secuencia figural

6 En el análisis de la tarea $t_{1,1,1}$, se identificó la necesidad de considerar a la figura rombo como una secuencia de segmentos consecutivos, lo que permitió identificar el ingrediente técnico mixto $\boldsymbol{T^*_m[unid.\,de\,patrón]}$ **,** para completar el bloque personalizado. Además, se advirtió que la orientación del sprite no se proporciona como dato, lo que genera una ambigüedad, dado que podrían existir diversas

soluciones posibles, tal como señalan Alayrangues et al. (2019). Esta situación ocurre, porque en la ventana de ejecución de Scratch, el plano de referencia tiene una orientación determinada, lo que permite que cada sprite cuente con una variable reservada que almacena su orientación.

Relacionada a la matemática

7 Con respecto a la tarea $t_{4,1}$, se utilizó los ingredientes de la técnica $\boldsymbol{T'_{und.patrón}}$, identificándose cuatro posibles rotaciones de la figura: 30° en sentido horario, 30° en sentido antihorario, 150° en sentido horario y 150° en sentido antihorario. Estos ángulos corresponden a los ángulos interiores del rombo. Hubiera sido interesante que no se especificaran los bloques y que se pidieran todas las soluciones posibles para crear la figura. Alternativamente, la tarea podría haberse reestructurado para que solo existiera una solución, por ejemplo, utilizando un bloque de giro en sentido antihorario y asegurando que el ángulo de giro fuera menor a 90°.Además, como indican Alayrangues et al. (2019), sería valioso que el solucionador reflexionara sobre el número de repeticiones necesarias para completar la roseta. Esto habría brindado la oportunidad de calcular cuántas veces debe repetirse el rombo para completar una vuelta de 360°. Este cálculo podría realizarse dividiendo 360° entre el ángulo de rotación (30°), es decir, 360° / 30°. En este contexto, sería relevante que el solucionador justificara por qué se utiliza 30° en lugar de 150°. Esta reflexión no solo permitiría al estudiante justificar sus soluciones, sino que también pondría a prueba sus conocimientos básicos de geometría y aritmética, así como su capacidad de argumentación.

8 En el análisis de las tareas relacionadas a las secuencias figurales $t_{1,1,1}$, $t_{1,1,3}$, $t_{3,2}$, $t_{3,3}$, se observó que el ingrediente $T^*{}_m[analizar\ bloque\ ]$ y $T^*{}_m[Dibujo]$ utilizado para representar a los polígonos regulares en Scratch no sigue exactamente la definición tradicional que se enseña en matemáticas. En matemáticas, un polígono regular se define como una figura geométrica con lados de igual longitud y ángulos internos iguales. Sin embargo, en el programa Scratch se describe el polígono como una figura que se construye mediante una serie de segmentos de igual longitud, con un ángulo constante entre cada segmento, pero ese ángulo no es el ángulo interno de la figura, sino su ángulo externo. Como señalan Broley et al. (2023), el enfoque

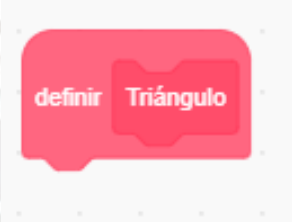
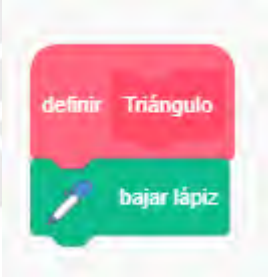
innovador de Scratch sobre la geometría plana transforma la manera en que comprendemos las figuras geométricas. El uso de esta plataforma de programación modifica la forma tradicional de abordar y trabajar con figuras. Esto permite a los usuarios experimentar con ellas de manera más dinámica y visual. En lugar de ceñirse a las reglas y definiciones estáticas propias de la geometría clásica, Scratch ofrece una perspectiva más interactiva, lo que fomenta un enfoque diferente y promueve el desarrollo de habilidades matemáticas. Así, los usuarios pueden pensar de manera más creativa y flexible sobre las matemáticas.

8.1 La construcción de figuras geométricas ha sido un ejercicio fundamental en el aprendizaje de las matemáticas desde tiempos antiguos. Tradicionalmente, este proceso se ha llevado a cabo utilizando herramientas sencillas como el lápiz, la regla y el compás. En este enfoque clásico, el dibujante debe seguir pasos precisos y lógicos para lograr una representación exacta de las formas, garantizando que las medidas y los ángulos sean correctos. Cada trazo se realiza a mano, y la habilidad de coordinar las herramientas se vuelve esencial para crear figuras como polígonos regulares.

Por otro lado, con el advenimiento de las tecnologías digitales, herramientas como Scratch permiten la creación de figuras geométricas de una manera más interactiva y dinámica. Scratch, un entorno de programación visual, ofrece bloques de código que los usuarios pueden combinar para crear secuencias precisas que generen figuras en la pantalla. Al igual que en el dibujo tradicional, la construcción de figuras en Scratch requiere seguir una serie de instrucciones, pero ahora se hace a través de bloques de código que controlan el movimiento, el giro y las repeticiones para crear las formas deseadas. En la tabla 13 se muestra un ejemplo de estos dos enfoques.

Tabla 13

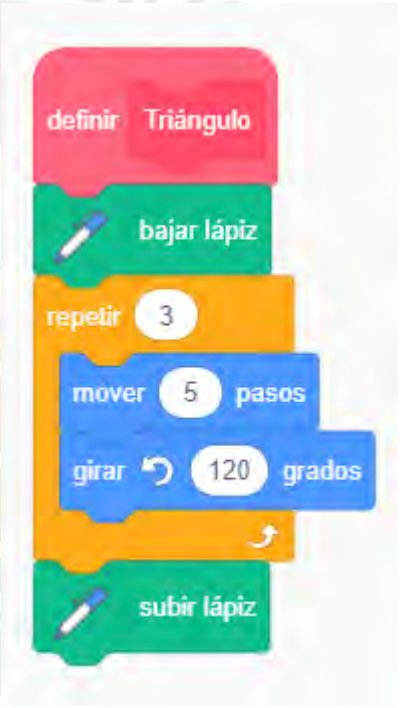
Comparación entre el enfoque tradicional y el uso de Scratch para la construcción de figuras geométricas

Aspecto	Lápiz y Papel	Scratch
Herramientas utilizadas	lápiz, la regla y el compás	Computadora, bloques de código
Construcción de un triángulo equilátero	Usamos una regla para trazar un segmento AB de 5 cm. Este será el primer lado del triángulo.	Crear el bloque personalizado" Triangulo" 
	Colocamos el compás en un extremo del segmento y ajustamos la apertura para que tenga la misma longitud del segmento AB. Luego, dibujamos un arco con centro A y radio AB	Se encaja el bloque "Bajar lápiz" con el bloque personalizado, para que el sprite empiece a dibujar en la ventana de ejecución, 
	Colocamos el compás en el otro extremo del segmento AB, asegurándonos de que la apertura siga siendo	Se configura el bloque de control "Repetir" para que se ejecute 3 veces, correspondiente a la

	de 5 cm. Dibujamos un arco con centro en B y radio AB tal que se cruce con el primer arco.	cantidad de lados de un triángulo.  Dentro de este bucle deben estar los siguientes bloques: El bloque de movimiento "mover" se ajusta a 5 pasos, lo que indica que el sprite se moverá esa distancia. 
	El punto donde los dos arcos se intersectan será el tercer vértice del triángulo.	El bloque de movimiento" girar", el sprite se configura para girar 120 grados en sentido antihorario, correspondiente al ángulo externo de un triángulo equilátero.  Finalmente, se conectan ambos bloques de movimiento dentro del bucle.

Encajamos el bloque “Subir lápiz” para detener el dibujo al final del proceso

Usamos la regla para conectar los tres puntos (los dos extremos del segmento y el punto de intersección).

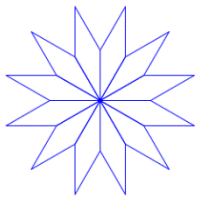
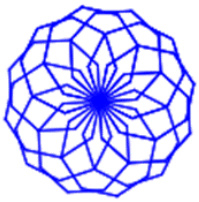


8.2 El bloque personalizado construido en las tareas de secuencias figurales permite generar la figura correspondiente a la secuencia. Además, este bloque me permite construir una figura más compleja. Por ejemplo en la tarea $t_{3,3}$ el bloque personalizado construye un pentágono regular, luego este mismo bloque

personalizado en el programa genera una figura compuesta formada por varios pentágonos rotados como podemos ver en la tabla 14

Tabla 14

Figuras compuestas creadas por el bloque personalizado

Figura compuesta		
Figura base	Rombo	Octógono regular
Bloque personalizado (para determinar la figura base)	número de lados = 4 Rotación 30º	número de lados = 8 Rotación 45º
Simetría	Simetría rotacional	Simetría rotacional
Script (determina el grado y número de rotaciones)	Rotación 30º Girar 12 veces	Rotación 30º Girar 12 veces

Relacionada a la praxeología mixta

9 En las técnicas utilizadas en las tareas $t_{1,1,3}$, $t_{3,1}$, $t_{3,2}$ y $t_{2,1}$ se emplearon ingredientes mixtos, como $T^*{}_m[unid.\,de\,patrón]$ y $T^*{}_m[Dibujo]$ lo que resalta la necesidad de una praxeología mixta. Esto se debe a que, para generar secuencias figurales en Scratch, es necesario comprender tanto aspectos matemáticos como los propios del entorno Scratch.

Los ingredientes algorítmicos ${T^*}_{prog}[coordenadas]$, ${T^*}_{prog}[Repetir\ n\ veces]$ y el ingrediente mixto ${T^*}_m[Dibujo]$ se consideran tareas. Tal como señala Kaspary (2020), una de las primeras ventajas de las técnicas es la dinámica entre los roles de "objeto" y "herramienta". Esta distinción es fundamental para comprender cómo las tareas evolucionan dentro de las técnicas. Las tareas, prestablecidas, inicialmente como simples, se transforman en herramientas claves para el aprendizaje y la implementación de técnicas más complejas. De esta manera, se facilita que los estudiantes apliquen de forma efectiva los conocimientos adquiridos en un contexto más amplio.

Se observa que la formulación de la tarea $t_{1,1,3}$ en Scratch, relacionada con las secuencias figurales, genera confusión en el uso del término "Patrón". Por un lado, se refiere a la figura que se produce, en este caso el rombo, y por otro lado, hace referencia al bloque personalizado definido como "bloque patrón", tal como lo mencionan Alayrangues et al. (2019). En esta investigación, el ingrediente ${T^*}_m[analizar\ el\ programa]$ de esta tarea podría prestarse a confusión desde dos perspectivas: la de la programación y la de la matemática. En el contexto de la programación, el patrón se entiende como la figura y el movimiento que realiza el sprite, mientras que, en matemáticas, el término se refiere exclusivamente a la figura.

Conclusiones

La **Teoría Antropológica de lo Didáctico (TAD)**, en particular, el **modelo praxeológico T4TEL**, ha proporcionado herramientas valiosas para formalizar y estructurar los contenidos de la organización matemática mixta en una secuencia figural. Estas herramientas nos permiten estudiar e identificar las praxeologías mixtas presentes en las tareas matemáticas realizadas con Scratch en relación con este objeto matemático.

En relación con el objetivo general planteado: “Describir y analizar las praxeologías mixtas que se presentan en las tareas matemáticas con Scratch asociadas a la secuencia figural en Educación Primaria”, se puede afirmar que se ha cumplido lo propuesto. Este cumplimiento nos lleva a señalar que el análisis de los materiales y recursos identificados en diversas investigaciones y libros de texto revela praxeologías centradas en la resolución de distintos tipos de tareas.

Para alcanzar el objetivo general, se ha definido tres objetivos específicos. El primero consistió en determinar una organización matemática de referencia para la secuencia figural. Las nociones de generador de tareas y variables permitieron identificar tres subtipos de tareas dentro del tipo de tarea $\boldsymbol{T_{SF}}$: Generalizar el patrón en una secuencia figural. Se usó la noción de "ingrediente" de una técnica y se describió los ingredientes de las técnicas para estos subtipos, lo que resultó pertinente para la construcción del Modelo de Praxeológico Referencia (MPR).

Además, la elaboración del MPR fue significativa, ya que se consideraron los subtipos de tareas y los ingredientes de las técnicas en la construcción de la Organización Matemática Mixta (OMM).

El segundo objetivo fue identificar las organizaciones matemáticas mixtas asociadas a la secuencia figural en tareas matemáticas realizadas con Scratch. Para lograrlo, fue fundamental reconocer las variables que influyen en el proceso del MPR, pues permitió identificar cuatro tipos de tareas y siete ingredientes técnicos en el plano praxeológico mixto.

El tercer objetivo consistió en construir un modelo praxeológico de referencia mixto que relacione las tareas matemáticas en Scratch con la secuencia figural. Para lograrlo, fue necesario utilizar la organización matemática (OM) de referencia de la

secuencia figural. El fin era comprender si y cómo la Organización Matemática Mixta (OMM) elaborada podría integrarse en este marco. Basándonos en la investigación de Crisci (2020), se define el tipo de tarea $\boldsymbol{T_{prog}}[\boldsymbol{T_{SF}}]$ "Desarrollar un programa en Scratch que ejecute $\boldsymbol{T_{SF}}$", siendo $\boldsymbol{T_{SF}}$ un tipo de tarea de la secuencia figural. Se definieron variables a partir del programa Scratch que permitió identificar cuatro tipos de tarea. Además, se analizaron cinco programas que modelizan las tareas de la OMM elaborada.

El uso de Scratch en el estudio de secuencias figurales permite trabajar conceptos geométricos fundamentales, tales como las transformaciones geométricas (rotaciones, simetrías, etc.). Permite la interacción de manera dinámica y visual con las figuras, Scratch transforma la manera tradicional de abordar la geometría plana, pues ofrece una visión más flexible y creativa de las matemáticas. Se observa que permite modificar las posiciones, tamaños y orientaciones de las figuras, lo que fomenta el trabajo intuitivo y visual de los conceptos geométricos.

Realizar el análisis de recursos didácticos tiene su limitación, este tipo de investigaciones solo permite ver qué tipos de tareas se ofrecen para el aprendizaje. Por ello, es fundamental contrastar el Modelo Epistemológico de Referencia (MER) con la construcción real del conocimiento dentro de la comunidad de estudio. Para futuras investigaciones se sugiere diseñar secuencias didácticas en la que se implemente actividades asociadas a los tipos de tareas descritos en este estudio, para así completar el análisis mediante la dialéctica medio-media**.** Nuestra hipótesis es que el software Scratch permite una retroalimentación explícita de los procedimientos realizados, lo cual genera medios de control para validar las conjeturas en la realización de las tareas.

Referencias

Alayrangues, S., Beffara, E., Daniel, S., Declercq, C., Héam, A., Loddo, J. V., ... & Volte, E. (2019). Une analyse des exercices d'algorithmique et de programmation du brevet 2017. *Repères IREM*, *116*, 47-81.

Balacheff N. (1994) Didactique et intelligence artificielle. *Recherches en didactique des mathématiques, 14*(1), 9-42.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, 3(2), 115–138. https://doi.org/10.1007/s40751-017-0028-x

Bogdan, R., & Biklen, S. (1994). Investigação qualitativa em educação: uma introdução à teoria e aos métodos. Porto editora. http://177.20.147.23:8080/handle/123456789/1119

Bosch i Casabo, M. (1994). *La dimensión ostensiva en la actividad matemática. El caso de la proporcionalidad* [tesis de Doctorado, Universidad Autónoma de Barcelona]. https://www.educacion.gob.es/teseo/imprimirFichaConsulta.do?idFicha=50023

Bosch, M., & Gascón, J. (2014). Introduction to the Anthropological Theory of the Didactic (ATD). *Networking of theories as a research practice in mathematics education*, 67-83.

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical thinking and learning*, *23*(2), 170-185.

Bressan, A., & Bogisic, B. (1996). Las regularidades: fuente de aprendizajes matemáticos. *Consejo Provincial de Educación Argentina, Fecha de Consulta (20/01/2015): http://www. gpdmatematica. org. ar/publicaciones/diseno_desarrollo/matematica3. pdf*.

Briant, N., & Bronner, A. (2015, October). Étude d'une transposition didactique de l'algorithmique au lycée: une pensée algorithmique comme un versant de la pensée mathématique. In *Actes du Colloque EMF2015–GT3* (pp. 231-246). https://bibnum.publimath.fr/ACF/ACF15115.pdf

Broley, L., Buteau, C., Modeste, S., Rafalska, M., & Stephens, M. (2023). Computational Thinking and Mathematics. In *Handbook of Digital Resources in Mathematics Education* (pp. 1-38). Cham: Springer International Publishing.

Bustamante, E. (2017). *Un modelo epistemológico de referencia asociado a las sucesiones en la educación básica regular del Perú* [Tesis de Maestría, Pontificia Universidad Católica del Perú]. https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/9884

Castro, E, Cañadas, M. C. & Molina, M. (2010). El razonamiento inductivo como generador de conocimiento matemático. *UNO, 54*, 55-67.

Cetina-Vázquez, M., & Cabañas-Sánchez, G. (2022). Estrategias de generalización de patrones y sus diferentes formas de uso en quinto grado. *Enseñanza de las Ciencias. Revista de investigación y experiencias didácticas*, *40*(1), 65-86. https://doi.org/10.5565/rev/ensciencias.3096

Castela, C. (2016). Cuando las praxeologías viajan de una institución a otra: una aproximación epistemológica del “boundary crossing”. *Educación Matemática, 28*(2), 9-29.

Chaachoua , H., & Bessot, A. (2019). La notion de variable dans le modèle praxéologique. *Educação Matemática Pesquisa (EMP), 21*(4), 234-247.

Chaachoua, H. (2020). T4TE: Un cadre de référence pour la formalisation et l'extension du modèle praxéologique T4TEL: A frame of reference for the formalization and extension of the praxeological. *Educação Matemática Pesquisa Revista do Programa de Estudos Pós-Graduados em Educação Matemática. 22*(4), 103-118. https://doi.org/10.23925/1983-3156.2020v22i4p103-118

Chaachoua, H., Crisci, R., & Tchounikine, P. (2022). Un modèle praxéologique de référence pour des praxéologies mixtes dans des tâches de programmation. *Pré-actes de CITAD7*, 361-372.

Chevallard, Y. (1999). L’analyse des pratiques enseignantes en théorie anthropologique du didactique. *Recherches en Didactique des Mathématiques, 19*(2), 221-266.

Chevallard, Y. (2001). Les mathématiques et le monde: dépasser «l'horreur instrumentale». *Quadrature*, *41*, 25-40. http://yves.chevallard.free.fr/spip/spip/IMG/pdf/YC_2000_-_Quadrature_2001.pdf

Chevallard, Y. (2006). Pasos hacia una nueva epistemología en la educación matemática. En M. Bosch (Ed.), Actas del Cuarto Congreso de la Sociedad Europea para la Investigación en Educación Matemática, CERME 4 (págs. 2130). FUNDEMI IQS-Universitat Ramon Llull.

Couderette, M. (2016). Enseignement de l'algorithmique en classe de seconde. Une introduction curriculaire problématique. *Annales de de didactique et de sciences cognitives*, *21*, 267–296.

Crisci, R. (2020). *Etude des conditions de viabilité d'une approche basée sur l'algorithmique et la programmation pour l'apprentissage de la division euclidienne à l'école primaire* [Tesis de Doctorado, Universidad de Grenoble-Alpes]. https://www.theses.fr/2020GRALM046

De la Hoz Ruiz, A., & Neira, R. H. (2022). Enseñanza de las Matemáticas a través del Uso de Scratch (Transversalidad STEM). *IE Comunicaciones: Revista Iberoamericana de Informática Educativa*, (36), 14-34.

Diago, P. D., del Olmo-Muñoz, J., González-Calero, J. A., & Arnau, D. (2022). Entornos tecnológicos para el desarrollo del pensamiento computacional y de la competencia en resolución de problemas. *Aportaciones al Desarrollo del Currículo Desde la Investigación en Educación Matemática; Blanco Nieto, LJ, Climent Rodríguez, N., González Astudillo, MT, Moreno Verdejo, A., Sánchez-Matamoros García, G., de Castro Hernández, C., Gestal, CJ, Eds*, 399-424.

Dreyfus, T (1991). Advanced mathematical thinking process. *Mathematics Education Library*, *11*, 25-41.

Elicer, R., Tamborg, A. L., Bråting, K., & Kilhamn, C. (2023). Comparing the Integration of Programming and Computational Thinking into Danish and Swedish Elementary Mathematics Curriculum Resources. *LUMAT: International Journal on Math, Science and Technology Education*, *11*(3), 77-102.

Haspekian et al. (2023). *Algebra Education and Digital Resources: A Long-Distance Relationship?.* En B. Pepin, G. Gueudet, J. Choppin (Eds.), *Handbook of Digital Resources in Mathematics Education* (pp.1-33). Springer International Publishing. https://link.springer.com/10.1007/978-3-030-95060-6_16-1

Hoyles, C., & Noss, R. (2015). Revisiting programming to enhance mathematics learning. Western University, Canada.

International Society for Technology in Education. (2016). *ISTE National Educational Technology Standards* (NETS). https://www.iste.org/iste-standards

Kaspary, D. (2020). *La noosphère, un lieu de tension pour le curriculum: étude didactique de la mise en place d'un système d'évaluation de manuels scolaires sur l'étude du champ additif à l'école primaire* [Doctoral dissertation, Université Grenoble Alpes, Universidade federal de Mato Grosso do Sul]. https://theses.hal.science/tel-03100691/

Kilhamn, C., Bråting, K., Helenius, O., & Mason, J. (2022). Variables in early algebra: exploring didactic potentials in programming activities. *ZDM–Mathematics Education, 54*(6), 1273-1288. https://doi.org/10.1007/s11858-022-01384-0

Macias, M. D. C., & Romo, A. (2013). *Metodología para el diseño de actividades didácticas basadas en la modelación matemática.* https://core.ac.uk/download/pdf/33251008.pdf

Mason, J., Graham, A. y Johnston-Wilder, S. (2005). Desarrollando el pensamiento en álgebra. SABIO.

Mason, J. (1996). Expressing generality and roots of Algebra. En N. Bednarz, C. Kieran y L. Lee (Eds.), *Approaches to Algebra. Perspectives for Research and Teaching* (pp.65-86). Kluwer Academic Publisher.

Miller, J. (2019*). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns*. ZDM Mathematics Education, 51(6), 915-927. https://doi.org/10.1007/s11858-019-01096-y

Ministerio de Educación (2020). Diseño Curricular Básico Nacional de la Formación Inicial Docente: programa de estudios Educación Secundaria, especialidad Matemática. Recuperado de https://hdl.handle.net/20.500.12799/6898

Ministerio de Educación (2016). Educación básica regular programa curricular de educación primaria. Recuperado de https://hdl.handle.net/20.500.12799/4549

Molina, Á. (2022). *Contribución del pensamiento computacional con scratch al proceso de enseñanza y aprendizaje de las matemáticas* [Tesis doctoral, Universidad de Córdoba]. https://helvia.uco.es/handle/10396/24462

Mulligan, J. y Mitchelmore, M. (2009). Awareness of pattern and structure in early mathematical development. *Mathematics Education Research Journal*, *21*(2), 33-49. https://doi.org/10.1007/BF03217544

Najarro, L. (2018). *Caracterización del modelo epistemológico dominante de la proporcionalidad en los textos de matemática de educación secundaria* [Tesis de Maestría, Pontificia Universidad Católica del Perú]. https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/13837

Nyman, R., Bråting, K., & Kilhamn, C. (2024). Can programming support mathematics learning? An analysis of Swedish lower secondary textbooks. *International Journal of Mathematical Education in Science and Technology*, 1-19.

Organisation for Economic Co-operation and Development [OECD] (2018). *PISA 2021 Mathematics Framework (draft).* *https://pisa.anep.edu.uy/sites/default/files/Recursos/Marcos%20conceptuales/2022-PISA-Uruguay-Marcos%20conceptuales-Marco%20matema%CC%81tica.pdf*

Papic, M., & Mulligan, J. (2007). The growth of early mathematical patterning: An intervention study. In *Mathematics Education Research Group of Australasia Conference (30th: 2007)* (pp. 591-600). Merga.

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365-376. https://doi.org/10.1016/j.compedu.2018.10.005

Oliveira, G. P. (2019). A elaboração do problema de pesquisa em Educação e Educação Matemática. *Pesquisa em Educação e Educação Matemática: um olhar sobre a metodologia. Curitiba: CRV*. https://www.researchgate.net/publication/349145257_A_elaboracao_do_problema_de_pesquisa_em_Educacao_e_Educacao_Matematica

Radford, L. (2014). El desarrollo progresivo del pensamiento algebraico incorporado temprano. *Revista de investigación de educación matemática, 26* (2), 257-277.

Rivera, F. (2013). *Teaching and Learning Patterns in School Mathematics. Psychological and Pedagogical Considerations*. Springer.

Rojas Suárez, C., & Sierra Delgado, T. Á. (2022). Un modelo epistemológico de referencia en torno a la determinación y construcción de sólidos para la enseñanza secundaria obligatoria. *Acta Scientiae (Canoas)*, *24*(8), 437-475. https://hdl.handle.net/20.500.14352/73141

Sáez-López, J. M., Sevillano-García, M. L., & Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: Educational use of mBot. *Educational Technology Research and Development, 67*(6), 1405–1425. https:// doi. org/ 10. 1007/s11423-019-09648-5

Strock, J.-M. et Artaud, M. (2019). Du logos des organisations algorithmiques dans l'enseignement secondaire. Educação Matemática Pesquisa: Revista do Programa de Estudos Pós-Graduados em Educação Matemática, 21.

Tchounikine, P. (2017). Initier les élèves à la pensée informatique et à la programmation avec Scratch. *Laboratoire d'informatique de Grenoble*, 1-36. https://pdfbib.com/pdf/0690-initier-les-eleves-a-la-programmation-avec-scratch.pdf

Uicab Ballote, G. R., Rojano Ceballos, M. T., & García Campos, M. (2022). Expresiones de generalización en escolares de 10 a 12 años durante la resolución de secuencias figurales-numéricas y numéricas. *Educación matemática*, *34*(1), 42-69. https://doi.org/10.24844/em3401.02

Utreras, E., Pontelli, E. (2020). Accessibility of Block-Based Introductory Programming Languages and a Tangible Programming Tool Prototype. In: Miesenberger, K., Manduchi, R., Covarrubias Rodriguez, M., Peňáz, P. (Eds). *Computers Helping People with Special Needs. ICCHP 2020. Lecture Notes in Computer Science* (pp.27-34). Springer, Cham. https://doi.org/10.1007/978-3-030-58796-3_4

Warren, E. (2005). Young Children's Ability to Generalise the Pattern Rule for Growing Patterns. *International Group for the Psychology of Mathematics Education*, *4*, 305-312.

Warren, E., & Cooper, T. (2006). Using repeating patterns to explore functional thinking. *Australian Primary Mathematics Classroom*, *11*(1), 9-14.

Wijns, N., Torbeyns, J., Bakker, M., De Smedt, B., & Verschaffel, L. (2019). Four-year olds' understanding of repeating and growing patterns and its association with early numerical ability. *Early Childhood Research Quarterly*, 49, 152-163. https://doi.org/10.1016/j.ecresq.2019.06.004

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Ye, H., Ng, O. L., & Leung, A. (2024). Examining mathematics teachers' creative actions in programming-based mathematical activities. *ZDM–Mathematics Education*, 56, 695–711. https://doi.org/10.1007/s11858-024-01579-7

Zapatera Llinares, A. (2018). Introducción del pensamiento algebraico mediante la generalización de patrones. Una secuencia de tareas para Educación Infantil y Primaria. *Números. Revista de didáctica de las matemáticas,* 97, 51-67. https://repositorioinstitucional.ceu.es/bitstream/10637/10482/4/Introduccion_Zapatera_NRDDDLM_2018.pdf

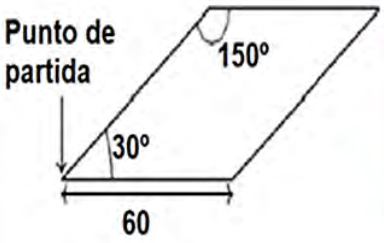
Zapatera Llinares, A. (2022). La generalización de patrones como herramienta para introducir el pensamiento algebraico en educación primaria. *Educación matemática*, *34*(2), 134-152. https://doi.org/10.24844/em3402.05.

Zazkis, R., & Liljedahl, P. (2002). Generalization of patterns: The tension between algebraic thinking and algebraic notation. *Educational studies in mathematics*, *49*, 379-402.

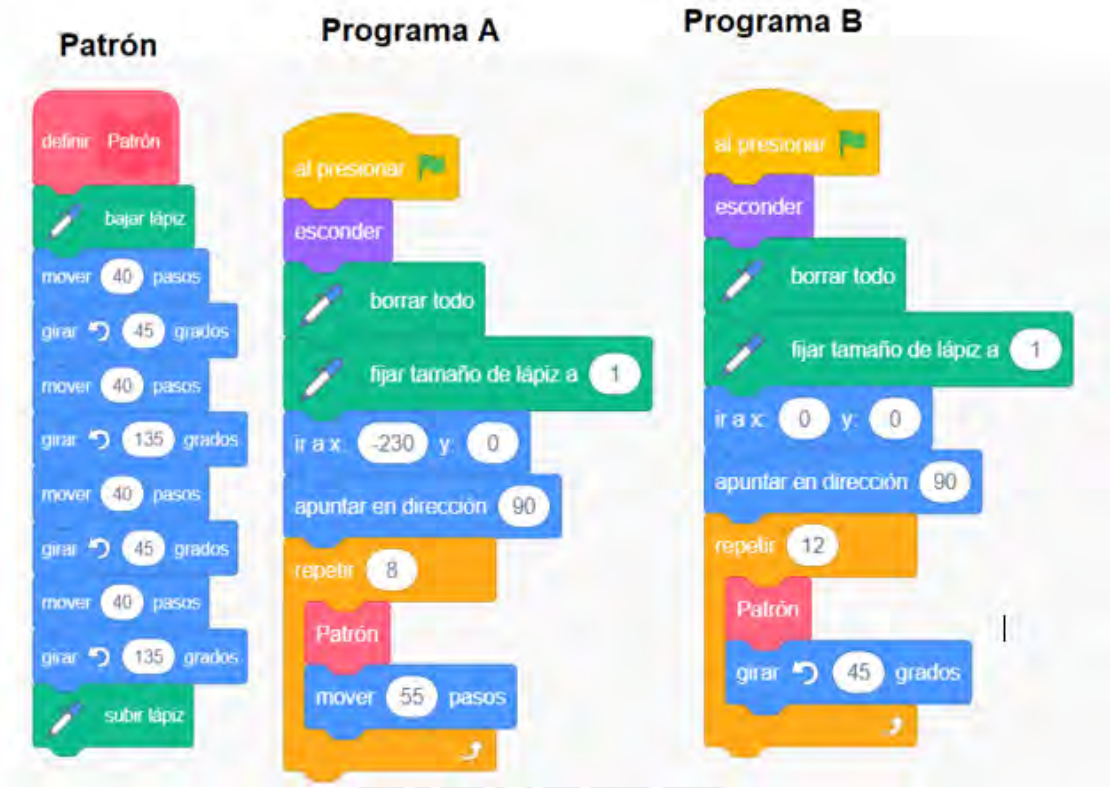
Apéndice

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_1	$T'_{prog}[T_1, 1]$	Poligonal cerrada	Patrón de repetición	constante	$\alpha < 90º$ $\alpha > 90º$ antihorario

1. Se desea trazar el patrón a continuación en forma de rombo. Completar en el anexo 1, el script del bloque Rombo para obtener este patrón.

El patrón Rombo	El bloque Rombo
	

1. Para realizar la figura anterior, se definió un patrón en forma de rombo y se utilizó uno de los dos programas A y B a continuación. Determina cuál de ellos y señala con un dibujo a mano alzada el resultado que se obtendría con el otro programa.



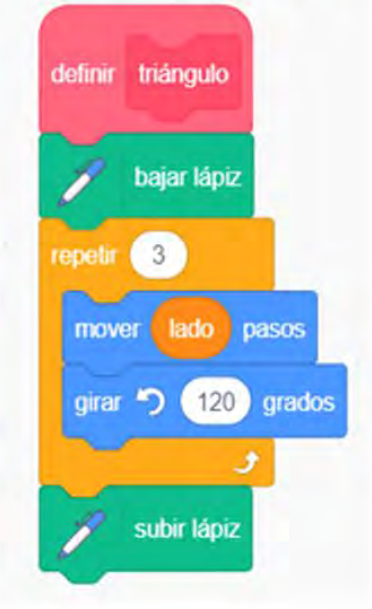
2. ¿Cuánto mide el espacio entre dos patrones sucesivos?

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_1	$T'_{prog}[T_1, 2]$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario

Se da el siguiente programa que permite trazar varios triángulos equiláteros de tamaños diferentes. Este programa contiene una variable llamada "lado". Las longitudes se dan en píxeles. Se recuerda que la instrucción "orientarse a 90" significa que se dirige hacia la derecha.

Script

bloque triángulo



3.

(a) ¿Cuál es la longitud (en píxeles) del lado del segundo triángulo trazado?

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_2	$T'_{prog}[T_2, 1]$	Poligonal abierta	Patrón recurrente creciente	creciente	$\alpha = 90º$ antihorario

Aquí hay tres figuras diferentes, ninguna está a la escala indicada en el ejercicio:

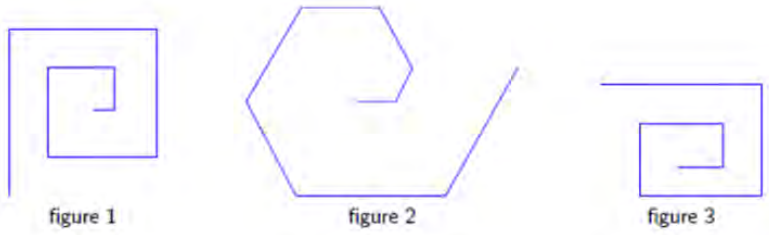


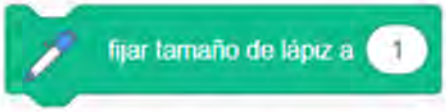
figure 1 figure 2 figure 3

El programa a continuación contiene una variable llamada "longitud".

3) ¿Qué modificación se debe realizar en el bloque "una vuelta " para obtener la figura 2 anterior?

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_3	$T'_{prog}[T_3, 1]$	Poligonal cerrada	Patrón recurrente creciente	constante	$\alpha < 90º$ $\alpha > 90º$ antihorario

3. Se desea realizar la figura a continuación:

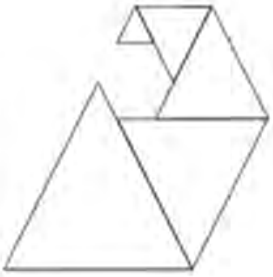
Para ello, se plantea insertar la instrucción 

en e programa utilizado en la pregunta 1. ¿Dónde se debe insertar esta instrucción?

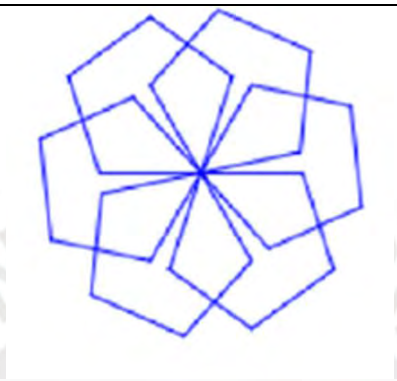
Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_3	$T'_{prog}[T_3, 2]$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario

4. Modificamos el script inicial para obtener la figura de al lado. Indica el número de una instrucción del script después de la cual se puede colocar la instrucción 

para obtener esta nueva figura.



Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_3	$T'_{prog}[T_3, 3]$	Poligonal cerrada	Patrón de repetición	constante	$\alpha > 90º$ horario

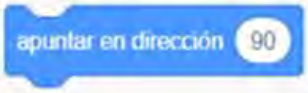


¿Qué necesitas modificar para dibujar doce octágonos regulares alrededor de un punto y para crear su simetría rotacional?

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_4	$T'_{prog}[T_4, 1]$	Poligonal cerrada	Patrón de repetición	constante	$\alpha < 90º$ $\alpha > 90º$ antihorario

2. Se desea realizar la figura a continuación, construida a partir del bloque Rombo completado en la pregunta 1

Recuerde que la instrucción

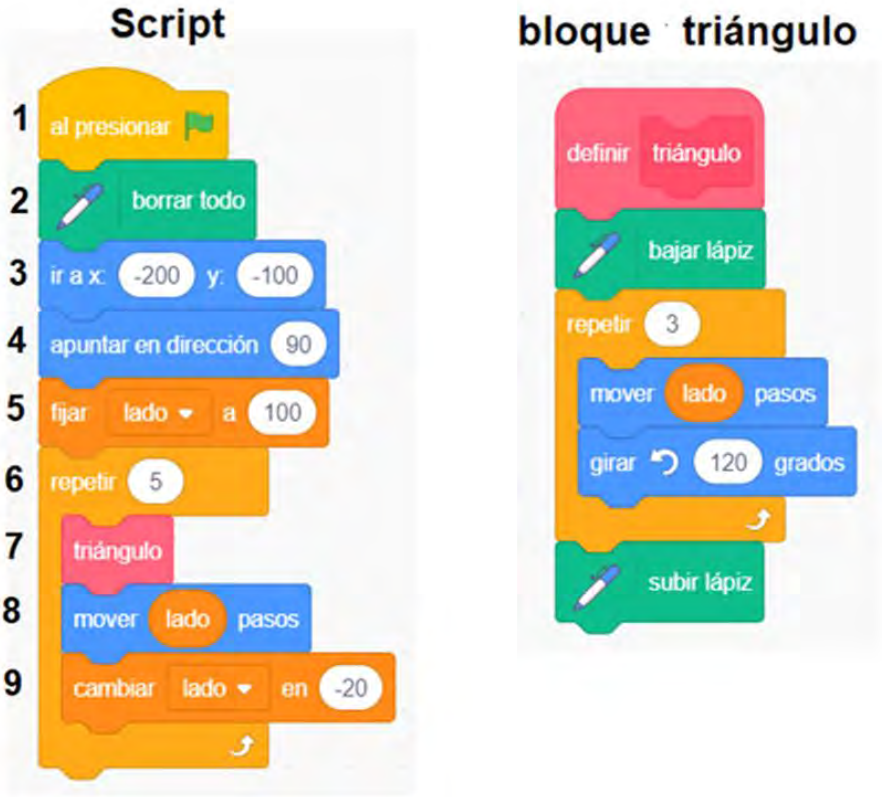


significa que está a la derecha.

Entre las instrucciones a continuación, indique en su copia, en el orden, las dos instrucciones que deben colocarse en el bucle a la derecha para finalizar el script a la derecha.

Tipo de tarea	Subtipo de tarea	$V_{prog}[tipo\ de\ figura]$	$V_{prog}[tipo\ de\ patrón]$	$V_{prog}[cant.\ de\ pasos]$	$V_{prog}[\measuredangle\ de\ giro]$
T_4	$T'_{prog}[T_4, 2]$	Poligonal cerrada	Patrón recurrente decreciente	constante	$\alpha > 90º$ antihorario

Se da el siguiente programa que permite trazar varios triángulos equiláteros de tamaños diferentes. Este programa contiene una variable llamada "lado". Las longitudes se dan en píxeles. Se recuerda que la instrucción "orientarse a 90" significa que se dirige hacia la derecha.



(b) Dibuja a mano alzada cómo sería la figura obtenida al ejecutar este script.