UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

ESCUELA DE POSGRADO

Influencia del lenguaje de programación Scratch en el desarrollo del Pensamiento Computacional en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, 2018

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN GESTIÓN E INNOVACIÓN EDUCATIVA

AUTORA
Karol Soto Ccoicca

ASESOR
William Jesús Rojas Gutiérrez

Lima, Perú

2020

METADATOS COMPLEMENTARIOS

Datos del autor

Nombres	Karol
Apellidos	Soto Ccoicca
Tipo de documento de identidad	DNI
Número del documento de identidad	40536407
Número de Orcid (opcional)	0000-0001-6165-1171

Datos del asesor

Nombres	William Jesús
Apellidos	Rojas Gutiérrez
Tipo de documento de identidad	DNI
Número del documento de identidad	40021221
Número de Orcid (obligatorio)	0000-0001-5296-2971

Datos del Jurado

Datos del presidente del jurado

Nombres	William Jesús
Apellidos	Rojas Gutiérrez
Tipo de documento de identidad	DNI
Número del documento de identidad	40021221

Datos del segundo miembro

Nombres	Guisella Ivonne
Apellidos	Azcona Ávalos
Tipo de documento de identidad	DNI
Número del documento de identidad	43991476

Datos del tercer miembro

Nombres	Rosa Mercedes
Apellidos	Cabrera Rondoy
Tipo de documento de identidad	DNI
Número del documento de identidad	09437845

Datos de la obra

Materia	Pensamiento computacional, Lenguaje de Programación SCRATCH, Conceptos computacionales, Prácticas computacionales.
Campo del conocimiento OCDEConsultar el listado: enlace	https://purl.org/pe-repo/ocde/ford#5.03.00
Idioma (Normal ISO 639-3)	SPA - español
Tipo de trabajo de investigación	Tesis
País de publicación	PE - PERÚ
Recurso del cual forma parte (opcional)	
Nombre del grado	Maestro en Gestión e Innovación Educativa
Grado académico o título profesional	Maestro
Nombre del programa	Maestría en Gestión e Innovación Educativa
Código del programa Consultar el listado: enlace	191147

*Ingresar las palabras clave o términos del lenguaje natural (no controladas por un vocabulario o tesauro).

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE
ESCUELA DE POSTGRADO
UNIDAD DE CIENCIAS DE LA EDUCACIÓN Y HUMANIDADES

ACTO DE SUSTENTACIÓN PÚBLICA *ONLINE* DE TESIS DE LA MAESTRÍA EN GESTIÓN E INNOVACIÓN EDUCATIVA

ACTA Nº 002

Hoy, **22** de **enero** de **2021** a las **15:00** horas, mediante sesión en línea a través de la Plataforma ZOOM, debidamente licenciada por la Escuela de Postgrado de la Universidad Católica Sedes Sapientiae,

KAROL SOTO CCOICCA

llevó a cabo el Acto de Sustentación Pública *Online* de su tesis titulada:

Influencia del Lenguaje de Programación Scratch en el desarrollo del Pensamiento Computacional en estudiantes de primer ciclo de laCarrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, 2018

frente al jurado conformado por:

PRESIDENTE	: William Jesús Rojas Gutiérrez
SEGUNDO MIEMBRO	: Guisella Ivonne Azcona Ávalos
TERCER MIEMBRO	: Rosa Mercedes Cabrera Rondoy

Finalizada la presentación, defendió su tesis durante 30 minutos ante el jurado y el público, respondiendo a satisfacción las preguntas planteadas; al concluir el acto y posterior a la deliberación respectiva, el jurado decidió otorgarle por **CONSENSO** la mención **CUM LAUDE**, con una calificación de **17 (diecisiete)** puntos sobre **20 (veinte).**

Presidente

Segundo Miembro

Tercer Miembro

Anexo 2

CARTA DE CONFORMIDAD DEL ASESOR(A) DE TESIS

Ciudad, 4 de abril de 2023

Señor(a),
Cristy Lourdes Ballesteros Molina
Jefe del Departamento de Investigación
Escuela de Post Grado UCSS

Reciba un cordial saludo.

Sirva el presente para informar que la tesis, bajo mi asesoría, con título , Influencia del Lenguaje de Programación Scratch en el desarrollo del Pensamiento Computacional en estudiantes de primer ciclo de la Facultad de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, 2018, presentado por Karol Soto Ccoicca (código de estudiante: 2016200525 y DNI:40536407) para optar el grado académico de Maestro, ha sido revisado en su totalidad por mi persona y CONSIDERO que el mismo se encuentra APTO para ser sustentado ante el Jurado Evaluador.

Asimismo, para garantizar la originalidad del documento en mención, se le ha sometido a los mecanismos de control y procedimientos antiplagio previstos en la normativa interna de la Universidad, **cuyo resultado alcanzó un porcentaje de similitud de 10%.** * Por tanto, en mi condición de asesor(a), firmo la presente carta en señal de conformidad y adjunto el informe de similitud del Sistema Antiplagio Turnitin, como evidencia de lo informado.

Sin otro particular, me despido de usted. Atentamente,

Firma del Asesor (a)
DNI N°: 40021221
ORCID: 0000-0001-5296-2971
Unidad de Ciencias de la Educación y Humanidades
Unidad de Post Grado - UCSS

* De conformidad con el artículo 8°, del Capítulo 3 del Reglamento de Control Antiplagio e Integridad Académica para trabajos para optar grados y títulos, aplicación del software antiplagio en la UCSS, se establece lo siguiente:

Artículo 8°. Criterios de evaluación de originalidad de los trabajos y aplicación de filtros
El porcentaje de similitud aceptado en el informe del software antiplagio para trabajos para optar grados académicos y títulos profesionales, será máximo de veinte por ciento (20%) de su contenido, siempre y cuando no implique copia o indicio de copia.

Dedicatoria

Dedico este trabajo, primeramente, a Dios, que siempre me cuida y guía por el buen sendero y a toda mi querida familia, porque gracias al apoyo incondicional, a la comprensión y al amor ayudaron a culminar esta ardua investigación.

"Si se siembra la semilla con fe y se cuida con perseverancia, sólo será cuestión de tiempo recoger sus frutos".

(Thomas Carlyle)

Agradecimientos

A Dios nuestro Señor, por su inmenso amor y misericordia que nos profesa día a día, que hace de nosotros, personas con vocación de servicio a nuestro prójimo.

Gracias al Rector de la Universidad Católica Sedes Sapientiae, doctor Gian Battista Fausto Bolis, por haberme permitido realizar el trabajo experimental en la casa de estudios que dirige.

Gracias al Decano de la Facultad de Ingeniería, el ingeniero José Pérez Fernández, por haber confiado en mi persona, para llevar a cabo dicha investigación aplicándola a los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas.

Gracias a mi profesor, el maestro William Jesús Rojas Gutiérrez, por orientarnos en el arduo camino de la investigación, lleno de aprendizajes y satisfacciones, que se ve reflejado en esta tesis.

Gracias a todos mis estudiantes que colaboraron en esta investigación, sin tener conocimiento de que eran parte de un trabajo experimental.

Gracias a mi familia y amigos, por su comprensión y paciencia, en los momentos en los que estuve ausente en sus vidas.

Resumen

El presente trabajo de investigación tuvo como propósito determinar el nivel de influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional en los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

El estudio realizado tuvo un enfoque cuantitativo, de tipo aplicado y de diseño preexperimental. Se utilizó como instrumento un pretest, que se aplicó al inicio de la investigación, a fin de evaluar el nivel del pensamiento computacional de los estudiantes. Luego se aplicó un postest, tras impartir un estímulo de siete (07) sesiones de clases del lenguaje de programación SCRATCH para medir los avances logrados.

Los resultados demostraron que existen diferencias significativas entre el pretest y el postest, al lograr que se incremente de 11% a 50% de estudiantes en el nivel logrado, que se reduzca de 72% a 48% en el nivel proceso; quedando solo un 2%, en el nivel inicio. En consecuencia, se concluye que la aplicación del programa SCRATCH influye favorablemente en el desarrollo del pensamiento computacional.

Palabras clave: Pensamiento computacional, lenguaje de programación SCRATCH, conceptos computacionales, prácticas computacionales.

Abstract

The purpose of this research was to determine the level of influence of the SCRATCH programming language in the development of computational thinking in the first cycle students of the Systems Engineering degree at the Sedes Sapientiae Catholic University.

The study has a quantitative approach, applied type and preexperimental design. A pretest was used as an instrument, which was applied at the beginning of the investigation in order to assess the level of computational thinking of the students. A postest was then applied after giving a stimulus of seven (07) sessions of the SCRATCH programming language classes to measure the progress.

The results show that there are significant differences between the pretest and the posttest, achieving that it increases from 11% to 50% of students in the level *achieved*, a reduction from 72% to 48% in the *process* level, leaving only 2% in the *start* level. Therefore, it is concluded that the application of the SCRATCH programming language influences positively the development of computational thinking.

Keywords: computational thinking, SCRATCH programming language, computational concepts, computer practices.

Índice

Índice de figuras

Índice de tablas

Introducción

La presente investigación tuvo por finalidad desarrollar el Pensamiento Computacional en los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, al utilizar el lenguaje de programación SCRATCH para la adquisición de las habilidades del pensamiento crítico, algorítmico, creativo, lógico y abstracto, necesarias para el análisis y la resolución de problemas, dentro de cualquier contexto.

Para el desarrollo de la investigación se utilizaron métodos y técnicas científicas, así como también, se contó con la colaboración de tratadistas e investigadores. Finalmente, se recurrió a experiencias reales desarrolladas en algunas partes del mundo y de nuestro país.

En la primera parte, se plantea la problemática, estableciéndose los parámetros de la investigación, se delimitan las acciones y los alcances para lo cual se ha planteado el problema principal y los problemas secundarios.

En la segunda parte, se presenta el marco teórico, el cual comprende los estudios e investigaciones nacionales e internacionales, referidos a la temática de la investigación, se plantea la base teórica que sustenta los problemas de investigación y se exponen las hipótesis planteadas.

En la tercera parte, se enfoca la metodología de la investigación, se sustenta el tipo de método, diseño, enfoque, población y muestra. Del mismo modo, se presentan definiciones conceptuales y operacionales de las variables a tratar, a través de los sustentos técnicos y el instrumento del caso. Finalmente, se hace mención a las delimitaciones de la investigación en los ámbitos temático, espacial y temporal; así como, las limitaciones del presente estudio.

En la cuarta parte, se realiza el análisis de los resultados obtenidos de la recolección de datos de los estudiantes, al utilizar herramientas estadísticas para la generación de reportes, tablas y figuras que aporten a la interpretación de dichos resultados.

En la quinta parte, se realiza la discusión de los resultados obtenidos, mediante las pruebas de hipótesis planteadas en la investigación, la contrastación con los antecedentes planteados en la base teórica; así como, las conclusiones y recomendaciones.

Capítulo I

El problema de la investigación

1.1. Planteamiento del problema

Las preguntas que se hacen, desde hace algún tiempo atrás, en los centros educativos como las escuelas, colegios, institutos y universidades son ¿cuáles son las competencias, destrezas, capacidades y habilidades, que necesitan obtener los estudiantes a través de la educación en la actualidad?, y ¿por qué la computación es fundamental como parte del proceso formativo de las personas en la sociedad actual?

Se han realizado muchas propuestas, pero sin duda, una de las que está tomando más importancia es la que ha puesto en evidencia Wing, al afirmar que el: "pensamiento computacional implica la resolución de problemas, el diseño de sistemas, y la comprensión de la conducta humana, haciendo uso de los conceptos fundamentales de la computación" (2006, p. 33). En dicho artículo, Wing también indica que el Pensamiento Computacional "representa un conjunto de habilidades y actitudes, aplicable universalmente que toda persona (no sólo científico de la computación) debería estar ansiosa por adquirir y usar" (2006, p. 34).

De esta forma, se puede decir que Wing define que lo esencial del Pensamiento Computacional es pensar qué y cómo haría un informático frente a una situación problemática, siendo una habilidad fundamental para cualquier persona que interactúe en el mundo digital, y que este pensamiento debería añadirse a la currícula básica del desarrollo de habilidades analíticas de cualquier niño.

Las experiencias educativas acerca del Pensamiento Computacional están localizadas a nivel internacional; sin embargo, para no ir muy lejos, se analizarán los trabajos realizados en

América Latina. Uno de ellos es el caso de Chile, donde existe una experiencia realizada por Hitschfeld et al. (2015), en la cual manifiestan que países como Finlandia, Inglaterra, Estonia, Estados Unidos, Singapur y Japón se enfocaron en el desarrollo del Pensamiento Computacional, en el nivel escolar, para liderar la revolución digital mundial, por ello, realizaron un análisis a los escolares. Chile es uno de los países que tiene mejor infraestructura tecnológica a nivel escolar de Latinoamérica, en base al análisis del Banco Interamericano de Desarrollo, nueve de cada diez centros educacionales cuentan con salas de cómputo, y esta infraestructura no está siendo aprovechada y, por tanto, expresan su preocupación al resaltar la necesidad de dar un salto en la educación, a fin de que los estudiantes utilicen la tecnología para convertirse en generadores de tecnología y no solo en consumidores. Ellos ven la programación como una fuente de conocimiento que hace posible realizar estos cambios, especialmente, si se empieza a trabajar con los niños sin esperar a que estos lleguen al nivel universitario.

Según Hitschfeld et al., se potencia la "confianza en el manejo de la complejidad, persistencia al trabajar con problemas difíciles, tolerancia a la ambigüedad, habilidad para lidiar con problemas abiertos, y habilidad para comunicarse y trabajar con otros para alcanzar una meta o solución común" (2015, pp. 31-32).

Usman (2013), en su tesis titulada *Aplicación de entornos elaborados con herramientas digitales gráficas animadas, para el desarrollo y fortalecimiento de habilidades de pensamiento de orden superior en el área de matemáticas de una institución educativa de la ciudad de Palmira*, plantea las siguientes interrogantes: ¿qué y cómo se enseña?, ¿para qué se enseña? y ¿cómo se califican los aprendizajes en los que intervienen estas tecnologías?, para ello, propone como recurso didáctico el uso del lenguaje de programación SCRATCH para diseñar objetos de aprendizaje, definiendo un conjunto de instrucciones secuenciales y estructuras de control

selectivas y repetitivas, mediante el uso de conceptos computacionales, así como: eventos, variables, secuencias, condicionales, ciclos o bucles y paralelismo, continuamente las interrogantes que se presenten se deben responder por ensayo y error; con el fin de solucionar problemas por medio de la creación de aplicaciones como historias interactivas, juegos y animaciones o simplemente transmitir alguna información sobre diversos temas matemáticos, los que luego se pueden compartir con otros usuarios en la Web, con el objetivo de elaborar insumos para las temáticas del curso de matemática, cuya función será servir al conocimiento y a la educación.

Blanco (2014), en su tesis titulada *Implementación de SCRATCH para potenciar el aprendizaje significativo a través de la lógica de programación en los estudiantes de Nivel Básica Secundaria*, analiza la actual sociedad de la información, en donde se requiere que los estudiantes aprendan a ser competentes y competitivos, ante las exigencias del mundo de hoy.

Existe una gran variedad de herramientas tecnológicas que pueden ser integradas al currículo para obtener una educación de calidad, surge el lenguaje de programación SCRATCH como una herramienta alternativa para el aprendizaje de lógica de programación. En esta tesis, se pretendió responder a la pregunta de investigación: ¿En qué forma el uso del lenguaje de programación SCRATCH estimula el aprendizaje significativo de lógica de programación en los alumnos de Educación Básica Secundaria?, para lo cual, se aplicó el instrumento de medición ficha de observación no participante, pretest y postest, tanto al grupo experimental como al grupo de control, el cual facilitó evidenciar que el programa educativo SCRATCH favorece el aprendizaje significativo de lógica de programación a través de la elaboración de aplicaciones que facilitaron la adquisición y fortalecimiento de conceptos fundamentales de programación. Asimismo, la

motivación como estrategia metodológica aplicada por el docente en las sesiones de clase, generaron espacios dinámicos que favorecieron el aprendizaje significativo.

En referencia a las experiencias nacionales, son pocas en comparación al ámbito internacional; sin embargo, tenemos:

Chancolla y Pacori (2017), en su tesis titulada *El uso del software SCRATCH para mejorar el pensamiento computacional en los estudiantes del quinto grado de primaria de la Institución Educativa N° 40009 San Martín de Porres del Distrito de Paucarpata, Arequipa, 2016*, sostiene que el uso del programa SCRATCH en las sesiones de aprendizaje desarrolla el pensamiento computacional en los estudiantes de quinto grado de primaria.

Las técnicas de recolección de datos fueron la observación y la evaluación, el instrumento de observación fue la lista de cotejo, y el de evaluación, la prueba de pretest y postest. Se trabajó con una muestra de 43 estudiantes de quinto grado. Por consiguiente, con esta muestra se procedió a analizar y a determinar el desarrollo del Pensamiento Computacional, para concluir si hubo una mejora significativa entre estas dos evaluaciones.

También Condo (2017), en su tesis titulada *El pensamiento computacional en estudiantes del VII ciclo de la institución educativa particular Ricardo Palma, San Juan de Miraflores, 2016,* se consideró como objetivo determinar el nivel de desarrollo del Pensamiento Computacional en los estudiantes del VII ciclo. La metodología de investigación que se utilizó fue el método descriptivo simple, del tipo básico y con diseño no experimental, con una variable de enfoque cuantitativa. Se trabajó con una muestra de 30 alumnos del VII ciclo y los instrumentos de evaluación utilizados fueron el cuestionario y las fichas prácticas.

El Ministerio de Educación, a través de PerúEduca, plataforma digital que ofrece cursos de capacitación a la comunidad educativa, actualmente, cuenta con más de un millón de miembros

que aprenden y comparten conocimientos (http://www.perueduca.pe/). En ella, hay una sección dedicada al uso del software SCRATCH, que incentiva a los docentes y a los estudiantes, en general, al uso de esta herramienta para desarrollar el Pensamiento Computacional.

El problema de la investigación se enfoca en las dificultades que tuvieron los estudiantes de la carrera de Ingeniería de Sistemas del II ciclo del curso de Algorítmica 1, para analizar, diseñar y plantear una posible solución a un problema presentado y, más aún, si se le incrementaba cierto grado de complejidad, debido al insuficiente desarrollo del Pensamiento Computacional, el cual es un proceso mental para analizar y resolver problemas de cualquier contexto, es una competencia necesaria del siglo XXI, que debe ser adquirida por todos los estudiantes y comprende las habilidades como el pensamiento algorítmico, lógico, abstracto, creativo y crítico. Según Wing (2006), se debe pensar como un científico informático para resolver un problema de cualquier contexto.

Por lo tanto, la carencia del desarrollo de esta competencia en los estudiantes del curso de Algorítmica 1, conlleva a que ellos presenten dificultades en el desarrollo del curso llegando a desaprobar la materia o desertar de la carrera de Ingeniería de Sistemas.

Como solución, se plantea incluir en el primer ciclo de la carrera de Ingeniería de Sistemas, el desarrollo del Pensamiento Computacional a través del lenguaje de programación Scratch, programa visual compuesto por bloques, según Resnick (2013), con el lenguaje de programación Scratch no sólo se aprende a programar, simultáneamente se aprenden estrategias para solucionar problemas de cualquier contexto, diseñar proyectos y comunicar ideas.

En el presente trabajo, se muestra la valoración del lenguaje de programación “SCRATCH” como herramienta de enseñanza/aprendizaje para el desarrollo del Pensamiento Computacional y,

por lo tanto, ayudar a reducir el número de estudiantes que se ven obligados a llevar nuevamente el curso de Algorítmica 1, y también disminuir la continuidad de estudiantes desertores.

Tabla 1

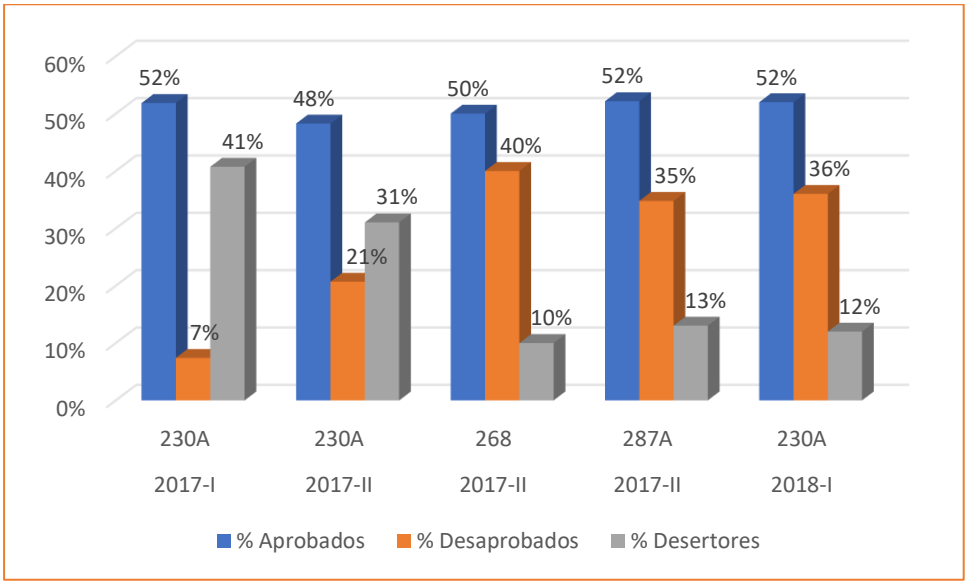
Frecuencias de la situación de los estudiantes del curso de Algorítmica 1

Semestre	Sección	Estudiantes	Aprobados	Desaprobados	Desertores	% Aprobados	% Desaprobados	% Desertores
2017-I	230A	27	14	2	11	52%	7%	41%
2017-II	230A	29	14	6	9	48%	21%	31%
2017-II	268	10	5	4	1	50%	40%	10%
2017-II	287A	23	12	8	3	52%	35%	13%
2018-I	230A	25	13	9	3	52%	36%	12%

Nota: Datos obtenidos de la DAIA–Dirección de Atención Integral al alumno de la Universidad Católica Sedes Sapientiae en los años 2017 y 2018.

Figura 1

Resultados del curso de Algorítmica 1



Nota: DAIA. Resultados del curso de Algorítmica 1 en los años 2017 y 2018.

Según la tabla 1, se observa que el porcentaje de estudiantes aprobados al siguiente semestre va en el rango de 48% al 52%, interpretándose que, casi solo la mitad de los alumnos logran ser promovidos. Otro dato importante es que la tasa de estudiantes que abandonan la carrera, por cambio a otra especialidad o incluso a otra universidad, es recurrente.

Además, se observa que el número de estudiantes que desaprobaron la asignatura se encuentra en el rango del 7% al 40%, siendo esta, una tasa muy alta. Es decir, los estudiantes que durante el semestre no desarrollaron las habilidades que se obtienen con la programación, no superaron estándares para ser promovidos.

Finalmente, se puede indicar que desarrollar el Pensamiento Computacional es una ventaja evidente y las habilidades mencionadas que se adquieren, respecto al uso de las tecnologías, son necesarios y tienen un alcance mayor que el entretenimiento. En la actualidad, resultan esenciales

para profesionales de hoy, ya que permite la evolución de una sociedad formada por consumidores de tecnología, en una de potenciales desarrolladores de esta.

1.2. Formulación del problema

1.2.1. Pregunta general

¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?

1.2.2. Preguntas específicas

¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del Pensamiento Computacional según la dimensión conceptos computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?

¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del Pensamiento Computacional según la dimensión prácticas computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?

1.3. Justificación del tema de la investigación

La presente investigación tiene como propósito el desarrollo del Pensamiento Computacional en los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas que se obtendría con el uso del lenguaje de programación SCRATCH, al permitir a los estudiantes aprender a programar fácilmente y dar soluciones a los problemas que se les plantee.

1.3.1. Justificación teórica

Ser una investigación que promueva el interés y la puesta en práctica de las habilidades que desarrolla el Pensamiento Computacional, tales como el pensamiento algorítmico, crítico, lógico, abstracto, creativo; con el fin de que se apliquen todos estos conocimientos para el análisis

y la resolución de problemas que se susciten en el medio donde el estudiante se desenvuelva, ya sea social, académica o profesionalmente.

1.3.2. Justificación metodológica

Esta investigación hace uso de una metodología de tipo aplicada y de alcance explicativo, ya que se busca explicar el nivel de influencia de la variable independiente (SCRATCH) sobre la variable dependiente (Pensamiento Computacional). Para ello, se aplicó una evaluación de pretest al inicio del experimento con el fin de conocer la situación actual de los estudiantes, tras impartir un estímulo de siete (07) sesiones de clases acerca del lenguaje de programación SCRATCH, se compararon los resultados iniciales con los resultados obtenidos al final del experimento, los cuales se obtuvieron a través de una evaluación de postest.

1.3.3. Justificación práctica

Se promueve el desarrollo del Pensamiento Computacional, a través de las siete (07) sesiones de clases del lenguaje de programación SCRATCH, los conocimientos obtenidos y las habilidades desarrolladas servirán como base para el dominio de los demás cursos de programación de mayor complejidad, que se encuentran en el plan de estudios de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

1.3.4. Justificación social

Desarrollando el Pensamiento Computacional en los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas, se logrará que tengan las habilidades para formular y resolver problemas académicos, personales y de cualquier contexto social.

1.4. Objetivos de la investigación

1.4.1. Objetivo general

Determinar la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

1.4.2. Objetivos específicos

Determinar la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, según la dimensión conceptos computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Determinar la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, según la dimensión prácticas computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Capítulo II

Marco teórico

2.1 Antecedentes del estudio

2.1.1 Internacionales

Román (2016), en su tesis titulada *Codigoalfabetización y pensamiento computacional en educación primaria y secundaria: validación de un instrumento y evaluación de programas*, tesis doctoral desarrollada en la Universidad Nacional de Educación a Distancia, España, tuvo como objetivo en su segundo estudio, la construcción y validación de un *Test de Pensamiento Computacional*. La metodología utilizada en la investigación fue de tipo aplicada con un enfoque cuantitativo de nivel descriptivo y diseño cuasiexperimental sobre una población de todos los estudiantes de 1er, 2do y 3er año de la Escuela Secundaria Obligatoria de España; como muestra se tomó a 1251 estudiantes de 24 centros educativos distintos, el instrumento utilizado fue un cuestionario de 28 preguntas. De los resultados obtenidos, el autor evidenció que se construyó y validó el Test de Pensamiento Computacional, al obtener resultados en promedio de 16.38 puntos, ligeramente superior al punto medio que equivale a 14 puntos de 28.

Pérez (2017), en su tesis titulada *Uso de SCRATCH como herramienta para el desarrollo del pensamiento computacional en Programación I de la carrera de Informática de la Universidad Central del Ecuador*, tesis doctoral desarrollada en la Universidad de Alicante, España, tuvo como objetivo determinar el incremento del Pensamiento Computacional en los alumnos del primer ciclo de la especialidad de Informática de la Facultad de Filosofía de la Universidad Central del Ecuador usando el programa SCRATCH como herramienta didáctica en su formación profesional.

La metodología utilizada en la investigación fue de tipo aplicada con un enfoque cuantitativo de nivel explicativo y diseño cuasiexperimental, sobre una población de 80 estudiantes

del primer semestre abril-septiembre del 2015, de la carrera de Informática, empleó el 100% de su población como muestra; el instrumento utilizado fue un cuestionario de pretest y postest compuesto de 16 preguntas. En base a los resultados, el autor demostró que no existió un crecimiento importante en todas las dimensiones; sin embargo, se mostraron mejoras en la dimensión de reconocimiento de patrones.

Blanco (2014), en *Implementación de SCRATCH para potenciar el aprendizaje significativo a través de la lógica de programación en los estudiantes de Nivel Básica Secundaria*, tesis de maestría desarrollada en el Instituto Tecnológico y de Estudios Superiores de Monterrey, México, tuvo como objetivo aplicar el software educativo SCRATCH para estimular el aprendizaje significativo de lógica de programación en los estudiantes de Educación Básica Secundaria.

La metodología utilizada en la investigación es de tipo aplicada con un enfoque cuantitativo de nivel descriptivo y diseño cuasiexperimental empleada sobre una población de 60 estudiantes y 2 profesores de la asignatura de Tecnología e Informática, utilizando el 100% de su población como muestra, el instrumento utilizado es un cuestionario de pretest y postest de 9 preguntas. De los resultados obtenidos, se evidenció que la motivación en el grupo experimental aumentó en 70%, al obtener un 73%; mientras que, se redujo la percepción de repetitividad en las clases en 44% y obtuvo 0% al final de la investigación, difiriendo de los resultados del grupo de control.

Cearreta (2015), en su tesis de maestría titulada *SCRATCH como recurso didáctico para el desarrollo del Pensamiento Computacional de los alumnos de Secundaria y Bachillerato en la asignatura de Informática y como recurso transversal en el resto de asignaturas*, desarrollada en la Universidad Internacional de la Rioja, España, tuvo como objetivo realizar un estudio para determinar si el uso de SCRATCH favorece el desarrollo del Pensamiento Computacional en los alumnos de Bachillerato. La metodología empleada en la investigación es de tipo básica con un

enfoque cualitativo de nivel exploratorio y diseño transeccional sobre una población de 10 estudiantes del 1° de Bachillerato. Las técnicas e instrumentos utilizados son la observación y el cuestionario, este último de 20 preguntas para los alumnos y 18 preguntas para la docente. De los resultados obtenidos, la investigadora concluyó que los estudiantes en una escala de 1 a 5 obtuvieron, en promedio, en conceptos computacionales un puntaje de 4.8, en prácticas computacionales un promedio de 3.4 y en perspectivas computacionales un promedio de 4.4.

Usman (2013), en *Aplicación de entornos elaborados con herramientas digitales gráficas animadas, para el desarrollo y fortalecimiento de habilidades de pensamiento de orden superior en el área de matemáticas de una institución educativa de la ciudad de Palmira*, tesis de maestría desarrollada en la Universidad Nacional de Colombia, cuya finalidad fue elaborar un objeto de aprendizaje usando el programa SCRATCH para mejorar la enseñanza de las matemáticas, según el plan pedagógico del centro educativo Nuestra Señora del Palmar.

La metodología utilizada en la investigación es de tipo básica con un enfoque cualitativo de nivel exploratorio y diseño transeccional sobre una población de 35 estudiantes, utilizando 32 estudiantes como muestra, las técnicas utilizadas son la observación y una encuesta de 7 preguntas. De los resultados obtenidos se concluyó que, los estudiantes mejoraron su competencia comunicativa en matemática razonando de forma creativa, sistemática y sobre todo trabajando colaborativamente.

2.1.2 Nacionales

Vera (2018), en su tesis titulada *Influencia de la capacitación en Algorítmica en el nivel de pensamiento computacional de los estudiantes de la Universidad Global del Cusco,* tesis doctoral desarrollada en la Universidad Global del Cusco, Perú, tuvo como objetivo establecer la

influencia de la enseñanza del curso de Algorítmica en el nivel de Pensamiento Computacional en los estudiantes del primer ciclo de la institución, en 2018.

La metodología utilizada fue de tipo aplicada con un enfoque cuantitativo de nivel explicativo y diseño cuasiexperimental sobre una población del 100% de los estudiantes del primer ciclo de las especialidades de Administración de Negocios Globales e Ingeniería en Tecnología de Información y Comunicación, el instrumento utilizado fue un cuestionario de opción múltiple y respuestas únicas de 10 preguntas. De los resultados obtenidos, el autor verificó que, tras recibir la capacitación, los estudiantes de Ingeniería en Tecnología de Información y Comunicación aumentaron la media del nivel de Pensamiento Computacional en 1.2 puntos y los de Administración de Negocios Globales aumentaron en 1.0 punto, quienes fueron el grupo de control.

Cori (2017), en *Aplicación de una herramienta de programación para mejorar el pensamiento computacional en estudiantes universitarios de Tacna*, tesis doctoral desarrollada en la Universidad Nacional Jorge Basadre Grohmann de Tacna, Perú, tuvo como objetivo mejorar el Pensamiento Computacional, al capacitar en un lenguaje de programación a los jóvenes universitarios de 16 a 18 años de edad en la ciudad de Tacna.

La metodología utilizada en la investigación es de tipo aplicada con un enfoque cuantitativo de nivel descriptivo y diseño preexperimental sobre una población de 250 estudiantes de la Escuela Profesional de Ingeniería en Informática y Sistemas, empleando 68 estudiantes como muestra, el instrumento de medición utilizado es un cuestionario de pretest y postest de 15 preguntas. De los resultados obtenidos, se evidenció que se encontraron incidencias positivas en el desarrollo del Pensamiento Computacional aumentando en el nivel *regular* de 57.3% a 61.8% y en el nivel *alto*

de 30.9% a 38.2% de los alumnos, lo que evidencia y corrobora la hipótesis planteada por el investigador.

Condo (2017), en su tesis de grado titulada *El pensamiento computacional en estudiantes del VII ciclo de la institución educativa particular Ricardo Palma*, desarrollada en la Universidad César Vallejo, Perú, tuvo como objetivo determinar el nivel de desarrollo del Pensamiento Computacional en los estudiantes del VII ciclo nivel secundario. La metodología utilizada en la investigación es de tipo básica con un enfoque cuantitativo de nivel descriptivo y diseño no experimental sobre una población de 30 estudiantes, utilizando el 100% de la población como muestra, la técnica e instrumento utilizados son la encuesta y un cuestionario de 24 preguntas cerradas para establecer el nivel de abstracción y Pensamiento Computacional en los alumnos. De los resultados obtenidos se concluye que 3.3% de los estudiantes tiene un nivel *bajo*, 66.7% nivel *medio* y 30% nivel *alto*.

Condo (2019), en *Uso de la plataforma Arduino en la mejora del pensamiento computacional, en la Institución Educativa Privada Ricardo Palma*, tesis de maestría de la Universidad César Vallejo, Perú, tuvo como objetivo establecer el nivel de influencia de la plataforma electrónica Arduino en el desarrollo del Pensamiento Computacional, en los estudiantes del primer año de educación secundaria de la Institución Educativa Privada Ricardo Palma. La metodología utilizada en la investigación fue de tipo aplicada con un enfoque cuantitativo de nivel descriptivo y diseño preexperimental sobre 18 alumnos que pertenecen al 1er grado de secundaria. El instrumento utilizado fue un cuestionario de 25 preguntas. De los resultados obtenidos, se evidenció que existen diferencias notables, al lograr un aumento de 0% a 88.9% de estudiantes en el nivel *logrado*, manteniéndose en 11.1% en el nivel *proceso* y reduciendo de 88.9% a 0% en el nivel *inicio*. Estos resultados señalan que el uso de la plataforma

electrónica Arduino influye de manera muy significativa en el desarrollo del Pensamiento Computacional.

Avalos (2017), en su tesis de maestría titulada *El software 'SCRATCH', para desarrollar el pensamiento creativo en estudiantes del 5to grado de secundaria de la IE Melchorita Saravia*, desarrollada en la Universidad César Vallejo, Lima, tuvo como objetivo establecer que el uso del software de programación SCRATCH desarrolla el pensamiento creativo. La metodología de investigación utilizada fue de tipo aplicada con un enfoque cuantitativo de nivel descriptivo y diseño preexperimental sobre una población de 26 estudiantes del 5to grado C, turno tarde, de secundaria de la IE Melchorita Saravia, utilizando como muestra el 100% de su población, el instrumento de medición utilizado es un cuestionario de pretest y postest de 20 preguntas. De los resultados obtenidos, se observa que en el pretest el 66% se encontraba en un rango de calificación de 0 a10, mientras que en el postest el 81% se encontraba en un rango de calificación 14 a 17. Estos resultados muestran la influencia positiva de la aplicación del software de programación SCRATCH en el desarrollo del pensamiento lógico de los alumnos.

2.2 Bases Teóricas

2.2.1 Pensamiento Computacional

2.2.1.1 Capacidades de Orden Superior. González (2002) señala que las capacidades de orden superior son principalmente ocho, las cuales son análisis o evaluación, síntesis o resumen, abstracción, manejo de información, pensamiento sistémico, pensamiento crítico, investigación y metacognición.

2.2.1.1.1 ***Análisis***. Joyanes (2008) la define como la primera fase para la resolución de un problema, la cual requiere precisar las entradas, el proceso que es lo que debe realizar el programa; para lograr el resultado deseado que es la salida.

González (2002) precisa que es la capacidad de identificar los principios o elementos de un todo tras haberlos separado.

2.2.1.1.2 ***Síntesis***. González (2002) la define como la capacidad de componer un todo a través de la comunión de sus partes.

2.2.1.1.3 ***Conceptualización***. González (2002) indica que es la capacidad de abstracción de los rasgos suficientes para descubrir y describir un fenómeno o problema.

2.2.1.1.4 Manejo de Información. Zapotecatl menciona que la información es: "un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje" (2014, p. 5).

Para González (2002), el manejo de la información hace referencia a la capacidad para ubicar los datos necesarios para la comprensión de un fenómeno o problema.

2.2.1.1.5 ***Pensamiento sistémico***. Capacidad de análisis y síntesis que agrega el carácter dinámico, centrándose en las interacciones de las partes de un sistema.

Peter Senge (2010) menciona que el pensamiento sistémico es la tarea mental con la finalidad de comprender cómo funciona un sistema y la manera en que sus partes se interrelacionan. Este pensamiento requiere una visión general del sistema para la comprensión de sus partes.

2.2.1.1.6 Pensamiento Crítico. Paul y Elder lo definen como: "el modo de pensar —sobre cualquier tema, contenido o problema— en el cual el pensante mejora la calidad de su pensamiento al apoderarse de las estructuras inherentes del acto de pensar y al someterlas a estándares intelectuales" (2003, p. 4).

Capacidad de pensar por cuenta propia, analizando y evaluando la situación, fenómeno o problema.

2.2.1.1.7 Investigación. Hernández et al. (2000) definen la investigación como "un conjunto de procesos sistemáticos, críticos y empíricos que se aplican al estudio de un fenómeno o problema" (p. 4).

Capacidad de profundizar el conocimiento de una problemática específica, proponer hipótesis y modelos conceptuales, recopilar datos y formular teorías.

2.2.1.1.8 ***Metacognición***. González (2002) la define como la capacidad de reflexionar sobre los pensamientos adquiridos, desde la planificación de una determinada tarea hasta la realización de la misma y su autoevaluación.

Para Zapata (2018), la metacognición es: "la conciencia de los propios recursos cognitivos con que cuenta el aprendiz" (p. 26).

2.2.1.2 Alfabetización Digital. Según Zapata (2015), la Alfabetización Digital es la adecuación y la formación en las actividades de comunicación, representación y proceso a las coordenadas en un ámbito completamente tecnológico. Se trata de la actualización por propia iniciativa para realizar cualquier transacción con la información, pero ahora utilizando las tecnologías y recursos actuales, al igual que la alfabetización clásica utilizaba las herramientas correspondientes a su época.

Según Gilster (1997), es la habilidad de comprender y utilizar la información de una gran diversidad de fuentes digitales. Adicionalmente, indica que, tiene que ver con la interpretación y dominio de las ideas, no con las pulsaciones que se efectúan sobre el teclado.

De acuerdo con Bawden (2001), se puede definir como la habilidad de leer y entender elementos de información en los formatos de hipertexto y multimedia.

En síntesis, se considera como un conjunto de habilidades necesarias para la vida y de manera adicional una habilidad para afrontar problemas científicos y tecnológicos, más allá de lo genérico que resulte este concepto.

2.2.1.3 Definición del Pensamiento Computacional. Diversos investigadores y asociaciones han dado sus aproximaciones a conceptos para definir lo que es el Pensamiento Computacional. El concepto más aceptado y utilizado actualmente es el que dio Wing (2006), quien define el Pensamiento Computacional como una forma de pensar que:

> Implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la ciencia de la computación. El pensamiento computacional incluye una amplia variedad de herramientas mentales que reflejan la amplitud del campo de la computación. (p. 33)

Menciona, además que, el Pensamiento Computacional tiene una serie de características como conceptualizar (no programar), fundamentar (no memorizar, una forma en que los humanos (no las computadoras) piensan, complementa y combina el pensamiento matemático y de ingeniería (STEM), ideas (no artefactos) y que es para todos y en todas partes.

Según Moursund (2006), el Pensamiento Computacional está vinculado a la interpretación y resolución de problemas utilizando el razonamiento humano, máquinas u otras formas que puedan aportar a la solución del problema.

Para la International Society for Technology in Education (ISTE, 2011) y la Computer Science Teachers Association (CSTA, 2011), el Pensamiento Computacional es una perspectiva para resolver un problema que integra las tecnologías digitales con el razonamiento humano. Adicionalmente, mencionan un conjunto de características que este posee, entre las cuales se encuentran la formulación de problemas, de una manera que permita el uso de una computadora; establecer y examinar datos lógicamente; representar datos a través de abstracciones, como modelos y simulaciones; automatizar soluciones; identificar, analizar y aplicar las posibles soluciones; generalizar y transferir estas soluciones.

En el 2011, Wing redefinió el concepto del término abordado anteriormente por ella, al proponer que “el pensamiento computacional son los procesos de pensamiento implicados en la formulación de problemas y sus respectivas soluciones para que estas sean representadas de forma que puedan realizarse de manera efectiva por un procesador de información” (p. 1).

Furber (2012) también da una aproximación definiéndolo como “el proceso de reconocimiento de los aspectos de la computación en el mundo que nos rodea, y la aplicación de herramientas y técnicas de la informática para entender y razonar sobre ambos sistemas y procesos naturales y artificiales” (p. 29).

Zapata (2015) menciona que el Pensamiento Computacional contempla quince (15) componentes, los cuales son el análisis sintáctico descendente, el análisis sintáctico ascendente, el método heurístico, el pensamiento lateral, la creatividad, la resolución de problemas, el pensamiento formal, la recursividad, la repetición, los métodos por aproximaciones sucesivas, los métodos cooperativos, los patrones o los elementos reutilizables, la sinéctica, la metacognición y la cinestesia.

Basándose en lo mencionado por Zapata, se reafirma la idea de que el Pensamiento Computacional se sirve de otros pensamientos de orden superior como el análisis, la metacognición, el pensamiento crítico, entre otros.

En el 2016, la CSTA redefinió el concepto que había propuesto anteriormente, indicando en esta ocasión que:

> El pensamiento computacional es una metodología de resolución de problemas que amplía el campo de la computación a todas las disciplinas, proporcionando un medio distinto de analizar y desarrollar soluciones a problemas que pueden ser resueltos computacionalmente que, a su vez, está centrado en la abstracción, la automatización y el análisis; siendo un elemento esencial de la disciplina de la computación. (p. 8)

Según el Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado (INTEF, 2017):

> El pensamiento computacional es el término utilizado para hacer referencia a las ideas y conceptos claves de las áreas disciplinarias de las Ciencias Informáticas y de la Computación. Se trata de un proceso de pensamiento que usa enfoques analíticos y algorítmicos para formular, analizar y resolver problemas. (p. 2)

Zapotecatl (2018) afianza las definiciones mencionadas, donde indica que:

> El objetivo del PC es desarrollar sistemáticamente las habilidades del pensamiento de orden superior, como el razonamiento abstracto, el pensamiento crítico y la resolución de problemas, con base en los conceptos de la computación. Además, potenciar el aprovechamiento del poder de cálculo que tienen las computadoras actualmente para innovar y volverlo una herramienta científica. (p. 5)

2.2.1.4 Habilidades que Desarrolla el Pensamiento Computacional

2.2.1.4.1 Análisis y Resolución de Problemas. Según la Real Academia de la Lengua Española, la definición del término problema es la formulación de una situación cuya respuesta debe obtenerse aplicando métodos científicos. Un problema a su vez tiene un estado inicial, una meta, recursos disponibles y un dominio. De acuerdo con Polya (1957), para la resolución de un problema, primero es necesario analizarlo y así comprenderlo en su totalidad, o en la mayor medida posible. Posteriormente, se traza un plan de acción que será ejecutado para luego analizar la solución planteada.

Acorde con López (2009), "para realizar el análisis a un problema se debe llevar a cabo también una serie de pasos los cuales son: la formulación del problema, identificar los resultados esperados, los datos disponibles, las restricciones y los procesos necesarios" (p. 11).

2.2.1.4.2 Abstracción. Según Csizmadia (a través del INTEF, 2017) el proceso de abstracción consiste en convertir un problema o fenómeno en uno más comprensible a través de la separación de los detalles innecesarios. El punto crítico de este proceso se encuentra en la selección de los detalles a separar, para volver el problema en uno más sencillo de afrontar y resolver, sin perder los datos importantes.

Wing (2009) menciona que el Pensamiento Computacional contempla todo lo que está relacionado con el proceso de abstracción ocultando los detalles. Sin embargo, menciona también

que en el Pensamiento Computacional no se debe centrar en la abstracción o en el modelo generado tras esta; sino que, también se debe tomar en cuenta las relaciones que se generan entre estas capas de abstracción y cómo es que estas interactúan.

2.2.1.4.3 Pensamiento algorítmico. Para comprender lo que es el pensamiento algorítmico, se requiere primero una definición de lo que es un algoritmo, así como sus características, clasificación y formas de representación.

a) Algoritmo. Según López (2009), una definición intuitiva del concepto de algoritmo conduce a la idea de un conjunto de procedimientos y reglas. En el ámbito computacional, los algoritmos son una herramienta que permite describir un conjunto finito de instrucciones, con un orden lógico y sin posibilidad de ambigüedades, que posteriormente serán ejecutadas por un ordenador para obtener un resultado.

Para Álvarez (2013), un algoritmo es un conjunto ordenado de instrucciones que generan la solución de un problema.

La Real Academia Española (2014) define como algoritmo a un "grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema".

Wing (2011) indica que "un algoritmo es una abstracción de un proceso que admite entradas, ejecuta una secuencia de pasos y produce resultados para alcanzar una meta determinada" (p. 1).

Para desarrollar un algoritmo se necesita identificar las tres (3) partes que son: Entrada, proceso y salida. En la *entrada* se definen todos los datos que el usuario debe ingresar o el programa generar, para ser almacenados en variables, dichos datos servirán en el *proceso* donde se realizan los cálculos necesarios para mostrar en la *salida* lo que el usuario solicita.

b) Características del algoritmo. López (2009) indica también que “un algoritmo, básicamente debe contar con tres características básicas, las cuales son que sea realizable, comprensible y preciso” (p. 22).

En primer lugar, que un algoritmo sea realizable, se refiere a que el proceso algorítmico debe finalizar después de una cantidad determinada de pasos. Bajo esta definición nace el concepto de que un algoritmo es incorrecto cuando se ejecuta con un conjunto de datos iniciales y proceso infinito o durante la ejecución se presenta un obstáculo insuperable, motivo por el cual, no se mostrará un resultado concreto.

En segundo lugar, se dice que un algoritmo es comprensible, ya que debe ser claro lo que hace, es decir sin confusiones, de forma que quien ejecute los pasos (ser humano o máquina) comprenda qué, cómo y cuándo realizarlo.

En tercer lugar, un algoritmo es preciso, cuando se indica las tareas objetivamente, sin ambigüedades con un orden lógico para llegar a la solución.

También se considera que un algoritmo debe ser definido debido a que, cuando se ejecuta múltiples veces, con los mismos datos, el resultado debe ser igual siempre, por lo cual, el orden de ejecución debe estar perfectamente indicado. Además, debe ser concreto, es decir, el algoritmo debe mostrar el resultado de acuerdo al planteamiento del problema.

c) Clasificación de algoritmos. Existen dos tipos de algoritmos, los algoritmos cualitativos y los algoritmos cuantitativos. Álvarez menciona que: "Los algoritmos cualitativos son aquellos en los que se describen los pasos utilizando palabras. Mientras que, los algoritmos cuantitativos son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso" (2013, p. 5).

d) Representación de algoritmos. Explica también Álvarez (2013), "que existen dos formas de representar algoritmos, mediante lenguajes gráficos y no gráficos" (p. 5).

Un lenguaje gráfico es el esquema de las acciones que realiza un algoritmo. La principal representación de este tipo es el diagrama de flujo de datos.

En un lenguaje no gráfico, se muestra en forma descriptiva las acciones que debe realizar un algoritmo. La principal representación de este tipo es el pseudocódigo.

Tras haber definido los conceptos previos, se procederá a dar las definiciones dadas por algunos investigadores y autores.

Según Csizmadia (como se citó en INTEF, 2017), "el pensamiento algorítmico es una forma de llegar a una solución a través de una definición clara de los pasos" (p. 7).

Acorde con Moursund (2006), el pensamiento algorítmico comprende el desarrollo y ejecución de algoritmos para resolver un tipo específico de problema o para realizar una tarea programada.

2.2.1.4.4 Descomposición de problemas. Csizmadia (como se citó en INTEF, 2017), indica que: "la descomposición es una manera de pensar acerca de los artefactos en términos de

sus partes y componentes. Cada pieza debe entenderse, solucionarse, desarrollarse y evaluarse por separado. Esto hace más fácil resolver problemas complejos" (p. 8).

2.2.1.4.5 Generalización. Para Csizmadia (como se citó en INTEF, 2017), "es una forma de resolver rápidamente los nuevos problemas sobre la base de las soluciones en los problemas anteriores y la construcción en la experiencia previa, haciendo uso del proceso de reconocimiento de patrones" (p. 8).

2.2.1.4.6 Depuración. De acuerdo con Csizmadia (como se citó en INTEF, 2017), "La depuración es la aplicación sistemática de las habilidades de análisis y evaluación utilizando el pensamiento lógico para predecir y verificar los resultados" (p. 10).

2.2.1.4.7 Automatización. La automatización es definida por Lee (como se citó en INTEF, 2017) como un "proceso en el cual se programa para ejecutar una serie de tareas repetitivas de manera rápida y eficiente en comparación con un ser humano, generando así un ahorro de recursos" (p. 33).

2.2.1.5 Cualidades que desarrolla el Pensamiento Computacional. El Pensamiento Computacional, además de tener características, permite desarrollar en las personas ciertas actitudes o disposiciones, tal como lo mencionan Barr, Harrison & Conery (2011) y Weintrop (2015), las cuales indican que son la "seguridad ante la complejidad, persistencia a la hora de trabajar con problemas difíciles, capacidad de gestionar la ambigüedad, capacidad de tratar con problemas abiertos, capacidad de comunicarse y trabajar colaborativamente para el logro de un objetivo o solución común" (pp. 51, 133). Adicionalmente, Woolbar (2016) menciona que estas cualidades o actitudes son la "creación, experimentación, depuración, perseverancia y colaboración" (p. 5).

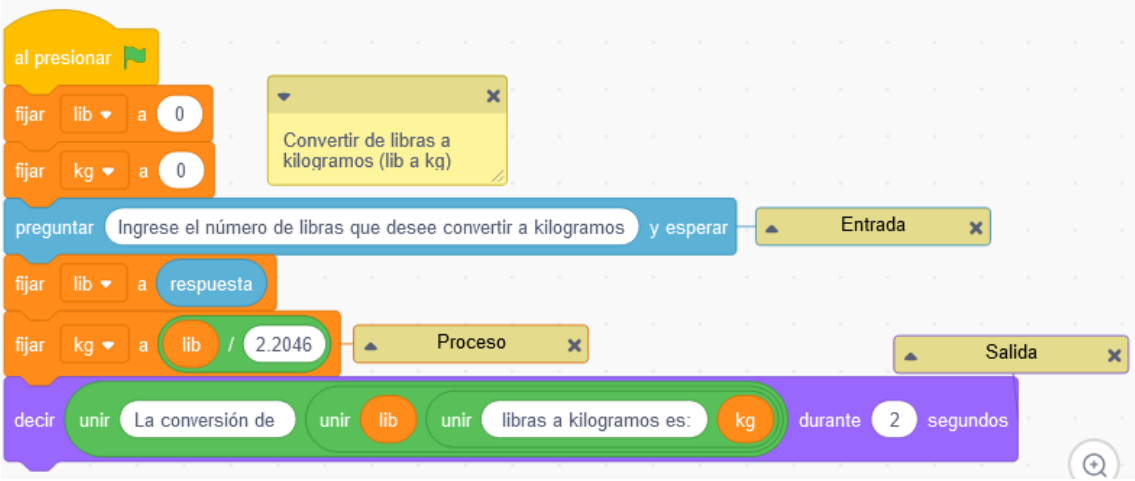
2.2.1.6 Dimensiones. Brennan y Resnick (2012) señalan que el Pensamiento Computacional se puede analizar mediante tres dimensiones principales, las cuales son los conceptos computacionales, las prácticas computacionales y las perspectivas computacionales. Cada una de ellas será detallada a continuación.

2.2.1.6.1 Conceptos Computacionales. Brennan y Resnick (2012) definen los conceptos computacionales como aquellos conceptos que emplean los diseñadores, a medida que programan, identificando siete conceptos principales que pueden o no utilizarse en otros ámbitos tales como las secuencias, ciclos, eventos, paralelismo, condicionales, operadores y datos.

a) *Secuencias.* Serie de pasos o de instrucciones individuales, que indican el comportamiento o acción que se debe producir.

Figura 2

Secuencia de instrucciones en Scratch.

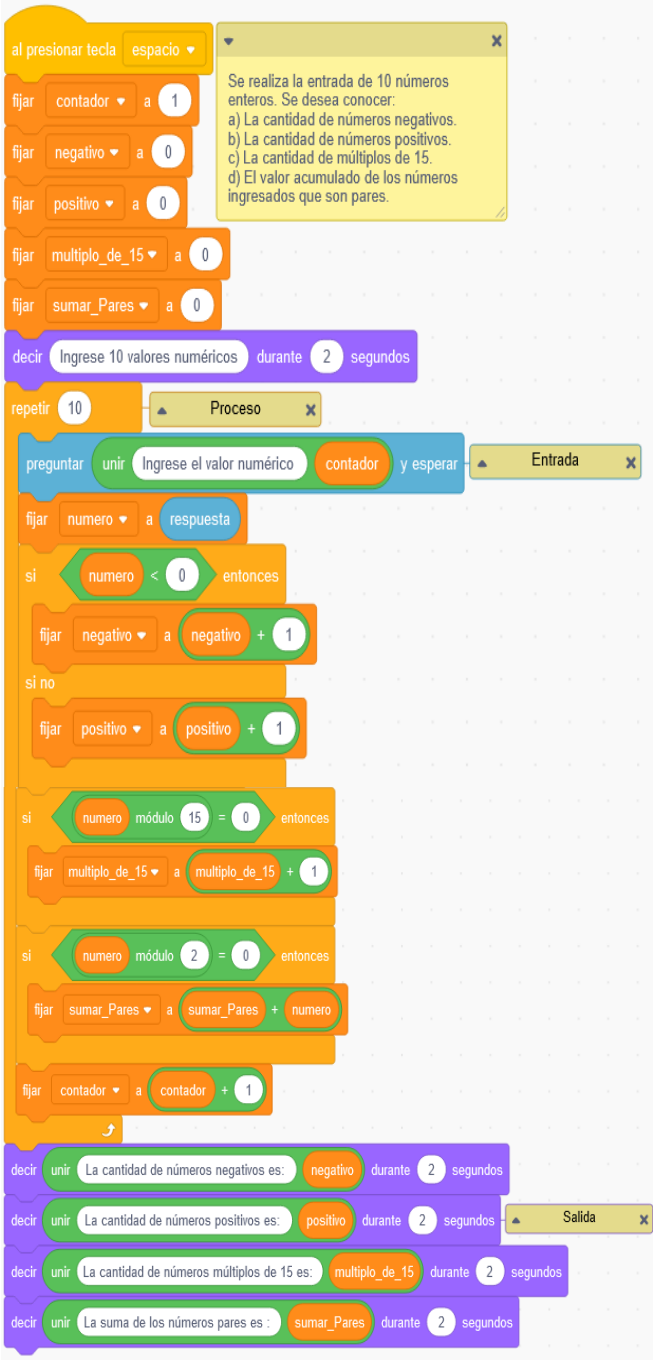


Nota: La figura muestra un ejemplo de la conversión de libras a kilogramos en Scratch.

b) Ciclos. Los ciclos o bucles son mecanismos que ejecutan la misma secuencia de instrucciones varias veces.

Figura 3

Ciclos o estructuras repetitivas en Scratch.

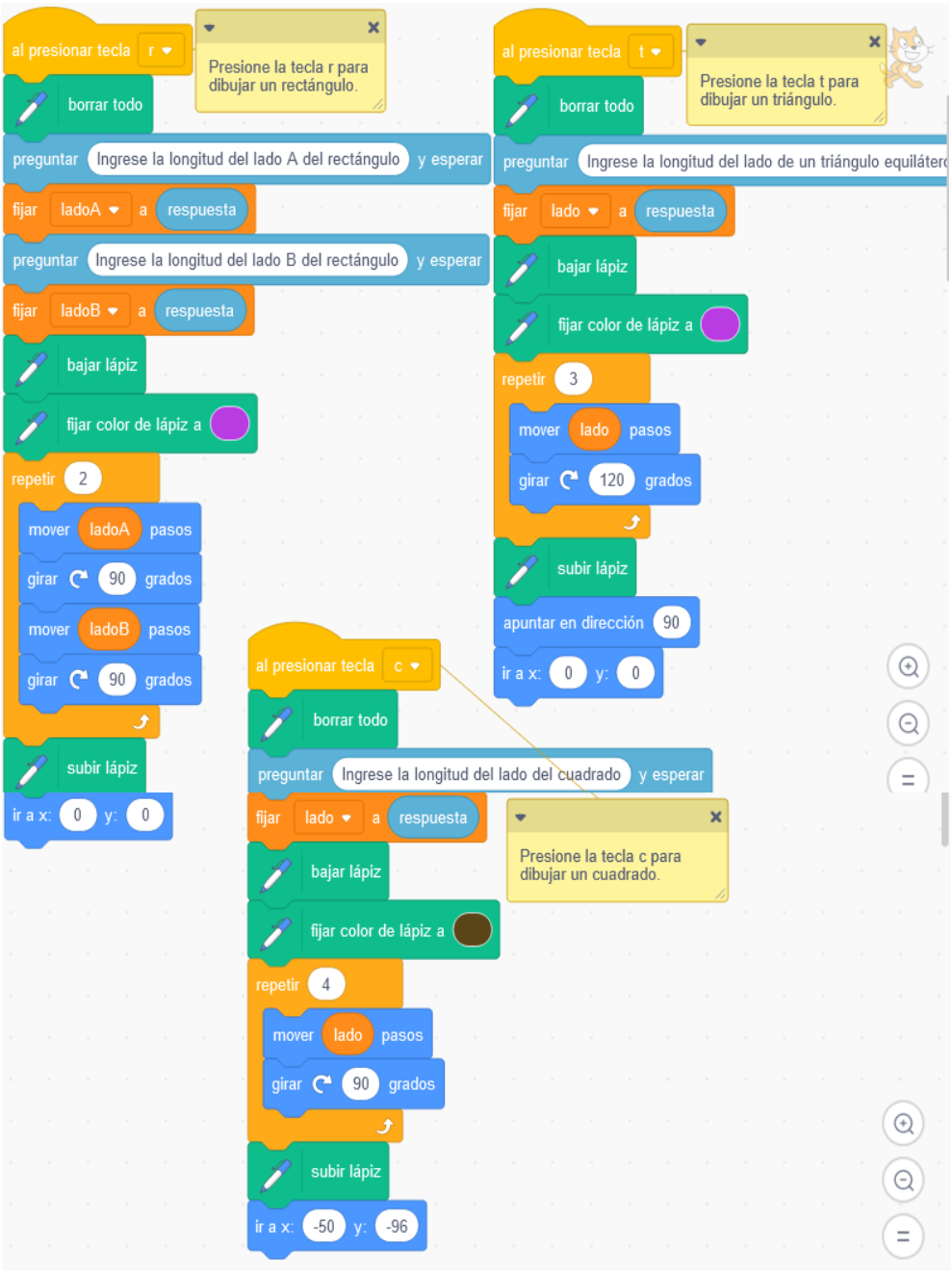


Nota: La figura muestra un ejemplo sobre el ingreso de 10 números enteros en Scratch.

c) Eventos. Un evento es una acción que desencadena la ejecución de otra, siendo un componente esencial en una secuencia de instrucciones.

Figura 4

Eventos en Scratch

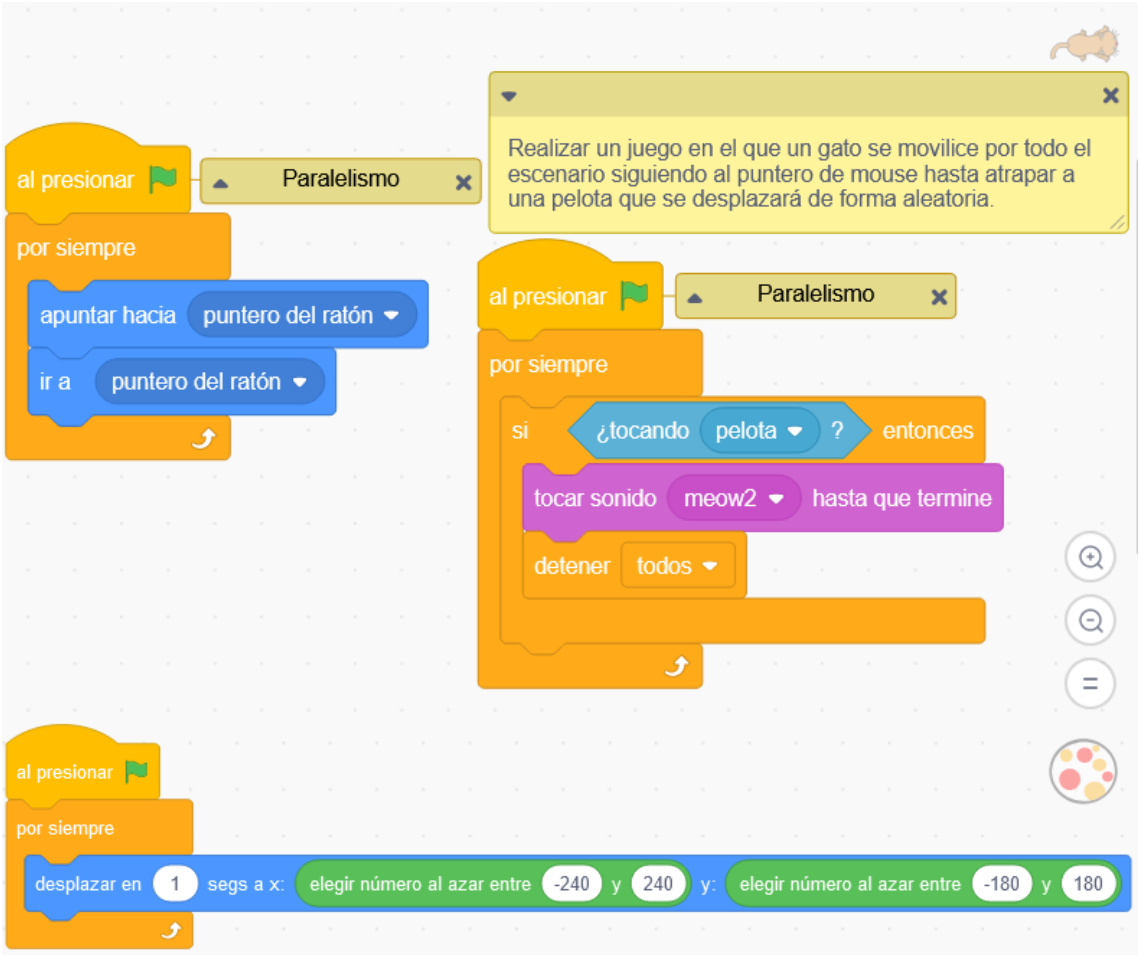


Nota: La figura muestra 3 eventos para dibujar un triángulo, rectángulo y cuadrado.

d) Paralelismo. El paralelismo es la capacidad de que se puedan ejecutar diversas secuencias o eventos a un mismo tiempo; es decir, simultáneamente.

Figura 5

Paralelismo de eventos en Scratch

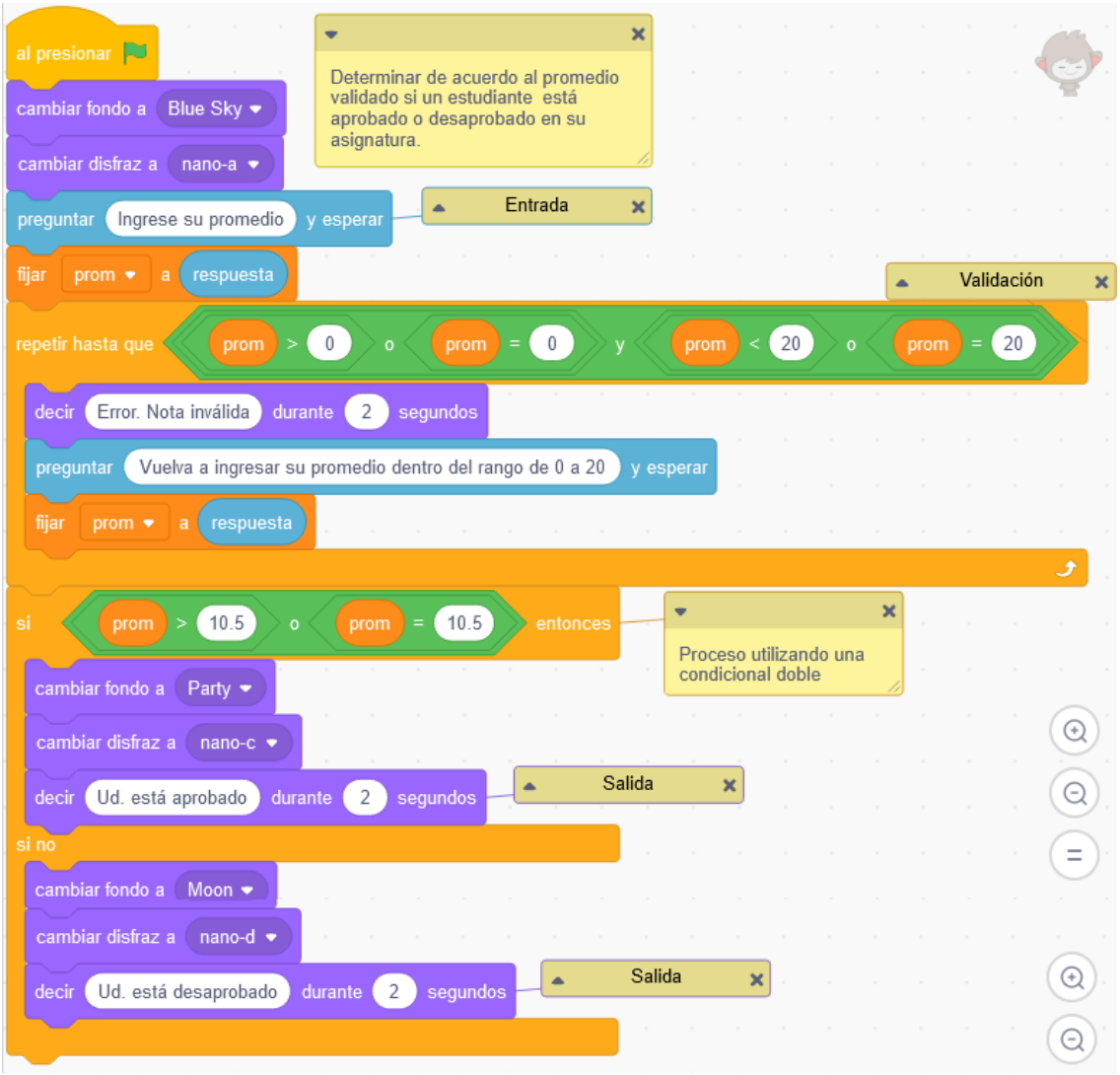


Nota: La figura muestra 2 eventos que se ejecutarán simultáneamente en el objeto gato.

e) Condicionales. Se refiere a la toma de decisiones, en base a ciertas condiciones, que determinarán el resultado que se obtendrá o el proceso que se ejecutará.

Figura 6

Estructuras condicionales en Scratch

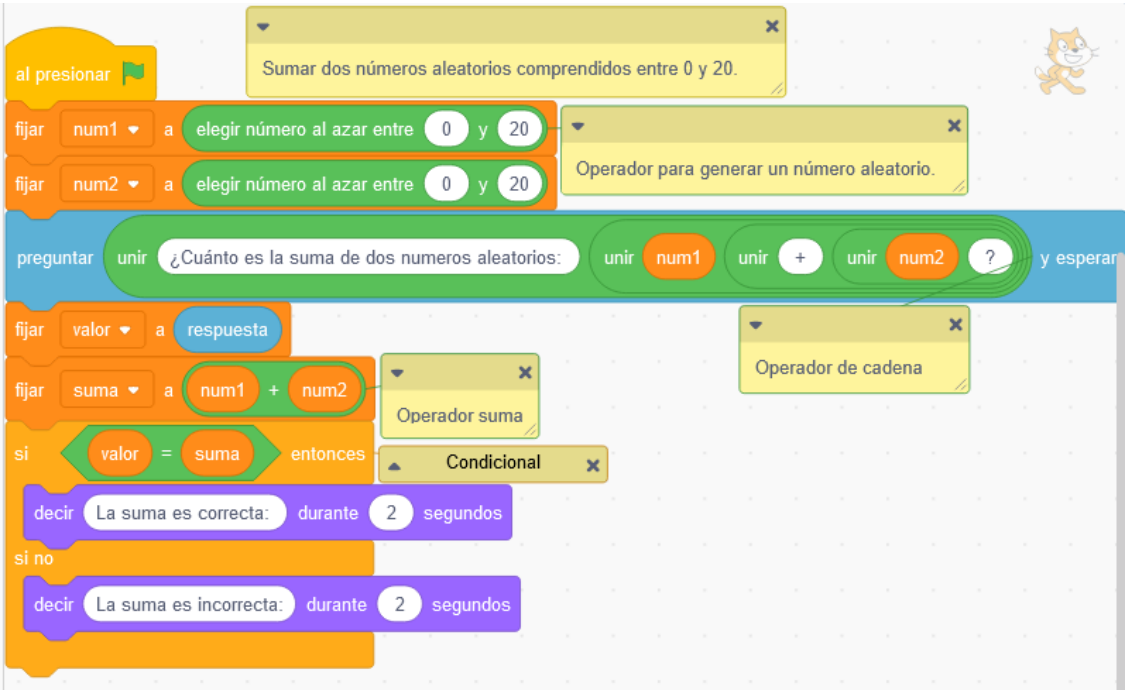


Nota: La figura muestra una condición que determina si un estudiante de acuerdo a su promedio está aprobado o desaprobado.

f) Operadores. Son una forma de apoyo a las expresiones lógicas y matemáticas que permiten realizar manipulaciones numéricas o de cadena.

Figura 7

Operadores en Scratch



Nota: La figura muestra la suma de 2 números aleatorios comprendidos entre 0 y 20.

g) Datos. Los datos son aquellos que ofrecen la opción de guardar, recuperar y actualizar valores durante una secuencia de instrucciones.

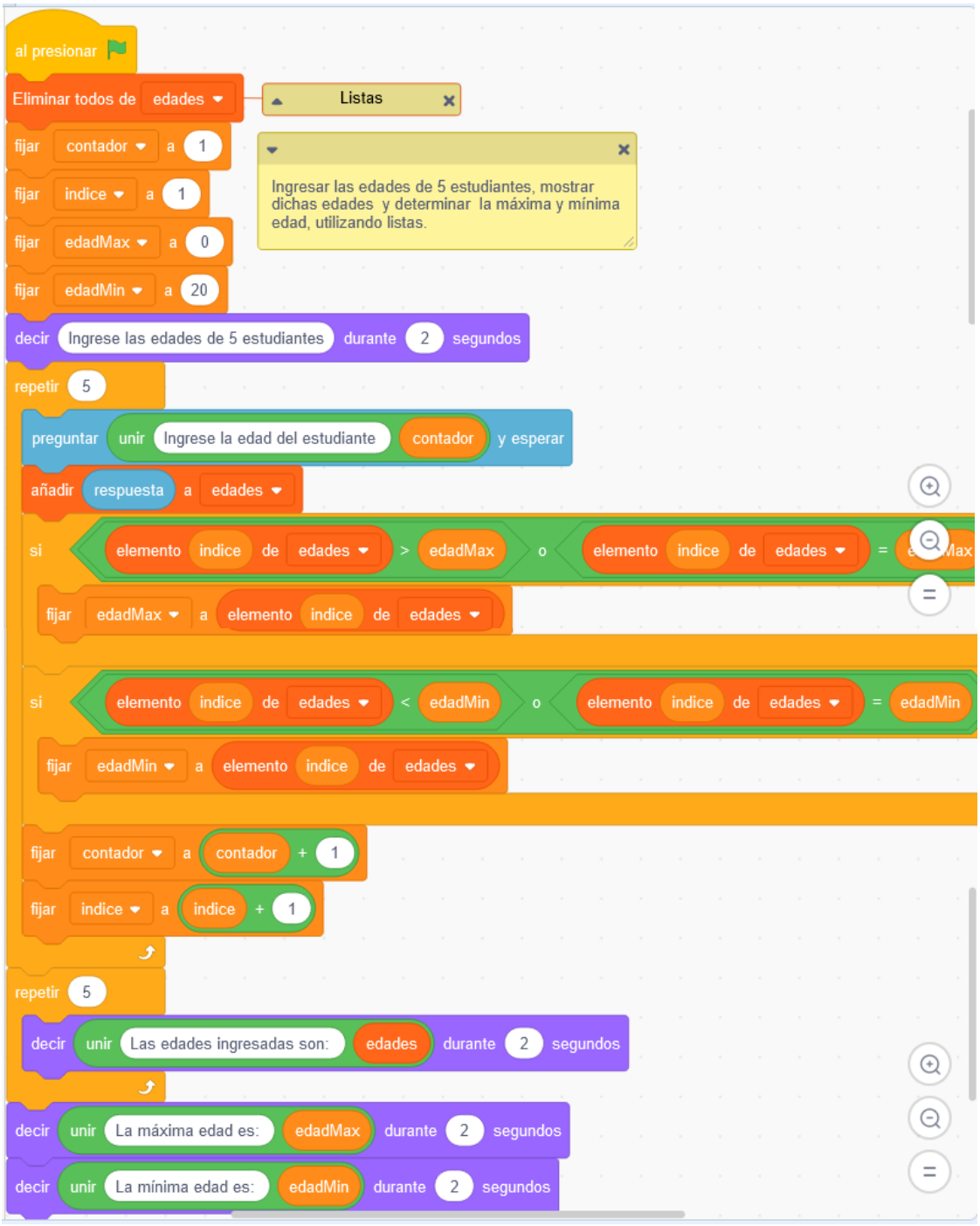
Figura 8

Variables en Scratch

Nota: La figura muestra las variables creadas para almacenar datos que serán utilizados durante la ejecución del programa. Cada variable almacena un dato.

Figura 9

Listas en Scratch



Nota: La figura muestra la lista creada para almacenar las edades de 5 estudiantes y determinar la máxima y mínima edad. Una lista almacena un conjunto de datos.

2.2.1.6.2 Prácticas computacionales. Brennan y Resnick (2012) señalan que las prácticas computacionales son las prácticas y estrategias que realizan los diseñadores cuando programan, clasificándose estas en cuatro grupos.

a) Ser Incremental e Iterativo. Según Brennan y Resnick (2012), para diseñar un proyecto primero se formula el proyecto, luego se desarrolla un plan de diseño y, finalmente, se implementa el diseño repitiéndose estos pasos de forma iterativa.

b) Ensayar y Depurar. Para Brennan y Resnick (2012), es el medio por el cual se prueba el diseño del proyecto para determinar la existencia de errores y poder corregirlos cada vez que se presenten. Es muy importante el uso de esta práctica en el diseño de cualquier proyecto, ya que permite anticipar cualquier error que se pudiera presentar durante el despliegue del mismo.

c) Reusar y Remezclar. Para Brennan y Resnick (2012), construir sobre lo que ya ha sido creado previamente por uno mismo o terceros es una práctica que beneficia el desarrollo de un proyecto en diversos aspectos, como el ahorro de tiempo volviendo a crear una forma de realizar una tarea, adaptando lo creado a los propios proyectos, dando la posibilidad de servirse del conocimiento reutilizando soluciones similares.

d) Abstraer y Modularizar. Es la práctica de construir algo grande a través de colecciones de partes más pequeñas, llevándose a cabo este proceso en capas de abstracción y modularización en múltiples niveles desde la concepción del proyecto, análisis de la problemática y corrección de errores.

2.2.1.6.3 Perspectivas computacionales. Brennan y Resnick (2012) definen a las perspectivas computacionales como las vistas que los diseñadores desarrollan sobre el mundo que les rodea y sobre ellos mismos.

a) Expresar. Es la posibilidad que encuentran los diseñadores a través de la computación para expresarse y proponer su punto de vista, así como nuevas ideas.

b) Preguntar. El hecho de tener nuevos retos y problemas que afrontar, genera la necesidad de buscar soluciones en distintas fuentes, lo que lleva a consultar en algún foro, algún amigo, docente o grupo de personas que conozca sobre el tema.

c) Conectar. Al haber generado la necesidad de preguntar a nuevas personas o a las que ya se conocía, se genera un nuevo entorno mediante el cual el diseñador conecta con las demás personas con un tema en común.

2.2.1.7 Pensamiento Computacional sin computadora. Según Zapotecatl (2014), la enseñanza de la computación sin utilizar las computadoras, a través de actividades, reduce las desatenciones y convenciones técnicas que producen las computadoras. La enseñanza se lleva a cabo usando distintas herramientas como juegos, puzles, crayones, dibujos, entre otros. Este tipo de actividad tiene por objetivos, iniciar en los estudiantes los conceptos subyacentes de la computación, indagar otras maneras de solucionar problemas basados en los conceptos de la computación para diferentes escenarios, entender que estos conceptos son esenciales para la solución de múltiples casos y que la computadora es una vía que permite automatizar los procesos; y expresar problemas y resoluciones computacionales.

2.2.1.8 Pensamiento Computacional con computadora. Indica también Zapotecatl (2014), que actualmente, han emergido varias herramientas por computadora que favorecen la aplicación del pensamiento computacional en el aula de clase. Ejemplo de este tipo de herramientas es el software y lenguaje de programación SCRATCH, sobre el cual, se extendió su uso en Estados Unidos, Europa y algunos países de América del Sur. El uso de estas herramientas por computadora tiene como finalidad estimular el desarrollo del pensamiento lógico y algorítmico,

aplicar métodos para solucionar problemas de forma metódica y ordenada, aprender conceptos matemáticos como coordenada, variables, algoritmos, aleatoriedad, entre otros. Asimismo, los fundamentos de programación, utilizando recursos multimedia para facilitar el aprendizaje cooperativo a través del intercambio de conocimientos.

2.2.1.9 Relación del Pensamiento Computacional con la codificación y la programación. Usualmente, se usan los términos codificación o programación como similares o iguales, para referirse al hecho de "escribir" las instrucciones que un ordenador ejecuta, sin tener en cuenta que el término programación hace referencia a la actividad de diseñar e implementar la solución de un problema tras haberlo analizado previamente. Mientras que, la codificación es la etapa de desarrollo de soluciones en un lenguaje de programación definido (INTEF, 2017).

Después de dar estas definiciones y ya haber abordado la de Pensamiento Computacional, se puede considerar que el Pensamiento Computacional, la codificación y la programación no son necesariamente lo mismo, sino que el pensamiento computacional se sirve de estos junto a otros fundamentos como el análisis y la descomposición.

A pesar de esto, el Pensamiento Computacional, por medio de la programación, puede desarrollar una herramienta que estimule los conceptos de abstracción, investigación empírica y otros fundamentos del Pensamiento Computacional (INTEF, 2017).

2.2.1.10 Razones para incorporar el Pensamiento Computacional en los currículos y normativas oficiales. Webb et al. (como se citó en INTEF, 2017) sostienen que: "Los aspectos de la Computación, incluida la programación, son una forma ideal de desarrollar el Pensamiento Computacional, que los alumnos pueden aplicar más ampliamente como una estrategia de resolución de problemas" (p. 17). En Austria, la computación es percibida como una manera de resolver problemas a través del análisis de procesos en el medio en que se desenvuelven las

personas. De este modo, los alumnos deben poder entender sistemas complejos e interdependencias.

Adicionalmente, Webb et al. (como se citó en INTEF, 2017) indican que: “La introducción del pensamiento computacional también se considera como una forma de reducir la brecha entre los currículos y las necesidades actuales de los estudiantes y de la sociedad en general” (p. 18).

Finalmente, la inclusión del Pensamiento Computacional en los currículos y normativas oficiales debe responder a dos tendencias principales según el INTEF (2017):

> El desarrollo de habilidades de pensamiento computacional en niños y jóvenes para que puedan pensar de manera diferente, expresarse a través de una variedad de medios, resolver problemas del mundo real y analizar temas cotidianos desde una perspectiva diferente; y el fomento del pensamiento computacional para impulsar el crecimiento económico, cubrir puestos de trabajo TIC y prepararse para futuros empleos. (p. 17)

Estas dos tendencias responden a lo que Wing (2010) se refería al mencionar que el Pensamiento Computacional es una habilidad clave del siglo XXI.

2.2.1.11 Formación del profesorado en el Pensamiento Computacional. La relevancia de la formación del profesorado como agentes capacitados para potenciar el Pensamiento Computacional en los estudiantes es evidente, esto se ve reflejado en el gran interés que ha surgido también de parte de los Ministerios de Educación en los diversos países, así como de otros organismos y programas privados. Tales son los casos de la Fundación Nacional de Ciencias de Estados Unidos que subvencionó el programa CSK10, por el cual, entre los años 2010 y 2016, diez mil (10,000) docentes fueron formados, se suma el Ministerio de Educación de Corea del Sur y el Ministerio de Ciencia, TIC y Planificación del Futuro con su programa Software Education, por el cual se formará a 60000 maestros, de los cuales 6000 recibirán una formación extendida.

Según el INTEF (2017), como resultado de esta preocupación por mejorar la calidad del profesorado en materia de Pensamiento Computacional, han surgido cuatro perspectivas principales. El primero es un programa de desarrollo profesional Partner4CS. El segundo es un curso denominado psicología educativa, el cual será obligatorio para futuros docentes, en el que se incluyen tópicos de resolución de problemas y pensamiento crítico con el Pensamiento Computacional. El tercero comprende la intervención preprofesional para colaborar con los docentes para la utilización del Pensamiento Computacional y la programación como herramienta de enseñanza transversal. El cuarto enfoque se orienta al uso de herramientas en las cuales se escribe en pseudocódigo para luego ser traducidos en un lenguaje de programación, siendo la primera una habilidad del Pensamiento Computacional y, la segunda, una habilidad de la programación.

2.2.1.12 Iniciativas para promover el Pensamiento Computacional

2.2.1.12.1 A nivel global. CoderDojo es un movimiento global gratuito que fue fundado en 2013 por James Whelton, donde los niños de siete (07) a diecisiete (17) años tienen la oportunidad de aprender a codificar utilizando su creatividad en un entorno seguro y social.

Code.org es una organización sin fines de lucro creada en el 2013 por los hermanos Hadi y Ali Partovi, dedicada a promover el acceso a las Ciencias de la Computación en las escuelas a nivel mundial con programas como Code Studio.

Bebras, organización sin fines de lucro, tiene por objetivo promover el desarrollo del Pensamiento Computacional —a través de desafíos en línea conocidos como Desafío Bebras—, a los cuales cualquier persona tiene la posibilidad de acceder. La idea surge en el año 2003 y, actualmente, se ha expandido mundialmente.

CS Unplugged es una organización sin fines de lucro que, a través de un repositorio gratuito de material didáctico, tiene como objetivo promover la informática en los jóvenes de una manera interesante a través de juegos, rompecabezas y dinámicas.

Code Club es un programa de la Fundación Raspberry Pi, que actualmente cuenta con más de 13.000 clubes en más de 160 países y apoyan a más de 180.000 jóvenes. Esta organización sin fines de lucro tiene por finalidad promover la informática a través de proyectos gratuitos abiertos a todos.

Made With Code fue creada por Google en el 2014, con el objetivo de formar a las mujeres en la programación, cerrando la brecha de género en la industria tecnológica.

2.2.1.12.2 Europa. EU Code Week es una iniciativa que busca promover la programación y la alfabetización digital a personas de todas las edades de una forma interesante para que tengan la posibilidad de aprender mientras se divierten y colaboran con otras personas. La iniciativa tuvo su primera edición en el 2013 y, en su última edición (2019), ha logrado convocar a 4.2 millones de participantes.

Iniciativa Europea de Programación, también llamada all you need is {C<3DE}. Esta iniciativa fue creada en el 2008 y cuenta con el auspicio de la Comisión Europea, busca fomentar la enseñanza de la programación y Pensamiento Computacional dentro y fuera del ámbito escolar.

Barefoot Computing es un programa creado en 2014 por BCS, The Chartered Institute of IT y su red Computing at School, con fondos del Departamento de Educación para apoyar y capacitar a los maestros de educación primaria en la enseñanza de ciencias de la computación.

Computing At School tiene como finalidad promover el desarrollo del Pensamiento Computacional, al apoyar a todos los involucrados en la enseñanza de informática en las escuelas.

Code Like a Girl, fundada en 2015 por Ally Watson, es una empresa social que brinda a mujeres y niñas en Australia las herramientas para introducirse, desarrollarse y prosperar en el campo de la programación.

Programamos, organización sin fines de lucro que promueve el desarrollo del Pensamiento Computacional desde niños hasta adultos, a través del desarrollo de videojuegos y aplicaciones móviles.

2.2.1.12.3 Otras regiones. Code@SG Movement es una iniciativa de Infocomm Media Development Authority, dirigida a personas de todas las edades para desarrollar el Pensamiento Computacional en Singapur, que forma una generación de innovadores y creadores digitales para resolver problemas del mundo real.

Computhink es una organización creada en el 2015, que ofrece programas como campamentos de verano, clases después de la escuela, entre otras, donde se promueve el desarrollo del Pensamiento Computacional y la programación de computadoras, dirigidos a estudiantes de todas las edades, centrándose en niños de 7 a 16 años.

2.2.2 SCRATCH

2.2.2.1 Lenguajes de programación orientada a objetos. Según la Universidad Autónoma del Estado de Hidalgo (s.f.), un lenguaje de programación orientada a objetos tiene como características el encapsulamiento de datos, ocultamiento de datos y formas de acceso, iniciación automática y aclarado de objetos, sobrecargada de operadores, herencia y adaptación dinámica. Los objetos se crean a partir de una plantilla denominada clase.

2.2.2.2 Lenguajes de programación visuales

2.2.2.2.1 ALICE. Alice nace como una herramienta de creación de prototipos de realidad virtual en 1995 para ser utilizada también por no programadores. En el 2004, Wanda Dann y Stephen Cooper presentan una versión de Alice junto a un plan de estudios para unir la pedagogía y la plataforma. Como resultado de esto se lanza el software Alice 2 y el libro Learning to Program with Alice.

2.2.2.2.2 MIT APP INVENTOR. Es un lenguaje de programación creado por el MIT y Google en el 2010, cuya plataforma en línea es de libre acceso, permite crear aplicaciones móviles para el sistema operativo Android. En la vista diseño, se elabora la interfaz de la APP, luego en el editor de bloques se establece la funcionalidad o el comportamiento de la APP. Su estructura hace que la programación sea más sencilla y dinámica.

2.2.2.2.3 LEGO MINDSTORM. El set de herramientas robóticas de LEGO denominado MINDSTORMS fue presentado en 1998 para ser utilizado como herramienta dentro de la programación. Con el paso del tiempo, se ha vuelto uno de los productos más vendidos de la compañía, impulsando el Desarrollo de una comunidad global enfocada en la robótica para todas las edades.

2.2.2.2.4 SCRATCH. Es un lenguaje de programación utilizado por muchas personas de todas las edades para construir en forma lúdica aplicaciones como historietas interactivas, animaciones, juegos, música y arte y compartir con otros usuarios de la comunidad a través de la red.

Su implementación es el resultado del trabajo cooperativo entre la Universidad de California (UCLA) y su Colegio de Graduados en Educación y Estudios de la Información con la subvención del Instituto Tecnológico de Massachusetts (MIT) y de su Fundación Nacional para la Ciencia, la Fundación Intel y el Laboratorio de Medios. SCRATCH, según Prudencio (2007):

> Es un medio de expresión mediante el cual los jóvenes y menos jóvenes pueden expresar sus ideas y responde a la pretensión de proporcionar una herramienta que facilite el uso de los ordenadores de forma creativa, superando el modelo de formación tradicional, que viene utilizando las nuevas tecnologías para reproducir prácticas educativas obsoletas. SCRATCH reconoce la aportación del Micromundo de Logo, los e-toys de Squeak y LogoBlocks como sus precedentes o fuentes en las que se ha inspirado. El lenguaje de programación de SCRATCH, por un lado, se basa en Logo, en especial en sus primitivas, y presenta un entorno en el que múltiples objetos pueden evolucionar e interactuar. A pesar de haber sido ideado como una sencilla herramienta para jóvenes, SCRATCH ya ha demostrado ser un instrumento valioso. Al trabajar con proyectos de SCRATCH, los jóvenes tienen oportunidades para aprender conceptos informáticos importantes tales como control de flujo, condicionales, variables, tipos de datos, eventos y procesos. Lo que hace verdaderamente atractivo a SCRATCH es la gran simplicidad con la que, en muy poco tiempo, un usuario sin conocimientos de programación puede comenzar a elaborar y ejecutar sus propios proyectos. (pp. 79-80)

a) Instalación. SCRATCH es una plataforma en línea, la cual puede ser accedida desde cualquier navegador, pero de ser requerido cuenta con la opción de descargar un archivo para instalarlo en el sistema operativo que se esté utilizando. SCRATCH también cuenta con una aplicación móvil disponible para dispositivos Android o Apple.

b) Características. SCRATCH está conformado por bloques gráficos con una interfaz sencilla e intuitiva. Asimismo, permite trabajar de forma cooperativa compartiendo proyectos, scripts y personajes en la web. Las aplicaciones se desarrollan mediante la unión de bloques que pueden ser eventos, movimientos de objetos y sonidos que se pueden ejecutar y visualizar sobre el navegador de Internet.

c) Entorno. SCRATCH tiene una plataforma que está disponible en setenta y cuatro (74) idiomas y conformada por 3 áreas de trabajo: *Área de bloque* clasificado en nueve (9) categorías que son movimiento, apariencia, sonido, eventos, control, sensores, operadores, variables y mis bloques. *Área de desarrollo*, espacio para construir los programas arrastrando los bloques necesarios y *Área de ejecución* para visualizar los programas desarrollados.

Además, cuenta con once (11) extensiones para agregar funcionalidades a su proyecto conectando con otros servicios externos como Lego, Google Traductor, Amazon Web Services, Vernier. Las extensiones son: Música, lápiz, sensor de video, texto a voz, traductor, makey makey, micro:bit, Lego Mindstorms, Lego Boots, Lego Education WeDo 2.0, Go direct force & Acceleration.

Para insertar un objeto o un fondo en un proyecto se tiene cuatro opciones, la primera consiste en insertar desde tu equipo un objeto o fondo, la segunda genera de forma aleatoria el objeto o fondo de la galería, la tercera permite en la plataforma realizar tu propio diseño para

insertarlo en el proyecto y la cuarta opción consiste en seleccionar e insertar de la galería de objetos o fondos.

Los proyectos se pueden desarrollar de forma local en su equipo o accediendo a la plataforma en línea con la opción de poder ser publicados y compartidos para toda la comunidad de Scratch.

d) Categorías. Las categorías disponibles dentro de SCRATCH son la de control, que permite incluir bucles, condicionales y esperas durante la ejecución; movimiento, que permite controlar la orientación, el desplazamiento, la posición y el número de pasos; apariencia, para insertar diálogos, cambiar el disfraz del personaje, hacer aparecer y desaparecer objetos en la pantalla y modificar su dimensión; sensores, para detectar señales y realizar acciones frente a ellas; operadores, para realizar cálculos básicos y establecer relaciones; variables, para almacenar, actualizar y eliminar datos; sonido, permite incluir sonidos predeterminados; eventos, iniciar la programación, finalizarla, realizar acciones al hacer clic o pasar el puntero por determinadas áreas. Adicionalmente, permite la opción de agregar plugins para utilizar un lápiz.

2.2.2.2.5 SCRATCH para ARDUINO (S4A). Fue creado por el equipo de SmallTalk del Citilab en el 2010. Es una modificación de Scratch, que en combinación con la plataforma de hardware de Arduino permite programar robots, inventos tecnológicos mediante sensores y actuadores. S4A permite adquirir fundamentos básicos de programación e iniciarse en el mundo de la robótica de una forma sencilla.

2.2.3 Aprendizaje

2.2.3.1 Estilos de aprendizaje. Para Alonso, Gallego y Honey (1995) existen cuatro estilos de aprendizaje; sin embargo, esta clasificación actualmente ha ido aumentando progresivamente hasta terminar en doce estilos, los cuales son activo, pensativo, teórico, práctico,

lógico o dialéctico, social o interpersonal, intrapersonal, aprendizaje visual, aural, verbal, representación kinestésico y multimodal.

2.2.3.1.1 Aprendizaje lógico. En este estilo de aprendizaje, los estudiantes aplican la lógica utilizando sus leyes y el razonamiento, en lugar de contextualizar, para tener un mejor entendimiento de la problemática. Suelen utilizar esquemas en lo que resaltan los términos o ideas relevantes y asocian palabras a las que incluso no les encuentren el sentido.

2.2.3.1.2 Aprendizaje visual. En este estilo de aprendizaje, los estudiantes asimilan de mejor manera las imágenes que los textos, mediante diagramas, gráficos y vídeos, asociando conceptos a estos. Les resulta más efectivo el uso de simbologías o taquigrafías visuales al tomar apuntes, ya que de ese modo retienen mejor la información.

2.2.3.1.3 Aprendizaje multimodal. Los estudiantes, al poseer múltiples capacidades, también pueden aprender de múltiples formas, combinándolas para encontrar un estilo óptimo de aprendizaje, resultando en uno más flexible.

2.2.3.2 Tipos de aprendizaje. Según García-Allen (2020) recopila los distintos tipos de aprendizaje y los enumera en una lista de trece tipos, donde se encuentran:

> El aprendizaje implícito o no intencional, aprendizaje explícito o intencional, aprendizaje asociativo, aprendizaje no asociativo, aprendizaje significativo o relacional, aprendizaje cooperativo, aprendizaje colaborativo, aprendizaje socioemocional, aprendizaje observacional o vicario, aprendizaje experiencial, aprendizaje por descubrimiento o activo, aprendizaje memorístico o mecánico y aprendizaje receptivo o pasivo.

2.2.3.2.1 Aprendizaje significativo. Álvarez (2020) define el aprendizaje significativo como un proceso que se inicia cuando la persona toma la información, la selecciona, organiza y asocia con el conocimiento previo. Es decir, el aprendizaje se hace presente cuando algún nuevo

contenido guarda relación con una experiencia o conocimiento adquirido previamente con la motivación y creencia de que guarda relación con un aspecto importante. Este proceso conlleva la generación y adquisición de un conocimiento con un sentido único y distinto por cada persona.

2.2.3.2.2 Aprendizaje colaborativo. Según García-Allen (2020), el aprendizaje colaborativo es un proceso que permite que cada estudiante aprenda, pero no de forma aislada, sino junto a sus compañeros, teniendo un mayor grado de libertad para trabajar en grupos. En este tipo de aprendizaje es usual que los docentes propongan un tema o una problemática y los estudiantes decidan la manera en la que este será abordado.

2.2.3.3 Teorías del Conocimiento

2.2.3.3.1 Teoría del Constructivismo. Desarrollada por Jean Piaget, donde focaliza el aprendizaje como un proceso, que consiste en que las personas construyen su conocimiento continuamente, a partir de experiencias del ambiente que les rodea. Siguiendo el lineamiento de esta teoría, las personas no consiguen ideas, sino que las crean. Esta teoría es la base de muchas iniciativas de reforma educativa, a lo largo de un amplio periodo de tiempo, la cual también se ha expandido a una amplia variedad de espacios geográficos.

2.2.3.3.2 Teoría del Construccionismo. Desarrollada por Seymour Papert, matemático, docente y pionero de la inteligencia artificial, sostiene que las personas desarrollan nuevos conocimientos de forma eficaz cuando se comprometen en la construcción de sus proyectos; siendo de suma importancia que las personas participen activamente en el proceso de creación de “algo” que tenga sentido e importancia tanto para ellos como para su entorno. Esta teoría está ligada fuertemente al aprendizaje experimental.

2.2.3.4 El Juego como estrategia de enseñanza. El juego constituye una situación que puede generar un momento de reflexión en los estudiantes, por lo cual, es labor del docente generar, participar y planificar juegos para los estudiantes, fomentando su interés en el tema o problemática propuesta.

Brousseau (1986) señala que el alumno aprende del mismo modo que lo hace la sociedad humana, adaptándose a las adversidades, generando conocimiento como resultado de esta adaptación.

Brousseau (1986) menciona también, que el juego como herramienta de aprendizaje contiene dos factores importantes, uno formativo y otro informativo. El formativo implica desarrollar una motivación sobre los estudiantes para que, a través de una experiencia directa, el estudiante use el juego como una estrategia para la adquisición de conocimiento. El segundo aspecto, agrega conocimientos, habilidades y destrezas en relación al contenido desarrollado.

2.3 Definición de términos básicos

Alfabetización. Acción de enseñar a alguien a leer y a escribir.

Análisis. Separar en partes el objeto de estudio para conocer y comprender su composición.

Problema. Planteamiento de un asunto que requiere una solución a través de métodos científicos.

Pensamiento Computacional. Es la capacidad de formular problemas y solucionar aplicando habilidades como abstracción, descomposición de problemas, reutilización de patrones, creatividad y el pensamiento algorítmico que se representa a través de una secuencia de instrucciones con un orden lógico. También es el proceso de reconocimiento y aplicación de herramientas y técnicas de la informática para razonar y comprender sobre los sistemas tanto naturales como artificiales.

Crítica. Opiniones o apreciaciones positivas o negativas acerca de un tema en particular que resultan de un análisis. Capacidad de discernir entre dos o más conceptos o ideas para hacer una aseveración.

Lógica. Modo de pensar y actuar que utiliza las leyes, modos y formas de las proposiciones para determinar su verdad o falsedad.

Algoritmo. Conjunto de instrucciones finitas con un orden lógico para hallar la solución a un problema.

Creatividad. Facultad de producir algo de la nada. Establecer, fundar, introducir por primera vez algo.

Abstracción. Proceso mental que consiste en separar e identificar las propiedades, características o cualidades esenciales de una situación u objeto para analizarlo de forma aislada.

Descomposición. Separación en las diversas partes que forman un compuesto.

Patrones. Modelos que sirven de guía o muestra para la creación de otras cosas iguales.

Generalización. Abstracción de las características comunes y esenciales de muchos objetos, personas, lugares, entre otros, para formar un concepto general.

Programa. Es un conjunto de instrucciones escritas en un lenguaje de programación que permite a una computadora realizar tareas determinadas como funciones o procedimientos, tratamiento de textos, diseño de gráficos, resolución de problemas matemáticos, administración de bancos de datos, etcétera.

Programación. Es el proceso de desarrollo de una aplicación informática, contempla no sólo la codificación sino también la prueba, depuración del código fuente que es compilado o interpretado y ejecutado por un lenguaje de programación para realizar una tarea específica en un

equipo. Asimismo, también se define como la preparación de máquinas o aparatos para que funcionen en el momento indicado.

Lenguaje de programación. Es un idioma artificial creado con reglas léxicas, sintácticas y semánticas, implementadas para desarrollar programas que le indican a la máquina qué hacer.

SCRATCH. Lenguaje de programación visual, dinámico, por bloques, que desarrolla habilidades mentales a través del aprendizaje de la programación sin contar con conocimientos previos sobre codificación.

2.4 Hipótesis de la Investigación

2.4.1 Hipótesis General

El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

2.4.2 Hipótesis Específicas

El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, según la dimensión conceptos computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, según la dimensión prácticas computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Capítulo III

Metodología

3.1 Enfoque de la investigación

Según Barrantes-Echevarría (2007), un enfoque es: "La manera en la que se plantea la resolución de un problema siguiendo grandes lineamientos y compromisos conceptuales" (pp. 57-58).

El proyecto actual está enfocado en una investigación cuantitativa, conformado por una secuencia de procesos lógicos, con instrumentos de medición como la encuesta, para demostrar con una serie de datos probabilísticos, el nivel de influencia que tiene la variable independiente sobre la dependiente.

Monje (2011), en su libro *Metodología de la investigación cuantitativa y cualitativa* menciona que:

> La investigación científica, desde el punto de vista cuantitativo, es un proceso sistemático y ordenado que se lleva a cabo siguiendo determinados pasos. Planear una investigación consiste en proyectar el trabajo de acuerdo con una estructura lógica de decisiones y con una estrategia que oriente la obtención de respuestas adecuadas a los problemas de indagación propuestos. Pese a tratarse de un proceso metódico y sistemático, no existe un esquema completo, de validez universal, aplicable mecánicamente a todo tipo de investigación. No obstante, si es posible identificar una serie de elementos comunes, lógicamente estructurados, que proporcionan dirección y guía en el momento de realizar una investigación, los cuales se pueden organizar en fases y etapas. (p. 10)

3.2 Tipo y alcance de la investigación

3.2.1 Tipo de investigación

Hernández et al. (2014) indican que: "La investigación científica cumple con dos propósitos fundamentales que a su vez sirven para clasificarla; estos son, producir conocimientos y teorías (investigación básica) y resolver problemas (investigación aplicada)" (p. xxiv).

Se ha definido el tipo de investigación como investigación aplicada, debido a que se busca resolver un problema del entorno que rodea al investigador en un área geográfica específica.

3.2.2 Alcance de la investigación

Bernal (2010) menciona que: "La investigación explicativa busca la comprobación de hipótesis para la determinación de leyes o principios. En este tipo de investigación se analizan causas y efectos de la relación entre variables" (p. 115).

Se ha definido el alcance de la investigación como explicativa, debido a que, se busca medir los niveles de Pensamiento Computacional que desarrollan los alumnos del primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, tras recibir clases del lenguaje de programación SCRATCH.

3.3 Diseño de la investigación

Para Bernal, los diseños experimentales consisten en: "Demostrar que la modificación de una variable (independiente) ocasiona un cambio predecible en otra (variable dependiente)" (2010, p. 145).

El diseño experimental aplicado en esta investigación es de tipo preexperimental. Hernández et al. (2014) definen el diseño preexperimental como: "Un grupo al que se le aplica una prueba previa al estímulo o tratamiento experimental, después se le administra el tratamiento y finalmente se le aplica una prueba posterior al estímulo" (p. 141).

Se ha definido el diseño de investigación como investigación preexperimental, debido a que se desea medir el impacto de la manipulación de la variable independiente (SCRATCH) sobre la variable dependiente. Adicionalmente, se está haciendo uso de un modelo de un solo grupo.

Tabla 2

Modelo preexperimental con un solo grupo

Grupo experimental	Aplicación del pretest o medición inicial	Aplicación del estímulo o tratamiento	Aplicación del postest o medición final
G	O1	X	O2

Nota: Hernández et al. (2014)

3.4 Descripción del ámbito de la investigación

La Universidad Católica Sedes Sapientiae tiene su sede principal y se ubica en la esquina de Constelaciones y Sol de Oro s/n, distrito de Los Olivos, en la provincia de Lima. La población fue representada por estudiantes del primer ciclo de la carrera de Ingeniería de Sistemas de la Facultad de Ingeniería.

Esta sede cuenta con dos (02) locales cuya infraestructura es de material noble y las condiciones de estudio son las adecuadas para el proceso y el desarrollo del aprendizaje.

3.5 Variables

3.5.1 Definición Conceptual

Variable Independiente: SCRATCH

Según el MIT (2013), SCRATCH es un lenguaje de programación visual y por bloques, que permite desarrollar capacidades mentales, a través de la programación, sin tener que contar con conocimientos sólidos, profundos, acerca de la misma.

Variable Dependiente: Pensamiento Computacional

Wing (2006) define el Pensamiento Computacional como “una forma de pensar que no es solo para programadores y consiste en la resolución de problemas, el diseño de sistemas, y la comprensión de la conducta humana haciendo uso de los conceptos fundamentales de la informática” (p. 33).

3.5.2 Definición Operacional

Variable Independiente: SCRATCH

Es un lenguaje de programación visual por bloques que ofrece herramientas clasificadas por categorías. Tiene un entorno amigable facilitando el aprendizaje de los fundamentos de la programación de una forma didáctica y dinámica.

Variable Dependiente: Pensamiento Computacional

El Pensamiento Computacional es una habilidad que toda persona debe desarrollar para solucionar problemas, comprende desde el análisis y formulación de problemas que se pueden automatizar, analizar y organizar datos de manera lógica para abstraerlos y representarlos en modelos y simulaciones mediante el pensamiento algorítmico, para luego generalizar y reutilizar ese proceso de solución para casos similares.

3.6 Operacionalización de las variables

VARIABLE	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES	TIPO Y ESCALA	CODIFICACIÓN	CRITERIOS DE MEDICIÓN	INSTRUMENTO	ITEMS
Pensamiento Computacional.	El pensamiento computacional es un proceso de solución de problemas con las siguientes características: Analizar y formular problemas que se puedan automatizar, analizar y organizar datos de manera lógica para abstraerlos y representarlos en modelos y simulaciones con el fin de automatizar soluciones mediante el pensamiento algorítmico, para luego generalizar y transferir ese proceso de solución de problemas a otros casos similares.	Conceptos computacionales	Las secuencias	Cuantitativa Ordinal	Correcto (1) Incorrecto (0)	Inicio (0-10) Proceso (11-15) Logrado (16-19)	**Cuestionario: Test del Pensamiento Computacional:** Conformado por 19 preguntas psicotécnicas.	1, 2, 3 y 4
			Los ciclos					5, 6, 7, 8 y 9
			Los datos					10
			Las condicionales					11, 12, 13, 14, 15 y 16
			Los operadores					17, 18 y 19
		Prácticas computacionales	Abstraer y modularizar.	Cuantitativa Ordinal	Correcto (1) Incorrecto (0)	Inicio (0-4) Proceso (5-7) Logrado (8-9)	**Cuestionario: Test del Pensamiento Computacional:** Conformado por 9 preguntas psicotécnicas.	20
			Ser incremental e iterativo.					21, 22 y 23
			Ensayar y depurar.					24
			Reusar y remezclar.					25, 26, 27 y 28

3.7 Delimitaciones

3.7.1 Temática

El tema de la presente investigación es determinar la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional de los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas.

3.7.2 Temporal

La presente investigación tendrá una duración de cuatro (04) semanas, se inició el 19 de setiembre y concluyó el 10 de octubre de 2018. En este periodo, se tomó inicialmente el pretest, se prosiguió con el desarrollo de las sesiones para impartir el estímulo y, finalmente, se tomó un postest.

3.7.3 Espacial

La presente investigación se llevó a cabo en la Facultad de Ingeniería, carrera de Ingeniería de Sistemas, Sede Lima de la Universidad Católica Sedes Sapientiae, ubicada en esquina Constelaciones y Sol de Oro s/n Urb. Sol de Oro, Los Olivos.

3.8 Limitaciones

Algunos alumnos de primer ciclo participantes en la investigación no contaban con el nivel de manejo de las tecnologías necesarias para llevar a cabo el experimento.

Para resolver situaciones de aprendizaje básicas medidas por SCRATCH, es requisito tener cierto grado de desarrollo del pensamiento lógico para dominar el proceso de programación de computadoras.

La falta de compromiso, por parte de algunos alumnos, para desarrollar los ejercicios durante las sesiones de clase del lenguaje de programación SCRATCH.

3.9 Población y muestra

3.9.1 Población

Para Hernández et al. (2014), es el "Conjunto de todos los casos que concuerdan con determinadas especificaciones" (p. 174).

Por ello, para la presente investigación, se delimitó como población a todos los estudiantes pertenecientes al primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae (Sede Lima).

Tabla 3

Población

Ciclo	Cantidad
I	46 alumnos

3.9.2 Muestra

Para Hernández et al. (2014), "la muestra es un subgrupo de la población de interés sobre el cual se recolectarán datos, y que tiene que definirse o delimitarse de antemano con precisión, éste deberá ser representativo de dicha población" (p. 173).

La fórmula que se establece para la determinación del tamaño de muestra en base a la población es la siguiente:

$$n = \frac{Z^2 N}{Z^2 + 4N(EE)^2}$$

Donde:

n = Tamaño de muestra

Z = Nivel de confianza

N = Tamaño de la población

EE = Error estimado

Cálculo del tamaño de muestra

$$n = \frac{1.95^2 . 46}{1.95^2 + 4.46 . (0.05)^2}$$

$$n = 41$$

Al aplicar la fórmula del cálculo de muestra, se obtiene que para una población de cuarenta y seis (46) estudiantes, se obtuvo una muestra de cuarenta y uno (41). La técnica de muestreo será no probabilística por conveniencia del investigador, de este modo, se trabajará con la totalidad de la población como grupo muestral.

3.10 Técnicas e instrumentos para la recolección de datos

3.10.1 Técnicas

La técnica seleccionada para la recolección de datos en la presente investigación ha sido la encuesta.

Según Hernández et al. (2014): "Las encuestas son cuestionarios que se proporcionan directamente a los participantes, quienes lo contestan, sin necesidad de intermediarios y las respuestas las marcan ellos mismos" (p. 235).

3.10.2 Instrumentos

Para Hernández et al. (2014), "Un cuestionario consiste en un conjunto de preguntas respecto de una o más variables a medir" (p. 217).

El instrumento que se ha elegido debido a la naturaleza de la técnica de recolección de datos ha sido el cuestionario: Evaluación o Test. Este está conformado de veintiocho (28) preguntas pictográficas con cuatro (04) alternativas cada una, resultado único y valoración dicotómica, las cuales están realizadas con la finalidad de recolectar la información necesaria para medir el nivel del Pensamiento Computacional en sus diferentes dimensiones.

3.10.2.1 Pretest

Según Heinemann (2003), el pretest es un cuestionario que se realiza sobre una población de estudio al comienzo de la investigación con el objetivo de verificar la validez del instrumento, su credibilidad, su propósito y su uso práctico.

3.10.2.2 Postest

Es el mismo cuestionario inicial pero realizado al final de la investigación, esto permite examinar y comparar el impacto generado en la forma como los estudiantes procesan la información tras recibir un estímulo (Heinemann, 2003).

3.11 Validez y confiabilidad del instrumento

3.11.1 Validez

Hernández et al. manifiestan que: "Para medir una variable se necesita un instrumento el cual establezca el grado real de lo que se desea medir" (2006, p. 314).

Otro autor afirma lo siguiente:

> La validez y confiabilidad son: "constructos" inherentes a la investigación, desde la perspectiva positivista, con el fin de otorgarle a los instrumentos y a la información recabada, exactitud y consistencia necesarias para efectuar las generalizaciones de los hallazgos, derivadas del análisis de las variables en estudio. (Hidalgo, 2005, p. 2)

Para la validación del instrumento se ha tomado en consideración la valoración de juicio de tres (03) expertos, las cuales se han ubicado entre 80% a 98% de validez y han sido enfocadas en los 28 ítems con respecto a los conceptos y prácticas del Pensamiento Computacional.

El instrumento utilizado en esta investigación es el TPC (Test de Pensamiento Computacional), creado por Marcos en el 2016 y expuesto en su tesis doctoral "Codigoalfabetización y Pensamiento Computacional en Educación Primaria y Secundaria:

Validación de un instrumento y evaluación de programas", fue sometido a un juicio de 20 expertos, conformado en su gran mayoría por docentes de Informática de nivel universitario y escolar, concluyeron que de las 40 preguntas planteadas, se tenía que depurar a 28 preguntas excluyendo las más complejas por el nivel al que iba dirigido y puedan ser completados en una sesión de clase de dos horas académicas.

El instrumento utilizado en la presente investigación se conforma de 28 preguntas pictográficas graduales seleccionadas del TPC de Román, de donde se eligen las más complejas, para que sean completadas en una sesión de dos horas académicas.

Con base en los juicios de los tres especialistas profesionales conocedores del tema de investigación, se efectuó la validación del instrumento, obteniendo como resultado los siguientes porcentajes:

Tabla 4

Promedio de calificación al Test de Pensamiento Computacional

N°	Especialista	%	Opinión
1	Mg. Bizarro Tapara Raúl	93	Aplicable
2	Mg. Elescano Córdova Wilfredo Clemente	93	Aplicable
3	Mg. José Manuel Huamán Gutiérrez	93	Aplicable
	Total	93	

Nota: Ficha de validación de expertos.

3.11.2 Confiabilidad

Con respecto a la confiabilidad, se aplicó la prueba de Alfa de Cronbach, en la cual se obtuvo un nivel de correlación de 0.835 con la variable dependiente.

$$\alpha = \frac{K}{K-1}\left[1 - \frac{\sum {S_i}^2}{{S_T}^2}\right]$$

Donde:

K: El número de ítems

S_i^2: Sumatoria de varianzas de los ítems

S_t^2: Varianza de la suma de los ítems

α: Coeficiente de Alfa de Cronbach

Tabla 5

Confiabilidad del instrumento por RK - 20

RK 20	N de elementos
0.835	28

Nota: Se demuestra como el valor del Rk - 20 es superior a 0.6 entonces, se puede afirmar que el instrumento es confiable para los fines de la presente investigación.

3.12 Plan de recolección y procesamiento de datos

3.12.1 Plan de recolección

El procedimiento bajo el cual se llevará a cabo la aplicación de los instrumentos en este estudio de enfoque cuantitativo se explica a continuación:

En primera estancia, se seleccionará el grupo sujeto de estudio, los cuales pertenecen a los estudiantes del primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Al iniciar la investigación se aplicará un pretest, para evaluar el estado en que se encontraban los estudiantes al comienzo del proceso investigativo.

Dentro de la asignatura de Introducción a la Ingeniería de Sistemas se incluyeron siete (07) sesiones del uso del lenguaje de programación SCRATCH, donde se crearon proyectos, con el objetivo de alcanzar aprendizajes significativos en conceptos y prácticas computacionales para potenciar el desarrollo del Pensamiento Computacional.

Al finalizar la investigación, se aplicó el postest con el objetivo de comparar cómo SCRATCH desarrolla el Pensamiento Computacional en los estudiantes participantes de la investigación.

3.12.2 Procesamiento de datos

Los resultados obtenidos de las pruebas de pretest y postest requieren ser procesados para su análisis e interpretación, con respecto a las respuestas, los puntajes obtenidos y el nivel de desarrollo del Pensamiento Computacional que estos resultados reflejan.

Para analizar los datos recolectados, se utilizó la estadística descriptiva a través de sus recursos como son gráficos de barras, tablas de frecuencia y la estadística inferencial. Del mismo modo, para la generación de cuadros, tablas y reportes, se utilizó el software estadístico SPSS en su versión 26 y la herramienta de cálculo Microsoft Excel.

Capítulo IV: Desarrollo de la investigación

4.1 Resultados de la Variable Dependiente: Pensamiento Computacional

Tabla 6

Datos por niveles del pretest del Pensamiento Computacional

Niveles	Estudiantes	% del pretest
Inicio	8	17%
Proceso	33	72%
Logrado	5	11%

Nota: De acuerdo al resultado, luego de aplicar el pretest de la variable dependiente, se puede observar que en el nivel inicio se obtuvo un 17%, en el nivel proceso un 72% y sólo un 11% en el nivel logrado.

Tabla 7

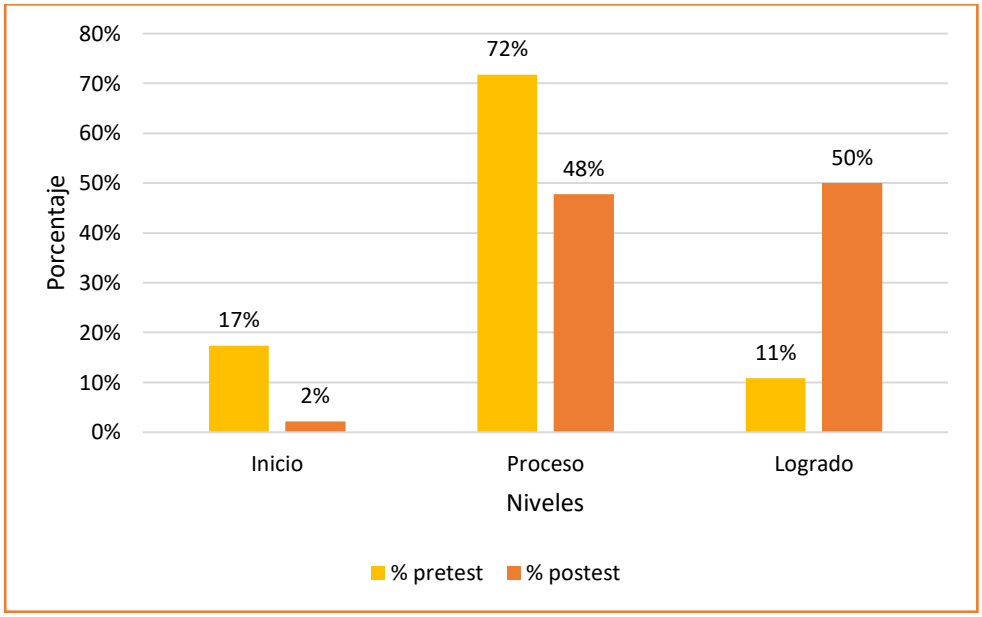
Datos por niveles del postest del Pensamiento Computacional

Nota: De acuerdo al resultado, luego de aplicar el postest de la variable dependiente, se puede observar que en el nivel inicio se obtuvo sólo un 2%, en el nivel proceso un 48% y un 50% en el nivel logrado.

Niveles	Estudiantes	% del postest
Inicio	1	2%
Proceso	22	48%
Logrado	23	50%

Figura 10

Comparación del pretest y postest del Pensamiento Computacional



Nota: Distribución por niveles del pretest y postest del Pensamiento Computacional.

De acuerdo al resultado obtenido, luego de aplicar el pretest y el postest de la variable dependiente, se puede observar en el gráfico de comparación, que el nivel *inicio* tuvo una reducción de 15%, en el nivel p*roceso* también se redujo un 24% y en el nivel *logrado* hubo un incremento de 39% en el desarrollo del Pensamiento Computacional en los alumnos de primer ciclo de la carrera de Ingeniería de Sistemas. En el nivel *inicio*, se disminuyó hasta quedar únicamente un 2% de los alumnos y, en el nivel *proceso*, se redujo hasta quedar el 48%. Esto evidencia que la mitad de los alumnos alcanzó obtener el nivel *logrado*.

Según Wing (2011), el Pensamiento Computacional implica la capacidad para la formulación y la resolución de problemas, representados de forma que sean ejecutados por un procesador de información.

4.2 Resultados de la Dimensión: Conceptos computacionales

Tabla 8

Datos por niveles del pretest de la dimensión: Conceptos computacionales

Niveles	Alumnos	% del pretest
Inicio	5	11%
Proceso	21	46%
Logrado	20	43%

Nota: De acuerdo al resultado, luego de aplicar el pretest de la dimensión: Conceptos computacionales, se puede observar que en el nivel inicio se obtuvo un 11%, en el nivel proceso un 46% y un 43% en el nivel logrado.

Tabla 9

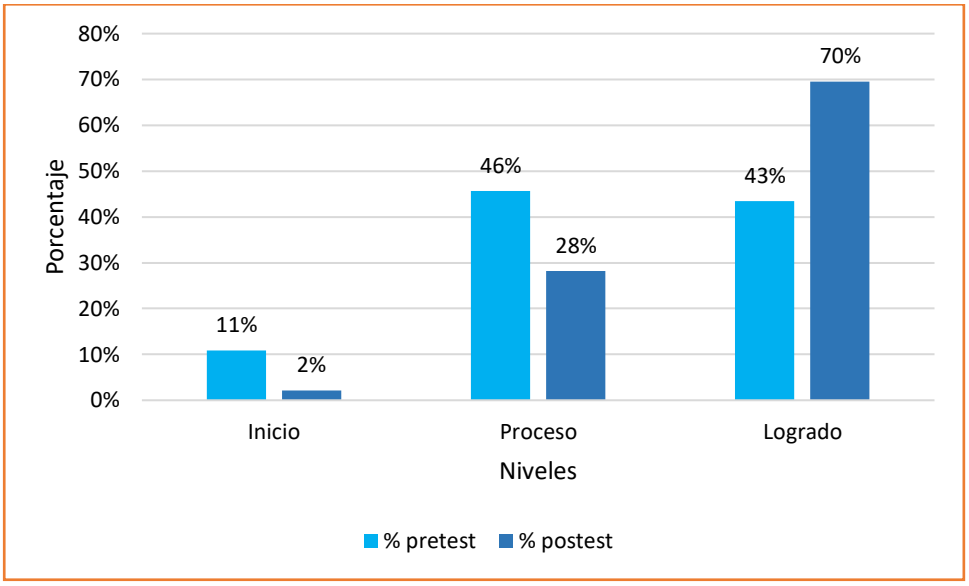
Datos por niveles del postest de la dimensión: Conceptos computacionales

Niveles	Alumnos	% del postest
Inicio	1	2%
Proceso	13	28%
Logrado	32	70%

Nota: De acuerdo al resultado, luego de aplicar el postest de la dimensión: Conceptos computacionales, se puede observar que en el nivel inicio se obtuvo sólo un 2%, en el nivel proceso un 28% y un 70% en el nivel logrado.

Figura 11

Comparación del pretest y postest de la dimensión: Conceptos computacionales



Nota: Datos del pretest y postest de la dimensión: Conceptos computacionales.

De acuerdo al resultado obtenido, luego de aplicar el pretest y postest a la dimensión Conceptos computacionales, podemos observar en el gráfico de comparación que el nivel *inicio* tuvo una reducción de 9%, en el nivel *proceso* también se redujo un 18% y en el nivel *logrado* hubo un incremento de 27% en el desarrollo de la dimensión Conceptos computacionales en los alumnos de primer ciclo de la carrera de Ingeniería de Sistemas. En el nivel *inicio*, se disminuyó hasta quedar únicamente un 2% de los alumnos y en el nivel *proceso* se redujo hasta quedar el 28%. Esto evidencia que más de la mitad de los alumnos obtuvo el nivel *logrado*.

Según Resnick (2012), los conceptos computacionales como Secuencias, ciclos, eventos, paralelismo, condicionales, operadores, datos, son necesarios para dar solución a un problema determinado, ya sea computacional o de otro contexto.

4.3 Resultados de la Dimensión: Prácticas computacionales

Tabla 10

Datos por niveles del pretest de la dimensión: Prácticas computacionales

Niveles	Alumnos	% del pretest
Inicio	22	48%
Proceso	21	46%
Logrado	3	6%

Nota: De acuerdo al resultado, luego de aplicar el pretest de la dimensión: Prácticas computacionales, se puede observar que en el nivel inicio se obtuvo un 48%, en el nivel proceso un 46% y sólo un 6% en el nivel logrado.

Tabla 11

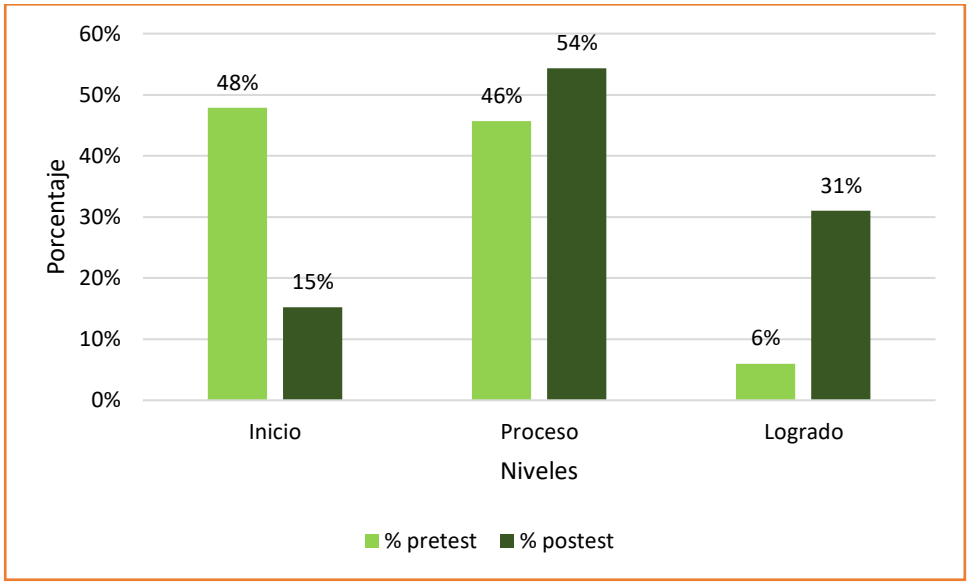
Datos por niveles del postest de la dimensión: Prácticas computacionales

Niveles	Alumnos	% del posttest
Inicio	7	15%
Proceso	25	54%
Logrado	14	31%

Nota: De acuerdo al resultado, luego de aplicar el postest de la dimensión: Prácticas computacionales, se puede observar que en el nivel inicio se obtuvo un 15%, en el nivel proceso un 54% y un 31% en el nivel logrado.

Figura 12

Comparación del pretest y postest de la dimensión: Prácticas computacionales



Nota: Distribución por niveles del pretest y postest de la dimensión Prácticas computacionales.

De acuerdo al resultado obtenido, luego de aplicar el pretest y postest a la dimensión Prácticas computacionales, podemos observar en el gráfico de comparación que el nivel *inicio* tuvo una reducción de 33%, en el nivel *proceso* hubo un incremento de 8% y en el nivel *logrado* hubo un incremento de 25% en el desarrollo de la dimensión Prácticas computacionales en los alumnos de primer ciclo de la carrera de Ingeniería de Sistemas. En el nivel *inicio* se disminuyó hasta quedar únicamente un 15% de los alumnos y en el nivel *proceso* hubo incrementó hasta lograr el 54%, mientras que en el nivel *logrado* se obtuvo un valor final de 31%.

Según Resnick (2012), las Prácticas computacionales como son Ser incremental y repetitivo, ensayar y depurar errores, reusar y remezclar y abstraer y modularizar; son prácticas que deben ser adquiridas por los estudiantes para entender, plantear y resolver problemas de cualquier contexto.

4.4 Prueba de Hipótesis

Prueba de normalidad

HO: Los datos tiene una distribución normal

Tabla 12

Prueba de normalidad

	KOLMOGOROV SMIRNOV		
	Estadístico	gl	Sig.
PENSAMIENTO	,163	90	,000
CONCEPTOS	,125	90	,000
PRACTICA	,144	90	,000

Nota: Si el P valor (sig.) es menor que 0.05 entonces se rechaza la hipótesis nula y se concluye que los datos no tienen distribución normal y, por lo tanto, se utilizará la prueba no paramétrica de wilconxo.

Prueba de hipótesis general

H_0: Lenguaje de programación SCRATCH no influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

H_1: El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Tabla 13

Prueba de hipótesis general

Hipótesis nula	Prueba	Sig.	Decisión
La mediana de las diferencias entre PENSAMIENTO y PENSAMIENTO es igual a 0.	Prueba de Wilcoxon de los rangos con signo para muestras relacionadas	.000	Rechace la hipótesis nula

Nota: Se muestran significaciones asintóticas. El nivel de significancia es .05. Como el P valor es menor que 0.05 entonces se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Prueba de hipótesis específicas

Hipótesis específica 1

H_0: El lenguaje de programación SCRATCH no influye en el desarrollo del Pensamiento Computacional, según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

H_1: El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Tabla 14

Prueba de hipótesis de la dimensión 1

Hipótesis nula	Prueba	Sig.	Decisión
La mediana de las diferencias entre CONCEPTO y CONCEPTO es igual a 0.	Prueba de Wilcoxon de los rangos con signo para muestras relacionadas	.000	Rechace la hipótesis nula

Nota. Se muestran significaciones asintóticas. El nivel de significancia es .05. Como el P valor es menor que 0.05 entonces se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Hipótesis específica 2

H_0: El lenguaje de programación SCRATCH no influye en el desarrollo del Pensamiento Computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

H_1: El lenguaje de programación SCRATCH si influye en el desarrollo del Pensamiento Computacional, según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Tabla 15

Prueba de hipótesis de la dimensión 2

Hipótesis nula	Prueba	Sig.	Decisión
La mediana de las diferencias entre PRÁCTICA y PRÁCTICA es igual a 0.	Prueba de Wilcoxon de los rangos con signo para muestras relacionadas	.000	Rechace la hipótesis nula

Nota. Se muestran significaciones asintóticas. El nivel de significancia es .05. Como el P valor es menor que 0.05 entonces se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Capítulo V

Discusión, conclusiones y recomendaciones

5.1 Discusión

La finalidad de esta investigación fue medir la influencia del lenguaje de programación SCRATCH, en el nivel de desarrollo del Pensamiento Computacional de los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Respecto a la hipótesis general "El lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae", en los resultados referentes a la medición del nivel del Pensamiento Computacional, se evidenció una mejora notable en el nivel *logrado* al comparar el pretest con el postest, notándose un aumento significativo de 11% a 50% tras las sesiones de SCRATCH, del mismo modo, en el nivel *proceso* se redujo de 72% a 48% y en el nivel *inicio*, se redujo de 17% a 2%.

Por otro lado, al realizar el análisis paramétrico, el P valor resultó 0.000, menor que 0.05, por lo cual, se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Según Barrera y Montaño (2015), "el pensamiento computacional refuerza los estándares educativos en todas las asignaturas para acrecentar la habilidad del estudiante de solucionar problemas y así desarrollar el pensamiento de orden superior" (p. 10). Esto constata lo expuesto por Vera (2019), donde demuestra que la capacitación en algorítmica, utilizando el lenguaje de

programación SCRATCH influyó positivamente en el desarrollo de las habilidades del Pensamiento Computacional en los estudiantes de primer ciclo que fueron capacitados.

Según Pérez (2017), en su investigación sobre el desarrollo del Pensamiento Computacional en los estudiantes de primer semestre de la carrera de Informática de la Facultad de Filosofía de la Universidad Central de Ecuador, mediante el uso de la herramienta Scratch como recurso didáctico para su formación profesional, concluye que no hubo una mejora sustancial en el desarrollo de esta competencia.

Respecto a la hipótesis específica 1, "El lenguaje de programación SCRATCH sí influye en el desarrollo del pensamiento computacional según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae", en el pretest el 11% de los estudiantes se ubicó en el nivel *inicio,* el 46% en nivel *proceso* y el 43% en el nivel *logrado*; resultados que contrastan con los obtenidos en el postest, donde únicamente el 2% de los estudiantes se ubicó en el nivel *inicio*, observándose una reducción significativa con respecto al pretest, en el nivel *proceso* se redujo a 28%, mientras que, en el nivel *logrado* se observó un incremento a 70% de los estudiantes.

Por otro lado, al realizar el análisis paramétrico, el P valor obtenido fue menor que 0.05, por lo cual, se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional, según la dimensión conceptos computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Brennan y Resnick (2012) identificaron algunos conceptos que resultan útiles para una gran variedad de proyectos con Scratch y que pueden aplicarse también a otros contextos, ya sean de programación o no. Esto verifica lo expuesto por Hamilton (2017) donde indica que el uso del

lenguaje de programación SCRATCH en estudiantes ayuda a conseguir mejoras significativas en las dimensiones de variables, secuencias, ciclos y operadores, las cuales en la presente tesis han sido incluidas como indicadores de la dimensión de conceptos computacionales.

Respecto a la hipótesis específica 2 "El lenguaje de programación SCRATCH sí influye en el desarrollo del pensamiento computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae", en el pretest el 48% de los estudiantes se ubicó en el nivel *inicio*, el 46% en nivel *proceso* y el 6% en el nivel *logrado*; resultados que contrastan con los obtenidos en el postest, donde únicamente el 15% de los estudiantes se ubicó en el nivel *inicio*, observándose una reducción significativa con respecto al pretest, en el nivel *proceso* se observó un incremento a 54%, mientras que, en el nivel *logrado* se observó un incremento a 31% de los estudiantes.

Por otro lado, al realizar el análisis paramétrico, el P valor obtenido fue menor que 0.05, por lo cual, se rechaza la hipótesis nula y se concluye que el lenguaje de programación SCRATCH sí influye en el desarrollo del Pensamiento Computacional según la dimensión Prácticas computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Brennan y Resnick (2012) afirman que "las prácticas computacionales se enfocan en el proceso de pensar y de aprender y van más allá de qué está usted aprendiendo para centrarse en cómo lo está aprendiendo" (p. 8). Esto respalda lo planteado por Condo (2019), donde expone que las prácticas computacionales están relacionadas significativamente con las prácticas de programación enfocándose en el "cómo se aprende".

5.2 Conclusiones

En los niveles educativos como primaria, secundaria, técnico y universitario tienen problemas en el aprendizaje de sus competencias, debido a la falta de desarrollo del Pensamiento Computacional.

Las facultades de Educación o Instituciones Pedagógicas, actualmente, no contemplan en los futuros docentes el desarrollo de estas capacidades, con el fin de que sean difundidas y desarrolladas en los estudiantes.

Primero. En esta investigación, se determinó la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, demostrándose la importancia del Pensamiento Computacional, (capacidad de pensar por cuenta propia, analizando y evaluando la situación, fenómeno o problema), como una competencia que debe ser adquirida no solo por los alumnos, sino por todos. El Pensamiento Computacional no solo es una competencia del entorno de Tecnologías de la Información y Comunicación, es una competencia que debe ser adquirida por todas las disciplinas del sector educativo. La plataforma digital "SCRATCH" impulsa el aprendizaje de esta competencia con mayor rapidez, gracias a la interactividad que ofrece frente a otras plataformas de programación. El Estado Peruano no toma como prioridad el desarrollo del Pensamiento Computacional y no lo declara como una competencia necesaria que debería ser adquirida por todos los estudiantes, tal como lo hacen otros países, dificultando con ello su aprendizaje. Según la Currícula Nacional de la Educación Básica (2016), en la competencia transversal N° 28, se expresa el desenvolvimiento en los entornos virtuales generados por las TIC para la interacción y elaboración de materiales digitales. Sin embargo, no hace referencia al desarrollo del Pensamiento Computacional.

Segundo. En esta investigación, se determinó la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, según la dimensión conceptos computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae. La aplicación de los conceptos computacionales (Secuencias, ciclos, eventos, paralelismo, condicionales, operadores y datos) desarrolló en los estudiantes el pensamiento crítico, lógico, algorítmico y creativo que son habilidades necesarias para la resolución de problemas de cualquier contexto.

Tercero. En esta investigación, se determinó la influencia del lenguaje de programación SCRATCH en el desarrollo del Pensamiento Computacional, según la dimensión prácticas computacionales, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae. La aplicación de las prácticas computacionales (Ser Incremental e Iterativo, Ensayar y Depurar, Reusar y Remezclar, Abstraer y Modularizar) permitió plantear soluciones utilizando la abstracción que consiste en enfocarse, en primera instancia, en lo esencial e incrementar los detalles de forma iterativa en el diseño e implementación de proyectos, reutilizando patrones existentes y modularizando la solución que estará en constante ensayo y depuración para corregir posibles errores en el proyecto antes de su implantación.

5.3 Recomendaciones

Las recomendaciones planteadas van en ese sentido, la de impulsar, promover e implementar acciones necesarias para que toda institución educativa primaria, secundaria, superior y por qué no, el Estado mediante una política de gobierno pueda tomar estas recomendaciones.

Primero. Se recomienda el uso de la plataforma del lenguaje de programación SCRATCH para impulsar el desarrollo del nivel de Pensamiento Computacional en los estudiantes de los

primeros ciclos en las carreras de Ingeniería, y en las demás especialidades de la Universidad Católica Sedes Sapientiae, para la implementación de proyectos que planteen soluciones a problemas de cualquier índole, incrementando la motivación, la creatividad y el interés de los estudiantes.

Segundo. Se recomienda el uso de la plataforma de lenguaje de programación SCRATCH en los niveles educativos como primaria, secundaria y superior, de manera continua y progresiva, incluyéndolo en los planes de estudios, con el fin de potenciar las habilidades del Pensamiento Computacional en los estudiantes.

Tercero. Se recomienda incorporar un curso en el primer ciclo de la carrera de Ingeniería de Sistemas y en las demás especialidades de la Facultad de Ingeniería, introduciendo el lenguaje de programación SCRATCH para iniciar en los estudiantes los conceptos básicos del Pensamiento Computacional.

Cuarto. Se recomienda a toda persona del entorno académico o no, que desee entender los cambios y/o problemas que surgen a diario que requieren una solución, que adquiera la competencia del Pensamiento Computacional, el cual les ayudará a comprender, analizar, diseñar y brindar una solución a un problema de cualquier índole.

Quinto. Se recomienda impulsar el aprendizaje del Pensamiento Computacional en la Facultad de Educación con el propósito de que todo docente implemente dicho aprendizaje en su centro de labores.

Sexto. Se recomienda impulsar el aprendizaje del Pensamiento Computacional en la Universidad Católica Sedes Sapientiae, a fin de sea un centro de referencia y capacitación de esta competencia para todas las instituciones educativas de los diferentes niveles.

Séptimo. Se recomienda que la Universidad Católica Sedes Sapientiae elabore y presente un plan de enseñanza del Pensamiento Computacional a la SUNEDU y al MINEDU, a fin de que el aprendizaje de dicha competencia sea un tema de interés nacional para la formación de los estudiantes del Perú y así acortar la brecha educativa que existe en el Perú en referencia a sus pares latinoamericanos y a nivel global.

Referencias

Avalos, F. (2017). *El software de programación "SCRATCH", para desarrollar el pensamiento creativo en estudiantes del 5to grado de secundaria de la IE "Melchorita Saravia" – Grocio Prado–2017* [Tesis de Maestría, Universidad César Vallejo].

Barrera, R., & Montaño, R. (2015). *Desarrollo del Pensamiento Computacional con SCRATCH Nuevas Ideas en Informática Educativa TISE 2015* (pp. 616-620).

Bawden, D. (2001). Information and digital literacies: a review of concepts. *Journal of Documentation*, 57(2), 218–259.

Blanco, D. (2014). *Implementación de SCRATCH para potenciar el aprendizaje significativo a través lógica de programación en los estudiantes de Nivel Básica Secundaria* [Tesis de Maestría, Tecnológico de Monterrey].

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2017). El Pensamiento Computacional en la Enseñanza Obligatoria (Computhink)–Implicaciones para la política y la práctica.

Brennan, K., Chung, M., & Hawson, J. (2011). *Computación creativa: Una introducción al pensamiento computacional orientado al diseño*.

Brennan, K., & Resnick, M. (2012). Nuevos marcos de referencia para estudiar y evaluar el desarrollo del pensamiento computacional.

Cearreta, I. (2015). *SCRATCH como recurso didáctico para el desarrollo del Pensamiento Computacional de los alumnos de Secundaria y Bachillerato en la asignatura de Informática y como recurso transversal en el resto de asignaturas* [Tesis de Maestría, Universidad Internacional de la Rioja].

Chancolla, G., & Pacori, E. (2016). *El uso del software SCRATCH para mejorar el pensamiento computacional en los estudiantes del quinto grado de primaria de la Institución Educativa n° 40009 San Martín de Porres del distrito de Paucarpata* [Tesis de grado, Universidad Nacional de San Agustín de Arequipa].

Chun, B., & Piotrowski, T. (s.f.). *Pensamiento computacional ilustrado: Una guía de dibujos animados para solucionar problemas, diseñar sistemas y comprender el comportamiento humano.*

Condo, A. (2017). *El pensamiento computacional en estudiantes del VII ciclo de la institución educativa particular "Ricardo Palma"- San Juan de Miraflores 2016* [Tesis de grado, Universidad César Vallejo].

Condo, A. (2019). *Uso de la plataforma Arduino en la mejora del pensamiento computacional, en la Institución Educativa Privada Ricardo Palma, año 2019* [Tesis de Maestría, Universidad César Vallejo].

Cori, A. (2017). *Aplicación de una herramienta de programación para mejorar el pensamiento computacional en estudiantes universitarios de Tacna* [Tesis Doctoral, Universidad Nacional Jorge Basadre Grohmann de Tacna].

CSTA & ISTE. (2011). *Operational definition of computational thinking*. Disponible en: http://www.iste.org/docs/ct-documents/computational-thinkingoperational-definition-flyer.pdf . [Consultado en octubre de 2017].

CSTA & ISTE. (2016). *Caja de herramientas para líderes en Pensamiento Computacional*. Disponible en: http://eduteka.icesi.edu.co/pdfdir/PensamientoComputacional1.pdf. [Consultado en octubre de 2017].

EDUCAR (2015). *El CFE declaró de importancia estratégica a la enseñanza y el aprendizaje de la Programación*. Disponible en: https://www.educ.ar/noticias/127730/el-cfe-declaro-de-importancia-estrategica-a-la-ensenanza-y-el-aprendizaje-de-la-programacion. [Consultado en setiembre de 2017].

Flores, C., & Maldonado, S. (2012). El Software Educativo SCRATCH Aplicado como Herramienta Transversal en el Currículo Educativo del Colegio Mayor San Lorenzo *Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, 8, ISSN 20007-2619.

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK schools. Technical report.* Londres: The Royal Society.

Gilster, P. (1997). *Digital Literacy*. New York: Wiley Computer Pub.

González, H. (2003). *Capacidades Intelectuales de Orden Superior.* Disponible en: http://eduteka.icesi.edu.co/articulos/capacidades-intelectuales-orden-superior. [Consultado en setiembre de 2017].

González, P. (2014). *Manual de SCRATCH 2: Manual para la asignatura de Informática.*

Hernández R, Baptista P & Fernández C. (2000). *Metodología de la Investigación.* Editorial Mc Graw Hill Education.

Hidalgo, L. (2005). *Validez y confiabilidad en la investigación cualitativa.* Disponible en: www.ucv.ve/uploads/media/Hidalgo2005.pdf . [Consultado en enero de 2017].

Hitschfeld, N., Pérez, J., & Simmonds, J. (2015). Pensamiento Computacional y programación a nivel escolar en Chile: El valor de formar a los innovadores tecnológicos del futuro *Bits de ciencia*, 12, 28 – 32.

Joyanes, L. (2008). *FUNDAMENTOS DE PROGRAMACIÓN: Algoritmos, estructura de datos y objetos*. Madrid: McGraw-Hill Interamericana de España.

López, J.C. (2009). *Algoritmos y Programación (Guía para docentes)*. Disponible en: http://www.eduteka.org/GuiaAlgoritmos.php. [Consultado en setiembre 2017].

Moursund, D. (2006): *Computational Thinking and Math Maturity: Improving Math Education in K-8 Schools*. Disponible en: http://uoregon.edu/~moursund/Books/ElMath/ElMath.html . [Consultado octubre 20016].

Paul, R., & Elder, L. (2003). *La mini-guía para el Pensamiento crítico: Conceptos y herramientas*. Disponible en https://www.criticalthinking.org/resources/PDF/SP-ConceptsandTools.pdf. [Consultado noviembre 2017].

Pérez, H. (2017). *Uso de SCRATCH como herramienta para el desarrollo del pensamiento computacional en Programación I de la carrera de Informática de la Universidad Central del Ecuador* [Tesis Doctoral, Universidad de Alicante].

Pérez, M. (2017). El pensamiento computacional para potenciar el desarrollo de habilidades relacionadas con la resolución creativa de problemas *3C TIC: Cuadernos de desarrollo aplicados a las TIC*, 6(1), 38-63.

Román, M. (2016). *Codigoalfabetización y pensamiento computacional en educación primaria y secundaria: validación de un instrumento y evaluación de programas* [Tesis Doctoral, Universidad de Educación a Distancia. España]

Sáes, J., & Cózar, R. (2016). Pensamiento computacional y programación visual por bloques en el aula de Primaria *Educar 2017*, 53/1, 129-146

Senge, P. (2010). *LA QUINTA DISCIPLINA: Cómo impulsar el aprendizaje en la organización inteligente*. Ediciones Granica.

Usman, S. (2013). *Aplicación de entornos elaborados con herramientas digitales gráficas animadas, para el desarrollo y fortalecimiento de habilidades de pensamiento de orden*

superior en el área de matemáticas de una Institución Educativa de la ciudad de Palmira [Tesis de Maestría, Universidad Nacional de Colombia].

Vera, H. (2019). *Influencia de la capacitación en Algorítmica en el nivel de pensamiento computacional de los estudiantes de la Universidad Global del Cusco 2018* [Tesis doctoral, Universidad Global del Cusco].

Vidal, C., Cabezas, C., Parra, J., & López, L. (2015). Experiencias Prácticas con el Uso del Lenguaje de Programación SCRATCH para Desarrollar el Pensamiento Algorítmico de Estudiantes en Chile *Formación Universitaria*, 8(4), 23-32.

Wing, J. (2006). Computational thinking, *Comm of ACM*, 49 (3), 33-35.

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society A*, 3717-3725.

Wing, J, (2008). Five Deep questions in computing, *Comm of CACM*, 51(1), 58-60.

Wing, J. (2010). Computational thinking: What and Why?, *Comm of CACM*

Wing, J. (2016). Computational thinking, 10 years later, *Comm of CACM*

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital *RED Revista de Educación a Distancia*. 1(40), 1-47.

Zapata-Ros, M. (2018). Pensamiento computacional: Una tercera competencia clave *RED Revista de Educación a Distancia*. 4(46), 4-87.

Zapotecatl, J. (2014). *Pensamiento Computacional*. Puebla: Academia Mexicana de Computación, A.C.

Zapotecatl, J. (2018). *Introducción al Pensamiento Computacional: Conceptos Básicos para Todos*. México: Academia Mexicana de Computación, A.C.

Zúñiga, M., Rosas, M., Fernández, J., & Guerrero, R. (2014). El Desarrollo del Pensamiento computacional para la Resolución de Problemas en la Enseñanza Inicial de la Programación *WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación* (pp. 340-343).

Anexos

6.1. Anexo 1: Matriz de Consistencia

<table>
<tr><td colspan="7">MATRIZ DE CONSISTENCIA</td></tr>
<tr><td colspan="7">Título de la investigación: Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, 2018.</td></tr>
<tr><td>Formulación del problema</td><td>Objetivos</td><td>Hipótesis de investigación</td><td colspan="3">Variable</td><td>Metodología de investigación</td></tr>
<tr>
<td><u>Problema general:</u>
¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?

<u>Problemas específicos:</u>
1.1. ¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del pensamiento computacional según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?

1.2. ¿En qué medida el lenguaje de programación SCRATCH influye en el desarrollo del pensamiento computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae?</td>
<td><u>Objetivo general:</u>
Determinar la Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

<u>Objetivos específicos:</u>
1.1 Determinar la Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

1.2 Determinar la Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.</td>
<td><u>Hipótesis general:</u>
El lenguaje de programación SCRATCH si influye en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

<u>Hipótesis específicas:</u>
1.1 El lenguaje de programación SCRATCH si influye en el desarrollo del pensamiento computacional según la dimensión concepto, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

1.2 El lenguaje de programación SCRATCH si influye en el desarrollo del pensamiento computacional según la dimensión práctica, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.</td>
<td><u>Tipos:</u>

Variable Independiente:
Lenguaje de programación SCRATCH

Variable Dependiente:
Pensamiento Computacional</td>
<td><u>Dimensiones:</u>

I. Sentencias entrada, salida (Secuencia)

II. Operaciones y variables.

III. Sentencias de decisión.

IV. Sentencias de repetición.

V. Funciones y Procedimientos

I. Conceptos computacionales

II. Prácticas computacionales</td>
<td><u>Indicadores:</u>

Planificación e intervención educativa.

I. Las secuencias.
I. Los ciclos
I. Los datos
I. Las condicionales
I. Los operadores

II. Abstraer y modularizar.
II. Ser incremental e iterativo.
II. Ensayar y depurar</td>
<td>Enfoque:
Cuantitativo
Tipo:
Aplicada
Alcance:
Explicativo
Diseño:
Experimental (Pre-experimental)
Población:
La presente investigación está conformada por los estudiantes del primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.

Muestra:
Cuarenta y seis (46) estudiantes pertenecientes al primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae.
<u>Receptor de la acción</u>:
Género: masculino y femenino
Edad: 17 - 20
Nivel: Universitario
Centro:
Universidad Católica Sedes Sapientiae.
Ubicación:
Los Olivos – Lima, Perú
Técnica:
Encuesta.</td>
</tr>
</table>

					II. Reusar y remezclar.	**Instrumentos:** - Cuestionario: Evaluación (TPC- Test de Pensamiento Computacional)
Línea de investigación:	Gestión del conocimiento en el campo educativo		**Campo de investigación:** Gestión del conocimiento.			

6.2. Anexo 2: Solicitud de Permiso para Aplicación del Experimento

UCSS

Los Olivos, 17 de septiembre del 2018

Ing.
Karol Soto Ccoicca
Docente de la Facultad de Ingeniería de la Universidad Católica Sedes Sapientiae

Estimada Docente,

Por medio de la presente le autorizamos aplicar el Pre test y Post test de Pensamiento Computacional a los estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas del semestre académico 2018-II de esta casa de estudios que dirijo.

Así mismo, podrá impartir en 7 sesiones de clases, la enseñanza del Lenguaje de Programación SCRATCH, demostrando lo que indica en su tesis, cuyo título es:

"Influencia del Lenguaje de Programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae, 2018".

Vicerrector Académico
Gian Battista Fausto Bolis

6.3. Anexo 3: Validaciones del Instrumento de Investigación

Lima, 10 de septiembre del 2018.

Mg. Elescano Córdova Wilfredo Clemente
Especialista en el área de investigación.
Presente.

ASUNTO: Validación de instrumento de investigación.

Karol Soto Ccoicca, identificada con DNI N° 40536407; en mi condición de estudiante de Postgrado de la Universidad Católica Sedes Sapientiae, sección **Maestría** en Gestión e Innovación Educativa; **Solicito a Usted su opinión profesional** para validar el instrumento del proyecto de investigación titulado:

"Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae"

Para tal efecto acompaño los siguientes documentos:

1) Matriz de consistencia.
2) Instrumentos de medición: Cuestionario de 28 preguntas.
3) Operacionalización de las variables
4) Ficha de validación.

Agradezco por anticipado la atención de la presente y aprovechar la oportunidad para reiterarle mi consideración y estima personal.

Karol Soto Ccoicca
DNI N° 40536407

Ficha de validación
(Juicio de expertos)

Título de la investigación: "Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae"

Nombre del instrumento: Cuestionario: TEST DE PENSAMIENTO COMPUTACIONAL

Maestrista : Karol Soto Ccoicca

Criterios	Indicadores	Deficiente				Malo					Regular			Bueno				Muy bueno			
		0 - 5	6 - 10	11 - 15	16 - 20	21 - 25	26 - 30	31 - 35	36 - 40	41 - 45	46 - 50	51 - 55	56 - 60	61 - 65	66 - 70	71 - 75	76 - 80	81 - 85	86 - 90	91 - 95	96 - 100
1. Claridad	Está formulado con un lenguaje apropiado y comprensible.																			√	
2. Objetividad	Describe conductas observables en relación con las variables.																		√		
3. Actualidad	Se basa en información teórica, tecnológica o científica vigente.																		√		
4. Organización	Tiene una estructura lógica para recoger la información requerida.																		√		
5. Suficiencia	Comprende los aspectos de las variables en cantidad y calidad suficientes.																			√	
6. Intencionalidad	Mide aspectos precisos de las variables.																				√
7. Consistencia	Se basa en aspectos teórico-científicos de las variables.																			√	
8. Coherencia	Hay relación entre variables, dimensiones, indicadores e ítems.																			√	
9. Metodología	Responde estratégicamente al propósito de estudio.																				√
10. Pertinencia	Ha sido adecuado al problema de investigación.																				√

Opinión de aplicabilidad:

El instrumento de medición está acorde con las variables e indicadores planteadas en la matriz de consistencia, por cuanto los ítems responden a los propósitos de la investigación; por ello el instrumento se encuentra apto para ser aplicado, garantizando objetividad y confiabilidad en su propósito.

Promedio de valoración: 93%

Lugar y Fecha: Lima, 10 de septiembre del 2018.

Apellidos y nombres del experto: Elescano Córdova Wilfredo Clemente

DNI N° 08581718 **Teléfono:** 997523336

Mg. Elescano Córdova Wilfredo Clemente
DNI: 08581718

PERÚ | Ministerio de Educación | Superintendencia Nacional de Educación Superior Universitaria | Dirección de Documentación e Información Universitaria y Registro de Grados y Títulos

REGISTRO NACIONAL DE GRADOS ACADÉMICOS Y TÍTULOS PROFESIONALES

GRADUADO	GRADO O TÍTULO	INSTITUCIÓN
ELESCANO CORDOVA, WILFREDO CLEMENTE DNI 08581718	MAESTRO EN INGENIERIA DE SISTEMAS Fecha de Diploma:31/08/2012	UNIVERSIDAD ALAS PERUANAS S.A.
ELESCANO CORDOVA, WILFREDO CLEMENTE DNI 08581718	LICENCIADO EN COMPUTACION Fecha de Diploma:14/07/1997	UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Lima, 10 de septiembre del 2018.

Mg. Bizarro Tapara Raúl
Especialista en el área de investigación.
Presente.

ASUNTO: Validación de instrumento de investigación.

Karol Soto Ccoicca, identificada con DNI N° 40536407; en mi condición de estudiante de Postgrado de la Universidad Católica Sedes Sapientiae, sección **Maestría** en Gestión e Innovación Educativa**; Solicito a Usted su opinión profesional** para validar el instrumento del proyecto de investigación titulado:

"Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae"

Para tal efecto acompaño los siguientes documentos:

1) Matriz de consistencia.
2) Instrumentos de medición: Cuestionario de 28 preguntas.
3) Operacionalización de las variables
4) Ficha de validación.

Agradezco por anticipado la atención de la presente y aprovechar la oportunidad para reiterarle mi consideración y estima personal.

Karol Soto Ccoicca
DNI N° 40536407

Ficha de validación
(Juicio de expertos)

Título de la investigación: "Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae"

Nombre del instrumento: Cuestionario: TEST DE PENSAMIENTO COMPUTACIONAL

Maestrista : Karol Soto Ccoicca

Criterios	Indicadores	Deficiente				Malo					Regular			Bueno				Muy bueno			
		0 - 5	6 - 10	11 - 15	16 - 20	21 - 25	26 - 30	31 - 35	36 - 40	41 - 45	46 - 50	51 - 55	56 - 60	61 - 65	66 - 70	71 - 75	76 - 80	81 - 85	86 - 90	91 - 95	96-100
1. Claridad	Está formulado con un lenguaje apropiado y comprensible.																			✓	
2. Objetividad	Describe conductas observables en relación con las variables.																		✓		
3. Actualidad	Se basa en información teórica, tecnológica o científica vigente.																		✓		
4. Organización	Tiene una estructura lógica para recoger la información requerida.																			✓	
5. Suficiencia	Comprende los aspectos de las variables en cantidad y calidad suficientes.																		✓		
6. Intencionalidad	Mide aspectos precisos de las variables.																				✓
7. Consistencia	Se basa en aspectos teórico-científicos de las variables.																			✓	
8. Coherencia	Hay relación entre variables, dimensiones, indicadores e ítems.																			✓	
9. Metodología	Responde estratégicamente al propósito de estudio.																				✓
10. Pertinencia	Ha sido adecuado al problema de investigación.																				✓

Opinión de aplicabilidad:

El instrumento de medición está acorde con las variables e indicadores planteadas en la matriz de consistencia, por cuanto los ítems responden a los propósitos de la investigación; por ello el instrumento se encuentra apto para ser aplicado, garantizando objetividad y confiabilidad en su propósito.

Promedio de valoración: 93%

Lugar y Fecha: Lima, 10 de septiembre del 2018.

Apellidos y nombres del experto: Mg. Raúl Bizarro Tapara

DNI N° 10199176 **Teléfono:** 994835302

Mg. Raúl Bizarro Tapara
DNI: 10199176

PERÚ | Ministerio de Educación | Superintendencia Nacional de Educación Superior Universitaria | Dirección de Documentación e Información Universitaria y Registro de Grados y Títulos

REGISTRO NACIONAL DE GRADOS ACADÉMICOS Y TÍTULOS PROFESIONALES

GRADUADO	GRADO O TÍTULO	INSTITUCIÓN
BIZARRO TAPARA, RAUL DNI 10199176	LICENCIADO EN EDUCACION SECUNDARIA EN LA ESPECIALIDAD DE INFORMATICA Fecha de Diploma:05/04/2011	UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE
BIZARRO TAPARA, RAUL DNI 10199176	MAESTRO EN GESTIÓN DE TECNOLOGÍAS DE INFORMACIÓN Fecha de Diploma:11/05/18	UNIVERSIDAD PRIVADA CÉSAR VALLEJO
BIZARRO TAPARA, RAUL DNI 10199176	BACHILLER EN EDUCACION Fecha de Diploma:03/09/2009	UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

Lima, 10 de septiembre del 2018.

Mg. José Manuel Huamán Gutiérrez
Especialista en el área de investigación.
Presente.

ASUNTO: Validación de instrumento de investigación.

Karol Soto Ccoicca, identificada con DNI N° 40536407; en mi condición de estudiante de Postgrado de la Universidad Católica Sedes Sapientiae, sección **Maestría** en Gestión e Innovación Educativa**; Solicito a Usted su opinión profesional** para validar el instrumento del proyecto de investigación titulado:

"Influencia del lenguaje de programación SCRATCH en el desarrollo del pensamiento computacional, en estudiantes de primer ciclo de la carrera de Ingeniería de Sistemas de la Universidad Católica Sedes Sapientiae"

Para tal efecto acompaño los siguientes documentos:

1) Matriz de consistencia.
2) Instrumentos de medición: Cuestionario de 28 preguntas.
3) Operacionalización de las variables.
4) Ficha de validación.

Agradezco por anticipado la atención de la presente y aprovechar la oportunidad para reiterarle mi consideración y estima personal.

Karol Soto Ccoicca
DNI N° 40536407

PERÚ | Ministerio de Educación | Superintendencia Nacional de Educación Superior Universitaria | Dirección de Documentación e Información Universitaria y Registro de Grados y Títulos

REGISTRO NACIONAL DE GRADOS ACADÉMICOS Y TÍTULOS PROFESIONALES

GRADUADO	GRADO O TÍTULO	INSTITUCIÓN
HUAMAN GUTIERREZ, JOSE MANUEL DNI 09905355	LICENCIADO EN ESTADISTICA Fecha de Diploma:08/11/2002	UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
HUAMAN GUTIERREZ, JOSE MANUEL DNI 09905355	BACHILLER EN ESTADISTICA Fecha de Diploma:23/10/2001	UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
HUAMAN GUTIERREZ, JOSE MANUEL DNI 09905355	MAESTRO EN ADMINISTRACION Y DIRECCION DE EMPRESAS Fecha de Diploma:28/05/15	UNIVERSIDAD ALAS PERUANAS S.A.

6.4. Anexo 4: Instrumento de Medición

TEST DE PENSAMIENTO COMPUTACIONAL
Creado por Marcos Román Gonzáles
Bienvenid@ al Test de Pensamiento Computacional

Apellidos y nombres: .....

INSTRUCCIONES
El test está compuesto por 28 preguntas. Todas las preguntas tienen 4 opciones de respuesta (A, B, C ó D) de las cuales sólo una es correcta. A partir de que comience el test dispones de 45 minutos para hacerlo lo mejor que puedas. No es imprescindible que contestes a todas las preguntas. Antes de comenzar el test, vamos a ver 3 ejemplos para que te familiarices con el tipo de preguntas que te irás encontrando, y en la que aparecerán los personajes que ya te presentamos. ¡ÁNIMO Y SUERTE!

'Pac-Man'

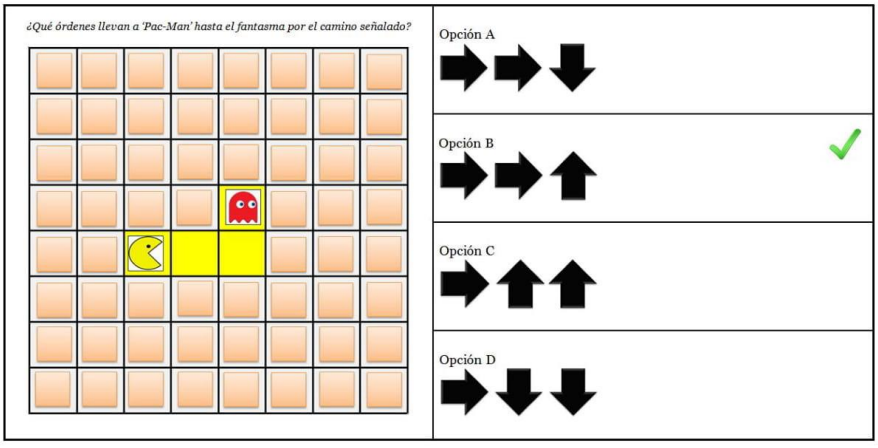
Fantasma

Artista

EJEMPLO I
En este primer ejemplo se te pregunta cuáles son las órdenes que llevan a 'Pac-Man' hasta el fantasma por el camino señalado. Es decir, llevar a 'Pac-Man' EXACTAMENTE a la casilla en la que se encuentra el fantasma (sin pasarse ni quedarse corto), y siguiendo estrictamente el camino señalado en amarillo (sin salirse y sin tocar las paredes, representadas por los cuadrados anaranjados) La opción correcta en este ejemplo es la B. Márcala en el botón de respuesta correspondiente, que está debajo de la pregunta.

Marca la opción correcta (en este ejemplo la opción correcta es la B)

☐ A ☐ B ☐ C ☐ D

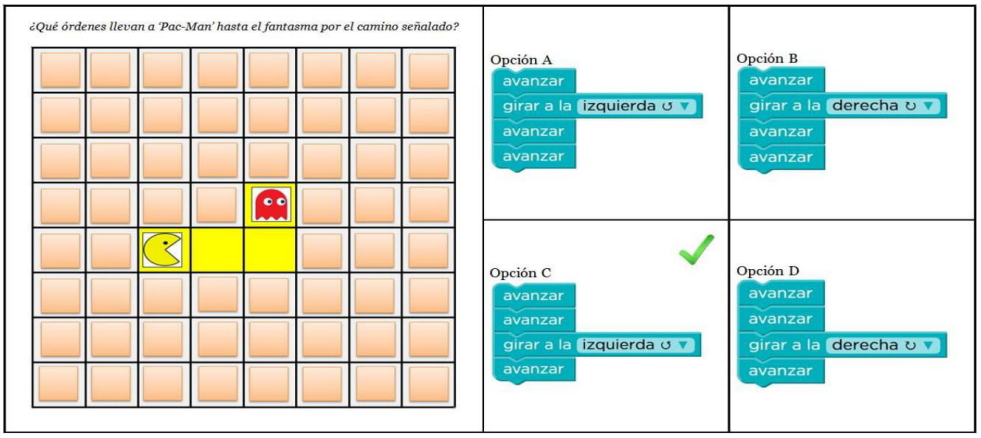


EJEMPLO II

En este segundo ejemplo se te pregunta de nuevo cuáles son las órdenes que llevan a 'Pac-Man' hasta el fantasma por el camino señalado. Pero en este caso las opciones de respuesta, en vez de ser flechas, son bloques que encajan unos con otros. Te recordamos que la pregunta te pide llevar a 'Pac-Man' EXACTAMENTE a la casilla en la que se encuentra el fantasma (sin pasarse ni quedarse corto), y siguiendo estrictamente el camino señalado en amarillo (sin salirse y sin tocar las paredes, representadas por los cuadrados anaranjados) La opción correcta en este ejemplo es la C. Márcala en el botón de respuesta correspondiente, que está debajo de la pregunta.

Marca la opción correcta (en este ejemplo la opción correcta es la C)

☐A ☐B ☐C ☐D

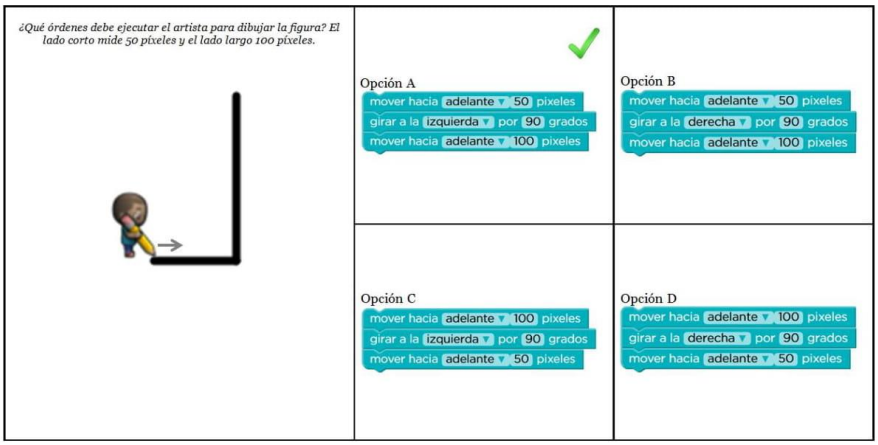


EJEMPLO III

En este tercer ejemplo se te pregunta qué órdenes debe seguir el artista para dibujar la figura que aparece en pantalla. Es decir, cómo debe MOVER el lápiz para que se dibuje la figura. La orden MOVER empuja el lápiz dibujando, mientras que la orden SALTAR hace pegar un salto al artista sin dibujar. La flecha gris indica la dirección del primer movimiento del lápiz. La opción correcta en este ejemplo es la A. Márcala en el botón de respuesta correspondiente, que está debajo de la pregunta.

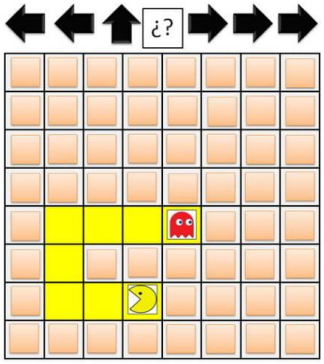
Marca la opción correcta (en este ejemplo la opción correcta es la A)

☐A ☐B ☐C ☐D



Pregunta 1: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué orden falta en la secuencia para llevar a 'Pac-Man' hasta el fantasma por el camino señalado?



Opción A

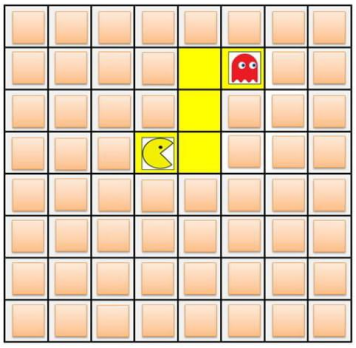
Opción B

Opción C

Opción D

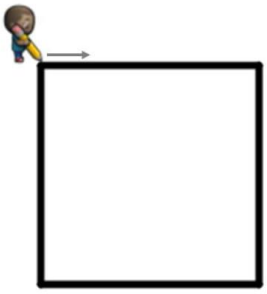
Pregunta 2: Marca la opción correcta ☐A ☐B ☐C ☐D

*Para llevar a 'Pac-Man' hasta el fantasma por el camino señalado, ¿en qué paso de la siguiente secuencia de órdenes hay un **error**?*



Pregunta 3: Marca la opción correcta

¿Qué órdenes debe ejecutar el artista para dibujar el cuadrado? Cada uno de los lados del cuadrado mide 100 píxeles.



Opción A

Opción B

Opción C

Opción D

Pregunta 4: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué bloque falta en la siguiente secuencia de órdenes para que 'Pac-Man' llegue hasta el fantasma por el camino señalado?

Opción A

girar a la izquierda

Opción B

girar a la derecha

Opción C

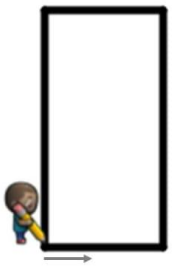
avanzar

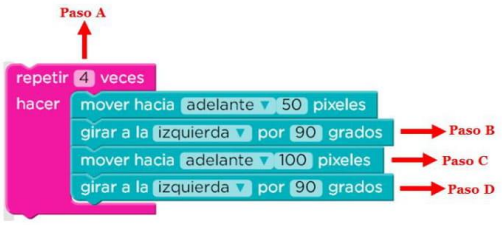
Opción D

No falta ningún bloque

Pregunta 5: Marca la opción correcta

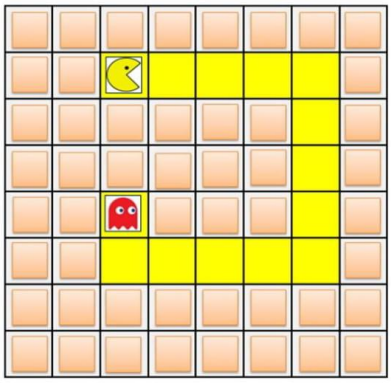
*Para que el artista dibuje **una vez** el siguiente rectángulo (50 píxeles de ancho y 100 píxeles de alto), ¿en qué paso de la siguiente secuencia de órdenes hay un **error**?*





Pregunta 6: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?



Opción A

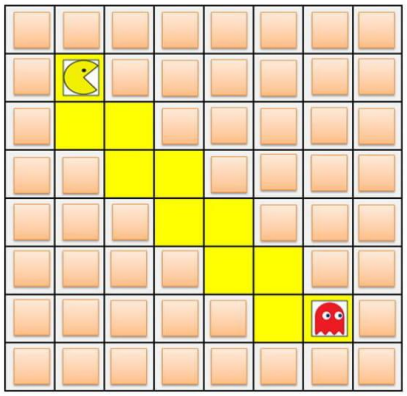
Opción B

Opción C

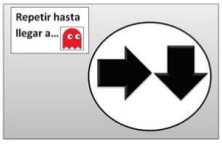
Opción D

Pregunta 7: Marca la opción correcta ☐A ☐B ☐C ☐D

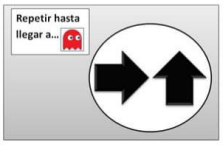
¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?



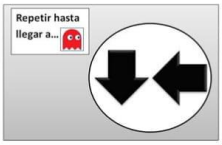
Opción A



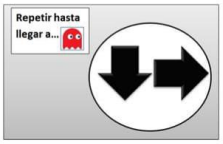
Opción B



Opción C

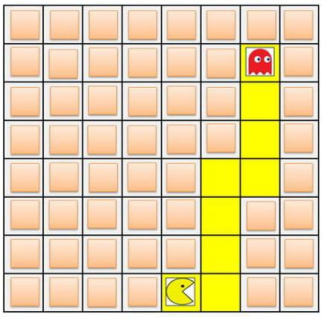


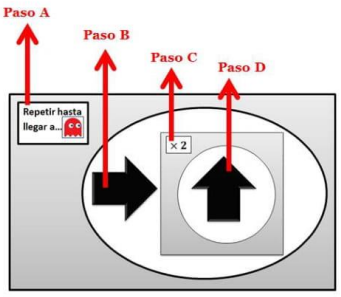
Opción D



Pregunta 8: Marca la opción correcta ☐A ☐B ☐C ☐D

*Para que 'Pac-Man' llegue hasta el fantasma por el camino señalado, ¿en qué paso de la siguiente secuencia de órdenes hay un **error**?*





Pregunta 9: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué secuencia de órdenes debe ejecutar el artista para dibujar la escalera que llegue hasta la flor? Cada peldaño sube 30 píxeles

Opción A

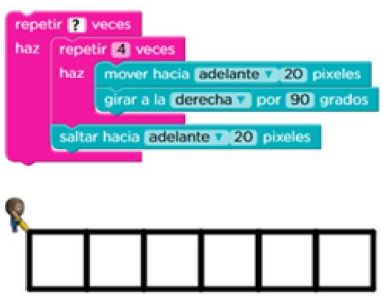
Opción B

Opción C

Opción D

Pregunta 10: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué número falta en la siguiente secuencia de órdenes para que el artista ejecute la figura? El ancho total de la figura es de 120 píxeles.



Opción A
4

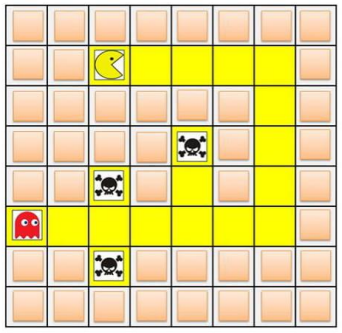
Opción B
6

Opción C
2

Opción D
5

Pregunta 11: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?



Opción A

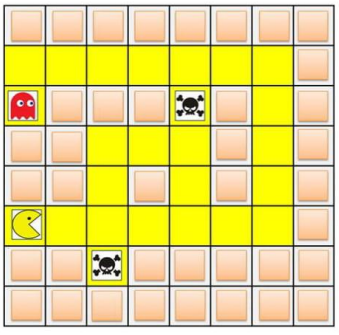
Opción B

Opción C

Opción D

Pregunta 12: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?



Opción A

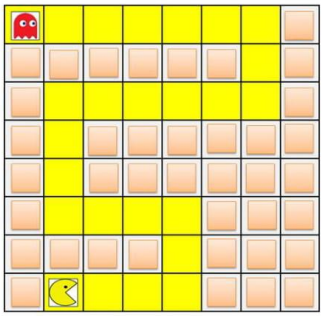
Opción B

Opción C

Opción D

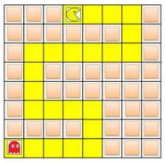
Pregunta 13: Marca la opción correcta ☐A ☐B ☐C ☐D

*Para que 'Pac-Man' llegue hasta el fantasma por el camino señalado, ¿en qué paso de la siguiente secuencia de órdenes hay un **error**?*



Pregunta 14: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué bloque falta en la siguiente secuencia de órdenes para que 'Pac-Man' llegue hasta el fantasma por el camino señalado?



Opción A

Opción B

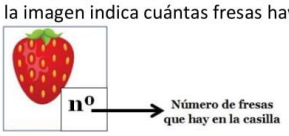
Opción C

Opción D

No falta ningún bloque

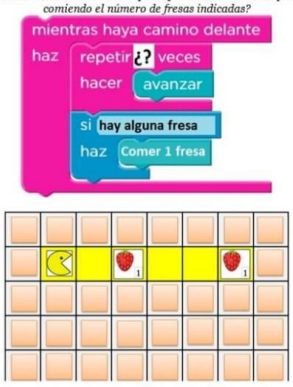
IMPORTANTE: LEE CON ATENCIÓN

En este grupo de preguntas aparece la imagen 'fresa' en algunas casillas. El número que aparece en la parte inferior derecha de la imagen indica cuántas fresas hay en dicha casilla.



Pregunta 15: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué falta en la siguiente secuencia de órdenes para que 'Pac-Man' avance por el camino señalado comiendo el número de fresas indicadas?



Opción A
1 vez

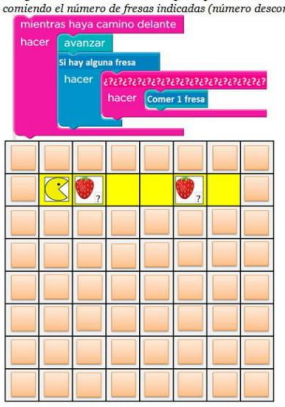
Opción B
2 veces

Opción C
3 veces

Opción D
5 veces

Pregunta 16: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué bloque falta en la siguiente secuencia de órdenes para que 'Pac-Man' avance por el camino señalado comiendo el número de fresas indicadas (número desconocido)?



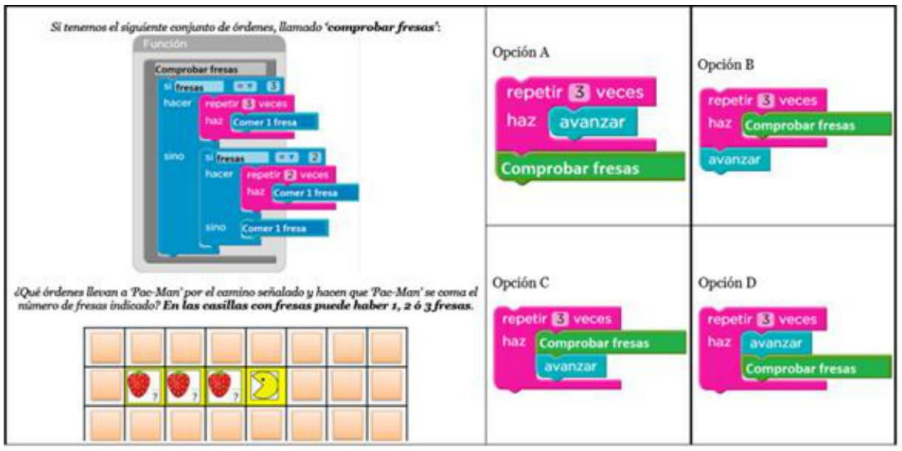
Opción A
Mientras haya camino delante

Opción B
Mientras no haya camino delante

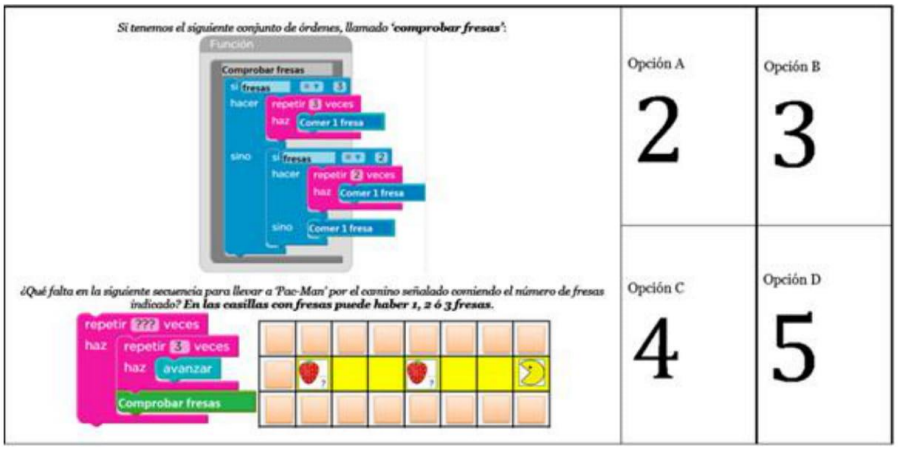
Opción C
Mientras haya alguna fresa

Opción D
Mientras no haya ninguna fresa

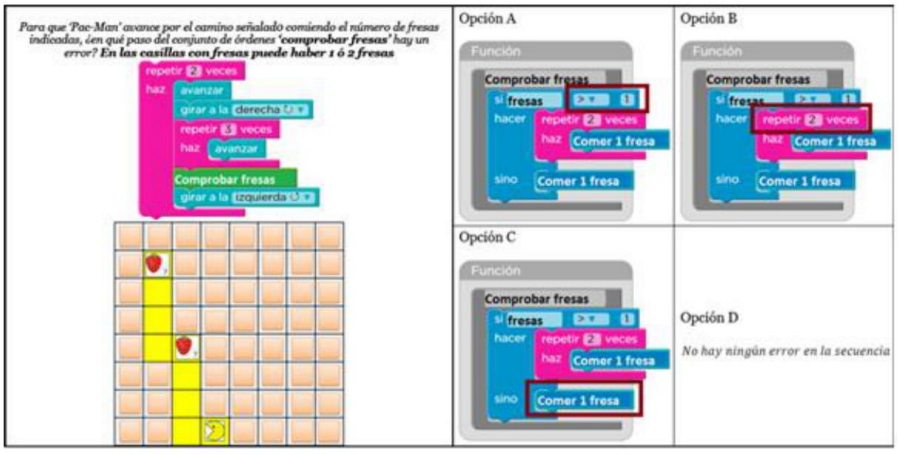
Pregunta 17: Marca la opción correcta ☐A ☐B ☐C ☐D



Pregunta 18: Marca la opción correcta ☐A ☐B ☐C ☐D

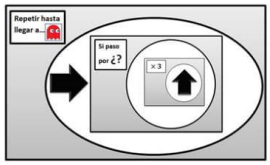


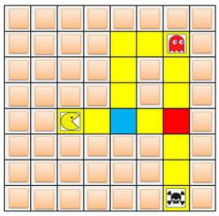
Pregunta 19: Marca la opción correcta ☐A ☐B ☐C ☐D



Pregunta 20: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué falta en la siguiente secuencia de órdenes para llevar a 'Pac-Man' hasta el fantasma por el camino señalado?





Opción A

Opción B

Opción C

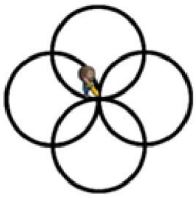
Opción D

Tanto la opción A como la opción C son correctas

Pregunta 21: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos el siguiente conjunto de órdenes, al que llamamos **'my function'**:*

Para que el artista ejecute el siguiente diseño ¿En qué paso de la secuencia de órdenes hay un error?



Opción A

Opción B

Opción C

Opción D

No hay ningún error en la secuencia

Pregunta 22: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos la siguiente orden, que dibuja un cuadrado de longitud-lado variable en función de un **'contador'**:*

Y el **'contador'** cuenta **de** un número inicial **a** otro número final, pasando **por** un incremento especificado

¿Qué secuencia debe ejecutar el artista para dibujar el siguiente diseño? El lado del cuadrado más pequeño es 30 píxeles, y el lado del cuadrado más grande es 150 píxeles.

Opción A

Opción B

Opción C

Opción D

Pregunta 23: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué le falta a la siguiente secuencia para que el artista dibuje el siguiente diseño? El lado más corto mide 20 píxeles y el lado más largo mide 200 píxeles

Opción A
10

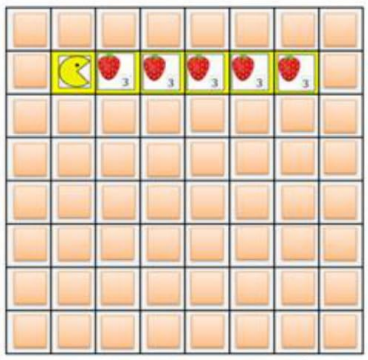
Opción B
20

Opción C
30

Opción D
40

Pregunta 24: Marca la opción correcta ☐A ☐B ☐C ☐D

¿Qué órdenes van llevando a 'Pac-Man' por el camino señalado e indicándole que se coma el número de fresas correspondiente?



Opción A
Mientras haya camino delante {Avanzar 5 veces + [3 × (Comer 1 fresa)]}

Opción B
Mientras haya camino delante {Avanzar 1 vez + [3 × (Comer 1 fresa)]}

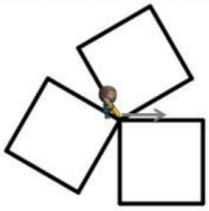
Opción C
Mientras haya camino delante {Avanzar 3 veces + [5 × (Comer 1 fresa)]}

Opción D
Mientras haya camino delante {Avanzar 1 vez} + [3 × (Comer 1 fresa)]

Pregunta 25: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos el siguiente conjunto de órdenes, al que llamamos **'my function'**, y que dibuja un cuadrado de 100 píxeles de lado:*

¿Qué secuencia debe ejecutar el artista para dibujar el siguiente diseño? Cada uno de los lados de cada cuadrado mide 100 píxeles.



Opción A


Opción B

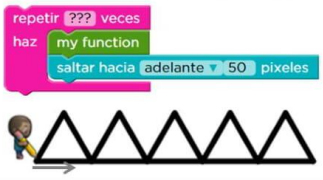

Opción C


Opción D


Pregunta 26: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos el siguiente conjunto de órdenes, al que llamamos **'my function'**, y que dibuja un triángulo de 50 píxeles de lado:*

¿Qué le falta a la siguiente secuencia para que el artista dibuje el siguiente diseño? Cada uno de los lados de cada triángulo mide 50 píxeles.



Opción A: 15

Opción B: 5

Opción C: 4

Opción D: 3

Pregunta 27: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos el siguiente conjunto de órdenes, al que llamamos **'get 5'**:*

¿Qué órdenes van llevando a 'Pac-Man' por el camino señalado e indicándole que se coma el número de fresas correspondiente?

Opción A

Opción B

Opción C

Opción D

Pregunta 28: Marca la opción correcta ☐A ☐B ☐C ☐D

*Si tenemos el siguiente conjunto de órdenes, llamado **'move and get 4'**:*

¿Qué falta en la siguiente secuencia para llevar a 'Pac-Man' por el camino señalado hasta las fresas, comiendo el número de fresas indicado?

Opción A: 3

Opción B: 4

Opción C: 5

Opción D: 6

6.5. Anexo 5: Base de Datos

Pretest

	Conceptos Computacionales																			Prácticas Computacionales									
	Secuencia				Ciclos					Datos	Condicional						Operadores			Abstraer y Modularizar	Incremental e Iterativa			Ensayo y Depuración	Reusar y Remezclar				Puntaje
N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	16
2	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	1	13
3	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	0	1	0	0	1	0	1	0	1	18
4	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	0	1	20
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	0	23
6	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	1	1	0	1	0	0	1	1	19
7	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	1	0	1	1	1	17
8	1	1	1	0	1	1	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	13
9	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	15
10	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	1	1	1	1	1	0	0	0	1	20
11	1	1	1	0	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	17
12	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	8
13	1	1	1	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	1	16
14	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	23
15	1	1	1	0	1	1	0	1	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1	16
16	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	16
17	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	1	1	0	1	1	1	0	1	1	1	1	1	19
18	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	0	1	21
19	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	1	1	0	1	20
20	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0	0	0	0	0	1	0	1	0	1	18
21	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	22
22	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	21
23	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	0	1	0	0	1	0	1	1	1	21
24	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	17
25	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	1	0	20
26	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	23
27	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	16
28	1	1	1	1	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	14
29	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	5
30	1	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	10
31	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	22
32	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	21
33	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	0	1	1	1	0	0	21
34	1	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0	1	1	1	1	1	19
35	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	24
36	1	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	1	14
37	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	1	0	17
38	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	1	1	1	0	1	0	0	1	1	17
39	1	1	1	0	1	1	1	0	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	13
40	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	19
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	27
42	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	1	22
43	1	1	1	0	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	21
44	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	21
45	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	1	1	0	0	0	18
46	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1	1	0	0	1	1	1	1	1	22

DIMENSIÓN 1: CONCEPTOS COMPUTACIONALES

	Conceptos Computacionales																			
	Secuencia				Ciclos					Operadores			Datos	Condicional						Puntaje
N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0	0	13
2	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	8
3	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	14
4	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	16
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
6	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	14
7	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	1	1	1	0	12
8	1	1	1	0	1	1	1	0	1	1	0	0	0	1	0	1	1	1	0	12
9	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0	14
10	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	14
11	1	1	1	0	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	12
12	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	6
13	1	1	1	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	13
14	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	16
15	1	1	1	0	1	1	0	1	1	1	1	0	0	0	1	0	1	1	0	12
16	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	0	13
17	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	1	1	0	11
18	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
19	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	16
20	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0	15
21	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	17
22	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	1	0	13
23	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	16
24	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	12
25	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	1	1	0	14
26	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	17
27	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	0	13
28	1	1	1	1	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	12
29	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	4
30	1	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	7
31	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	17
32	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	16
33	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	16
34	1	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	1	1	13
35	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	17
36	1	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	9
37	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	16
38	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	11
39	1	1	1	0	1	1	1	0	1	0	1	1	0	1	0	1	1	1	0	13
40	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	15
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
42	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	16
43	1	1	1	0	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	14
44	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	17
45	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	16
46	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	15

DIMENSIÓN 2: PRÁCTICAS COMPUTACIONALES

	Prácticas Computacionales									
	Abstraer y Modularizar	Incremental e Iterativa			Ensayo y Depuración	Reusar y Remezclar				Puntaje
N°	20	21	22	23	24	25	26	27	28	
1	1	1	0	1	0	0	0	0	0	3
2	0	1	0	0	1	0	1	1	1	5
3	0	1	0	0	1	0	1	0	1	4
4	0	0	0	1	1	0	1	0	1	4
5	1	1	0	0	1	1	0	0	0	4
6	0	1	1	0	1	0	0	1	1	5
7	0	1	0	0	1	0	1	1	1	5
8	1	0	0	0	0	0	0	0	0	1
9	0	0	0	0	1	0	0	0	0	1
10	1	1	1	1	1	0	0	0	1	6
11	1	1	0	0	1	1	1	0	0	5
12	0	0	0	0	0	1	1	0	0	2
13	0	0	0	0	0	0	1	1	1	3
14	0	1	1	0	1	1	1	1	1	7
15	0	1	0	1	0	0	1	0	1	4
16	0	0	0	0	0	0	1	1	1	3
17	1	1	1	0	1	1	1	1	1	8
18	0	0	0	0	1	0	1	0	1	3
19	0	1	0	0	0	1	1	0	1	4
20	0	0	0	0	1	0	1	0	1	3
21	0	0	0	0	1	1	1	1	1	5
22	1	1	1	0	1	1	1	1	1	8
23	0	1	0	0	1	0	1	1	1	5
24	1	1	0	0	0	0	1	1	1	5
25	1	1	1	0	1	0	1	1	0	6
26	0	1	0	0	1	1	1	1	1	6
27	0	0	1	1	1	0	0	0	0	3
28	0	0	0	0	1	0	1	0	0	2
29	0	0	0	0	0	0	1	0	0	1
30	0	1	0	1	0	0	1	0	0	3
31	0	0	1	0	1	0	1	1	1	5
32	0	0	0	0	1	1	1	1	1	5
33	0	1	1	0	1	1	1	0	0	5
34	0	0	1	0	1	1	1	1	1	6
35	0	0	1	1	1	1	1	1	1	7
36	0	0	0	1	1	0	1	1	1	5
37	0	0	0	0	0	0	0	1	0	1
38	1	1	1	0	1	0	0	1	1	6
39	0	0	0	0	0	0	0	0	0	0
40	0	1	0	0	1	0	0	1	1	4
41	1	1	1	0	1	1	1	1	1	8
42	0	1	0	0	1	1	1	1	1	6
43	0	1	1	0	1	1	1	1	1	7
44	0	1	0	0	0	1	1	1	0	4
45	0	0	0	0	1	1	0	0	0	2
46	1	1	0	0	1	1	1	1	1	7

Posttest

	Conceptos Computacionales																			Prácticas Computacionales								
	Secuencia				Ciclos					Datos	Condicional						Operadores			Abstraer y Modularizar	Incremental e Iterativa			Ensayo y Depuración	Reusar y Remezclar			Puntaje
N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	1	0	1	0	1	1	0	1	0	1	19
2	1	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	0	1	0	0	1	0	1	1	0	0	1	0	15
3	1	1	1	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	0	1	1	0	1	0	1	1	1	20
4	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	1	0	1	22
5	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	25
6	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	22
7	1	1	1	1	1	1	1	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	1	0	1	1	1	17
8	1	1	1	0	1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1	15
9	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1	22
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	27
11	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	0	1	1	1	0	1	1	1	1	1	22
12	1	1	0	0	1	1	0	0	1	0	1	1	0	1	1	1	0	1	0	0	1	1	0	0	1	1	0	1	16
13	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	0	1	1	1	0	1	22
14	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	25
15	1	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	0	1	0	0	1	1	0	0	1	1	18
16	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	0	0	1	1	1	0	1	20
17	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	0	21
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	26
19	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	23
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	25
21	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	23
22	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	24
23	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	23
24	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	22
25	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	24
26	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	23
27	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0	1	1	0	0	1	0	1	1	1	20
28	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0	1	21
29	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	8
30	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	16
31	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	24
32	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	27
33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	26
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	26
35	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	26
36	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	1	1	22
37	1	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	23
38	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	27
39	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	18
40	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	25
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	28
42	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	24
43	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	24
44	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	25
45	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	19
46	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1	0	1	1	1	1	1	22

DIMENSIÓN 1: CONCEPTOS COMPUTACIONALES

	Conceptos Computacionales																		
	Secuencia				Ciclos					Operadores			Datos	Condicional						Puntaje
N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	1	14
2	1	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	0	1	0	11
3	1	1	1	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	14
4	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	16
5	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	18
6	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	14
7	1	1	1	1	1	1	1	0	1	0	1	1	0	0	0	0	1	1	1	13
8	1	1	1	0	1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	11
9	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	16
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
11	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	0	14
12	1	1	0	0	1	1	0	0	1	0	1	1	0	1	1	1	0	1	0	11
13	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	16
14	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
15	1	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	0	13
16	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	15
17	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	17
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
19	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	18
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	17
21	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	16
22	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	17
23	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	17
24	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	1	1	14
25	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	16
26	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	0	1	1	1	15
27	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0	14
28	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	16
29	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	4
30	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	13
31	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	17
32	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	18
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	17
35	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	18
36	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	16
37	1	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	16
38	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
39	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	0	13
40	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	17
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
42	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	17
43	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	17
44	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
45	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	0	15
46	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	15

DIMENSIÓN 2: PRÁCTICAS COMPUTACIONALES

	Prácticas Computacionales									
	Abstraer y Modularizar	Incremental e Iterativa			Ensayo y Depuración	Reusar y Remezclar				Puntaje
N°	20	21	22	23	24	25	26	27	28	
1	0	1	0	1	1	0	1	0	1	5
2	0	1	0	1	1	0	0	1	0	4
3	0	1	1	0	1	0	1	1	1	6
4	0	1	1	0	1	1	1	0	1	6
5	0	1	1	0	1	1	1	1	1	7
6	1	1	1	0	1	1	1	1	1	8
7	0	0	0	0	1	0	1	1	1	4
8	0	1	0	1	0	0	1	0	1	4
9	0	0	1	0	1	1	1	1	1	6
10	1	1	1	0	1	1	1	1	1	8
11	1	1	1	0	1	1	1	1	1	8
12	0	1	1	0	0	1	1	0	1	5
13	0	1	1	0	1	1	1	0	1	6
14	0	1	1	0	1	1	1	1	1	7
15	1	0	0	1	1	0	0	1	1	5
16	1	0	0	0	1	1	1	0	1	5
17	1	0	0	0	1	1	1	0	0	4
18	1	1	1	0	1	1	0	1	1	7
19	0	0	0	0	1	1	1	1	1	5
20	1	1	1	0	1	1	1	1	1	8
21	0	1	1	0	1	1	1	1	1	7
22	0	1	1	0	1	1	1	1	1	7
23	0	1	1	1	1	1	0	0	1	6
24	1	1	1	0	1	1	1	1	1	8
25	1	1	1	0	1	1	1	1	1	8
26	1	1	1	0	1	1	1	1	1	8
27	1	1	0	0	1	0	1	1	1	6
28	0	0	1	0	1	1	1	0	1	5
29	0	1	0	1	1	0	0	0	1	4
30	0	1	0	0	1	0	1	0	0	3
31	0	1	1	1	1	1	1	1	0	7
32	1	1	1	0	1	1	1	1	1	8
33	1	1	1	0	1	1	1	1	1	8
34	1	1	1	1	1	1	1	1	1	9
35	0	1	1	1	1	1	1	1	1	8
36	1	1	0	0	0	1	1	1	1	6
37	1	0	1	0	1	1	1	1	1	7
38	1	1	1	0	1	1	1	1	1	8
39	0	1	0	1	1	1	1	0	0	5
40	1	1	1	0	1	1	1	1	1	8
41	1	1	1	1	1	1	1	1	1	9
42	1	1	1	0	1	1	1	1	0	7
43	0	1	1	0	1	1	1	1	1	7
44	0	1	1	1	1	1	1	0	1	7
45	0	0	1	0	0	1	1	0	1	4
46	0	1	1	0	1	1	1	1	1	7

6.6. Anexo 6: Sesiones

Sesión 1

FUNDAMENTOS DE ALGORITMOS Y LENGUAJE DE PROGRAMACIÓN

Profesora: Karol Soto Ccoicca

INDICE

Capítulo I: Algoritmos

1.1 Definición

Existen muchas definiciones referentes a algoritmos, entre las cuales tenemos:

- Un algoritmo es un conjunto de instrucciones las cuales le dicen a la computadora como ejecutar una tarea específica.
- Un algoritmo es un conjunto ordenado y finito de instrucciones que conducen a la solución de un problema.
- "Una lista de instrucciones donde se especifica una sucesión de operaciones necesarias para resolver cualquier problema de un tipo dado".

1.2 Características de los algoritmos

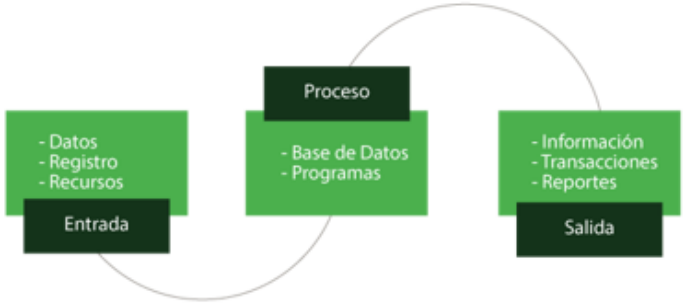
Asimismo, los algoritmos presentan una serie de características comunes. Son:

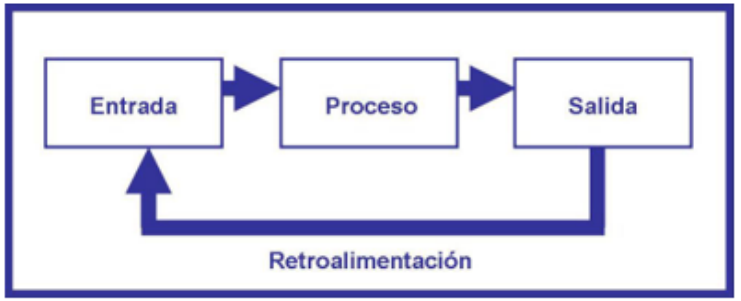
- Precisos. Objetivos, sin ambigüedad.
- Ordenados. Presentan una secuencia clara y precisa para poder llegar a la solución.
- Finitos. Contienen un número determinado de pasos.
- Concretos. Ofrecen una solución determinada para la situación o problema planteados.
- Definidos. El mismo algoritmo debe dar el mismo resultado al recibir la misma entrada.

1.3 Partes de un algoritmo

Las tres partes de un algoritmo son:

- Input (entrada). Datos que damos al algoritmo con la que va a trabajar para ofrecer la solución esperada.
- Proceso. Conjunto de pasos para que, a partir de los datos de entrada, llegue a la solución de la situación.
- Output (salida). Resultados que se generan a partir de la transformación de los valores de entrada durante el proceso.





De este modo, un algoritmo informático parte de un estado inicial y de unos valores de entrada, sigue una serie de pasos sucesivos y llega a un estado final en el que ha obtenido una solución.

Ejemplos de algoritmos

Aunque es un término habitual en áreas como las matemáticas, la informática, la lógica y demás disciplinas relacionadas, lo cierto es que en la vida cotidiana también usamos algoritmos para solucionar cuestiones, por ejemplo:

- Recetas de cocina

  Explican el paso a paso para crear una comida con una cantidad finita de ingredientes. El estado inicial serían los ingredientes sin procesar y el estado final la comida preparada.
- Manuales

Sirven de guía para ejecutar procesos, desde cómo armar una biblioteca hasta cómo activar un teléfono móvil. En estos casos, el estado final es el producto armado, instalado, encendido, en funcionamiento, etc.

- Operaciones matemáticas

  En matemáticas, algunos ejemplos de algoritmos son la multiplicación, en donde seguimos una secuencia de operaciones para obtener un producto; o la división, que nos permite determinar el cociente de dos números. El algoritmo de Euclides, con el cual sacamos el máximo común divisor de dos enteros positivos es otro ejemplo de algoritmo.

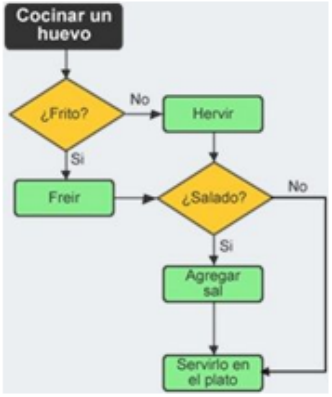
1.4 Representación de los algoritmos:

Los algoritmos se representan utilizando las siguientes herramientas:

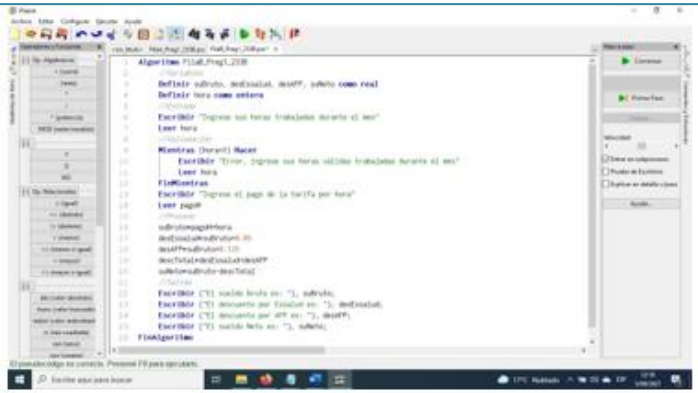
- DFD o diagrama de flujo de datos es una de las técnicas de representación gráfica de algoritmos más antiguas.

  Ventajas: Permite altos niveles de estructuración y modularización y es fácil de usar.

  Desventajas: Son difíciles de actualizar y se complican cuando el algoritmo es grande.



- Pselnt o pseudocódigo, nos permite una aproximación del algoritmo utilizando un lenguaje natural y por tanto una redacción sencilla del mismo.



- Scratch, nos permite representación los algoritmos mediante bloques como rompecabezas.

1.5 Traza de los algoritmos:

La traza de un algoritmo se puede definir como la ejecución manual de forma secuencial de las sentencias que lo componen. La traza de un algoritmo (o programa) indica la secuencia de acciones (instrucciones) de su ejecución, así como, el valor de las variables del algoritmo (o programa) después de cada acción (instrucción).

La función principal que posee realizar la traza de un algoritmo es la de comprobar que éste funciona correctamente o para realizar la etapa de depuración en la que se intenta corregir errores, simplificar el algoritmo al máximo e incrementar su eficacia y velocidad.

Ejemplos:

PSEUDOCODIGO

```
Algoritmo Suma
Inicio
Variable Entera a,b,c
Escribir "Indique el Primer Sumando"
Leer a
Escribir "Indique el Segundo Sumando"
Leer b
c = a + b
Escribir "El Resultado es "; c
Fin
```

TRAZA (corrida en frio)

```
Inicio
a = 5
b = 4
c = 4 + 5 = 9
c = 9
Fin
```

Secuencia:	Acción (instrucción):	Valor de: a	Valor de: n
	© carlospes.com		
1	`a ← 0`	**0**	**?**
	Inicio de la iteración 1.		
2	`escribir( "Introduzca un número entero:" )`	**0**	**?**
3	`leer( n )`	**0**	**15**
4	(Comprobar si `n = 0`)	**0**	**15**
	La condición de la alternativa simple es **falsa**.		
5	`escribir( "El opuesto es: ", -n )`	**0**	**15**
6	`a ← a + n`	**15**	**15**
	Fin de la iteración 1.		
7	(Comprobar si `n >= -10 y n <= 10`)	**15**	**15**
	La condición del bucle es **falsa**. El bucle finaliza después de 1 iteración.		
8	`escribir( "Suma: ", a )`	**15**	**15**

1.6 Tipos de Algoritmos

5 TIPOS DE ALGORITMOS

¿QUÉ ES UN ALGORITMO?

Procedimiento paso a paso para resolver un problema o conseguir un fin.

1. ALGORITMOS DE BÚSQUEDA

En una estructura de datos, localizan uno o varios elementos que presenten ciertas características. Como la búsqueda lineal y la binaria.

2. ALGORITMOS DE ORDENAMIENTO

Reorganizan un listado de elementos a acuerdo a una relación de orden.

Destacan el ordenamiento por inserción, por mezcla, por selección, de burbuja y el ordenamiento rápido.

3. PROGRAMACIÓN DINÁMICA

Método que reduce el tiempo de ejecución de un algoritmo, al dividir problemas en subproblemas y almacenar su solución, para que no haya que volver a calcularlos.

La serie de Fibonacci, las Torres de Hanói y el algoritmo de Dijkstra son algunos ejemplos.

4. ALGORITMOS VORACES

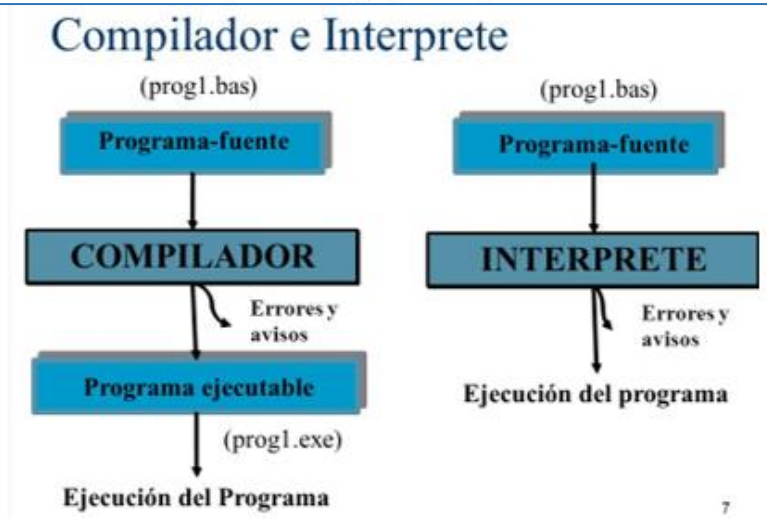
Adoptan la decisión local más óptima para llegar a la mejor solución global. Por ejemplo, el algoritmo de Prim, el de Dijkstra y el problema de la mochila (KP).

5. ALGORITMOS PROBABILÍSTICOS

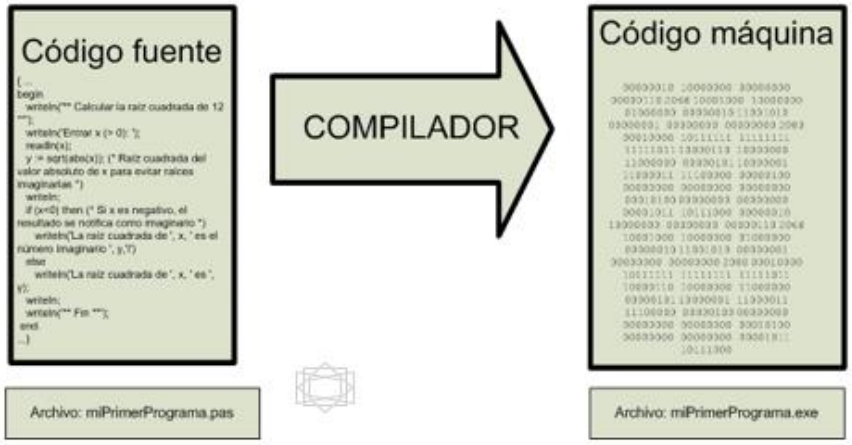
Utilizan un cierto grado de azar para proporcionar un resultado. De media proporcionan una buena solución al problema. Existen los algoritmos numéricos, de Montecarlo y de Las Vegas.

profile.es/blog

profile

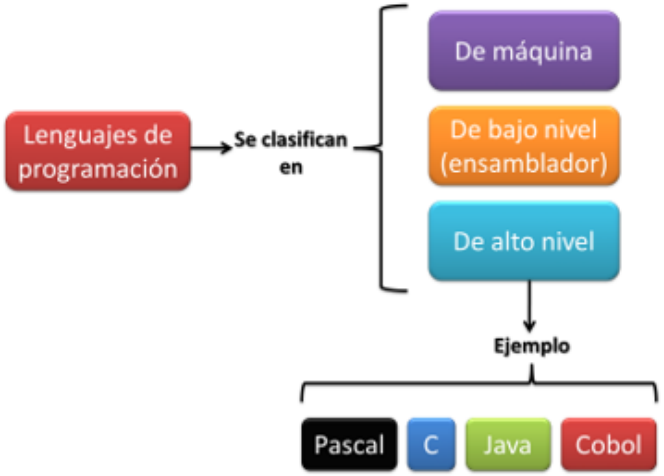


En cuanto a los compiladores, traducen los símbolos de un lenguaje de programación a su equivalencia escrito en lenguaje máquina (proceso conocido como compilar). Por último, se obtiene un programa ejecutable.



2.3 Tipos de Lenguajes de Programación

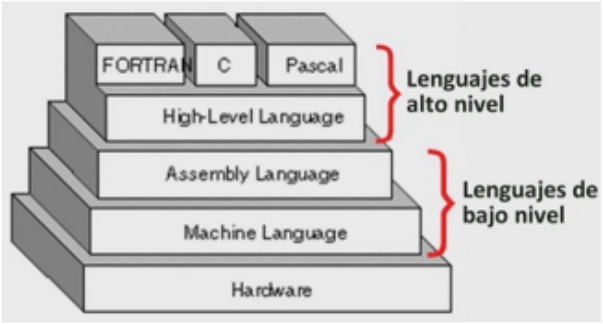
Los lenguajes de programación se dividen en tres tipos claramente diferenciados:



2.4 Generaciones de los lenguajes de programación

- Lenguaje máquina: También llamado lenguaje de primera generación, es en realidad el único lenguaje que hablan los ordenadores. Todos los demás se traducen de una forma u otra a este. Se basa en un sistema de numeración binaria de ceros y unos. Ejemplo: 1111000 representa el número 120 en código binario.
- Lenguajes de bajo nivel: También llamados lenguajes de segunda generación, son los primeros lenguajes de programación que aparecieron, como el COBOL o el FORTRAN. A pesar de ser un poco más fáciles de entender, seguían siendo muy complicados de leer y escribir, lo que lastraba la posibilidad de desarrollar software de forma ágil y eficaz. Ejemplo: ejemplo de "Hola Mundo" en COBOL.
- Lenguajes de alto nivel: estos tipos de lenguajes son los que se utilizan en la actualidad, son muy parecidos al inglés y cualquier persona los puede entender. Se diferencian tres generaciones distintas de lenguajes:

- Tercera generación: son similares a los de segunda generación, pero más fáciles de interpretar, mejor optimizados y con más posibilidades. Lenguajes como el Pascal o el C, C++, Delphi o el PHP se siguen utilizando a día de hoy.
- Cuarta generación: son los más cercanos al lenguaje humano y son los que más se utilizan actualmente. Java, Visual Basic .NET o SQL son algunos de los más populares y suelen estar presentes en muchas de las aplicaciones actuales.
- Quinta generación: son lenguajes naturales y muy avanzados que están pensados para ser usados en inteligencia artificial (IA). Aún se está trabajando en ellos.



2.5 10 tipos de lenguajes de programación más populares y solicitados por el mercado

En su última actualización 2019, el índice Tiobe, un indicador elaborado por una empresa de software holandesa que se especializa en la evaluación y seguimiento de la calidad de los programas informáticos, ha contemplado un nuevo ranking referido a los lenguajes de programación más usados en la actualidad.

La empresa holandesa revisa en tiempo real más de 300 millones de códigos de diversos programas informáticos por día actualizando mes a mes su índice que se elabora a partir de diversas variables, entre las cuales podemos destacar:

- El número de ingenieros cualificados en determinado lenguaje.
- Las búsquedas que hacen los usuarios a través de los buscadores solicitando información de los distintos lenguajes de programación
- La demanda de cursos o los lenguajes que están siendo más utilizados.

Es importante que comprendas que el índice Thiobe no indica cuál es el mejor o en qué lenguaje de programación se escribió la mayor cantidad de líneas de código. Más bien sirve para que un programador pueda determinar si sus conocimientos en un determinado lenguaje han quedado obsoletos, o si por el contrario sus conocimientos están vigentes.

a) Java

Reconocido por su legibilidad y simplicidad, Java es uno de los lenguajes de programación más adoptados: más 9 millones de desarrolladores lo usan y está presente en 7 mil millones de dispositivos en todo el mundo. Desde 2001 se mantiene en las primeras posiciones.

Su enorme popularidad se debe a su poder de permanencia, cuestión que asegura el funcionamiento a largo plazo de las aplicaciones que lo utilizan.

b) C

Creado entre 1969 y 1972 en los Laboratorios Bell, es uno de los más utilizados en el mundo. Si bien es ejecutado en la mayoría de los sistemas operativos, es de propósito general, con lo cual es muy flexible.

Es muy popular para el desarrollo de aplicaciones de escritorio, como el conocido editor gráfico GIMP.

c) Python

Un lenguaje de programación multiplataforma y multiparadigma, que también es de propósito general y el año pasado ha superado al que conocerás en el puesto número.

Su simpleza, legibilidad y similitud con el idioma inglés lo convierten en un gran lenguaje ideal para principiantes

d) C++

Conocido por el nombre "C Plus Plus", se orienta a objetos surge como una continuación y ampliación del C. Hay una gran cantidad de programas escritos en C++, como por ejemplo los paquetes de Adobe.

e) **C#**

También llamado "C Sharp", está orientado a objetos y fue desarrollado en el año 2000 por Microsoft para ser empleado en una amplia gama de aplicaciones empresariales ejecutadas en el framework .NET. C Sharp es una evolución del C y C++ que se destaca por su sencillez y modernidad.

f) **Visual Basic. NET**

Ha ascendido del número 9 en junio de 2016 al sexto lugar en 2017 siendo utilizado por una gran cantidad de personas que no cuentan con conocimientos profundos como desarrolladores, quienes encuentran en Visual Basic, además de una sintaxis sencilla, la posibilidad de automatizar sus propios procesos y crear sus propias aplicaciones web.

g) **JavaScript**

No debemos confundirlo con Java. Son lenguajes distintos. JavaScript es un lenguaje de programación que puede ser utilizado para crear programas que luego son acoplados a una página web o dentro de programas más grandes. Sirve para crear efectos y realizar acciones interactivas.

Podemos ver funcionando este lenguaje en servicios de chat, calculadoras o buscadores de información.

h) **PHP**

Creado en 1994 por el programador canadiense Rasmus Lerdorf, con la intención de contar con un conjunto de herramientas para el mantenimiento de las páginas web y no como lenguaje.

Es de fácil acceso para nuevos programadores y a su vez ofrece grandes herramientas a los más experimentados.

i) **SWIFT**

Se trata de un lenguaje multiparadigma creado por Apple y focalizado en el desarrollo de aplicaciones para iOS y macOS.? A partir de su presentación en el año 2014, se ha convertido en código abierto y el índice Thiobe, a diferencia de otros años, lo ha ubicado en este puesto por ser uno de los lenguajes de programación más usados actualmente.

j) **SQL**

Este lenguaje de programación ha sido diseñado para administrar, proteger y recuperar los datos de sistemas de gestión de información, lo cual ha sido utilizado fuertemente en los últimos años a partir del desarrollo de la ciberseguridad.

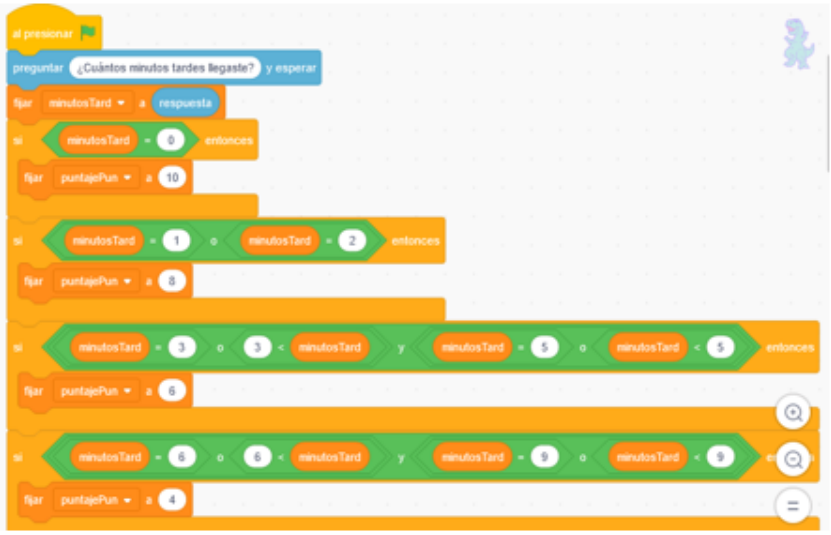
OPERADORES LÓGICOS AND, OR Y NOT

OPERADOR LÓGICO AND

El operador AND lógico condicional **Y**, también denominado operador AND lógico "de cortocircuito", calcula el operador AND lógico de sus operandos. El resultado de **X Y Y** es true si **X Y Y** se evalúan como true. De lo contrario, el resultado es false. Si **X** se evalúa como false, **Y** no se evalúa.

Operador AND o Conjunción: (Y) OBLIGATORIO

X	Y	X Y Y
V	V	V
V	F	F
F	V	F
F	F	F




```
14      Repetir
15          Escribir("Ingrese el nombre")
16          Leer nombre;
17
18          Escribir("Ingrese el tipo de habitación")
19          Leer tipo;
20          tipo←mayusculas(tipo)
21          //Validación
22
23          Mientras (tipo≠"A" y tipo≠"B" y tipo≠"C" y tipo≠"D" y tipo≠"E") Hacer
24              Escribir("Vuelva a ingresar un tipo de habitación válido que puede ser: A, B, C, D o E")
25              Leer tipo;
26              tipo←mayusculas(tipo)
27          FinMientras
28
29
30          Escribir("Ingrese la cantidad de días que se hospedará")
31          Leer cantDia;
32
33          segun(tipo) hacer
34              "A": costo←90;
35                   contA←contA+1
36              "B": costo←80;
37                   contB←contB+1
38              "C": costo←70;
```

OPERADOR LÓGICO OR

El operador OR lógico condicional **O**, también denominado operador OR lógico "de cortocircuito", calcula el operador OR lógico de sus operandos. El resultado de **X O Y** es `true` si **X O Y** se evalúan como `true`. De lo contrario, el resultado es `false`. Si **X** se evalúa como `true`, **Y** no se evalúa.

Operador OR o Disyunción: (O) OPCIONAL

X	Y	X O Y
V	V	V
V	F	V
F	V	V
F	F	F

```
6       //Entrada
7       Escribir("Ingrese la nota de Matemática")
8       Leer notaMate
9       //Validación de la nota de Matemática
10      Mientras (notaMate<0 o notaMate>20) Hacer
11          Escribir("Error, vuelve a ingresar una nota válida de Matemática")
12          Leer notaMate
13      FinMientras
14
15
16      Escribir("Ingrese la nota de Física")
17      Leer notaFisi
18      //Validación de la nota de Física
19      Mientras (notaFisi<0 o notaFisi>20) Hacer
20          Escribir("Error, vuelve a ingresar una nota válida de Física")
21          Leer notaFisi
22      FinMientras
23
24
25      Escribir("Ingrese la nota de Historia del Perú")
26      Leer notaHis
27      //Validación de la nota de Historia del Perú
28      Mientras (notaHis<0 o notaHis>20) Hacer
29          Escribir("Error, vuelve a ingresar una nota válida de Historia del Perú")
```

OPERADOR LÓGICO NOT

El operador **NOT** devuelve el valor opuesto al valor booleano. Si la expresión es True el valor devuelto es False, de lo contrario si la expresión es False el valor devuelto es True.

X	NOT X
V	F
V	F
F	V
F	V

Compuertas Lógicas: NOT

- ***Realiza la Complementaicón***
- ***Este operador "invierte" el valor lógico de la entrada.***

A	F
0	1
1	0

Sesión 2:

FUNDAMENTOS DE PROGRAMACIÓN

MANUAL DE SCRATCH

PARTE 1

INDICE

Lenguaje de Programación Scratch

Definición

Scratch es un lenguaje de programación visual desarrollado por el Grupo Lifelong Kindergarten del MIT Media Lab. Su principal característica consiste en que permite el desarrollo de habilidades mentales mediante el aprendizaje de la programación sin tener conocimientos profundos sobre el código. Sus características ligadas al fácil entendimiento del pensamiento computacional han hecho que sea muy difundido en la educación de niños, adolescentes y adultos.

Scratch es un lenguaje de programación visual que nos permite agrupar bloques para ir creando las ordenes de nuestros programas. Y así dar forma a nuestras historietas, animaciones, videojuegos, etc.

Las ventajas de la programación por bloques:

- La programación por bloques nos permite generar programas sin necesidad de escribir código por lo que es más difícil equivocarse y se aprende más rápido.
- Además, los bloques están clasificados por categorías y colores que nos hacen mucho más intuitivo el código para poder entenderlo.
- Podemos ver justo al lado el resultado de nuestro código con lo cual a medida que avanzamos ya notamos los resultados.
- Adquirimos el pensamiento computacional de forma mucho más rápida que con otro tipo de lenguajes.
- Nos facilita mucho la tarea para luego aprender otros lenguajes más complejos.

Habilidades que desarrolla el lenguaje de programación Scratch

El lenguaje de programación Scratch desarrolla las siguientes habilidades:

- **Pensamiento Algorítmico:**

  Es diseñar algoritmos es decir determinar los pasos apropiados a seguir y organizarlos en una serie de instrucciones (un plan) para resolver un problema o completar una tarea correctamente.

- **Pensamiento Abstracto:**

  Abstracción implica filtrar, o ignorar, detalles sin importancia, lo que esencialmente hace que un problema sea más fácil de entender y resolver. Esto permite a los estudiantes desarrollar sus modelos, ecuaciones, una imagen y/o simulaciones para representar solamente lo importante.

  Un ejemplo es que los estudiantes hagan un dibujo simple de un postre enfocándose en las características importantes / comunes (como clasificaciones) y abstrayendo el resto (textura, fruta, chispas). El proceso de abstracción les ayudará a crear una idea general de qué es un problema y cómo resolverlo eliminando todos los detalles y patrones irrelevantes (la abstracción también se usa en matemáticas y al crear modelos - el ciclo del agua, el ciclo del nitrógeno, el ciclo de las rocas, etc.).

- **Descomposición de problemas:**

  Enfrentar problemas grandes y complejos a menudo desalentará y desacoplará a los estudiantes que no están completamente equipados para comenzar el proceso de construcción.

  La descomposición (como la factorización) desarrolla la habilidad de dividir problemas complejos en partes más pequeñas y manejables, lo que hace que incluso la tarea o problema más complicado sea más fácil de entender y resolver.

  Los estudiantes deberán analizar problemas/escenarios más complejos que sean desconocidos y lo suficientemente atractivos como para descomponerlos, como, por ejemplo: investigar la escena de un crimen, hacer frente al problema de desastres naturales o plantar un huerto escolar.

- **Reconocimiento de patrones:**

  Reconocimiento de formas es una habilidad que implica mapear similitudes y diferencias o patrones entre problemas pequeños (descompuestos) y es esencial para ayudar a resolver problemas complejos. Los estudiantes que son capaces de reconocer patrones pueden hacer predicciones, trabajar más eficientemente y establecer una base sólida para diseñar algoritmos.

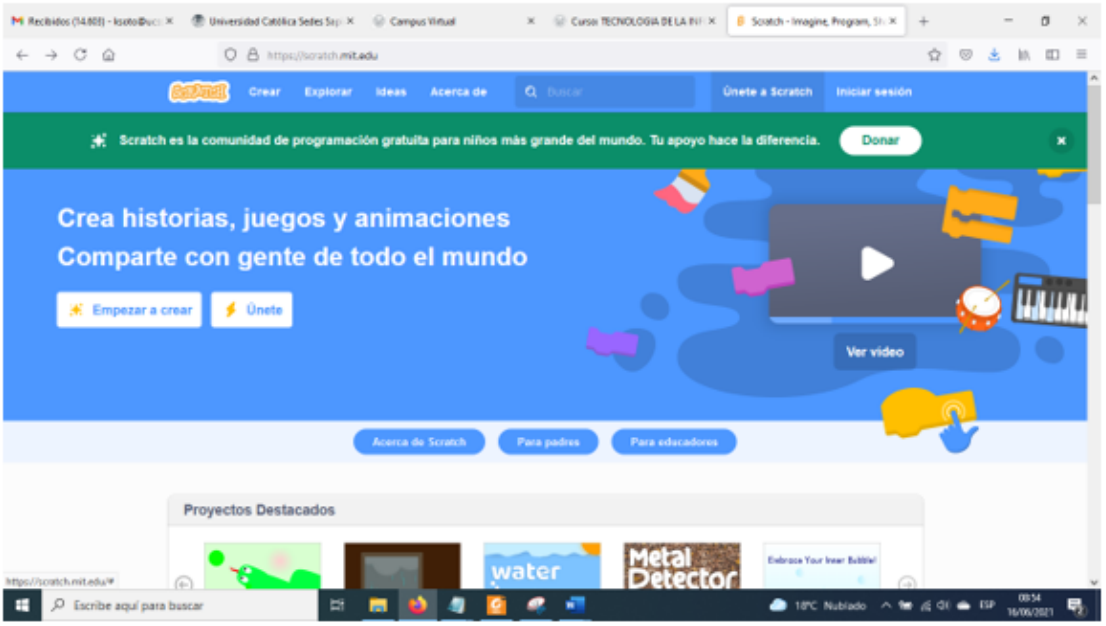
  Por ejemplo, las características generales de los postres son que todos son dulces; pueden ser de frutas, natillas, budines o congelados; y generalmente se sirven al final de una comida. Uno o más postres pueden ser rosados, tener frutas y servirse fríos, mientras que otro tipo puede ser amarillo, tener chispas y no usar frutas.

  El objetivo principal aquí es encontrar patrones que ayuden a simplificar las tareas porque las mismas técnicas de resolución de problemas se pueden aplicar cuando los problemas comparten patrones (el reconocimiento de patrones también se usa en matemáticas, música y literatura, Inteligencia humana, historia, tiempo, etc.).

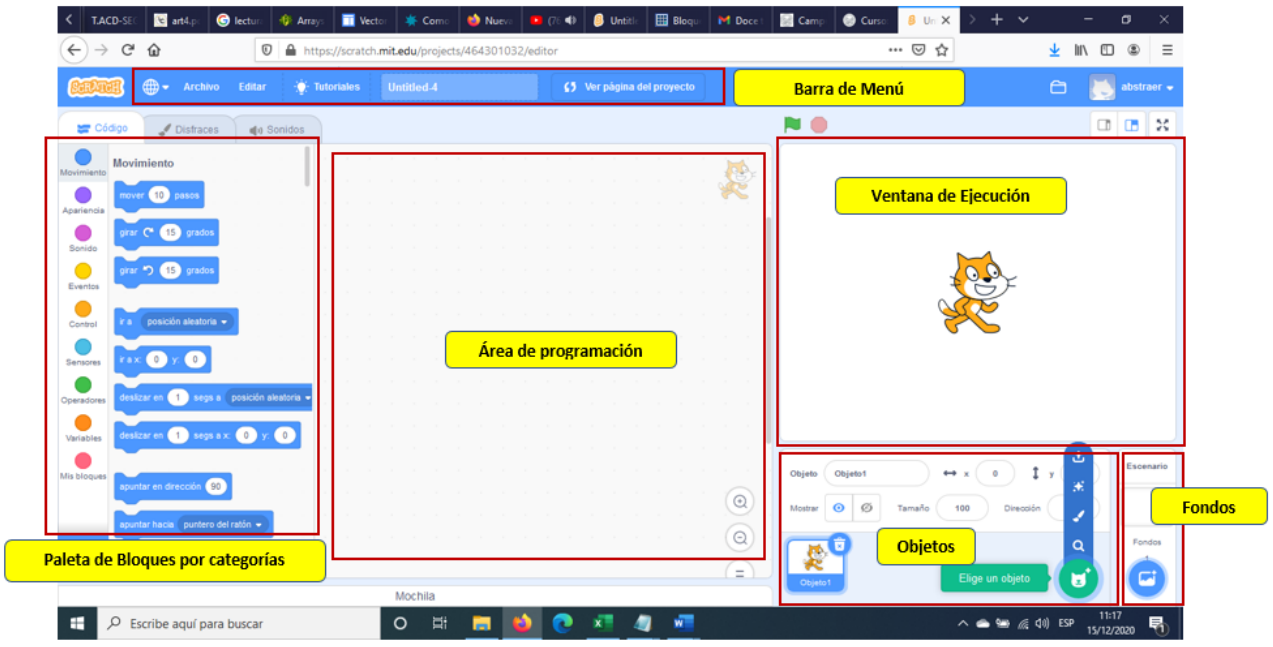
Scratch 3.0

Creación de usuario

Para acceder a Scratch, es necesario acceder al siguiente enlace: https://scratch.mit.edu/
Luego crear una cuenta dándole clic en la opción: Únete a Scratch
En el usuario deberá colocar su código de estudiante y su DNI como contraseña.

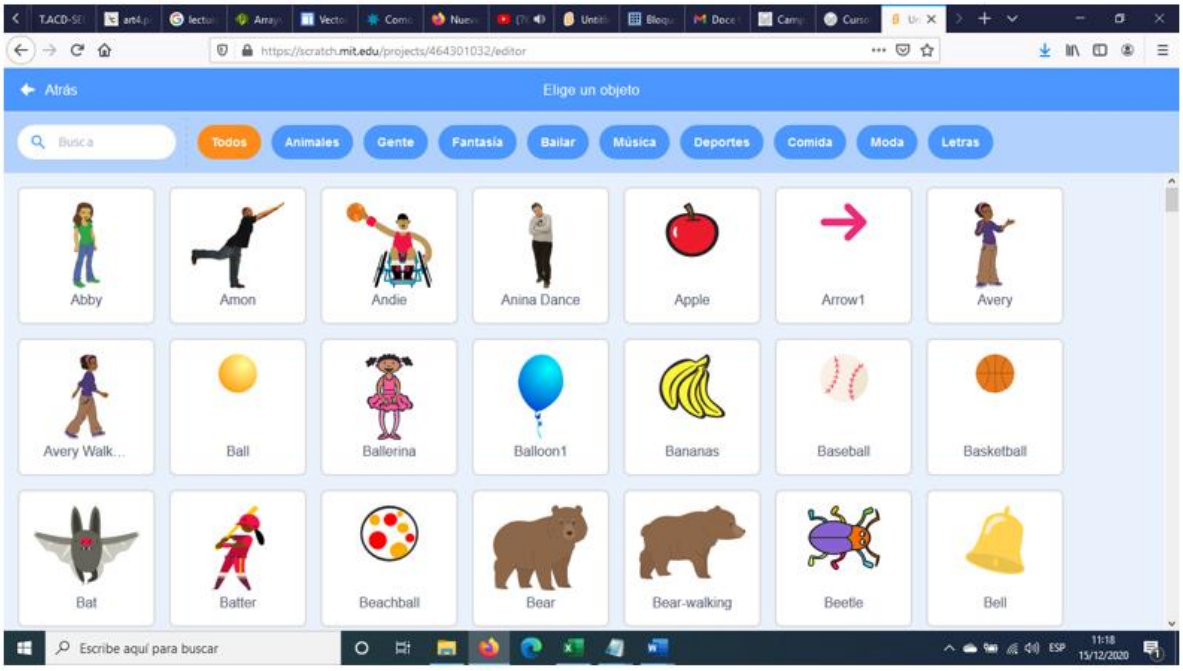


Identificación de la Plataforma Scratch



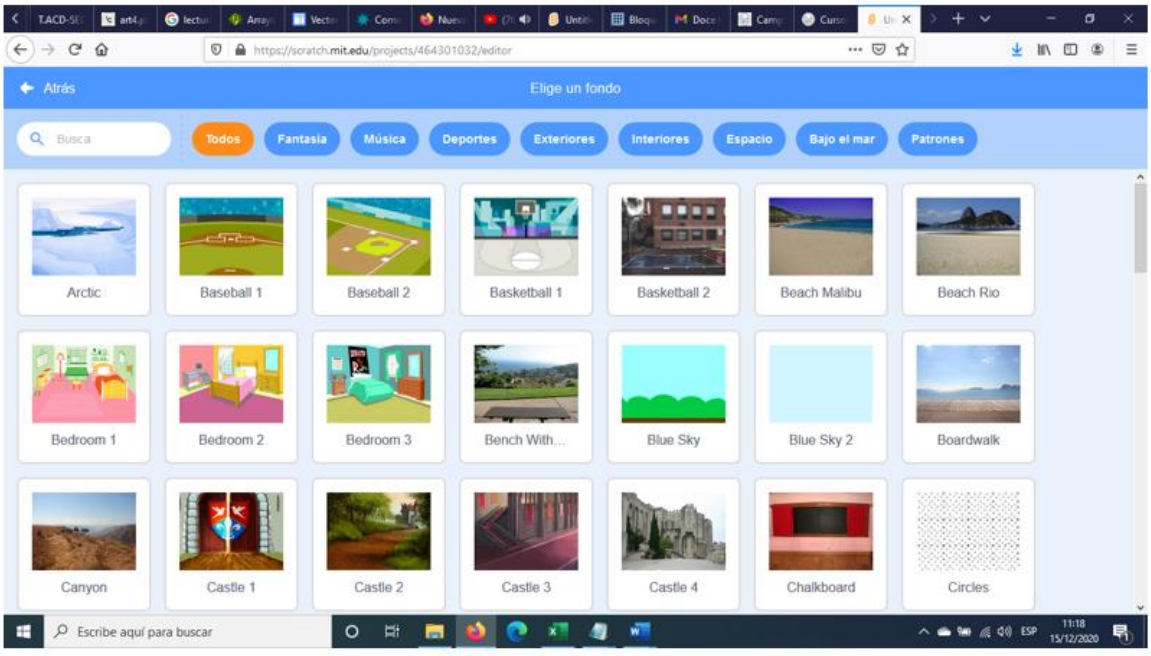
Biblioteca de Objetos

Scratch tiene una **Biblioteca de Objetos** por categorías.



Biblioteca de fondos

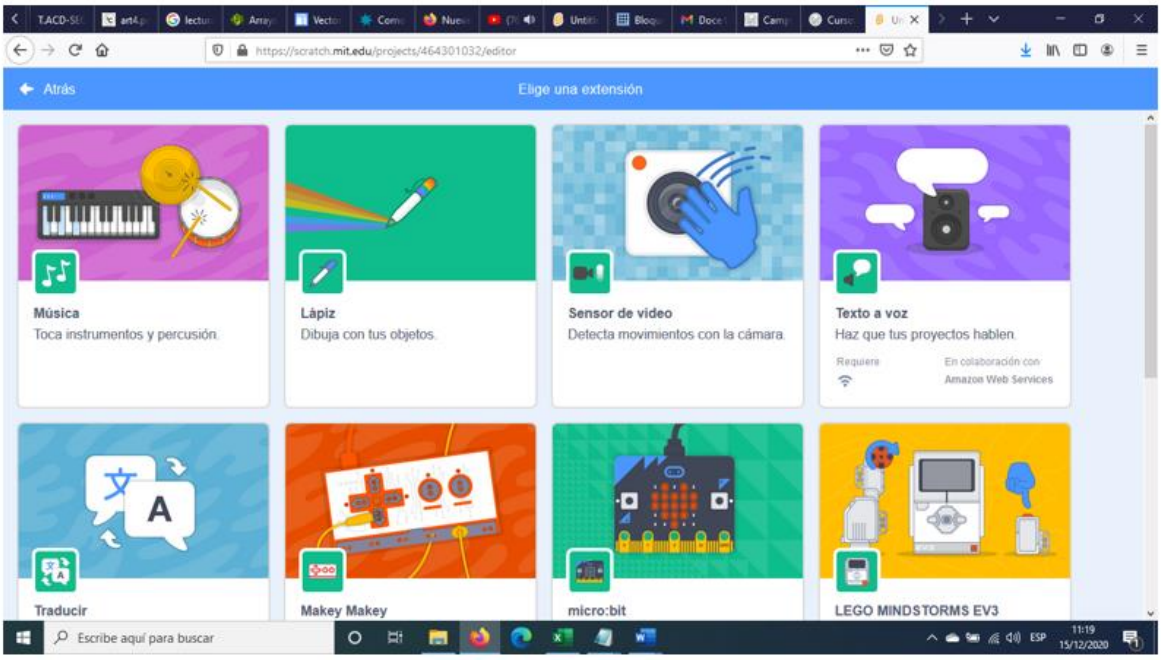
Scratch tiene una **Biblioteca de Fondos** por categorías



Profesora Karol Soto Ccoicca

Extensión

Scratch tiene más categorías en la opción **extensión**, sólo se debe seleccionar para agregar la categoría.



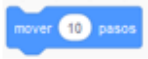
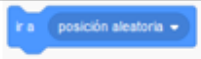
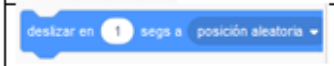
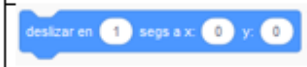
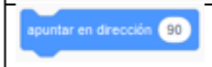
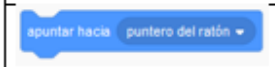
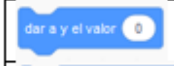
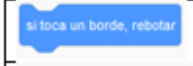
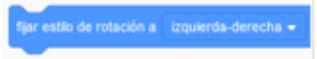
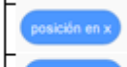
CATEGORÍAS DE SCRATCH

Scratch es un lenguaje de programación orientado a objetos, visual trabaja con bloques como si fueran rompecabezas.

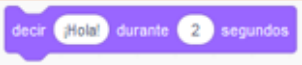
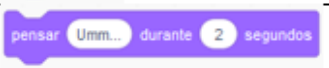
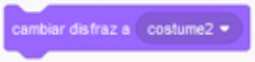
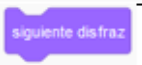
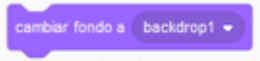
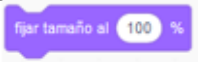
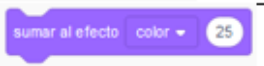
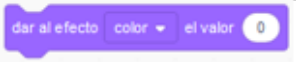
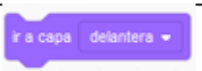
Scratch está compuesto de 9 categorías, entre ellos tenemos:

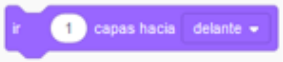
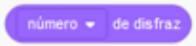
- Movimientos
- Apariencia
- Sonido
- Eventos
- Control
- Sensores
- Operadores
- Variables
- Bloques

Movimientos:

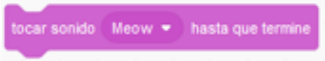
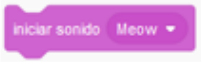
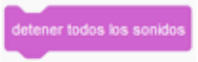
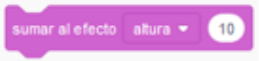
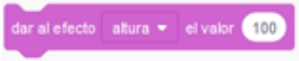
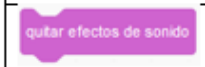
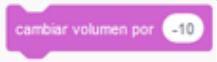
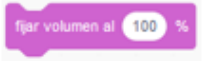
CATEGORÍA: MOVIMIENTOS	
BLOQUES	**DESCRIPCIÓN**
	Mover por el escenario con una cantidad de pasos determinado. Por defecto se muestra 10.
	Girar un determinado número de grados hacia la derecha.
	Girar un determinado número de grados hacia la izquierda.
	Permite que el objeto se mueva a una posición aleatoria, posición del puntero o de otro objeto.
	Permite que el objeto se mueva a la posición X, Y en el escenario.
	Permite al objeto moverse a la posición del puntero o de otro objeto en un tiempo determinado.
	Permite que el objeto se mueva a la posición X, Y en el escenario, en un tiempo específico.
	Permite al objeto mirar hacia adelante, atrás, derecha o izquierda. La dirección 90 hacia adelante.
	Podemos mover al objeto en cualquier dirección. En este caso, hacia el puntero del ratón.
	Sumar un valor a la posición del objeto en el eje X.
	Proporciona un valor determinado a la posición X.
	Sumar un valor a la posición del objeto en el eje Y.
	Proporciona un valor determinado a la posición Y.
	Gira el objeto en sentido contrario si éste está tocando un borde.
	Determinar el estilo de rotación del objeto.
	Informa sobre la dirección del objeto en el eje x.
	Informa sobre la dirección del objeto en el eje y.

Apariencia

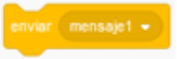
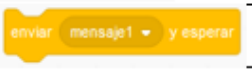
CATEGORÍA: APARIENCIA	
BLOQUES	**DESCRIPCIÓN**
	Despliega la nube de diálogo del objeto durante un tiempo determinado.
	Despliega la nube de diálogo del objeto y se mantiene durante todo el programa.
	Despliega la nube de pensamiento del objeto durante un tiempo determinado.
	Despliega la nube de pensamiento del objeto.
	Modifica la apariencia del objeto, cambiando a un disfraz determinado.
	Modifica la apariencia del objeto cambiando a otro disfraz.
	Modifica el fondo, cambiando a un fondo o imagen determinada.
	Modifica el fondo cambiando a otro fondo o imagen.
	Permite modificar el tamaño del objeto, ya sea aumentando o disminuyéndolo.
	Recupera el tamaño del objeto original al fijar al 100% o al porcentaje que seleccionemos.
	Aumenta un determinado valor al efecto color del objeto. (tonos)
	Proporciona un determinado valor al efecto color del objeto. (tonos)
	Elimina los efectos gráficos utilizados, recuperando el estado inicial.
	Hace aparecer un objeto en el escenario.
	Hace desaparecer un objeto del escenario.
	Permite que el objeto sea colocado en una determinada capa.

	Permite mover al objeto un determinado número de capas hacia una posición específica.
	Informa el número correspondiente al presente disfraz que se está usando.
	Informa el número correspondiente al presente fondo que se está usando.
	Informa el tamaño del objeto.

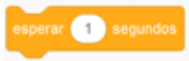
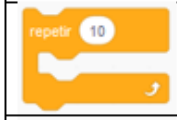
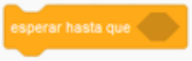
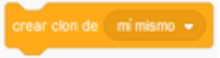
Sonido

CATEGORÍA: SONIDO	
BLOQUES	**DESCRIPCIÓN**
	Reproduce un sonido determinado hasta que el programa termine.
	Comienza a reproducir un sonido determinado.
	Detiene todos los sonidos que se estén reproduciendo.
	Aumentar al efecto altura un determinado valor.
	Para dar al efecto altura un determinado valor.
	Elimina los efectos de sonido utilizados.
	Modifica el volumen del sonido, ya sea aumentando o disminuyéndolo.
	Fija el volumen del objeto a un porcentaje específico.
	Informa el volumen del sonido del objeto.

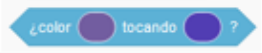
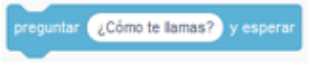
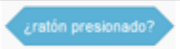
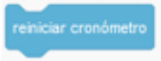
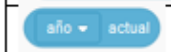
Eventos

CATEGORÍA: EVENTOS	
BLOQUES	**DESCRIPCIÓN**
	Se inicia el juego al presionar la bandera verde.
	Se ejecuta el programa debajo del bloque al presionar una determinada tecla.
	Se ejecuta el programa debajo del bloque al dar clic a un determinado objeto.
	Se realizará una determinada acción cuando el fondo o imagen cambie a otro.
	Se realizará una determinada acción cuando el volumen del sonido sea mayor a un número determinado.
	Se ejecuta el programa debajo del bloque al recibir un mensaje determinado.
	Se ejecuta el programa debajo del bloque al enviar un mensaje determinado.
	Se envía un mensaje determinado y espera la acción debajo del bloque.

Control

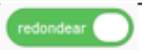
CATEGORÍA: CONTROL	
BLOQUES	**DESCRIPCIÓN**
	Esperar por un tiempo determinado.
	Repetir un determinado número de veces el programa que queremos ejecutar.
	Repetir indefinidamente el programa que queremos ejecutar cuando se cumpla una condición específica.
	Si se cumple una condición específica el programa que queremos se ejecutará.
	Si se cumple una condición específica el programa que queremos se ejecutará. De lo contrario, se ejecutará otro programa (el que está debajo del bloque de si no).
	Esperar hasta que una condición específica se cumpla.
	Repetir el programa que queremos hasta que se cumpla una condición específica.
	Permite detener el objeto.
	Se realizará una acción determinada al comenzar como clon.
	Permite crear un clon de un determinado objeto.
	Permite eliminar a un determinado clon del objeto.

Sensores

CATEGORÍA: SENSORES	
BLOQUES	**DESCRIPCIÓN**
	Pregunta si el objeto está tocando un objeto específico.
	Pregunta si el objeto está tocando un color específico.
	Pregunta si un color específico está tocando a otro.
	Informa la distancia desde un objeto específico.
	Preguntar y esperar a que se ingrese la respuesta debajo del bloque.
	Guarda la respuesta contestada, para utilizarla posteriormente.
	Pregunta si una tecla específica está siendo presionada.
	Pregunta si el ratón se encuentra presionado.
	Informa la posición del ratón en el eje X.
	Informa la posición del ratón en el eje Y.
	Fija un modo de arrastre a un objeto.
	Informa el volumen del sonido del objeto, captado por el micrófono.
	Informa el valor del cronómetro en segundos.
	Vuelve a iniciar el cronómetro.
	Permite que una imagen o fondo sea colocado como escenario.
	Permite colocar el año actual.
	Informa la cantidad de días que pasaron desde el 2000.
	Informa el nombre del usuario.

Operadores

Operadores Matemáticos:

OPERADORES MATEMÁTICOS	
BLOQUES	**DESCRIPCIÓN**
	Suma dos variables.
	Restar dos variables.
	Multiplicar dos variables.
	Dividir dos variables.
	Elige un número al azar entre un rango determinado.
	Nos da el resto de la división de los números indicados.
	Aproxima o redondea un número.
	Da el resultado de una función específica de número determinado.

Operadores de Comparación:

OPERADORES DE COMPARACIÓN	
BLOQUES	**DESCRIPCIÓN**
	Informa verdadero, si el primer número es mayor al segundo.
	Informa verdadero, si el primer número es menor al segundo.
	Informa verdadero, si el primer número es igual al segundo.

Operadores Booleanos:

OPERADORES BOOLEANOS	
BLOQUES	**DESCRIPCIÓN**
	Informa verdadero, si ambas condiciones son verdaderas.
	Informa verdadero, si una de las condiciones es verdadera.
	Si la condición es falsa, informa verdadero. Y si la condición es verdadera, informa falso.

Operadores de Cadena:

OPERADORES DE CADENA	
BLOQUES	**DESCRIPCIÓN**
	Combina cadena de letras.
	Indica que letra ocupa la posición adecuada.
	Indica la cantidad de letras que contiene una palabra específica.
	Pregunta si una letra se encuentra dentro de una palabra determinada.

Variables

CATEGORÍA: VARIABLES	
BLOQUES	**DESCRIPCIÓN**
Crear una variable	Permite crear una nueva variable.
my variable	Indica el nombre de la variable.
dar a my variable el valor 0	Permite dar un valor específico a una variable determinada.
sumar a my variable 1	Sumar un determinado valor a una variable determinada.
mostrar variable my variable	Muestra una variable específica.
esconder variable my variable	Esconde una variable específica.
Crear una lista	Permite crear una lista.

Bloques

CATEGORÍA: BLOQUES	
BLOQUES	**DESCRIPCIÓN**
Crear un bloque	Permite crear un bloque.

Sesión 3

Variable

VARIABLES

Definición:

Es un espacio de la memoria donde se almacenan los datos ingresados.

Las variables se crean en la memoria RAM durante la ejecución del programa, cuando finaliza la ejecución se libera la memoria de las variables creadas.

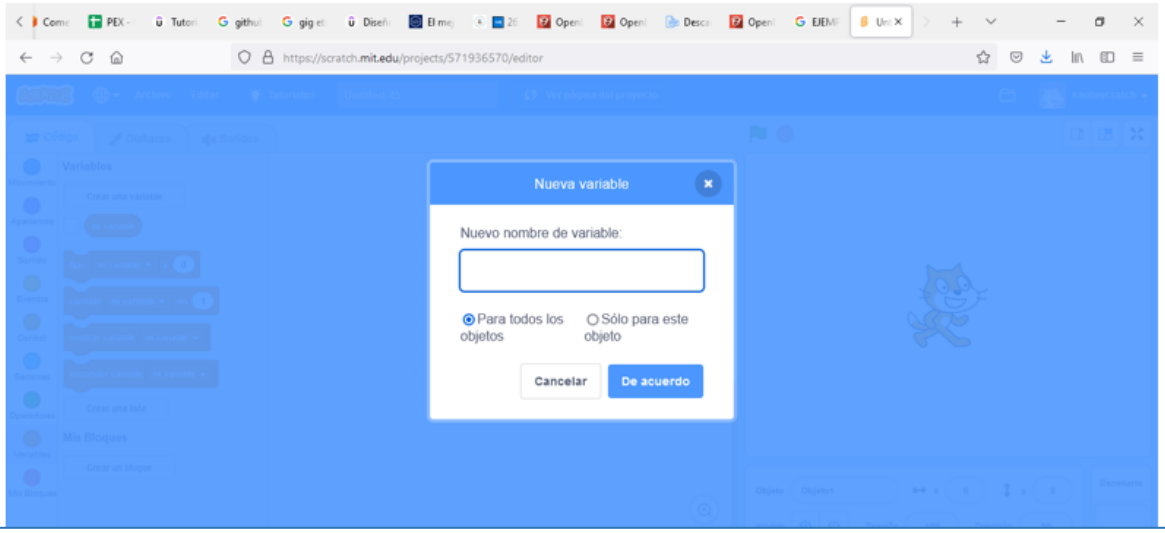
Los nombres de las variables deben ser mnemotécnicos, es decir se debe sobreentender qué valores se almacenarán en dichas variables.

Consideraciones para crear variables:

- No debe empezar con mayúscula.
- Puede incluir número, pero no al inicio.
- No debe contener la letra ñ
- No debe tener espacio en blanco
- No debe contener tilde

Creación de variables:

Para crear una variable debemos ir a la categoría: variable/ crear una variable y luego colocar un nombre.



Estructura del análisis y desarrollo de un problema

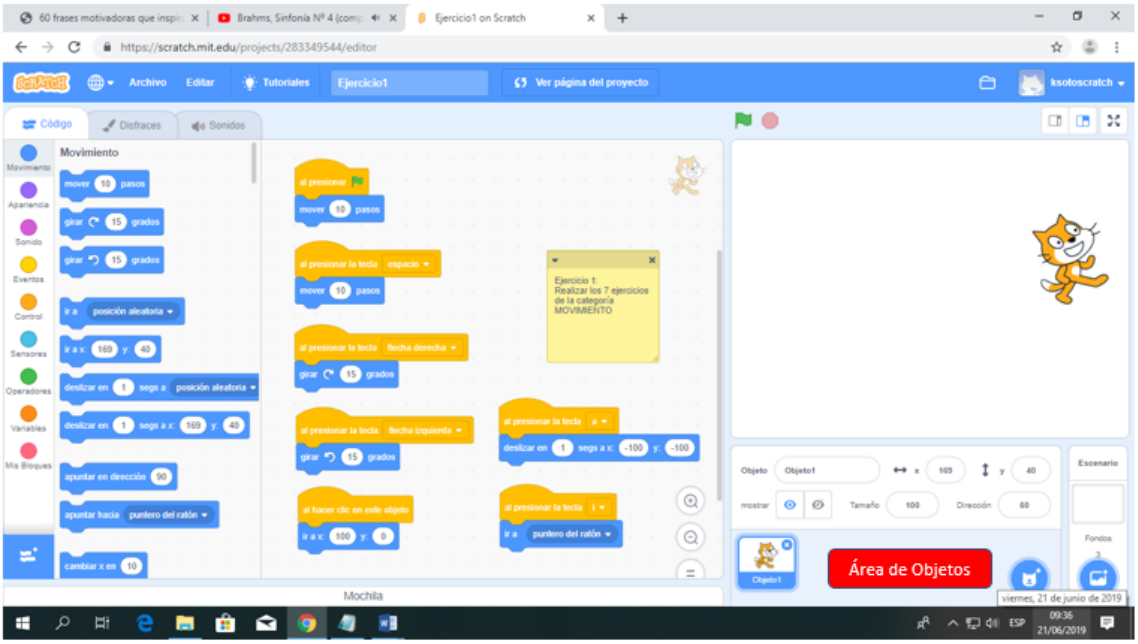
Para identificar las funcionalidades que debe tener un programa en Scratch o en otro Lenguaje de Programación, se necesita realizar los siguientes pasos por cada enunciado:

1	Identificar los datos que nos da el problema
2	Identificar las entradas que solicita el programa
3	Identificar las restricciones o reglas de negocio
4	Definir el proceso, es decir el algoritmo a implementar
5	Mostrar la salida o pantalla de ejecución

EJERCICIO 1:

Para crear un nuevo proyecto darle clic en el menú **Archivo/ Nuevo**, luego asignarle el nombre **ejercicio1**, seguidamente acceder a la categoría evento y arrastrar el evento indicado, repetir el mismo procedimiento para la categoría **Movimiento.**

IMPORTANTE: Para programar debemos seleccionar el objeto en el área de Objetos, y ese mismo objeto debe estar como sello de agua en el Área de Programación.



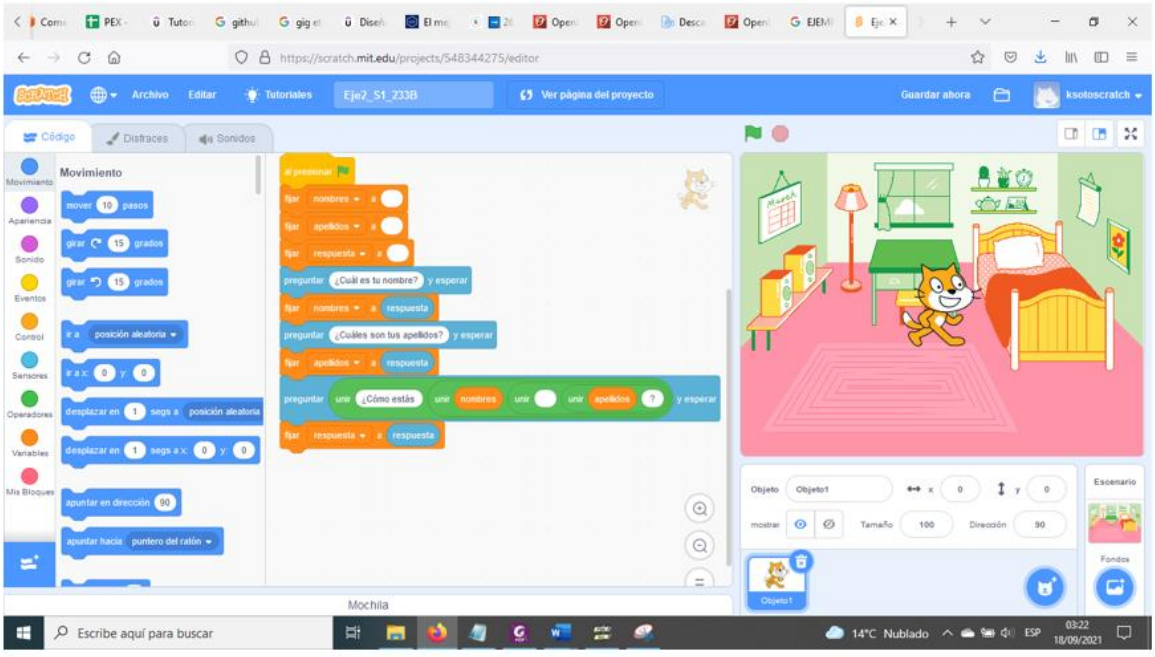
EJERCICIO 2:

Realizar un programa, donde **Michu** le solicite ingresar su nombre y apellidos, y al final le responda **Hola! ¿Cómo estas tu nombre y apellidos?**

Primer Paso: Se debe crear 2 **variables**: nombre y apellidos, en la categoría variables, se puede crear para todos los objetos, o solo para el objeto que estamos programando.

VARIABLES: En programación, las variables son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador (RAM).

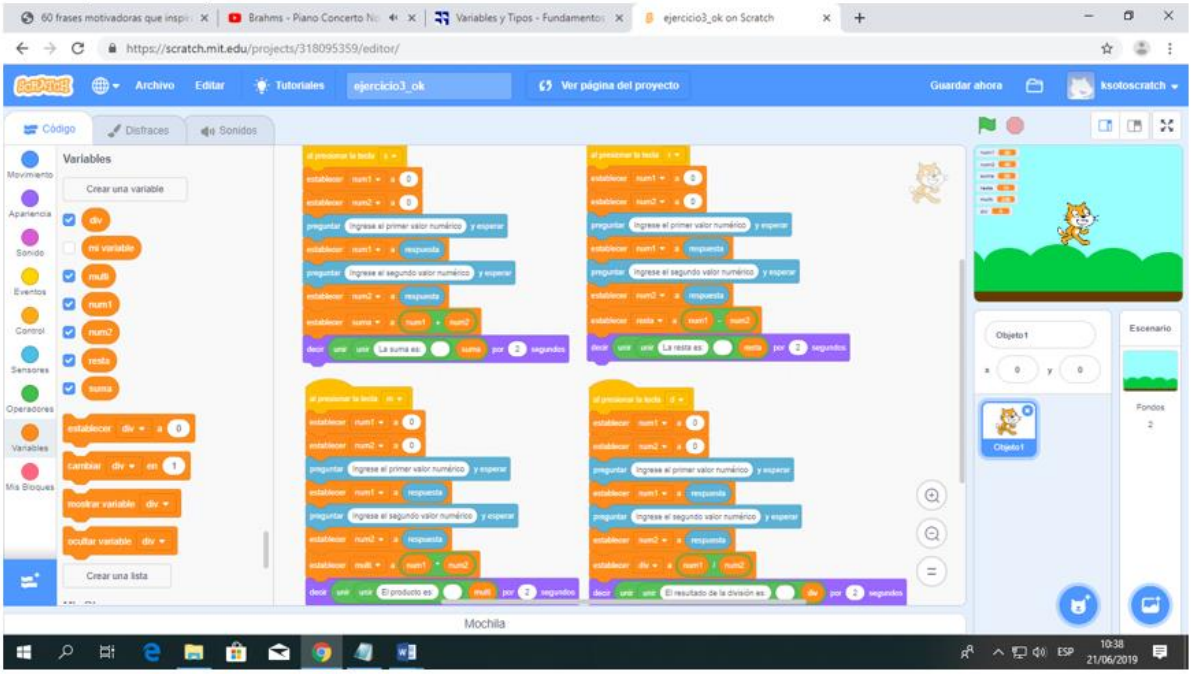
Segundo Paso: Arrastra las instrucciones o bloques, luego concatena las variables con el bloque **unir**.



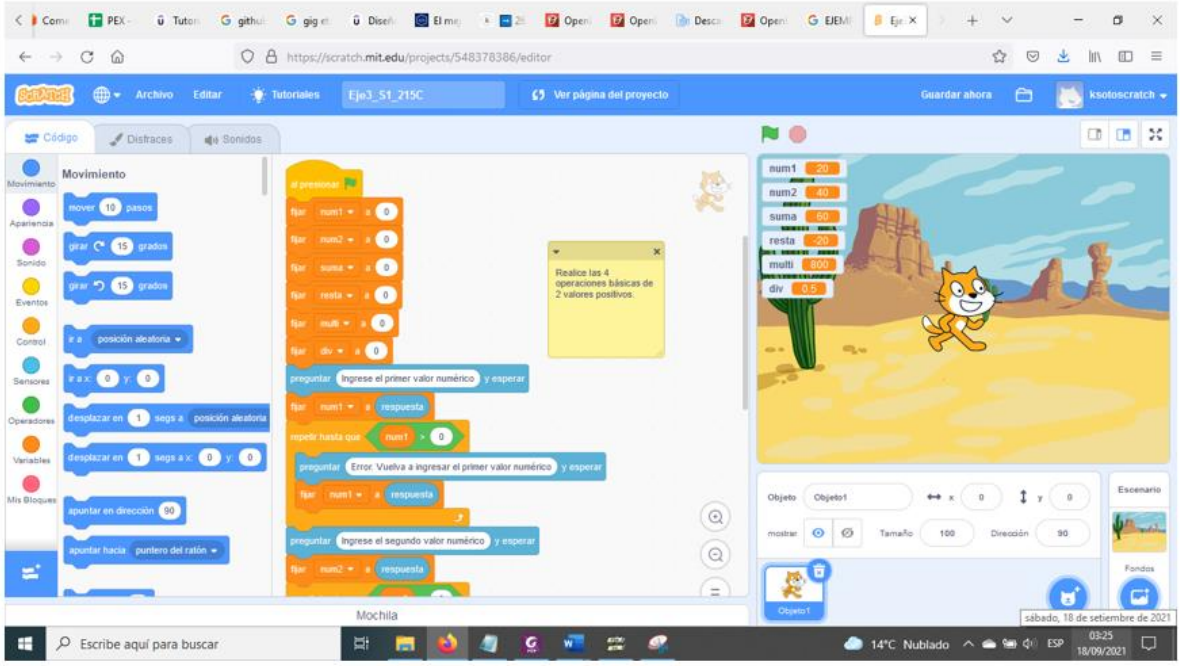
Ejercicio 3:

Realizar 4 programas que realicen la suma, resta, multiplicación y división, ejecutándose con la tecla s=suma, r=resta, m=multiplicación, d=división, para ello debemos de crear las variables: num1, num2, suma, resta, multi, div **(las variables no deben tener tildes ni espacios en blanco)**

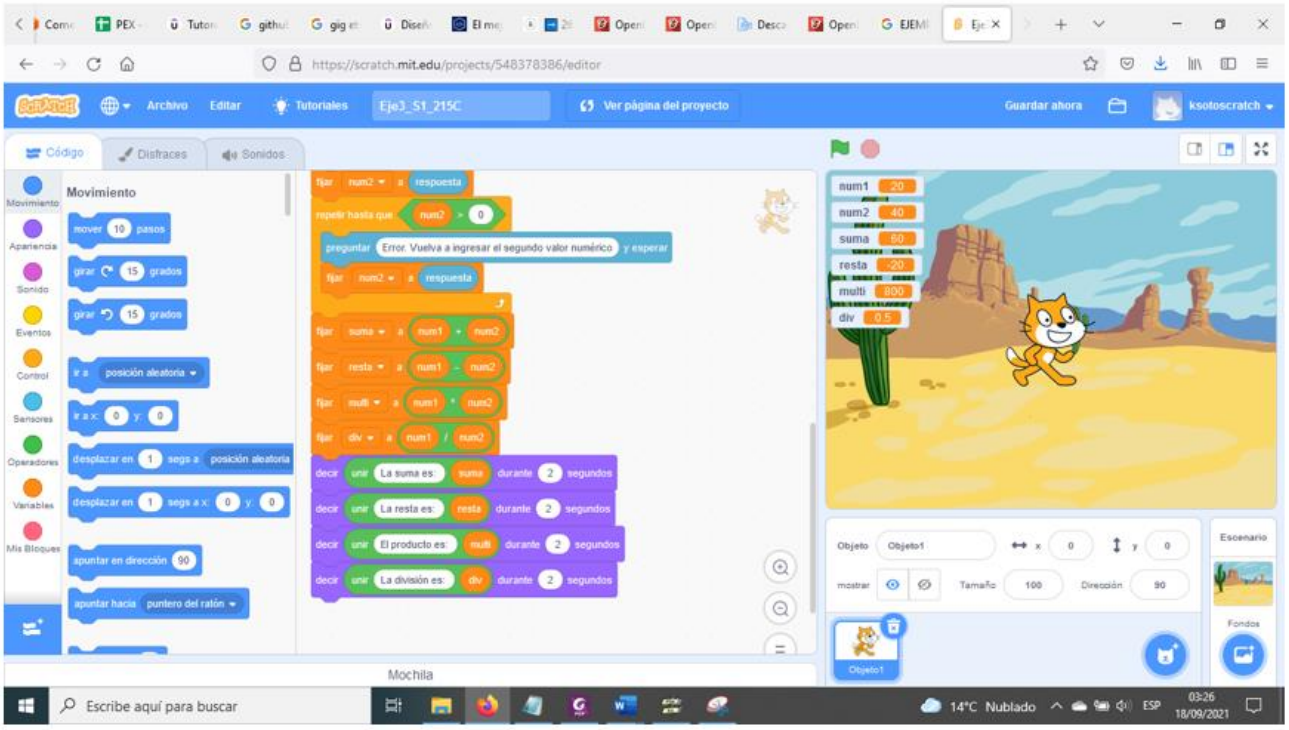
Forma 1: Ejecutando 4 bloques



Forma 2: Ejecutando en un solo bloque las 4 operaciones.

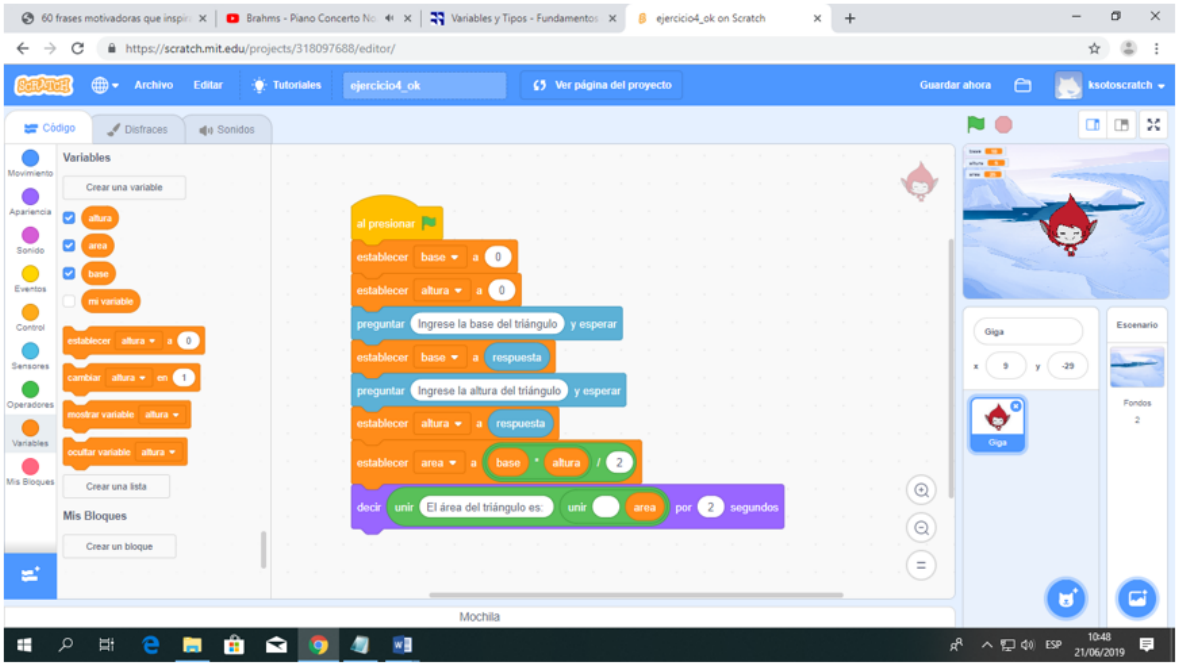


Forma 2: Ejecutando en un solo bloque las 4 operaciones.



Ejercicio4:

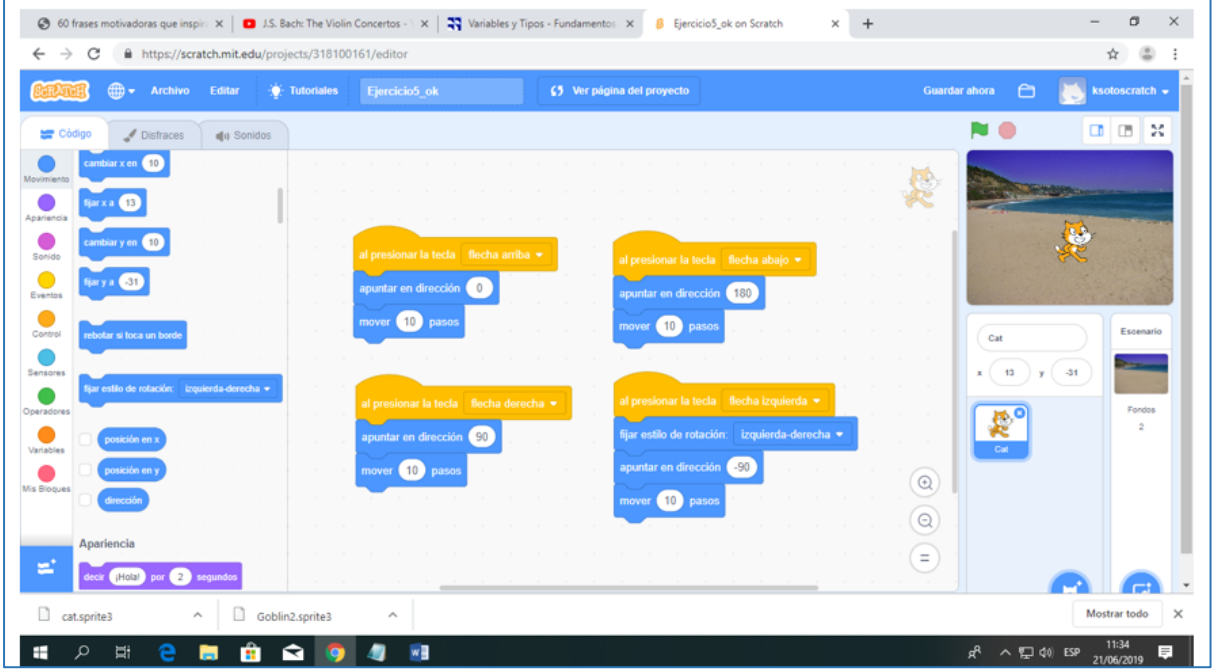
Realizar un programa, que me halle el área de un triángulo, para ello se debe de crear las variables: base, altura y area.



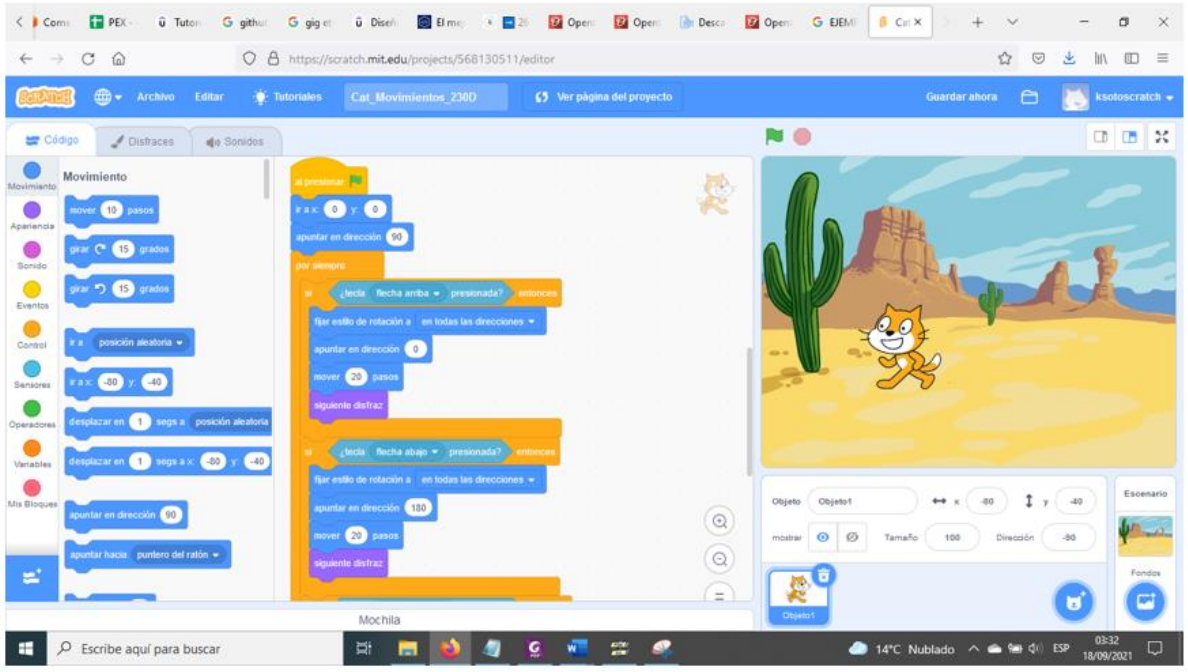
Ejercicio5:

Realice un programa, en donde **Michu** de desplace de arriba, abajo, izquierda y derecha.

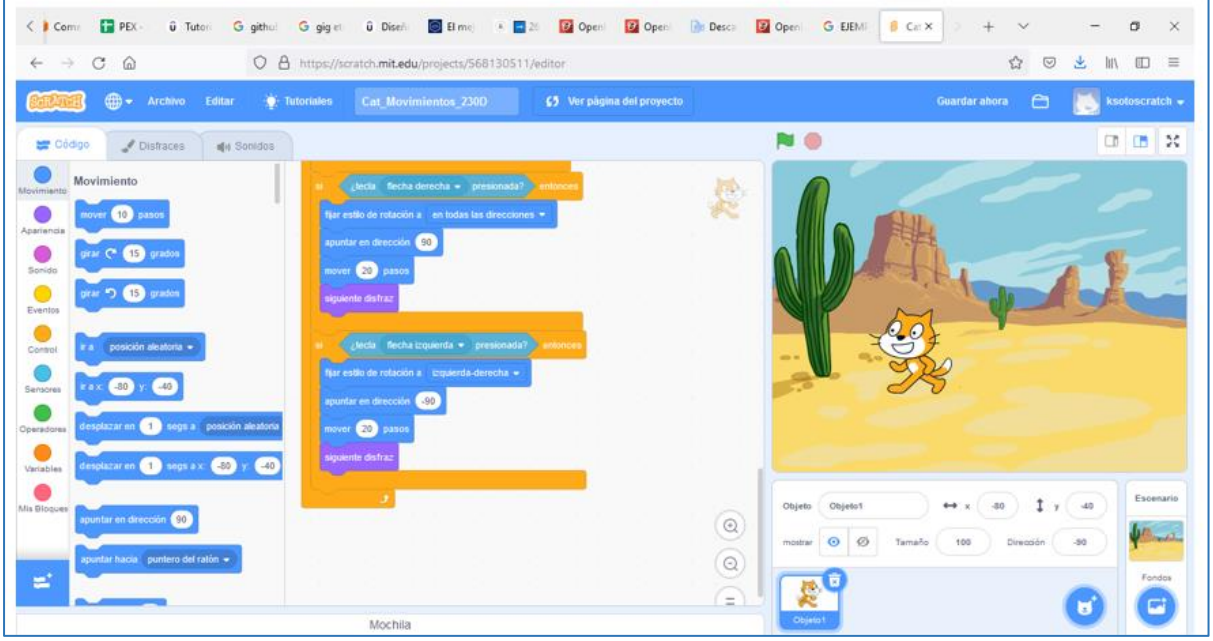
Forma 1:



Forma 2:



Forma 2:



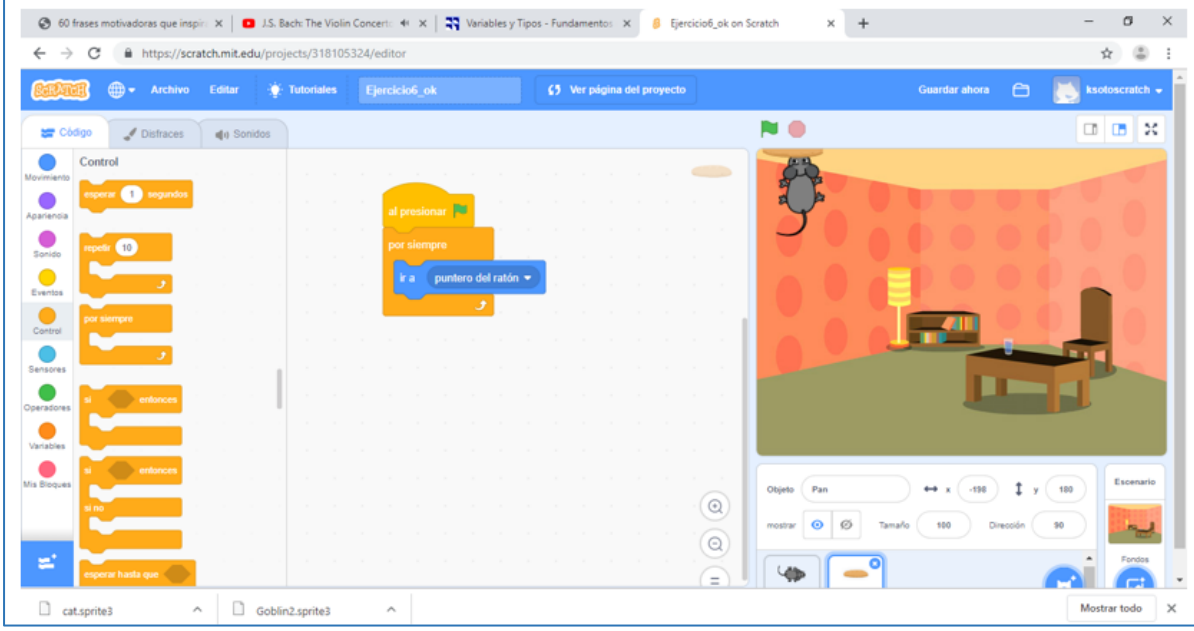
Ejercicio6:

Realizar un programa, que incluya 2 objetos: **Pan y ratón**, el objeto Ratón deberá seguir al objeto pan, por siempre.

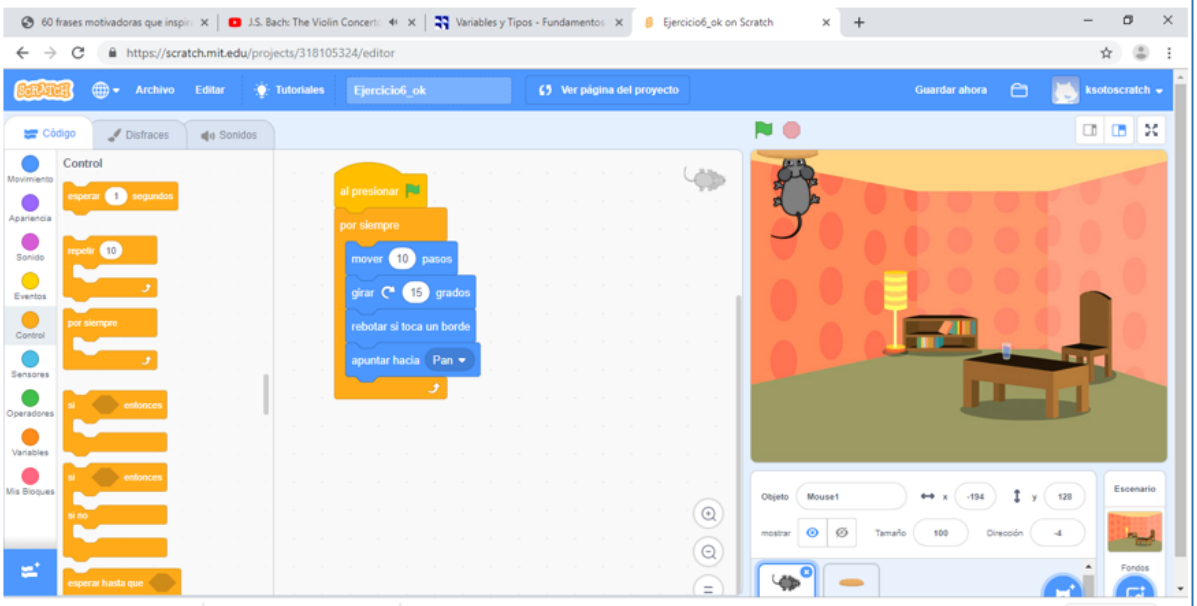
Incluir de la categoría **Control**, la instrucción **Por siempre** (Este bloque hará que se repita por siempre, es decir hasta que termine el programa).

Se tiene que realizar la programación para los 2 objetos.

OBJETO: PAN



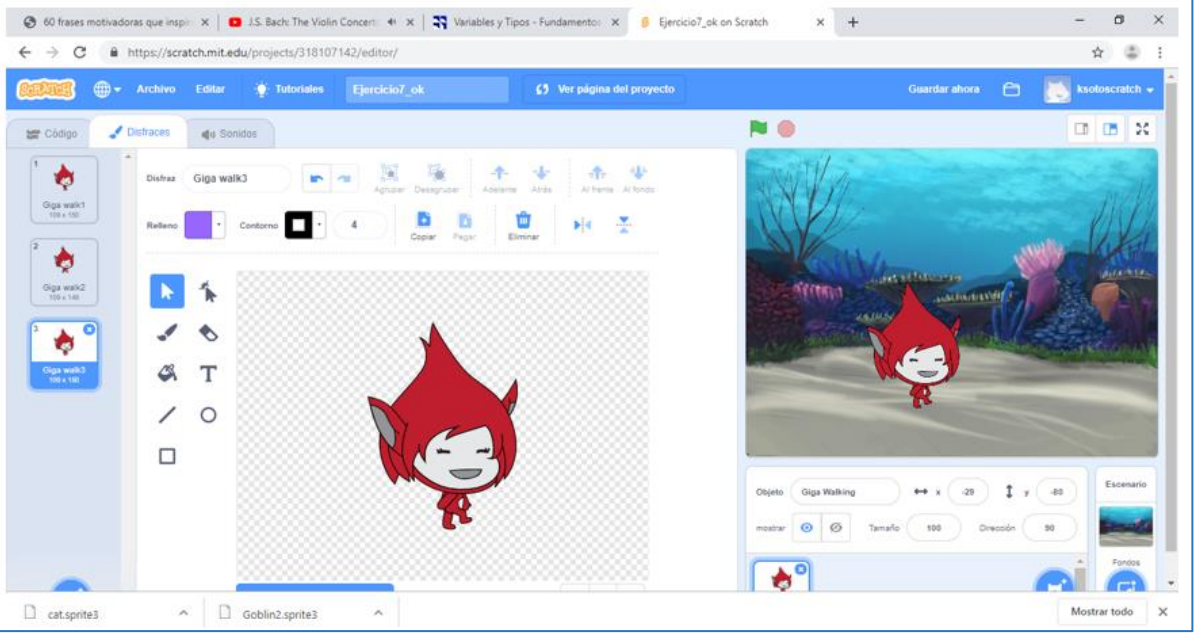
OBJETO: RATÓN



Ejercicio7:

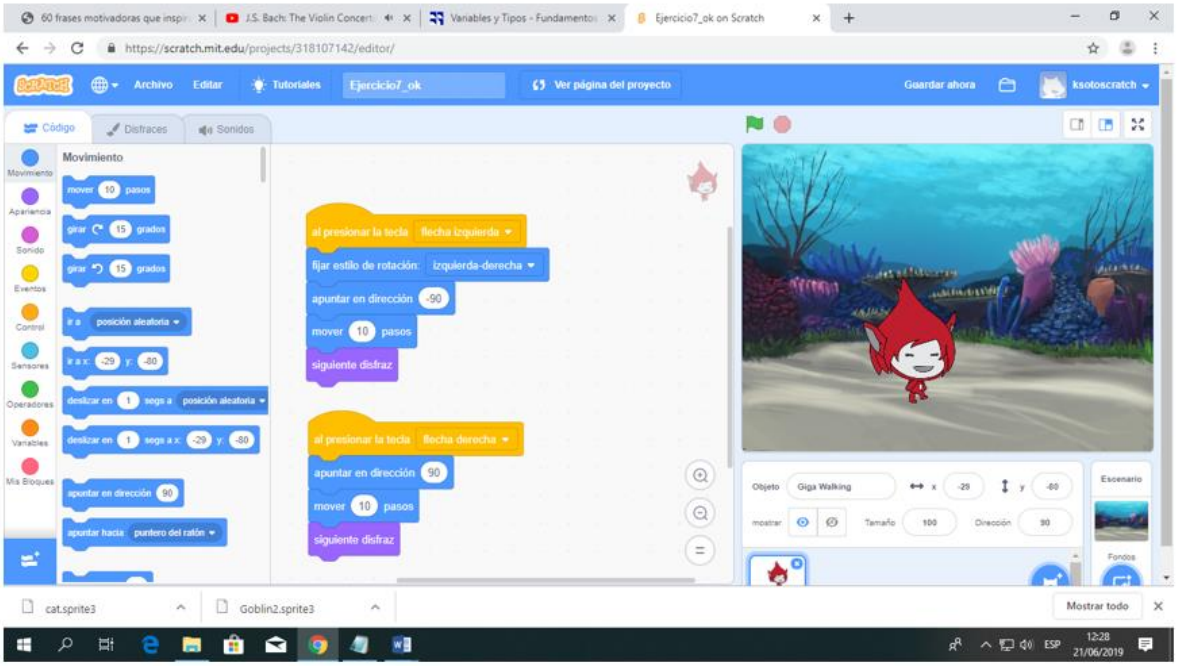
Realizar un programa en el que se muestre a Giga caminando, en ambas direcciones (De izquierda a derecha o de derecha a izquierda) este objeto tiene **3 disfraces**. Giga se desplazará sobre un escenario al inicio, luego se mostrará en otro escenario.

DISFRACES DE GIGA: Al seleccionar a Giga, automáticamente se agregarán sus 3 disfraces.



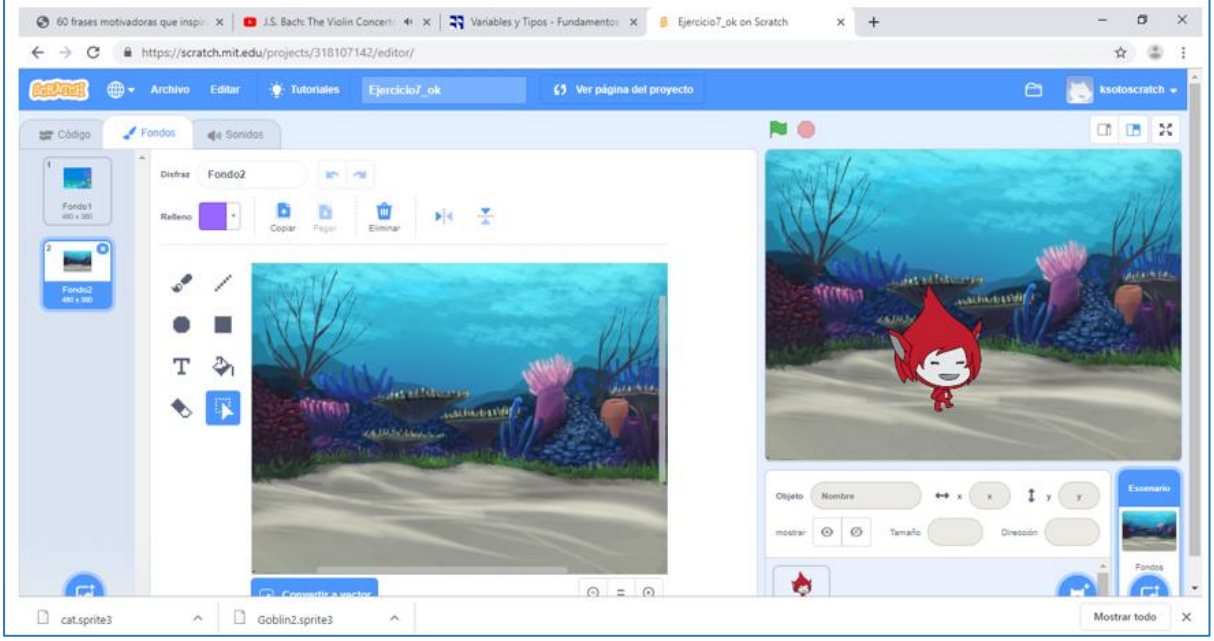
OBJETO: GIGA

Realizar la siguiente programación al objeto Giga.



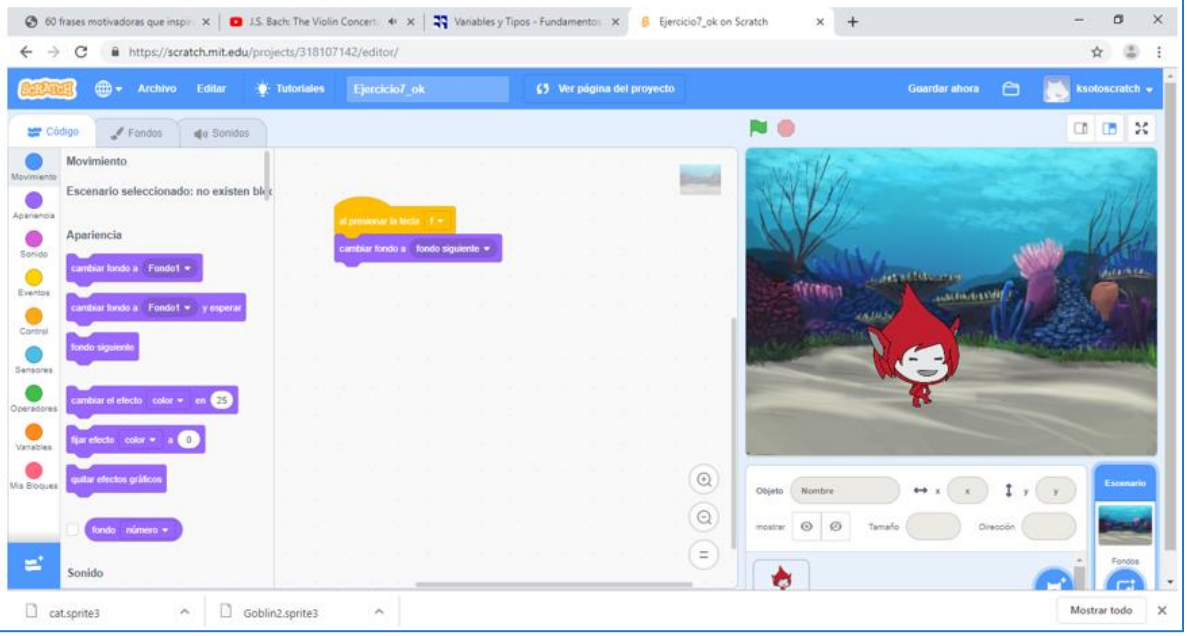
OBJETO: FONDO

En escenario, seleccionar 2 fondos, y renombrarlo con fondo1 y fondo2.



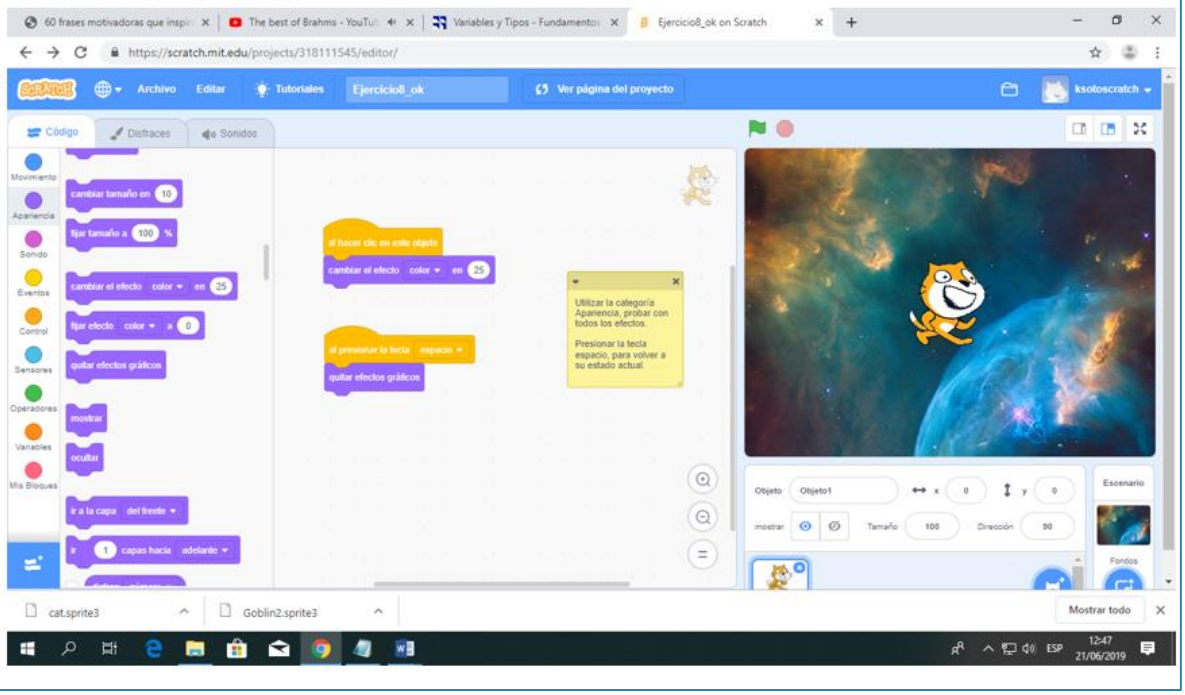
OBJETO: FONDO

Seleccionar el escenario y realizar la siguiente programación:



Ejercicio8:

Aplicar efectos al objeto Michu, al hacer clic sobre el objeto, debe mostrarse el efecto que tiene asignado, para volver a su estado inicial, presione la tecla espacio, luego al objeto aplicarle el siguiente efecto, probar con todos los efectos.



Sesión 4

Ejercicios Propuestos:

Renombrar los ejercicios como, por ejemplo: **Pro1_AreaCuadrado**

1. Hallar el área de cuadrado.
2. Hallar el área de un rectángulo.
3. Hallar el área de un círculo.
4. Realice un algoritmo que eleve a la potencia un valor ingresado.
5. El usuario debe ingresar dos números y el programa mostrará el resultado de la operación (a+b)*(a-b).
6. Realice un programa que solicite el ingreso de un número decimal, utilizando las funciones matemáticas devolver el valor absoluto, raíz cuadrada, parte entera, redondeo del valor ingresado.
7. Diseñe un programa que reciba del usuario el valor de la temperatura en grados Celsius y muestre el resultado convertido a Fahrenheit.

$$F = (\frac{9}{5} * C) + 32$$

8. Realice un programa que solicite el ingreso de su nombre, luego sus apellidos, utilizando las funciones de cadena: concatenar ambos textos, mostrar la longitud de caracteres de su nombre completo.
9. Generar un número aleatorio entre 0 y 20.
10. Realizar un programa que calcule el promedio de 3 notas ingresadas, al final debe mostrarle su promedio e indicarle si está aprobado o desaprobado.
11. Dado un monto calcular el descuento, considerando que por encima de S/. 350, el descuento es del **35%**, caso contrario **10%**.
12. Dada la duración en minutos de una llamada telefónica, calcular su costo, de la siguiente manera: Hasta 5 minutos el costo es **0.90**. Por encima de los 5 min, el costo es **0.90 + 0.20**, por cada minuto adicional a los 5 primeros minutos

Sesión 5

FUNDAMENTOS DE PROGRAMACIÓN

MANUAL DE SCRATCH

PARTE 2

INDICE

ESTRUCTURAS DE CONTROL EN SCRATCH

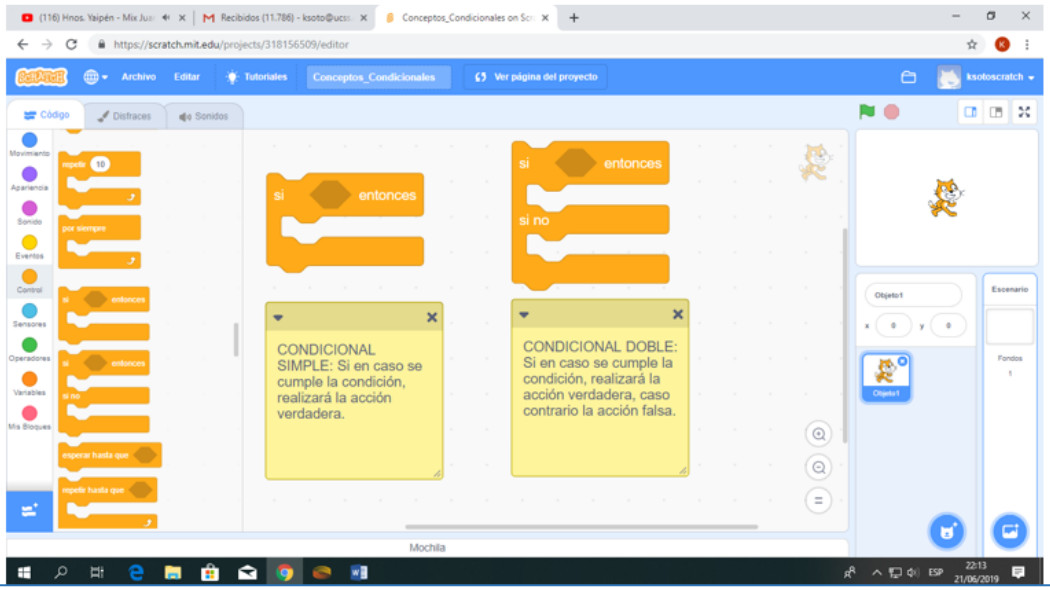
Estructuras Condicionales

Se utilizan para la toma de decisiones, en programación una sentencia condicional es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición.

Las condiciones tienen forma de bloque de extremos en ángulo recto.

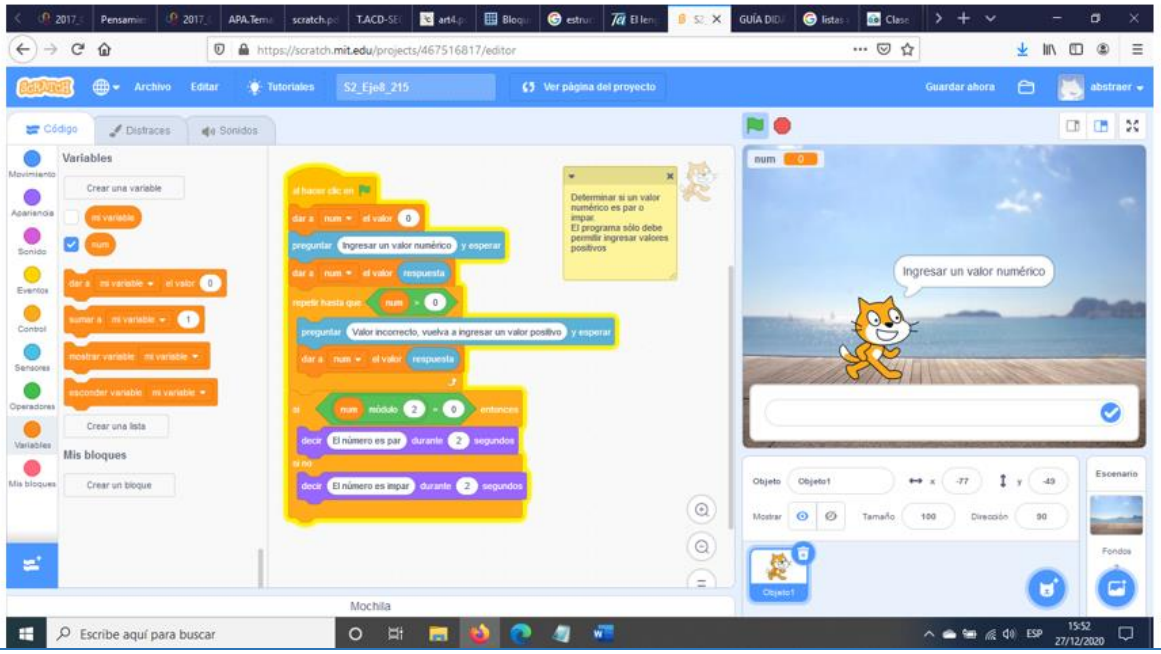
Para establecer las condiciones se utilizan los **operadores de comparación** (>, <, =)

Para unir dos o más condiciones se utiliza los **operadores booleanas**: **conjunción, disyunción y negación**, que se escriben and, or, not (y, o, no, respectivamente).



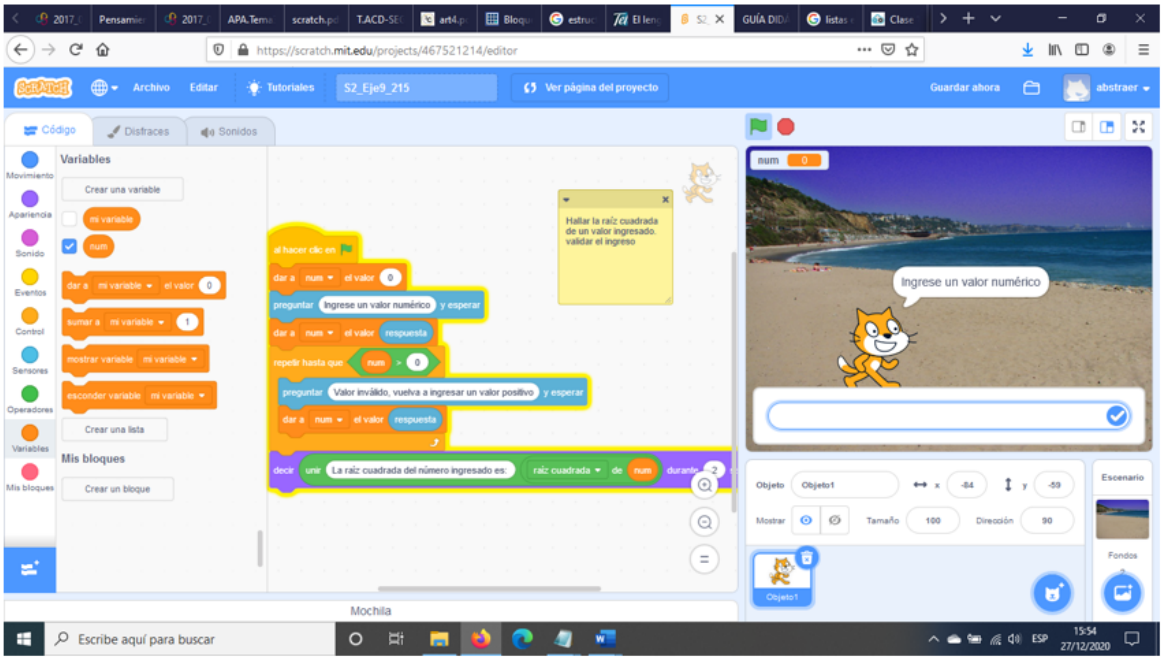
Ejercicio8:

Realizar un programa que determine si un valor numérico es par o impar, validar que ese valor ingresado sea positivo.



Ejercicio9:

Hallar la raíz cuadrada de un valor numérico, validar que la entrada sea un valor positivo.



Estructuras Condicionales Anidadas

Cuando utilizamos una condición dentro de otra estructura condicional, según el grado de complejidad del problema planteado.

Ejercicio10:

Realizar un programa, que solicite el ingreso del promedio final del curso de TIC, el promedio debe ser validado, es decir debe estar dentro del rango de 0 a 20, caso contrario, debe mostrarle el siguiente mensaje "Nota inválida, vuelva a ingresar un promedio dentro del rango establecido".

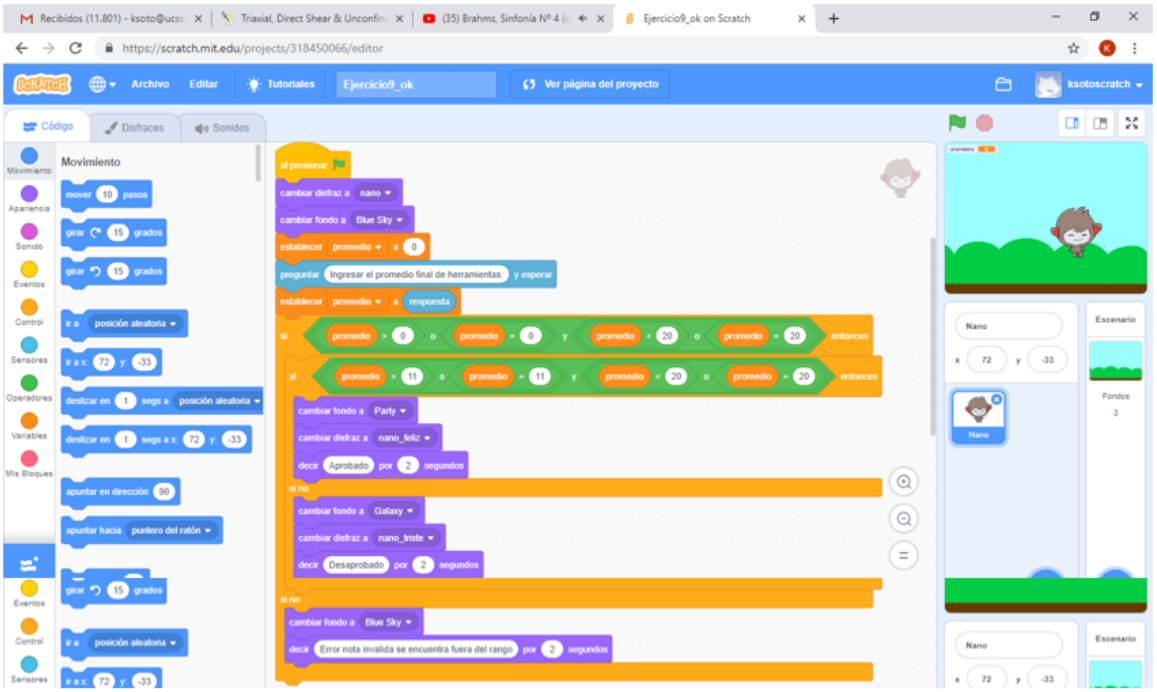
Si la nota es válida:

a) Debe mostrar, el mensaje: "Estas Aprobado", si su nota es mayor a 10.4,
b) caso contrario "Estas Desaprobado".

Si es que está aprobado, se debe agregar un fondo agradable, también deberá cambiar de disfraz a uno contento, **caso contrario** fondo desagradable y disfraz triste.

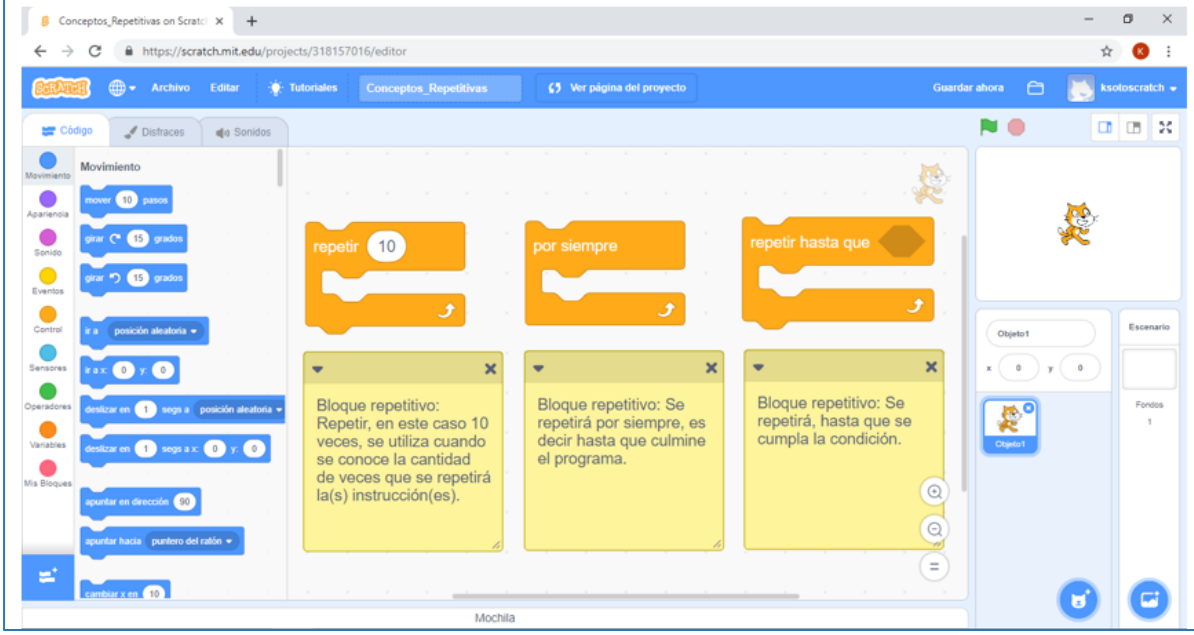
Solución del Ejercicio10:

Se está validando el ingreso del promedio con una estructura condicional, otra opción es utilizar una estructura repetitiva para la validación.



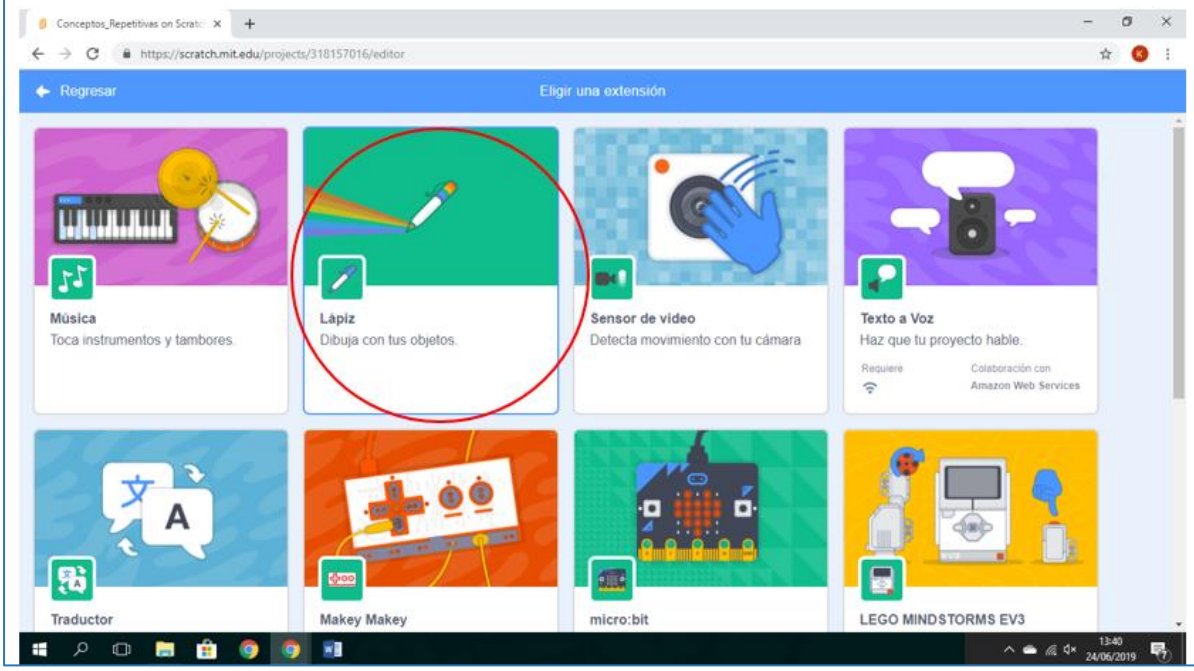
Estructuras Repetitivas

La estructura iterativa o de repetición permite ejecutar una o varias instrucciones un número determinado de veces, indefinidamente o mientras se cumpla una condición.



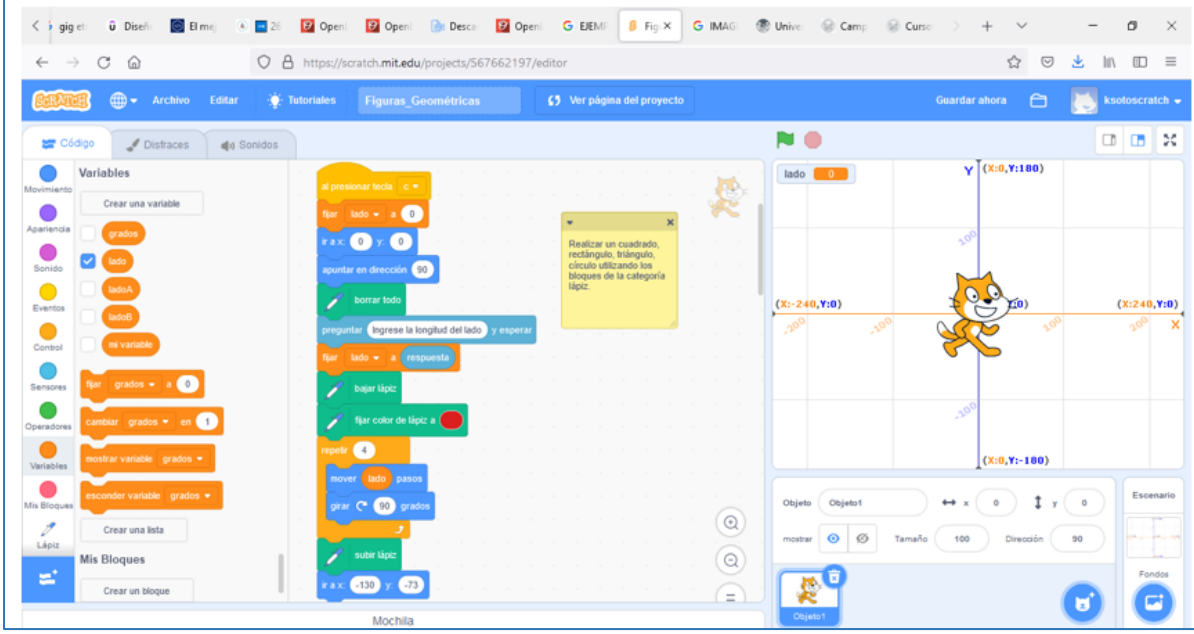
Sesión 6

Extensión: Lápiz.



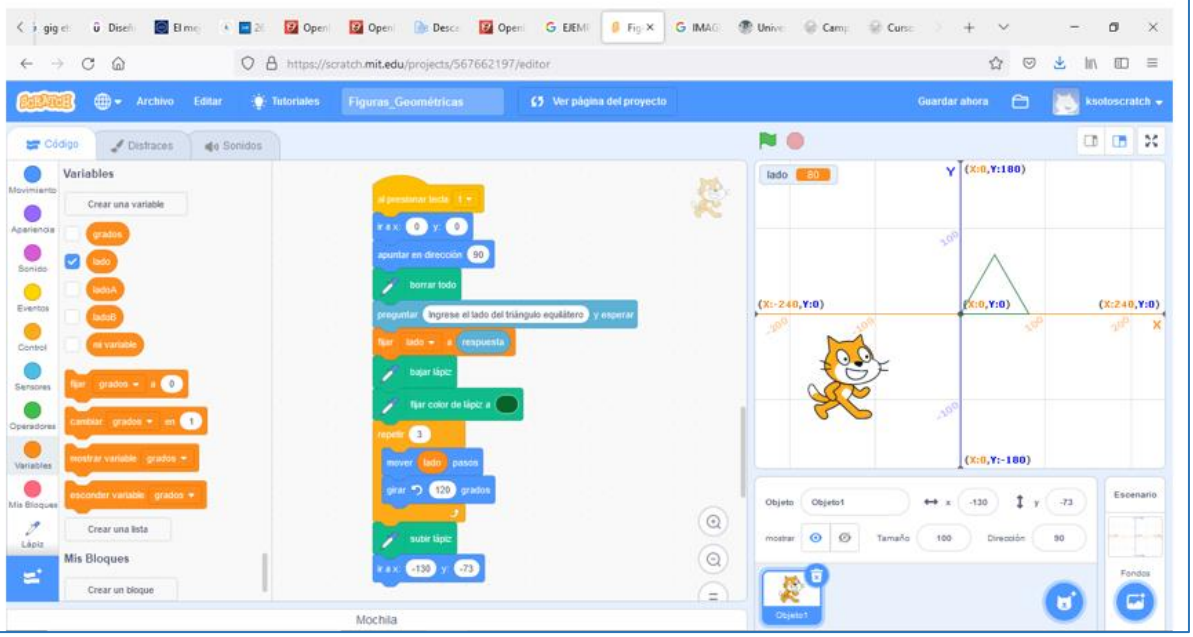
Ejercicio11:

Dibujar un cuadrado, triángulo, rectángulo y círculo, utilizando la Herramienta de Extensión: **Lápiz.**



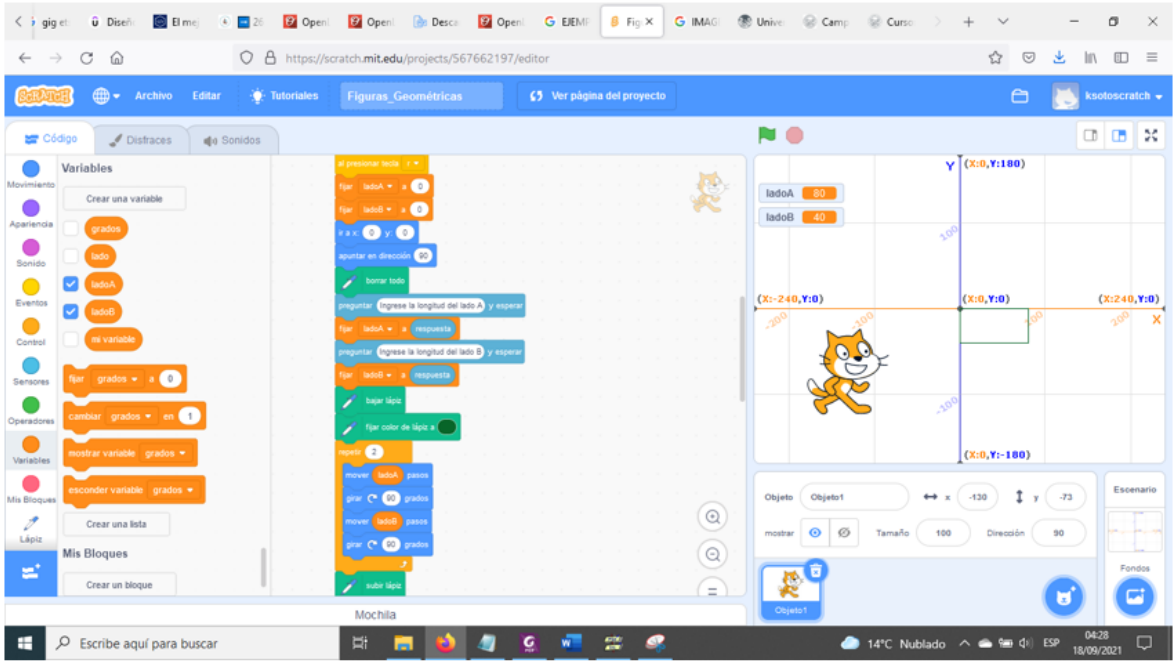
Ejercicio11:

Dibujar un cuadrado, triángulo, rectángulo y círculo, utilizando la Herramienta de Extensión: **Lápiz.**



Ejercicio11:

Dibujar un cuadrado, triángulo, rectángulo y círculo, utilizando la Herramienta de Extensión: **Lápiz.**



Estructuras Repetitivas Anidadas

Es cuando se utiliza una estructura repetiva interna dentro de otra repetitiva externa, según el grado de complejidad.

Contadores y Acumuladores

En las Estructuras repetitivas se utilizan los contadores o acumuladores.

CONTADOR: Es una variable, que almacena el valor de ella misma más un valor constante, es muy útil para controlar el número de veces que debe ejecutarse un grupo de instrucciones.

En Scratch se utiliza esta instrucción como contador.

contador=contador +1

Es decir, se incrementará de uno en uno, o también se puede incrementar de 2 en 2, es un valor constante.

ACUMULADOR: Es una variable, a la cual le vamos a incrementar su valor de forma variable dentro de un ciclo, para ello, la variable debe tener un valor inicial antes de entrar al ciclo y debemos incrementar o decrementar su valor dentro del ciclo.

acumulador= acumulador + Variable

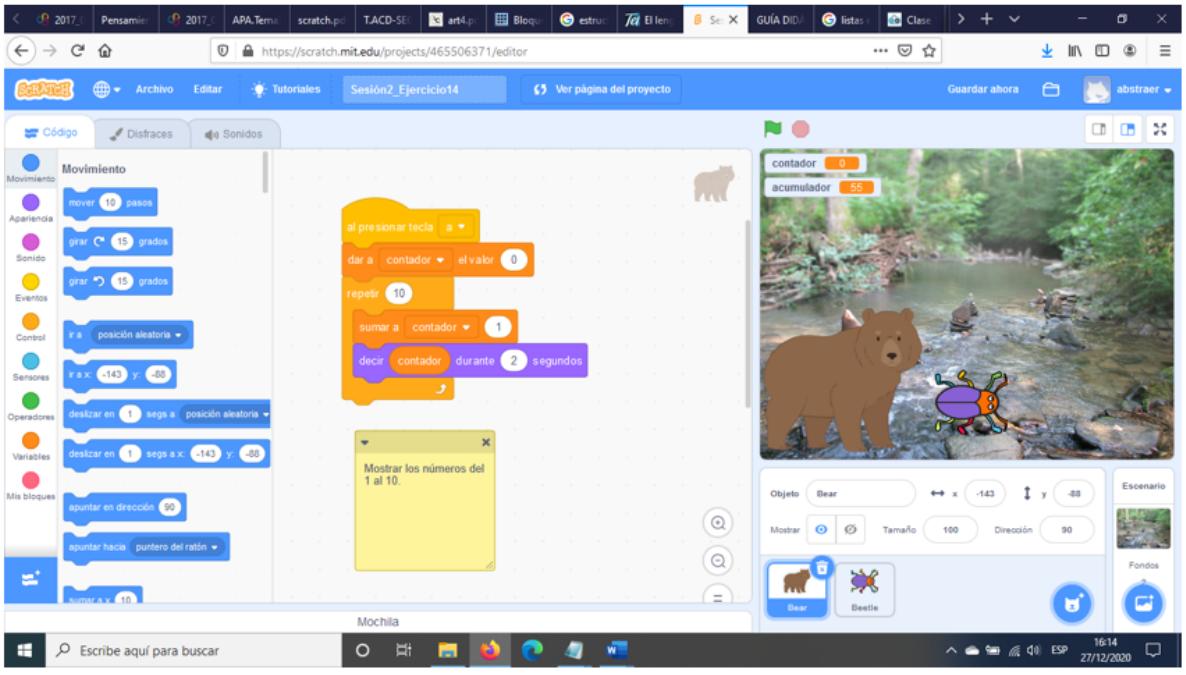
Variable: Es un valor variable, Ejemplo: Cuando me piden sumar las edades de los estudiantes de un salón de clases.

Ejercicio14:

a) Realice un programa que muestre los números del 1 al 10.
b) Realice un programa que muestre los números pares comprendidos entre 1 y 10
c) Realice un programa que muestre los números impares comprendidos entre 1 y 10

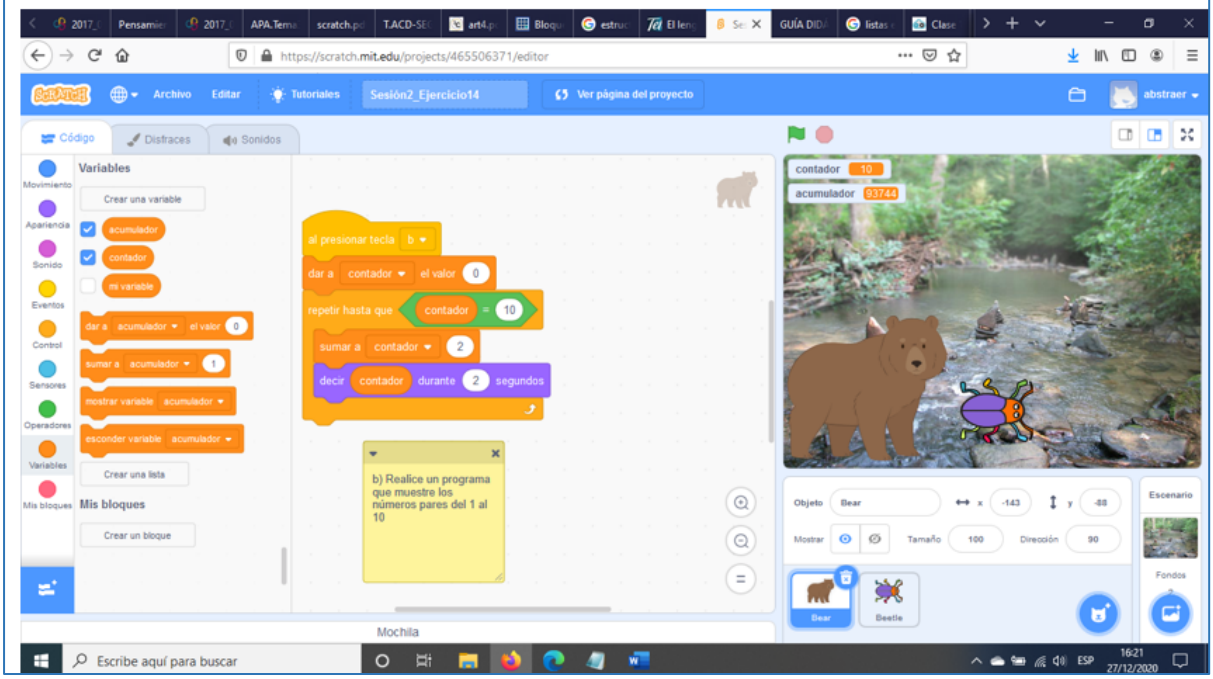
Solución de la Pregunta 14 (a)

a) Realice un programa, que muestre los números del 1 al 10.



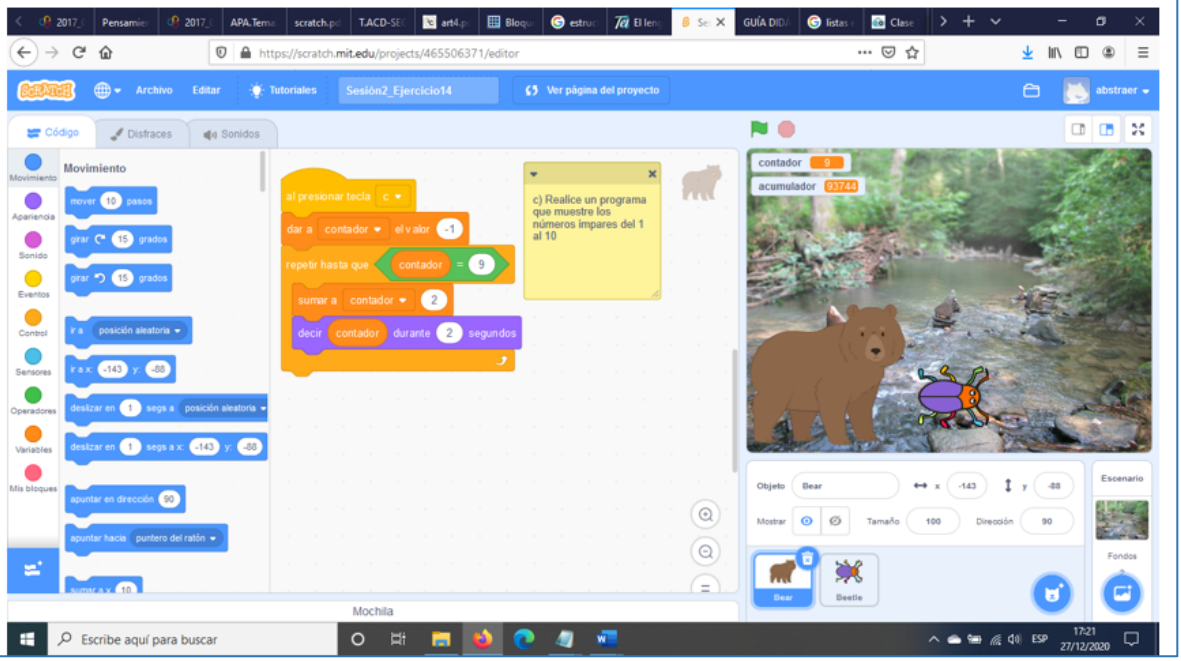
Solución de la Pregunta 14(b)

b) Realice un programa que muestre los números pares del 1 al 10



Solución de la Pregunta 14(c)

c) Realice un programa que muestre los números impares del 1 al 10

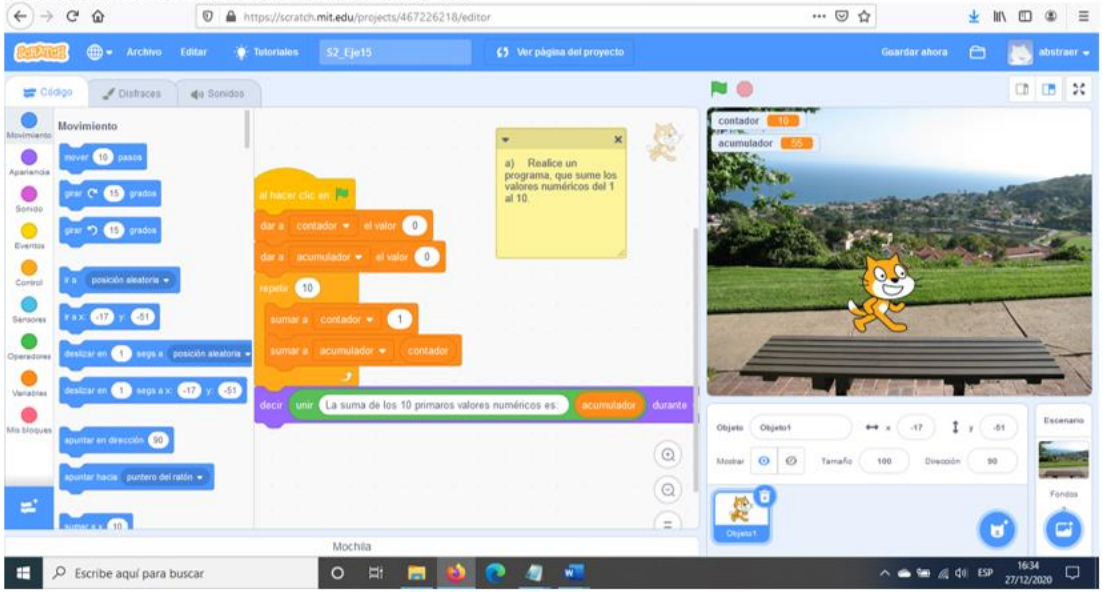


Ejercicio15:

a) Realice un programa, que sume los valores numéricos del 1 al 10.
b) Realice un programa, que sume los valores numéricos pares del 1 al 10.
c) Realice un programa, que sume los valores numéricos impares del 1 al 10.
d) Realice un programa, que sume los valores numéricos del 1 al 100.

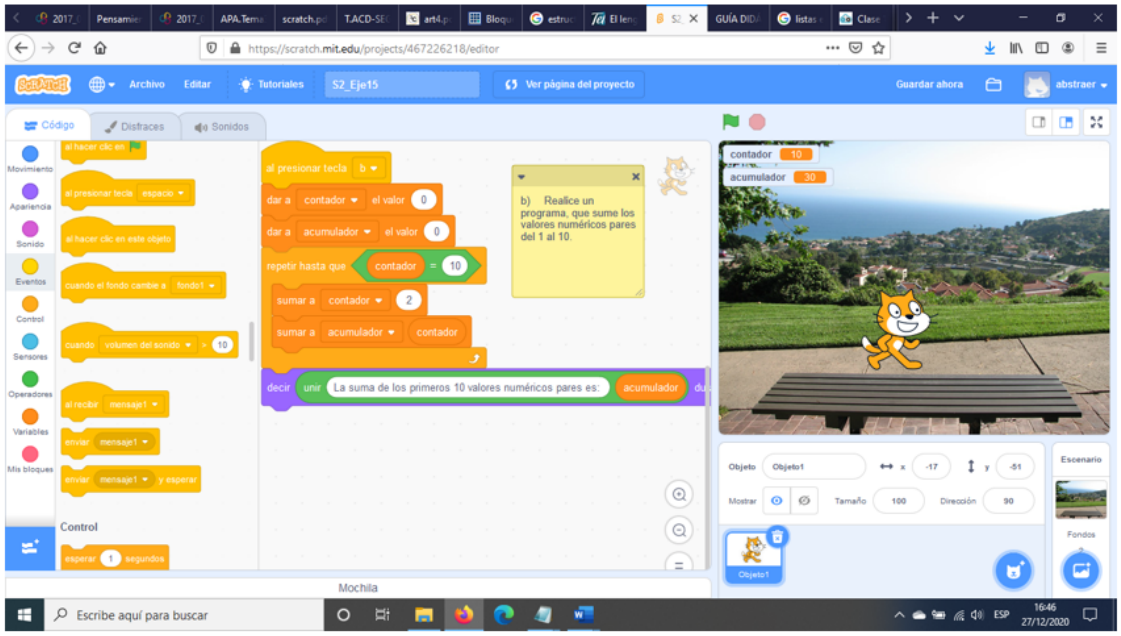
Solución de la Pregunta 15 (a)

a) Realice un programa, que sume los valores numéricos del 1 al 10.



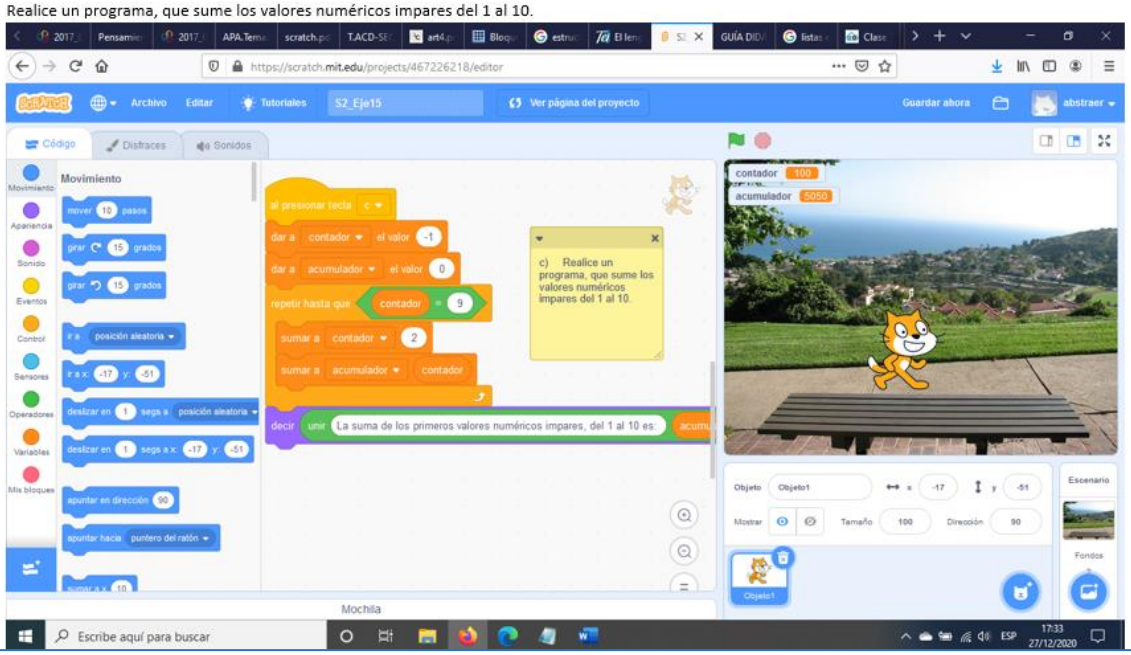
Solución de la Pregunta 15 (b)

b) Realice un programa, que sume los valores numéricos pares del 1 al 10.



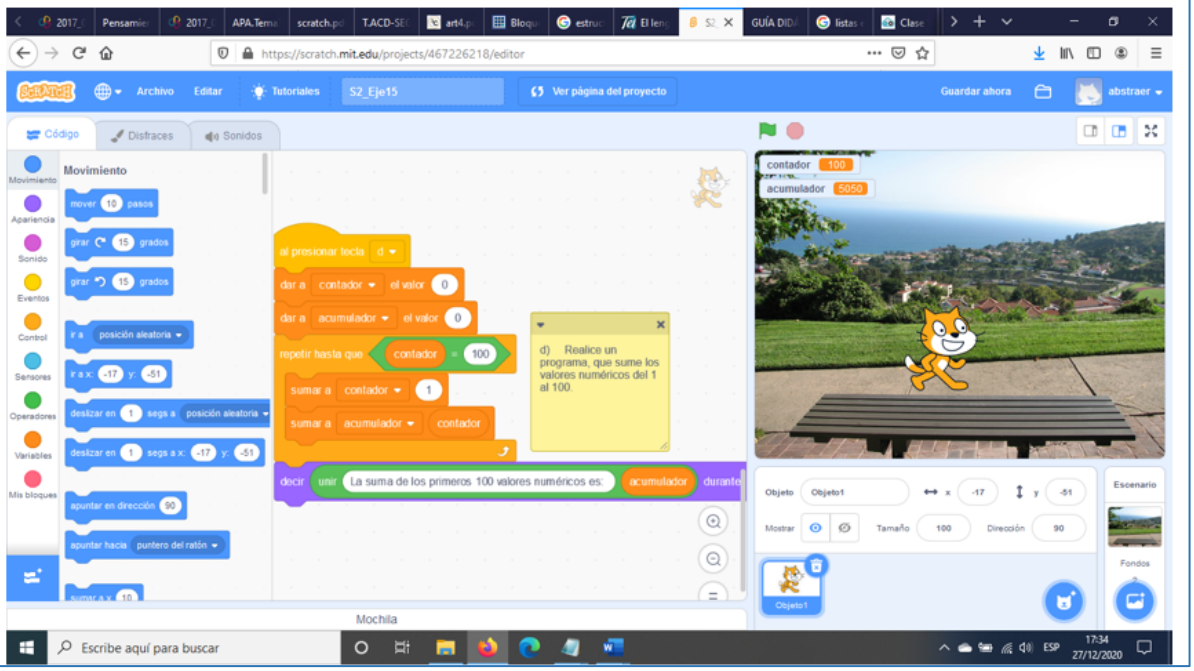
Solución de la Pregunta 15(c)

c) Realice un programa, que sume los valores numéricos impares del 1 al 10.



Solución de la Pregunta 15(d)

d) Realice un programa, que sume los valores numéricos del 1 al 100.



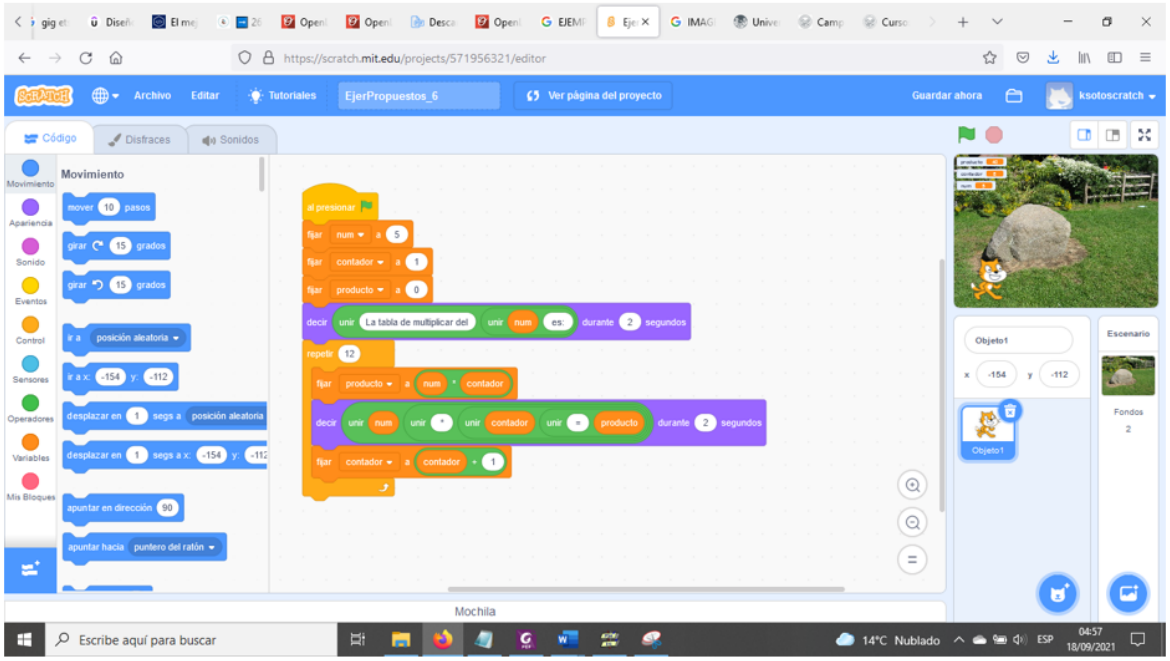
Ejercicio16:

a) Realice un programa para la profesora Karol, para registrar las notas de **n estudiantes** y luego promediarlas, siendo las evaluaciones: eva_continua, parcial1, parcial2, parcial3 y exa_final. El programa debe solicitar el ingreso de la cantidad de estudiantes.

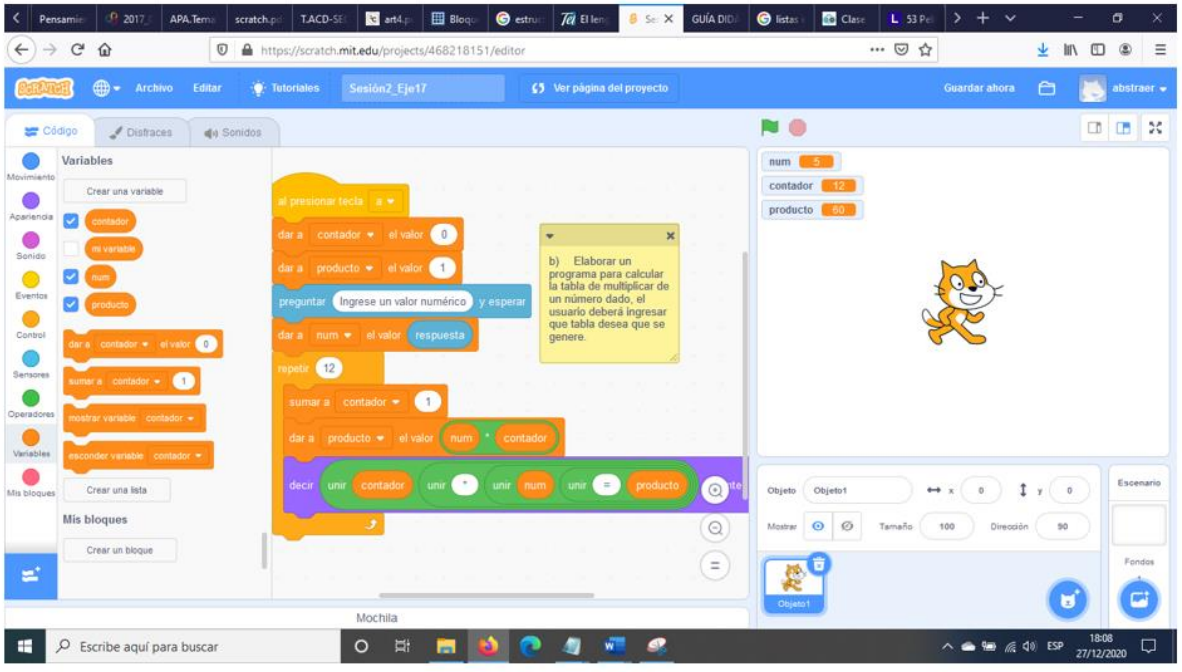
Sesión 7

Ejercicio17:

a) Elaborar un programa para calcular la tabla de multiplicar del **número 5.**



b) Elaborar un programa para calcular la tabla de multiplicar de **un número dado**, el usuario deberá ingresar que tabla desea que se genere.



Ejercicio18:

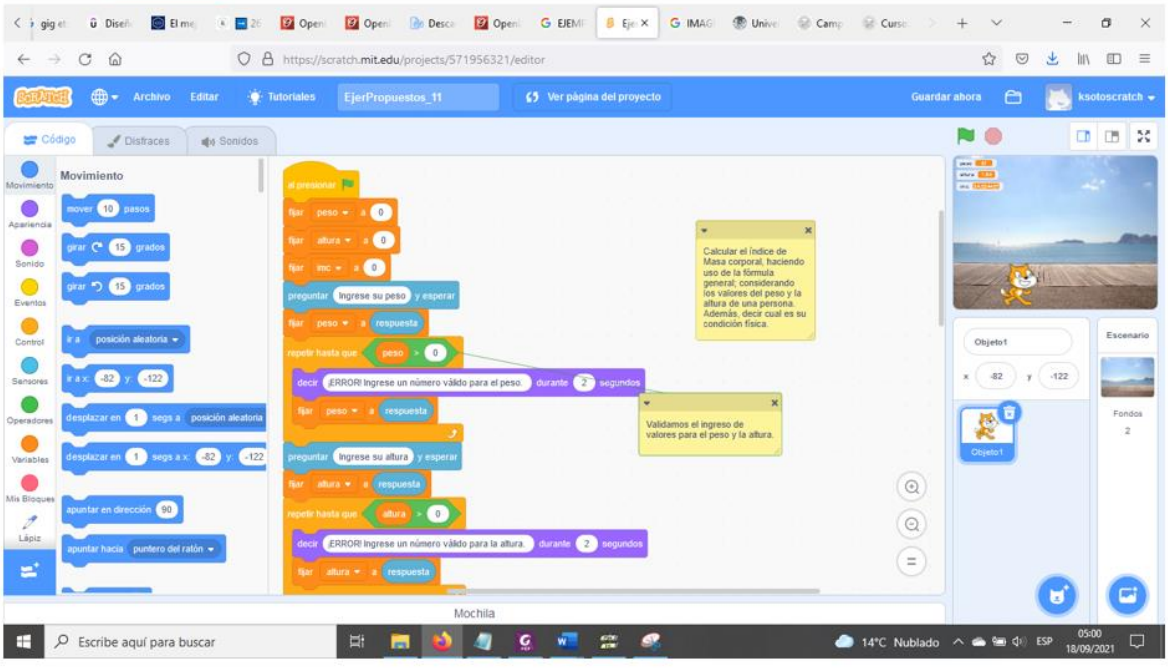
Realizar un ejercicio, que calcule el índice de Masa Corporal **(IMC), según la fórmula**, ingresando el peso y la altura, al final debe mostrarle el estado en el que se encuentra.

Índice de Masa Corporal

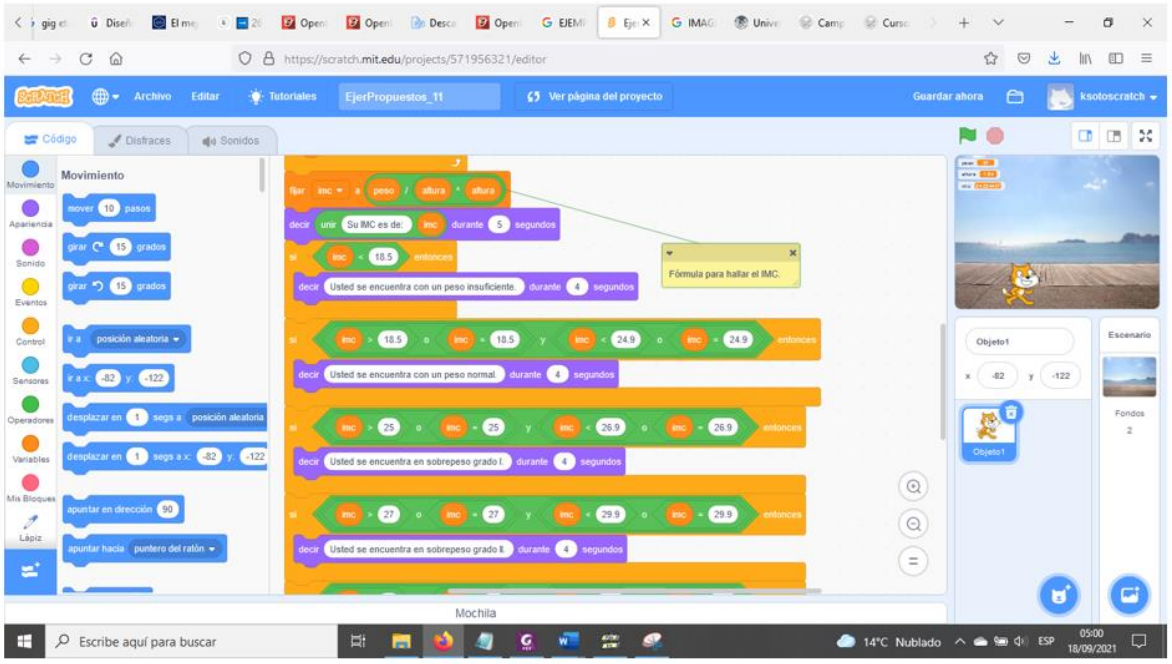
$$IMC = \frac{\text{Peso (Kg)}}{\text{Altura (m)}^2}$$

Peso Insuficiente	< 18,5
Peso Normal	18,5 – 24,9
Sobrepeso Grado I Sobrepeso Grado II	25 – 26,9 27 – 29,9
Obesidad I Obesidad II Obesidad III (mórbida) Obesidad IV (extrema)	30 – 34,9 35 – 39,9 40 – 49,9 > 50,00

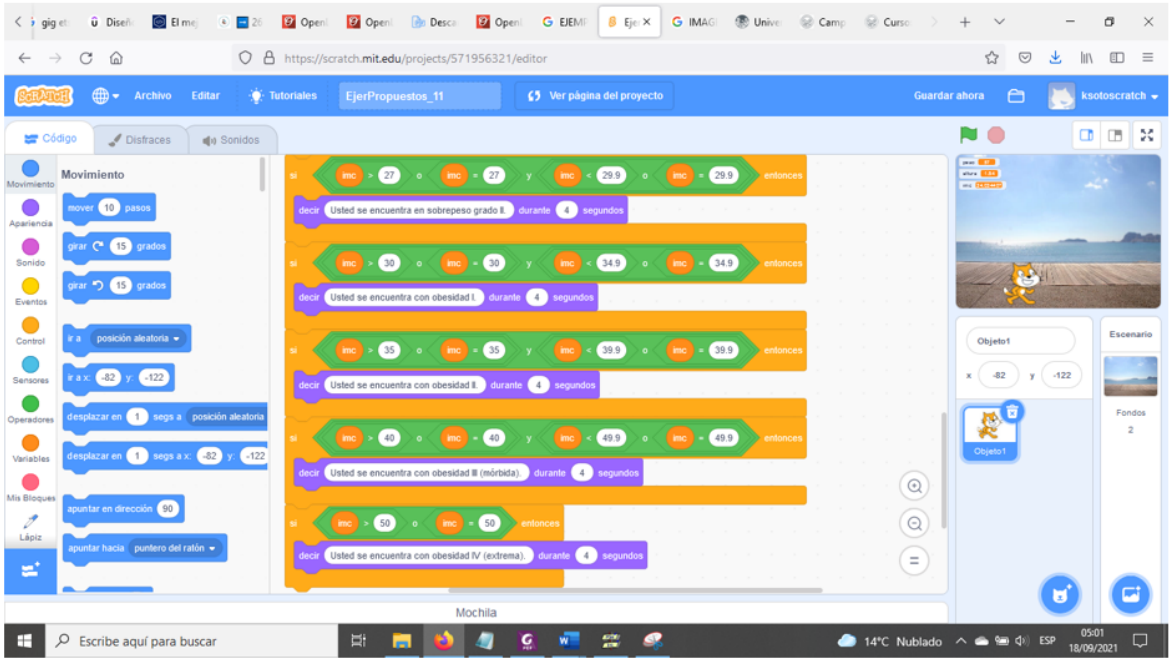
Solución de la Ejercicio18:



Solución de la Ejercicio18:



Solución de la Ejercicio18:



Ejercicios Propuestos de Scratch

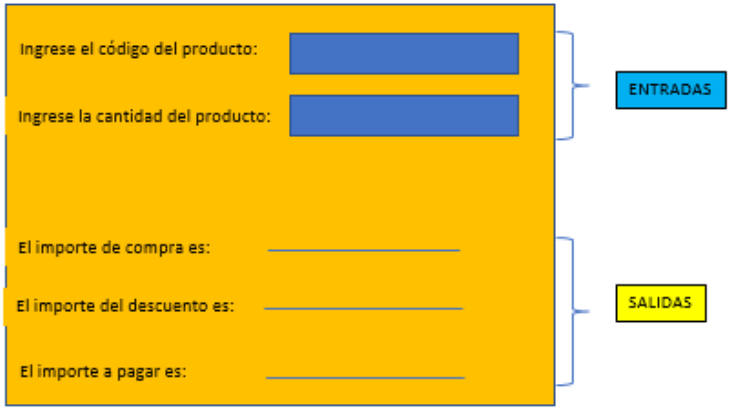
1. Una tienda vende tres tipos de productos cuyos códigos son 101, 102 y 103 a los precios unitarios dados en la siguiente tabla:

Código	Precio unitario
101	S/.21.5
102	S/.30.0
103	S/.15.5

Como oferta, la tienda ofrece un porcentaje de descuento sobre el importe de la compra de acuerdo a la siguiente tabla:

Importe Compra	Descuento
≥ 700	16%
≥ 500 pero < 700	14%
≥ 200 pero < 500	12%
< 200	10%

Diseñe un algoritmo que determine el importe de la compra, el importe del descuento y el importe a pagar por la compra de cierta cantidad de unidades de un mismo tipo de producto.



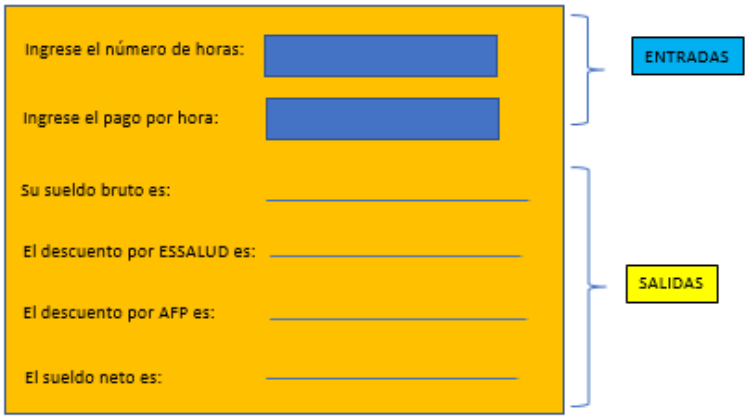
2. Una empresa de préstamos tiene el siguiente esquema de cobros:

Monto del préstamo (S/.)	Número de cuotas
Hasta 5000	2
Más de 5000 hasta 10000	4
Más de 10000 hasta 15000	6
Más de 15000	10

Si el monto del préstamo es mayor a S/. 10000, la empresa cobra 3% de interés mensual; en caso contrario, cobra 5% de interés mensual.

Dado el monto del préstamo de un cliente, diseñe un programa que determine el monto de la cuota mensual y el monto del interés total entre todas las cuotas.

3. Diseñe un algoritmo que calcule el sueldo bruto, el descuento por ESSALUD, el descuento por AFP y el sueldo neto del empleado de una empresa de acuerdo a los siguientes criterios: el sueldo bruto se calcula multiplicando el número de horas trabajadas por una tarifa horaria, el descuento por ESSALUD es igual al 9% del sueldo bruto, el descuento por AFP es igual al 12.5% del sueldo bruto, el sueldo neto es la diferencia entre el sueldo bruto y el descuento total.



4. Una empresa calcula el sueldo bruto de sus 4 trabajadores multiplicando las horas trabajadas por una tarifa horaria que depende de la categoría del trabajador de acuerdo a la siguiente tabla:

Categoría	Tarifa
A	S/.21.0
B	S/.19.5
C	S/.17.0
D	S/.15.5

Por ley, todo trabajador se somete a un porcentaje de descuento del sueldo bruto:
20% si el sueldo bruto es mayor que S/. 2500 y 15% en caso contrario.
Diseñe un programa que determine el sueldo bruto, el descuento y el sueldo neto que le corresponden a un trabajador de la empresa.

5. Diseñe un programa que genere aleatoriamente las notas de una práctica calificada para n estudiantes de una sección y determine la nota promedio de la sección y las notas máxima y mínima generadas.

6. Elaborar un programa para calcular la tabla de multiplicar del **número 5.**

7. Un padre ha decidido dar una propina a su hijo en base a sus notas en los cursos de Matemáticas, Física e Historia del Perú.
   - Si la nota de Matemática es mayor a 17, le dará S/. 3 de propina por cada punto; en caso contrario, sólo le dará S/. 1.0 por cada punto.
   - Si la nota de Física es mayor a 15, le dará S/. 2.0 de propina por cada punto; en caso contrario, sólo le dará S/.0.5 por cada punto.
   - Si la nota de Historia del Perú es mayor a 15, le dará S/. 1.5 por cada punto; en caso contrario, sólo le dará S/. 0.30 por cada punto.
   - Además, si la nota de Matemática es mayor a 17, le obsequiará un reloj; en caso contrario, le obsequiará un lapicero.

   Diseñe un programa que determine el monto total de la propina y el obsequio que le corresponde al hijo.

8. Realice un programa, que al inicio solicite números diferentes al 6, y si ingreso el número 6, el programa debe detenerse y decirme **¡Haz acertado!**

9. Una persona tiene registrado en un papel las compras que efectuó sobre los productos A, B, C y D a lo largo de todo un año. Por cada registro, tiene anotado el tipo del producto adquirido, el precio unitario del producto y la cantidad de unidades adquiridas. Diseñe un programa que permita ingresar de uno en uno, los registros con que cuenta la persona y muestre luego:
   a) El número de registros de cada tipo de producto.
   b) La cantidad total de unidades adquiridas de cada tipo de producto.
   c) El precio unitario promedio de cada tipo de producto.

10. Un hotel turístico tiene cinco tipos de habitaciones, cuyos costos por día se dan en la tabla adjunta. Como oferta, el hotel ofrece un descuento del 15% del importe bruto para más de 5 días de hospedaje y 10% en caso contrario. Diseñe un programa que determine el importe bruto, el importe del descuento y el importe a pagar por parte de un cliente. Además, el programa debe mostrar información actualizada sobre el número de clientes por tipo de habitación y el importe total pagado para los clientes ingresados hasta el momento.

Categoría	Costo por día
A	S/. 90
B	S/. 80
C	S/. 70
D	S/. 60
E	S/. 20

11. Realizar un ejercicio, que calcule el índice de Masa Corporal **(IMC), según la fórmula**, ingresando el peso y la altura, al final debe mostrarle el estado en el que se encuentra. Por ejemplo: "Peso normal" o "Obesidad I", etc.

Índice de Masa Corporal

$$IMC = \frac{\text{Peso (Kg)}}{\text{Altura (m)}^2}$$

Peso Insuficiente	< 18,5
Peso Normal	18,5 – 24,9
Sobrepeso Grado I Sobrepeso Grado II	25 – 26,9 27 – 29,9
Obesidad I Obesidad II Obesidad III (mórbida) Obesidad IV (extrema)	30 – 34,9 35 – 39,9 40 – 49,9 > 50,00

12. Realizar un ejercicio que calcule el promedio final del curso de Algorítmica 1 de un estudiante, ingresar las siguientes evaluaciones:
   - **Las notas son las siguientes:** Promedio de Evaluaciones continuas, EP1, EP2, EP3 y EF, luego calcule el promedio utilizando los pesos de la tabla según corresponda,

Prom_Evac	EP1	EP2	EP3	EF
10%	20%	20%	20%	30%

   - Indicarle si está Aprobado o Desaprobado.
   - Validar cada una de las notas ingresadas.

13. **Crear un juego de 5 preguntas y respuestas con puntuación**, cuando el usuario responda correctamente la pregunta, debe mostrarse un objeto y escenario feliz, caso contrario cuando pierda debe mostrarse otro objeto y escenario con la respuesta correcta, la puntuación debe acumularse por cada pregunta acertada, al finalizar el cuestionario debe mostrarse la puntuación final.

14. Dibujar polígonos, indicándole la cantidad de lados, dato que deberá ser ingresado por el usuario.

15. Una tienda vende tres tipos de productos a los precios unitarios dados en la siguiente tabla:

Producto	Precio
Colores	S/. 15.0
Plumones	S/. 17.5
Acuarelas	S/. 20.0

Como oferta la tienda ofrece un regalo de acuerdo a la siguiente tabla:

Unidades adquiridas	Regalo
1 a 25	un lapicero
26 a 50	un cuaderno
Más de 50	una agenda

Diseñe un programa que determine el importe a pagar y el regalo para un cliente de la tienda, además deberá preguntarle si desea continuar con la compra de otro producto.

16. Un vivero municipal vende plantones a los precios indicados en la siguiente tabla:

Plantones	Costo por unidad
Forestal	S/. 0.35
Forestal ornamental	S/. 0.50
Frutal	S/. 2.00
Frutal injertado	S/. 3.00

Por otro lado, si el cliente adquiere más de 10 docenas de plantones de cualquier tipo, el vivero le obsequia 3 plantones por cada docena adquirida; en caso contrario, sólo le obsequia 1 plantón por cada docena.

Diseñe un programa que determine el importe a pagar y el número de plantones de obsequio por la compra de cierta cantidad de plantones de un mismo tipo.

17. Una compañía de seguros ofrece a sus clientes cuatro tipos de seguro de sepelio:

Tipo	Máximo número de Personas	Pago mensual (S/.)
A	8	40
B	6	30
C	4	20
D	2	10

Si el cliente asegura a más personas de la indicadas en el cuadro anterior tendrá que pagar S/.8.00 mensuales por cada persona adicional si es que el seguro es de tipo A o B, y S/.5.00 mensuales por cada persona adicional si es que el seguro es de tipo C o D. Calcular el monto anual que tiene que pagar un determinado cliente.

18. Una empresa química paga a sus vendedores un sueldo bruto que es igual a la suma de un sueldo básico quincenal de S/.250 más una comisión igual a un porcentaje del total de las ventas efectuadas de acuerdo a la siguiente tabla:

Monto vendido	Comisión
≥ 20000	16%
≥ 15000 pero < 20000	14%
≥ 10000 pero < 15000	12%
< 10000	10%

Por otro lado, si el sueldo bruto del vendedor supera los S/.1800, este se somete a un descuento del 11%. Diseñe un programa que determine el sueldo bruto, el descuento y el sueldo neto de un vendedor.

19. Un club social a clasificado a sus socios en 3 categorías, como se muestra en el siguiente cuadro:

Categoría	Cant. Boletos	Pago Mensual (S/.)	Dscto (%)
A	25	200	4
B	20	150	3
C	15	100	2

Dicho club realiza mensualmente un tipo de evento (rifas, almuerzos, etc.)., de esta manera, un socio está obligado a vender la cantidad de boletos que se indican la tabla, pero si un socio vende más de los boletos indicados, se le descuenta S/.2.00 por cada boleto extra vendido. Además, si el socio tiene más de 55 años recibe un porcentaje de descuento de su pago mensual. Calcular el monto total que paga un socio en un mes.

20. En una empresa cada empleado tiene un código entero de tres cifras. Diseñe un programa que lea el código de un empleado y determine de qué tipo de empleado se trata de acuerdo a los siguientes criterios:

- ✓ Si el código es divisible por 2, por 3 y por 5, el tipo de empleado es "Administrativo".
- ✓ Si el código es divisible por 3 y por 5 pero no por 2, el tipo de empleado es "Directivo".
- ✓ Si el código es divisible por 2, pero no por 3 ni por 5, el tipo de empleado es "Vendedor".
- ✓ Si el código no es divisible por 2, ni por 3 ni por 5, el tipo de empleado es "Seguridad".

21. Realizar un programa que lea los datos de "n" triángulos e informar:

a. De cada uno de ellos, que tipo de triángulo es: equilátero, isósceles o escaleno.
b. Cantidad de triángulos de cada tipo.
c. Tipo de triángulo que posee menor cantidad.

22. Dada una cantidad de dinero en soles, diseñe un programa que descomponga dicha cantidad en billetes de S/.100, S/.50, S/.20, S/.10 y monedas de S/.5, S/.2 y S/.1. Así, por ejemplo, S/. 3778 puede descomponerse en 37 billetes de S/. 100, más 1 billete de S/. 50, más 1 billete de S/. 20, más 1 moneda de S/. 5, más 1 moneda de S/.2 y más 1 moneda de S/. 1.

23. Diseñe un algoritmo que determine la cantidad de divisores de un número natural.

24. Determine si una letra ingresada es un caracter o una vocal.

25. Una empresa comercializa 3 tipos de productos A, B y C. Validar el tipo de producto.

Tipo	Precio Unitario S/.	Cantidad	Descuento
A	2.5	<=30	7%
		>30	15%
B	4.7	<20	5%
		[20-100]	25%
		>100	35%
C	16.4	-	2%

Solicitar el nombre del cliente, el tipo de producto y la cantidad de unidades a comprar para n clientes.
El programa mostrará el nombre del cliente, monto antes del descuento, el descuento otorgado y el monto a pagar del cliente con más unidades adquiridas.

UNIVERSIDAD CATOLICA SEDES SAPIENTIAE

EXAMEN DE SCRATCH

Nombre Estudiante		
Docente:	Karol Soto Ccoicca	
Indicaciones	El programa se trabaja en Scratch.	

Nota: En este mismo documento de Word, realizar captura de pantalla de TODA la codificación de Scratch y de las ventanas de ejecución con los datos ingresados.

Fila A

1. Realizar un cuestionario de 10 preguntas de su especialidad con alternativas para que el usuario indique la respuesta, si en caso sea correcta la respuesta, debe de acumular 2 puntos por cada pregunta, si es incorrecto no debe acumular puntaje. El programa deberá decirle la respuesta correcta por cada respuesta incorrecta. Al final debe de mostrarle el puntaje total obtenido. Utilizar disfraces, escenarios, imágenes, sonidos. Se recomienda utilizar listas

2. Diseñe un programa que permita ingresar de una en una las edades de un conjunto de personas y muestre luego de cada ingreso:

   a) El número de personas mayores de edad. (1 pto)
   b) El promedio de edad de personas mayores de edad. (2 ptos)
   c) El número de personas menores de edad. (1 ptos)
   d) El promedio de edad de personas menores de edad (2 ptos)
   e) La mayor edad ingresada. (1 pto)
   f) La menor edad ingresada. (1 pto)

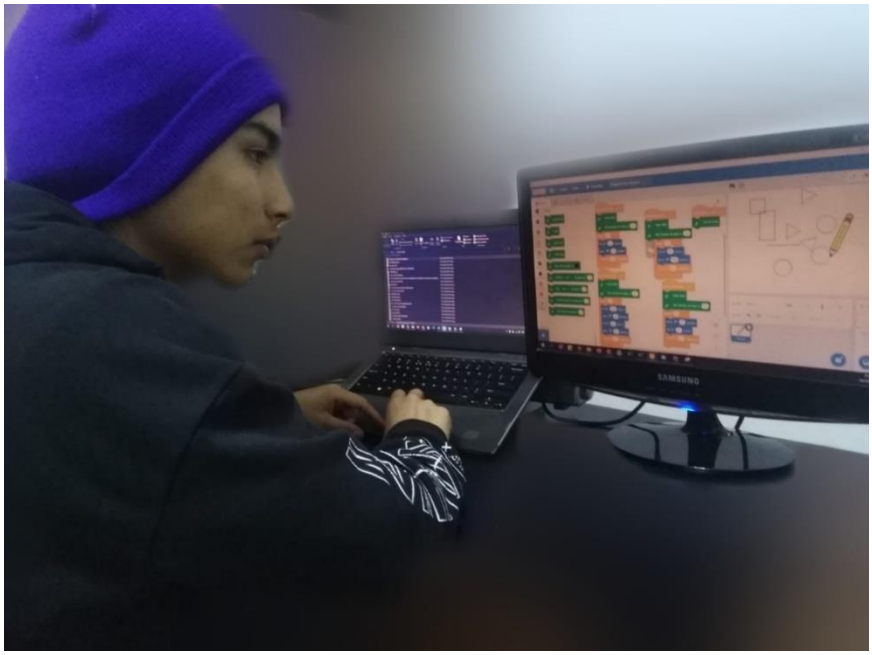
   Nota: Los datos de ingreso deben ser validados. (1 pto)

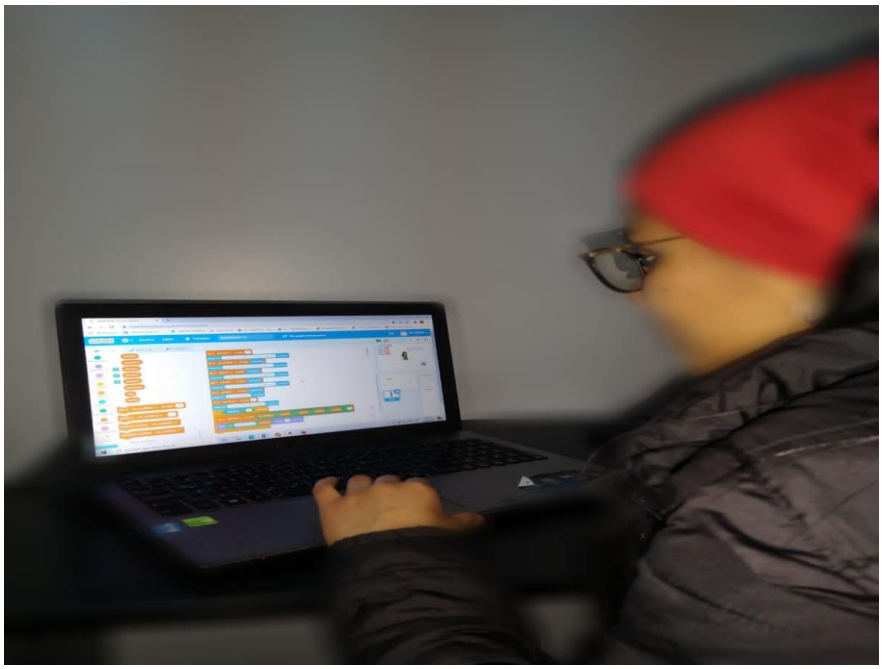
Identificar la(s) entrada(s), proceso y salida(s) (1 pto)

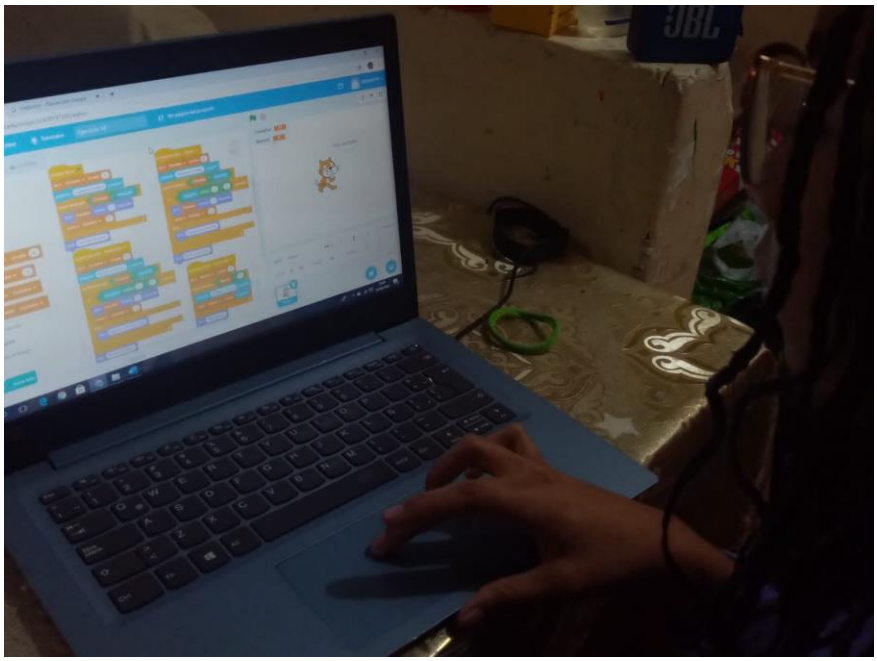
Entrada	Proceso	Salida
•		

	CRITERIOS		
Preguntas	Muy bien	En proceso	En inicio
Pregunta 1	• Contiene las 10 preguntas. • Acumula puntaje por cada respuesta correcta. • Por cada respuesta incorrecta muestra la respuesta correcta. • Al finalizar el cuestionario muestra el puntaje total. • Utiliza disfraces, escenarios, imágenes, sonidos. Max 10 puntos	• Contiene 5 preguntas. • Acumula puntaje por cada respuesta correcta. • Por cada respuesta incorrecta no muestra la respuesta correcta. • Al finalizar el cuestionario muestra el puntaje total. • Utiliza algunos disfraces, escenarios, imágenes, sonidos. Max 5 puntos	• Contiene menos de 5 preguntas. • No acumula puntaje por cada respuesta correcta. • Por cada respuesta incorrecta no muestra la respuesta correcta. • Al finalizar el cuestionario no muestra el puntaje total. • Utiliza algunos disfraces, escenarios, imágenes, sonidos. Max 2 puntos
Pregunta 2	• Identificar las entradas, procesos y salida. • Se declaran las variables de forma nemotécnicas. • Utilizan las estructuras de control necesarias. • Se validan las entradas. • Las salidas durante la ejecución son correctas. • Max 10 puntos	• Se declaran las variables nemotécnicas de forma parcial. • Utilizan las estructuras de control parcialmente. • Se validan las entradas parcialmente. • Las salidas durante la ejecución tienen algunos errores. • Max 5 puntos	• Se declaran sólo algunas variables de forma nemotécnicas. • Utilizan algunas estructuras de control necesarias. • No se validan las entradas. • Las salidas durante la ejecución no son las correctas. • Max 2 puntos

6.7. Anexo 7: Evidencias






JBL