CIENCIAS DE LA COMPUTACIÓN PARA EL AULA

Manual para docentes

1º CICLO SECUNDARIA

fundación SADOSKY
Investigación y Desarrollo en TIC

<Program.AR/>


fundación
ADOSKY
Investigación y Desarrollo en TIC

MANUAL 1º CICLO DE SECUNDARIA

AUTORES
Pablo E. “Fidel” Martínez López
Federico Aloi
Daniel A. Ciolek
Federico Martínez
Denise Pari
Pablo Tobia

COORDINADOR DEL EQUIPO AUTORAL
Pablo E. “Fidel” Martínez López

COORDINADORA PEDAGÓGICA
Denise Pari

REVISORES DE CONTENIDOS
Julián Dabbah
Herman Schinca
Daniela Villani

EDITORES
Ignacio Miller
Alejandro Palermo

CORRECTORA
Luz Rodríguez

DISEÑADORES GRÁFICOS
Luciano Andújar
Jaqueline Schaab
Juan Martín Serrovalle

ILUSTRADOR
Tony Ganem

COLECCIÓN CIENCIAS DE LA COMPUTACIÓN PARA EL AULA

EDITORES GENERALES
Hernán Czemerinski
Vanina Klinkovich

SUPERVISOR DISCIPLINAR
Franco Frizzo

FUNDACIÓN DR. MANUEL SADOSKY

COORDINADORES DE LA INICIATIVA PROGRAM.AR
María Belén Bonello
Fernando Schapachnik

DIRECTOR EJECUTIVO
Esteban Feuerstein

PRESIDENTE
Secretario de Gobierno de Ciencia, Tecnología e Innovación Productiva de la Nación, José Lino Barañao

Ciencias de la computación para el aula : 1er. ciclo de secundaria / Pablo Ernesto Martínez López ... [et al.] ; compilado por Pablo Ernesto Martínez López ... [et al.] ; coordinación general de Vanina Klinkovich ; Hernán Czemerinski ; editado por Ignacio David Miller ; Alejandro Palermo ; editor literario Luz Luz María Rodríguez ; ilustrado por Luciano Andújar ... [et al.] ; prólogo de María Belén Bonello ; Fernando Pablo Schapachnik. - 1a ed . - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2019.
Libro digital, PDF - (Ciencias de la Computación para el aula / Klinkovich, Vanina; Czemerinski, Hernán; 3)

Archivo Digital: descarga
ISBN 978-987-27416-7-9

1. Informática. 2. Programación. I. Martínez López, Pablo Ernesto II. Martínez López, Pablo Ernesto, comp. III. Klinkovich, Vanina, coord. IV. Czemerinski, Hernán, coord. V. Miller, Ignacio David, ed. VI. Palermo, Alejandro, ed. VII. Luz María Rodríguez, Luz, ed. Lit. VIII. Andújar, Luciano, ilus. IX. Bonello, María Belén, prolog. X. Schapachnik, Fernando Pablo, prolog.
CDD 004.0712

DISTRIBUCIÓN LIBRE Y GRATUITA

<Program.AR/>
fundación SADOSKY Investigación y Desarrollo en TIC
Ministerio de Ciencia, Tecnología e Innovación Productiva Presidencia de la Nación
CAF BANCO DE DESARROLLO DE AMÉRICA LATINA
Universidad Nacional de Quilmes

CIENCIAS DE LA COMPUTACIÓN PARA EL AULA

Manual para docentes

1º CICLO SECUNDARIA

ÍNDICE

PRÓLOGO

La tarea de prologar estos manuales se asemeja, para nosotros, a la de colocar el cartel que dice "bienvenidos" en la puerta de un edificio de varios pisos. Este gesto, que es final e inaugural a la vez, corona años (literalmente hablando) de duro trabajo y, a la vez, anticipa la espera ansiosa de la etapa siguiente: su uso en las aulas.

Se trata de los **primeros manuales escolares sobre Ciencias de la Computación en el escenario editorial argentino, que se ponen a disposición del público de manera libre y gratuita**, y que se suman a un pequeño grupo de ejemplos pioneros a nivel mundial en esta temática. Somos conscientes de que, al hablar de "manuales escolares sobre Ciencias de la Computación", queda mucho por aclarar. Comencemos por el principio: ¿por qué Ciencias de la Computación?

Ciencias de la Computación es el nombre que recibe el área del conocimiento que aporta una serie de saberes (programación, funcionamiento de las computadoras e Internet, inteligencia artificial, etc.) **que resultan fundamentales para comprender el mundo cada vez más tecnológico en el que viven y se desarrollan los alumnos que transitan su escolaridad hoy en día**. Sin estos conocimientos, su comprensión de la realidad se verá limitada, y no podrán participar como ciudadanos activos e informados en los debates actuales sobre las múltiples interacciones entre la tecnología informática y la sociedad. Argentina ha decidido avanzar sobre esta materia y es por eso que **el Consejo Federal de Educación declaró, mediante su resolución 263/15, que la enseñanza y el aprendizaje de programación es de importancia estratégica para fortalecer el desarrollo socioeconómico de la nación**.

Estos manuales **se concibieron para el aula**, como una herramienta para el docente, al que le brindan secuencias didácticas detalladas y fichas de trabajo para entregar a sus estudiantes. ¿Para qué aula, para qué docentes, para qué estudiantes? En principio, estas actividades están pensadas para el aula argentina. Este material fue **escrito en su totalidad por y para argentinos y argentinas**, tomando como referencia la realidad de la escuela argentina. Esto se refleja en el lenguaje, en las referencias y en los marcos culturales que se utilizan, características que no impiden que incentivemos su uso en otros países de la región y del mundo.

Los cuatro manuales que componen esta colección **cubren, respectivamente, el primer ciclo de la educación primaria, el segundo ciclo de la educación primaria, el primer ciclo de la educación secundaria y el segundo ciclo de la educación secundaria**. El rango etario determina, en cada caso, el recorte de temas, la profundidad con que se abordan, la complejidad del texto y la línea

estética. En su mayoría **son manuales iniciales** (es decir, tres de ellos están concebidos para alumnos y alumnas que dan sus primeros pasos en Informática, a distintas edades). El manual destinado al segundo ciclo de la escolaridad primaria tiene como antecedente nuestro primer manual para programar en el aula.[1]

En cuanto a sus destinatarios principales, los docentes, estos manuales buscan interpelar a un conjunto de profesionales cuyas formaciones en el área son heterogéneas. Es por esto que, sin pretender presentar explicaciones teóricas exhaustivas, a lo largo de los distintos capítulos hay desarrollos conceptuales que contribuyen a que aquellos y aquellas docentes que no poseen un dominio fluido de ciertos temas puedan contar con las nociones fundamentales. Al respecto, vale destacar que la Fundación Sadosky ha capacitado hasta abril de 2018 (a través de convenios con universidades públicas de todo el país) a más de 1500 docentes, que se suman a los que han formado los ministerios de educación nacional y provinciales. Estos docentes encontrarán particular provecho en nuestro material.

Aunque gran parte del material fue testeado y consultado con los y las docentes de primaria y secundaria que tomaron nuestros cursos, somos conscientes de que solo su uso de manera sistemática en la escuela permitirá mejorarlo.

Asimismo, resulta pertinente aclarar que estos manuales **están pensados para abordar contenidos de Ciencias de la Computación en espacios disciplinares específicos**. Distintas jurisdicciones del país (Neuquén, CABA, Tucumán, entre otras) cuentan con estos espacios curriculares, mientras que otras están discutiendo su incorporación. Desde Program.AR entendemos que este material constituye un aporte a ese camino, que se suma a las Planificaciones anuales para Tecnología de la Información de 3er y 4to año de CABA[2] publicadas anteriormente.

En cuanto al enfoque didáctico, las secuencias propuestas están pensadas, en buena parte, desde la **perspectiva del aprendizaje por indagación**. Imaginamos que los manuales serán usados por docentes y estudiantes que transitan un camino de descubrimiento, asociado a la tecnología informática que media en buena parte de nuestras interacciones con el mundo.

[1] *Manual de actividades para Program.AR*, disponible en http://program.ar/manual-docentes-primaria/

[2] Disponibles en http://program.ar/planificacion-anual-ti3/ y http://program.ar/planificacion-anual-ti4/.

Vale la pena ahondar en el proceso que hoy encuentra un hito en la publicación de estos manuales. Al pensar de qué manera debía construirse este material y quiénes debían ser los encargados de hacerlo, recurrimos a la herramienta que más garantías ofrece en términos de calidad, apertura y transparencia: **se realizó una convocatoria pública a las universidades que componen el sistema de generación de conocimiento de nuestro país**. Como resultado de esa convocatoria, y mediante la evaluación de un jurado internacional, resultaron elegidas cuatro universidades nacionales que formaron equipos autorales compuestos por profesionales de las Ciencias de la Computación y de Educación.

Los manuales fueron escritos por colegas de la **Universidad Nacional del Centro de la Provincia de Buenos Aires** (primer ciclo de primaria), la **Universidad Nacional de Córdoba** (segundo ciclo de primaria), la **Universidad Nacional de Quilmes** (primer ciclo de secundaria) y la **Universidad Nacional de La Plata** (segundo ciclo de secundaria), con quienes estamos profundamente agradecidos por su compromiso y profesionalismo en un proceso que fue novedoso para todos los involucrados. Cada manual tiene una impronta propia, a través de la cual se traducen las diferentes miradas, prioridades y valoraciones que los distintos colegas otorgan a diversos aspectos de las Ciencias de la Computación.

Deseamos destacar la labor de todo el **equipo de la Fundación Sadosky** que participó en el desarrollo de estos manuales. Estas palabras apenas resumen un proceso que implicó años de trabajo y, por sobre todas las cosas, el compromiso de todos los que lo hicieron posible. En primer lugar, agradecemos a Hernán Czemerinski y Vanina Klinkovich, editores generales de la colección, quienes realizaron un gran trabajo dirigiendo este proyecto desde su génesis en 2016 y se ocuparon, además, de revisar, corregir, unificar y armonizar todas las visiones.

Sumamos a este agradecimiento a Franco Frizzo, que ha supervisado el contenido de toda la colección, y a Jaqueline Schaab, a cargo del diseño gráfico. Queremos también destacar la participación del conjunto de revisores, desarrolladores y gestores de la Fundación Sadosky que participaron en este proyecto (por orden alfabético): Julián Dabbah, Pablo Factorovich, Mariana Labhart, Alfredo Sanzo, Herman Schinca, Daniela Villani. Sin su colaboración, estos manuales no serían una realidad.

Dedicamos un párrafo especial al querido Alfredo Olivero, a la vez mentor y compañero de ruta, de un humor tan incisivo como su inteligencia, que fue parte de este equipo y a quien extrañamos mucho.

Asimismo, no queremos dejar de agradecer al equipo de Legales, Administración y Gestión de la Fundación Sadosky, compuesto por Roxana Ríos, Andrea Córdoba, Rosa Córdoba, Mariano Tiseyra, Melina Rodríguez y a su Director Ejecutivo, Esteban Feuerstein. Queremos también expresar nuestro agradecimiento a Santiago Ceria, quien ocupaba la Dirección Ejecutiva al momento de comenzar este proyecto.

Al mismo tiempo, fue necesario contar con los servicios de profesionales encargados de la edición y corrección de los textos, el diseño y la ilustración. Es por esto que también agradecemos al equipo cuyo talento permitió tener este material en su forma actual: Ediciones Colihue, Ignacio Miller, Alejandro Palermo, Luciano Andújar, Celeste Maratea y Luz Rodríguez.

En cuanto al **uso del género gramatical en esta colección**, hemos decidido respetar la norma vigente del uso del masculino genérico para grupos mixtos. Conscientes de que esta decisión deja pasar una oportunidad para contribuir a la construcción de una norma más inclusiva, optamos por un texto que resultara menos disruptivo.

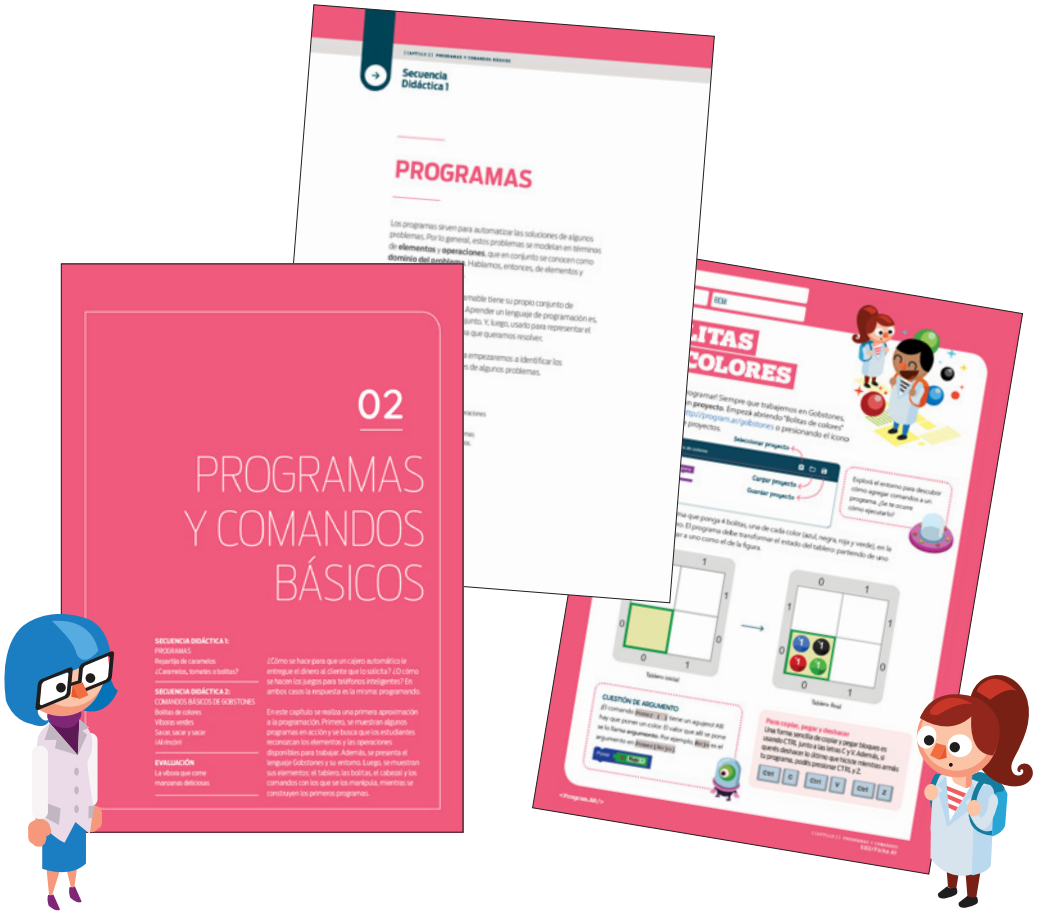
Esta colección se pone a disposición del público con **licencia Creative Commons**[1]**,** como una forma de incentivar la creación de obras derivadas. Dicho de otra forma, fomentamos activamente que las y los colegas generen sus propias versiones de este material y las compartan con la comunidad.

Estos manuales son para nosotros una versión 1.0 que, con un fuerte anclaje en el aula, **no deja de tener carácter experimental**. Muchos de los temas abordados cuentan con pocos antecedentes en la bibliografía internacional: existe mucho material para enseñar programación inicial a niños y adolescentes, mucho menos sobre cómo funciona una computadora, y muy poco sobre otras cuestiones tales como el funcionamiento de Internet, la representación de la información o la inteligencia artificial. Sabemos también que algunos temas que merecerían tener un lugar han quedado afuera, algo que pensamos subsanar en próximas ediciones.

Nos encantaría enriquecer este material con los aportes de la comunidad: docentes, académicos, investigadores e interesados en la temática en general están invitados a acercarnos sus comentarios, críticas y sugerencias escribiendo a info@program.ar. A su vez, queda abierta la invitación a revisar periódicamente nuestro sitio web o seguirnos en las redes sociales, para mantenerse al tanto de futuras versiones.

María Belén Bonello y Fernando Schapachnik
Coordinadores de la Iniciativa Program.AR
Fundación Dr. Manuel Sadosky

[1] Específicamente, una licencia Creative Commons Atribución-No Comercial-CompartirIgual 4.0 Internacional, cuyos detalles pueden consultarse en https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es.



INTRODUCCIÓN

La computación es la ciencia que se ocupa de los procesos que describen y transforman información. A través de su estudio se sientan las bases para el diseño, la programación y el uso de computadoras digitales. Desde hace varios años, son muchos los ámbitos que aprovechan elementos de este campo de conocimiento e incorporan computadoras. Solo por citar algunos, se puede mencionar la agricultura (que utiliza la computación para realizar análisis de suelos, plagas y hacer un control automático de maquinaria, entre otras cosas), la arquitectura (para diseñar, planificar y hacer seguimientos de obras y procesos arquitectónicos) y la medicina (en la realización de diagnósticos, intervenciones quirúrgicas, etc.).

Además, muchas herramientas de uso cotidiano como el correo electrónico, las redes sociales, las plataformas de comercio electrónico y los servicios de mensajería instantánea existen únicamente como consecuencia del desarrollo de esta ciencia. La ubicuidad de las soluciones computacionales hace imprescindible comprender los principios de la computación para tener una visión crítica del mundo que nos rodea.

OBJETIVO

Este manual es una contribución a la enseñanza de las Ciencias de la Computación en cuanto disciplina fundamental del saber humano. Aborda principalmente conocimientos de programación de computadoras, pero también incluye una perspectiva histórica de la disciplina, reflexiones sobre su presencia en nuestro quehacer cotidiano, una introducción a los elementos de *hardware* que permiten que las computadoras funcionen y al modo en que las computadoras representan información. No se presupone ningún conocimiento previo de computación; sin embargo, al finalizar, los estudiantes serán capaces de crear programas y juegos de computadora simples, con lo que se busca fomentar su imaginación y la comprensión del funcionamiento de muchos de los programas que manejan todos los días.

CONTEXTO

La existencia de este manual se enmarca en un trabajo mucho más vasto. En el año 2013, el entonces Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación y la Fundación Sadosky comenzaron la Iniciativa Program.AR, que busca promover la enseñanza de las Ciencias de la Computación en las escuelas primarias y secundarias de todo el país. Esta iniciativa responde a la preocupación que comparte Argentina con una veintena de países por el desconocimiento que muestran los niños y jóvenes respecto de los sistemas informáticos que utilizan.

Los "chicos" son grandes usuarios y consumidores de tecnologías elaboradas por terceros, pero no pueden modificarlas, crearlas ni producirlas porque no comprenden sus rudimentos. Este manual, así como otros desarrollados por otras universidades en el marco de la misma iniciativa, es un instrumento más de una política educativa orientada a mejorar la oferta de enseñanza en contenidos de las Ciencias de la Computación en las escuelas. Estos saberes ayudan a entender cómo funcionan las soluciones informáticas a nuestro alrededor, y contribuyen así a la formación de pensamiento crítico y al desarrollo de ciudadanos digitales responsables, en lugar de meros consumidores.

LA COMPUTACIÓN EN EL AULA

En nuestro país, la computación se comenzó a enseñar en las escuelas primarias y secundarias en la década de 1980. Inicialmente, los contenidos se centraban en los comandos básicos y nociones elementales de *hardware*. Más adelante, en la década de 1990, se le dio a la enseñanza un enfoque utilitario, orientado principalmente a la ofimática –el uso de las computadoras para resolver problemas de oficina, como escribir textos o realizar cuentas en planillas de cálculo–. En la primera década del siglo XXI, se integraron las Tecnologías de la Información y la Comunicación a la enseñanza de varias disciplinas como un recurso para facilitar y potenciar la práctica de cada área. No obstante, todos estos enfoques consideraban las computadoras como herramientas y no tenían en cuenta la Computación como objeto de estudio en sí mismo. Recién en la segunda década del siglo XXI se reconoció la necesidad de enseñar Computación de la misma forma en que se enseña Matemáticas o Prácticas del Lenguaje: con el objetivo de que los alumnos adquieran los conocimientos fundamentales de la disciplina que les permitan entender el mundo que los rodea y hacer un uso eficiente y responsable de la tecnología.

PROPUESTA

En esta propuesta se incorporan varios avances realizados en la última década. Uno de ellos es el uso de plataformas didácticas para iniciarse en la programación. Se trabaja con ambientes basados en bloques, que simplifican el proceso de construcción de programas, recuperan un aspecto lúdico que los hace amenos para los jóvenes y permiten construir programas con una cantidad mínima de trabajo. Otros avances tienen que ver con la didáctica específica de la programación. El uso de la didáctica por indagación, la selección cuidadosa de conceptos fundamentales y la presentación gradual de las ideas son algunas de las innovaciones incorporadas en este libro. Nuestro país es pionero en algunos de estos avances que se están desarrollando simultáneamente en todo el mundo, y el presente manual da cuenta de ello.

PRINCIPIOS QUE GUIARON EL DESARROLLO DEL MANUAL

Didáctica por indagación

La didáctica por indagación es un enfoque pedagógico que propone que sean los mismos estudiantes quienes descubren los conceptos y las ideas que se necesitan para resolver un problema. Las actividades del manual proponen preguntas, consignas y dinámicas que serán de utilidad a los docentes para guiar este proceso de descubrimiento. En general, comienzan con un desafío que requiere o bien la combinación novedosa de conceptos previamente presentados, o bien la búsqueda de uno nuevo. Solamente después que los estudiantes hayan intentado solucionar el problema es el momento de presentar en forma acabada las ideas trabajadas. De esta forma, el concepto nuevo no les resulta extraño a los estudiantes. Simplemente, se trata de la conclusión de ideas que ya pusieron en práctica.

Al trabajar con este enfoque didáctico, hay que tener en cuenta el modo de abordar los errores y las emociones. El proceso de adquisición de conocimientos nuevos inevitablemente implica atravesar una serie de estados emocionales que van desde la frustración frente a la dificultad hasta el entusiasmo y la satisfacción cuando se consigue dar con la solución de un problema. Estas emociones se potencian cuando se trabaja en un ambiente con propuestas que desafían a los estudiantes. Resulta importante que los docentes acompañen el proceso de aprendizaje ayudando a la clase a manejar sus emociones. Se recomienda, en consecuencia, abordar el error como una oportunidad para reflexionar sobre lo realizado y buscar alternativas.

De forma intencional no se especifica cuánto tiempo debe dedicarse a cada actividad. Dado que se espera que los estudiantes se involucren con las consignas propuestas y puedan realizar efectivamente la exploración que conduce al aprendizaje, se deja a criterio del docente el manejo de los tiempos. Se recomienda, sin embargo, estar atentos a que se respeten tres etapas fundamentales: el tiempo al comienzo de cada actividad, para que se familiaricen con los contenidos; el tiempo destinado a que los estudiantes inventen y desarrollen posibles soluciones y el tiempo dedicado al cierre de la actividad, siempre teniendo en cuenta el objetivo buscado.

Trabajo en equipo

Muchas de las actividades están pensadas para que los estudiantes trabajen en parejas o grupos. Esto alienta el trabajo colaborativo, favorece la discusión de alternativas, mejora la comunicación y permite desarrollar aprendizajes mucho más profundos. El trabajo en equipo no solo es una característica propia de esta disciplina, sino que promueve aprendizajes genuinos. Es por ello que debe favorecerse y alentarse el trabajo en grupos en todas las actividades que así lo indiquen.

En las actividades de programación se propone que los estudiantes trabajen en parejas: uno de ellos cumple el rol de piloto que opera la computadora y el otro el de copiloto que aporta ideas y sugerencias. Se espera que alternen los roles al pasar de una actividad a otra. Esta práctica sirve para fortalecer el respeto por el trabajo del otro: el copiloto debe expresar sus ideas claramente y tratar de no utilizar la computadora antes del cambio de roles. Debe supervisarse, entonces, el respeto en la comunicación y el intercambio de roles durante el desarrollo de las actividades.

Mínima necesidad de conocimientos previos en el área

Dado que el foco está puesto fundamentalmente en la programación, la mayoría de las actividades precisan de una computadora y un entorno de programación. Sin embargo, solamente se presupone un uso básico de las computadoras: ejecutar una aplicación, abrir o guardar archivos, utilizar menús y botones a través del ratón. No hace falta ningún conocimiento previo de programación ni de matemáticas para comprender los contenidos. Esto es importante porque el material puede ser útil para los estudiantes y para un amplio número de docentes, no solo para un grupo específico ya formado en el área.

Existen experiencias de equipos de investigación y desarrollo didáctico en la enseñanza de las Ciencias de la Computación que muestran que una gran mayoría de docentes con diversas trayectorias de formación y sin conocimientos previos en computación pueden apropiarse de conceptos del área ofrecidos de modo introductorio y presentarlos a sus alumnos de manera efectiva.

Foco en aspectos conceptuales

La propuesta se basa en un conjunto de conceptos e ideas fundamentales de la disciplina, más que en los detalles específicos de la tecnología actual. Es bien sabido que las tecnologías cambian a gran velocidad; sin embargo, los conceptos subyacentes permanecen. El estudio de las Ciencias de la Computación se centra en estos conceptos. Por este motivo, en este manual las actividades se orientan a la transmisión de un conjunto básico de ideas. Al realizar cambios en la presentación de las actividades, en su secuenciación o en sus detalles, debe tenerse en cuenta este conjunto de conceptos que constituye el núcleo de la contribución de este trabajo.

Al poner el foco en ideas y conceptos novedosos, un aspecto importante de la enseñanza es la adquisición del vocabulario específico de esta disciplina y su uso adecuado. En cada capítulo se trabaja especialmente la incorporación de léxico propio del área, se definen los términos con precisión y se explica cómo utilizarlos; además, todas las palabras técnicas presentadas están compiladas en un glosario. Se

recomienda prestar especial atención a la utilización adecuada del vocabulario, ya que suele ser una gran fuente de confusión, especialmente para personas que se están iniciando en el tema, o cuando el mismo concepto se nombra de diferentes maneras o en forma imprecisa, o cuando se utiliza un mismo término para denominar dos conceptos diferentes. Las explicaciones que acompañan cada actividad están pensadas para ayudar a la incorporación de los conceptos y el léxico por parte del docente, para que luego pueda utilizarlos de modo adecuado durante el trabajo en el aula.

Multiplicidad de soluciones

En programación, es usual que cada problema pueda abordarse de múltiples y diversas formas, y que muchas de ellas permitan llegar a soluciones satisfactorias. En las actividades propuestas se enfatiza que cada una de las soluciones que se proponen son **una** solución y no **la** solución. Por lo tanto, los docentes deben estar atentos a la aparición de soluciones no contempladas en este manual, y estar dispuestos a juzgarlas en función de los conceptos que se espera trabajar. Esto alienta la creatividad de los estudiantes y también promueve la exploración.

Filosofía de producción y distribución de los contenidos

Todo el material aquí presentado se distribuye bajo una licencia Creative Commons, Compartir Derivadas Igual, que permite y fomenta la copia y reproducción del contenido y también su modificación por parte de los docentes para adaptarlo a las realidades de su escuela y del entorno social de sus alumnos. Esto es de capital importancia, pues no se trata de un material definitivo, bajo control estricto de los autores, sino de un contenido vivo que puede ser –y se espera que sea– apropiado por los docentes para ampliarlo y mejorarlo, de modo de favorecer la difusión de las ideas fundamentales y contribuir así al desarrollo de la enseñanza de las Ciencias de la Computación.

PRINCIPALES CONCEPTOS ABORDADOS

Programas, legibilidad y estrategias de solución

El tema central de este manual es la programación de computadoras. Un **programa** es una forma de expresar una solución a un problema de forma tal que una máquina pueda ejecutarlo. En este sentido, los programas no son más que un conjunto de símbolos –letras, números, etc.– que describen la solución que pensó alguien. Para poner efectivamente en funcionamiento un programa, tiene que haber una máquina que lo ejecute. La Programación, como área dentro de las Ciencias de la Computación, trabaja sobre los programas y la forma de construirlos y no sobre las máquinas que los ejecutan.

Una cuestión fundamental es que, además de ser ejecutados por una máquina, los programas tienen que poder ser leídos y entendidos por personas: para poder corregir, adaptar o modificar un programa, es imprescindible que pueda leerse y comprenderse con facilidad. Por ese motivo, la **legibilidad** de los

programas es uno de los ejes fundamentales que se trabajan en este manual.

Por otro lado, las máquinas solo pueden ejecutar un conjunto reducido de acciones simples. Para referirnos a ellas contamos con lo que llamamos **comandos básicos**. Sin embargo, al enfrentarnos a problemas complejos, resulta engorroso y complicado pensar formas de solucionarlos en términos de estos comandos. La Programación, como disciplina, provee herramientas para que podamos pensar programas considerando los elementos del problema abordado. En tal sentido, otro de los ejes del manual consiste en la construcción de **estrategias de solución**. Es decir, las ideas sobre cómo encarar la solución particular a un problema, qué elementos disponer para lograrla y de qué manera.

Procedimientos

Los lenguajes de programación proveen diversas herramientas para expresar ideas al construir programas, por ejemplo, los comandos básicos, que podemos usar para indicarle a la computadora qué acciones básicas llevar a cabo. Sin embargo, en general, estos comandos resultan insuficientes para construir programas legibles y expresar soluciones con el vocabulario del problema abordado.

Una de las herramientas fundamentales para construir programas son los **procedimientos**, que permiten que un programador defina sus propios comandos y decida su comportamiento. Si se les ponen nombres adecuados, los procedimientos pueden expresar soluciones en términos de los elementos que son propios del problema a resolver. Por lo tanto, contribuyen a la construcción de programas legibles. Además, se pueden usar para dividir la solución de un problema en partes más simples (o subproblemas) y luego combinarlas para resolver el problema original. De este modo, dan la posibilidad de plasmar en el programa la estrategia de solución escogida.

En síntesis, es muy importante que cada vez que se busque resolver un problema se empiece por pensar cuáles son la estrategia de solución y las subtareas involucradas. Luego se deben utilizar estas ideas para definir los procedimientos que constituirán el programa.

Otras herramientas de los lenguajes de programación

Si bien en este manual se utiliza el lenguaje de programación Gobstones, no es allí donde está puesto el foco. Las herramientas que se estudian son transversales a todos los lenguajes de programación. Una vez que se las conoce, se puede aprender rápidamente un nuevo lenguaje investigando de qué forma aparecen en él esas mismas herramientas. Concretamente, además de los procedimientos, se trabaja sobre repetición simple, repetición condicional, alternativa condicional, parametrización, expresiones, operadores, tipo de datos, asignación, etc.

Representación de la información

Todos los programas utilizan y transforman **información**. El texto, el sonido y las imágenes son algunos ejemplos de información que procesan las computadoras. Para poder escribir programas capaces

de manipular estos datos, la información debe representarse de algún modo.

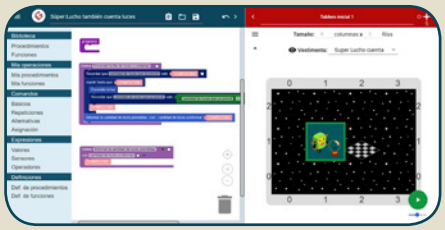
Internamente, las computadoras modernas representan la información utilizando dos estados que corresponden a la presencia o ausencia de corriente. Sin embargo, para referirnos a estos dos estados usamos los números 0 y 1. A esta unidad mínima de información se la denomina *bit*, y es la base de todas las representaciones de información de las computadoras actuales. No obstante, rara vez un programador piensa en términos de bits; en su lugar, se utilizan abstracciones que permiten razonar en términos de texto, imágenes, etc., y no tener que detenerse a pensar cómo se las representa con un sistema binario. En este manual se trabajan las ideas de representación de información en forma similar a como lo hacen los sistemas de computación.

EL LENGUAJE DE PROGRAMACIÓN GOBSTONES

Gobstones –que se pronuncia *Góbstouns* porque viene del inglés– es el lenguaje de programación que se usa en este manual. Su nombre fue inventado por J.K. Rowling en las novelas de Harry Potter para denominar un juego de bolitas mágicas. Este lenguaje fue desarrollado por un equipo de la Universidad Nacional de Quilmes con el exclusivo propósito de servir como primer lenguaje de programación al aprender a programar, y se viene utilizando con éxito en diferentes ámbitos educativos.

HECHO EN QUILMES

Gobstones fue desarrollado por un equipo de la Universidad Nacional de Quilmes, especialmente diseñado para enseñar a programar.



Un programa en Gobstones está formado por una serie de bloques cuyas formas, funciones y características se irán descubriendo a lo largo de las actividades. Los programas se utilizan para operar sobre el **cabezal** de una máquina que trabaja sobre un **tablero**. En las celdas del tablero el cabezal puede colocar y retirar **bolitas de colores**. Además, puede desplazarse de una celda a celdas vecinas.

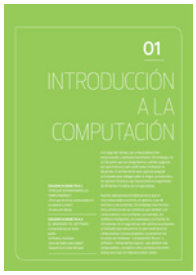
Al ejecutar un programa, se comienza con un tablero particular, al que llamamos **tablero inicial**. El resultado final es otro tablero, denominado **tablero final**, que se obtiene luego de modificar el tablero inicial siguiendo las instrucciones del programa. En Gobstones, los programas siempre se usan para transformar un tablero inicial en un tablero final.

El **efecto** del programa sobre el tablero es el conjunto de cambios entre un tablero inicial y uno final. En este manual, todos los desafíos que se presentan se expresan en términos del efecto esperado sobre el tablero. En muchos casos, los programas consisten en instrucciones para transformar un conjunto inicial de cosas –una biblioteca desordenada, los ingredientes para hacer pizza, etc.– en otro final –la misma biblioteca ordenada, la pizza lista–.

El entorno que usaremos para trabajar con Gobstones se llama Gobstones Jr., y puede utilizarse tanto en línea como sin conexión a Internet, luego de haberlo descargado e instalado en la computadora. En este entorno, las instrucciones se representan con bloques que pueden encastrarse y desencastrarse para formar programas, con lo que se evitan problemas de tipo sintáctico, que suelen ser muy frustrantes para aquellos que se inician en la programación.

¿QUÉ CONTENIDOS ABORDA CADA CAPÍTULO?

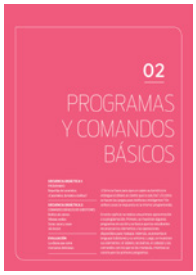
Cada capítulo de este manual está integrado por una serie de secuencias didácticas que tienen una coherencia conceptual. Cada secuencia didáctica está formada por actividades que pueden o no requerir el uso de una computadora. Cada actividad propone formas de abordar la clase, un desarrollo y un cierre. Además, muchas están acompañadas de una ficha para entregar a los estudiantes. La mayoría de los capítulos incluyen una actividad que integra los contenidos trabajados y se propone como modelo de evaluación.



CAPÍTULO 1: INTRODUCCIÓN A LA COMPUTACIÓN

A lo largo del tiempo, las computadoras han evolucionado y continúan haciéndolo. Sin embargo, no es frecuente que nos preguntemos cuándo surgieron, por qué motivos y qué condiciones motivaron su desarrollo. El primer eje de este capítulo propone actividades para indagar sobre el origen, la evolución y las razones históricas que favorecieron el surgimiento de diferentes modelos de computadoras.

Nuestra representación habitual de lo que es una computadora se limita, en general, a las de escritorio y las portátiles. Sin embargo, hay muchos otros artefactos de uso cotidiano que también son computadoras o las contienen, por ejemplo, los teléfonos inteligentes, los lavarropas y los hornos de microondas. En el segundo eje del capítulo se proponen actividades para deconstruir la idea tradicional de computadora. Con ese propósito, se presentan las nociones de *hardware* –componentes físicos– y *software* –componentes lógicos– que definen una computadora y se explica cómo la interacción entre ambos hace que la máquina realice tareas.



CAPÍTULO 2: PROGRAMAS Y COMANDOS BÁSICOS

¿Cómo se hace para que un cajero automático le entregue el dinero al cliente que lo solicita? ¿O cómo se hacen los juegos para teléfonos inteligentes? En ambos casos la respuesta es la misma: programando.

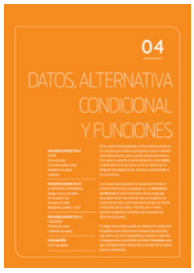
En este capítulo se realiza una primera aproximación a la programación. Primero, se muestran algunos programas en acción y se busca que los estudiantes reconozcan los elementos y las operaciones disponibles para trabajar. Además, se presenta el lenguaje Gobstones y su entorno. Luego, se muestran sus elementos: el tablero, las bolitas, el cabezal y los comandos básicos con los que se los manipula, mientras se construyen los primeros programas.

CAPÍTULO 3: PROCEDIMIENTOS Y REPETICIÓN SIMPLE

Programar usando únicamente comandos básicos puede volverse muy engorroso. El resultado son programas difíciles de comprender cuando los leemos. Por eso, los lenguajes brindan una herramienta llamada procedimiento, que permite que quien programa defina sus propios comandos. Bien utilizados, los procedimientos resultan muy útiles para construir programas claros y ordenados. Una vez definido su comportamiento, un procedimiento puede utilizarse igual que cualquier otro comando.

Por otro lado, es habitual que, al escribir programas, nos encontremos frente a la necesidad de repetir varias veces una serie de acciones. Casi todos los lenguajes de programación nos permiten expresar repeticiones sin necesidad de reiterar comandos en forma explícita. Una de las formas más simples de hacerlo es la repetición simple, que se utiliza para repetir instrucciones una cantidad fija de veces.

En este capítulo se abordan los procedimientos y las repeticiones simples, que son esenciales para construir programas.



CAPÍTULO 4: DATOS, ALTERNATIVA CONDICIONAL Y FUNCIONES

En los capítulos precedentes, el foco estuvo puesto en las acciones que realiza un programa, tanto mediante comandos básicos como a través de procedimientos. Este capítulo aborda en profundidad el uso de datos. Si entendemos las acciones como los verbos de un lenguaje de programación, los datos corresponden a los sustantivos.

Los valores que adquieren los datos determinan el comportamiento de los programas. La alternativa condicional es una herramienta de los lenguajes de programación que permite que un programa se comporte de uno u otro modo de acuerdo con ciertas condiciones de los datos. Permite, por lo tanto, construir programas versátiles que funcionen en distintos escenarios.

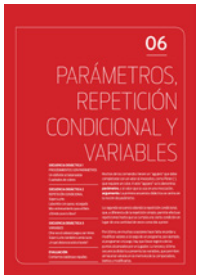
El origen de los datos puede ser diverso. En Gobstones se pueden usar tanto valores literales (por ejemplo, rojo, norte o 5) como operadores y sensores. Además, el lenguaje da la posibilidad de definir funciones, para que el programador calcule datos de una forma que le resulte conveniente.

CAPÍTULO 5: REPRESENTACIÓN DE LA INFORMACIÓN

Al utilizar una computadora, interactuamos con información de muy variado tipo: texto, números, imágenes, música, video, etc. Como usuarios, recibimos esa información a través de una pantalla o parlantes, por ejemplo. Sin embargo, internamente, la computadora la almacena utilizando solamente secuencias de bits, es decir, secuencias de ceros y unos.

Este capítulo se centra en cómo representar tres tipos diferentes de información: números, imágenes y texto. Comenzamos abordando la representación binaria de los números y anali-

zando cómo se determina cuánto ocupa o "pesa" la información. A continuación, se trabaja sobre la representación de imágenes a color, imágenes en blanco y negro e imágenes comprimidas. Por último, incursionamos en la representación de texto y trabajamos sobre estrategias para enviar mensajes de forma secreta con un método de cifrado simétrico clásico.



CAPÍTULO 6: PARÁMETROS, REPETICIÓNES CONDICIONAL Y VARIABLES

Muchos de los comandos tienen un "agujero" que debe completarse con un valor al invocarlos, como *Poner [ ]*, que requiere un color. A este "agujero" se lo denomina parámetro, y al valor que se usa en una invocación, argumento. La primera secuencia didáctica se centra en la noción de parámetro.

La segunda secuencia aborda la repetición condicional, que, a diferencia de la repetición simple, permite efectuar repeticiones hasta que se cumpla una cierta condición en lugar de una cantidad de veces conocida *a priori*.

Por último, en muchas ocasiones hace falta recordar y modificar valores a lo largo de un programa; por ejemplo, al programar un juego, hay que llevar registro de los puntos alcanzados por un jugador. La tercera y última secuencia didáctica presenta las variables, que permiten almacenar valores en la memoria de la computadora, leerlos y modificarlos.

CAPÍTULO 7: INTERACTIVIDAD

En muchos de los programas que habitualmente utilizamos existe una interacción entre el programa y quien lo usa: presionamos las teclas y vemos que aparece texto en la pantalla, movemos el ratón y vemos que un puntero se desplaza, etc. En este capítulo se trabaja con los programas de esta clase, que se conocen como programas interactivos.

El capítulo consta de dos secuencias didácticas. La primera presenta la programación interactiva a través de dos proyectos simples. La segunda propone la programación de dos videojuegos que, además, permiten utilizar muchas de las herramientas estudiadas en capítulos anteriores.

Además de lo presentado en este libro, también desarrollamos actividades sobre Internet, Sistemas operativos y desafíos adicionales de programación que complementan los de este manual. Estos contenidos también son libres y gratuitos y se encuentran disponibles en el sitio web de Program.AR. Se ofrece a continuación un breve resumen de este material complementario.

ANEXO I: INTERNET

A diario escuchamos hablar de Internet o de la nube como algo que existe aunque no sabemos dónde se encuentra. Nos permite llevar a cabo muchas de nuestras tareas cotidianas: escuchar música, mirar películas, hacer una copia de seguridad de todas nuestras fotos y videos, y editar colaborativamente un documento. Los dispositivos móviles que usamos ya están conectados a Internet, los televisores inteligentes también y, poco a poco, también lo están haciendo los electrodomésticos más clásicos como las heladeras o los lavarropas.

A pesar de esto, rara vez hemos reflexionado sobre este fenómeno: ¿dónde está físicamente la información que guardamos en la nube? ¿Tiene dueños Internet? ¿Es posible que una persona cree su propia nube? ¿Qué costos tiene y quién los paga? ¿Alguien puede ver los sitios por los que otra persona navega en Internet? A lo largo de este anexo se promueve la reflexión sobre estas preguntas para comenzar a esbozar algunas respuestas.

ANEXO II: SISTEMAS OPERATIVOS

Una computadora de escritorio moderna tiene en general un procesador principal –o varios–, memoria RAM, discos rígidos, placa de red, monitores, teclado, ratón y otros dispositivos. Según el tipo de computadora y su uso, podemos encontrar distintas combinaciones; por ejemplo, esperamos ver un teclado y un ratón en una computadora de escritorio, mientras que nos conformamos con una pantalla táctil en una tableta. Llamamos *plataforma* al conjunto de componentes compatible con el propósito de un dispositivo dado.

Si al programar tuviéramos que tener en cuenta todas las combinaciones posibles de dispositivos y las particularidades de cada fabricante, la tarea sería muy compleja. Para reducir esta complejidad, las computadoras utilizan una pieza de *software* llamada *sistema operativo*, que se encarga de proveer una interfaz homogénea a los distintos tipos de plataformas. Es decir, quien programa no tiene que conocer los detalles de cada dispositivo y fabricante, y puede administrar los recursos disponibles. El objetivo de este anexo es presentar la noción de sistema operativo.

ANEXO III: ACTIVIDADES DE PROGRAMACIÓN

En este anexo se incluyen ocho desafíos de programación que complementan los del Manual. Con ellos, se podrá trabajar sobre representación de números, procedimientos con parámetros, repetición condicional y recorridos, cálculo y memorización de datos y programas interactivos.

Fichas para estudiantes

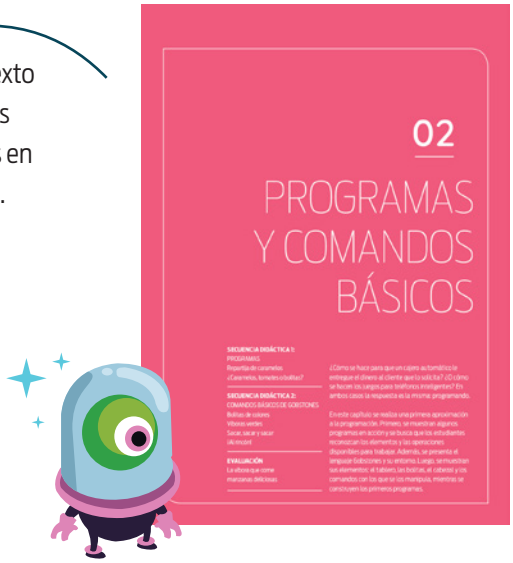
A fin de facilitar el trabajo, están disponibles en un solo lugar todas las fichas para estudiantes, reunidas en un archivo PDF libre y gratuito. Se encuentra en dos versiones: a color y en blanco y negro, y puede descargarse desde el sitio web de Program.AR.

¿CÓMO ORGANIZAMOS ESTE MANUAL?

Cada uno de los capítulos tiene las siguientes características:

APERTURA

Consta de un breve texto introductorio sobre los conceptos abordados en el capítulo y un índice.



SECUENCIAS DIDÁCTICAS

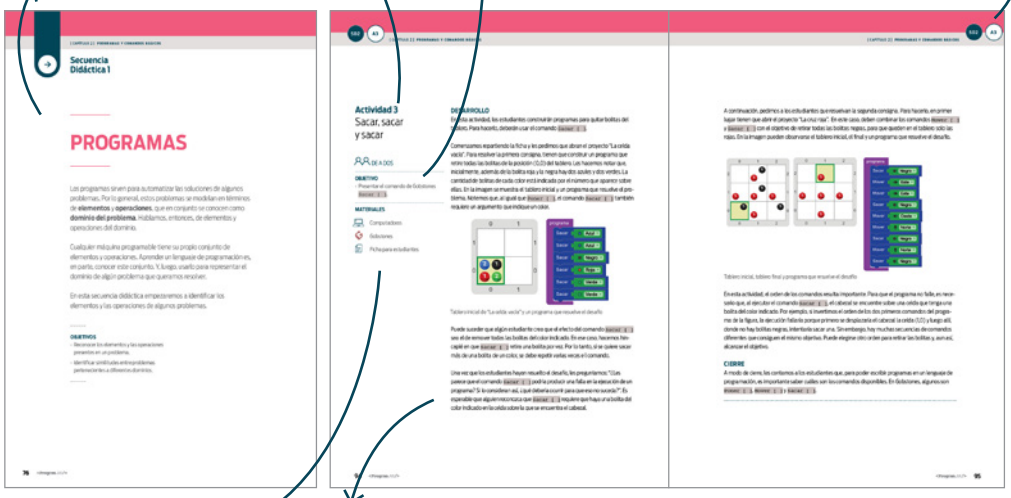
Cada capítulo está integrado por secuencias didácticas que tienen una coherencia conceptual. Cada secuencia didáctica está conformada por actividades que pueden o no requerir el uso de una computadora.

Presentación de la secuencia.

Título de la actividad.

Modalidad y objetivos.

Datos de orientación sobre el número de secuencia didáctica (SD) y el número de la actividad a la que corresponde la página.



Materiales para hacer la actividad.

Cada actividad propone formas de abordar la clase, un desarrollo, una sugerencia para resolver la propuesta y un cierre.

FICHAS PARA ESTUDIANTES

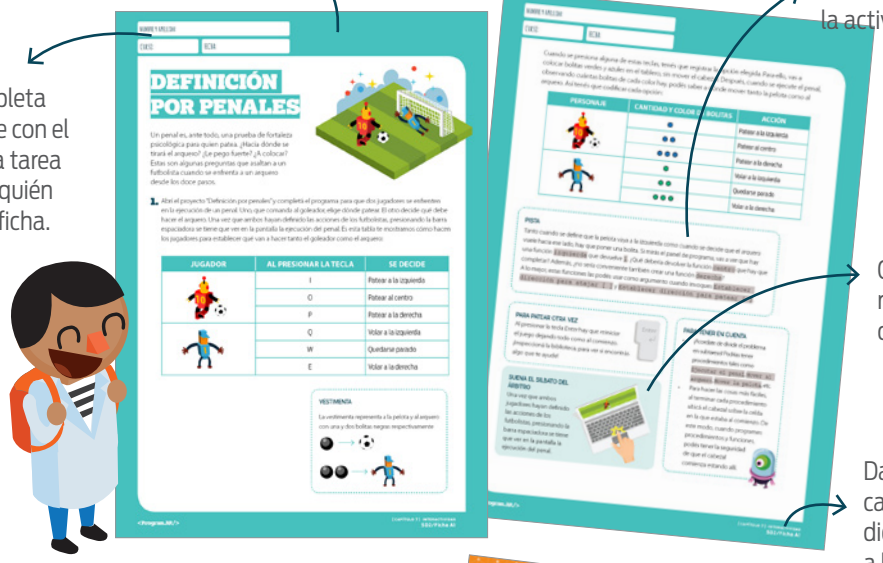
Muchas actividades vienen acompañadas de una ficha para entregar a los estudiantes.

Datos que completa cada estudiante con el fin de facilitar la tarea de identificar a quién corresponde la ficha.

Datos a tener en cuenta para resolver la actividad

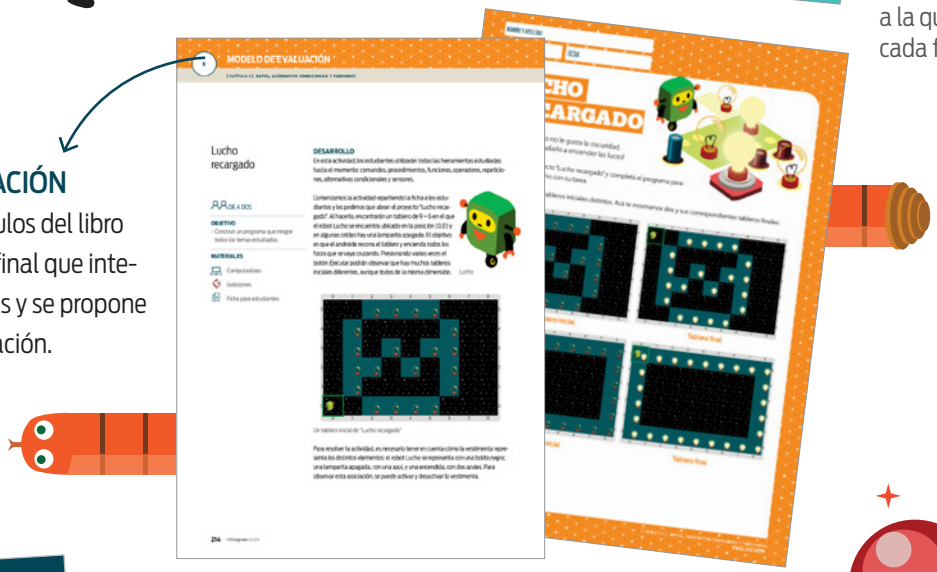
Curiosidades relacionadas con el tema.

Datos sobre el capítulo, secuencia didáctica y actividad a la que corresponde cada ficha.



MODELO DE EVALUACIÓN

La mayoría de los capítulos del libro incluyen una actividad final que integra los temas abordados y se propone como modelo de evaluación.

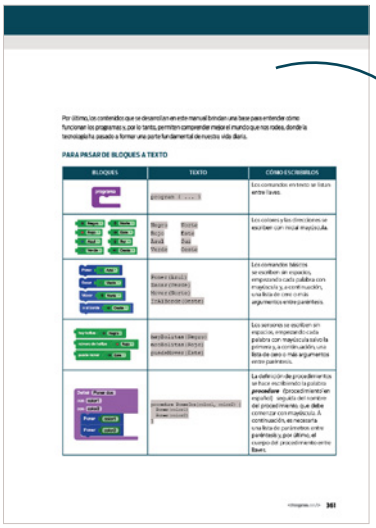


GLOSARIO

Las definiciones no están pensadas para que el docente se las dé a los estudiantes y se sugiere no presentarlas de forma descontextualizada.

EPÍLOGO

Destinado a quienes les interesa seguir aprendiendo programación más allá de un lenguaje basado en bloques. Se facilita información conceptual y una tabla para pasar los bloques de Gobstones a texto.



01

INTRODUCCIÓN A LA COMPUTACIÓN

A lo largo del tiempo, las computadoras han evolucionado y continúan haciéndolo. Sin embargo, no es frecuente que nos preguntemos cuándo surgieron, por qué motivos y qué condiciones motivaron su desarrollo. El primer eje de este capítulo propone actividades para indagar sobre el origen, la evolución y las razones históricas que favorecieron el surgimiento de diferentes modelos de computadoras.

Nuestra representación habitual de lo que es una computadora se limita, en general, a las de escritorio y las portátiles. Sin embargo, hay muchos otros artefactos de uso cotidiano que también son computadoras o las contienen, por ejemplo, los teléfonos inteligentes, los lavarropas y los hornos de microondas. En el segundo eje del capítulo se proponen actividades para deconstruir la idea tradicional de computadora. Con ese propósito, se presentan las nociones de *hardware* –componentes físicos– y *software* –componentes lógicos– que definen una computadora y se explica cómo la interacción entre ambos hace que la máquina realice tareas.

SECUENCIA DIDÁCTICA 1:
¿POR QUÉ SE INVENTARON LAS COMPUTADORAS?
¿Para qué sirven las computadoras si ya sabemos contar?
Un poco de historia

SECUENCIA DIDÁCTICA 2:
EL *HARDWARE* Y EL *SOFTWARE*
Computadoras por todas partes
Software y *hardware*
¿Qué son todos esos cables?
¡Saquemos el robot del aula!

Secuencia Didáctica 1

¿POR QUÉ SE INVENTARON LAS COMPUTADORAS?

Las computadoras y las tareas que realizan ya son indispensables en nuestras vidas; sin embargo, muchos no saben sobre sus orígenes. ¿En qué momento surgieron y por qué? ¿Quiénes las usaban? ¿Qué tamaño tenían? A lo largo de esta secuencia didáctica buscaremos dar respuesta a estas y otras preguntas para comprender mejor la historia, el desarrollo, la utilidad y el impacto de las computadoras en nuestra sociedad.

OBJETIVOS

- Presentar una reseña de la historia de la computación.
- Conocer los motivos que propiciaron el nacimiento de las computadoras.
- Reflexionar acerca de la evolución de las computadoras.

Actividad 1
¿Para qué sirven las computadoras si ya sabemos contar?

INDIVIDUAL

OBJETIVOS

- Identificar las posibilidades que brindan las computadoras para procesar grandes volúmenes de datos.
- Comparar el tiempo que demanda procesar información con una computadora y sin ella.

MATERIALES

Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es que los estudiantes reflexionen sobre las posibilidades que tienen las computadoras para procesar grandes volúmenes de datos, y que comprendan que, para ciertas tareas –como el procesamiento de los datos de un censo–, los dispositivos computacionales resultan indispensables.

Comenzamos repartiéndoles las fichas y les indicamos que para resolver esta actividad no pueden utilizar ningún dispositivo electrónico como calculadoras, teléfonos inteligentes, computadoras, etc. Les pedimos que resuelvan la primera consigna, en la que encontrarán una tabla con los datos de un hipotético censo de habitantes del territorio argentino: provincia de residencia, edad, nivel educativo alcanzado, si usa o no computadoras y si tiene trabajo o se encuentra desempleado. Les explicamos que cada fila de la tabla corresponde a un individuo censado. Por ejemplo, la primera se refiere a una persona que vive en la provincia de Buenos Aires, de 61 años de edad, que completó el secundario pero no fue a la universidad –o no la terminó–, que no usa computadoras y que se encuentra desempleada.

PROVINCIA	EDAD	NIVEL EDUCATIVO ALCANZADO	¿USA COMPUTADORAS?	¿TRABAJA?
Buenos Aires	61	Secundario	No	No
Neuquén	35	Ninguno	No	Sí
Mendoza	70	Primario	No	No
Córdoba	73	Secundario	Sí	No
Santa Cruz	70	Ninguno	No	Sí
Catamarca	18	Secundario	No	Sí
La Rioja	68	Universitario	No	Sí
Formosa	17	Secundario	No	Sí
La Pampa	41	Secundario	No	No
San Luis	17	Secundario	Si	Sí

Datos del censo ficticio

Los estudiantes tienen que responder tres preguntas a partir de los datos brindados: ***(i)*** ¿qué cantidad de individuos menores de 18 años del nordeste argentino (Formosa, Chaco, Corrientes y Misiones) terminó el secundario y tiene trabajo?; ***(ii)*** ¿cuántos mayores de 65 años del Nuevo Cuyo (Mendoza, San Juan, San Luis y La Rioja) utilizan una computadora?; y ***(iii)*** ¿qué cantidad de habitantes de la Patagonia (Neuquén, Río Negro, Chubut, Santa Cruz, Tierra del Fuego, Antártida Argentina e Islas del Atlántico Sur) que no haya completado la primaria tiene trabajo? Las respuestas son 1, 0 y 2, respectivamente.

PROVINCIA	EDAD	NIVEL EDUCATIVO ALCANZADO	¿USA COMPUTADORAS?	¿TRABAJA?
Buenos Aires	61	Secundario	No	No
Neuquén	35	Ninguno	No	Sí
Mendoza	70	Primario	No	No
Córdoba	73	Secundario	Sí	No
Santa Cruz	70	Ninguno	No	Sí
Catamarca	18	Secundario	No	Sí
La Rioja	68	Universitario	No	Sí
Formosa	17	Secundario	No	Sí
La Pampa	41	Secundario	No	No
San Luis	17	Secundario	Sí	Sí

Filas que satisfacen las restricciones de las preguntas 1 (amarillo) y 3 (verde)

Luego de hacer una puesta en común, resaltamos que, a pesar de trabajar con muy pocos datos, hubo que chequear varias cosas antes de dar con las respuestas. A continuación les indicamos que continúen con la segunda consigna. Verán una tabla que tiene los datos de 33 individuos. Les indicamos: "Repitan el ejercicio, pero ahora con esta tabla". Unos instantes después preguntamos: "¿Qué les sucede cuando se hace el mismo ejercicio, pero con más datos?". Es probable que no les guste la propuesta y pongan cara de fastidio. Continuamos: "¿Les resulta engorroso tener que analizar esta información mirando las tablas?". Varios, si no todos, contestarán que sí. "¿Cuánto tiempo creen que les llevaría hacerlo? ¿Y si fueran los datos de un censo real, como el que en el 2010 en la Argentina dio como resultado que hay más de 40 millones de habitantes en el país?". Luego de discutir estas ideas con los estudiantes preguntamos: "¿Qué les parece que se puede usar para procesar estos volúmenes de información?". La respuesta es: "¡Computadoras!".

CIERRE

Reflexionamos junto con los alumnos sobre la velocidad a la que se pueden procesar datos usando computadoras. Mientras que hacerlo a mano lleva muchísimo tiempo, con una computadora puede hacerse a muy alta velocidad. Además, comentamos que, al controlar información a mano, es muy probable que se cometan errores que luego pueden ser muy difíciles de rastrear.

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿PARA QUÉ SIRVEN LAS COMPUTADORAS SI YA SABEMOS CONTAR?

Los censos sirven para relevar las principales características de las personas que habitan en una ciudad, un país o una región. Esta información es vital para planificar qué políticas públicas llevar adelante para mejorar las condiciones de vida de los habitantes. Pero ¿cómo se hace para procesar todos esos datos y obtener información valiosa?

1. Mirá la siguiente tabla y respondé las preguntas que están a continuación.

PROVINCIA	EDAD	NIVEL EDUCATIVO ALCANZADO	¿USA COMPUTADORAS?	¿TRABAJA?
Buenos Aires	61	Secundario	No	No
Neuquén	35	Ninguno	No	Sí
Mendoza	70	Primario	No	No
Córdoba	73	Secundario	Sí	No
Santa Cruz	70	Ninguno	No	Sí
Catamarca	18	Secundario	No	Sí
La Rioja	68	Universitario	No	Sí
Formosa	17	Secundario	No	Sí
La Pampa	41	Secundario	No	No
San Luis	17	Secundario	Si	Sí

a. ¿Qué cantidad de individuos menores de 18 años del nordeste argentino (Formosa, Chaco, Corrientes y Misiones) terminó el secundario y tiene trabajo?

_____

b. ¿Cuántos mayores de 65 años del Nuevo Cuyo (Mendoza, San Juan, San Luis y La Rioja) utilizan una computadora?

_____

c. ¿Qué número de habitantes de la Patagonia (Neuquén, Río Negro, Chubut, Santa Cruz, Tierra del Fuego, Antártida Argentina e Islas del Atlántico Sur) que no haya completado la primaria tiene trabajo?

_____

NOMBRE Y APELLIDO:

CURSO: FECHA:

2. A continuación hay una tabla con más datos.

PROVINCIA	EDAD	NIVEL EDUCATIVO ALCANZADO	¿USA COMPUTADORAS?	¿TRABAJA?
Formosa	80	Terciario/Universitario	No	No
Río Negro	26	Ninguno	Sí	Sí
La Rioja	57	Ninguno	Sí	No
Misiones	53	Terciario/Universitario	No	Sí
San Juan	61	Terciario/Universitario	No	No
Córdoba	72	Secundario	Sí	No
La Rioja	66	Ninguno	Sí	No
Misiones	76	Terciario/Universitario	Sí	Sí
Chaco	17	Primario	No	No
Salta	65	Ninguno	No	No
La Pampa	73	Terciario/Universitario	No	No
Santa Fe	73	Secundario	Sí	Sí
Santa Cruz	57	Primario	No	No
San Luis	37	Terciario/Universitario	Sí	No
Misiones	32	Primario	Sí	Sí
Formosa	79	Secundario	No	Sí
Río Negro	29	Ninguno	Sí	Sí
Catamarca	51	Primario	No	No
Catamarca	61	Terciario/Universitario	No	Sí
Salta	55	Primario	Sí	Sí
Chaco	16	Primario	Sí	Sí
Salta	43	Primario	Sí	No
Chaco	74	Ninguno	Sí	Sí
Río Negro	20	Terciario/Universitario	Sí	Sí
La Rioja	43	Primario	No	No
Tucumán	24	Secundario	No	No
Santa Fe	54	Primario	Sí	Sí
Formosa	14	Ninguno	Sí	Sí
Corrientes	15	Primario	No	Sí
Chubut	22	Primario	No	No
Salta	55	Secundario	No	No
Río Negro	58	Terciario/Universitario	Sí	No
Córdoba	79	Terciario/Universitario	Sí	No

NOMBRE Y APELLIDO:

CURSO: FECHA:

a. ¿Cuánto tiempo estimás que te llevaría responder las preguntas de la primera consigna con los datos de esta tabla? ¿Y si la tabla contuviera los datos de los más de 40 millones de personas que se censaron en el 2010 en la Argentina?

---

b. Al procesar mucha información a mano, ¿te parece probable que se cometan errores, tales como contar dos veces a una persona, o que nos olvidemos de alguna? ¿Por qué?

---

c. ¿Qué herramienta te parece que es útil para procesar grandes volúmenes de datos?

---

40 MILLONES Y CONTANDO...

El censo realizado en 2010 dio como resultado que, en aquel momento, vivían en Argentina 40.091.359 personas. Por primera vez, para procesar los datos relevados se usaron muchas computadoras con una gran capacidad de procesamiento. Eso permitió la publicación de la información apenas dos meses después de que los censistas hubieran pasado por las casas con una simple encuesta.

Actividad 2
Un poco de historia

GRUPAL (4)

OBJETIVOS

- Presentar una breve reseña de la historia de la computación.
- Identificar problemas que motivaron la construcción de algunas computadoras.

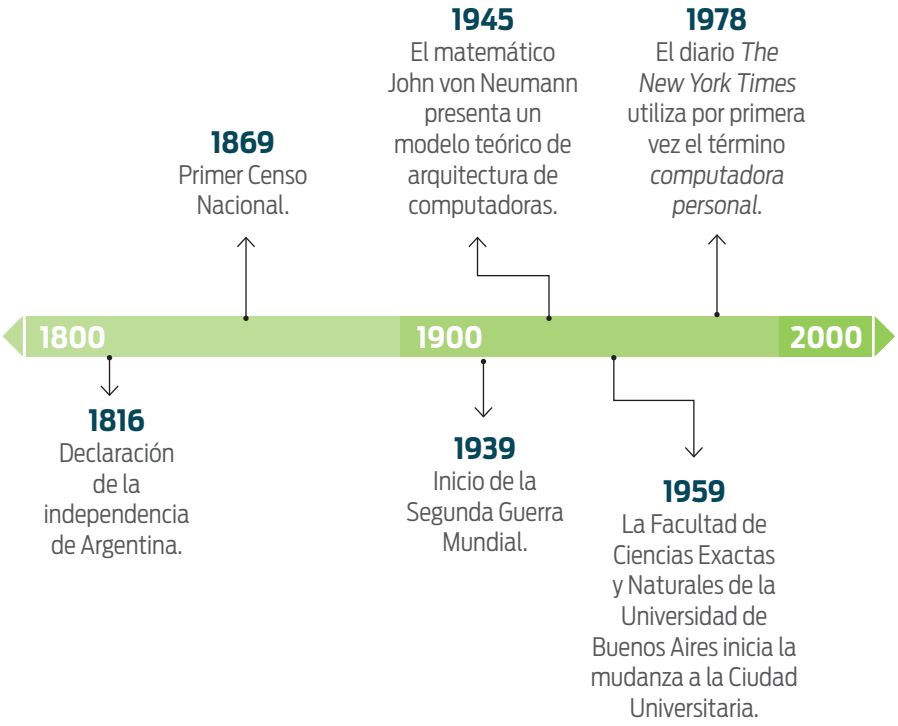
MATERIALES

- Computadora
- Internet
- Ficha para estudiantes

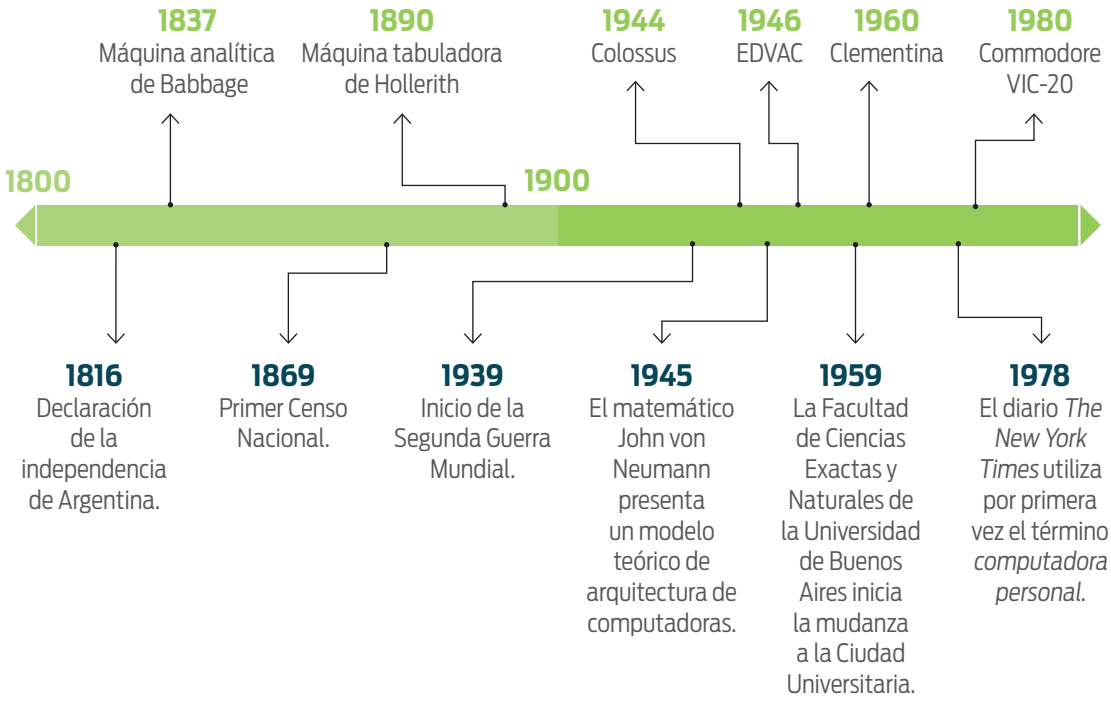
DESARROLLO

En esta actividad se realiza una pequeña reseña histórica sobre el origen y la evolución de las computadoras. Los estudiantes investigarán, en relación con algunas computadoras puntuales, los siguientes aspectos: *(i)* qué problemas motivaron su creación, *(ii)* por qué fueron significativas en el momento de su aparición y *(iii)* qué impacto tuvieron en la sociedad.

Antes de comenzar la actividad, copiamos en el pizarrón una línea de tiempo como la que se muestra a continuación. Los acontecimientos expuestos sirven tanto para dar una idea general de la época en que se construyeron las máquinas como para marcar hechos relacionados con su creación. Aquí proponemos incluir la declaración de la independencia de Argentina (1816), el primer censo nacional (1869), el inicio de la Segunda Guerra Mundial (1939), la presentación del modelo de arquitecturas de computadoras de Von Neumann (1945), la mudanza de la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires a la Ciudad Universitaria (1959) y la primera mención a las computadoras personales (1978). Estos hechos pueden complementarse con otros que resulten significativos para los estudiantes.



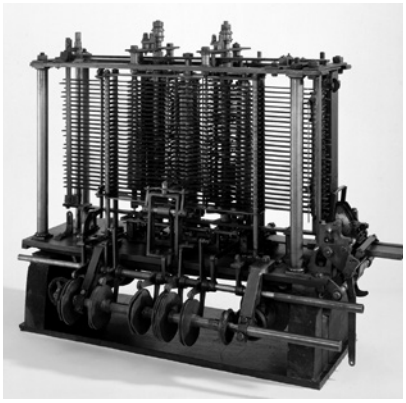
Iniciamos la conversación con los estudiantes preguntándoles: "¿Cuándo creen que comenzaron a existir las computadoras?". Escuchamos sus respuestas y les comentamos que ya en el siglo XIX se diseñaron computadoras, e incluso mucho antes se utilizaron instrumentos que de algún u otro modo facilitaban la realización de cuentas aritméticas. Les decimos: "En esta línea de tiempo vamos a marcar la aparición de seis computadoras sobre las que van a investigar". Entonces, agregamos en la línea de tiempo el año de creación de las siguientes computadoras: la máquina analítica de Babbage (1837), la máquina tabuladora de Hollerith (1890), la Colossus (1944), la EDVAC (1946), Clementina (1960) y la Commodore VIC-20 (1980).



Les repartimos la ficha a los estudiantes y les pedimos que armen grupos de cuatro para resolver la consigna. Allí se proporciona una serie de acontecimientos históricos que deben relacionar con las computadoras presentadas. A continuación se hace una breve reseña de cada una que sirve para completar la actividad.

Máquina analítica de Babbage

El matemático británico Charles Babbage, al que debe su nombre, la diseñó alrededor de 1837. En esa época, se utilizaban tablas de números con referencias para realizar los cálculos matemáticos sobre funciones trigonométricas y logarítmicas. Babbage se abocó a la tarea de crear una máquina para facilitar la creación de esas tablas y luego sofisticó su diseño para poder realizar funciones analíticas y todo tipo de cálculos, utilizando un formato similar al de los telares de la época. Su diseño contemplaba que la entrada de información se realizara con tarjetas perforadas, distintas unidades que efectuaban cálculos aritméticos y un mecanismo de salida para registrar los resultados, que combinaba el uso de un trazador de curvas y una campana.



Máquina analítica de Babbage

Por diversos desacuerdos entre su diseñador, la persona que la estaba construyendo y el gobierno de Gran Bretaña, la máquina no se hizo en su época, pero su diseño constituyó el origen de las computadoras creadas años más tarde. La matemática británica Ada Lovelace investigó las ideas de Babbage y creó varios programas para la máquina analítica. Ella fue, por lo tanto, la primera persona en hacer programas, y es por eso que se la conoce como la primera programadora.

Retrato de Ada Lovelace, la primera programadora

Máquina tabuladora de Hollerith

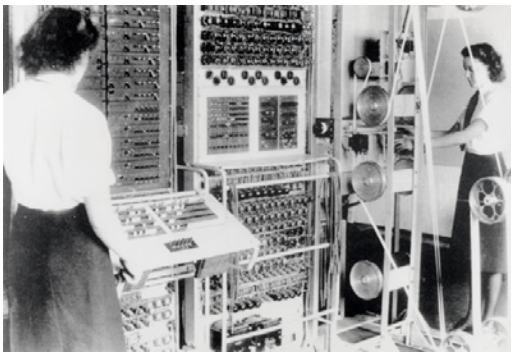
La máquina tabuladora fue creada por el inventor estadounidense Herman Hollerith en 1890. A diferencia de la máquina analítica de Babbage, pudo concretarse. Se diseñó para procesar mecánicamente los datos del censo que iba a realizarse en Estados Unidos en 1890, ya que el tabulado y análisis de resultados en forma manual del censo anterior –en 1880– había requerido más de siete años. Apenas construida, el gobierno de Estados Unidos aprobó su uso. Esta fue la primera vez que el procesamiento de información de manera mecánica reemplazó exitosamente tareas que previamente se realizaban a mano. Los datos se ingresaban con tarjetas perforadas, que la máquina recopilaba y analizaba para arrojar finalmente resultados significativos.

La máquina tabuladora de Hollerith

El trabajo posterior de Hollerith sobre la máquina permitió perfeccionarla para realizar tareas más sofisticadas, lo que dio lugar a nuevos avances en el procesamiento de datos. En sociedad con algunas empresas de la época, Hollerith constituyó la compañía Tabulating Recording Company, que más tarde pasó a llamarse International Business Machines, mundialmente conocida como IBM.

Colossus

Durante la Segunda Guerra Mundial, distintas unidades del ejército de la Alemania nazi se comunicaban mediante mensajes cifrados. Con el objetivo de descifrarlos, Gran Bretaña desarrolló una serie de computadoras denominadas Colossus. Cada computadora tenía alrededor de 1500 tubos de vacío, leía tarjetas perforadas y podía almacenar valores. Además, era capaz de imprimir la información de salida a través de una máquina de escribir. Estas computadoras ocupaban más de nueve metros cuadrados y, debido a que al encenderlas se corría el riesgo de que se dañaran las válvulas, las solían mantener prendidas.



Una computadora Colossus, usada para descifrar mensajes de la Alemania nazi

Se construyeron 10 computadoras Colossus; pero, debido a que se trataba de un secreto militar, fueron destruidas y no se informó de su existencia y funcionamiento hasta varios años después de concluida la Segunda Guerra.

EDVAC

EN 1946, el Laboratorio de Investigación Balística de Estados Unidos comenzó a desarrollar la computadora EDVAC (de las siglas del nombre en inglés Electronic Discrete Variable Automatic Computer). La EDVAC pesaba casi 8000 kg, ocupaba una habitación entera y utilizaba un sistema binario para realizar operaciones matemáticas de manera automática. Podía leer y grabar cintas magnéticas, contaba con memoria, reloj, unidad de control y unidad para operaciones aritméticas, entre otros componentes. Fue una de las primeras máquinas en las que se implementó el modelo arquitectónico propuesto por el matemático de origen austrohúngaro John von Neumann, que se sigue utilizando en la gran mayoría de las computadoras modernas.

John von Neumann junto a la EDVAC

Clementina

Clementina fue la primera computadora que llegó a Argentina para ser usada con fines académicos y científicos. Arribó al país en 1960 y se instaló en 1961 en el Instituto de Cálculo de la Facultad de Ciencias Exactas y Naturales, que funcionaba en el Pabellón I de la Ciudad Universitaria de la Universidad de Buenos Aires. Fue adquirida por una licitación pública internacional a través de las gestiones del matemático Manuel Sadosky, considerado el padre de la computación en nuestro país.

Clementina, la primera computadora con fines científicos que llegó a Argentina

La computadora, modelo Mercury y fabricada por la compañía Ferranti, fue desarrollada en Gran Bretaña. Medía 18 metros de largo. Se la utilizó en proyectos científicos y tecnológicos, y se mantuvo en funcionamiento hasta 1971, cuando las dificultades para conseguir repuestos provocaron que quedara fuera de servicio. Clementina facilitó la enseñanza de la programación: su adquisición fue fundamental para el desarrollo de la computación en el país y la región.

Commodore VIC-20

Con el objetivo de competir con las consolas de videojuegos, en 1980, la empresa Commodore comercializó un modelo de computadora para uso personal. La Commodore VIC-20 fue la primera computadora que alcanzó el millón de ventas. Por sus características técnicas, solo llegó a ejecutar *software* con propósitos educativos y videojuegos. Sin embargo, su llegada masiva al mercado doméstico posibilitó que un gran número de personas investigaran su funcionamiento y se interesaran por la programación. Con una VIC-20 se inició en la informática un joven finlandés-estadounidense llamado Linus Torvalds, quien años más tarde crearía el sistema operativo Linux.

Commodore VIC-20

En la ficha, los hechos presentados se encuentra identificados con letras de la *A* a la *J*.
La solución esperada se muestra a continuación.

COMPUTADORA	HECHOS RELACIONADOS
Máquina analítica de Babbage	A, B y F
Máquina tabuladora de Hollerith	E
Colossus	G
EDVAC	H
Clementina	D e I
Commodore VIC-20	C y J

Una vez que todos hayan completado la actividad, hacemos una puesta en común y formulamos preguntas para repasar y complementar lo investigado por los estudiantes. A continuación se muestran algunas a modo de ejemplo.

¿Quiénes se imaginan que usaban cada una de las máquinas analizadas?

Las primeras cinco computadoras –la máquina analítica de Babbage, la tabuladora de Hollerith, las Colossus, la EDVAC y Clementina– fueron utilizadas por equipos de profesionales: científicos, técnicos, matemáticos, expertos en cálculo, etc., que las usaban para resolver problemas científicos y técnicos que requerían muchos cálculos matemáticos, tales como cuánto combustible hace falta para llevar un cohete a la luna o determinar cuál es la mejor manera de establecer rutas de vuelo para aprovechar al máximo los aviones y la tripulación de una aerolínea. Si bien la Commodore VIC-20 requería ciertos conocimientos, su uso no estaba limitado a profesionales de la informática y era similar al que se les da a las computadoras de escritorio o portátiles que existen en la actualidad.

Mirando las imágenes y leyendo las descripciones, ¿qué máquinas les parece que podríamos tener en el salón de clases?

Solo la Commodore VIC-20, porque es la única con dimensiones apropiadas para un aula.

¿Alrededor de qué año fue posible acceder a una computadora para uso personal?

Alrededor de 1980. La Commodore VIC-20, que comenzó a venderse a partir de entonces, fue una de las primeras en comercializarse a gran escala.

CIERRE

Reflexionamos con los alumnos acerca de que las computadoras existen desde hace mucho tiempo. Les comentamos que las primeras –e incluso las de una década atrás– tenían mucha menos capacidad de cómputo que la que tiene actualmente cualquier teléfono inteligente. La idea de realizar cálculos en forma mecánica desveló a la humanidad durante siglos. Uno de los instrumentos más antiguos que se conocen es el ábaco, que era una herramienta para hacer cálculos aritméticos sencillos. Se cree que comenzó a usarse alrededor del año 2000 a. C.

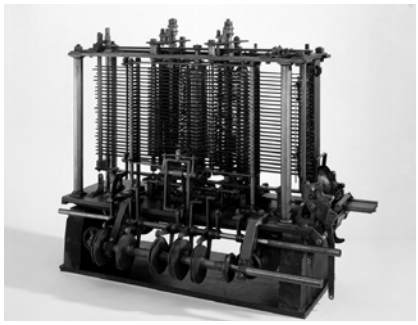
NOMBRE Y APELLIDO:

CURSO: FECHA:

UN POCO DE HISTORIA

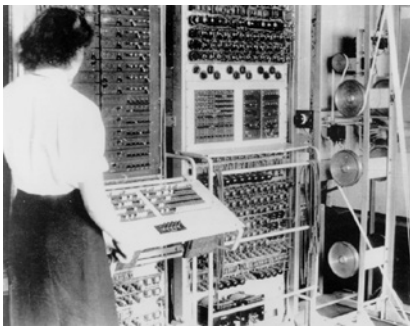
Cuando hablamos de computadoras, en general pensamos en las de escritorio y las portátiles. Sin embargo, ¡existen desde muchísimo antes! En esta actividad vamos a viajar al pasado para ver cómo eran algunas de ellas.

1. Hoy estas computadoras nos resultan de lo más extrañas. Sin embargo, cada una tuvo una gran relevancia al momento de su aparición. Miralas:



Máquina analítica de Babbage (1837)

La máquina tabuladora de Hollerith (1890)



Colossus (1944)

EDVAC (1946)

Clementina (1960)

Commodore VIC-20 (1980)

NOMBRE Y APELLIDO:

CURSO: FECHA:

A continuación enumeramos algunos hechos vinculados a una o varias de ellas.

A. Nació con el objetivo de automatizar la creación de unas tablas de números que se usaban para facilitar cálculos de funciones logarítmicas y trigonométricas que, en aquel entonces, se hacían a mano.

B. Ada Lovelace trabajó en el modo de utilizarla y creó un programa que hubiese podido funcionar en ella. Gracias a esto, hoy se la conoce como la primera persona programadora de la historia.

C. Fue la primera computadora que vendió más de un millón de unidades. Debido a su bajo poder de cómputo, se usaba principalmente para ***software*** educativo y juegos.

D. Comenzó a funcionar en enero de 1961 y siguió funcionando hasta mediados del año 1971, cuando su mantenimiento por falta de repuestos se hizo imposible.

E. Luego de la experiencia del censo de 1880 en Estados Unidos, cuyo análisis había demorado ¡siete años! en completarse, el creador de esta máquina decidió ponerse a trabajar para automatizar parte del proceso. Gracias al uso de esta nueva tecnología, el censo de 1890 se completó en tan solo seis semanas.

F. Nunca terminó de fabricarse por desacuerdos entre su diseñador, la persona que la estaba construyendo y el gobierno de Gran Bretaña, que finalmente canceló el proyecto.

G. Estas máquinas, de las que se estima que se construyeron alrededor de diez, fueron utilizadas por los británicos para descifrar mensajes que mandaba el ejército de la Alemania nazi durante la Segunda Guerra Mundial. Terminada la guerra, todas fueron destruidas por orden del entonces primer ministro del Reino Unido, Winston Churchill.

H. Fue una de las primeras computadoras con una organización de partes muy parecida a la de las máquinas que usamos hoy en día. Esta forma de organizar una computadora fue diseñada por John von Neumann y por eso hoy la llamamos arquitectura de von Neumann.

I. Introducida en el país gracias a las gestiones del Dr. Manuel Sadosky, fue la primera computadora para fines científicos y académicos en llegar a Argentina. Se instaló en el Instituto de Cálculo dependiente de la Universidad de Buenos Aires, que en aquel entonces funcionaba en el Pabellón 1 de la Ciudad Universitaria.

J. A pesar de ser una máquina con limitada capacidad de procesamiento, podía utilizarse para iniciarse en el mundo de la computación. Fue a través de ella como se interesó en la informática un joven llamado Linus Torvalds, quien luego crearía el sistema operativo Linux.

Tu tarea es relacionar estas computadoras con el o los hechos enumerados.

NOMBRE Y APELLIDO:

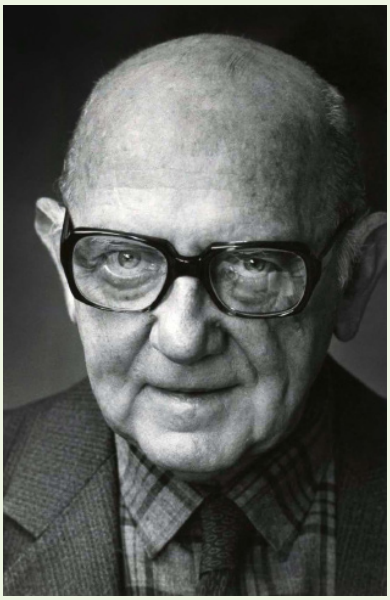
CURSO: FECHA:

COMPUTADORA	HECHOS RELACIONADOS
Máquina analítica de Babbage	
Máquina tabuladora de Hollerith	
Colossus	
EDVAC	
Clementina	
Commodore VIC-20	

MANUEL SADOSKY

Manuel Sadosky (1914-2005) fue un matemático, físico y científico de la computación, considerado por muchos el padre de la computación en Argentina. Ejerció como profesor y vicedecano de la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires en las décadas de 1950 y 1960, y como Secretario de Ciencia y Tecnología en la década de 1980. Entre sus muchas contribuciones se puede mencionar que fue el creador de la primera carrera de computación del país, el responsable de la compra de la primera computadora con fines científicos que llegó a Argentina y el creador de la Escuela Superior Latinoamericana de Informática.

A raíz de la Noche de los Bastones Largos, ocurrida en 1966 durante la dictadura de Onganía, debió exiliarse y se radicó en Uruguay. Años más tarde volvió al país, aunque debió abandonarlo nuevamente debido a que recibió amenazas de la triple A (Alianza Anticomunista Argentina). Volvió definitivamente en 1983, con el advenimiento de la democracia.



Secuencia Didáctica 2

EL *HARDWARE* Y EL *SOFTWARE*

Si bien solemos asociar la palabra *computadora* con las computadoras de escritorio y las portátiles, en la actualidad están presentes en muchos artefactos con los que interactuamos cotidianamente: desde teléfonos inteligentes hasta televisores, pasando por automóviles, semáforos, cámaras fotográficas, de video, etc. Casi cualquier máquina automática actual que realice una tarea compleja está montada sobre una computadora.

En esta secuencia didáctica se proponen actividades para deconstruir la idea clásica de computadora. Además, se explican las diferencias entre *hardware* y *software*, se describen los componentes físicos que forman una computadora y sus funciones, y se presenta una noción de programa.

OBJETIVOS

- Reconocer la presencia de las computadoras en la vida cotidiana.
- Diferenciar *hardware* y *software*.
- Describir las funciones de los principales componentes de *hardware*.
- Presentar la idea de programa.

Actividad 1
Computadoras por todas partes

DE A DOS

OBJETIVOS

- Reconocer que muchos artefactos con los que interactuamos de modo cotidiano son computadoras o las contienen.
- Mostrar que con una computadora pueden realizarse una gran cantidad de tareas.

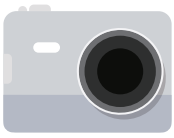
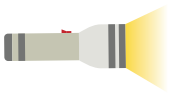
MATERIALES

- Computadora
- Internet
- Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es mostrar que, además de las computadoras de escritorio y las portátiles, muchos de los artefactos con los que interactuamos de modo cotidiano también son computadoras o las contienen. Asimismo, la finalidad de esta actividad es que los estudiantes descubran que distintas tareas que hoy pueden realizarse con una sola computadora –como, por ejemplo, las que realiza un teléfono inteligente– antes requerían de varios artefactos distintos.

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que, en parejas, resuelvan la primera consigna. Allí encontrarán una tabla en la que tienen que describir cinco tareas que realizan con teléfonos inteligentes y aclarar con qué herramientas se hacían antes de que existiera este tipo de aparatos. A continuación se dan posibles respuestas:

TAREA QUE HACÉS CON EL TELÉFONO INTELIGENTE	OTRA HERRAMIENTA PARA REALIZAR LA MISMA TAREA	
	Descripción	Foto
Sacar fotos	Una máquina fotográfica	
Escuchar música	Un equipo de audio	
Escribir un texto	Máquina de escribir	
Hablar por teléfono	Teléfono fijo	
Iluminar cuando se corta la luz	Una linterna	

Posible respuesta de los estudiantes

Una vez que hayan completado la consigna, hacemos una puesta en común. Anotamos en el pizarrón las tareas que los estudiantes hayan identificado y las analizamos con toda la clase. Si ningún estudiante menciona algunas de las propuestas que aparecen en la tabla anterior –sacar fotos, escuchar música, escribir texto, hablar por teléfono e iluminar cuando se corta la luz–, las agregamos a la lista. A continuación comentamos: "Ahora podemos realizar todas estas tareas con una sola máquina. ¿Qué sucedía antes de la existencia de los teléfonos inteligentes?". Para llevar a cabo cada una, hacía falta un artefacto específicamente diseñado con ese fin.

Continuamos: "¿Qué es, para ustedes, una computadora?". Es probable que la mayoría responda que las computadoras son las de escritorio y las portátiles. Escuchamos las respuestas y guiamos la discusión hacia la conclusión de que las computadoras son máquinas que manipulan distintos tipos de información, como por ejemplo texto, imágenes y sonidos, usando distintos programas, como reproductores de música, procesadores de texto o editores de imágenes. "Entonces, ¿les parece que un teléfono inteligente es una computadora?". Seguramente responderán que sí. Si algunos estudiantes aún tuvieran dudas, reforzamos la respuesta: "Con el teléfono sacamos fotos, leemos y escribimos –por ejemplo, al usar programas de mensajería instantánea–, y escuchamos música. Por lo tanto, manipula imágenes, texto y sonido. Además, cada actividad la hacemos con un programa distinto".

Les pedimos, entonces, que completen la segunda consigna. Allí se presenta una serie de artefactos que son computadoras o las contienen. Los estudiantes tienen que describir, en cada caso, qué función cumplen. Además, deben describir cómo se realizaban las mismas tareas cuando esos artefactos no funcionaban con una computadora. A continuación se dan ejemplos de posibles respuestas.

Lavarropas automático

La computadora que tiene incorporada permite, por ejemplo, elegir entre varios tipos de lavado: intenso, ropa blanca, prelavado, ropa de color, etc. A través de sensores, puede determinar el peso de la ropa, la temperatura, la dureza y calidad del agua, etc., y usando esa información, definir la cantidad de jabón, el nivel de agua, la agitación y el tiempo de lavado necesarios para lograr la máxima eficiencia. Previamente, la ropa se lavaba a mano.

Automóvil moderno

Puede tener varias computadoras. Por ejemplo, para regular el consumo de combustible de modo que sea óptimo y con baja emisión de gases, para diagnosticar fallas, para ayudar en el estacionamiento o para bloquear el motor en caso de accidente, entre otras. Antes, los autos eran solamente mecánicos, por lo que no podían diagnosticar fallas ni asistir al conductor en situaciones como, por ejemplo, estacionar.

Cajero automático

Una computadora ejecuta el programa que ofrece opciones de extracción, depósito, pagos, cambios de clave, etc. Además, se comunica con la computadora central del banco para informar sobre las operaciones realizadas y controla los componentes mecánicos que entregan el dinero, reciben los cheques, etc. Para efectuar las operaciones que hoy se realizan en cajeros automáticos, antes había que ir al banco donde la gente era atendida por empleados.

Finalmente, hacemos una puesta en común y discutimos entre todos las respuestas.

CIERRE

Para concluir, subrayamos que, en nuestras vidas, las computadoras están mucho más presentes de lo que solemos creer. Esta es una tendencia que creció mucho en la última década y se encuentra en plena expansión, cada vez a mayor velocidad.

NOMBRE Y APELLIDO:

CURSO: FECHA:

COMPUTADORAS POR TODOS LADOS

Las computadoras están por todos lados. ¿Qué nos permiten hacer? ¿Cómo se hacían esas cosas antes de que se usaran computadoras? En esta actividad vamos a trabajar sobre estas preguntas.

1. Completá la tabla con cinco tareas que realices con un teléfono inteligente. Además, describí con qué herramientas se hacían antes de contar con estos teléfonos e incluí una foto de esa herramienta.

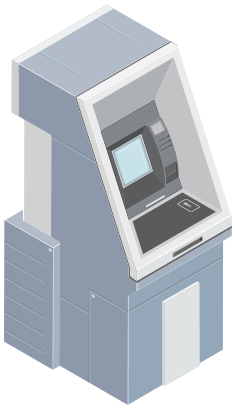
TAREA QUE HACÉS CON EL TELÉFONO INTELIGENTE	OTRA HERRAMIENTA PARA REALIZAR LA MISMA TAREA	
	Descripción	Foto

NOMBRE Y APELLIDO:

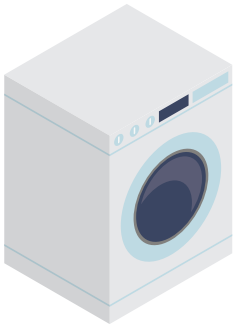
CURSO: FECHA:

2. Aunque suene raro, los artefactos mencionados a continuación tienen computadoras en su interior. Para cada uno, describí qué función cumple la computadora y cuál fue el impacto que tuvo integrarlas. Por ejemplo, ¿cómo se hacían antes esas tareas? ¿Hay algo que esos artefactos antes no podían hacer y ahora sí?

Automóvil moderno



Cajero automático



Lavarropas automático

Actividad 2
Software y *hardware*

INDIVIDUAL

OBJETIVO

- Diferenciar *hardware* y *software*.

MATERIALES

Computadora (opcional)

Internet (opcional)

Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es que los estudiantes comprendan las diferencias entre el *software* y el *hardware*, y la interacción que existe entre ambos.

Comenzamos repartiendo la ficha a los estudiantes y les indicamos que observen la imagen de la primera consigna: un teléfono inteligente que en la pantalla muestra los íconos de cuatro aplicaciones: Instagram –red social orientada a la fotografía, donde las publicaciones tienen que tener una imagen–; WhatsApp –aplicación que permite enviar mensajes y realizar llamadas a través de Internet–; Google Maps –aplicación de mapas que permite buscar una ubicación y da indicaciones para trasladarse de un punto de origen a un destino–; y YouTube –aplicación para ver videos en Internet–.

Teléfono inteligente y aplicaciones

Les preguntamos: "¿Qué representan los íconos que se ven en el pantalla?". Es esperable que surjan palabras como *programas*, *aplicaciones* o simplemente *apps*. Les pedimos, entonces, que completen la primera consigna. Tienen que contestar las preguntas que se muestran a continuación, algunas de las cuales admiten más de una respuesta.

¿Todas las aplicaciones de la imagen están en el teléfono inteligente desde el primer día?

WhatsApp no suele venir instalada en los teléfonos. En el caso de las otras tres –Instagram, YouTube y Google Maps–, según la marca y el modelo del teléfono, y la versión del sistema operativo que traigan instalada de fábrica, pueden estar o no disponibles desde el día en que se usa el teléfono por primera vez.

¿Cómo se hace para instalar las aplicaciones que no vienen en el teléfono?

Los teléfonos inteligentes modernos cuentan con una aplicación que permite entrar a una plataforma de distribución (o tienda virtual) desde donde se pueden descargar e instalar aplicaciones. En el caso de los teléfonos que usan el sistema operativo Android, esta aplicación se llama Google Play; en el caso de los que usan iOS, App Store.

¿Se pueden sacar del teléfono?

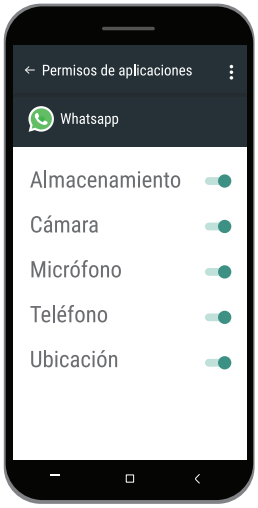
Salvo algunas excepciones, las aplicaciones instaladas pueden desinstalarse. Al usar Android, se desinstalan usando el mismo programa que para instalarlas, Google Play; en iOS, hay que mantener presionado el ícono de la aplicación que se quiere desinstalar y, luego de que aparezca la opción *borrar*, elegirla.

¿Es necesario tener instaladas estas aplicaciones para que un teléfono funcione?

No, un teléfono funciona sin problemas aun cuando los programas mencionados no se encuentren instalados. Las aplicaciones brindan la posibilidad de hacer distintas cosas, como compartir una foto en una red social, mirar videos en Internet, etc, pero no forman parte constitutiva del teléfono. Cada usuario instala aquellas que le sirven para un propósito específico y le resultan de interés.

Una vez que hayan completado la consigna, hacemos una puesta en común y discutimos entre todos las ideas que surjan. Luego, les contamos: "Todas las aplicaciones son piezas de *software*. El *software* es el conjunto de componentes lógicos que forman parte de un sistema de computación. En general, podemos pensar que es todo aquello de una computadora que es intangible. Por ejemplo, ¿podemos tocar YouTube? No, no podemos. Es el *software* el que nos permite interactuar con una computadora y acceder a las funciones que nos resulten de interés".

A continuación, les pedimos que observen la imagen del destacado "Con su permiso..." de la ficha. Proseguimos: "Lo que ven en la imagen es una pantalla de configuración de permisos de la aplicación WhatsApp. ¿Para qué son esos permisos?". Escuchamos sus respuestas y continuamos: "Se trata de opciones para conceder o denegar a la aplicación el acceso a distintos componentes del teléfono: la unidad de almacenamiento, la cámara fotográfica, el micrófono, el teléfono y la ubicación, para lo cual hace falta poder usar el sistema de posicionamiento global, habitualmente llamado GPS por sus siglas en inglés. Todos ellos son componentes de *hardware*. En contraposición al *software*, el *hardware* de una computadora es el conjunto de todos los componentes físicos. Es decir, aquellos que podemos tocar. En el caso de algunos componentes como el GPS, para tocarlos habría que desarmar previamente el teléfono".



Pantalla de configuración de permisos

Les pedimos que trabajen sobre la segunda consigna. Allí se presentan preguntas acerca de algunas diferencias entre el *software* y el *hardware*. Los invitamos a que respondan de acuerdo con sus conocimientos previos y, si hace falta, que hagan consultas en Internet. A continuación, se muestran posibles respuestas:

¿Quiénes crean *software*? ¿Qué hace falta para hacerlo?

Toda aplicación surge a partir de que la piensa y la programa alguien, ya sea un individuo, una empresa, el Estado u otro tipo de organización. Cualquier persona puede aprender a programar.

¿Cualquiera puede crear *hardware*? ¿Qué hace falta para hacerlo?

Por tratarse de componentes físicos, para fabricarlos hace falta equipamiento especial y mucho conocimiento de ingeniería. Suelen elaborarlos empresas especializadas.

¿Todos los programas son gratuitos?

Hay aplicaciones que se distribuyen de forma gratuita y otras que no. Algunas tienen una versión paga y una libre, pero esta última suele proveer acceso solo a una parte de la funcionalidad de la aplicación, además de incluir publicidades que no se muestran si la aplicación se compra.

¿Hay *hardware* gratuito?

Al ser elementos con costos de fabricación, por ejemplo, el que se deriva de poner en marcha una maquinaria, comprar insumos, etc. y costos de distribución, los componentes de *hardware* en su gran mayoría son pagos.

¿Es complicado instalar programas?

En general, es sencillo instalar y probar un *software* nuevo, y existe la posibilidad de prescindir de él posteriormente si no se trata de lo esperado.

¿Es fácil armar un teléfono inteligente?

En contraposición a la instalación de aplicaciones, para ensamblar y configurar el *hardware* de un teléfono –y, en general, de cualquier tipo de computadora– se requiere un conocimiento de electrónica muy específico, por lo que no resulta posible para la gran mayoría de las personas.

CIERRE

Reflexionamos junto con los estudiantes acerca de que una computadora requiere tanto de componentes de *hardware* como de *software*. En general, el *software* actúa como intermediario entre los componentes físicos y los usuarios, pues permite que estos últimos accedan a distintas funcionalidades para realizar tareas, como sacar fotos, escribir mensajes, etc.

NOMBRE Y APELLIDO:

CURSO: FECHA:

SOFTWARE Y HARDWARE

¿De qué hablamos cuando hablamos de *software* y *hardware*? En esta actividad vamos a comenzar a investigarlo.

1. Mirá el teléfono inteligente y contestá las preguntas.

a. ¿Todas las aplicaciones de la imagen están en el teléfono desde el primer día?

b. ¿Cómo se hace para instalar las que no vienen en el teléfono?

c. ¿Se pueden sacar del teléfono?

EL *SOFTWARE*

El *software* es el conjunto de componentes lógicos que forman parte de un sistema de computación. En general, podemos pensar que es todo aquello de una computadora que es intangible. Por ejemplo, las aplicaciones de los teléfonos son componentes de *software*.

NOMBRE Y APELLIDO:

CURSO: FECHA:

d. ¿Es necesario tener instaladas estas aplicaciones para que un teléfono funcione?

2. Respondé las preguntas y analizá algunas diferencias entre el ***software*** y el ***hardware***.

a. ¿Quiénes crean ***software***? ¿Qué hace falta para hacerlo?

b. ¿Cualquiera puede crear ***hardware***? ¿Qué hace falta para hacerlo?

c. ¿Todos los programas son gratuitos?

EL *HARDWARE*

El *hardware* de una computadora es el conjunto de todos sus componentes físicos. Es decir, aquellos que podemos tocar. Algunos ejemplos son las pantallas, las cámaras, los micrófonos, etc.

NOMBRE Y APELLIDO:

CURSO: FECHA:

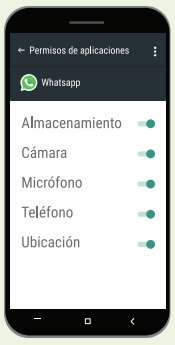
d. ¿Hay *hardware* gratuito?

e. ¿Es complicado instalar programas?

f. ¿Es fácil armar un teléfono inteligente?

CON SU PERMISO...

Muchas aplicaciones requieren permisos para poder acceder a distintos componentes de una computadora. Por ejemplo a la cámara de fotos de un teléfono inteligente, a nuestra ubicación, etc. Si algo nos genera dudas, ¡siempre podemos decir que no!



Actividad 3
¿Qué son todos esos cables?

DE A DOS

OBJETIVOS

- Identificar los componentes de *hardware* de una computadora.
- Comprender la función de cada componente de *hardware*.

MATERIALES

- Computadora
- Internet
- Ficha para estudiantes

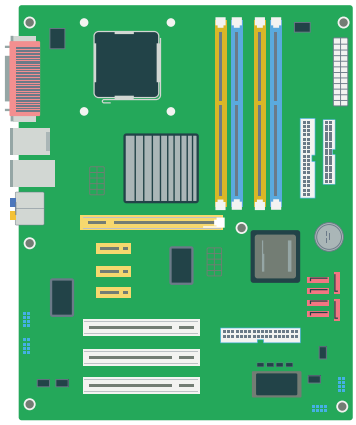
DESARROLLO

El objetivo de esta actividad es que los estudiantes identifiquen el aspecto de los distintos componentes de *hardware* de una computadora y comprendan las funciones que cumplen. A continuación, se hace una breve descripción de cada componente abordado en la actividad, y luego se desarrolla la puesta en práctica de la actividad en el aula.

DESCRIPCIÓN DE COMPONENTES

Placa madre

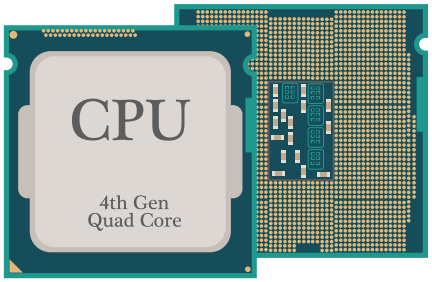
La placa madre o *motherboard* es una placa a la que se conectan los componentes principales de una computadora, como la unidad central de procesamiento y la memoria. Además, posee ranuras para incorporar placas de extensión a las que se conectan otros dispositivos, como monitores, teclados, impresoras, puertos de red, etc. Tiene circuitos impresos que permiten la comunicación entre los componentes; por ejemplo, cuando el procesador requiere comunicarse con la memoria principal o con los dispositivos de entrada y salida. También contiene conectores para la alimentación de energía eléctrica, un reloj interno y otros componentes indispensables para el funcionamiento de la computadora y la comunicación entre sus partes.



Placa madre

Unidad central de procesamiento

La unidad central de procesamiento, también llamada procesador o CPU (por las siglas en inglés de Central Processing Unit), es el componente que se encarga de ejecutar una por una las instrucciones de un programa realizando operaciones aritméticas y lógicas. Se coloca en un lugar especial de la placa madre y requiere refrigeración para disipar el calor que produce durante su funcionamiento.



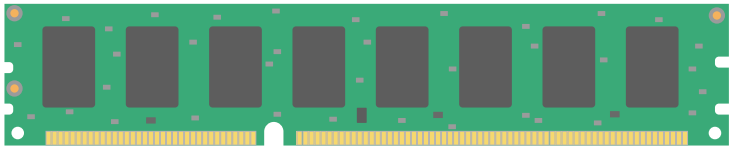
Unidad central de procesamiento

La CPU controla el resto de los componentes de la computadora y dirige el flujo de datos entre ellos. Por ejemplo, lee información del disco rígido y la carga en la memoria principal (o memoria RAM) para procesarla, y envía el resultado a un dispositivo de salida –como un monitor o una impresora– para mostrarlo. Además, tiene una pequeña memoria interna, denominada *caché*, a la que puede acceder muy rápidamente. Allí mantiene una copia de la porción de la memoria RAM que utiliza con mayor frecuencia y, de este modo, acelera su velocidad de trabajo.

En la actualidad, se fabrica como un único circuito integrado o microchip y, generalmente, contiene varias unidades de procesamiento con las que lleva a cabo en paralelo el procesamiento de información.

Memoria principal

La memoria principal o RAM (por la sigla en inglés de Random Access Memory) es el componente físico en el que se almacenan los programas y los datos que usa el procesador para realizar cómputos. Para funcionar necesita energía, por lo que no preserva su contenido cuando la computadora se apaga. Por tal motivo, se dice que es una memoria volátil. Físicamente, suele presentarse en forma de placas o módulos que se insertan en ranuras de la placa madre destinadas especialmente para este fin.



Módulo de memoria RAM

Si bien la memoria está organizada como una tira de celdas, una a continuación de la otra, se puede acceder a todas las posiciones directamente y en un tiempo constante (en contraposición a lo que sería un acceso secuencial, en el que para alcanzar una posición hay que comenzar por el principio y avanzar de a una celda hasta llegar a la posición buscada). Por tal motivo, se la denomina *memoria de acceso aleatorio*.

Unidades de almacenamiento

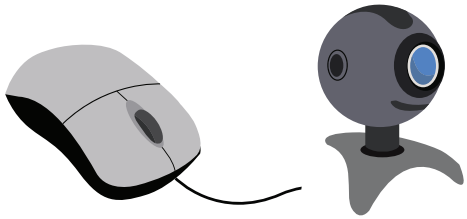
Las unidades de almacenamiento son los componentes en los que se puede almacenar información en forma persistente. Es decir, conservan la información aun cuando no posean suministro de energía –a diferencia de la memoria RAM, que es volátil–. La velocidad de acceso a estos dispositivos es mucho menor que la de acceso a la RAM. Sin embargo, permiten almacenar un volumen de información mucho mayor. Estos dispositivos se conectan a la placa madre a través de placas de expansión que contienen los enchufes necesarios para incorporarlos a la computadora.

Un disco rígido y una memoria USB

Los discos externos, las memorias USB y las tarjetas SD son ejemplos de este tipo de componentes que utilizamos con frecuencia. También son unidades de almacenamiento los CD, DVD, Blu-ray y los discos rígidos. Antes se utilizaban cintas magnéticas, casetes y discos flexibles.

Dispositivos de entrada

Los dispositivos de entrada permiten que la computadora reciba información. Entre ellos, están los que sirven para que los usuarios ingresen datos y, de esta manera, controlen el funcionamiento de las computadoras. Algunos ejemplos son el teclado, el ratón, las pantallas táctiles y los lectores de códigos de barras. Hay otros que, en algunas circunstancias, funcionan sin intervención humana, como los sensores de distintos tipos –de temperatura, de proximidad, etc.–, las cámaras digitales o los módems –que permiten que la computadora reciba información de Internet–.



Un ratón y una cámara web

Dispositivos de salida

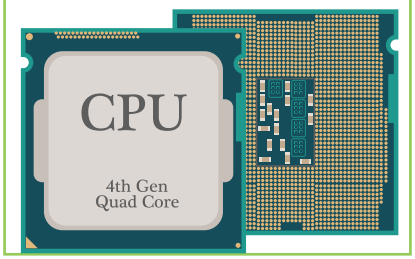
Los dispositivos de salida son aquellos que usa la computadora para comunicar al exterior los resultados de un procesamiento; por ejemplo, una impresora, un monitor, parlantes, auriculares, proyectores, etc.

Un monitor y un par de parlantes

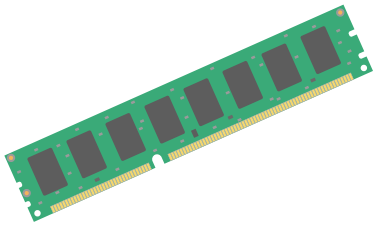
Comenzamos repartiendo la ficha a los estudiantes y les pedimos que, en parejas, resuelvan la primera consigna. Allí se les pide que busquen en Internet imágenes de distintos componentes de *hardware*: una unidad central de procesamiento o CPU, una memoria principal RAM, una placa madre, tres unidades de almacenamiento distintas, tres dispositivos de entrada y tres dispositivos de salida. De ser posible, les pedimos que las impriman y las peguen en el espacio de la ficha reservado para tal propósito.[1]

[1] En caso de que no se pueda imprimir en la escuela, esta primera consigna puede darse como tarea y retomar la actividad en la clase siguiente. También puede obviarse la impresión.

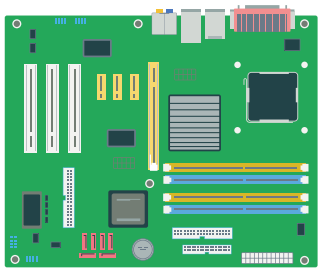
Unidad central de procesamiento (CPU)



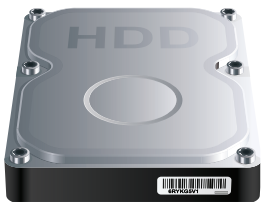
Memoria principal (RAM)



Placa madre (*motherboard*)

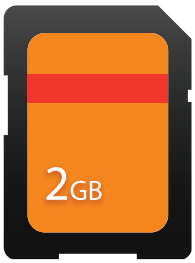


Unidades de almacenamiento



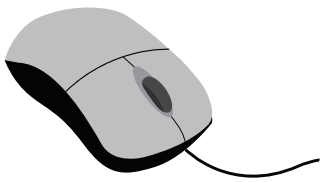
Disco rígido

Memoria USB

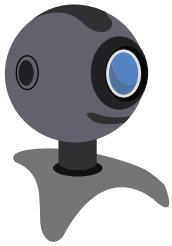


Memoria SD

Dispositivos de entrada



Ratón



Cámara web

teclado

Dispositivos de salida

Impresora

Monitor

Parlantes

Posible solución de la primera consigna

Una vez que todos hayan encontrado imágenes y observado el aspecto de los distintos componentes, les indicamos que resuelvan la segunda consigna. Se presentan una serie de características de los componentes mencionados, aunque no se indica a qué componente corresponde cada una. La tarea consiste en relacionar cada característica con un componente. Los estudiantes, de ser necesario, pueden buscar información en Internet. A continuación se da la solución esperada:

También se la llama *procesador*. Controla al resto de los componentes de la computadora.

- [x] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Los lectores de código de barras y los micrófonos son ejemplos de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [x] Dispositivo de entrada
- [ ] Dispositivo de salida

Es una placa a la que se conectan los componentes principales de una computadora. Tiene circuitos impresos que permiten la comunicación entre ellos.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [x] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Las memorias USB y las tarjetas SD son formas modernas de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [x] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Es el componente de *hardware* que se encarga de ejecutar las instrucciones de los programas. Para hacerlo, realiza operaciones aritméticas y lógicas.

- [x] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Se utiliza para almacenar información. No puede funcionar sin energía, y por lo tanto no conserva el contenido al apagar la computadora. Por tal motivo, se dice que es volátil.

- [ ] Unidad central de procesamiento
- [x] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Permite que la computadora reciba información. Con varios de ellos, un usuario puede controlar el funcionamiento de los programas y de la computadora en general.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [x] Dispositivo de entrada
- [ ] Dispositivo de salida

Los parlantes y auriculares son ejemplos de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [x] Dispositivo de salida

Tiene una pequeña memoria interna, denominada *caché*, a la que puede acceder muy rápidamente. Allí mantiene una copia de la porción de la RAM que utiliza con mayor frecuencia y de este modo acelera su velocidad de trabajo.

- [x] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

El más usado de este tipo de componentes es el monitor.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [x] Dispositivo de salida

Permite la comunicación entre el procesador y la memoria RAM, y entre el procesador y los dispositivos de entrada y salida.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [x] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Componente utilizado para leer, grabar o guardar datos, que no necesita estar encendido ni recibir un suministro permanente de energía para conservar la información.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [x] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Los teclados y los ratones son ejemplos de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [x] Dispositivo de entrada
- [ ] Dispositivo de salida

Se puede acceder a este componente directamente y en un tiempo constante (en contraposición a lo que sería un acceso secuencial, en el que para alcanzar una posición hay que comenzar por el principio e ir avanzando de a una hasta llegar a la posición buscada). Por tal motivo, se dice que es de acceso aleatorio.

- [ ] Unidad central de procesamiento
- [x] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Permite que la computadora comunique al exterior los resultados de un procesamiento.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [x] Dispositivo de salida

Aunque han quedado en desuso, las cintas magnéticas y los discos flexibles (o *floppy disks*) son componentes de este tipo que fueron muy populares cuando las computadoras empezaron a difundirse masivamente.

- [ ] Unidad central de procesamiento
- [x] Unidad de almacenamiento
- [ ] Memoria principal
- [ ] Dispositivo de entrada
- [ ] Placa madre
- [ ] Dispositivo de salida

También se la conoce como *motherboard*.

- [ ] Unidad central de procesamiento
- [ ] Unidad de almacenamiento
- [ ] Memoria principal
- [ ] Dispositivo de entrada
- [x] Placa madre
- [ ] Dispositivo de salida

Su función principal es guardar grandes volúmenes de información. La velocidad a la que se puede leer o escribir esta información suele ser mucho menor que la de la memoria principal.

- [ ] Unidad central de procesamiento
- [x] Unidad de almacenamiento
- [ ] Memoria principal
- [ ] Dispositivo de entrada
- [ ] Placa madre
- [ ] Dispositivo de salida

Una pantalla táctil es un ejemplo de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Unidad de almacenamiento
- [ ] Memoria principal
- [x] Dispositivo de entrada
- [ ] Placa madre
- [ ] Dispositivo de salida

Además de zócalos para la CPU, la memoria RAM y las ranuras para placas de expansión, contiene conectores para la fuente de alimentación de energía, un reloj y otras partes importantes para el funcionamiento de la computadora y la comunicación de los componentes.

- [ ] Unidad central de procesamiento
- [ ] Unidad de almacenamiento
- [ ] Memoria principal
- [ ] Dispositivo de entrada
- [x] Placa madre
- [ ] Dispositivo de salida

Ejemplos típicos de este tipo de componentes son los discos rígidos y las unidades de CD, DVD o Blu-ray.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [x] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Carga información en la memoria RAM para su posterior procesamiento.

- [x] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Las impresoras son un ejemplo de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [x] Dispositivo de salida

Suele presentarse en forma de placas –también llamadas *módulos*– y se inserta en unas ranuras de la placa madre. Su función es almacenar los datos y programas que usa el procesador para realizar cómputos.

- [ ] Unidad central de procesamiento
- [x] Memoria principal
- [ ] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

Suele contener ranuras que permiten incorporar placas de expansión. Mediante estas placas es posible conectar a la computadora diferentes dispositivos, como monitores, teclados, impresoras, etc., como así también conectarse a Internet.

- [ ] Unidad central de procesamiento
- [ ] Memoria principal
- [x] Placa madre
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de entrada
- [ ] Dispositivo de salida

En la actualidad, se fabrica como un único circuito integrado o microchip y, generalmente, contiene varias unidades de procesamiento con las que lleva a cabo en paralelo el procesamiento de información.

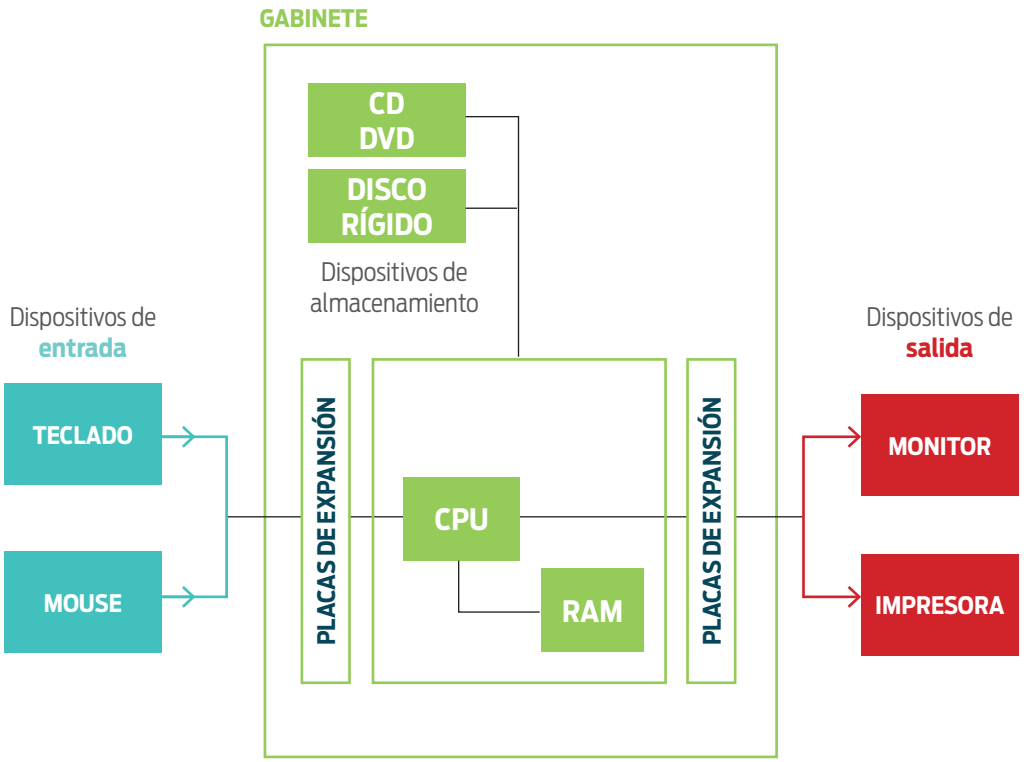
[x] Unidad central de procesamiento	[ ] Unidad de almacenamiento
[ ] Memoria principal	[ ] Dispositivo de entrada
[ ] Placa madre	[ ] Dispositivo de salida

Una vez que todos hayan completado la consigna, hacemos una puesta en común y repasamos aquellas referencias que hayan generado dudas. Hacemos notar que las pantallas táctiles son tanto dispositivos de entrada –ya que permiten ingresar información– como dispositivos de salida –pues permiten que la computadora muestre información a los usuarios–. A los dispositivos que permiten tanto el ingreso de información como su salida se los conoce como *dispositivos de entrada y salida*.

Es pertinente diferenciar también las unidades de almacenamiento de los dispositivos de entrada que permiten el ingreso de la información que contienen a la computadora. Por ejemplo, mientras que los CD, DVD y Blu-ray son unidades de almacenamiento, las lectoras de CD, DVD y Blu-ray son dispositivos de entrada. Del mismo modo, las lectoras y grabadoras de estas unidades de almacenamiento son dispositivos de entrada y salida.

Les comentamos a los estudiantes: "Todos los componentes se conectan a la placa madre: la CPU y la memoria RAM de manera directa, y los demás componentes, a través de placas de expansión –como las placas de video, de sonido, puertos USB, etc.–. La CPU es la encargada de controlar las acciones de los demás componentes: carga información en la memoria principal usando dispositivos de entrada, la procesa ejecutando de a una las instrucciones de un programa y produce información que comunica al exterior mediante dispositivos de salida. La placa madre, la CPU, la RAM y las placas de expansión, además de cables y otros componentes como la fuente de energía y los *coolers* –ventiladores pequeños para disipar el calor que produce la computadora cuando está funcionando– se encuentran todos juntos dentro de un gabinete".

Continuamos: "Toda la información circula a través de los cables y circuitos que relacionan los componentes, y son estas relaciones las que permiten que la computadora cumpla sus funciones: guardar un archivo en una memoria USB, proyectar una imagen en la pantalla, ejecutar algún juego, reproducir música por los parlantes, etc. Esto mismo ocurre con otro tipo de computadoras, como los teléfonos inteligentes, que incluyen esquemas similares para organizar sus componentes pero de manera más compacta. Los programas, es decir el *software*, son los que nos permiten vincular esos componentes físicos entre sí y utilizarlos para realizar las tareas que mencionamos. La unión de *software* y *hardware* y todas las relaciones que los vinculan son los aspectos constitutivos de una computadora".



Organización de una computadora

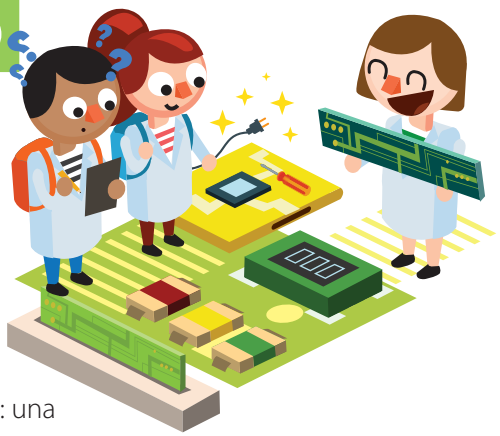
CIERRE

Cerramos la actividad contándoles a los estudiantes que, en 1945, el matemático de origen austrohúngaro John von Neumann presentó un modelo teórico de computadora que fue la base de las computadoras modernas. Von Neumann proponía como partes constitutivas de una computadora una unidad central de procesamiento, una memoria y dispositivos de entrada y de salida. En su honor, este diseño de computadoras recibe el nombre de arquitectura de Von Neumann.

NOMBRE Y APELLIDO:

CURSO: FECHA:

¿QUÉ SON TODOS ESOS CABLES?



¿Alguna vez desarmaste una computadora? ¿Sabés qué son y para qué sirven los componentes que contiene? Vamos a ver de qué se trata por dentro este artefacto.

1. Buscá en Internet imágenes de estos componentes de *hardware*: una unidad central de procesamiento –o CPU, por sus siglas en inglés–, una memoria principal -o memoria RAM-, una placa madre, tres unidades de almacenamiento distintas, tres dispositivos de entrada y tres dispositivos de salida. Una vez que los tengas, pegalos en la tabla.

Unidad central de procesamiento (CPU)	Memoria principal (RAM)	Placa madre (*motherboard*)

Unidades de almacenamiento		

Dispositivos de entrada		

NOMBRE Y APELLIDO:

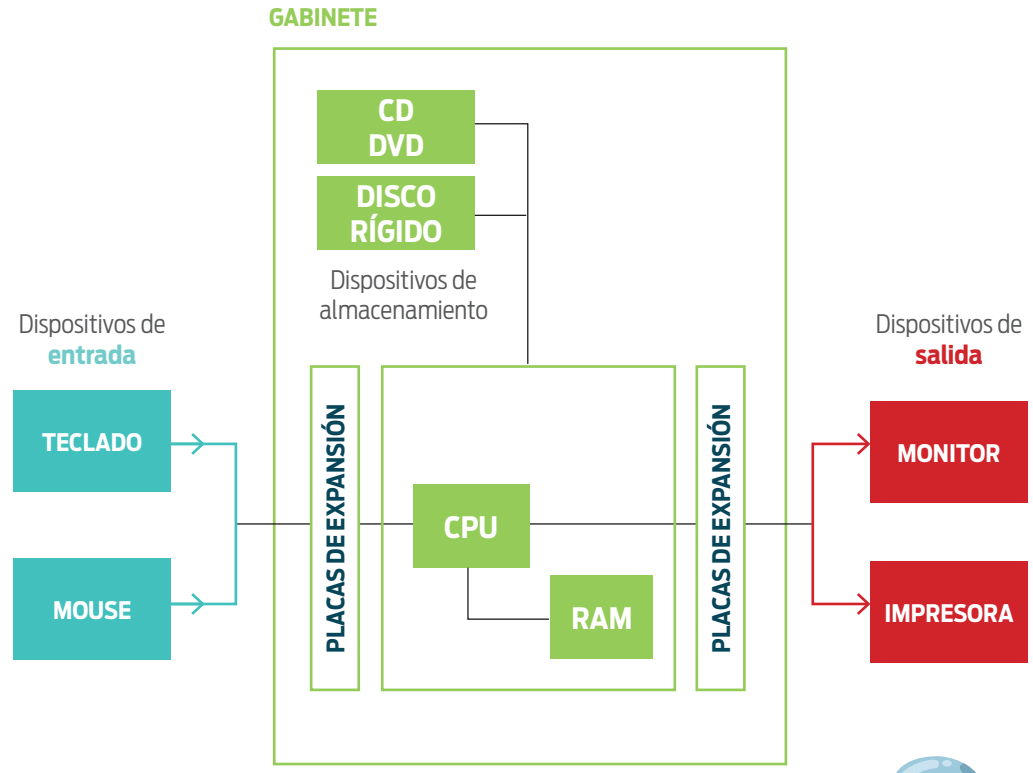
CURSO:

FECHA:

Dispositivos de salida

¿QUÉ HAY AHÍ ADENTRO?

Todos los componentes de la computadora se conectan a la placa madre: la CPU y la memoria RAM de manera directa, y los demás componentes a través de placas de expansión, como las placas de video, de sonido, puertos USB, etc. La CPU es la encargada de controlar las acciones de los demás componentes: carga información en la memoria principal, ya sea desde una unidad de almacenamiento o usando dispositivos de entrada, la procesa ejecutando las instrucciones de un programa una por una, y produce nueva información que guarda en un dispositivo de almacenamiento o comunica al exterior mediante dispositivos de salida.



NOMBRE Y APELLIDO:

CURSO: FECHA:

2. Cada componente tiene características propias. Indicá a qué componente corresponde cada característica. ¡Podés buscar las respuestas en Internet!

También se la llama *procesador*. Controla al resto de los componentes de la computadora.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

Los lectores de código de barras y los micrófonos son ejemplos de este tipo de componentes.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

Es una placa a la que se conectan los componentes principales de una computadora.
Tiene circuitos impresos que permiten la comunicación entre ellos.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

Las memorias USB y las tarjetas SD son formas modernas de este tipo de componentes.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

Es el componente de *hardware* que se encarga de ejecutar las instrucciones de los programas.
Para hacerlo, realiza operaciones aritméticas y lógicas.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

Se utiliza para almacenar información. No puede funcionar sin energía, y por lo tanto no conserva el contenido al apagar la computadora. Por tal motivo, se dice que es volátil.

☐ Unidad central de procesamiento ☐ Placa madre ☐ Dispositivo de entrada
☐ Memoria principal ☐ Unidad de almacenamiento ☐ Dispositivo de salida

NOMBRE Y APELLIDO:

CURSO: FECHA:

Permite que la computadora reciba información. Con varios de ellos, un usuario puede controlar el funcionamiento de los programas y de la computadora en general.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Los parlantes y auriculares son ejemplos de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Tiene una pequeña memoria interna, denominada *caché*, a la que puede acceder muy rápidamente. Allí mantiene una copia de la porción de la RAM que utiliza con mayor frecuencia y de este modo acelera su velocidad de trabajo.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

El más usado de este tipo de componentes es el monitor.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Permite la comunicación entre el procesador y la memoria RAM, y entre el procesador y los dispositivos de entrada y salida.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Componente utilizado para leer, grabar o guardar datos, que no necesita estar encendido ni recibir un suministro permanente de energía para conservar la información.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

NOMBRE Y APELLIDO:

CURSO: FECHA:

Los teclados y los ratones son ejemplos de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Se puede acceder a este componente directamente y en un tiempo constante (en contraposición a lo que sería un acceso secuencial, en el que para alcanzar una posición hay que comenzar por el principio e ir avanzando de a una hasta llegar a la posición buscada). Por tal motivo, se dice que es de acceso aleatorio.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Permite que la computadora comunique al exterior los resultados de un procesamiento.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Aunque han quedado en desuso, las cintas magnéticas y los discos flexibles (o *floppy disks*) son componentes de este tipo que fueron muy populares cuando las computadoras empezaron a difundirse masivamente.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

También se la conoce como *motherboard*.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Su función principal es guardar grandes volúmenes de información. La velocidad a la que se puede leer o escribir esta información suele ser mucho menor que la de la memoria principal.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

NOMBRE Y APELLIDO:

CURSO: FECHA:

Una pantalla táctil es un ejemplo de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Además de zócalos para la CPU, la memoria RAM y las ranuras para placas de expansión, contiene conectores para la fuente de alimentación de energía, un reloj y otras partes importantes para el funcionamiento de la computadora y la comunicación de los componentes.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Ejemplos típicos de este tipo de componentes son los discos rígidos y los CD, DVD y Blu-ray.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Carga información en la memoria RAM para su posterior procesamiento.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Las impresoras son un ejemplo de este tipo de componentes.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

Suele presentarse en forma de placas –también llamadas *módulos*– y se inserta en unas ranuras de la placa madre. Su función es almacenar los datos y programas que usa el procesador para realizar cómputos.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

NOMBRE Y APELLIDO:

CURSO: FECHA:

Suele contener ranuras que permiten incorporar placas de expansión. Mediante estas placas es posible conectar a la computadora diferentes dispositivos, como monitores, teclados, impresoras, etc., como así también conectarse a Internet.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

En la actualidad, se fabrica como un único circuito integrado o microchip y, generalmente, contiene varias unidades de procesamiento con las que lleva a cabo en paralelo el procesamiento de información.

- [ ] Unidad central de procesamiento
- [ ] Placa madre
- [ ] Dispositivo de entrada
- [ ] Memoria principal
- [ ] Unidad de almacenamiento
- [ ] Dispositivo de salida

JOHN VON NEUMANN

En 1945, el matemático de origen austrohúngaro John von Neumann presentó un modelo teórico de computadora que fue la base de las computadoras modernas. Von Neumann proponía como partes constitutivas de una computadora una unidad central de procesamiento, una memoria y dispositivos de entrada y de salida. En su honor, este diseño de computadoras recibe el nombre de arquitectura de Von Neumann.

Actividad 4
¡Saquemos al robot del aula!

GRUPAL (4)

OBJETIVOS

- Identificar qué es un programa.
- Comprender que un autómata únicamente ejecuta instrucciones predefinidas.
- Diferenciar las instancias de armado y ejecución de un programa.

MATERIALES

- Pizarrón
- Tizas

DESARROLLO

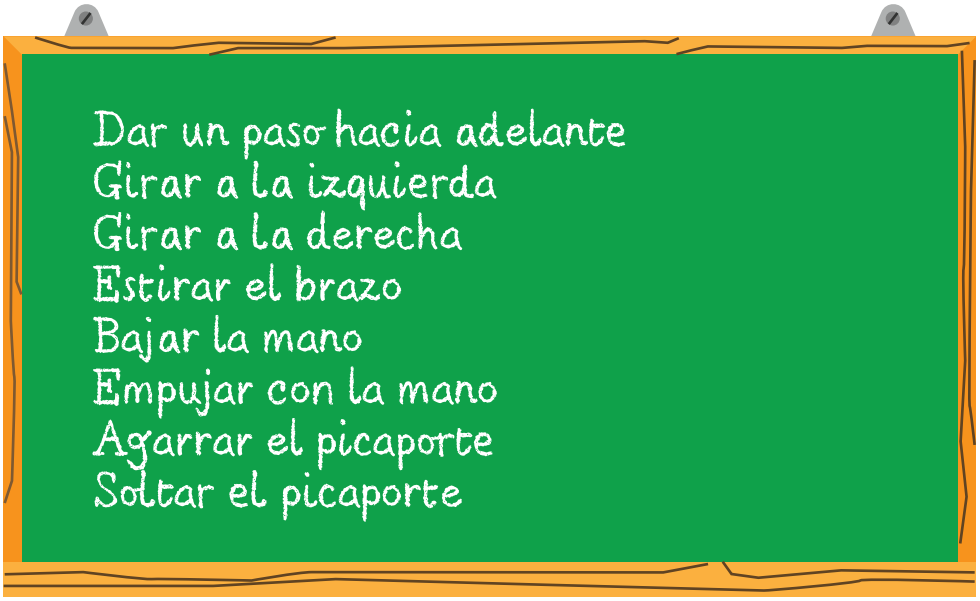
En esta actividad vamos a trabajar sobre **programas**. Un programa es una descripción del comportamiento que se espera que siga una computadora y tiene como objetivo solucionar algún problema. Puede ser creado, leído e interpretado por personas que, de esta manera, son capaces de comprender su funcionamiento, su objetivo y la propuesta de solución específica que intenta brindar.

Un **autómata** es una máquina que se puede programar para realizar ciertas acciones de manera automática. Las computadoras y los robots son ejemplos de autómatas. Las instrucciones que puede ejecutar un autómata se denominan *comandos*. Cada autómata está provisto de determinados comandos, con una sintaxis rígida, y no reconoce otros diferentes. Los programas se construyen combinando de diversas maneras los comandos para hacer funcionar un autómata.

Para comenzar, elegimos a algún miembro de la clase –podemos ser nosotros o un estudiante– para que interprete a un robot. El robot tendrá que ejecutar los comandos que se le indiquen y no podrá tomar decisiones. Solo reconocerá los siguientes: `Dar un paso hacia adelante`, `Girar a la izquierda`, `Girar a la derecha`, `Estirar el brazo`, `Bajar la mano`, `Empujar con la mano`, `Agarrar el picaporte`, `Soltar el picaporte`.

Les pedimos a los estudiantes que comiencen a darle instrucciones al robot para que descubran cuáles reconoce. Para guiarlos, les comentamos que los comandos, en general, son acciones concretas, como por ejemplo saltar, mirar hacia arriba o tomar medio vaso de agua. A medida que vayan diciendo los comandos tal como los reconoce el robot, los vamos anotando en el pizarrón. Además, quien interprete al robot debe realizar la acción indicada. Por ejemplo, si algún estudiante dice `Dar un paso hacia adelante`, el robot lo hará o al menos lo intentará, incluso si hay un obstáculo delante o se encuentra frente a una pared. De la misma forma, si le indican `Agarrar el picaporte` o `Soltar el picaporte` cuando se encuentra lejos de la puerta, el robot hará la mímica como si tuviera el picaporte adelante. Buscamos, de este modo, que reconozcan que los autómatas siguen instrucciones **al pie de la letra**, sin cuestionarlas ni reflexionar sobre ellas. Ante cualquier instrucción que no sea uno de los comandos que reconoce, debe contestar: "No entiendo ese comando", aun cuando el sentido sea equivalente a uno que sí comprende –por ejemplo, si alguien dijera: "Caminar un paso hacia adelante" en lugar de "Dar un paso hacia adelante"–.

A medida que se mencionan, escribimos los comandos en el pizarrón.



Si luego de un tiempo no se han descubierto todos los comandos, les sugerimos a los estudiantes que le pregunten al robot cuáles son los que faltan. El robot responderá. Cuando la lista de comandos válidos esté completa, le pedimos al robot que se ubique en un lugar alejado de la puerta del aula. Organizamos la clase en grupos de cuatro estudiantes y les pedimos que escriban en un papel una serie de comandos para que el robot salga del salón. Es decir, que escriban un programa para alcanzar un objetivo.

Una vez que los grupos hayan escrito sus programas, el robot debe ejecutarlos partiendo siempre desde el mismo punto de inicio. De este modo, los estudiantes podrán verificar si el programa construido sirve para alcanzar el objetivo propuesto. Si ningún programa logra que el robot salga del aula en el primer intento, los grupos deberán escribir uno nuevo. Es importante aclarar que los programas se arman antes de ejecutarlos y, salvo algunos sistemas específicamente diseñados para eso, no se pueden modificar comandos durante su ejecución. Para modificar un programa debe interrumpirse la ejecución, cambiarlo y volver a probarlo.

Una solución posible para la ubicación del robot y de la puerta que muestra la figura podría ser la siguiente:

```
Girar a la izquierda
Dar un paso hacia adelante
Dar un paso hacia adelante
Dar un paso hacia adelante
Girar a la derecha
Dar un paso hacia adelante
Dar un paso hacia adelante
Girar a la izquierda
Dar un paso hacia adelante
Estirar el brazo
Agarrar el picaporte
Bajar la mano
Empujar con la mano
Soltar el picaporte
```

Ir hasta la puerta

Abrir la puerta

Salir

Una vez verificadas las soluciones, copiamos en el pizarrón una solución correcta e indicamos al costado las partes del programa que resuelven subproblemas: las instrucciones que se usan para dirigir el robot hacia la puerta, las que se utilizan para abrir la puerta y la que se utiliza para salir. De esta manera, quedan bien diferenciados los comandos que sirven para un mismo propósito y podemos mostrar a la clase que es mucho más claro entender el objetivo del programa si agrupamos los comandos que tienen un mismo fin y les ponemos un nombre descriptivo.

CIERRE

Comentamos que lo que hicimos fue realizar una simulación en la que escribimos un programa (*software*) para guiar el comportamiento de un robot o autómata (computadora) dándole instrucciones respecto de cómo moverse utilizando sus partes como piernas, brazos y manos (*hardware*).

02

PROGRAMAS Y COMANDOS BÁSICOS

¿Cómo se hace para que un cajero automático le entregue el dinero al cliente que lo solicita? ¿O cómo se hacen los juegos para teléfonos inteligentes? En ambos casos la respuesta es la misma: programando.

En este capítulo se realiza una primera aproximación a la programación. Primero, se muestran algunos programas en acción y se busca que los estudiantes reconozcan los elementos y las operaciones disponibles para trabajar. Además, se presenta el lenguaje Gobstones y su entorno. Luego, se muestran sus elementos: el tablero, las bolitas, el cabezal y los comandos con los que se los manipula, mientras se construyen los primeros programas.

Secuencia Didáctica 1

PROGRAMAS

Los programas sirven para automatizar las soluciones de algunos problemas. Por lo general, estos problemas se modelan en términos de **elementos** y **operaciones**, que en conjunto se conocen como **dominio del problema**. Hablamos, entonces, de elementos y operaciones del dominio.

Cualquier máquina programable tiene su propio conjunto de elementos y operaciones. Aprender un lenguaje de programación es, en parte, conocer este conjunto. Y, luego, usarlo para representar el dominio de algún problema que queramos resolver.

En esta secuencia didáctica empezaremos a identificar los elementos y las operaciones de algunos problemas.

OBJETIVOS

- Reconocer los elementos y las operaciones presentes en un problema.
- Identificar similitudes entre problemas pertenecientes a diferentes dominios.

Actividad 1
Repartija de caramelos

TODA LA CLASE

OBJETIVOS

- Señalar que una máquina programable solo reconoce un pequeño conjunto de comandos.
- Identificar los elementos y las operaciones de un problema.
- Mostrar que ciertas dinámicas pueden simularse en entornos virtuales.

MATERIALES

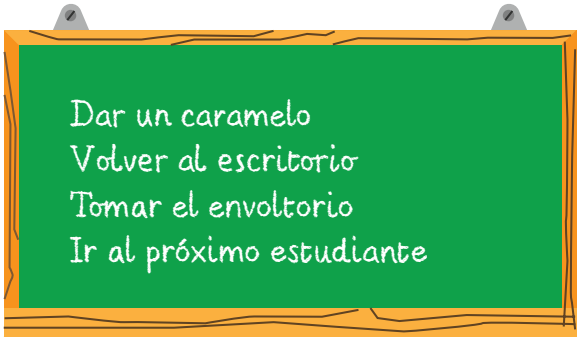
- Bolsa con caramelos envueltos en papel
- Computadora
- Gobstones
- Proyector (opcional)

DESARROLLO

En esta actividad comenzamos jugando un juego de rol. Este nos permitirá mostrar que una máquina programable reconoce únicamente un pequeño conjunto de comandos que interpreta de una sola forma. Una vez finalizado el juego, veremos un programa que simula su dinámica.

El juego de rol

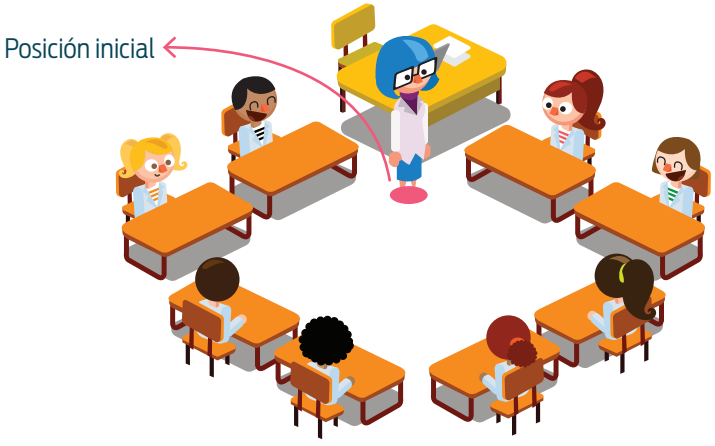
El objetivo del juego es que los estudiantes consigan que un robot llamado Kioscobot reparta un caramelo a cada alumno y, una vez que los hayan comido, recoja todos los envoltorios. Antes de comenzar, anotamos en el pizarrón los comandos con los que los estudiantes operarán al robot.



Comandos para operar a Kioscobot

Nosotros, los docentes, interpretaremos a Kioscobot. Este robot solo entiende unas pocas instrucciones. Los estudiantes, usando los comandos que puede reconocer Kioscobot, darán órdenes que debemos obedecer al pie de la letra.

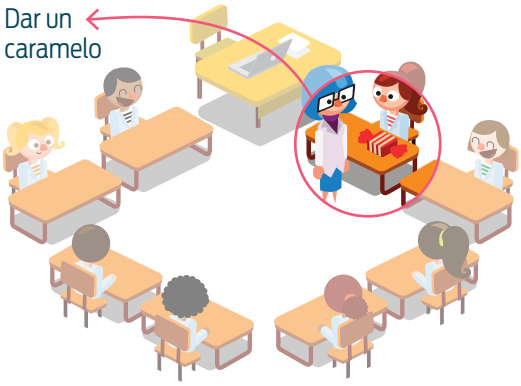
Les pedimos a los estudiantes que acomoden sus bancos y formen una ronda alrededor de la sala. Nosotros nos ubicamos en lo que denominaremos *posición inicial*. Les explicamos a los estudiantes las reglas del juego. Ellos deben dar instrucciones que nosotros seguiremos.



Cada comando tiene un efecto en Kioscobot. `Ir al próximo estudiante` lo desplaza hacia el siguiente estudiante de la ronda; `Volver al escritorio` lo ubica en la posición inicial; `Dar un caramelo` hace que entregue un caramelo al estudiante más próximo; y `Tomar el envoltorio`, provoca que retire el envoltorio del estudiante que se encuentra a su lado.

Ir al próximo estudiante

Volver al escritorio



Dar un caramelo

Tomar el envoltorio

Insistimos en que, para que tengan efecto, los comandos deben decirse tal como figuran en el pizarrón. Por ejemplo, cuando los estudiantes digan: "Dar un caramelo", le entregaremos un caramelo al estudiante más próximo. Debemos tener presente que solo tenemos que reaccionar frente a comandos expresados correctamente. Por lo tanto, si nos dicen: "Dar otro caramelo", debemos quedarnos quietos, ya que no corresponde a ninguna de las instrucciones que comprende Kioscobot.

El primer objetivo que tienen que alcanzar los estudiantes consiste en obtener un caramelo cada uno. Para ello, una posibilidad es que alternen los comandos `Ir al próximo estudiante` y `Dar un caramelo` tantas veces como estudiantes haya en el curso. De esta forma, Kioscobot pasará, uno por uno, por todos los pupitres, y dejará un caramelo en cada uno. Una vez que todos hayan comido su golosina, los estudiantes tendrán que darle órdenes a Kioscobot para que recoja todos los envoltorios, de a uno por vez. En este caso, podrán conseguirlo alternando las instrucciones `Ir al próximo estudiante` y `Tomar el envoltorio`.

Si bien menos eficientes, otras soluciones podrían incluir recorridos más exóticos en los que los estudiantes, usando el comando `Volver al escritorio`, enviaran a Kioscobot a la posición inicial una o más veces. A modo de ejemplo, ilustramos la situación con un curso de 20 estudiantes. Suponiendo que los estudiantes comiencen impartiendo, intercaladas, las órdenes `Ir al próximo estudiante` y `Dar un caramelo` 10 veces, Kioscobot habrá repartido un dulce a los 10 primeros estudiantes de la ronda. Si a continuación le ordenan `Volver al escritorio`, Kioscobot se ubicará en la posición inicial. En esta instancia, para conseguir que todos los estudiantes obtengan su caramelo, en primer lugar tendrán que ordenarle a Kioscobot que avance hasta el décimo estudiante usando 10 veces `Ir al próximo estudiante`. Luego, desde allí, los estudiantes tendrán que volver a alternar los comandos `Ir al próximo estudiante` y `Dar un caramelo` hasta que Kioscobot alcance a los 20 participantes.

A continuación preguntamos: "¿Qué elementos manipuló Kioscobot?". Caramelos y envoltorios. "¿Qué hizo Kioscobot con ellos?". Entregó los caramelos y recogió los envoltorios. "¿Qué otras cosas hizo Kioscobot?". Se desplazó por la sala. Luego, señalamos que lo que se hizo fue reconocer los **elementos** y las **acciones** que intervinieron en la situación. En su conjunto, elementos y acciones forman lo que denominamos **dominio de un problema**. Para fijar esta idea, podemos escribir en el pizarrón: **elementos + acciones = dominio**.

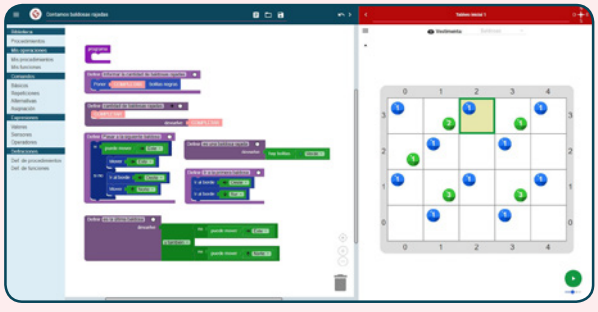
La simulación virtual

Para la segunda parte de la actividad, usaremos Gobstones y, preferentemente, un proyector. De no contar con uno, se pueden formar grupos de estudiantes alrededor de algunas computadoras.

¿Qué es Gobstones?

Gobstones es un lenguaje de programación desarrollado por un equipo de la Universidad Nacional de Quilmes, especialmente diseñado para enseñar a programar. Existen distintos entornos de Gobstones. En este manual se trabaja con Gobstones Jr., que permite construir programas usando bloques encastrables.

Puede usarse tanto en línea como fuera de línea. La versión fuera de línea puede descargarse de www.gobstones.org y se encuentra disponible para Linux y Windows.



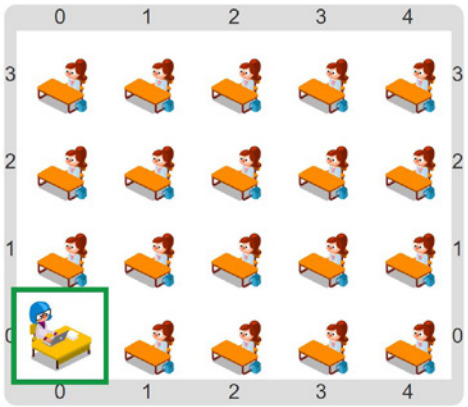
Gobstones

Universidad Nacional de Quilmes

Cargamos el entorno de programación de Gobstones y abrimos el proyecto “Repartija de caramelos”. Si trabajamos con la versión en línea el proyecto puede seleccionarse desde http://program.ar/gobstones.

Menú de selección de proyectos

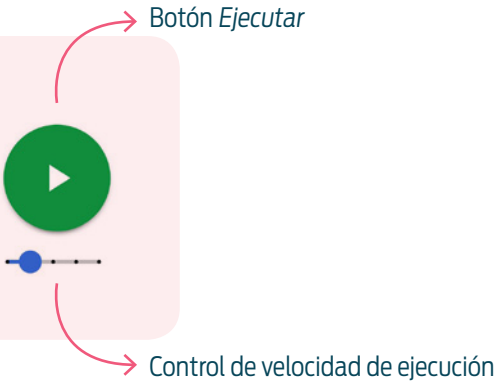
Una vez cargado, presionamos el botón *Ejecutar* para que comience una corrida del programa. Podrá observarse entonces una situación muy parecida a la del juego de rol. Un docente reparte caramelos a un grupo de estudiantes y recoge luego los envoltorios. En caso de que se quiera modificar la velocidad de ejecución del programa, puede usarse la barra lateral que se encuentra debajo del botón *Ejecutar*.



Proyecto “Repartija de caramelos”

Para ver el paso a paso

La barra azul debajo del botón *Ejecutar* permite variar la velocidad de la ejecución de los programas. Para poder ver el paso a paso, hay que mover el control de velocidad hacia la izquierda.



Preguntamos: "¿Qué similitudes observan con lo hecho anteriormente?". Es muy parecido, pero en la computadora. "O sea que los problemas del mundo real pueden simularse virtualmente. ¿Cómo creen, en este caso, que se le dieron las instrucciones a Kioscobot?". ¡Mediante un programa!

CIERRE

Reflexionamos con los alumnos sobre la posibilidad de representar la realidad con programas. Las computadoras son una herramienta muy poderosa que permite resolver problemas del mundo real. Destacamos que, cuando se diseña un programa, se seleccionan algunos elementos de la realidad con los que trabajar y se descartan otros. Por ejemplo, en el programa mostrado en la actividad se incluyó a Kioscobot, a los caramelos y a los envoltorios, pero se dejó de lado el pizarrón del aula. La elección de qué elementos incluir depende de lo que previamente hayamos identificado como importante en el problema abordado.

Actividad 2
¿Caramelos, tomates o bolitas?

TODA LA CLASE

OBJETIVOS

- Mostrar que varios problemas con dominios diferentes pueden tener la misma dinámica.
- Presentar los elementos y operaciones básicas de Gobstones.

MATERIALES

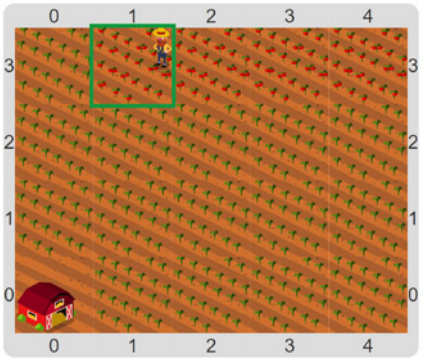
- Computadora
- Gobstones
- Proyector (opcional)

DESARROLLO

En esta actividad observaremos tres problemas aparentemente muy distintos. Sin embargo, a medida que avancemos, notaremos que todos poseen una dinámica similar.

De contar con un proyector, lo usaremos para mostrar los programas en acción. En caso contrario, les pediremos a los estudiantes que se ubiquen en grupos frente a algunas computadoras.

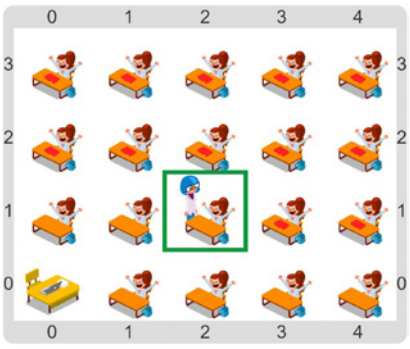
Comenzamos abriendo el entorno de Gobstones. Luego, cargamos el proyecto "¿Caramelos, tomates o bolitas?" y lo ejecutamos. Podremos observar a un granjero que recorre un campo sembrando plantines de tomate, y luego vuelve al lugar de donde partió. Entonces, los plantines maduran y crece un tomate de cada uno. El granjero, en su tarea de recolección, cosecha todos los tomates en el mismo orden en que sembró los plantines.



El granjero recolecta tomates

Una vez concluida la ejecución preguntamos: "¿Qué elementos manipuló el granjero?". Plantines y tomates. "¿Qué hizo el granjero con esos elementos?". Sembró plantines y cosechó tomates. "¿Qué otras acciones llevó adelante el granjero?". Se desplazó, tanto para plantar como para recolectar.

A continuación le preguntamos a la clase: "¿Qué similitudes encuentran entre lo que acabamos de ver y 'Repartija de caramelos'?". Guiamos la discusión para llegar a la conclusión de que ambos programas tienen la misma dinámica, en tanto se trata de distribuir ciertos elementos y luego recolectar otros. Más allá de que la actividad plantea un contexto diferente, es importante que los estudiantes identifiquen las semejanzas.



Repartija de caramelos

Para continuar la actividad, ejecutamos nuevamente el programa, pero, esta vez, desactivamos antes la vestimenta[1] presionando el ojo que muestra la figura. Se verá un tablero vacío en el que primero aparecen bolitas y, cuando cambian de color, se retiran. Al finalizar la corrida, formulamos preguntas para que los estudiantes identifiquen los elementos y las acciones que intervinieron esta vez. Los elementos son el cabezal y las bolitas; y las acciones incluyen los desplazamientos del cabezal y depositar y retirar bolitas del tablero.

Vestimenta desactivada

A continuación preguntamos: "¿Qué similitudes encuentran entre lo que vimos recién y el docente que reparte caramelos o el granjero que siembra tomates?". En los tres casos se observa una misma dinámica, en tanto se trata de distribuir ciertos elementos y luego recolectar otros.

Corremos nuevamente el programa. Sin embargo, esta vez, durante la ejecución, vamos cambiando las vestimentas de manera que se vea aparecer y desaparecer al docente, al granjero y las bolitas. Así, mostramos tres formas distintas de visualizar la ejecución de un único programa. Concluimos, entonces, que siempre hemos ejecutado el mismo programa y que lo único que cambió es la forma de presentarlo.

CIERRE

Como cierre, señalamos que vimos ejecuciones de un único programa escrito en el lenguaje de programación Gobstones, que es el que se usa a lo largo de todo el curso. Este lenguaje provee comandos para poner y sacar bolitas de colores en un tablero. Además, como se mostró, permite que la ejecución de un programa pueda visualizarse de diversas formas o, en la jerga de Gobstones, usando distintas **vestimentas**.

[1] Las vestimentas de Gobstones se explican en actividades posteriores.

Secuencia Didáctica 2

COMANDOS BÁSICOS DE GOBSTONES

El universo de Gobstones consiste en un tablero, un cabezal y bolitas verdes, rojas, negras y azules. Los programas realizan transformaciones al modificar la cantidad de bolitas de cada color que hay en las celdas del tablero. Para llevar a cabo estas acciones, el lenguaje provee comandos, tanto para desplazar el cabezal como para poner y sacar bolitas.

En esta secuencia didáctica, los estudiantes construirán sus primeros programas. Para completar las actividades, deberán valerse de algunos de los comandos provistos por este lenguaje.

OBJETIVOS

- Construir los primeros programas.
- Presentar comandos del lenguaje Gobstones.

Actividad 1
Bolitas de colores

DE A DOS

OBJETIVOS

- Construir un programa.
- Presentar el comando de Gobstones `Poner [ ]`.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes construirán su primer programa en Gobstones. Para ello, usarán el entorno de programación Gobstones Jr.

Les contamos a los estudiantes que, a lo largo del curso, cada vez que trabajen con la computadora, lo harán en parejas. Siempre habrá un piloto y un copiloto. El piloto será el encargado de operar la computadora, mientras que el copiloto ayudará con sus ideas y consejos. Preferentemente, en el transcurso de las actividades, deben alternar el rol que desempeña cada uno.

Comenzamos pidiéndoles que se ubiquen en parejas, abran el entorno de programación de Gobstones y carguen el proyecto "Bolitas de colores". Si ya se encontrasen dentro del entorno, pueden hacerlo presionando el botón de selección de proyectos de la barra superior del entorno.

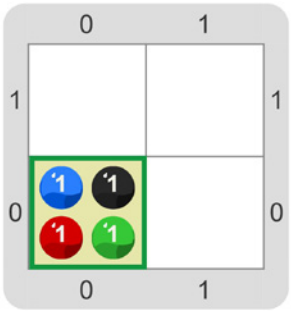
Observarán entonces tres partes claramente diferenciables. A la izquierda, el *espacio de bloques disponibles*. Se trata de un menú que contiene los bloques que podemos usar al construir programas.[1] En el centro se encuentra el *espacio del programa*. Allí iremos encastrando bloques para crear programas. Por último, a la derecha, está el *espacio del tablero*. El tablero tiene un cabezal que siempre se encuentra sobre una celda y se representa con un cuadrado verde.

Entorno de programación de Gobstones

Les contamos que, en general, las actividades de programación del curso consisten en producir transformaciones sobre un tablero operando un cabezal que saca y pone bolitas de colores en las celdas. El cabezal está representado con un cuadrado verde y las bolitas son de color verde, azul, negro y rojo.

[1] Los elementos disponibles no son siempre los mismos. Distintos proyectos pueden presentar diferentes opciones.

En esta primera actividad, se parte de un tablero vacío de dimensión 2 × 2 y los estudiantes deben poner una bolita de cada color en la celda de la posición (0,0).[1]

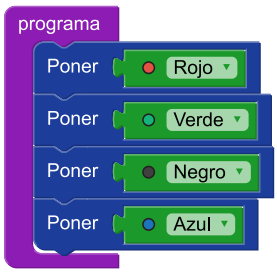


Tablero final del desafío

Luego de explicar la consigna, les pedimos a los estudiantes que exploren el entorno y traten de armar un programa que resuelva el desafío propuesto. Los elementos del lenguaje que hacen falta para completar la actividad son el comando básico `Poner [ ]`, que provoca que el cabezal ponga una bolita en la celda sobre la que se encuentra; y los valores `Azul`, `Negro`, `Rojo` y `Verde`. Les podemos hacer notar que todos los colores aparecen en un único bloque desplegable. Es importante que mostremos que el comando `Poner [ ]` tiene un agujero en el que hay que indicar el color de la bolita que debe depositar el cabezal sobre el tablero. El valor con el que completamos el espacio se llama **argumento**. Por ejemplo, `Rojo` es el argumento en `Poner[Rojo]`.

Bloques necesarios para armar el programa

Un programa posible para completar la actividad consiste en cuatro instrucciones: cada una ordenará poner una bolita de un color diferente. En este caso, el orden de las instrucciones es irrelevante. Solo importa poner una de cada color. Para ver la ejecución del programa tenemos que hacer clic en el botón *Ejecutar*.



Un programa que alcanza el objetivo

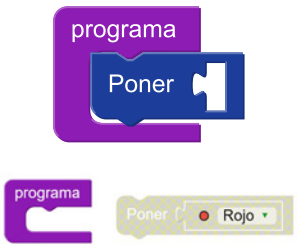
Para copiar, pegar y deshacer

Una forma sencilla de copiar y pegar bloques es usando CTRL junto a las letras C y V. Además, si querés deshacer lo último que hiciste mientras armás tu programa, podés presionar CTRL y Z.

Ctrl C Ctrl V Ctrl Z

[1] Cada celda del tablero se identifica con un par ordenado (i,j), donde i es el número de una columna y j el de una fila.

Es esperable que los estudiantes cometan errores cuando armen sus programas. Por ejemplo, podrían olvidarse de colocar un argumento. Aunque no es un error, también puede suceder que dejen bloques sin encastrar en el programa. En este caso, el bloque aparecerá en gris, lo que indica que no está activo.

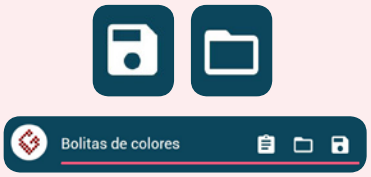


Usos incorrectos de los bloques

Una vez que todos hayan completado sus programas, les pedimos a los estudiantes que cuenten cómo resolvieron el problema. "¿Qué hubiese pasado si hubiéramos puesto las bolitas en otro orden? ¿Importa en qué orden lo hicimos?". Deberían reconocer que hubieran obtenido el mismo tablero final y que, por lo tanto, en este problema el orden de las instrucciones es irrelevante.

Guardar programas

Para guardar los proyectos se debe presionar el ícono del disquete que se encuentra en la barra superior del entorno. Si además se desea cambiar el nombre, hay que hacer clic sobre la línea roja de la barra y tipear un nombre nuevo. Finalmente, para volver a cargar nuestras producciones, hay que presionar el ícono de la carpeta.



CIERRE

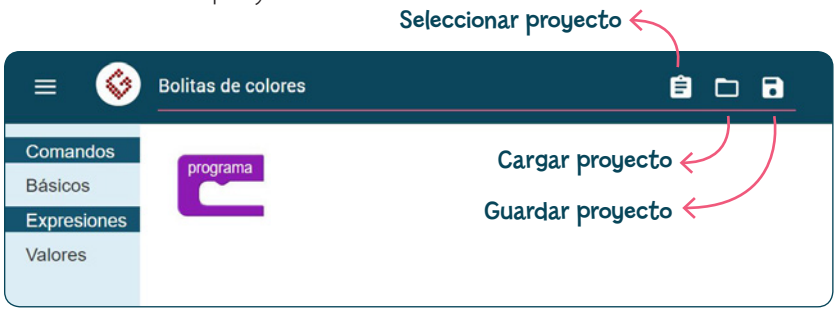
Les comentamos a los estudiantes que, al construir un programa, se indican las acciones que debe realizar el cabezal en un orden determinado. Sin embargo, en esta actividad el orden podría haber sido otro y, aun así, el programa seguiría siendo correcto. Concluimos reflexionando acerca de que pueden existir distintos programas que resuelvan un mismo problema.

NOMBRE Y APELLIDO:

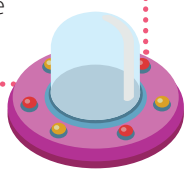
CURSO: FECHA:

BOLITAS DE COLORES

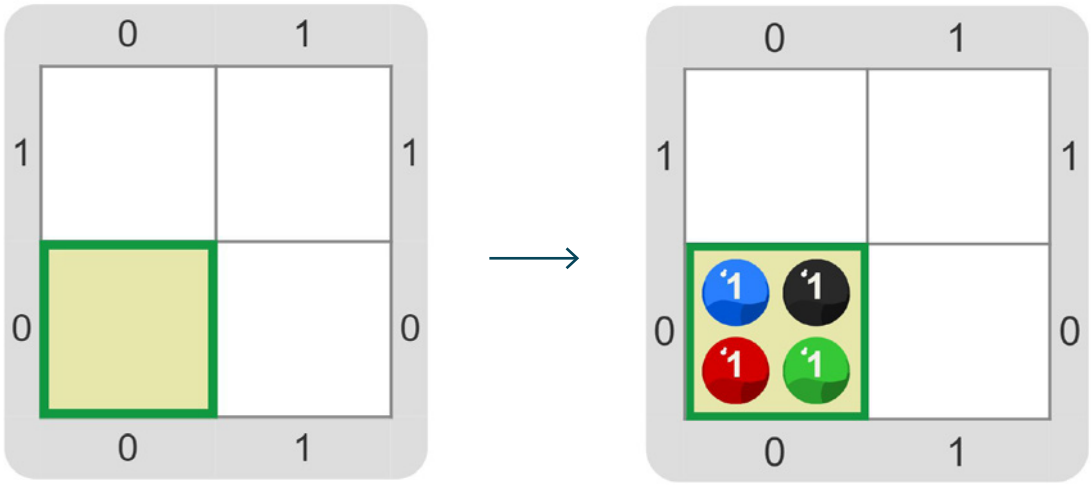
¡Ahora vas a programar! Siempre que trabajemos en Gobstones, vamos a usar un **proyecto**. Empezá abriendo "Bolitas de colores" ingresando a http://program.ar/gobstones o presionando el ícono de selección de proyectos.



Explorá el entorno para descubrir cómo agregar comandos a un programa. ¿Se te ocurre cómo ejecutarlo?



Construí un programa que ponga 4 bolitas, una de cada color (azul, negra, roja y verde), en la celda (0,0) del tablero. El programa debe transformar el estado del tablero: partiendo de uno vacío, debemos llegar a uno como el de la figura.



Tablero inicial

Tablero final

CUESTIÓN DE ARGUMENTO

¡El comando `Poner [ ]` tiene un agujero! Allí hay que poner un color. El valor que allí se pone se lo llama **argumento**. Por ejemplo, `Rojo` es el argumento en `Poner[Rojo]`.

Para copiar, pegar y deshacer

Una forma sencilla de copiar y pegar bloques es usando CTRL junto a las letras C y V. Además, si querés deshacer lo último que hiciste mientras armás tu programa, podés presionar CTRL y Z.

Ctrl C Ctrl V Ctrl Z

Actividad 2
Víboras verdes

DE A DOS

OBJETIVOS

- Presentar el comando de Gobstones `Mover [ ]`.
- Exponer fallas durante la ejecución de un programa.

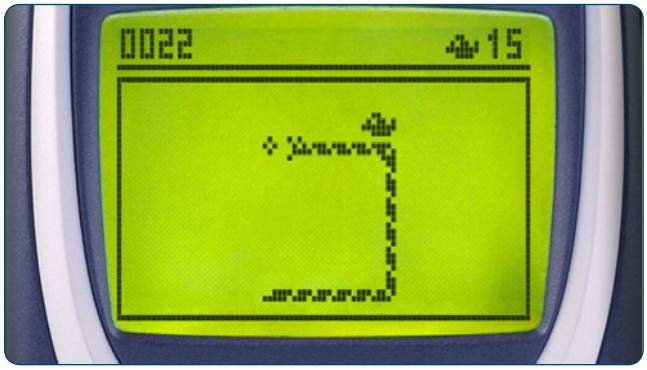
MATERIALES

- Proyector
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes construirán tres programas usando distintos proyectos. Para alcanzar los objetivos que se plantean en las consignas, tendrán que mover el cabezal sobre las celdas del tablero.

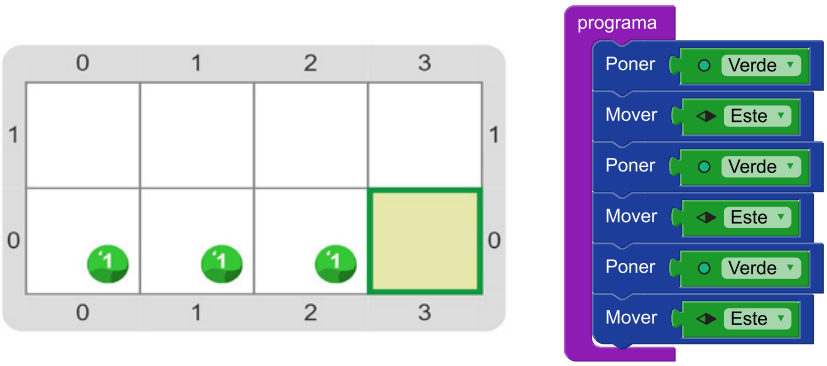
Comenzamos la clase preguntándoles a los estudiantes: "¿A qué juegos les gusta jugar en sus teléfonos inteligentes? Cuando salieron las primeras generaciones de celulares que traían jueguitos instalados, uno de los más conocidos era el Snake –'víbora', en inglés–. ¿Lo conocen? Se trataba de una víbora que tenía que comer varias presas; a medida que lo hacía, iba creciendo su tamaño. Además, debía evitar que su cabeza chocara contra otra parte de sí misma; si eso ocurría, la partida finalizaba. Miremos de qué se trata y, de paso, van a descubrir cómo eran los videojuegos a los que se jugaba en esa época". Invitamos a los estudiantes a mirar durante unos minutos algunos de los videos de Snake que se encuentran disponibles en Internet, como por ejemplo el que está en https://goo.gl/RqScHh. Es recomendable usar un proyector para que se familiaricen con la dinámica del juego. Si no se cuenta con una conexión a Internet, puede descargarse el video previamente.



Versión clásica del juego Snake

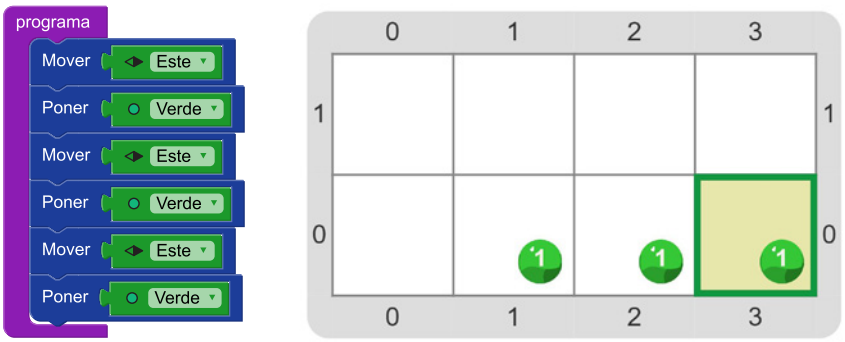
También podemos invitarlos a que jueguen a alguna versión en línea, como la que se encuentra en https://goo.gl/efnEnL. Les contamos que, al finalizar el curso, van a poder programar su propio juego de Snake. Además, les comentamos que a lo largo de esta actividad van a dibujar sobre el tablero de Gobstones las primeras víboras, que se van a representar con una serie de bolitas verdes.

Les repartimos la ficha de la actividad a los estudiantes y les indicamos que resuelvan la primera consigna. Para ello tienen que abrir el proyecto "La víbora esperando una presa". Al hacerlo, se encontrarán con un tablero vacío de 4 × 2 con el cabezal situado sobre la celda (0,0). Deben construir un programa para conseguir que el tablero se vea como el de la figura. Para resolver el desafío, basta con que el programa alterne `Poner [Verde]` y `Mover [Este]`.



Tablero final de "La víbora esperando una presa" y un programa que resuelve el desafío

Puede suceder que los estudiantes confundan el orden de los comandos `Poner [ ]` y `Mover [ ]` y la ejecución termine con un tablero final con las bolitas en lugares diferentes a los indicados en la consigna.



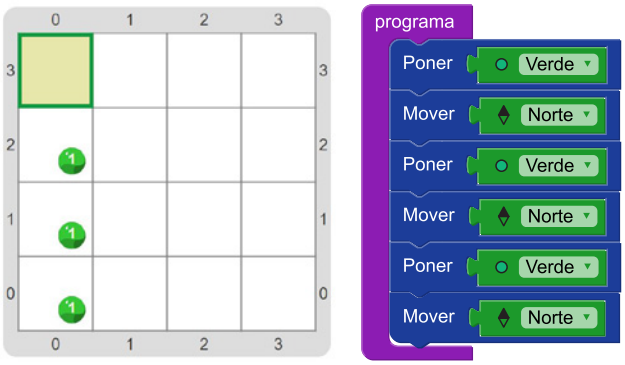
Tablero de "La víbora esperando una presa" con un programa que no resuelve el desafío

También puede ocurrir que utilicen mal el comando `Mover [ ]` y el cabezal se desplace más allá de la última posición, como si se cayera del tablero. En ese caso, se producirá una explosión. Esta es la forma en que se manifiesta una falla durante la ejecución de un programa en Gobstones. En cualquier caso, hay que pedirles que modifiquen el programa hasta obtener el resultado deseado.

BOOM

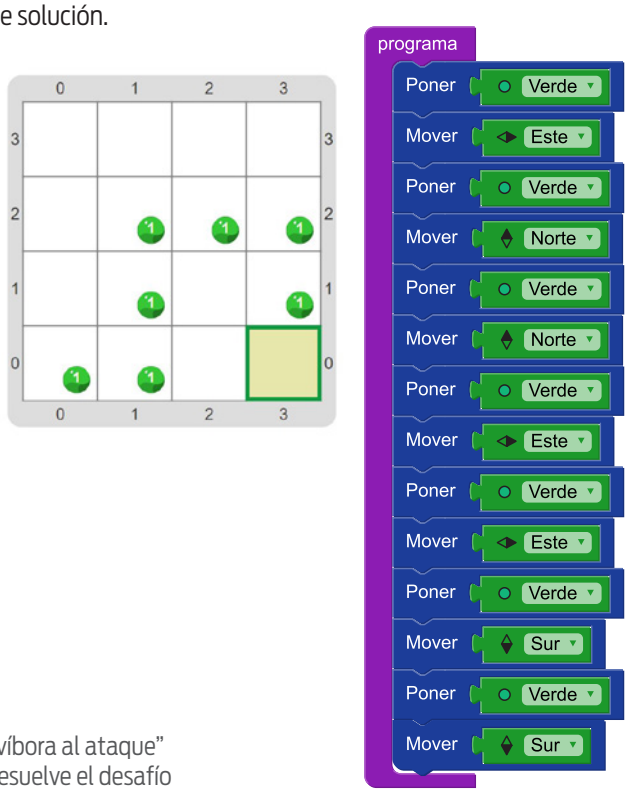
Manifestación de una falla durante la ejecución de un programa

Una vez que los estudiantes hayan completado la tarea, les pedimos que trabajen sobre la segunda consigna. Para eso, deben cargar el proyecto "La víbora, a punto de atacar". En este caso, deben construir un programa que transforme un tablero inicialmente vacío en uno con bolitas verdes en las celdas (0,0), (1,0) y (2,0). Para completar esta tarea tienen que variar el argumento del comando `Mover [ ]`: mientras que en la consigna anterior los desplazamientos del cabezal eran hacia el este, en este caso son hacia el norte.



Tablero final de "La víbora, a punto de atacar" y un programa que resuelve el desafío

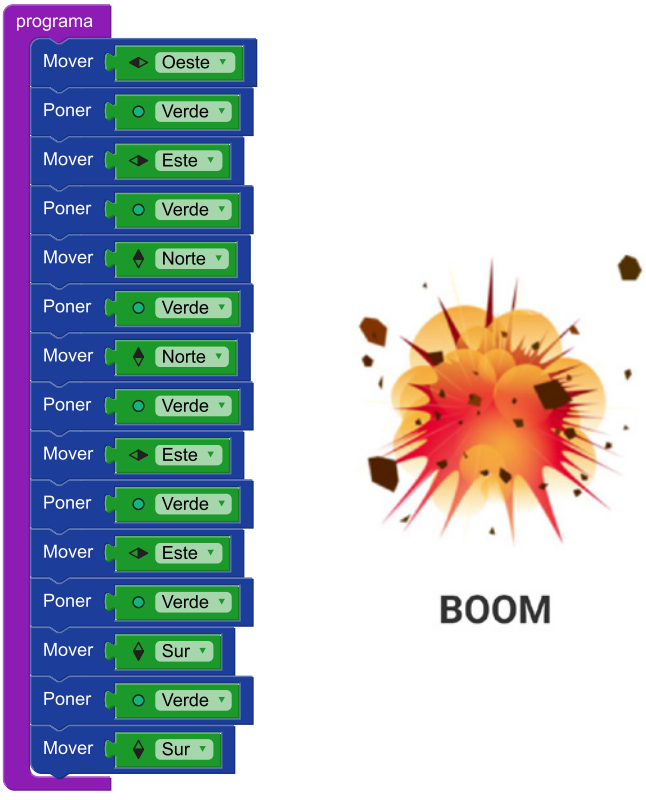
Para resolver la tercera consigna, los estudiantes trabajarán con el proyecto "La víbora al ataque". En esta oportunidad, el propósito es obtener un tablero final como el que se ve en la figura, que también muestra una posible solución.



Tablero final de "La víbora al ataque" y un programa que resuelve el desafío

Una vez completadas las tres consignas de la ficha, reflexionamos con los estudiantes sobre el efecto del comando `Mover [ ]`. Este permite desplazar el cabezal entre celdas adyacentes. Para indicar la dirección del desplazamiento, precisa un argumento que indique una dirección.

Finalmente, les pedimos que agreguen `Mover[Oeste]` como primera instrucción del programa, lo ejecuten y analicen el resultado. Al hacerlo, verán una explosión en la pantalla. En este programa se está dando una instrucción que mueve el cabezal fuera de los límites del tablero, y esto no es posible en Gobstones. Les contamos a los estudiantes que, en este lenguaje de programación, las fallas se manifiestan en la pantalla como explosiones. Llegamos a la conclusión de que un programa puede fallar si no se cumplen ciertas condiciones; en este caso, para que el cabezal pueda moverse debe existir una celda en la dirección indicada.



Programa que produce una falla

CIERRE

Le comentamos a los estudiantes que, si se combinan adecuadamente los comandos `Poner [ ]` y `Mover [ ]`, se pueden disponer bolitas de colores en cualquier celda del tablero. Ambos comandos precisan un argumento, cuyo valor determina su comportamiento. En el caso de `Poner [ ]`, debe indicarse el color de la bolita que se deposita en la celda. En el caso de `Mover [ ]`, la dirección en la que el cabezal debe moverse.

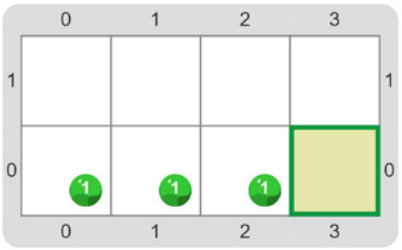
NOMBRE Y APELLIDO:

CURSO: FECHA:

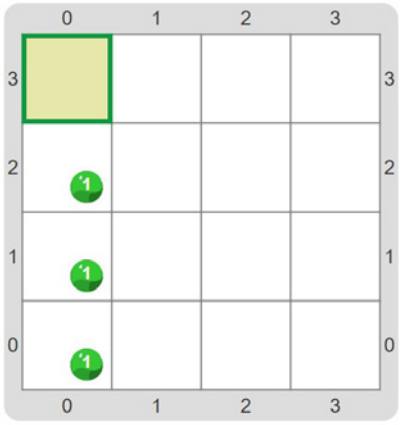
VÍBORAS VERDES

Hay muchos tipos de víboras, algunas muy peligrosas. Hoy nos vamos a acercar a ellas, pero, por suerte, con una computadora de por medio.

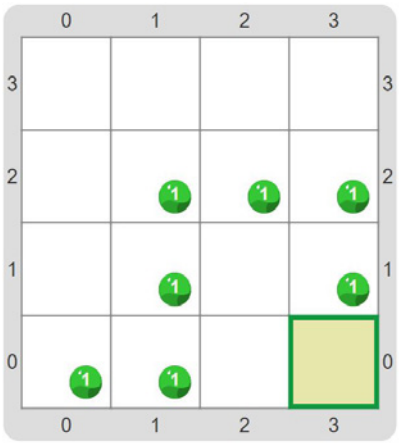
1. Como es sabido, las víboras son reptiles muy pacientes y pueden esperar un largo rato por una presa. Eso sí, para no llamar la atención, lo hacen acostadas. Abrí el proyecto "La víbora esperando una presa" y construí un programa para obtener una víbora de bolitas como la de la imagen.



2. Ahora cargá el proyecto "La víbora, a punto de atacar". Tenés que hacer un programa parecido al del punto anterior. La diferencia es que esta víbora está yendo hacia a su presa y se dispone a arrojarse sobre ella.

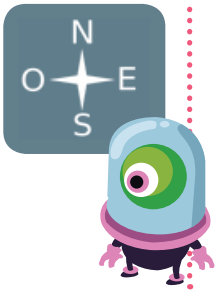


3. En esta ocasión, vas a trabajar sobre el proyecto "La víbora al ataque". Luego de detectar su alimento, la víbora salta sobre la presa y se contorsiona de formas que parecen imposibles. ¿Podés ubicarla con un programa tal como se ve en la figura?



LA ROSA DE LOS VIENTOS

El comando `Mover [ ]` precisa una dirección para el cabezal. En Gobstones, las direcciones son `Norte`, `Este`, `Sur` y `Oeste`. Cuando tengas dudas, consultá la rosa de los vientos que se encuentra arriba a la derecha. ¿Notaste que, si leés las direcciones desde el N hacia la izquierda, se forma "no sé"?



¿DE DÓNDE VIENE GOBSTONES?

El nombre Gobstones –que se pronuncia 'Góbstouns'– fue inventado por la escritora J.K. Rowling, autora de la saga de Harry Potter. Designa un juego de bolitas mágicas. Tal vez no hayas oído hablar de él porque aparece en los libros, pero no en las películas.

Actividad 3
Sacar, sacar y sacar

DE A DOS

OBJETIVO

- Presentar el comando de Gobstones `Sacar [ ]`.

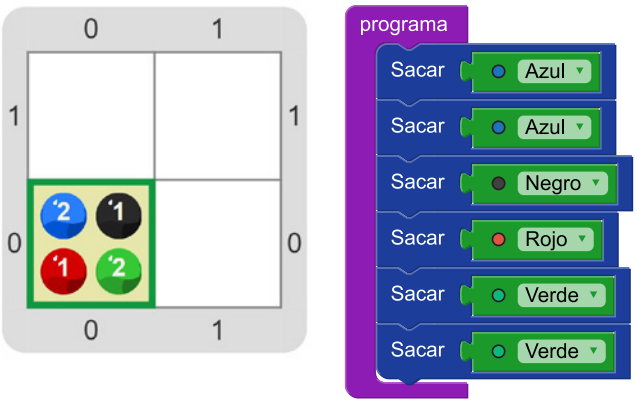
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes construirán programas para quitar bolitas del tablero. Para hacerlo, deberán usar el comando `Sacar [ ]`.

Comenzamos repartiendo la ficha y les pedimos que abran el proyecto "La celda vacía". Para resolver la primera consigna, tienen que construir un programa que retire todas las bolitas de la posición (0,0) del tablero. Les hacemos notar que, inicialmente, además de la bolita roja y la negra hay dos azules y dos verdes. La cantidad de bolitas de cada color está indicada por el número que aparece sobre ellas. En la imagen se muestra el tablero inicial y un programa que resuelve el problema. Notemos que, al igual que `Poner [ ]`, el comando `Sacar [ ]` también requiere un argumento que indique un color.

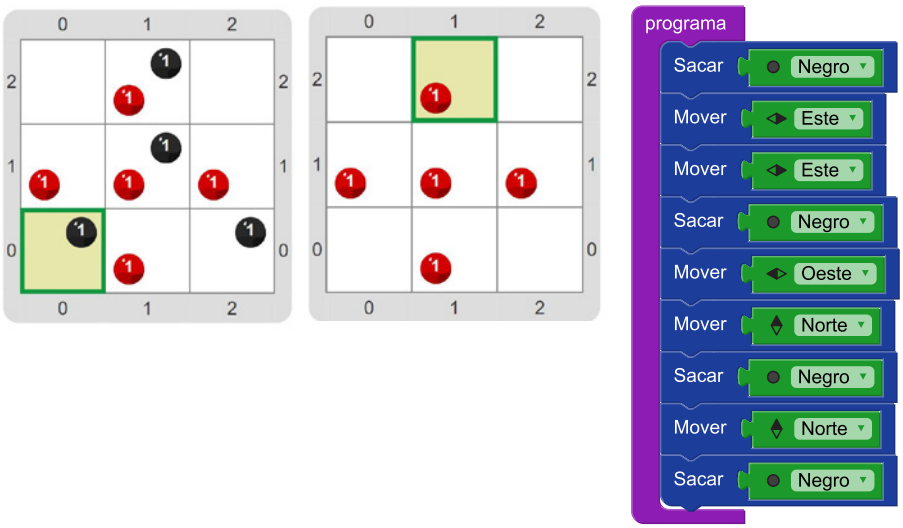


Tablero inicial de "La celda vacía" y un programa que resuelve el desafío

Puede suceder que algún estudiante crea que el efecto del comando `Sacar [ ]` sea el de remover todas las bolitas del color indicado. En ese caso, hacemos hincapié en que `Sacar [ ]` retira una bolita por vez. Por lo tanto, si se quiere sacar más de una bolita de un color, se debe repetir varias veces e l comando.

Una vez que los estudiantes hayan resuelto el desafío, les preguntamos: "¿Les parece que el comando `Sacar [ ]` podría producir una falla en la ejecución de un programa? Si lo consideran así, ¿qué debería ocurrir para que eso no suceda?". Es esperable que alguien reconozca que `Sacar [ ]` requiere que haya una bolita del color indicado en la celda sobre la que se encuentra el cabezal.

A continuación, pedimos a los estudiantes que resuelvan la segunda consigna. Para hacerlo, en primer lugar tienen que abrir el proyecto "La cruz roja". En este caso, deben combinar los comandos `Mover [ ]` y `Sacar [ ]` con el objetivo de retirar todas las bolitas negras, para que queden en el tablero solo las rojas. En la imagen pueden observarse el tablero inicial, el final y un programa que resuelve el desafío.



Tablero inicial, tablero final y programa que resuelve el desafío

En esta actividad, el orden de los comandos resulta importante. Para que el programa no falle, es necesario que, al ejecutar el comando `Sacar [ ]`, el cabezal se encuentre sobre una celda que tenga una bolita del color indicado. Por ejemplo, si invertimos el orden de los dos primeros comandos del programa de la figura, la ejecución fallaría porque primero se desplazaría el cabezal la celda (1,0) y luego allí, donde no hay bolitas negras, intentaría sacar una. Sin embargo, hay muchas secuencias de comandos diferentes que consiguen el mismo objetivo. Puede elegirse otro orden para retirar las bolitas y, aun así, alcanzar el objetivo.

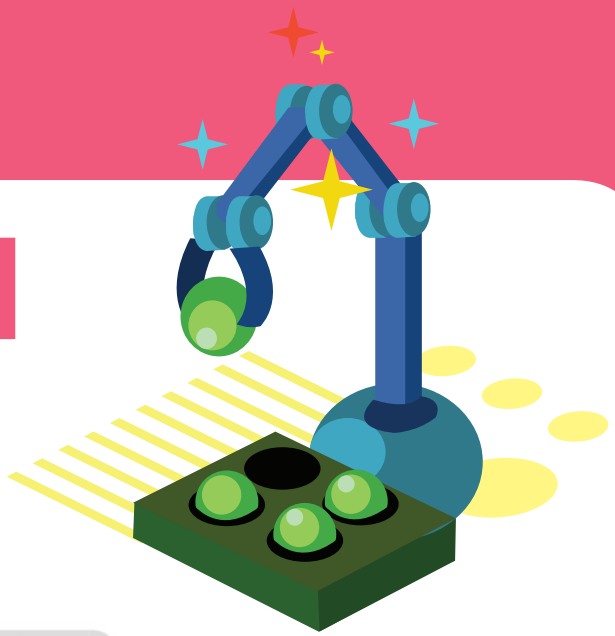
CIERRE

A modo de cierre, les contamos a los estudiantes que, para poder escribir programas en un lenguaje de programación, es importante saber cuáles son los comandos disponibles. En Gobstones, algunos son `Poner [ ]`, `Mover [ ]` y `Sacar [ ]`.

NOMBRE Y APELLIDO:

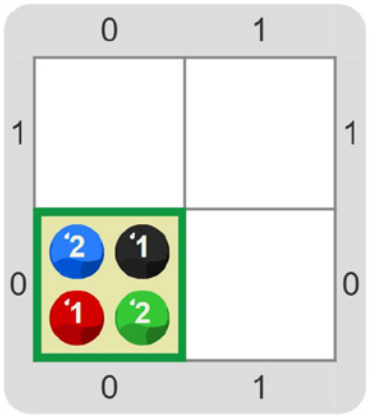
CURSO: FECHA:

SACAR, SACAR Y SACAR

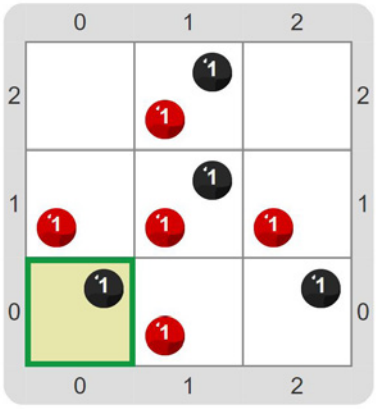


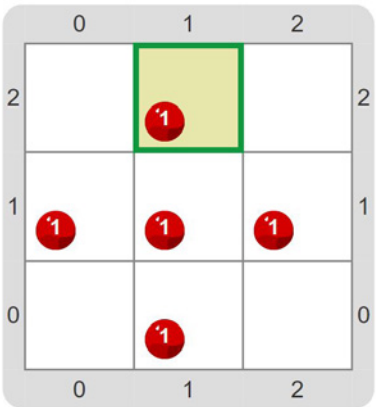
En esta actividad vamos a construir programas para que el cabezal saque bolitas del tablero.

1. Abrí el proyecto "La celda vacía". Tenés que armar un programa que logre que el tablero quede vacío. ¡A programar!



2. Un dibujo de la cruz roja hecho con bolitas se manchó con ceniza. Construí un programa que saque todas las bolitas negras. Para hacerlo, primero abrí el proyecto "La cruz roja". A continuación, se muestran el tablero inicial y el tablero final.





3. ¿Hay más de un programa que funcione para resolver este problema?

Actividad 4
¡Al rincón!

DE A DOS

OBJETIVOS

- Mostrar que un programa puede ser adecuado en distintas condiciones iniciales.
- Presentar el comando de Gobstones `Ir al borde [ ]`.

MATERIALES

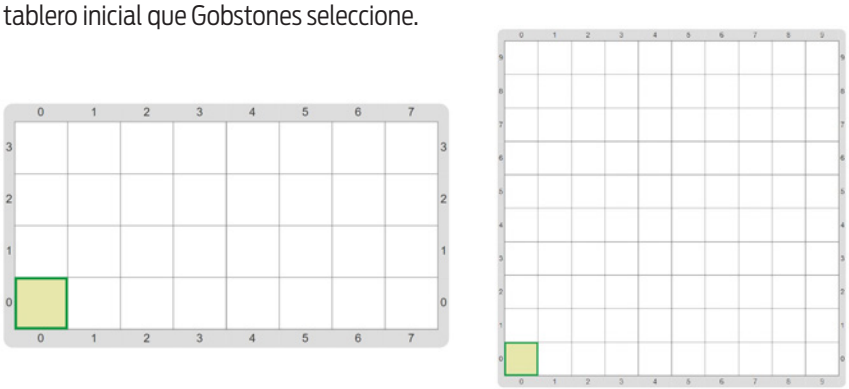
Computadoras

Gobstones

DESARROLLO

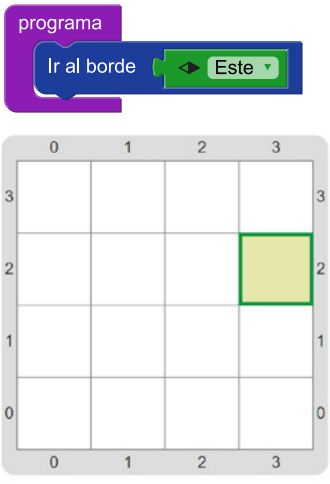
Hasta el momento, todos los programas que realizamos se ejecutaron siempre sobre un tablero inicial fijo. Sin embargo, es usual que queramos construir programas que sean lo suficientemente versátiles como para resolver problemas en condiciones que desconocemos al momento de armarlos. En el caso de Gobstones, esto quiere decir que funcionen correctamente para distintos tableros iniciales. A lo largo de esta actividad, los estudiantes crearán programas con esta característica.

Comenzamos pidiéndoles que abran el proyecto "¡Al rincón!" Les explicamos que el propósito del programa que tienen que armar consiste en colocar una bolita verde en cada una de las esquinas del tablero. Además, les decimos que, cada vez que ejecuten el programa, Gobstones elegirá un tablero inicial distinto. Para ilustrar este fenómeno les pedimos que presionen varias veces el botón *Ejecutar*. Entonces, verán diversos tableros iniciales de distintas dimensiones, todos ellos vacíos. Además, también notarán que la posición inicial del cabezal no es siempre la misma. El programa que armen tiene que funcionar independientemente del tablero inicial que Gobstones seleccione.



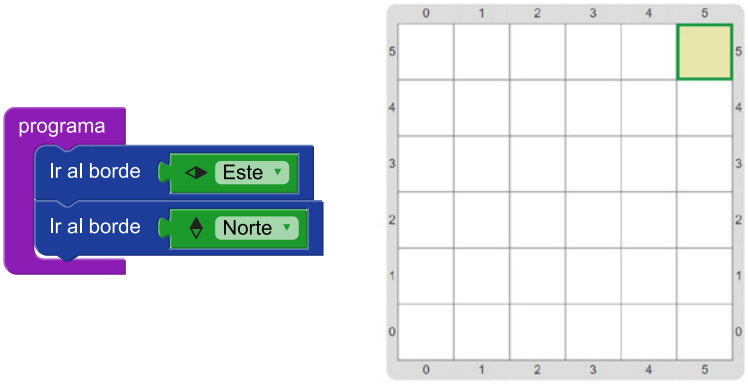
Posibles tableros iniciales

Como *a priori* no sabemos cuál es el tablero del cual partiremos, en este caso no podemos valernos del comando `Mover [ ]` para posicionar el cabezal sobre una esquina. Sin embargo, sí resultará de utilidad un comando que no se usó hasta ahora: `Ir al borde [ ]`. Al usarlo, el cabezal se desplazará hasta el borde de la dirección indicada. Por ejemplo, `Ir al borde [Este]` ubicará el cabezal en el extremo derecho del tablero pero no lo moverá de la fila en la que se encontraba.



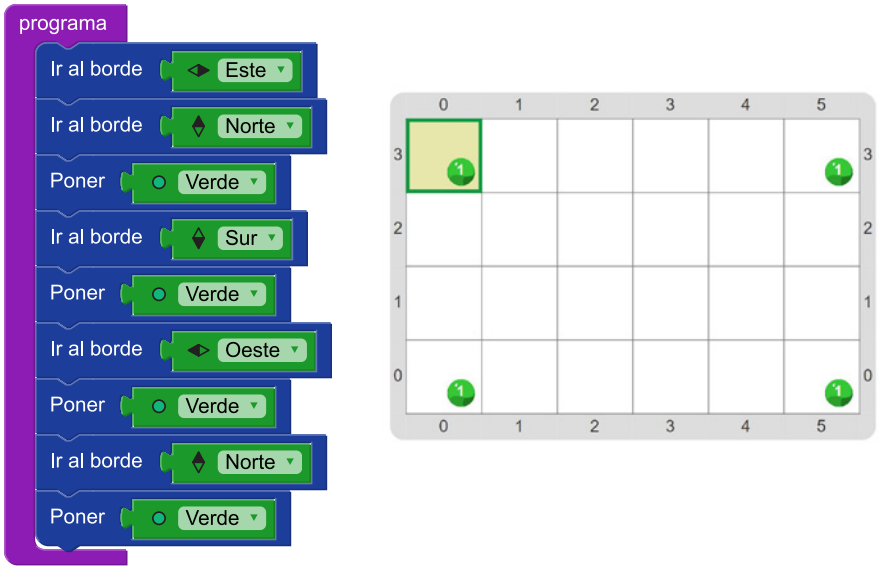
Efecto del comando `Ir al borde [ ]`

Debemos tener en cuenta que una invocación de este comando garantiza que el cabezal se posicione sobre un borde, aunque no necesariamente sobre una esquina. Sin embargo, combinando distintas direcciones podemos lograrlo. Por ejemplo, un programa que primero haga `Ir al borde [Este]` y luego `Ir al borde [Norte]` será apropiado para posicionar el cabezal sobre el extremo superior derecho.



Dos usos de Ir `al borde [ ]` para posicionar el cabezal en una esquina

Para resolver el problema planteado, tendremos que ir ubicando el cabezal sobre cada esquina y colocando una bolita verde en cada una. A continuación se muestra una posible solución, que coloca las bolitas en el sentido horario.



Programa que resuelve el desafío

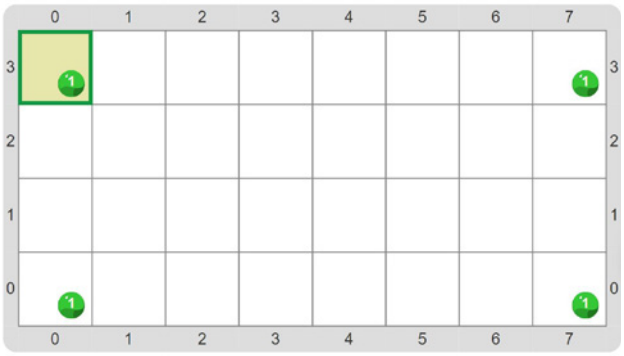
NOMBRE Y APELLIDO:

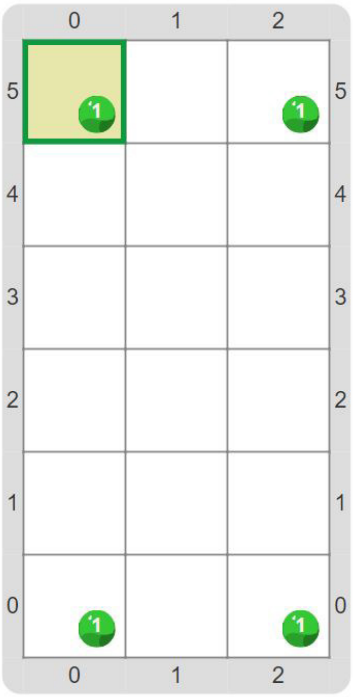
CURSO: FECHA:

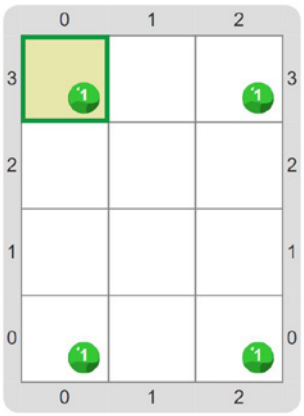
¡AL RINCÓN!

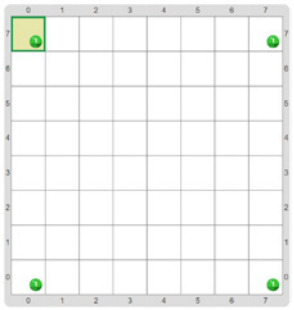
El desafío de esta actividad es construir **un único programa** que funcione en **cualquier tablero inicial**.

Abrí el proyecto "¡Al rincón!" y construí un programa que ubique 4 bolitas verdes, una en cada esquina del tablero. Te mostramos algunos ejemplos de cómo deberían quedar diferentes tableros finales:









Investigá el entorno para descubrir algo que te permita conseguir este objetivo. ¡Atención! Puede ser que a veces el cabezal no empiece posicionado en una esquina.

MUCHOS TABLEROS, UN SOLO PROGRAMA

Para ver distintos tableros iniciales antes de empezar a armar tu programa, presioná varias veces el botón *Ejecutar*.

Una vez que hayas terminado el programa, asegurate de ejecutarlo varias veces, con diferentes tableros iniciales. Vas a notar que **el mismo programa** da siempre el resultado esperado.

La víbora que come manzanas deliciosas

DE A DOS

OBJETIVO

- Construir un programa que use los comandos `Poner [ ]`, `Mover [ ]` y `Sacar [ ]`.

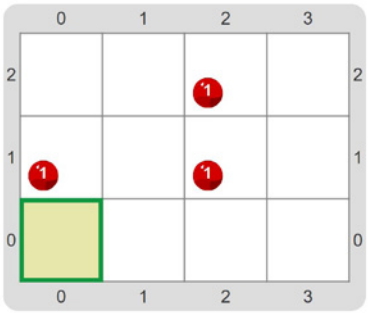
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

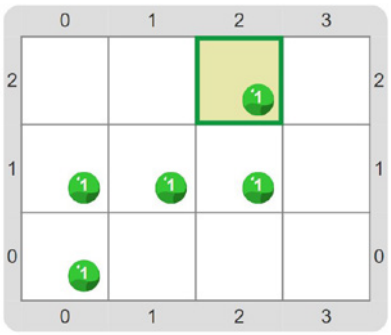
En esta actividad, se pondrán en juego las habilidades ejercitadas a lo largo del capítulo. Los estudiantes tendrán que construir un programa que combine los comandos `Poner [ ]`, `Mover [ ]` y `Sacar [ ]`.

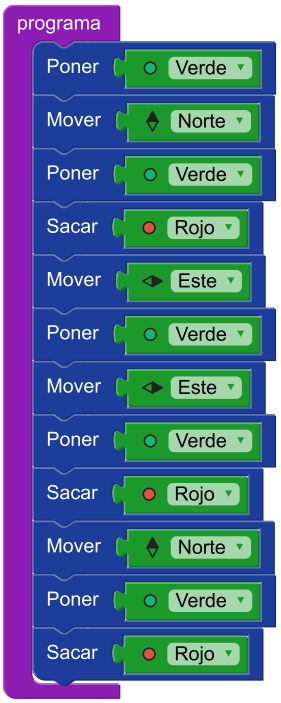
Comenzamos repartiendo las fichas a los estudiantes y les pedimos que abran el proyecto "La víbora que come manzanas deliciosas". Al abrirlo, se encontrarán con un tablero de 4 × 3 donde inicialmente están dispuestas tres bolitas rojas. Cada una representa una manzana deliciosa. En este proyecto, tanto las manzanas como el cabezal comenzarán ubicados siempre en el mismo lugar. El objetivo es construir un programa que dibuje una víbora que pase por las celdas en las que hay manzanas y las coma. Es decir, la idea es operar el cabezal con instrucciones para que coloque bolitas verdes y retire las bolitas rojas. Es importante aclarar que la víbora no debe pasar nunca por encima de sí misma.



Tablero inicial

Existen muchas soluciones posibles debido a que las manzanas se pueden comer en diferente orden; además, se pueden seguir diferentes caminos para llegar a ellas. A modo de ejemplo, se muestra un programa que soluciona el desafío planteado.





Un programa que resuelve el desafío y el tablero final

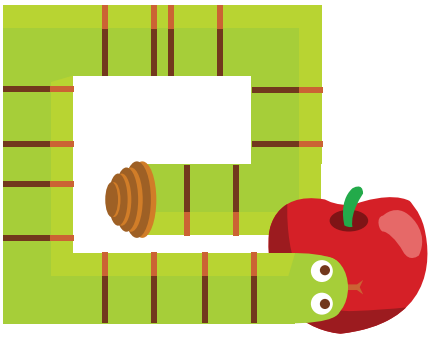
CIERRE

Una vez que todos hayan completado sus programas, les preguntamos a los estudiantes qué hubiese pasado si, en lugar de usar un tablero de 4 × 3, la actividad se hubiese planteado sobre uno de 100 × 100, con muchas manzanas rojas dispuestas en diversos lugares. Al trabajar únicamente con los comandos básicos de Gobstones, el programa hubiese quedado excesivamente extenso. Esto, además de volver muy engorrosa la actividad de programación, plantearía otros problemas. Por ejemplo, se haría muy difícil entenderlo y también sería complejo detectar posibles errores. Comentamos, entonces, que todos los lenguajes de programación modernos proveen herramientas que nos permiten evitar este tipo de problemas. En actividades posteriores estudiaremos algunas de ellas.

NOMBRE Y APELLIDO:

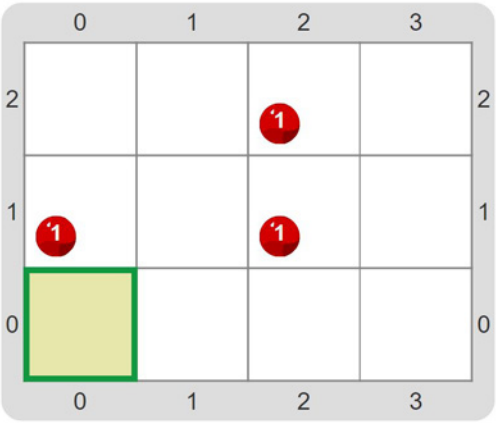
CURSO: FECHA:

LA VÍBORA QUE COME MANZANAS DELICIOSAS

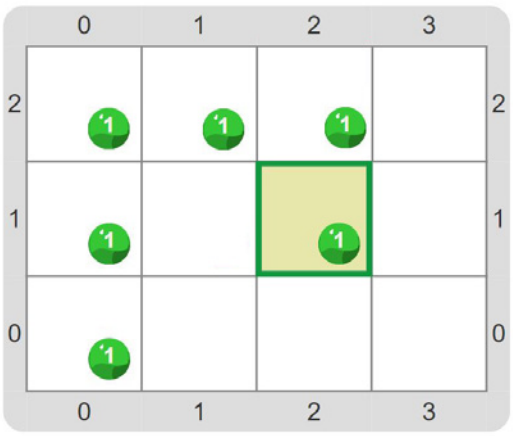


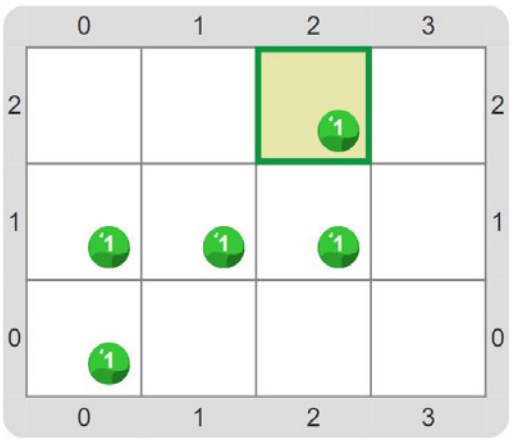
Abrí el proyecto "La víbora que come manzanas deliciosas". Vas a ver que el tablero inicial muestra algunas bolitas rojas, que en este caso representan manzanas deliciosas. El objetivo es que dibujes una víbora que pase por cada una y la coma. En términos de bolitas, tenés que hacer un camino de bolitas verdes que pase por las celdas donde hay rojas y retirar estas últimas.

Las manzanas se pueden comer en cualquier orden. Además, la víbora no puede pasar por encima de ella misma. Es decir, no deber pasar por celdas que ya tengan una bolita verde. Tené en cuenta que el cabezal siempre comienza posicionado en la esquina suroeste.



Hay muchos programas que resuelven el desafío planteado. No importa en qué orden la víbora coma las manzanas ni por cuáles celdas pase, siempre y cuando obtengas a un tablero que cumpla con las condiciones mencionadas. A modo de ejemplo, te mostramos posibles tableros finales. ¿Te animás a conseguir uno distinto a los que se muestran?





03

PROCEDIMIENTOS Y REPETICIÓN SIMPLE

Programar usando únicamente comandos básicos puede volverse muy engorroso. El resultado son programas difíciles de comprender cuando los leemos. Por eso, los lenguajes brindan una herramienta llamada **procedimiento**, que permite que quien programa defina sus propios comandos. Bien utilizados, los procedimientos resultan muy útiles para construir programas claros y ordenados**.** Una vez definido su comportamiento, un procedimiento puede utilizarse igual que cualquier otro comando.

Por otro lado, es habitual que, al escribir programas, nos encontremos frente a la necesidad de repetir varias veces una serie de acciones. Casi todos los lenguajes de programación nos permiten expresar repeticiones sin necesidad de reiterar instrucciones en forma explícita. Una de las formas más simples de hacerlo es la **repetición simple**, que se utiliza para repetir instrucciones una cantidad fija de veces.

En este capítulo se abordan los procedimientos y las repeticiones simples, que son esenciales para construir programas.

Secuencia Didáctica 1

PROCEDIMIENTOS

Un **procedimiento** es una forma de definir un comando nuevo que permite encapsular una tarea específica dentro de un programa más grande. Los procedimientos suelen usarse para descomponer problemas complejos en piezas más simples. Además, son útiles para no repetir secuencias de instrucciones idénticas en los programas. Bien utilizados, mejoran notablemente la legibilidad de los programas.

En esta secuencia didáctica presentamos esta herramienta que ofrecen los lenguajes de programación. Comenzamos con una actividad sin computadora que nos permite acercarnos a la noción de procedimiento y continuamos con proyectos de Gobstones que evidencian algunas de las ventajas de usar procedimientos al programar.

OBJETIVOS

- Presentar la noción de procedimiento.
- Usar procedimientos en programas.

Actividad 1
Estribillos

TODA LA CLASE

OBJETIVO

- Presentar la noción de procedimiento.

MATERIALES

- Reproductor de audio
- Lápiz
- Papel

DESARROLLO

Con el fin de presentar la noción de procedimiento, trabajaremos con una canción. La analogía entre los estribillos de las canciones y los procedimientos usados en programación resulta adecuada para acercarnos a esta herramienta. A modo de ejemplo, se ilustra la actividad con la canción *Sobre el puente de Aviñón*, pero se puede seleccionar una canción que resulte significativa para el curso.

Comenzamos la actividad indicándoles a los estudiantes que tendrán que escribir la letra completa de la canción que van a escuchar. Podemos reproducirla más de una vez, porque es probable que no todos consigan transcribir toda la canción en un primer intento. Además, sugerimos disponer de algunas fotocopias con la letra para repartir entre los estudiantes que no lleguen a completarla.

Canción: *Sobre el puente de Aviñón*

Sobre el puente de Aviñón
todos bailan, todos bailan.
Sobre el puente de Aviñón
todos bailan y yo también.

Hacen así...
así las lavanderas.
Hacen así...
así me gusta a mí.

Sobre el puente de Aviñón
todos bailan, todos bailan.
Sobre el puente de Aviñón
todos bailan y yo también.

Hacen así...
así las planchadoras.
Hacen así...
así me gusta a mí.

Sobre el puente de Aviñón
todos bailan, todos bailan.
Sobre el puente de Aviñón
todos bailan y yo también.

Hacen así...
así las costureras.
Hacen así...
así me gusta a mí.

Sobre el puente de Aviñón
todos bailan, todos bailan.
Sobre el puente de Aviñón
todos bailan y yo también.

Hacen así...
así los zapateros.
Hacen así...
así me gusta a mí.

Una vez que todos hayan concluido de transcribir la letra, les preguntamos: "¿Cuántas veces tuvieron que escribir el estribillo, es decir, la parte que dice: 'Sobre el puente de Aviñón...'? ¿Se les ocurre alguna forma más corta de hacerlo?". De este modo, guiamos la discusión para que los chicos lleguen a la conclusión de que lo más conveniente es escribir el estribillo entero una sola vez y luego referenciarlo donde corresponda. Teniendo esto en cuenta, la letra podría escribirse de la siguiente manera:

Canción: *Sobre el puente de Aviñón*

ESTRIBILLO
Sobre el puente de Aviñón
todos bailan, todos bailan.
Sobre el puente de Aviñón
todos bailan y yo también.

Hacen así...
así las lavanderas.
Hacen así...
así me gusta a mí.

[ESTRIBILLO]

Hacen así...
así las planchadoras.
Hacen así...
así me gusta a mí.

[ESTRIBILLO]

Hacen así...
así las costureras.
Hacen así...
así me gusta a mí.

[ESTRIBILLO]

Hacen así...
así los zapateros.
Hacen así...
así me gusta a mí.

Sobre el puente de Aviñon, sin repeticiones explícitas del estribillo

CIERRE

Reflexionamos con los estudiantes sobre lo tedioso que resulta repetir versos idénticos al escribir y observamos que es mucho más práctico escribir el estribillo solo una vez. Luego, mediante una simple referencia evitamos repetir un texto cuando tenemos que escribir la letra de una canción. Por supuesto, nunca cantamos la palabra *estribillo*, sino los versos que corresponden a la parte que este engloba. Terminamos contándoles a los estudiantes que algo parecido podemos hacer al programar.

Actividad 2
Cuadrados

DE A DOS

OBJETIVO

- Utilizar procedimientos en un ejercicio de programación.

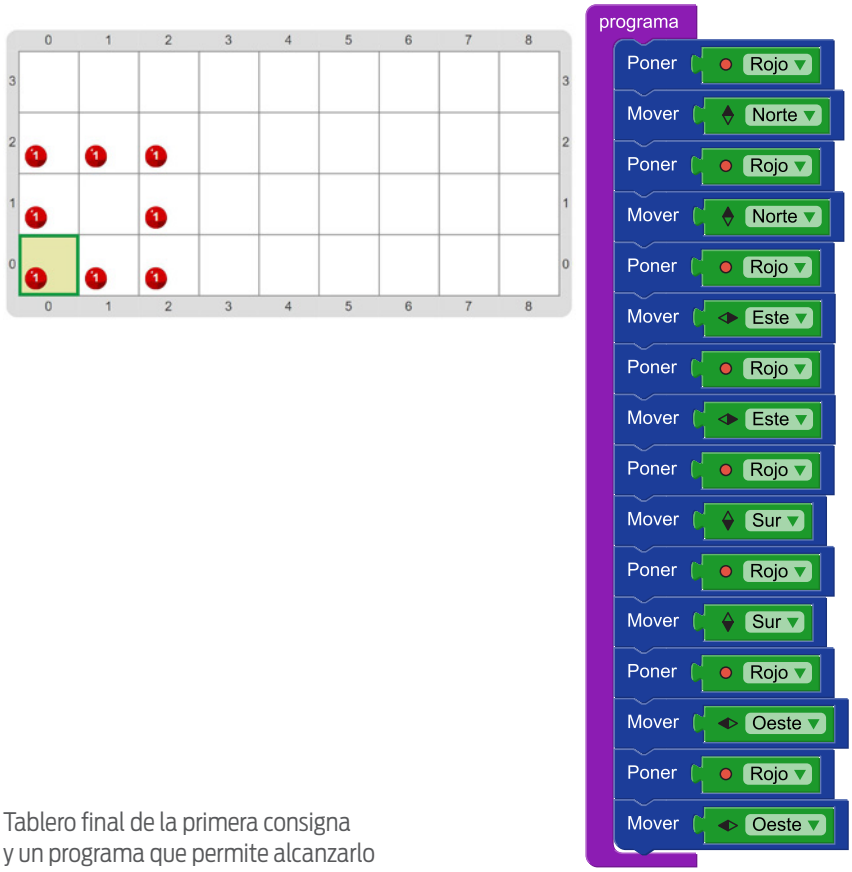
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes realizarán un ejercicio de programación en el que usarán procedimientos: a partir de dibujar una serie de cuadrados en el tablero de Gobstones, se espera que reconozcan que, para no hacer muchas veces lo mismo, es conveniente definir un procedimiento que dibuje un cuadrado y luego usarlo en diferentes partes del programa.

Les repartimos la ficha a los estudiantes y les pedimos que resuelvan la primera consigna. Para ello, deben abrir el proyecto "Un cuadrado" y transformar un tablero vacío de 9 × 4 en uno como el que muestra la imagen. A la derecha, puede observarse una posible solución del problema planteado.

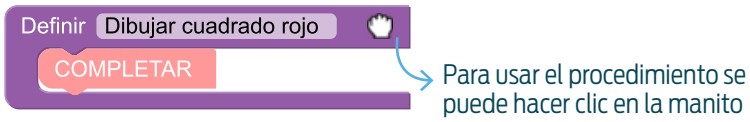


Tablero final de la primera consigna y un programa que permite alcanzarlo

Una vez que todos hayan completado el desafío, preguntamos a los estudiantes: "Si en lugar de dibujar solo ese cuadrado, hubieran tenido que dibujar otro igual en el extremo inferior derecho del tablero, ¿cómo lo habrían hecho?". Es probable que algún estudiante responda que, a continuación de la última instrucción, posicionaría el cabezal sobre la celda (6,0) y luego repetiría la misma secuencia de instrucciones que usaron para dibujar el primer cuadrado. Aunque esta es una forma posible de realizar la tarea, estaríamos repitiendo dos veces una larga se-

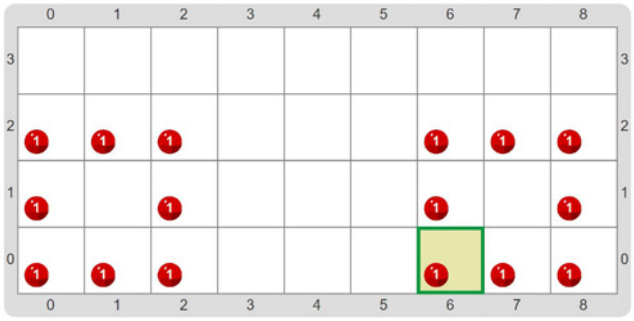
cuencia de instrucciones. Una respuesta más adecuada consiste en proponer algo similar a lo realizado con la canción y el estribillo en la actividad anterior.

Luego, les pedimos que abran el proyecto "Dos cuadrados" y resuelvan la segunda consigna de la ficha. En este caso, tienen que dibujar dos cuadrados, tal como se describió en el párrafo anterior. Al abrir el proyecto, se encontrarán con un bloque que dice `Definir [Dibujar cuadrado rojo]`. Este bloque nos permite definir lo que en programación se conoce como **procedimiento**: una serie de instrucciones que después puede invocarse desde otros puntos del programa, de igual modo que la letra de una canción nos remite a un estribillo. Les explicamos a los estudiantes que todos los bloques que encastremos allí formarán un procedimiento, que se transformará en un nuevo comando que podrán usar en otras partes del programa.



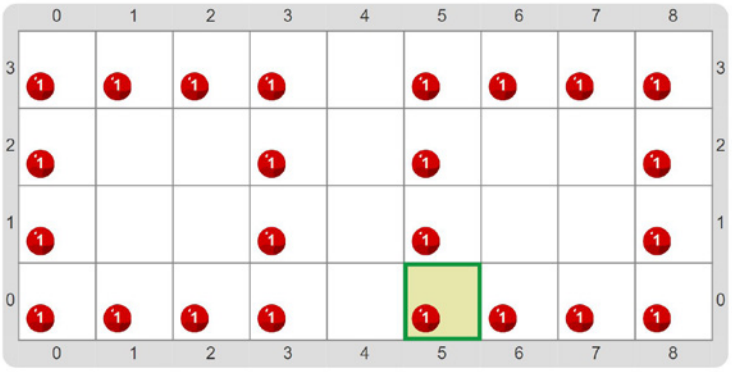
Bloque para definir un procedimiento

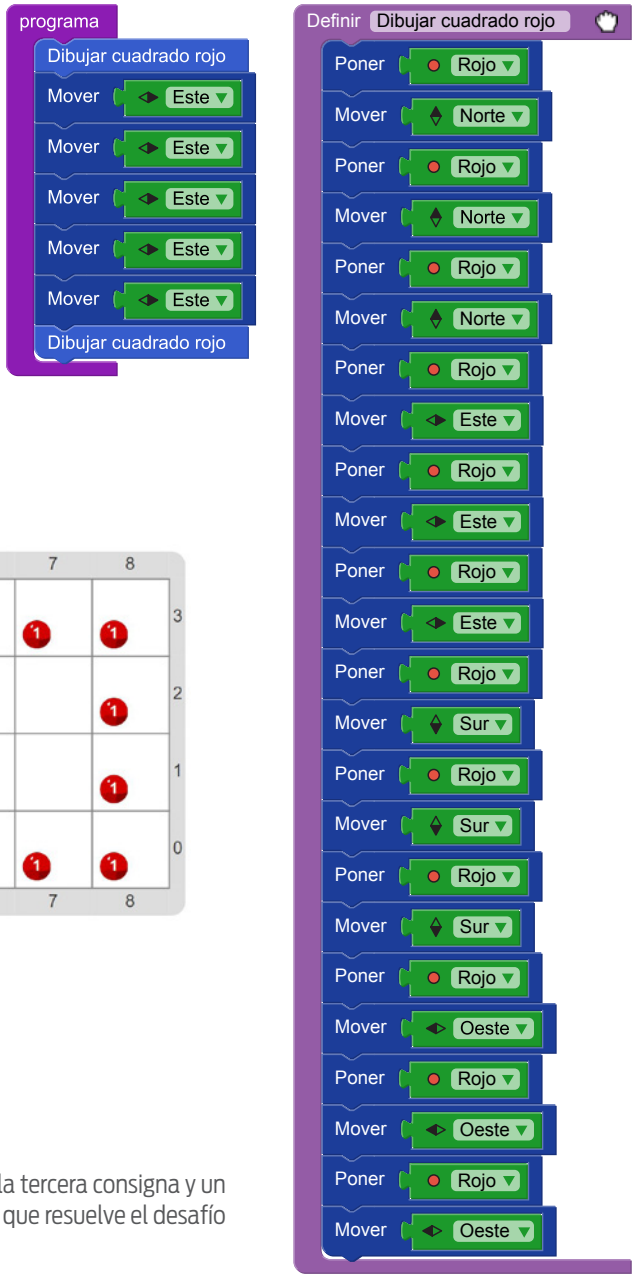
Para resolver el desafío planteado alcanza con completar `Dibujar cuadrado rojo` para que dibuje un cuadrado, y luego invocarlo dos veces desde el cuerpo principal del programa. De este modo, se formarán en el tablero ambos cuadrados. El bloque para invocar al procedimiento se encuentra disponible en el menú *Mis operaciones* > *Mis procedimientos* para ser usado.



Tablero final de la segunda consigna y un programa que resuelve el desafío

Una vez que todos hayan concluido sus programas, les pedimos que guarden sus soluciones y luego resuelvan la tercera consigna de la ficha, en la que se pide que los cuadrados, en lugar de ser de 3 × 3, sean de 4 × 4. En este caso, independientemente de la cantidad de cuadrados que tengamos que dibujar, solo hay que cambiar la cantidad de bolitas rojas que depositamos sobre el tablero en un solo lugar del programa: en el procedimiento. Además, se debe tener en cuenta que, al posicionar el cabezal para dibujar el segundo cuadrado, el cabezal debe desplazarse una vez menos que en el desafío anterior.





Tablero final de la tercera consigna y un programa que resuelve el desafío

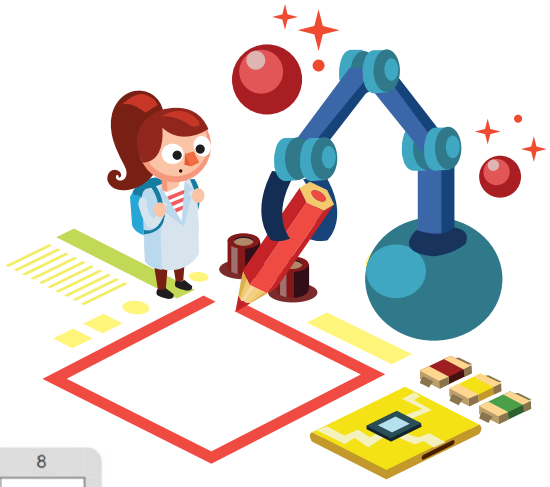
CIERRE

A modo de cierre, reflexionamos con los estudiantes sobre las posibilidades que nos brinda el uso de procedimientos. En primer lugar, resultan útiles para no repetir porciones idénticas de un programa. Dibujar un cuadrado constituye una unidad de sentido dentro de los desafíos propuestos, y por eso fue conveniente usar un procedimiento cuyo propósito fuese exclusivamente ese. Por otro lado, al cambiar alguna de las características del cuadrado, como el largo del lado, localizamos los cambios en un único lugar en vez de desparramarlos por todo el programa. La ventaja de hacerlo de este modo sería aún más evidentes si, por ejemplo, tuviéramos que dibujar cien cuadrados en lugar de dos.

NOMBRE Y APELLIDO:

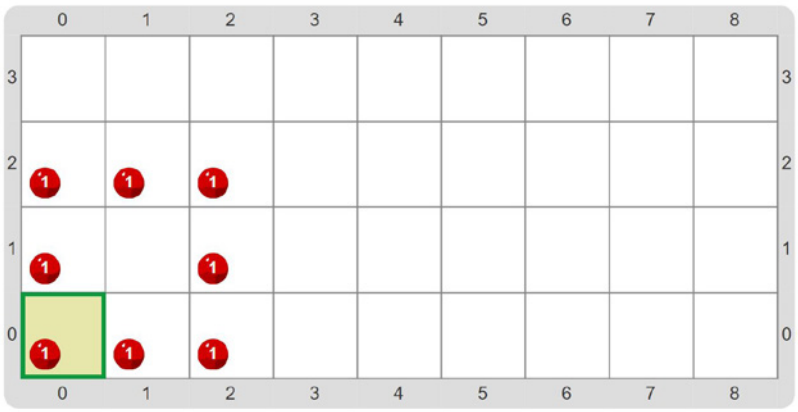
CURSO: FECHA:

CUADRADOS



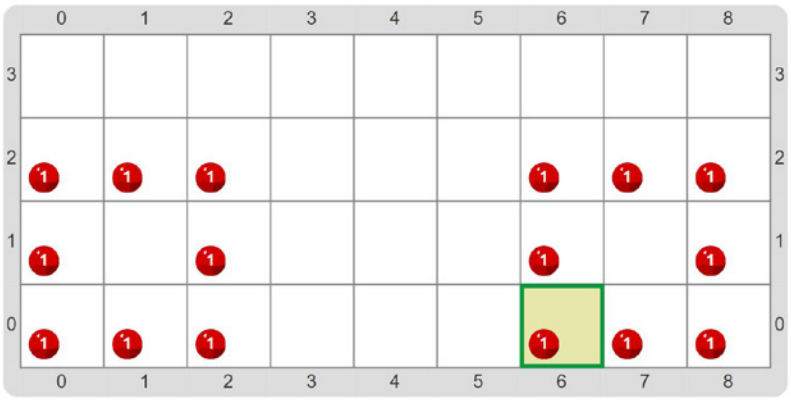
En esta actividad, vas a tener que ponerte a dibujar cuadrados.

1. Abrí el proyecto "Un cuadrado" y armá un programa para obtener el siguiente tablero.



¿Cuántos bloques tiene el programa?

2. Abrí el proyecto "Dos cuadrados". El objetivo ahora es que dibujes dos cuadrados, como los de este tablero.



NOMBRE Y APELLIDO:

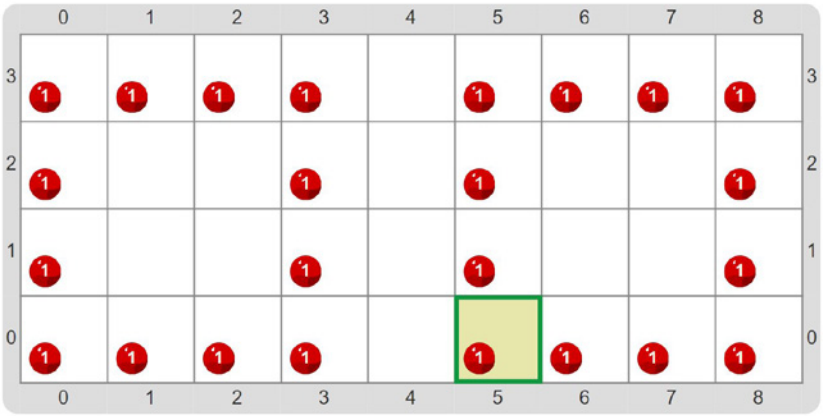
CURSO: FECHA:

¿Cuántos bloques tiene el nuevo programa? ¿Cuáles son las ventajas de usar procedimientos?

PROCEDIMIENTOS

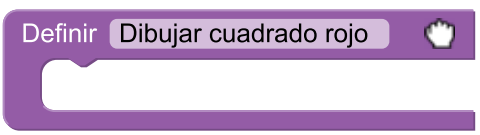
Un procedimiento es una forma de definir un comando nuevo que permite encapsular una tarea específica dentro de un programa más grande. Los procedimientos suelen usarse para descomponer problemas complejos en piezas más simples. Además, son útiles para evitar repetir secuencias de instrucciones idénticas en los programas.

3. Guardá el proyecto con un nombre distinto para conservar la solución, y modificá tu programa para que ahora los cuadrados sean de 4 × 4.



¿Qué cambios tendrías que hacer para que los cuadrados fueran verdes en lugar de rojos?

Un **comando** es la descripción de una acción. Puede ser un comando básico o uno definido mediante un procedimiento. Como los comandos describen acciones, al definir uno nuevo el nombre tiene que comenzar con un verbo; en este caso, usamos *Dibujar*.



Secuencia Didáctica 2

ESTRATEGIAS Y VOCABULARIO

El uso de procedimientos mejora la calidad y la claridad de los programas. Al nombrar los procedimientos de forma adecuada, se pueden alcanzar soluciones que quedan expresadas en términos del vocabulario del dominio del problema abordado. Pero para lograrlo, es necesario tener una idea previa sobre cómo abordar el problema como primer paso para resolverlo. Nos referimos a esto como **estrategia de solución**. Una forma habitual de explicitar una estrategia consiste en descomponer una tarea en tareas más pequeñas. Por ejemplo, si la tarea es planear un viaje, algunas subtareas podrían ser elegir el lugar, el medio de transporte y el alojamiento, pensar qué cosas llevar, etc. Cada una de esas subtareas, a su vez, podría descomponerse en otras más pequeñas. En programación es importante realizar una adecuada **división en subtareas** y expresarlas mediante procedimientos.

OBJETIVOS

- Comprender la importancia de los nombres asignados a los procedimientos.
- Concebir estrategias para resolver problemas.
- Poner en práctica la división en subtareas.
- Usar procedimientos que reflejen una estrategia de solución en la construcción de programas.

Actividad 1
El alienígena toca el botón

DE A DOS

OBJETIVOS

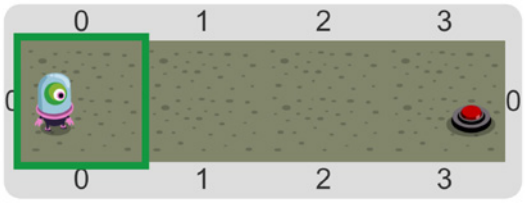
- Programar la solución de un problema con un dominio distinto al de Gobstones.
- Presentar la biblioteca de procedimientos.
- Reflexionar sobre la relación entre vocabulario y procedimientos.
- Presentar las vestimentas de Gobstones.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

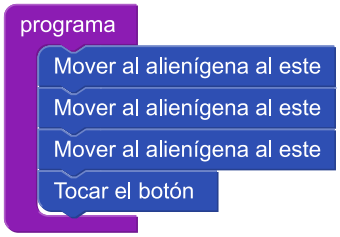
En esta actividad, los estudiantes trabajarán sobre dos proyectos de Gobstones. En el primero usarán procedimientos que están integrados al proyecto, pero que no forman parte de los comandos básicos del lenguaje. Se han programado previamente y están disponibles para usarlos. Luego, en el segundo proyecto, tendrán que encargarse de programar ellos mismos los procedimientos que en el primero estaban dados.



Tablero inicial de "El alienígena toca el botón"

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto "El alienígena toca el botón". Se encontrarán con un tablero de 4 × 1 en el que se observa a un alienígena en la celda (0,0) y un botón en la (3,0). El objetivo es que el alienígena se desplace hacia la derecha hasta llegar a la posición del botón y lo presione.

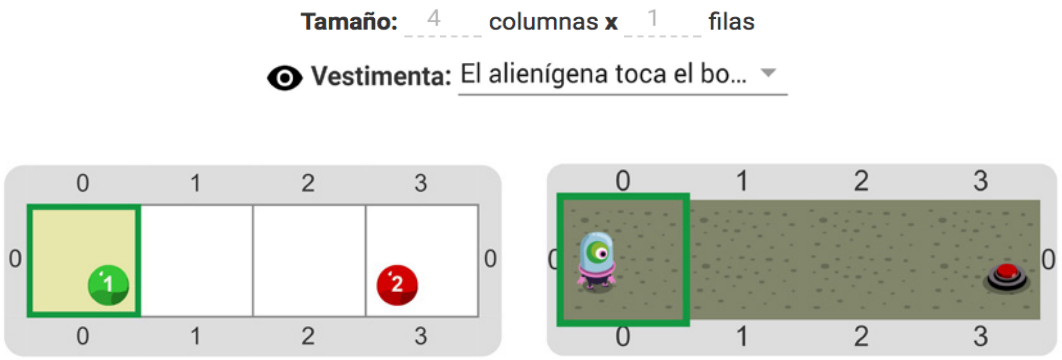
Dejamos que los estudiantes exploren el entorno para que descubran las opciones disponibles en este proyecto. Así, observarán que en *Biblioteca > Procedimientos* se encuentran los procedimientos `Mover al alienígena al este` y `Tocar el botón`. Luego de que todos hayan examinado el entorno, tienen que resolver la consigna. Es probable que lleguen a la solución que se muestra en la imagen.



Programa que resuelve el desafío

Una vez que hayan finalizado, les comentamos: "En este caso, nos encontramos con opciones que no son las que habitualmente están disponibles en el lenguaje Gobstones. Se trata de procedimientos que programó otra persona y los integró en el proyecto para resolver el desafío. Sin embargo, no debemos perder de vista que los podríamos haber hecho nosotros. Por otro lado, aquí no hay bolitas: nos topamos directamente con un extraterrestre y un botón. Esto se debe a que usamos lo que en Gobstones se conoce como *vestimentas*, que nos permiten cambiarle el aspecto al tablero".

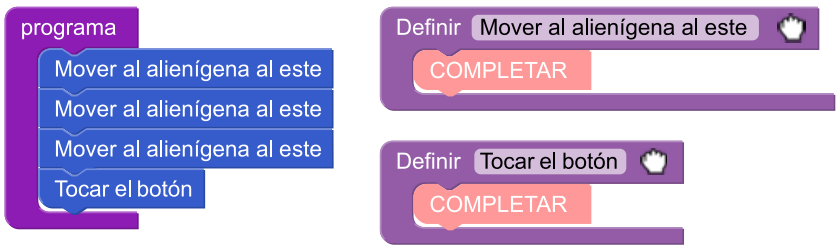
A continuación les indicamos que carguen el proyecto "Armamos 'El alienígena toca el botón'". El objetivo, en este caso, es que programen los procedimientos `Mover al alienígena al este` y `Tocar el botón`, que en el proyecto anterior estaban disponibles en la biblioteca. Se encontrarán con un tablero de 4 × 1 con una bolita verde en la celda (0,0) y dos rojas en la (3,0). Les indicamos que presionen varias veces el ojo, que hará aparecer y desaparecer la vestimenta.



Tablero de "Armamos 'El alienígena toca el botón'" sin y con vestimenta

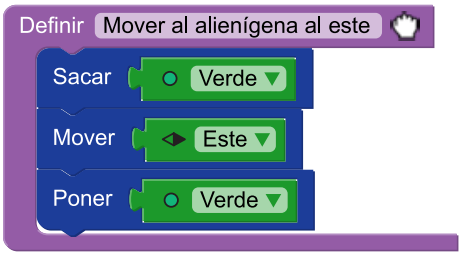
Comentamos: "Lo que acaban de hacer es activar y desactivar una vestimenta. Las vestimentas nos permiten ver de diversas formas el tablero. Por ejemplo, en este caso, vestimos el tablero con un extraterrestre y un botón. Internamente, las vestimentas asocian bolitas con imágenes. Una bolita verde está asociada con un alienígena y dos bolitas rojas, con el botón sin presionar. Además, aunque aquí no lo veamos, una bolita roja está asociada con el botón presionado. Sin embargo, en Gobstones, lo único que existen son las bolitas, que es lo que vemos cuando ocultamos la vestimenta".

En el espacio del programa van a observar que ya está programado el cuerpo principal, pero falta completar los procedimientos `Mover al alienígena al este` y `Tocar el botón`.



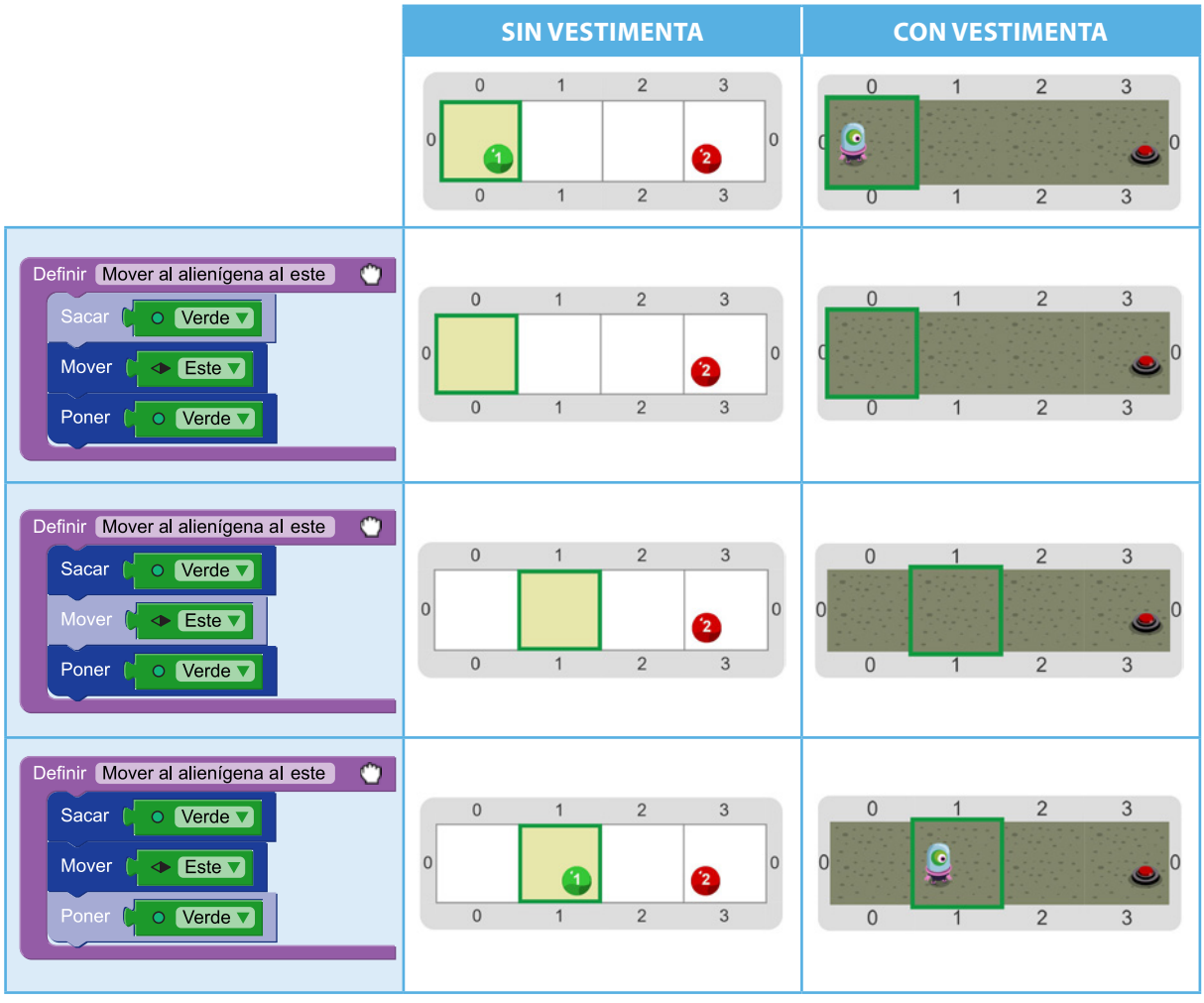
Espacio del programa al abrir el proyecto "Armamos 'El alienígena toca el botón'"

Al programar los procedimientos tendrán que operar el cabezal de Gobstones y tener presentes las asociaciones entre bolitas e imágenes definidas en la vestimenta del proyecto: una bolita verde representa al alienígena, dos bolitas rojas al botón sin presionar y una bolita roja, al botón presionado. Para completar `Mover al alienígena al este` hay que *(i)* sacar una bolita verde –para retirar al extraterrestre de la celda que se encuentra bajo el cabezal–, *(ii)* mover el cabezal hacia el este y *(iii)* poner una bolita verde –para que aparezca el alienígena–.



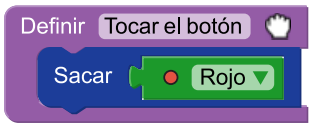
Procedimiento
`Mover al alienígena al este`

Se muestra a continuación el tablero con y sin vestimenta a medida que se ejecutan las instrucciones del procedimiento.



Ejecución del procedimiento `Mover al alienígena al este`

Para completar el desafío solo falta programar `Tocar el botón`. Al observar el cuerpo principal del programa puede verse que este es invocado una vez que el alienígena —y el cabezal— ya se encuentran sobre la celda (0,3) del tablero, que es donde está el botón. Como el botón sin presionar se representa con dos bolitas rojas y el botón presionado con una, alcanza con que el cabezal retire una de ellas.



Procedimiento `Tocar el botón`

Ejecución del procedimiento `Tocar el botón`

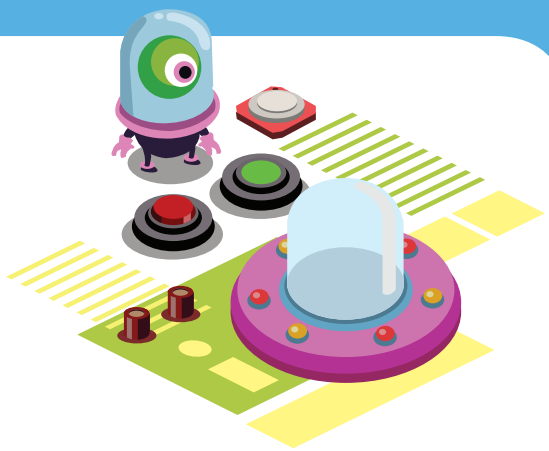
CIERRE

Reflexionamos con los estudiantes acerca de lo trabajado en la actividad. En la primera parte, contar con procedimientos con nombres adecuados nos permitió razonar en función del problema que queríamos resolver; en esta actividad, en términos del alienígena y el botón. En la segunda parte, programamos los procedimientos que antes ya estaban en la biblioteca. En este caso, tuvimos que tener presente la asociación entre bolitas e imágenes de la vestimenta provista.

NOMBRE Y APELLIDO:

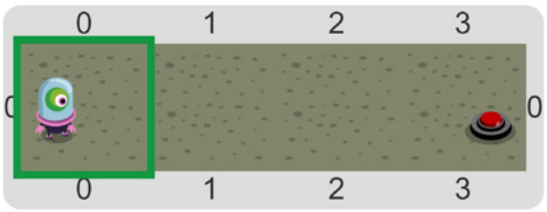
CURSO: FECHA:

EL ALIENÍGENA TOCA EL BOTÓN



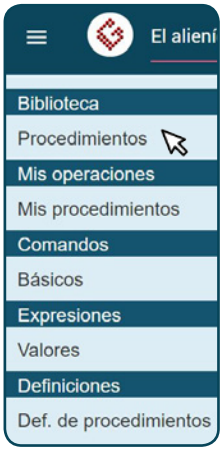
En esta actividad hay un alienígena que, para poder activar su nave, tiene que apretar un botón. Tu trabajo es fácil: escribí un programa que haga que el extraterrestre llegue hasta el botón y lo apriete.

1. Abrí el proyecto "El alienígena toca el botón", explorá el entorno y fijate cómo podés resolver el desafío.

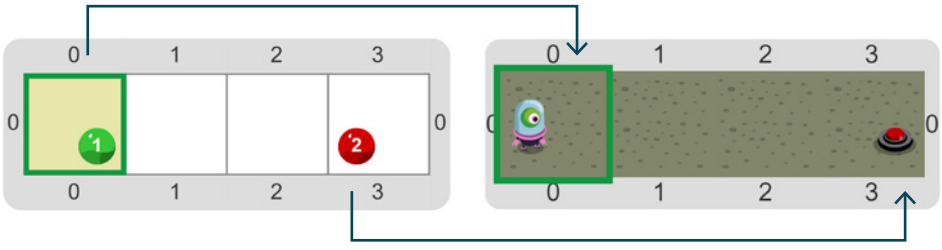


BIBLIOTECA

¿En qué se diferencian los comandos básicos de Gobstones y los procedimientos de la biblioteca? Los comandos, como `Poner [ ]`, `Sacar [ ]` y `Mover [ ]` son parte del lenguaje. En cambio, los procedimientos disponibles en la biblioteca solo pueden usarse en algunos proyectos. De hecho, ¡alguien los programó usando Gobstones!



2. Abrí "Armamos 'El alienígena toca el botón'". Ahora tenés que ocuparte de armar los procedimientos que antes ya estaban en la biblioteca: `Mover al alienígena al este` y `Tocar el botón`. Después de todo, alguien tiene que programarlos, ¿no?
Presioná varias veces el ojo para activar y desactivar la vestimenta. Para completar los procedimientos, tené en cuenta que esta vestimenta representa al alienígena con una bolita verde, al botón sin presionar con dos bolitas rojas y al botón presionado con una roja.



VESTIMENTAS

En Gobstones, las vestimentas nos permiten visualizar de diversas formas el tablero. Por ejemplo, en este caso, vestimos al tablero con un extraterrestre y un botón. Internamente, las vestimentas asocian bolitas con imágenes.

Vestimenta

Actividad 2
El Beto, el robot goleador

DE A DOS

OBJETIVOS

- Mostrar la ventaja de usar procedimientos al programar.
- Destacar la importancia de elegir nombres adecuados para los procedimientos.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

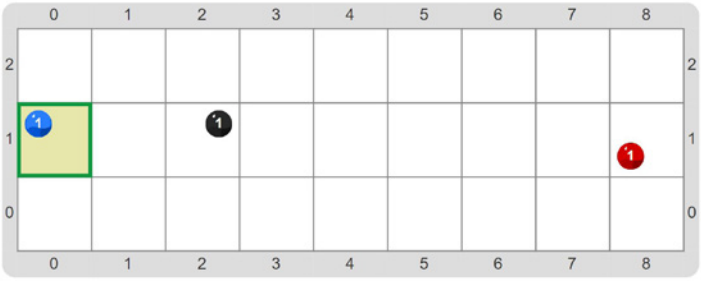
DESARROLLO

En esta actividad, los estudiantes trabajarán con el proyecto "El Beto, el robot goleador". Está compuesto por una serie de procedimientos, algunos ya están construidos y otros deberán completarlos los estudiantes. El objetivo es que un robot llamado el Beto patee la pelota tres veces hasta hacer un gol.

EN HONOR A LA PELÍCULA *METEGOL*

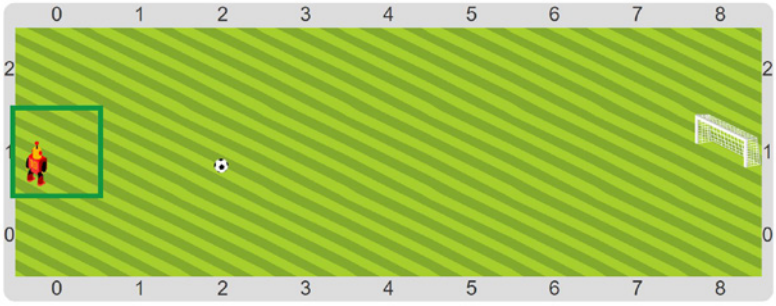
Inspirado en el cuento de Roberto Fontanarrosa "Memorias de un *wing* derecho", Juan José Campanella dirigió *Metegol*, la primera película de animación en 3D realizada en el país. El Beto es uno de los jugadores de fútbol que acompaña al protagonista y que, de alguna manera, se homenajea en este proyecto de Gobstones.

Les repartimos las fichas a los estudiantes y les indicamos que abran el proyecto. Al hacerlo, se encontrarán con un tablero que tiene una bolita azul en la posición (0,1), una negra en la (2,1) y una roja en la (8,1). Les explicamos que la bolita azul representa al robot el Beto, la negra a una pelota de fútbol y la roja a un arco.



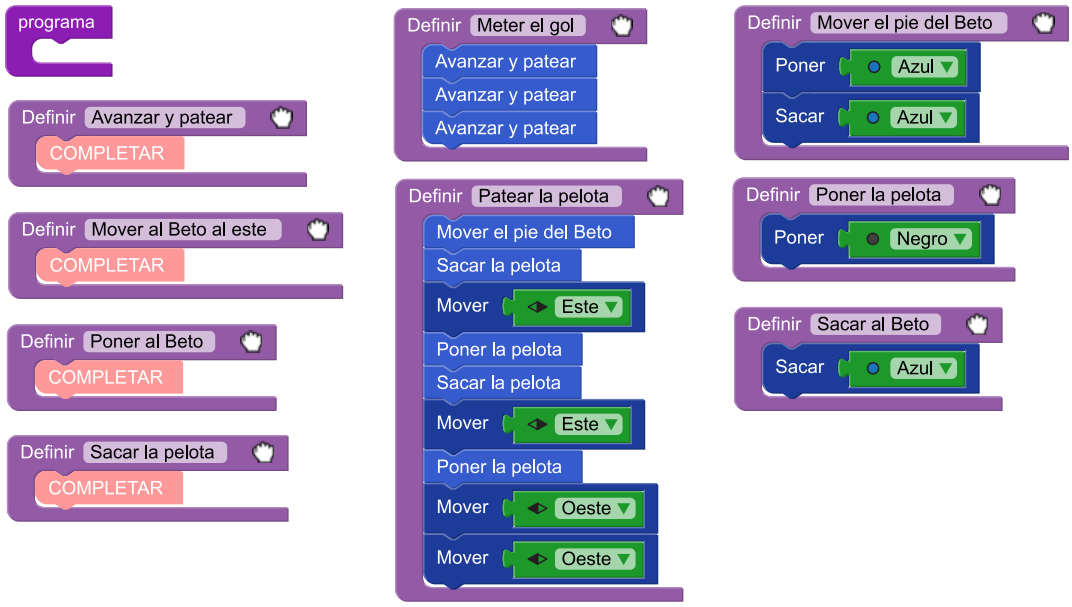
Tablero inicial

Para que comprendan cómo se visualiza la representación usando la vestimenta que viene incorporada en el proyecto, les sugerimos que presionen el ojo. Entonces verán que el tablero se transforma en un campo de fútbol y sobre él aparecen el Beto, la pelota y el arco. Sin entrar en detalles, les recordamos que, en definitiva, las vestimentas son asociaciones entre las bolitas de las celdas del tablero e imágenes dispuestas sobre estas. A partir de esta información, algún estudiante podría reconocer que en la vestimenta de esta actividad una celda sin bolitas representa un cuadrado de césped.



Tablero inicial con vestimenta

En el espacio del programa encontrarán una serie de procedimientos: 5 ya están construidos y otros 4 tienen que completarse. Los que ya están hechos son `Sacar al Beto`, `Mover el pie del Beto`, `Poner la pelota`, `Patear la pelota` y `Meter el gol`. Los que hay que completar son `Sacar la pelota`, `Poner al Beto`, `Mover al Beto al este` y `Avanzar y patear` –además del cuerpo principal del programa, `programa`–.

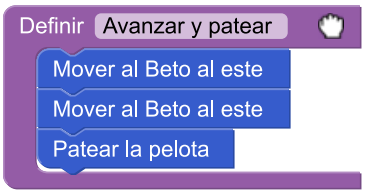


Espacio del programa al cargar el proyecto

Una forma práctica de abordar el desafío consiste en guiarnos por los nombres de los procedimientos para completar las partes que faltan. Como lo que tiene que hacer el programa es conseguir que el Beto meta un gol, el cuerpo principal del programa consiste en una única invocación a `Meter el gol`.

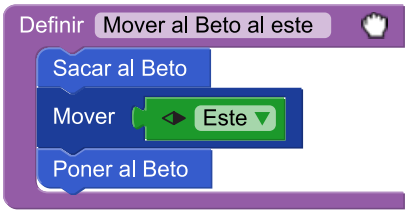
Cuerpo principal del programa

El procedimiento `Meter el gol`, que ya viene construido en el proyecto, invoca tres veces `Avanzar y patear`. En primer lugar, el Beto tiene que llegar hasta la pelota: para esto tiene que moverse dos celdas a la derecha, lo que se resuelve invocando dos veces `Mover al Beto al este` –aún incompleto–. Luego, simplemente nos ocupamos de que patee la pelota usando `Patear la pelota`.



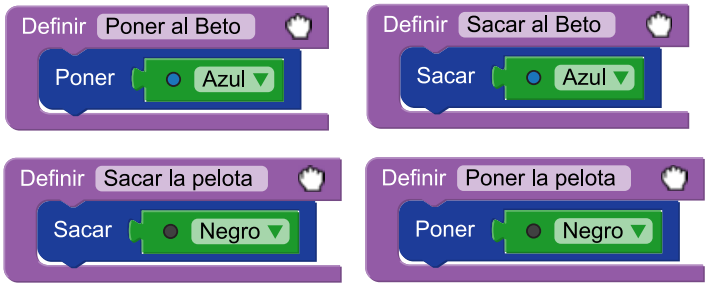
Procedimiento `Avanzar y patear`

Para `Mover al Beto al este` debemos sacar al robot de la celda donde se encuentra, mover el cabezal una celda a la derecha y ubicar allí al robot. Para ello, invocamos `Sacar al Beto`, `Mover [Este]` y `Poner al Beto`.



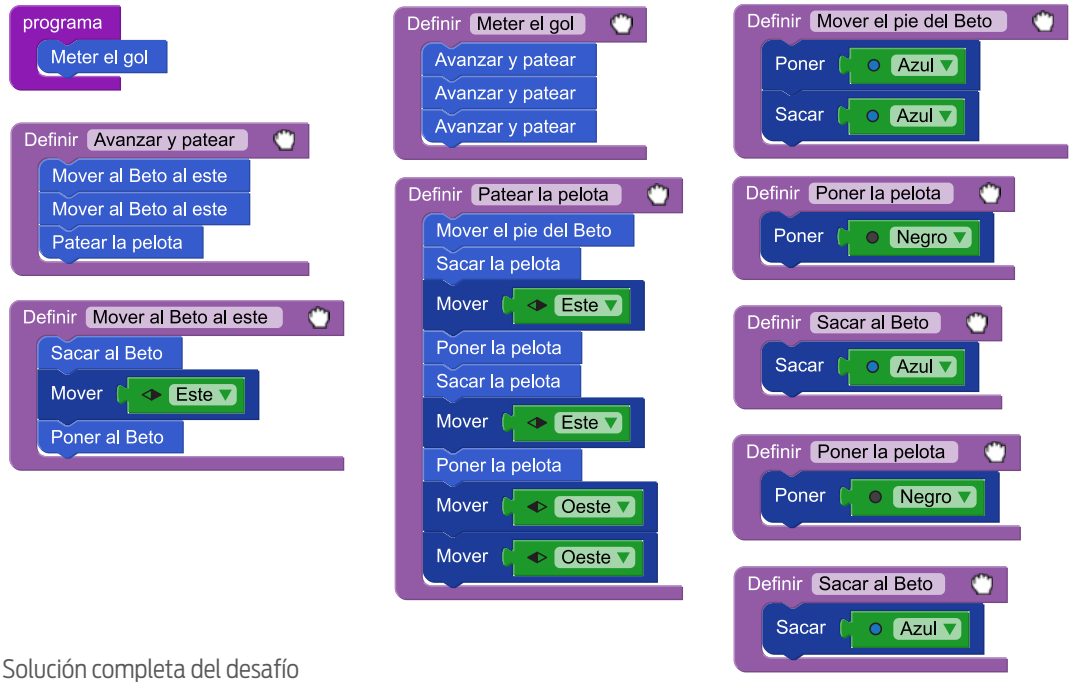
Procedimiento `Mover al Beto al este`

Todavía falta completar `Poner al Beto` y `Sacar la pelota`. Al hacerlo, en ambos casos –y al igual que como están resueltos sus procedimientos duales `Sacar al Beto` y `Poner la pelota`–, debemos tener presente la asociación entre bolitas e imágenes: el Beto está representado por una bolita azul y la pelota por una negra. Por lo tanto, para que aparezca el robot tenemos que depositar una bolita azul sobre el tablero, y para sacar la pelota hay que retirar una negra.



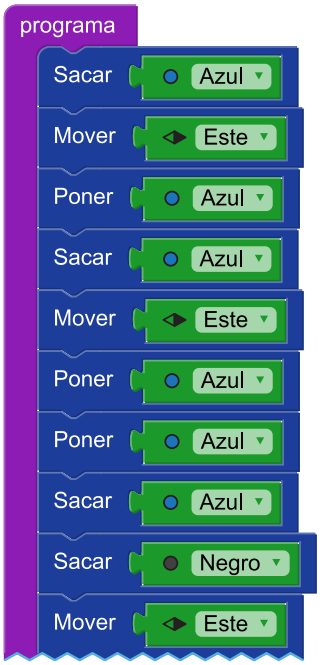
Procedimientos para poner y sacar al Beto y a la pelota

A continuación se observa el programa completo que resuelve el desafío:



Solución completa del desafío

Una vez que los estudiantes hayan completado sus programas, conversamos con ellos sobre la importancia de usar procedimientos. Cada procedimiento engloba una unidad de sentido del problema abordado. Sus nombres se expresan usando el vocabulario del problema y son muy descriptivos, y en consecuencia obtenemos un programa que se lee con facilidad y es sencillo de entender. Además, al usar los procedimientos, descompusimos un problema complejo en porciones más sencillas que, al combinarse, resolvieron el problema original. "¿Se imaginan cómo sería el programa si lo hubiésemos resuelto sin usar procedimientos?". Luego de discutir con ellos las respuestas, les pedimos que abran el proyecto "El Beto, el robot goleador, sin procedimientos",



Fragmento de un programa que resuelve el problema sin usar procedimientos

donde se resuelve el mismo problema sin usar procedimientos. De este modo, se puede ver que hace falta disponer 48 instrucciones, una detrás de otra. Es evidente que ese programa es difícil de entender y, por lo tanto, de corregir en caso de que tenga errores.

CIERRE

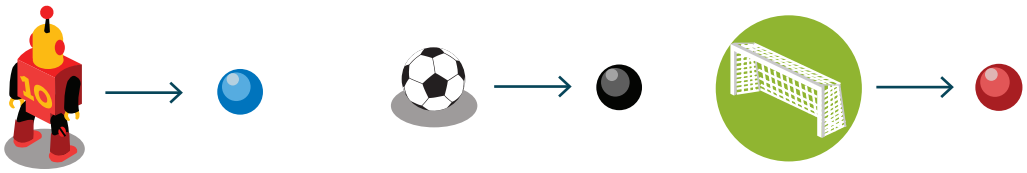
Como conclusión, repasamos con los estudiantes la importancia de utilizar procedimientos al programar. Por un lado, esto nos permite expresar soluciones a problemas usando el vocabulario inherente del problema abordado. Por otro lado, nos permite crear programas modulares que son fáciles de comprender y de corregir en caso de que contengan errores.

NOMBRE Y APELLIDO:

CURSO: FECHA:

EL BETO, EL ROBOT GOLEADOR

En esta actividad vas a completar un programa para que el robot el Beto juegue al fútbol: tiene que llevar una pelota hacia el arco y hacer un gol. En este caso, el Beto está representado con una bolita azul, la pelota con una negra y el arco, con una roja.



EN HONOR A LA PELÍCULA METEGOL

Inspirado en el cuento "Memorias de un *wing* derecho" de Roberto Fontanarrosa, Juan José Campanella dirigió *Metegol*, la primera película de animación en 3D realizada en el país. El Beto es uno de los jugadores de fútbol que acompaña al protagonista y que, de alguna manera, se homenajea en este proyecto de Gobstones.

1. Abrí el proyecto "El Beto, el robot goleador" y exploralo. Algunas cosas ya están resueltas. Indicá qué procedimientos tuviste que completar vos y, para cada uno de ellos, cuáles usaste de los que ya vinieron programados.

DOS AYUDITAS...

- Para completar `Poner al Beto` y `Sacar la pelota`, fijate cómo están hechos `Sacar al Beto` y `Poner la pelota`.
- Asumí que cada vez que empieza un procedimiento, el cabezal se encuentra posicionado sobre el Beto.

Actividad 3
La gran aventura del mar encantado

DE A DOS

OBJETIVOS

- Definir procedimientos.
- Comprender la importancia de los nombres de los procedimientos.
- Descomponer problemas en subproblemas de menor complejidad.

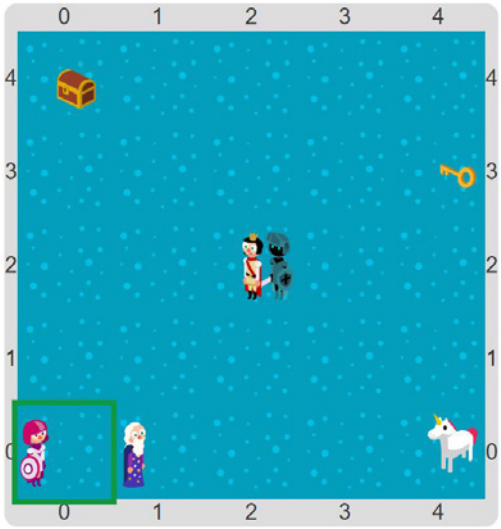
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

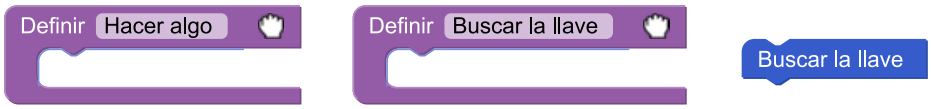
A lo largo de esta actividad, los estudiantes van a descomponer un problema en problemas más sencillos que luego combinarán para resolver el problema original. La ficha de la actividad cuenta una historia que los estudiantes deberán programar.

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que lean la consigna. Luego les solicitamos que abran el proyecto "La gran aventura del mar encantado". Se encontrarán con un tablero que tiene distintos elementos: un cofre, una llave, un príncipe atrapado por un caballero de traje negro, una princesa, un mago y un unicornio. El desafío es que los estudiantes hagan un programa para que la princesa rescate al príncipe y huyan juntos en el unicornio. Para alcanzar el objetivo, la princesa debe realizar una serie de acciones en el siguiente orden: *(i)* recoger la llave; *(ii)* ir hasta el cofre, usar la llave para abrirlo y retirar de su interior un sombrero mágico; *(iii)* dirigirse hasta donde se encuentra el mago e intercambiar el sombrero por una espada encantada; *(iv)* liberar al príncipe atacando al caballero con la espada; y *(v)* dirigirse hasta el unicornio y huir con el príncipe. Al tratarse de etapas bien diferenciadas, una solución adecuada consiste en definir un procedimiento para cada una.



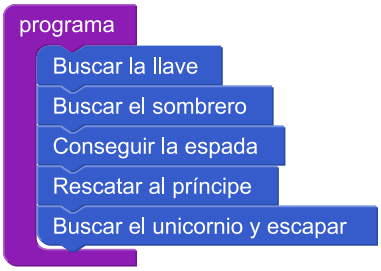
Tablero inicial de "La gran aventura del mar encantado"

Para definir un procedimiento nuevo hay que ir a *Definiciones* > *Definición de procedimientos* en el menú. Allí hay que tomar el bloque `Definir [Hacer algo]` y arrastrarlo hasta el espacio del programa. Para cambiarle el nombre, basta hacer clic sobre el texto *Hacer algo* y escribir allí el nuevo nombre. Retomando el desafío propuesto, un buen nombre para la primera acción que debe realizar la princesa es *Buscar la llave*. Una vez hecho esto, en *Mis operaciones* > *Mis procedimientos* aparecerá el bloque `Buscar la llave`, que puede usarse como cualquier otro –aunque aún no hayamos programado lo que hace–.



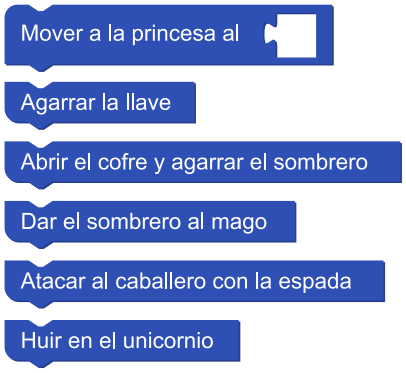
Definición de un nuevo procedimiento

Del mismo modo podemos definir procedimientos para el resto de las etapas que tiene que completar la princesa: `Buscar el sombrero`, `Conseguir la espada`, `Rescatar al príncipe` y `Buscar al unicornio y escapar`. Así completamos el cuerpo principal del programa, que de este modo describe en forma clara la estrategia elegida para resolver el desafío.



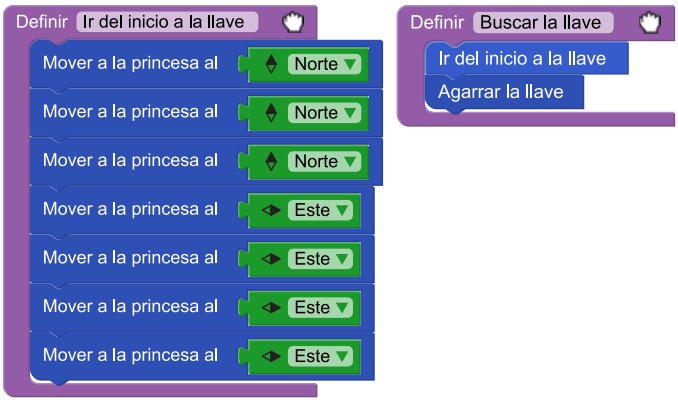
Cuerpo principal del programa

El proyecto provee una serie de procedimientos que resultan muy útiles para completar los que hemos definido nosotros, y se encuentran disponibles en *Biblioteca* > *Procedimientos*: `Mover a la princesa al [ ]`, `Agarrar la llave`, `Abrir el cofre y agarrar el sombrero`, `Dar el sombrero al mago`, `Atacar al caballero con la espada` y `Huir en el unicornio`.



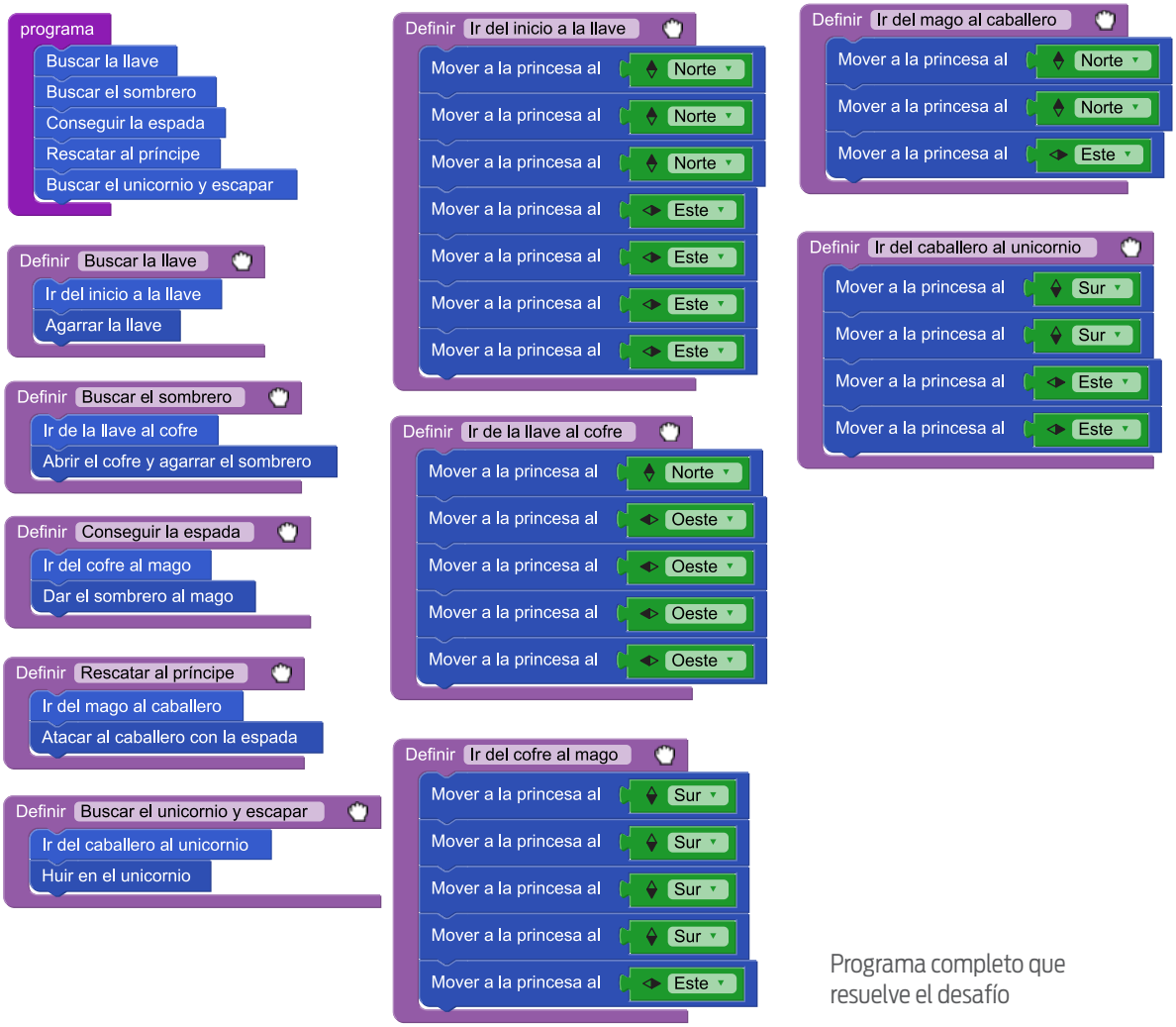
Procedimientos disponibles en *Biblioteca* > *Procedimientos*

Así como descompusimos en partes la proeza que tiene que realizar la princesa, podemos hacer lo mismo con cada una de las partes –es decir, con cada uno de los procedimientos que definimos–. Por ejemplo, `Buscar la llave` puede separarse en *(i)* dirigirse desde el inicio hasta donde se encuentra la llave y *(ii)* recogerla. Para la primera parte definimos un nuevo procedimiento `Ir del inicio a la llave`, que invoca sucesivas veces `Mover a la princesa al [ ]` (disponible en la biblioteca); para la segunda, `Agarrar la llave` (también de la biblioteca).



Procedimientos `Buscar la llave` e `Ir del inicio a la llave`

De manera similar, podemos resolver el resto de los procedimientos que definimos. A continuación se muestra una solución completa del desafío.



Programa completo que resuelve el desafío

Durante el desarrollo de la actividad, es recomendable chequear que los estudiantes dividan el problema en sus distintas etapas usando procedimientos. Además, debemos prestar atención a los nombres elegidos para los procedimientos. Al respecto, es conveniente remarcar que los procedimientos se usan para encapsular acciones, por lo que lo correcto es que sus nombres comiencen con un verbo. Si los estudiantes tienen dificultades para elaborar los procedimientos y nombrarlos adecuadamente, los guiamos para que corrijan sus propuestas y obtengan una solución prolija.

CIERRE

Para finalizar la actividad, reflexionamos grupalmente sobre la forma en que se usaron los procedimientos en este programa y los resultados obtenidos. Les recordamos a los estudiantes que, cuando se eligen y nombran correctamente los procedimientos, estos son de gran ayuda para que el programa se pueda leer y entender con facilidad. Una parte importante de la tarea de programar consiste en elegir los procedimientos que vamos a definir y determinar cómo se van a llamar. Para esto último, es útil pensar en el problema y en su dominio; es decir, en los elementos y las operaciones del problema.

NOMBRE Y APELLIDO:

CURSO: FECHA:

LA GRAN AVENTURA DEL MAR ENCANTADO

La mañana empezó mal para la princesa: el caballero negro secuestró al príncipe y ella aún no había arreglado las cosas con el mago. Hacía dos días había discutido con él, le había sacado su sombrero mágico y lo había escondido en un cofre.
Pero el mago era el único que podía darle la espada encantada que le permitiría vencer al caballero negro. La cosa no podía quedar así: tenía que salvar al príncipe. Y cumplir juntos el sueño de pasear en unicornio.

1. Hay que ayudar a la princesa a rescatar al príncipe. Ella tiene que cumplir con una serie de pruebas, en orden. Primero tiene que buscar la llave; luego, tiene que usarla para abrir el cofre y sacar el sombrero mágico; después, debe buscar al mago e intercambiar con él el sombrero mágico por la espada encantada; por último, tiene que atacar al caballero con la espada para cumplir su misión de rescate. La aventura termina cuando la protagonista lleva al príncipe hasta el unicornio y escapan juntos.
Para contar la historia, programá un procedimiento para cada parte.

2. ¿Cuántos bloques tiene el programa principal? ¿Cómo se llama cada uno?

CLAVES A TENER EN CUENTA

- El proyecto viene con vestimenta y con varios procedimientos disponibles en la biblioteca.
- Si la princesa se cae del tablero o intenta hacer alguna acción sin tener los elementos necesarios, el programa falla y te lo hace saber con un ¡BOOM!.
- Cuando alguien lea el cuerpo principal del programa, tiene que poder comprender la historia narrada.

Actividad 4
Lucho enciende las luces

DE A DOS

OBJETIVOS

- Descomponer un problema en varias partes.
- Concebir diferentes estrategias de solución para un problema.

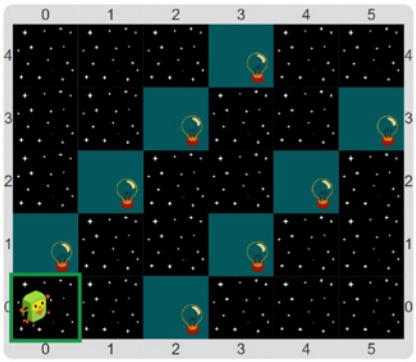
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes abordarán un problema y propondrán distintas estrategias de solución.

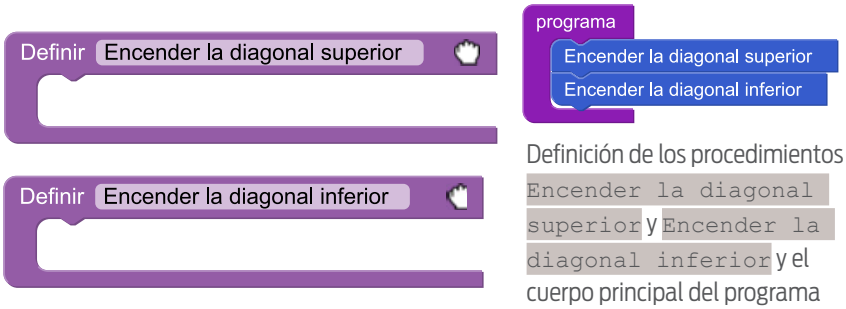
Comenzamos pidiéndoles que carguen el proyecto "Lucho enciende las luces". Les explicamos que el objetivo es que el robot Lucho se desplace por las celdas del tablero y encienda las lámparas que aparecen en las dos diagonales, que originalmente se encuentran todas apagadas.



Tablero inicial de "Lucho enciende las luces"

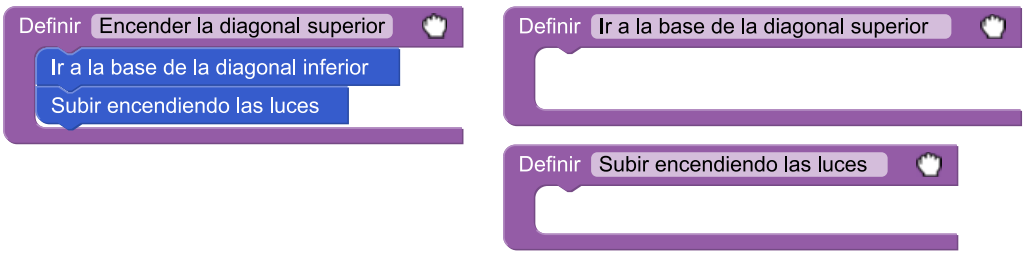
A continuación, les pedimos que, antes de ponerse a programar, piensen una estrategia de solución: "¿Qué pasos seguirían para resolver el problema?". Una respuesta posible es: "Primero me ocupo de las luces de la diagonal superior y luego de las de la diagonal inferior". En este caso, se está dividiendo el problema original en dos partes claramente diferenciables.

Una vez que los estudiantes hayan definido su estrategia general, les indicamos que un abordaje posible consiste en definir un procedimiento para cada una de las partes que componen la estrategia. Siguiendo con el ejemplo, habría que crear al menos dos procedimientos: `Encender la diagonal superior` y `Encender la diagonal inferior`. Aun cuando falte definir qué hace cada uno de ellos, ya se puede completar el cuerpo principal del programa. Más allá de la estrategia elegida, les recordamos que un indicio de que la propuesta es buena es que, al leer programa, la estrategia se entienda con facilidad.



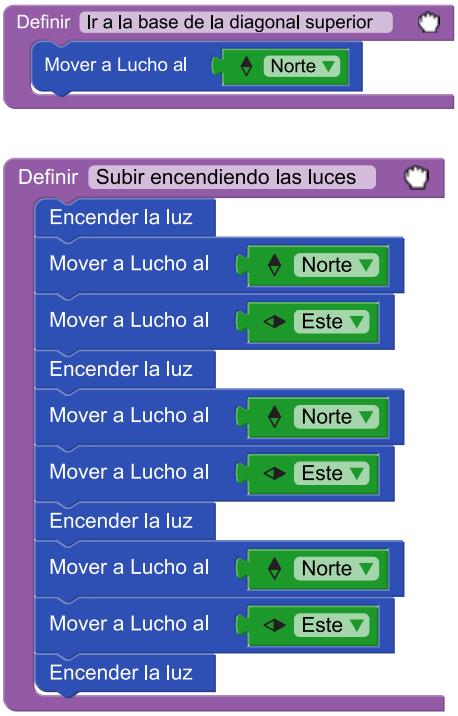
Definición de los procedimientos `Encender la diagonal superior` y `Encender la diagonal inferior` y el cuerpo principal del programa

Para completar `Encender la diagonal superior` es conveniente continuar haciendo el ejercicio de identificar las distintas partes que involucra la resolución de esta tarea. Una posibilidad consiste en ubicar a Lucho sobre la base de la diagonal y luego hacer que suba encendiendo las luces. Definimos, entonces, dos nuevos procedimientos: `Ir a la base de la diagonal superior` y `Subir encendiendo las luces`.



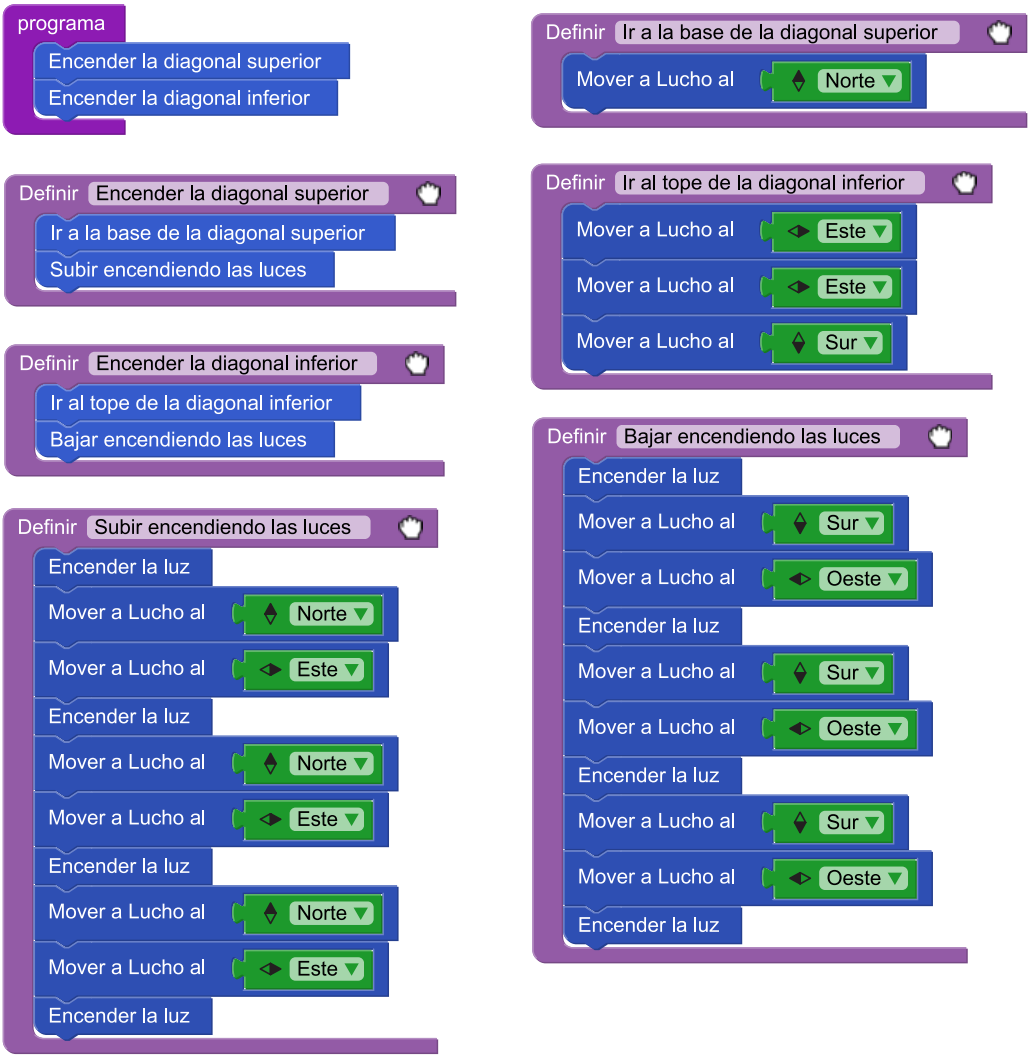
Definición y uso de los procedimientos `Ir a la base de la diagonal superior` y `Subir encendiendo las luces`

En *Biblioteca* > *Procedimientos*, hay disponibles dos procedimientos que son de gran ayuda para completar el programa: `Encender la luz` y `Mover a Lucho al [ ]`. Entonces, `Ir a la base de la diagonal superior` se resuelve invocando una vez `Mover a Lucho al [Norte]` y `Subir encendiendo las luces` en sucesivas llamadas a `Encender la luz` y `Mover a Lucho al [ ]`. De este modo, completamos la primera parte de la estrategia.



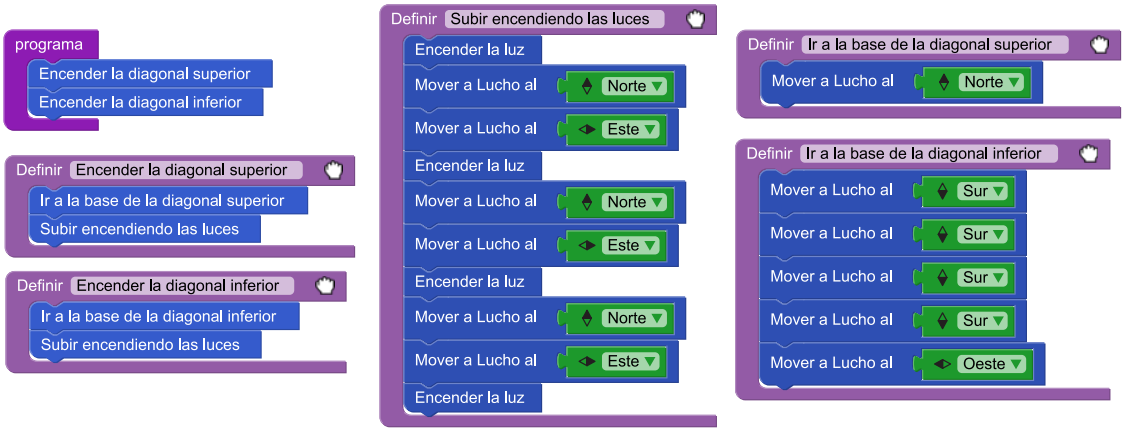
Procedimientos `Ir a la base de la diagonal superior` y `Subir encendiendo las luces`

En esta instancia hay que definir cómo encender las luces de la diagonal inferior –es decir, la segunda parte de la estrategia–. Una posibilidad es posicionarnos en el extremo superior de esta diagonal y bajar encendiendo las luces. Nuevamente, ubicamos cada una de estas partes en un procedimiento separado: `Ir al tope de la diagonal inferior` y `Bajar encendiendo las luces`. A continuación se encuentra una solución completa del desafío para la estrategia planteada:



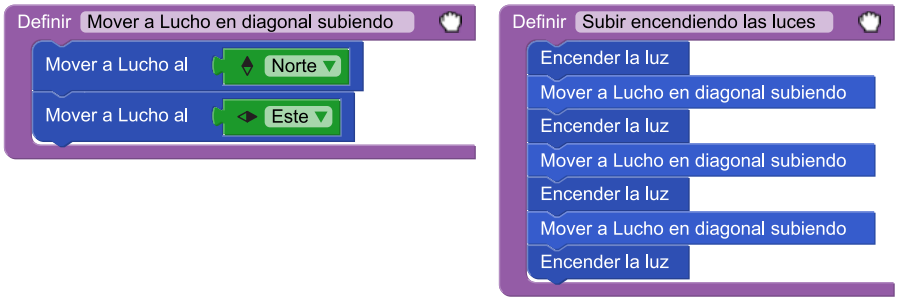
Posible solución del desafío

Otra estrategia consiste en encender las lamparitas subiendo por las dos diagonales –o bajando por ambas–; en ese caso no hacen falta los procedimientos `Bajar encendiendo las luces` e `Ir al tope de la diagonal inferior`. En cambio hay que incluir un nuevo procedimiento que ponga a Lucho en la base de la diagonal inferior: `Ir a la base de la diagonal inferior`.



Solución para la estrategia que sube por ambas diagonales

En ambas soluciones propuestas se podría haber definido un procedimiento `Mover a Lucho en diagonal subiendo` que encapsulara un movimiento en diagonal del robot. De este modo, el procedimiento `Subir encendiendo las luces` hubiese resultado aún más claro y expresivo, debido a que se hubiera evitado usar en forma reiterada `Mover a Lucho al [Norte]` y `Mover a Lucho al [Este]`.



Procedimientos `Mover a Lucho en diagonal subiendo` y `Subir encendiendo las luces`

Una vez que los estudiantes hayan completado un programa, les indicamos que lo guarden y que comiencen de nuevo, pero en este caso usando una estrategia diferente.

Se describen a continuación algunas dificultades que pueden surgir al resolver el problema y posibles maneras de abordarlas:

Los estudiantes hacen una mala elección de nombres de sus procedimientos
De ser así, la estrategia de solución no queda claramente expresada. Por ejemplo, esto sucedería si, en lugar de `Encender la diagonal superior`, escogiesen el nombre `Diagonal superior`. En este caso, les recordamos que es preferible que los nombres de los procedimientos comiencen con verbos, y que deben ser lo suficientemente claros como para que, al leer el programa, se entienda cuál es la estrategia que pensaron.

No definen procedimientos para moverse de una diagonal a la otra
Este punto es más sutil, y en una primera aproximación podría dejarse pasar. Es usual que los estudiantes coloquen las acciones de movimiento de una diagonal a la otra dentro de alguno de los procedimientos para encender luces. Si un procedimiento se llamara `Encender la diagonal superior`, no sería adecuado que incluyera instrucciones para moverse entre las diagonales. Esto se puede trabajar indicando que quizás les haga falta definir un procedimiento más.

La estrategia elegida no es clara o no resuelve el problema
Esta dificultad es la que requiere mayor atención. En este caso, hay que guiar a los estudiantes para que analicen su solución y vean que no funciona o no es clara, y ayudarlos a mejorarla guiándolos hacia alguna de las soluciones correctas.

Una vez que eligieron una estrategia de solución, no se les ocurre una segunda
En este caso, alcanza con preguntarles: "¿En qué orden encendieron las diagonales? ¿Sería posible hacerlo en otro orden?". Como vimos, al igual que en la mayoría de los programas, hay muchas otras estrategias de solución posibles. Además de las mencionadas, otras estrategias podrían ser: encender las dos diagonales al mismo tiempo, pasando de una a otra diagonal antes de subir; encender las diagonales bajando, en lugar de subiendo. Estas estrategias no son tan comunes; en caso de que algún estudiante las sugiera, deben considerarse correctas siempre que sean claras y estén expresadas con procedimientos, ya que la intención es que el programa cumpla su cometido de alguna manera, y que los procedimientos estén adecuadamente nombrados.

Luego de que todos hayan elaborado las soluciones, realizamos una puesta en común. Una posibilidad es que los estudiantes compartan las soluciones que programaron para ver si se entiende rápidamente a qué estrategias de solución corresponden. Además, se puede analizar si los nombres elegidos para los procedimientos son o no adecuados. Finalmente, si nadie construyó la solución que solo enciende las luces subiendo por las diagonales con el procedimiento `Subir encendiendo las luces`, sería interesante mostrarla y discutirla entre todos. Podemos preguntar: "¿Sería muy difícil extender estas soluciones para un tablero donde haya más diagonales?". Si se utilizase un procedimiento por cada diagonal, aumentar la cantidad de diagonales tendría como consecuencia un aumento de la cantidad

de procedimientos. En cambio con esta propuesta no haría falta, pues `Subir encendiendo las luces` se usaría para todas las diagonales.

CIERRE

Para concluir la actividad, se puede recordar que no hay una única estrategia de solución para resolver un problema; de hecho, para algunos problemas puede haber muchísimas estrategias diferentes. También es importante que los estudiantes comprendan que, si nombran adecuadamente los procedimientos, es fácil entender cuál es la estrategia elegida al leer el programa. En este sentido, deben tener presente que los programas no se usan solamente para guiar el funcionamiento de las máquinas, sino también para comunicar ideas entre diferentes programadores. Cuando eso sucede, se dice que el programa es legible.

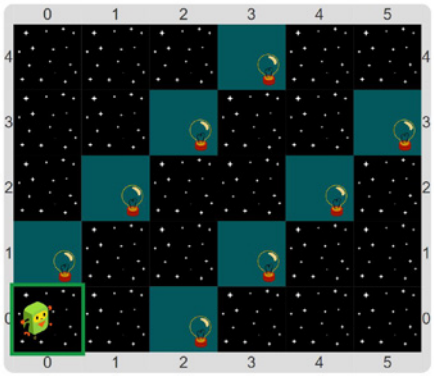
NOMBRE Y APELLIDO:

CURSO: FECHA:

LUCHO ENCIENDE LAS LUCES

En esta actividad vamos a guiar a Lucho, un robot hogareño, para que encienda las luces de dos caminos paralelos.

1. Abrí el proyecto "Lucho enciende las luces". El tablero inicial es el siguiente:



CONSEJO

Resolvé cada parte de la estrategia que pensaste en un procedimiento separado. ¡Y recordá elegir nombres descriptivos!

Pensá una estrategia para que el robot encienda todas las lamparitas y escribila acá abajo con tus propias palabras:

2. Programá la estrategia que pensaste. Escribí acá abajo el cuerpo principal del programa.

Al leerlo, ¿queda claramente expresada la estrategia que pensaste?

NOMBRE Y APELLIDO:

CURSO: FECHA:

3. ¿Te atrevés a armar otro programa que resuelva el problema con una estrategia distinta? Escribí acá abajo el cuerpo principal del programa y fijate si la estrategia que pensaste queda clara.

NO EXISTE "LA" SOLUCIÓN

Para solucionar un problema, puede haber más de un programa que sirva. O sea, no existe "la" solución. ¡Hay MUCHAS soluciones posibles!

Secuencia Didáctica 3

REPETICIÓN SIMPLE

Es habitual que, al escribir programas, nos enfrentemos a la necesidad de repetir varias veces una serie de instrucciones. Para facilitar esta tarea, casi todos los lenguajes de programación ofrecen la posibilidad de expresar repeticiones sin necesidad de reiterar instrucciones en forma explícita.

En esta secuencia didáctica se trabaja sobre **repeticiones simples**, que consisten en repetir instrucciones un número fijo de veces. Las actividades propuestas integran esta herramienta con temas previamente estudiados, como el uso de procedimientos y las estrategias de solución.

OBJETIVOS

- Presentar la noción de repetición simple.
- Combinar la repetición simple con otros comandos.
- Mostrar casos de borde en repeticiones.
- Encapsular repeticiones en procedimientos para evitar que ocurran anidadas en forma explícita.

Actividad 1
Muchas bolitas rojas

DE A DOS

OBJETIVO

- Presentar el uso de repeticiones en programas.

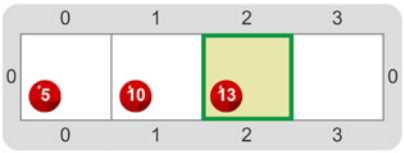
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

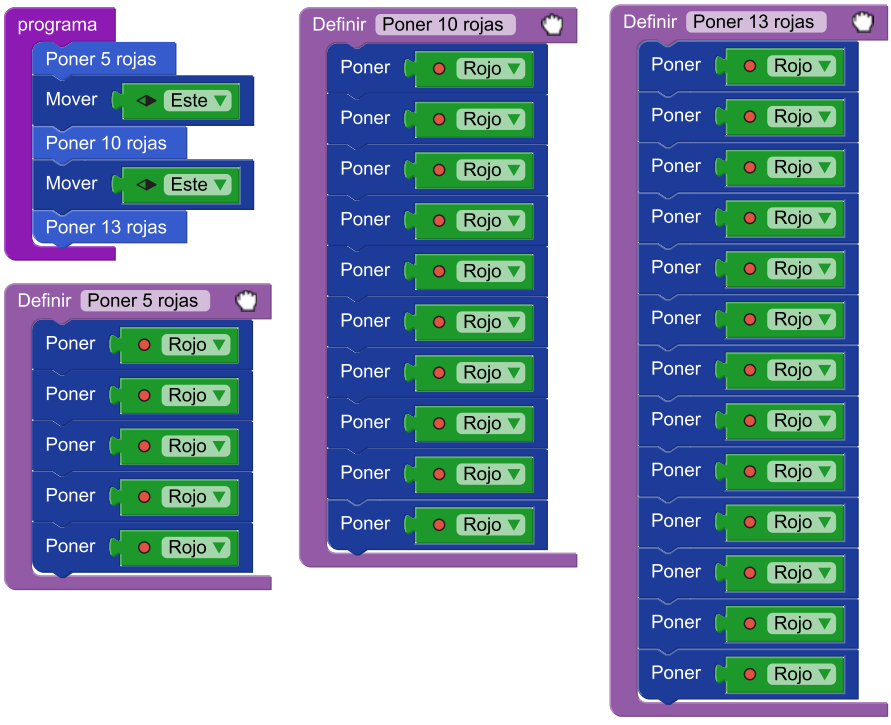
En esta actividad, los estudiantes se enfrentarán a una serie de problemas cuyas soluciones requieren que algunas instrucciones se repitan una cierta cantidad de veces. Si bien la primera consigna puede resolverse con lo presentado hasta el momento, a medida que la actividad avanza se pone de manifiesto la necesidad de contar con un nuevo comando que permita repetir una serie de instrucciones.

Comenzamos pidiéndoles que carguen el proyecto "Muchas bolitas rojas" y que resuelvan la primera consigna de la ficha. Allí se solicita que definan tres procedimientos distintos: `Poner 5 rojas`, `Poner 10 rojas` y `Poner 13 rojas`. Luego, usando estos procedimientos deben construir un programa que, dado un tablero de 4 × 1 vacío, lo transforme de forma tal que queden 5 bolitas rojas en la celda de la posición (0,0), 10 en la de (1,0) y 13 en la de (2, 0).



Tablero final de la primera consigna

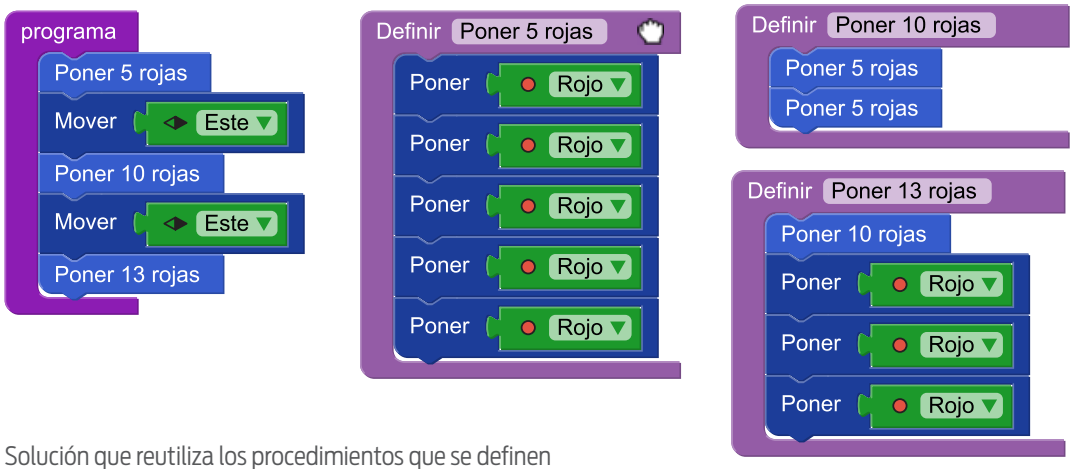
Es posible que algunos estudiantes construyan los procedimientos `Poner 5 rojas`, `Poner 10 rojas` y `Poner 13 rojas` con secuencias de 5, 10 y 13 usos de `Poner[Rojo]`.



Posible propuesta de solución para la primera consigna

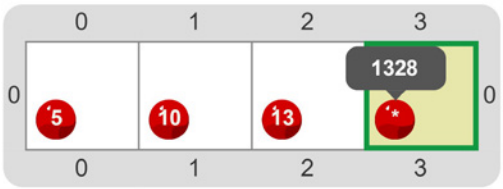
Si bien este programa es correcto para alcanzar el tablero final que se pide, también puede indicar que posiblemente los estudiantes no hayan comprendido que los procedimientos que definen en sus programas pueden usarse del mismo modo que usan los comandos que provee Gobstones. A quienes hayan llegado a esta solución, les sugerimos que piensen una solución alternativa en la que algunos de los procedimientos invoquen otros de los que ellos mismos van definiendo.

Una solución más sintética, que reutiliza los procedimientos que se definen en la actividad, consiste en resolver `Poner 10 rojas` invocando dos veces `Poner 5 rojas` y, del mismo modo, `Poner 13 rojas` invocando primero `Poner 10 rojas` y luego tres veces `Poner [Rojo]`.



Solución que reutiliza los procedimientos que se definen

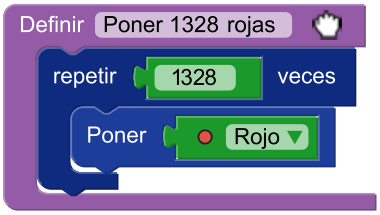
Una vez que todos hayan completado sus programas, les pedimos que resuelvan la segunda consigna de la ficha. En este caso se les pide que creen el procedimiento `Poner 1328 rojas` y que lo usen para depositar esa cantidad de bolitas rojas en la celda de la posición (3,0).



Tablero final de la segunda consigna

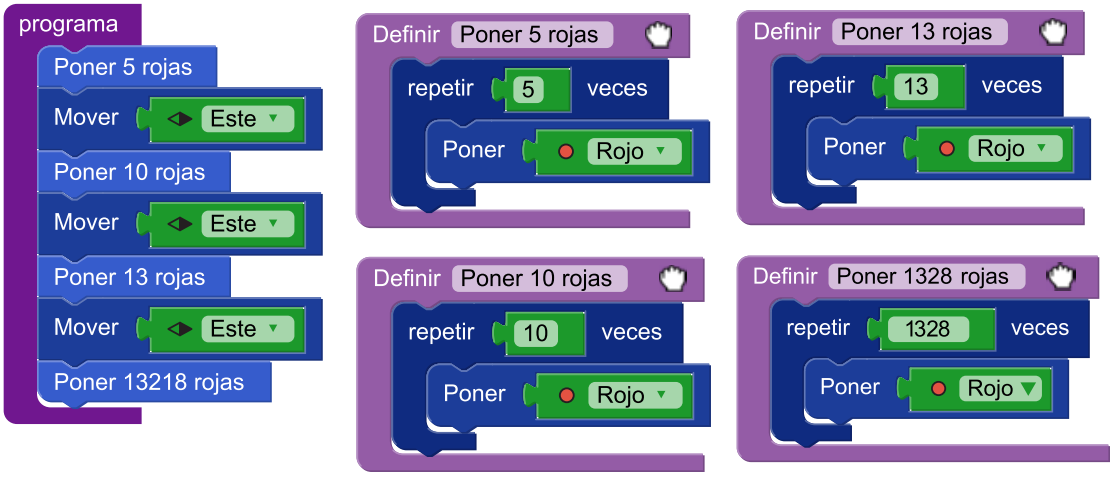
Luego de unos minutos, preguntamos: "¿Pudieron resolver el desafío?". Es probable que nadie lo haya hecho. En primer lugar, no es sensato encastrar 1328 veces `Poner [Rojo]`. En segundo lugar, tampoco es práctico en este caso reutilizar los procedimientos ya definidos, ya que para poder determinar cuántas veces se debe utilizar cada uno hay que hacer cuentas muy engorrosas. Les sugerimos, entonces, que exploren el entorno y busquen algún bloque que los ayude a resolver el problema de otro modo.

En la categoría *Comandos* > *Repeticiones* del menú lateral izquierdo se encuentra el bloque `repetir [ ] veces`. Este permite repetir una secuencia de instrucciones tantas veces como queramos. Para indicar el número de repeticiones se debe encastrar un valor numérico en el hueco que se encuentra entre *repetir* y *veces*. En la categoría *Expresiones* > *Valores* hay un bloque que permite trabajar con números, cuyo valor por defecto es 0. Combinando estos bloques con `Poner [Rojo]` podemos construir el procedimiento `Poner 1328 rojas`.



Procedimiento que usa `repetir [ ] veces`

Finalmente, les pedimos que resuelvan la tercera y última consigna de la ficha, en la que se pide que construyan todos los procedimientos usando el bloque `repetir [ ] veces`.



Solución de la tercera consigna

CIERRE

Para concluir la actividad, les comentamos a los estudiantes que al programar es muy frecuente que tengamos que repetir instrucciones. Al igual que en Gobstones, en casi todos los lenguajes de programación existen herramientas que nos permiten indicar que ciertas instrucciones deben repetirse, delegando en la máquina su ejecución. Repetir muchas veces "a mano" una misma secuencia de instrucciones puede volverse extremadamente tedioso y, además, es muy fácil perderse y cometer errores.

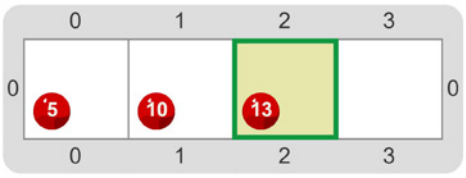
NOMBRE Y APELLIDO:

CURSO: FECHA:

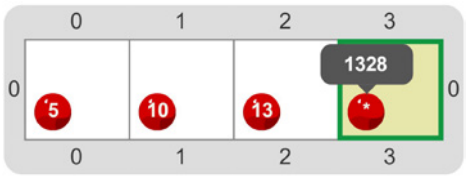
MUCHAS BOLITAS ROJAS

En esta actividad vas a tener que poner muchísimas bolitas rojas.

1. Abrí el proyecto "Muchas bolitas rojas" y definí los procedimientos `Poner 5 rojas`, `Poner 10 rojas` y `Poner 13 rojas`. Luego, armá un programa que los use para obtener, a partir de un tablero inicial vacío de 4 × 1, el siguiente tablero final.



2. Agregá a tu programa un procedimiento `Poner 1328 rojas` para alcanzar el tablero final que se muestra a continuación.



3. Ahora, armá todos los procedimientos usando el bloque `repetir [ ] veces`.

¡MENOS MAL!

¿Sabías que al programar es muy habitual tener que repetir instrucciones? Por suerte, casi todos los lenguajes de programación nos permiten hacerlo sin que nosotros lo tengamos que hacer a mano.

Actividad 2
El final del pasillo

DE A DOS

OBJETIVO

- Combinar repeticiones y otros comandos en un procedimiento.

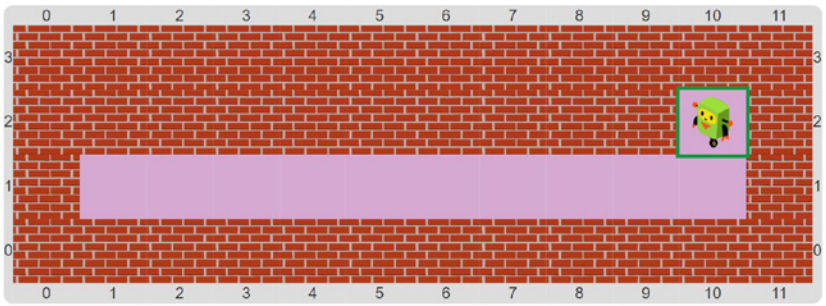
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

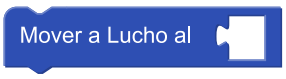
Esta actividad es de un nivel de complejidad parecido al de la actividad anterior. Sin embargo, se diferencia en que combina el comando `repetir [ ] veces` con otro comando en un mismo procedimiento.

Comenzamos repartiéndoles las fichas a los estudiantes y les pedimos que abran el proyecto "El final del pasillo". Al hacerlo, se encontrarán con un tablero de 12 × 4, sobre el cual se observa un pasillo en forma de L con el robot Lucho ubicado en uno de sus extremos.



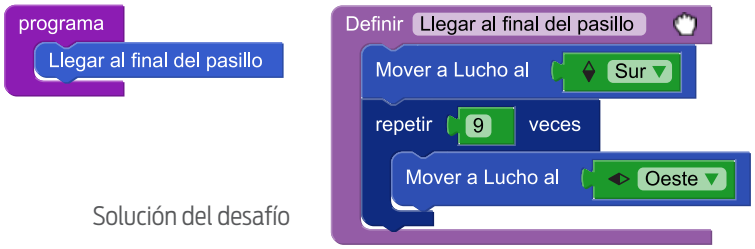
Tablero inicial de "El final del pasillo"

La consigna consiste en crear un procedimiento llamado `Llegar al final del pasillo` que mueva a Lucho de un extremo al otro del corredor. Para conseguir el objetivo, el proyecto contiene en *Biblioteca* > *Procedimientos* el procedimiento `Mover a Lucho al [ ]`, que desplaza al robot una celda en la dirección recibida como argumento. Por ejemplo, la invocación `Mover a Lucho al [Sur]` lo mueve una celda hacia abajo, mientras que `Mover a Lucho al [Oeste]` lo hace hacia la izquierda.



Procedimiento disponible en la biblioteca

Para resolver el desafío, primero hay que desplazar a Lucho una posición en dirección sur y luego nueve veces hacia el oeste. Se espera que lo realicen usando una repetición simple.



Solución del desafío

Podría ocurrir que algunos estudiantes, en lugar de usar el comando `repetir [ ] veces`, encastren nueve veces `Mover a Lucho al [Oeste]`. En este caso, debemos indicarles que exploren el entorno y busquen una solución alternativa.

Posible propuesta incorrecta

También puede suceder que haya estudiantes que, en lugar de crear el procedimiento `Llegar al final del pasillo`, encastren las instrucciones directamente en el cuerpo principal del programa. De ser así, haremos notar que el enunciado pide explícitamente crear el procedimiento.

Posible propuesta que no crea el procedimiento pedido

CIERRE

Concluimos comentando que, al definir un procedimiento que utilice una repetición, puede suceder que antes haya que realizar otras acciones –como mover a Lucho al sur, en este caso–. Dado que la repetición de comandos es a su vez un comando, resulta perfectamente posible armar una secuencia que incluya una repetición.

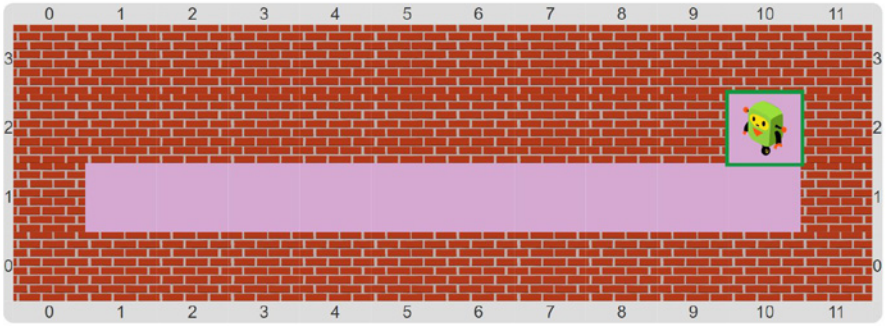
NOMBRE Y APELLIDO:

CURSO: FECHA:

EL FINAL DEL PASILLO

Los pasillos son espacios largos y estrechos que nos permiten ir de un lugar a otro. Hay algunos muy bellos y bien decorados, otros tienen techos bajos que nos producen una sensación de encierro y otros tienen techos altos, que nos hacen sentir muy pequeños. Sin embargo, los peores son los que no nos conducen a ningún lado. ¿En qué tipo de pasillo se metió Lucho?

1. Abrí el proyecto "El final del pasillo" y definí un procedimiento `Llegar al final del pasillo` que le permita a Lucho moverse hasta el otro extremo del pasillo.



2. Copiá acá abajo el procedimiento que armaste y observá dónde aparece la repetición.

UNA AYUDITA

En la biblioteca hay un procedimiento que te ayudará a resolver el desafío. Tené en cuenta que en esta actividad Lucho no puede moverse adonde hay una pared: ¡las paredes tienen explosivos que se activan por contacto!

Actividad 3
Siga la flecha

DE A DOS

OBJETIVOS

- Repetir secuencias de instrucciones.
- Usar más de una repetición en un procedimiento.

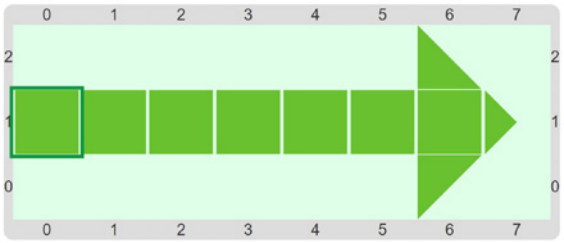
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

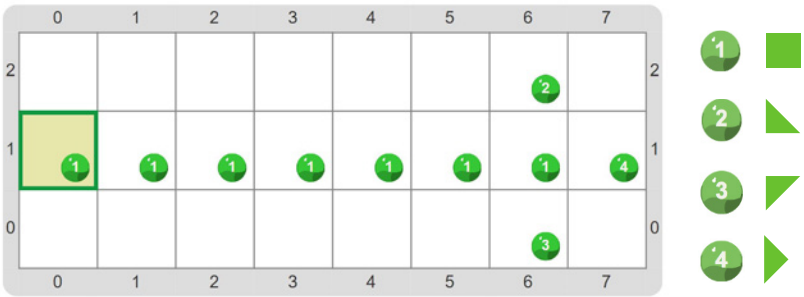
Esta actividad propone un desafío que incorpora dos características que lo diferencian de los anteriores en relación con el uso del comando `repetir [ ] veces`: en primer lugar, el comando se utilizará para repetir la ejecución de más de una instrucción; en segundo lugar, se lo incluirá más de una vez en un mismo procedimiento. Como la complejidad es mayor que la de las actividades anteriores, durante el desarrollo es recomendable ir monitoreando a los estudiantes y, en caso de que haga falta, guiarlos para que puedan llegar a una solución.

El objetivo de la actividad es transformar un tablero de 8 × 3 inicialmente vacío en uno que muestre una flecha verde. Al finalizar, el cabezal deberá quedar ubicado en la misma posición que al inicio, sobre la celda (0,1). La posibilidad de ver la flecha sobre el tablero se debe a que la actividad usa una vestimenta.



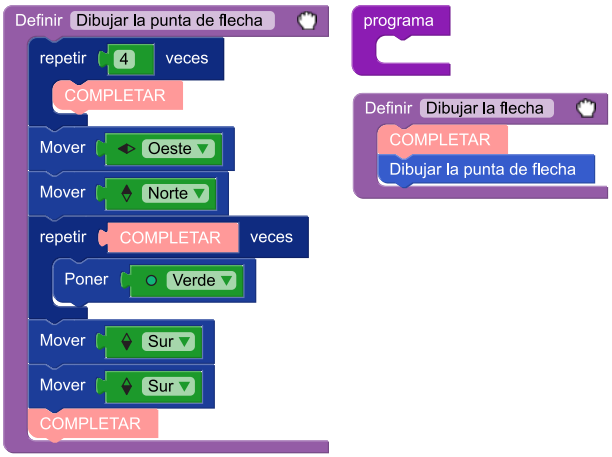
Tablero final

En este caso, no hay procedimientos en la biblioteca que nos ayuden a resolver el desafío –como podría ser `Dibujar un cuadrado verde`–. Por lo tanto, en principio, no contamos más que con los comandos básicos de Gobstones. En consecuencia, debemos tener presente la asociación entre bolitas e imágenes que define la vestimenta de la actividad: una bolita verde representa un cuadrado verde; dos, el triángulo superior de la punta de la flecha; tres, el triángulo inferior; y cuatro, el delantero. Conociendo esta información, sabemos cómo disponer bolitas sobre el tablero de forma tal de que, cuando la vestimenta esté activada, veamos la flecha.



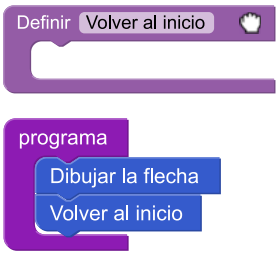
Tablero final sin vestimenta

Comenzamos la actividad repartiendo las fichas a los estudiantes y a continuación les pedimos que carguen el proyecto "Siga la flecha". Se encontrarán con un programa incompleto en el que aparecen el cuerpo principal del programa vacío y dos procedimientos incompletos: `Dibujar la flecha` y `Dibujar la punta de flecha`.



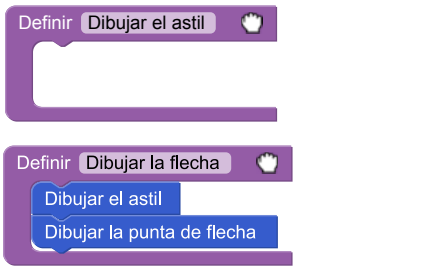
Espacio del programa al abrir el proyecto

Teniendo en cuenta el objetivo de la actividad, una estrategia posible es dividir el problema en dos partes: en primer lugar, dibujar la flecha –usando `Dibujar la flecha`–, y luego ocuparnos de posicionar el cabezal sobre la celda (0,1), para lo que creamos un nuevo procedimiento: `Volver al inicio`. De este modo, podemos completar el cuerpo principal del programa para que quede claramente expresada la estrategia elegida.



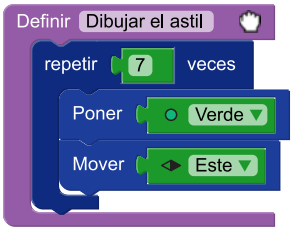
Definición de `Volver al inicio` y cuerpo del programa principal

En el procedimiento `Dibujar la flecha` falta completar qué hacer antes de invocar `Dibujar la punta de flecha`. Teniendo en cuenta que el cabezal comienza posicionado en el extremo izquierdo de donde debe ubicarse el eje de la flecha –también llamado astil–, es conveniente comenzar dibujándolo. Para ello, definimos un nuevo procedimiento: `Dibujar el astil`. De esta forma completamos `Dibujar la flecha`.



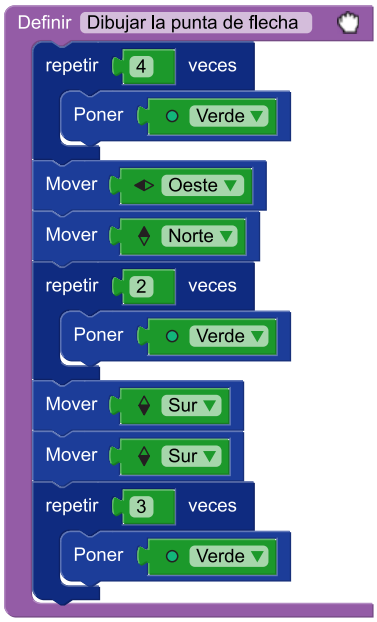
Definición y uso de `Dibujar el astil`

El astil de la flecha comprende las primeras 7 celdas de la fila del medio: (0, 1), (1,1)... (6,1). En términos de bolitas, lo que debemos hacer es colocar una verde en cada una de ellas; es decir, poner una en la celda (0,1) y mover el cabezal a (1,1), colocar otra en (1,1) y mover el cabezal a (2,1), y así hasta colocar una en (6,1) y dejar el cabezal posicionado sobre (7,1). En otras palabras, se trata de repetir siete veces las instrucciones `Poner [Verde]` y `Mover [Este]`.



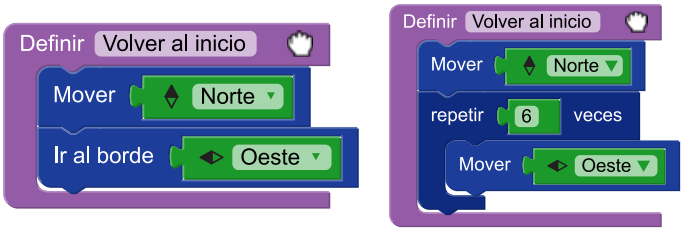
Procedimiento `Dibujar el astil`

Siguiendo con el desafío, en el procedimiento `Dibujar la punta de flecha` hay dos apariciones del comando `repetir [ ] veces`, ambas incompletas. En la primera faltan las instrucciones del cuerpo del bloque y en la segunda, la cantidad de veces que deben ponerse bolitas verdes. Debemos ocuparnos de poner la cantidad de bolitas adecuada en cada celda de la punta de la flecha. Estas son las de las posiciones (7,1), (6,2) y (6,0) para los triángulos delantero, superior e inferior, respectivamente. Si prestamos atención, notaremos que, una vez dibujado el astil, el cabezal queda posicionado sobre la celda (7,1), donde hay que colocar las 4 bolitas verdes que usa la vestimenta para representar esa porción de la punta. Por lo tanto, completamos el cuerpo de `repetir [4] veces` usando `Poner [Verde]`. Las dos instrucciones inmediatamente posteriores posicionan el cabezal sobre la celda del triángulo superior de la punta. Para dibujarlo hace falta ubicar allí 2 bolitas verdes, con lo que el "agujero" de la segunda aparición de `repetir [ ] veces` lo completamos con un `2`. Las dos instrucciones restantes que vienen predefinidas mueven el cabezal hasta la celda del triángulo inferior, en la que para completar la punta de flecha hay que colocar tres bolitas verdes. Nuevamente, nos valemos de `repetir [ ] veces`, en este caso usando `3` como argumento.



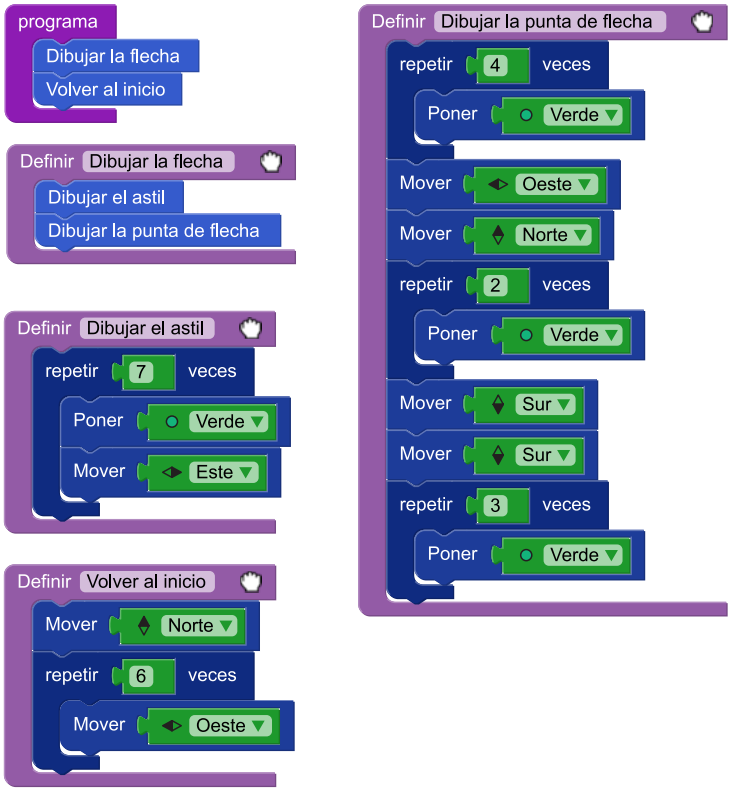
Procedimiento `Dibujar la punta de flecha`

Una vez dibujados el astil y la punta de la flecha, falta completar `Volver al inicio`. Considerando que, luego de dibujar la punta de la flecha, el cabezal queda posicionado sobre la celda (6, 0), esto puede conseguirse combinando `Mover [Norte]` tanto con el comando `repetir [ ] veces` como con `Ir al borde [ ]`. Ambas opciones son correctas.



Dos versiones de `Volver al inicio`

A continuación se encuentra una solución completa de la actividad:



Programa que resuelve el desafío

CIERRE

Luego de que todos los estudiantes hayan hecho el programa, los invitamos a reflexionar sobre el uso de los procedimientos y las repeticiones. Los procedimientos permiten crear programas que expresan claramente una estrategia de solución. Las repeticiones pueden incluir tantas instrucciones como haga falta, y, además, coexistir con otras repeticiones dentro de un mismo procedimiento.

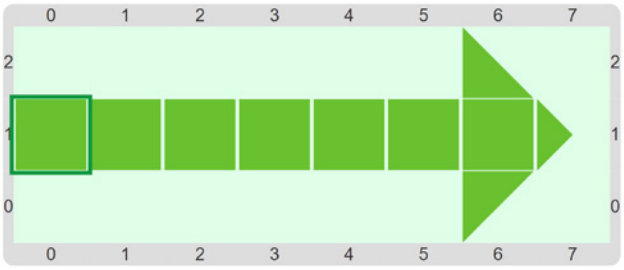
NOMBRE Y APELLIDO:

CURSO: FECHA:

SIGA LA FLECHA

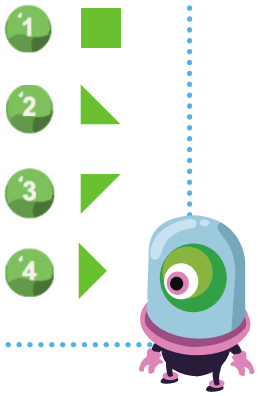
¿Sabías que las flechas existen desde la prehistoria? Se han encontrado puntas de flecha que datan de principios de la Edad de Piedra. Los antiguos egipcios, hace 5000 años, eran excelentes arqueros. Mucho después, con el desarrollo de los lenguajes simbólicos, las flechas se empezaron a usar para indicar direcciones. En esta actividad, vas a tener que poner en práctica tus habilidades para dibujar una flecha sobre el tablero de Gobstones.

1. Abrí el proyecto "Siga la flecha". Te vas a encontrar con un programa que tenés que completar. El objetivo es que, al final de una ejecución del programa, quede dibujada una flecha verde sobre el tablero. ¡Tené en cuenta que al final el cabezal tiene que quedar posicionado sobre la misma celda del inicio, que corresponde a la posición (0,1)!



LA VESTIMENTA

Esta vez no contás con procedimientos en la biblioteca que te ayuden a completar el desafío. Por suerte, el proyecto viene con una vestimenta que asocia bolitas verdes con imágenes.



Actividad 4
Candela, ¡me quemo!

DE A DOS

OBJETIVO

- Usar repeticiones que incluyan casos de borde.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

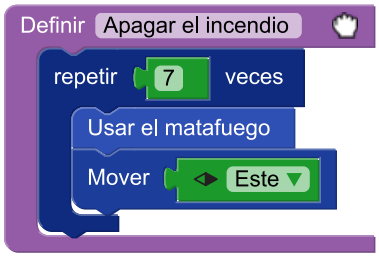
En esta actividad se propone un desafío para presentarles a los estudiantes lo que en programación se conoce como **caso de borde**. Se trata de realizar una misma acción sobre todas las celdas de un tablero, para lo que nos valdremos del uso del comando `repetir [ ] veces`, excepto para la última, que, como se verá, debe tratarse por separado.

Comenzamos pidiéndoles a los estudiantes que abran el proyecto "Candela, ¡me quemo!". Se encontrarán con un tablero de 7 × 1 en el que en todas las celdas hay una llamarada. El objetivo es crear un procedimiento `Apagar el incendio` para extinguir el fuego.

Tablero inicial de "Candela, ¡me quemo!"

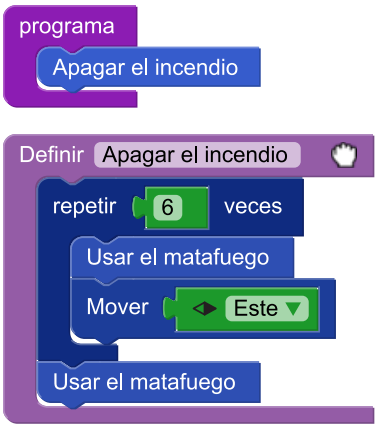
Les sugerimos a los estudiantes que exploren el entorno con el objetivo de descubrir que en *Biblioteca* > *Procedimientos* hay un procedimiento disponible llamado `Usar el matafuego`, que será de utilidad para resolver el desafío.

Como hay que apagar el fuego de siete celdas, es probable que algún estudiante proponga repetir siete veces `Usar el matafuego` y moverse a la siguiente celda, al este. Esta propuesta tiene un problema: luego de apagar el último foco del incendio –en la última celda–, el cabezal tratará de moverse hacia el este, pero al hacerlo se caerá del tablero y se producirá una falla.



Definición incorrecta

El problema mencionado puede remediarse fácilmente. En lugar de repetir siete veces ambas instrucciones, lo haremos seis. Al finalizar la repetición, el cabezal se encontrará sobre la última celda, que corresponde a la posición (6,0). En ese momento, solo hay que apagar la última llama sin mover el cabezal; se invocará entonces una vez `Usar el matafuego`.



Programa que resuelve el desafío

CIERRE

Para finalizar, reflexionamos acerca de la necesidad de analizar con cuidado la cantidad de veces que hay que repetir una serie de acciones. En esta actividad se observa que, aunque hay que usar siete veces el matafuego, solo hay que desplazar seis veces el cabezal. Entonces, la repetición solo tiene que hacerse seis veces, y el último uso del matafuego se hace por separado. Este último caso separado se conoce en programación como un caso de borde, porque siempre se da antes de comenzar o luego de terminar una repetición.

NOMBRE Y APELLIDO:

CURSO: FECHA:

CANDELA, ¡ME QUEMO!

¡Se prende fuego el tablero! Definí un procedimiento `Apagar el incendio` que te ayude a extinguir todas las llamas. Explorá el entorno para ver si encontrás algo que pueda ayudarte a evitar esta catástrofe.

¿Cuántas veces hay que usar el matafuego? ¿Cuántas repeticiones pusiste en `repetir [] veces`? ¿Por qué?

¡ATENCIÓN!

Hay siete focos de incendio, pero... ¿hay que moverse tantas veces?

Actividad 5
El entrenamiento del robot goleador

DE A DOS

OBJETIVOS

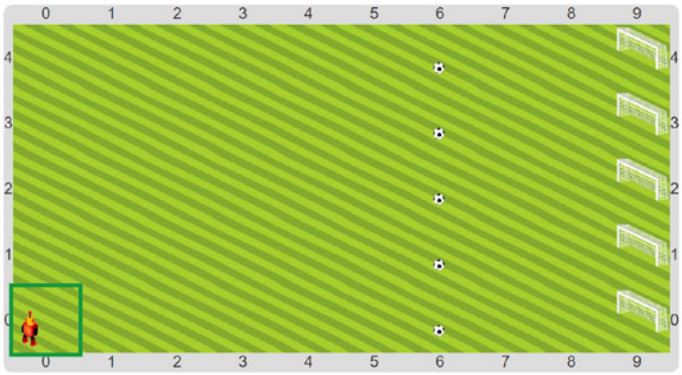
- Construir un programa que anide repeticiones implícitamente.
- Crear un programa con una estrategia de arriba hacia abajo.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

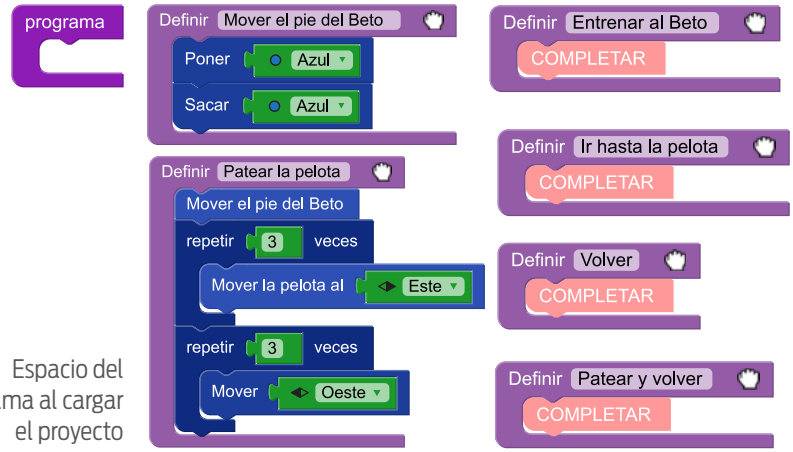
La novedad que presenta esta actividad es que en el programa se **anidarán** dos repeticiones. Es decir que, entre las acciones que se realizan dentro de un `repetir [] veces`, habrá otro `repetir [] veces`. De todas formas, cada uno de esos `repetir [] veces` quedará encapsulado dentro un procedimiento distinto, pues dentro del programa pertenecen a unidades de sentido diferentes.



Tablero inicial de "El entrenamiento del robot goleador"

Repartimos la ficha a los estudiantes y les indicamos que abran el proyecto "El entrenamiento del robot goleador". Se encontrarán con un tablero de 10 × 5 en el que el robot el Beto se encuentra parado en la posición (0,0). En cada celda de la columna 6 hay una pelota y en todos los de la 9, un arco. El objetivo es que el Beto meta 5 goles, uno con cada pelota.

En el panel izquierdo encontrarán un programa incompleto que contiene 4 procedimientos vacíos: `Entrenar al Beto`, `Patear y volver`, `Ir hasta la pelota` y `Volver`. Para completar el desafío, los estudiantes deben completar la definición de cada uno, además del cuerpo principal del programa. Los procedimiento `Patear la pelota` y `Mover el pie del Beto` ya están resueltos.



Espacio del programa al cargar el proyecto

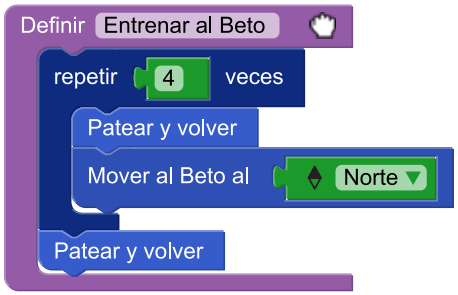
Antes de comenzar la resolución, hacemos notar que en *Biblioteca* > *Procedimientos* contamos con el procedimiento `Mover al Beto al [ ]`, que es de utilidad para completar el desafío.

Procedimiento disponible en la biblioteca

Les indicamos a los estudiantes que, como primer paso, lean los nombres de los procedimientos y, en función de ellos, piensen una estrategia para resolver el problema. Lo que se busca es que construyan el programa **de arriba hacia abajo**, es decir, que se ocupen primero del enfoque general de la resolución y luego, a medida que avancen, de los detalles de cada parte del programa. Así, podemos preguntarles: "Teniendo en cuenta los nombres de los procedimientos, ¿qué les parece que debería haber dentro del bloque `programa`?". A partir de este planteo, guiamos la discusión de forma tal de que los estudiantes lleguen a la conclusión de que el programa se trata simplemente de entrenar al Beto y que, por lo tanto, podría solo contener una invocación `Entrenar al Beto`, independientemente de cómo se complete la definición de este último. Esto nos permite empezar por `programa` y luego resolver los procedimientos restantes.[1]

Cuerpo principal del programa

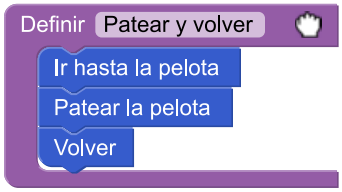
Para continuar preguntamos: "Ahora bien, ¿en qué consiste el entrenamiento del Beto?". Tal como se indica en la consigna, el Beto tiene que meter un gol pateando cada una de las cinco pelotas del tablero. Por lo tanto, una opción es usar el comando `repetir [ ] veces` para que patee una pelota, vuelva a la primera columna y, a continuación, se posicione sobre la fila inmediatamente superior. Notemos que aquí hay un caso de borde: la última invocación a `Patear y volver` debe hacerse fuera del ciclo para evitar que el Beto se caiga del tablero.



Procedimiento `Entrenar al Beto`

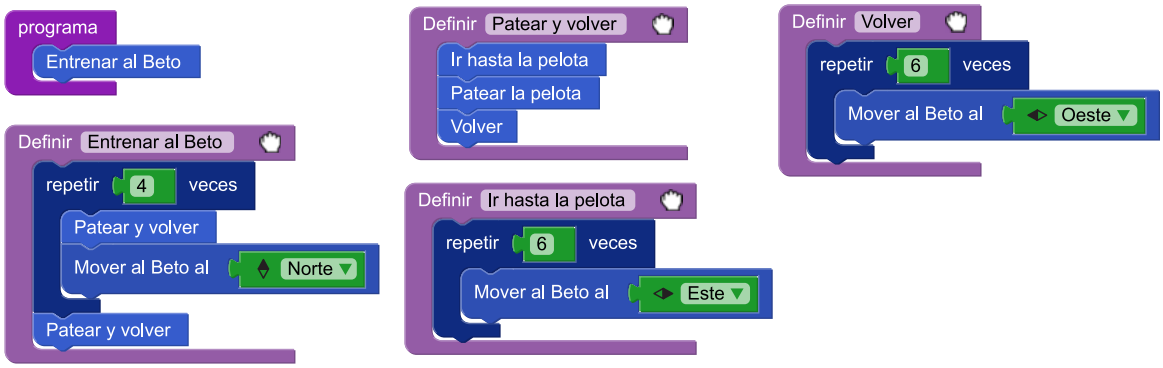
[1] A esto nos referimos con un enfoque de arriba hacia abajo.

Siguiendo el mismo enfoque, en esta instancia los estudiantes tienen que completar `Patear y volver`. Prestando atención a los nombres de los procedimientos, no es complicado notar que el procedimiento puede resolverse invocando a `Ir hasta la pelota`, `Patear la pelota` y `Volver`, en ese orden.



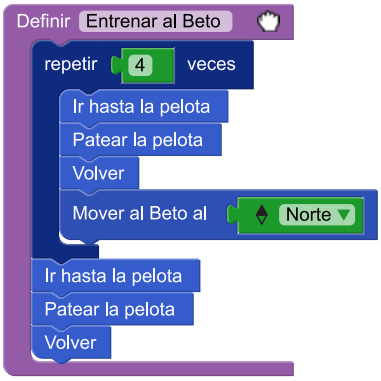
Procedimiento `Patear y volver`

Solo falta completar `Ir hasta la pelota` y `Volver`. El primero tiene que desplazar al Beto desde la columna 0 hasta la 6 y el segundo, de la 6 a la 0. Ambos pueden resolverse usando el comando `repetir [ ] veces`. Con esto, obtenemos una solución completa de nuestro problema, enfocándonos primero en la estrategia y, más adelante, en los detalles de cada una de sus partes. A continuación se muestra la solución completa de la actividad.



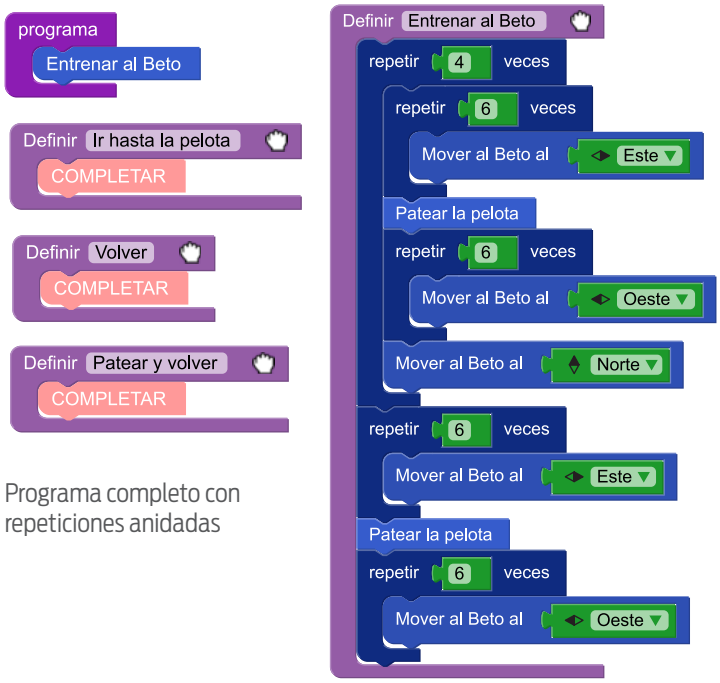
Programa que resuelve el desafío

Al analizar la resolución del problema, puede ser interesante reflexionar sobre qué hubiese sucedido si hubiéramos localizado el comportamiento de todos los procedimientos que definimos únicamente en `Entrenar al Beto`. Para observar cómo hubiera quedado el programa, comenzamos reemplazando en `Entrenar al Beto` las invocaciones `Patear y volver` por su definición: `Ir hasta la pelota`, `Patear la pelota` y `Volver`.



Incorporación de la definición de `Patear y volver` dentro de `Entrenar al Beto`

Luego, reemplazamos las invocaciones `Ir hasta la pelota` y `Volver`. Así quedan los procedimientos una vez realizados todos los reemplazos:



Programa completo con repeticiones anidadas

Si bien el comportamiento de ambos programas es idéntico, la última propuesta no es adecuada. Por un lado, a diferencia de lo que sucede con la primera solución, al leer la segunda es difícil comprender qué hace el procedimiento `Entrenar al Beto`. Por otro lado, en este caso hay ocurrencias de bloques `repetir [ ] veces` dentro de otro bloque `repetir [ ] veces`. En general, cada aparición de `repetir [ ] veces` en un programa constituye una unidad de sentido. Por lo tanto, es conveniente que cada una quede encapsulada dentro de un procedimiento distinto. En esta actividad, la función de un `repetir [ ] veces` es que el Beto se desplace desde el inicio de una fila hasta la pelota, hay otro para que vuelva al inicio de la fila y el último –que engloba a los dos anteriores– es para que se mueva entre las distintas filas. Si algún estudiante opta por una solución así, le sugerimos que piense otra alternativa en la que no ocurran repeticiones explícitamente anidadas.

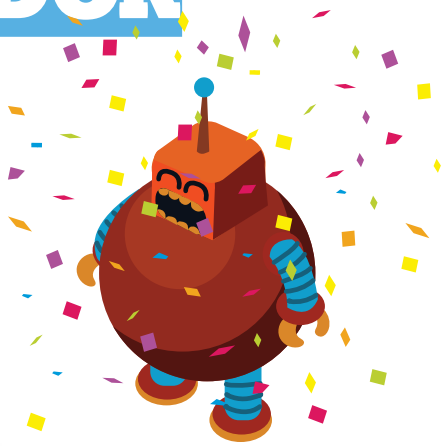
CIERRE

Para terminar, conversamos con los estudiantes acerca de cómo resolvimos esta actividad, de arriba hacia abajo. En primer lugar, nos focalizamos en qué debía hacer el programa, y pasamos por alto los detalles de cómo resolver cada parte. A medida que fuimos construyendo la solución, nos ocupamos de las porciones más pequeñas que fueron apareciendo, hasta alcanzar una versión completa del programa que resuelve el desafío. En resumen, fuimos estudiando estrategias que resolvimos usando procedimientos. Por último, les comentamos que, para resolver muchos problemas de programación, hace falta anidar repeticiones. Es necesario hacer hincapié en que cada repetición cumple una función diferente y que, por lo tanto, cada una debe encapsularse en un procedimiento distinto.

NOMBRE Y APELLIDO:

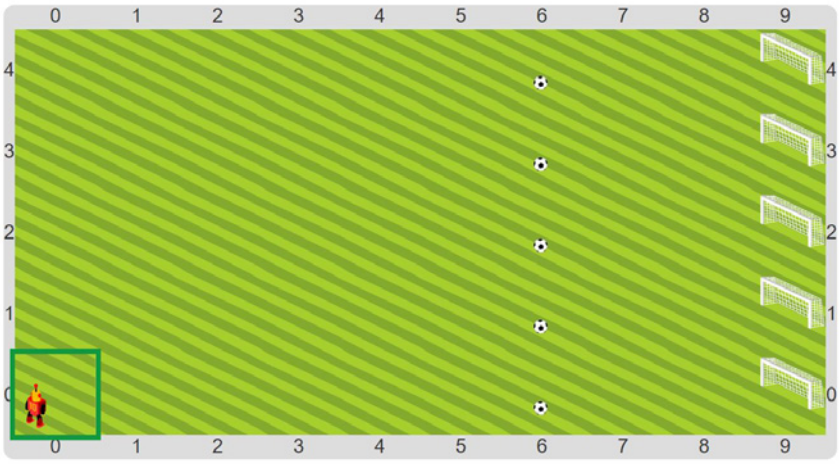
CURSO: FECHA:

EL ENTRENAMIENTO DEL ROBOT GOLEADOR



En una actividad anterior le enseñamos al robot el Beto a patear la pelota. Ahora nos vamos a ocupar de entrenarlo. Para eso, el entrenador dispuso cinco pelotas en el campo de juego. El Beto debe patear cada una y meter goles.

1. Abrí el proyecto "El entrenamiento del robot goleador" y completá el programa para que el Beto pueda cumplir su objetivo. El tablero inicial es el siguiente:



PARA TENER EN CUENTA

- Pensá una estrategia de solución y expresala con procedimientos.
- Revisá la biblioteca y fijate si encontrás algo que pueda ayudarte.

¿En qué orden completaste los procedimientos? ¿Por qué?

El mecánico de naves espaciales

DE A DOS

OBJETIVOS

- Definir una estrategia para resolver un problema.
- Definir procedimientos y nombrarlos adecuadamente.
- Utilizar repeticiones simples.

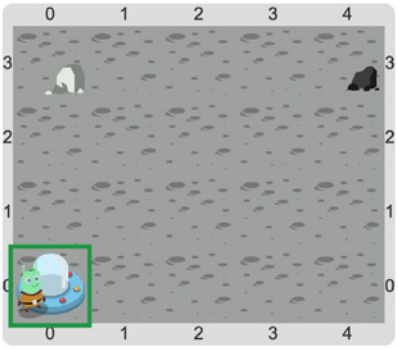
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

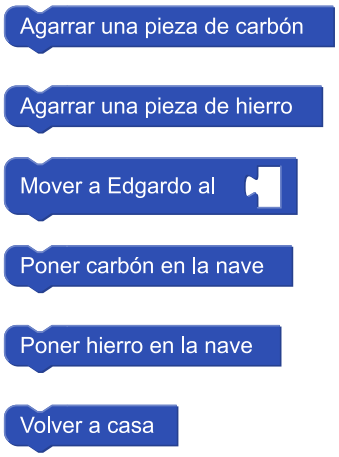
Esta actividad integra los temas abordados en el capítulo: estrategias, procedimientos y repeticiones. Una vez presentado el problema, el primer paso consiste en concebir una estrategia para resolverlo. Para que una propuesta de solución se considere correcta, la estrategia que piensen los estudiantes debe quedar claramente expresada mediante el uso de procedimientos con nombres adecuados. Aquí analizaremos dos estrategias posibles, aunque no son las únicas.

Para comenzar, les indicamos a los estudiantes que abran el proyecto "El mecánico de naves espaciales". Se encontrarán con un tablero de 5 × 4. En la posición (0,0) está el marciano Edgardo y su nave espacial, que está descompuesta. Para repararla, Edgardo tiene que incorporar a la nave tres piezas de hierro y tres de carbón, que puede recoger de las posiciones (0,3) y (4,3), respectivamente. Por tratarse de piezas muy pesadas, Edgardo debe llevar una por vez. Una vez que el vehículo espacial esté en condiciones, Edgardo tiene que volver a su casa en Marte.



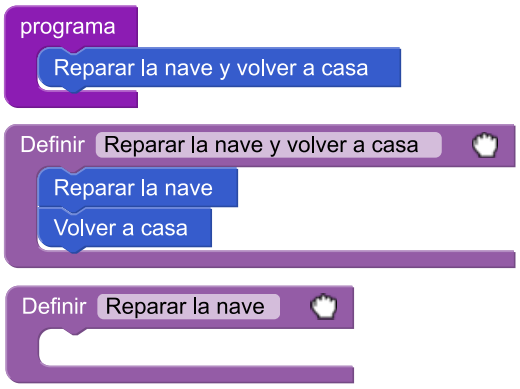
Tablero inicial de "El mecánico de naves espaciales"

En la *Biblioteca* > *Procedimientos* hay varios procedimientos que son de utilidad para resolver el desafío: `Agarrar una pieza de carbón`, `Agarrar una pieza de hierro`, `Mover a Edgardo al [ ]`, `Poner carbón en la nave`, `Poner hierro en la nave` y `Volver a casa`.



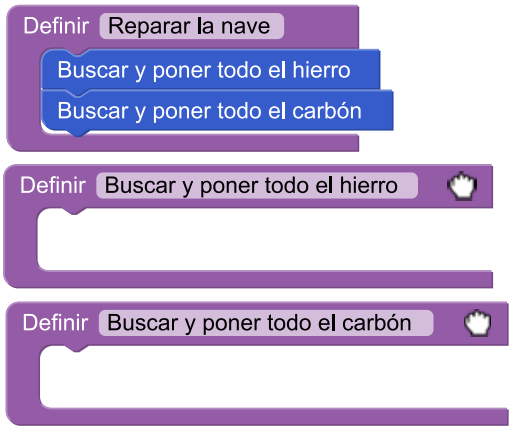
Procedimientos disponibles en la biblioteca

Cualquiera sea la estrategia que utilicemos, lo que Edgardo debe conseguir es reparar la nave y volver a su casa. Es una buena idea, entonces, crear un procedimiento `Reparar la nave y volver a casa`, y que el cuerpo principal del programa consista en una invocación a este procedimiento. Para definirlo, podemos crear un nuevo procedimiento `Reparar la nave` y usar `Volver a casa`, disponible en la biblioteca. Luego, dependiendo de la estrategia elegida, `Reparar la nave` se resolverá de una u otra forma.



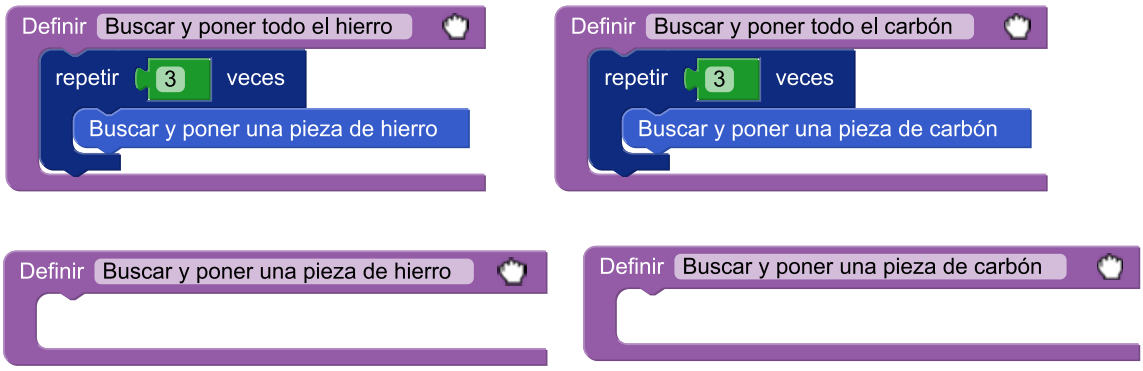
Cuerpo principal del programa y definición de `Reparar la nave y volver a casa`

Una estrategia posible consiste en buscar primero tres piezas de hierro y ponerlas en la nave y, a continuación, hacer lo mismo con las de carbón. Este enfoque diferencia dos grandes partes; cada una debe ser atendida por un procedimiento. Podrían llamarse, por ejemplo, `Buscar y poner todo el hierro` y `Buscar y poner todo el carbón`, y usarse para completar `Reparar la nave`.



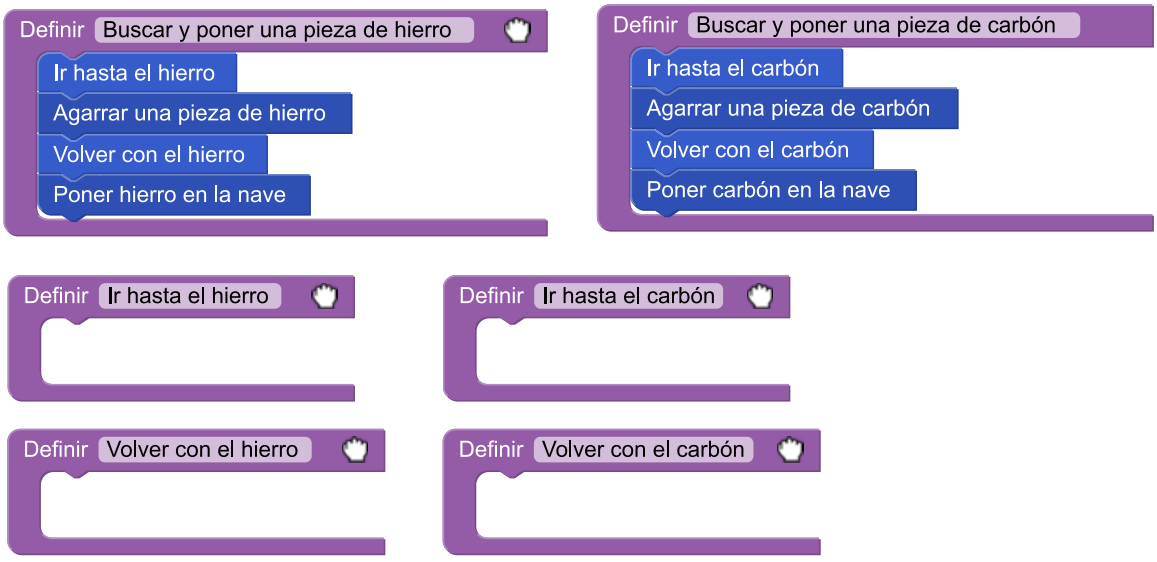
Definición de procedimientos para completar `Reparar la nave`

Como Edgardo puede transportar una pieza por vez, para buscar y poner todo el hierro tiene que ir y volver tres veces. Esto puede resolverse con un nuevo procedimiento `Buscar y poner una pieza de hierro` y el comando `repetir [ ] veces`. La misma idea sirve para incorporar carbón a la nave.



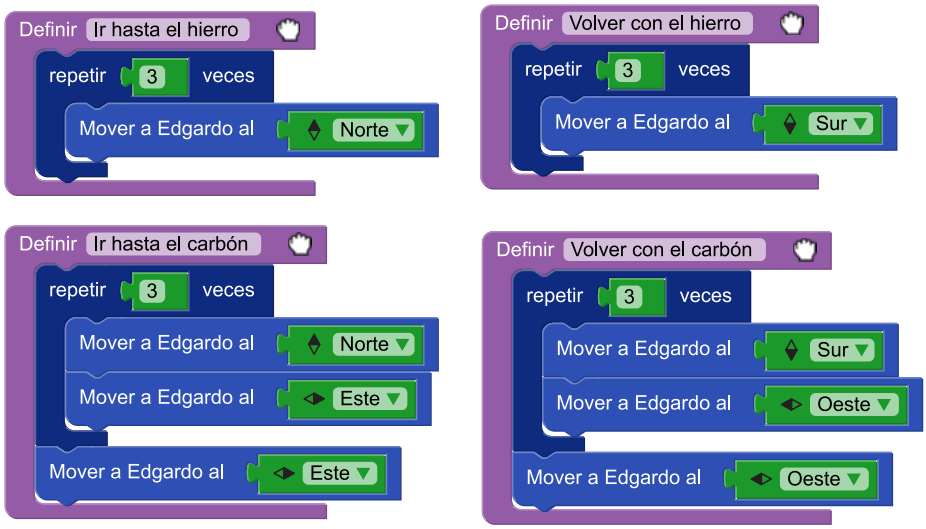
Definición de procedimientos para transportar minerales

Para buscar y poner en la nave una pieza de cualquiera de los dos materiales, Edgardo debe *(i)* ir hasta donde están las piezas del mineral, *(ii)* agarrar una, *(iii)* volver hasta la nave y *(iv)* depositarla. Para recoger y poner las piezas usaremos los procedimientos `Agarrar una pieza de hierro`, `Poner hierro en la nave`, `Agarrar una pieza de carbón` y `Poner carbón en la nave`, disponibles en la biblioteca. Para los desplazamientos, lo más conveniente es crear nuevos procedimientos. Una posibilidad es denominarlos `Ir hasta el hierro`, `Volver con el hierro`, `Ir hasta el carbón` y `Volver con el carbón`.



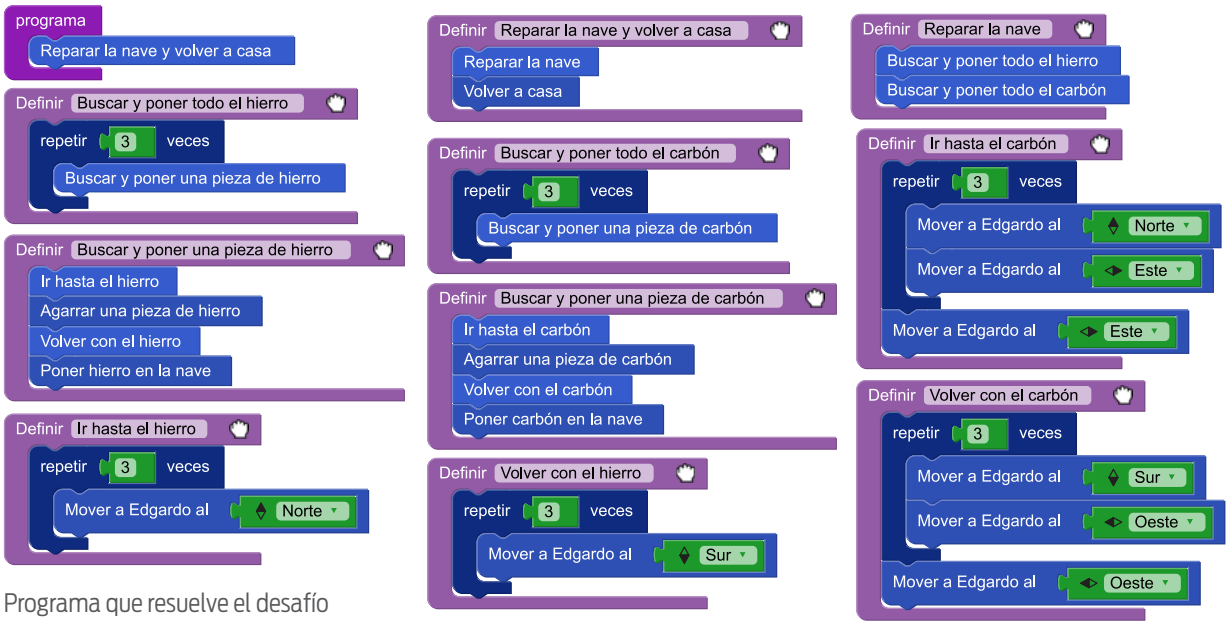
Procedimientos para buscar y transportar minerales

Finalmente, los procedimientos para que Edgardo vaya de un lugar a otro se resuelven combinando el procedimiento de la biblioteca `Mover a Edgardo al [ ]` y el comando `repetir [ ] veces`.



Procedimientos para desplazar a Edgardo.

A continuación, se muestra el programa completo.



Programa que resuelve el desafío

Otra estrategia posible para reparar la nave consiste en ir alternando la búsqueda de piezas de hierro y carbón. Con este enfoque, ya no son necesarios los procedimientos `Buscar y poner todo el hierro` y `Buscar y poner todo el carbón`; bastará con repetir tres veces `Buscar y poner una pieza de hierro` y `Buscar y poner una pieza de carbón`. Al eliminar los procedimientos `Buscar y poner todo el hierro` y `Buscar y poner todo el carbón`, también se modifica `Reparar la nave`. A la derecha, se muestra un programa que resuelve el desafío de este modo.

Programa que resuelve el desafío con una estrategia alternativa

CIERRE

Al terminar la actividad, señalaremos que la estrategia de solución debe quedar claramente expresada en el programa. Para lograrlo, es imprescindible elegir nombres adecuados para los procedimientos. Las personas normalmente tratan de entender un programa leyéndolo, no ejecutándolo.

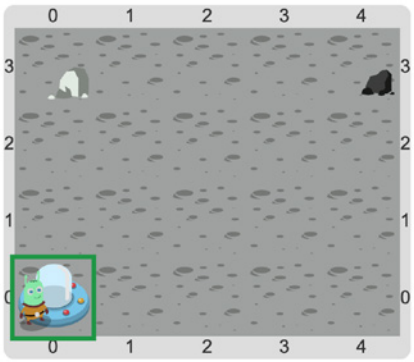
NOMBRE Y APELLIDO:

CURSO: FECHA:

EL MECÁNICO DE NAVES ESPACIALES

¡Menos mal que el marciano Edgardo es tuerca!
Si no, nunca hubiese podido regresar a Marte.

1. Abrí el proyecto "El mecánico de naves espaciales" y pensá una estrategia para que Edgardo repare su nave y pueda volver a su planeta. El tablero inicial muestra al marciano en su nave y los depósitos de hierro y carbón.



Para reparar la nave, Edgardo tiene que conseguir tres unidades de hierro y tres de carbón, que se encuentran en las esquinas superior izquierda y superior derecha respectivamente. Las piezas son muy pesadas, por lo que el extraterrestre no puede llevar más de una unidad de mineral por vez. Entonces, hay que planificar una estrategia para que Edgardo haga varios viajes para completar su tarea.

Escribí acá abajo la estrategia que pensaste:

RECORDATORIO

Cuando programes los procedimientos para cada una de las tareas, ¡no te olvides de usar el comando de repetición para las cosas que se repiten! Además, tené cuidado con los casos de borde.

2. Ahora armá un programa que ponga en práctica tu estrategia. Escribí un procedimiento por cada tarea que hayas identificado y ponele un nombre adecuado. Por ejemplo, `Ir hasta el carbón` y `Volver con el carbón` podrían ser algunos de los procedimientos. ¿Podés organizar bien el programa para que todas las tareas tengan su procedimiento? Al leer el programa final, ¿se entiende la estrategia de solución que pensaste?

04

DATOS, ALTERNATIVA CONDICIONAL Y FUNCIONES

En los capítulos precedentes, el foco estuvo puesto en las acciones que realiza un programa, tanto mediante comandos básicos como a través de procedimientos. Este capítulo aborda en profundidad el uso de **datos**. Si entendemos las acciones como los verbos de un lenguaje de programación, los datos corresponden a los sustantivos.

Los valores que adquieren los datos determinan el comportamiento de los programas. La **alternativa condicional** es una herramienta de los lenguajes de programación que permite que un programa se comporte de uno u otro modo de acuerdo con ciertas condiciones de los datos. Permite, por lo tanto, construir programas versátiles que funcionen en distintos escenarios.

El origen de los datos puede ser diverso. En Gobstones se pueden usar tanto valores literales (por ejemplo, *Rojo*, *Norte* o 5) como operadores y sensores. Además, el lenguaje da la posibilidad de definir **funciones**, para que el programador calcule datos de una forma que le resulte conveniente.

Secuencia Didáctica 1

DATOS

En esta secuencia didáctica se presentan algunos aspectos esenciales relativos al uso de datos en los programas. En primer lugar, se explica cómo puede dividirse el universo de los posibles valores de los datos en conjuntos que agrupan a los que comparten ciertas características. Por ejemplo, *rojo*, *verde*, *negro* y *azul* son colores; *norte*, *sur*, *este* y *oeste* son direcciones; y 1, 2, 3, 4 y 5 son números. En programación, cada uno de estos grupos forma lo que se conoce como un **tipo de datos**.

Por otro lado, se presenta el uso de **sensores** como fuentes de datos que son de interés en el contexto de un problema. En particular, se trabaja con un sensor que permite conocer el número de bolitas de un color que hay en la celda del tablero de Gobstones que se encuentra bajo el cabezal.

Asimismo, se presentan los **operadores**, que se utilizan para llevar a cabo operaciones que, a partir de uno o más datos de un tipo, obtienen un nuevo valor. Específicamente, en las actividades de esta secuencia se usan operadores para realizar cálculos sobre números y también una operación definida para direcciones.

OBJETIVOS

- Presentar la noción de tipo de datos.
- Introducir el uso de sensores.
- Presentar el uso de operadores.

Actividad 1
Tipos de datos

TODA LA CLASE

OBJETIVO

- Presentar la noción de tipo de datos.

MATERIALES

- Tapitas de botellas
- Hojas de árboles
- Monedas
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad se presentará la noción de tipo de datos. Con ese propósito, se comienza con un juego de clasificación de objetos.[1] Luego, se traslada esta idea al contexto del lenguaje de programación Gobstones y se analiza qué sucede cuando elementos de un tipo se usan de forma indebida.

Clasificación de objetos

Comenzamos la actividad reuniendo a los estudiantes alrededor de una mesa sobre la cual disponemos tapitas de botellas, hojas de árboles y monedas, todo mezclado.[2]

Pedimos a un estudiante que organice los elementos en tres grupos. Se espera que ubique por un lado todas las tapitas; por otro, las hojas; y por otro, las monedas. En caso de que no ocurra así, les pedimos a los compañeros que den sus impresiones sobre la clasificación propuesta y que indiquen cómo la hubieran hecho ellos, hasta llegar a un planteo que resuelva la actividad del modo esperado.

Se pueden formular preguntas tales como: "¿Qué tienen en común los elementos de cada grupo? ¿Qué cosas se pueden hacer con todos los elementos de un grupo que no se pueden hacer con los otros? ¿Tiene sentido hacer las cosas que se mencionaron para un grupo con elementos de los otros grupos?".

Finalizamos esta primera parte de la actividad presentando la noción de **tipo de elementos**, como un conjunto de objetos que comparten determinadas características.

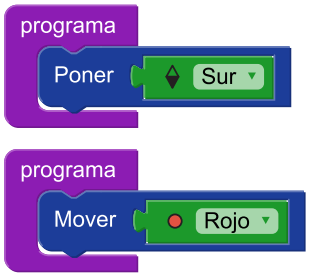
Tipos de datos en Gobstones

Les repartimos la ficha de la actividad a los estudiantes y les pedimos que resuelvan la primera consigna. En primer lugar, se pide que describan qué creen que pasaría al ejecutar un programa en Gobstones cuya única instrucción sea `Poner [Sur]` y otro compuesto solo por `Mover [Rojo]`. Luego, les proponemos que abran el entorno de Gobstones, que creen un proyecto nuevo, que

[1] La propuesta es trabajar con tapitas de botellas, hojas de árboles y monedas. Sin embargo, estos objetos pueden reemplazarse por otros, teniendo cuidado de que cada tipo de objeto se diferencie claramente del resto.

[2] Se sugiere no usar menos de 15 objetos de cada tipo.

construyan ambos programas y que observen qué sucede al ejecutarlos. Preguntamos: "¿Por qué falla la ejecución de los programas? ¿Qué tipo de dato espera el comando `Poner [ ]`? ¿El valor `Sur` es de ese tipo? ¿Y qué sucede con el otro programa?". Los dos programas invocan un comando con un valor de un tipo distinto al esperado. El comando `Poner [ ]`, que espera un valor de tipo *color*, es invocado con una *dirección*; y `Mover [ ]`, que espera una *dirección*, es invocado con un *color*.

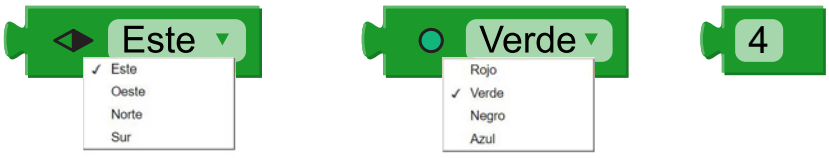


Argumentos de tipo incorrecto

Crear un proyecto nuevo

1. Hacé clic en las tres líneas del menú superior
2. Presioná el ícono de creación de proyectos

A continuación, presentamos la noción de **tipo de datos** como un conjunto de valores que comparten características comunes. "¿Qué tipos de datos existen en Gobstones?". *Colores*, *direcciones* y *números*.[1] En caso de que no mencionen el último, podemos preguntar: "¿Qué recibe el comando `repetir [ ] veces`?". Indagamos entonces sobre los posibles valores de cada tipo. "¿Cuáles son los valores de cada tipo?". Los de los *colores* son `Azul`, `Negro`, `Rojo` y `Verde`; los de las *direcciones* `Norte`, `Este`, `Sur` y `Oeste`; y los de los *números* `0`, `1`, `2`, `3`, etc.

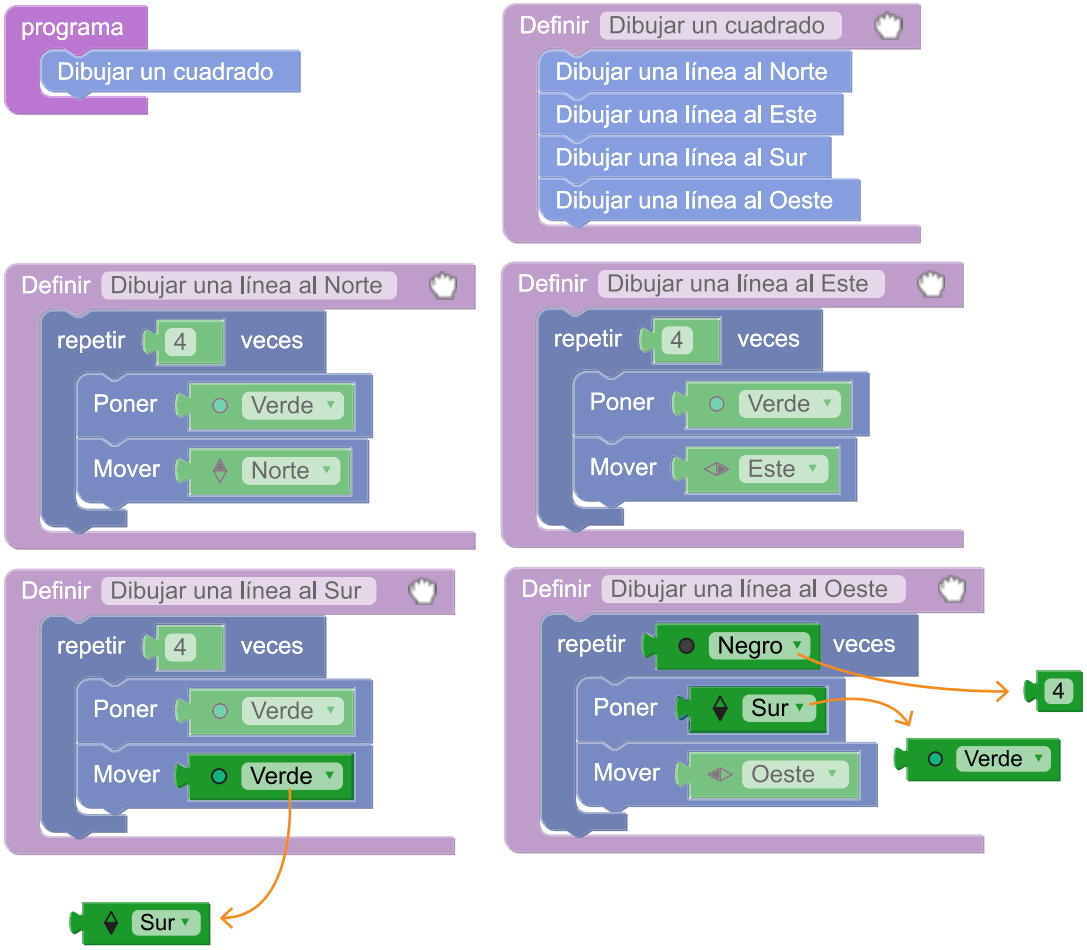


Colores, direcciones y números

Por último, les pedimos que resuelvan la segunda consigna de la actividad, para lo cual tienen que abrir el proyecto "Por qué falla" en Gobstones. Al hacerlo, se van a encontrar con un programa cuyo propósito es dibujar un cuadrado de 4 × 4 usando bolitas verdes. Sin embargo, el programa contiene errores: hay ciertos comandos que se invocan con valores de un tipo distinto al esperado. En el proce-

[1] En Gobstones también existe el tipo booleano, que aún no ha sido presentado.

dimiento `Dibujar una línea al sur` aparece `Mover [Verde]`; y en `Dibujar una línea al Oeste`, `repetir [Negro] veces` y `Poner [Sur]`. En primer lugar, los estudiantes deben buscar los errores sin ejecutar el programa. Luego, tienen que corregirlo y ejecutarlo hasta que funcione correctamente.



Programa con errores de tipo

CIERRE

Para concluir, reiteramos que los tipos de datos son conjuntos de valores que comparten características comunes. Además, invocar un comando con un valor de un tipo distinto al esperado es un error. Por último, mencionamos que muchos lenguajes de programación usan tipos de datos.

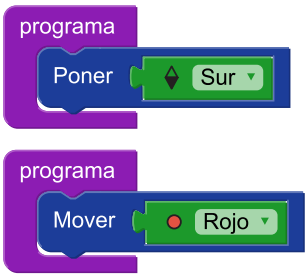
NOMBRE Y APELLIDO:

CURSO: FECHA:

TIPOS DE DATOS

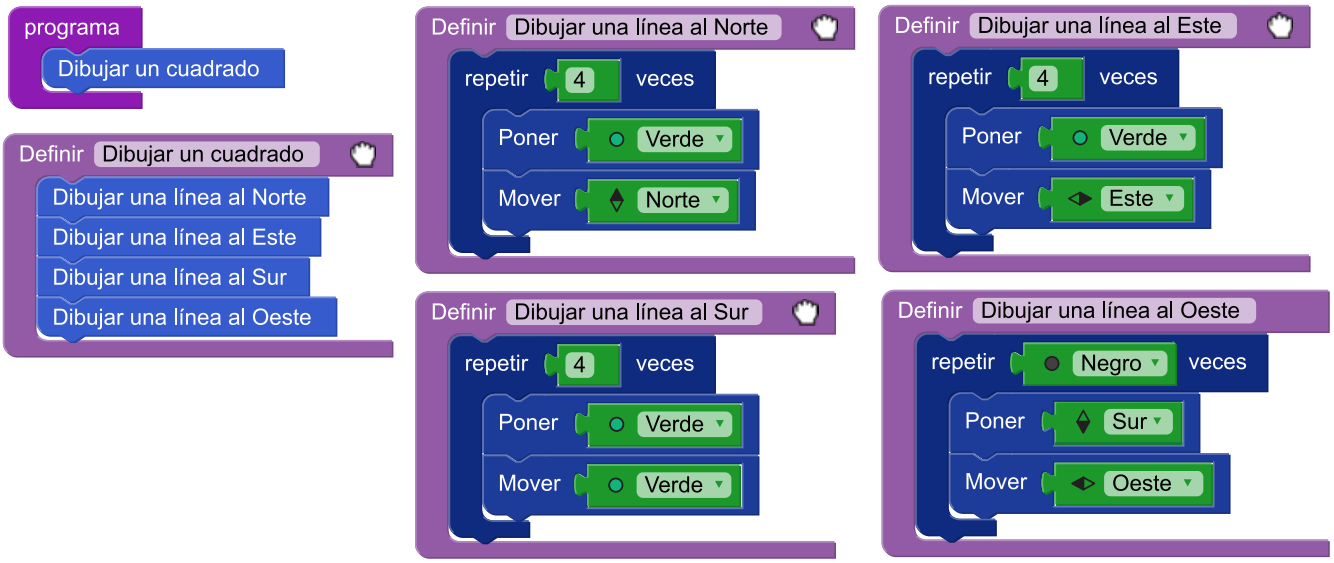
¿Los colores y las direcciones son lo mismo?
¿Por qué a veces usamos unos y, otras veces, otros?

1. ¿Qué ocurre si le damos la instrucción `Poner[Sur]` al cabezal de Gobstones? ¿Y si le damos la instrucción `Mover [Rojo]`? Justificá tu respuesta.



Ahora, abrí el entorno de Gobstones, creá un nuevo proyecto, armá ambos programas y fijate qué sucede. ¿Es lo que vos esperabas?

2. Este es un programa para dibujar un cuadrado de 4 × 4 con bolitas verdes. Sin embargo, no funciona bien. Miralo con atención y marcá todos los errores que encuentres.



Ahora, abrí el proyecto "Por qué falla", corregí todos los errores que identificaste y ejecutalo. ¿Te faltó encontrar alguno? Realizá todos los cambios que hagan falta para que el programa dibuje el cuadrado.

TIPOS DE DATOS

Los tipos de datos son conjuntos de valores que tienen características comunes. En Gobstones tenemos los siguientes: *colores*, *direcciones*, *números* y *booleanos*. Ya veremos de qué se tratan estos últimos.

Actividad 2
El cabezal puede contar

DE A DOS

OBJETIVOS

- Presentar el uso de sensores en Gobstones.
- Presentar el uso de operadores aritméticos en Gobstones.

MATERIALES

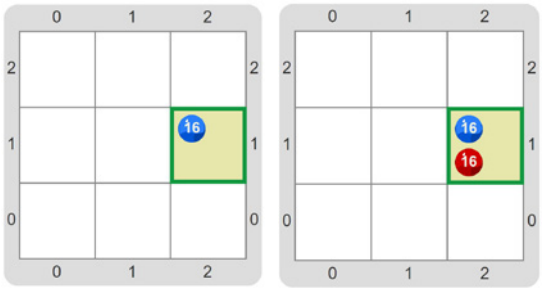
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad se usarán **sensores** y **operadores** de Gobstones por primera vez. En general, los sensores son dispositivos que permiten obtener información del ambiente. Por ejemplo, existen sensores de temperatura, de proximidad y de intensidad de luz. En Gobstones, hay sensores para conseguir información sobre condiciones del tablero en la proximidad de la posición del cabezal. Estos sensores nos permiten saber si hay bolitas de un determinado color en la celda sobre la cual se encuentra el cabezal, cuántas de ellas hay, y si es posible que el cabezal se mueva en una cierta dirección. En esta actividad empezaremos usando únicamente el que reporta la cantidad de bolitas de un color.

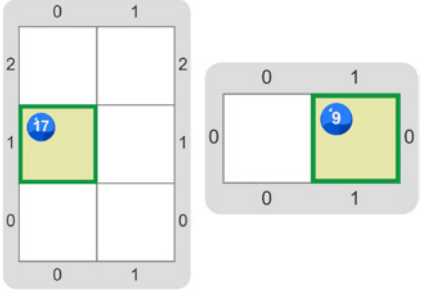
Por su parte, los operadores permiten realizar operaciones sobre ciertos valores. En este caso, usaremos algunos para realizar operaciones aritméticas.

Comenzamos la actividad repartiendo la ficha a los estudiantes y les indicamos que carguen el proyecto "El cabezal juega a ser mimo". Se encontrarán con un tablero inicial en el que el cabezal está ubicado sobre una celda que contiene una cierta cantidad de bolitas azules. El objetivo consiste en colocar la misma cantidad de bolitas rojas en esa celda.



Tablero inicial y tablero final de la primera consigna

Antes de que comiencen a construir sus programas, les pedimos que presionen varias veces el botón *Ejecutar*. Verán entonces que el tablero va cambiando: varía tanto su dimensión como la cantidad de bolitas azules que hay en la celda bajo el cabezal. El programa que construyan debe funcionar siempre, independientemente de cuál sea el tablero inicial.



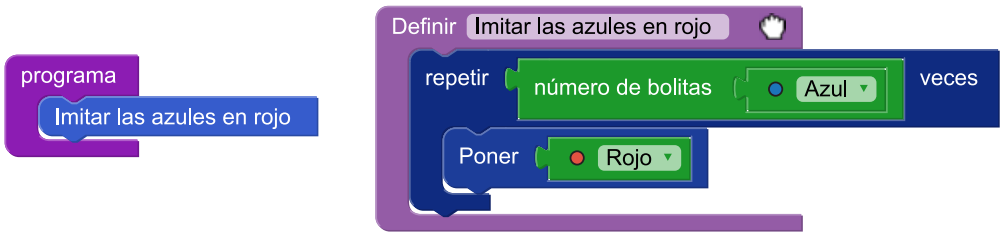
Posibles tableros iniciales

Preguntamos "¿Cuántas bolitas rojas tenemos que agregar?". Depende de con cuántas azules nos encontremos. Como *a priori* no se conoce este valor, con lo estudiado hasta el momento no alcanza para resolver el problema. Invitamos a los estudiantes a que exploren el entorno en busca de algún bloque que pueda resultarles útil.

En la categoría *Expresiones* > *Sensores* del menú lateral izquierdo se encuentra el bloque `número de bolitas [ ]`. Para usarlo, debemos indicar un valor de tipo *color*. Por ejemplo, `número de bolitas [Azul]` nos permite averiguar cuántas bolitas de color azul hay en la celda bajo el cabezal. Es importante notar que este bloque siempre arroja como resultado un valor de tipo *número*.

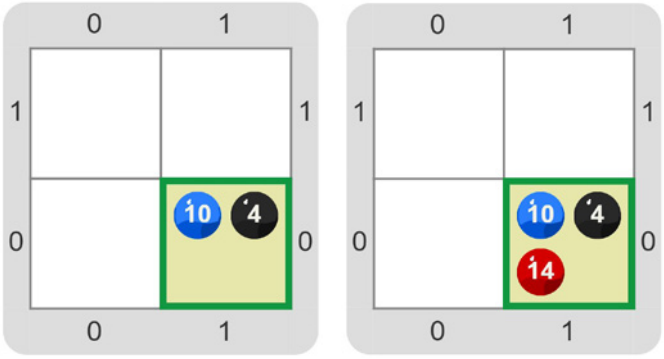
Sensor de número de bolitas de un color

Combinando `número de bolitas [ ]` con `repetir [ ] veces` podemos llegar a una solución.



Solución de la primera consigna

Una vez que todos hayan construido una solución adecuada, les indicamos que resuelvan la segunda consigna, para la que utilizarán el proyecto "El mimo que suma". En este caso, el objetivo es poner tantas bolitas rojas como haya de azules y negras. Por ejemplo, si en el tablero inicial hay 10 bolitas azules y 4 negras, al final deberá haber 14 rojas. Al igual que en la consigna previa, el tablero inicial cambia en distintas ejecuciones.

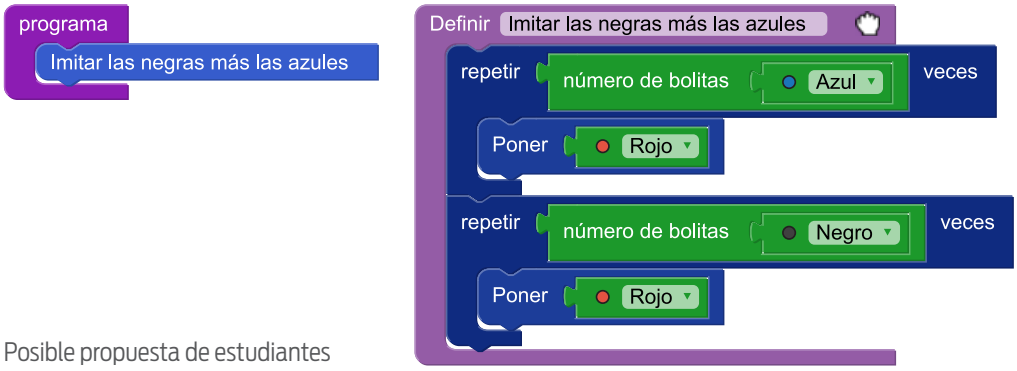


Posible tablero inicial y tablero final de la segunda consigna

Además de un tablero, en el espacio del programa encontrarán un programa incompleto que deben completar para resolver el desafío. Concretamente, tendrán que programar el cuerpo principal del programa y el procedimiento `Imitar las negras más las azules`.

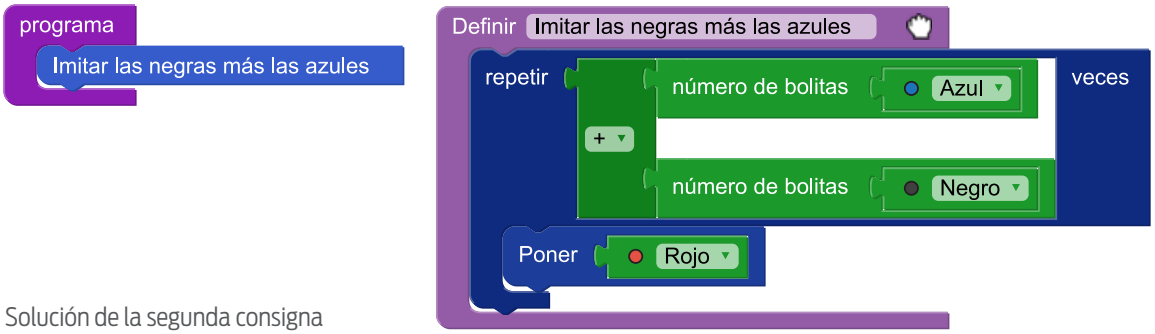
Espacio del programa al cargar el proyecto "El mimo que suma"

Puede ocurrir que algún estudiante utilice dos veces el comando `repetir [ ] veces`: la primera para agregar tantas bolitas rojas como azules haya en el tablero y la segunda para completar de acuerdo con la cantidad de negras. Si bien con este programa se alcanza el tablero final esperado, la consigna pide en forma explícita que se use `repetir [ ] veces` una sola vez. En este caso, debemos indicarle al alumno que su programa no se adecúa a las pautas de la consigna.



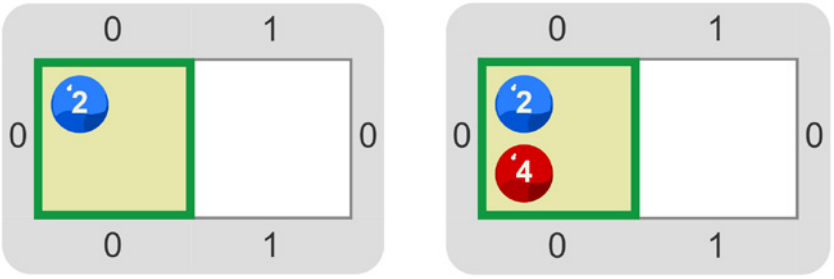
Posible propuesta de estudiantes

Al explorar el entorno, en la categoría *Expresiones* > *Operadores*, los estudiantes encontrarán un bloque para realizar operaciones aritméticas, una de las cuales es sumar dos valores. En este caso, lo usaremos para sumar las cantidades de bolitas azules y negras, y lo combinaremos con `repetir [ ] veces` para resolver el desafío.



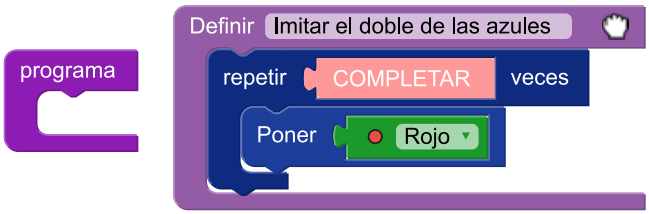
Solución de la segunda consigna

La tercera y última consigna trabaja con el proyecto "Supermimo multiplica". En este caso, en el tablero final tiene que quedar el doble de bolitas rojas que de azules. Nuevamente, el programa tiene que funcionar para distintos tableros iniciales.



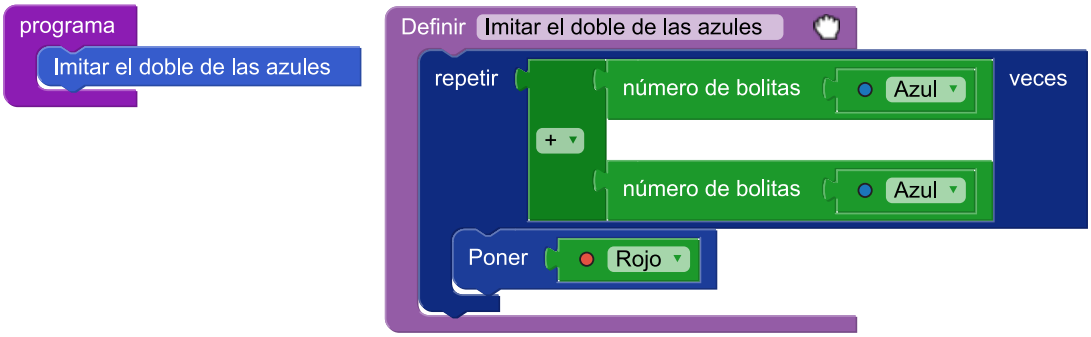
Posible tablero inicial y tablero final de la tercera consigna

Al abrir el proyecto, se encontrarán con un programa al que le falta completar el cuerpo principal del programa y un procedimiento llamado `Imitar el doble de las azules`. Además de la restricción de no poder usar más de una vez `repetir [ ] veces`, en este caso tampoco puede haber más de una ocurrencia tanto de `Poner [ ]` como de `número de bolitas [ ]`.



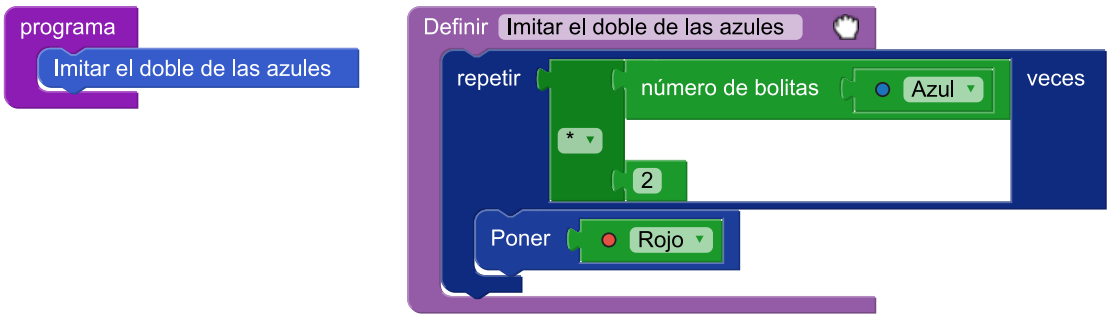
Espacio del programa al cargar el proyecto "Supermimo multiplica"

Podría ocurrir que un estudiante proponga un programa que calcule la multiplicación por 2 usando el operador de suma. Sin embargo, para hacerlo hay que utilizar 2 veces el bloque `número de bolitas [ ]`, y la consigna prohíbe explícitamente esta posibilidad.



Posible propuesta de estudiantes

Al explorar el entorno, los estudiantes podrán observar que multiplicar dos valores es una de las opciones del bloque para realizar operaciones aritméticas. El desafío, entonces, puede resolverse de manera muy similar al anterior.



Solución de la tercera consigna

CIERRE

Reflexionamos junto con los estudiantes sobre las nuevas herramientas usadas en esta actividad. Por un lado, el sensor `número de bolitas [ ]` nos permitió conocer información sobre el estado del tablero: el resultado que arroja refleja la cantidad de bolitas de un color que se encuentran en la celda sobre la que está el cabezal. En general, los sensores permiten obtener información sobre el entorno. Además, para resolver los desafíos, hizo falta llevar a cabo operaciones aritméticas, como sumar y multiplicar. Tanto al usar valores literales como sensores y operadores, se obtiene un valor. A todas las formas de denotar valores se las llama *expresiones*.

NOMBRE Y APELLIDO:

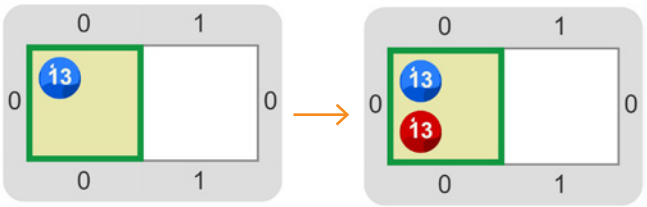
CURSO: FECHA:

EL CABEZAL PUEDE CONTAR

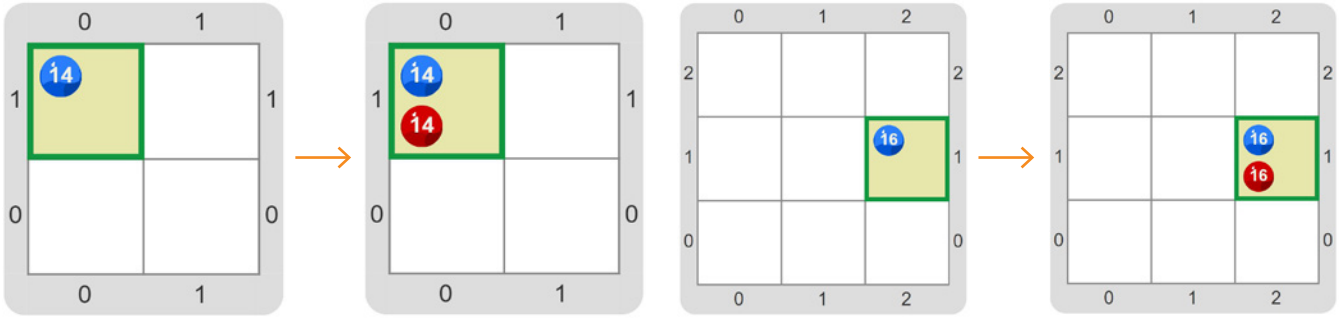
Los mimos son actores que se valen exclusivamente de gestos y de movimientos corporales para actuar ante el público. Algunos se especializan en imitar a otras personas que ven a su alrededor. ¡Ahora vamos a hacer que las bolitas rojas hagan de mimos imitando a las de las de otros colores!

1. Abrí el proyecto "El cabezal juega a ser mimo" y construí un programa que ponga bolitas rojas para igualar la cantidad de bolitas azules.

¡ATENCIÓN!
La cantidad de bolitas azules va variando de ejecución en ejecución. Te mostramos algunos tableros iniciales y sus correspondientes tableros finales.

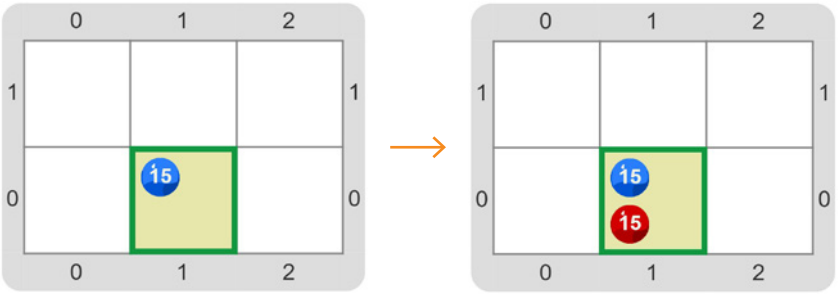


Tablero inicial → Tablero final



Tablero inicial → Tablero final | Tablero inicial → Tablero final

Recordá que el mismo programa tiene que servir para transformar cualquiera de los tableros iniciales. Probalo cuantas veces necesites.

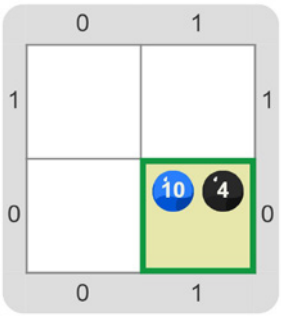


Tablero inicial → Tablero final

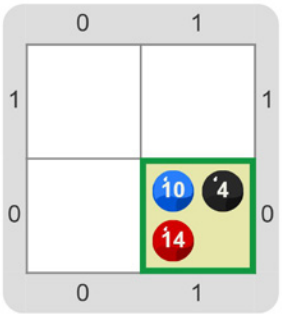
NOMBRE Y APELLIDO:

CURSO: FECHA:

2. Ahora abrí el proyecto "El mimo que suma" y hacé un programa que ponga tantas bolitas rojas como haya de azules y negras. El desafío es que en tu programa uses una única repetición. Buscá una forma de que el número de bolitas azules y el número de bolitas negras se puedan usar juntos en la repetición. Te mostramos algunos tableros iniciales y finales:

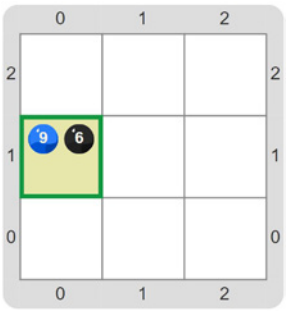


Tablero inicial

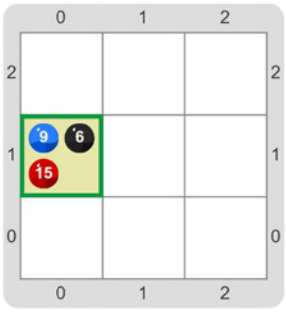


Tablero final

Explorá el entorno para ver si hay alguna herramienta nueva que te ayude a resolver el desafío.

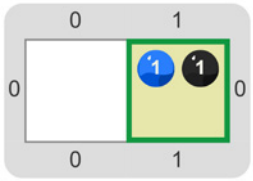


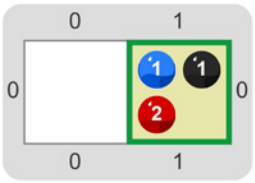
Tablero inicial



Tablero final

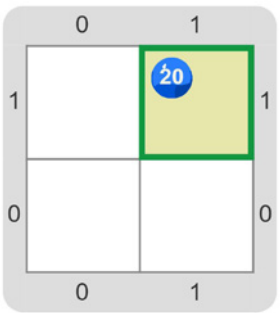
Tablero inicial



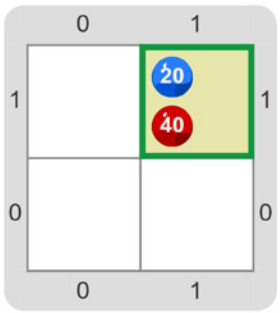


Tablero final

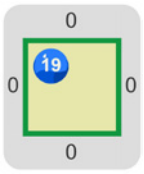
3. Abrí el proyecto "Supermimo multiplica" y completá el programa. Esta vez hay que poner el doble de bolitas rojas que de azules. El desafío, nuevamente, es hacerlo con una única repetición. Además, la solución debe usar una sola vez el comando `Poner [ ]` y una sola vez el sensor `número de bolitas [ ]`. Mirá algunas de las combinaciones de tablero inicial y final.



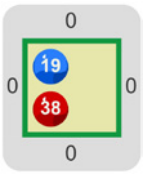
Tablero inicial



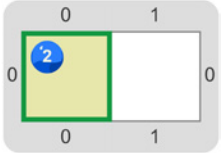
Tablero final



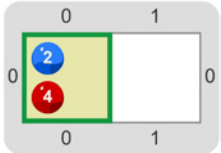
Tablero inicial



Tablero final



Tablero inicial



Tablero final

4. ¿Qué bloques nuevos usaste para resolver los desafíos?

Actividad 3
Atrapado con salida

DE A DOS

OBJETIVOS

- Presentar el uso de operadores sobre direcciones en Gobstones.
- Usar funciones disponibles en la biblioteca.

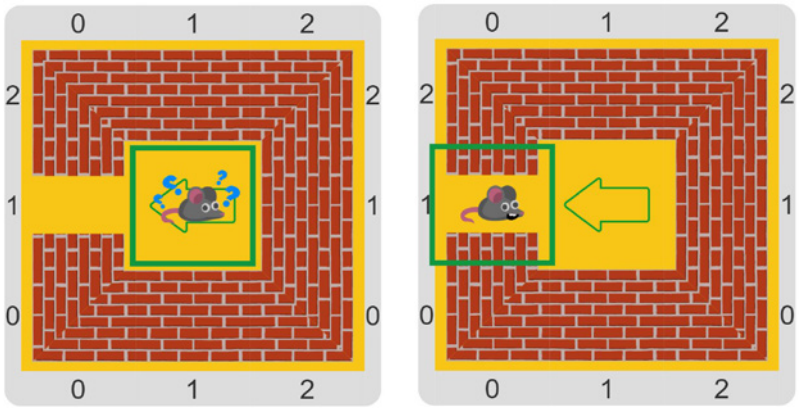
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

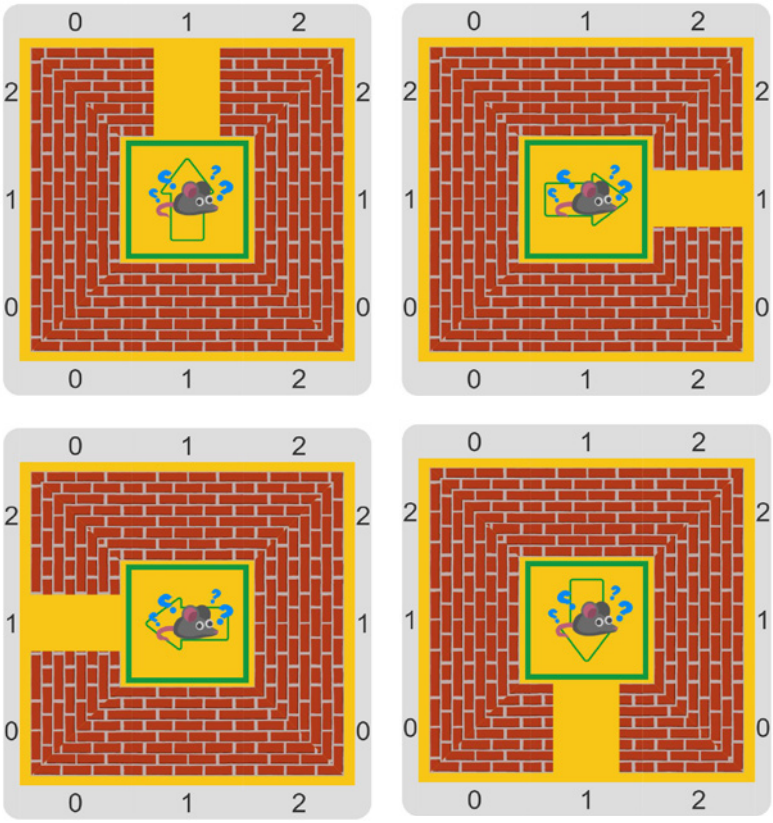
En esta actividad, por un lado, los estudiantes conocerán el operador de Gobstones `opuesto [ ]`, que, dada una dirección, retorna la dirección opuesta. Por otro, también usarán una función. En Gobstones, las funciones son una forma de definir expresiones nuevas. En general, el programador es quien las define. Sin embargo, algunas actividades –como esta– vienen con funciones en la biblioteca, disponibles para usar.

Comenzamos repartiendo la ficha de la actividad a los estudiantes y les pedimos que resuelvan la primera consigna. Para hacerlo, tienen que trabajar sobre el proyecto "Escapar de la jaula". Al abrirlo encontrarán un tablero de 3 × 3 que, usando una vestimenta, muestra a un ratón encerrado en una jaula. El objetivo es armar un programa para que el roedor pueda escapar.



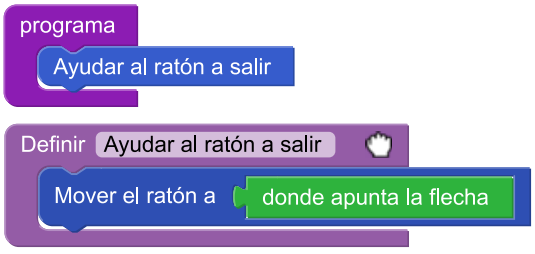
Tablero inicial y tablero final de "Escapar de la jaula"

No se puede hacer ninguna presunción sobre dónde está la salida debido a que hay distintos tableros iniciales. El programa debe funcionar para todos ellos. Como puede observarse, en todos los casos el ratón comienza parado sobre una flecha que indica la dirección de escape.



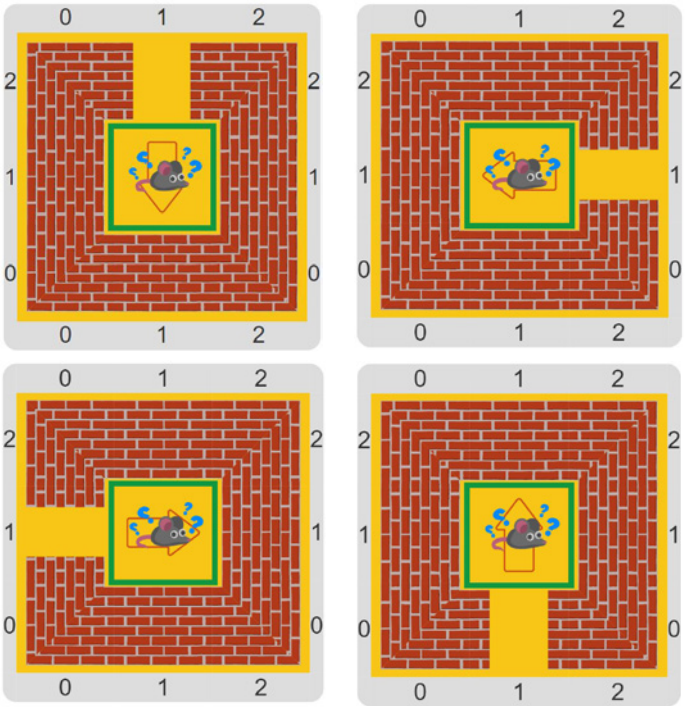
Tableros iniciales de "Escapar de la jaula"

Al explorar el entorno, los estudiantes encontrarán que en *Biblioteca* > *Procedimientos* tienen disponible el procedimiento `Mover al ratón a [ ]` y en *Biblioteca* > *Funciones* la función `donde apunta la flecha`. El procedimiento desplaza al roedor en la dirección indicada y la función retorna un valor de tipo *dirección* que indica hacia dónde se encuentra la salida. Al combinarlos, se llega a una solución.



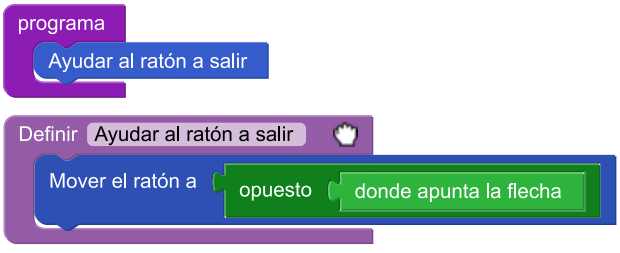
Solución de "Escapar de la jaula"

Una vez que todos hayan completado sus programas, les indicamos que trabajen sobre la segunda consigna. En este caso deberán abrir el proyecto "La flecha enloqueció". Nuevamente, el objetivo es ayudar al ratón para que escape de la jaula. Sin embargo, en todos los tableros iniciales la flecha apunta en la dirección opuesta a la salida. Por lo tanto, mover al ratón hacia donde apunta la flecha no resuelve el problema en este caso.



Tableros iniciales de "La flecha enloqueció"

En la categoría *Expresiones* > *Operadores* se encuentra `opuesto [ ]`, que, a partir de una dirección, retorna la dirección opuesta. Es decir, `opuesto [Norte]` es `Sur`, `opuesto [Este]` es `Oeste`, etc. Usando este operador podemos modificar ligeramente el programa de la consigna anterior para resolver el desafío.



Solución de "La flecha enloqueció"

CIERRE

Como cierre, reflexionamos con los alumnos sobre la diferencia entre los procedimientos y las funciones que se encuentran en la biblioteca. Si bien están construidos *ad hoc* para este proyecto en particular, los procedimientos realizan acciones que tienen efectos sobre el tablero, mientras que las funciones devuelven un valor. Comentamos que, así como existen operadores sobre *números*, también los hay sobre otros tipos de datos, como por ejemplo las *direcciones*.

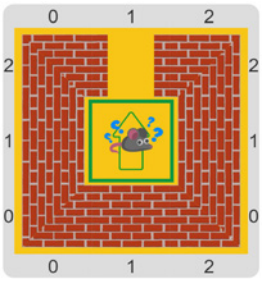
NOMBRE Y APELLIDO:

CURSO: FECHA:

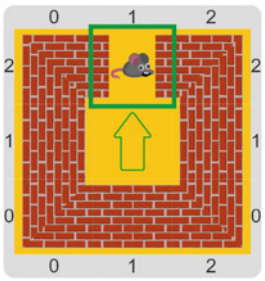
ATRAPADO CON SALIDA

Un ratón está atrapado en una jaula y necesita tu colaboración para encontrar la salida. Tenés que construir un procedimiento `Ayudar al ratón a salir` que le permita escapar. En la biblioteca hay un procedimiento para mover al ratón, pero la dirección que tenés que darle como argumento depende de cada jaula.

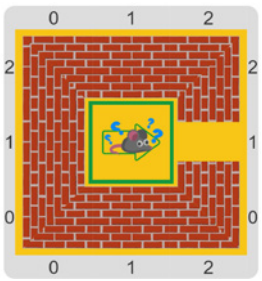
1. Abrí el proyecto "Escapar de la jaula" y observá que, en el piso, hay una flecha que le indica al ratón la dirección que debe tomar. Estos son los posibles tableros iniciales y sus correspondientes tableros finales.



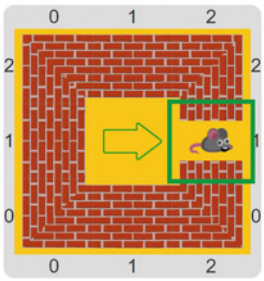
Tablero inicial
Salida al norte



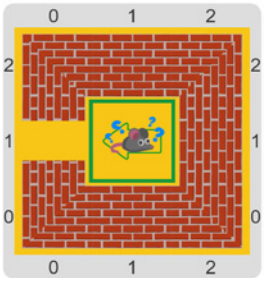
Tablero final
El ratón llegó a la salida



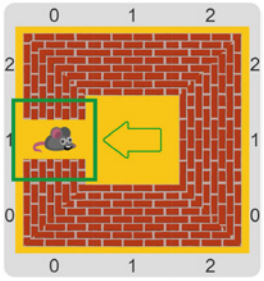
Tablero inicial
Salida al este



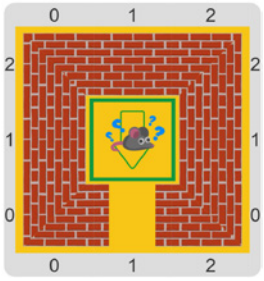
Tablero final
El ratón llegó a la salida



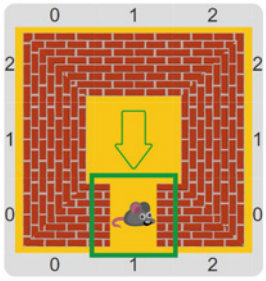
Tablero inicial
Salida al oeste



Tablero final
El ratón llegó a la salida



Tablero inicial
Salida al sur



Tablero final
El ratón llegó a la salida

Copiá acá abajo el procedimiento `Ayudar al ratón a salir` que armaste.

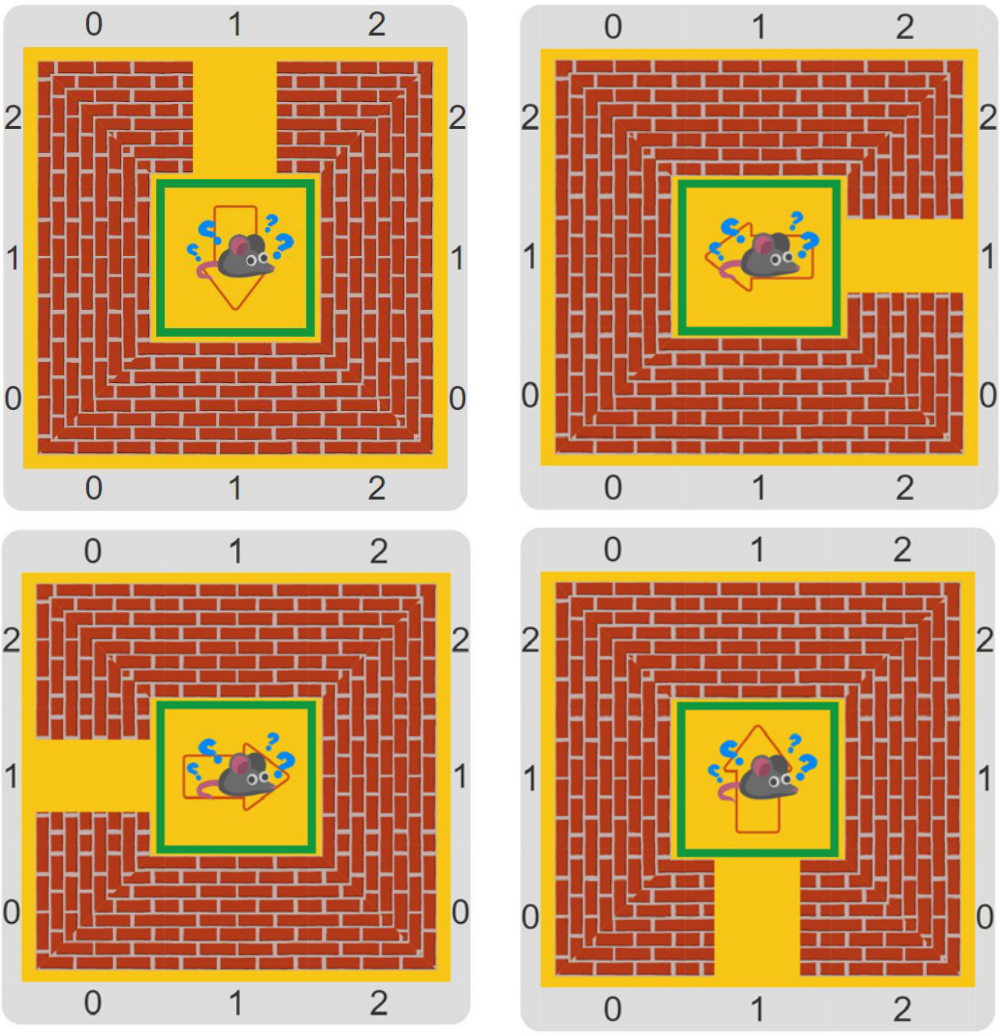
¿ADÓNDE APUNTA LA FLECHA? ¿CÓMO MOVEMOS AL RATÓN?
Explorá el entorno para descubrirlo.

NOMBRE Y APELLIDO:

CURSO: FECHA:

2. Abrí el proyecto "La flecha enloqueció". La flecha que tiene que apuntar a la salida ¡enloqueció! Ahora apunta en la dirección opuesta. ¿Qué hacemos?

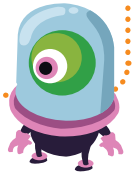
Mirá los tableros iniciales.



En este caso, ¿cómo resolviste `Ayudar al ratón a salir`?

¡SEGUÍ EXPLORANDO!

La flecha apunta en la dirección opuesta. ¿Cómo le indicamos al programa hacia dónde mover al ratón?



Actividad 4
Laberinto

DE A DOS

OBJETIVO

- Combinar el uso de procedimientos, repeticiones y funciones.

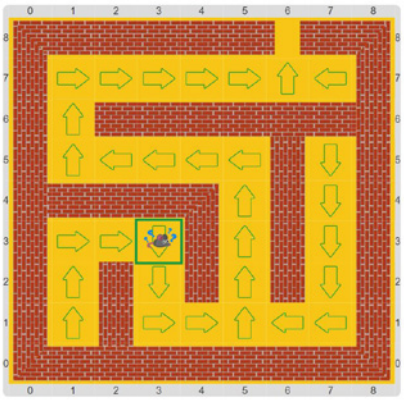
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

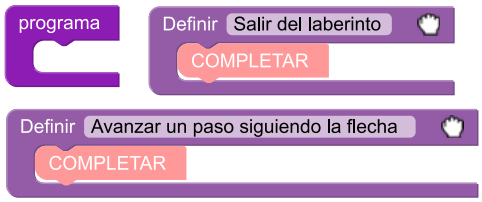
En esta actividad, los estudiantes combinarán el uso de procedimientos, repeticiones y funciones.

Comenzamos repartiendo la ficha de la actividad y les pedimos que carguen el proyecto "Laberinto". Encontrarán un tablero de 9 × 9 que muestra a un ratón atrapado en un laberinto. Antes de que comiencen a construir sus programas, les sugerimos que presionen el botón *Ejecutar* varias veces para que observen que hay distintos tableros iniciales: si bien son diferentes, en todos ellos el ratón se encuentra a 20 celdas de la salida. Además, en cada celda hay una flecha que indica en qué dirección debe moverse el roedor. El objetivo es que construyan un programa para que el pequeño mamífero pueda escapar, independientemente de cuál sea el laberinto en el que se encuentre.



Uno de los tableros iniciales

En el espacio del programa, verán dos procedimientos incompletos: `Salir del laberinto` y `Avanzar un paso siguiendo la flecha`. Deben completarlos, además del cuerpo principal del programa, para resolver el desafío.



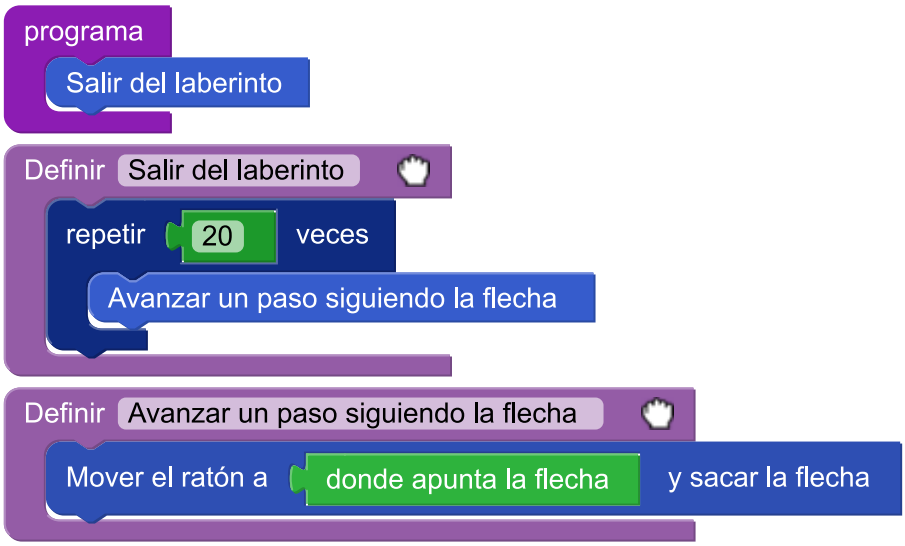
Espacio del programa al abrir el proyecto

En la biblioteca están disponibles el procedimiento `Mover al ratón a [ ] y sacar la flecha` y la función `donde apunta la flecha`. El primero desplaza al ratón en la dirección indicada y la segunda retoma la dirección a la que apunta la flecha dibujada en la celda en la que se encuentra el roedor.

Procedimiento y función disponibles en la biblioteca

¿Por qué el procedimiento `Mover al ratón a [ ] y sacar la flecha` incluye *y sacar la flecha*? Porque al eliminar las flechas, las celdas quedan vacías y se observa mejor el recorrido que realiza el ratón para salir del laberinto.

Teniendo en cuenta que el ratón siempre debe desplazarse 20 veces, una solución consiste en usar la instrucción `repetir [20] veces` con el procedimiento `Avanzar un paso siguiendo la flecha`. Por otra parte, este último puede valerse de la función `donde apunta la flecha` para desplazar al ratón en la dirección correcta.



Solución que resuelve el desafío

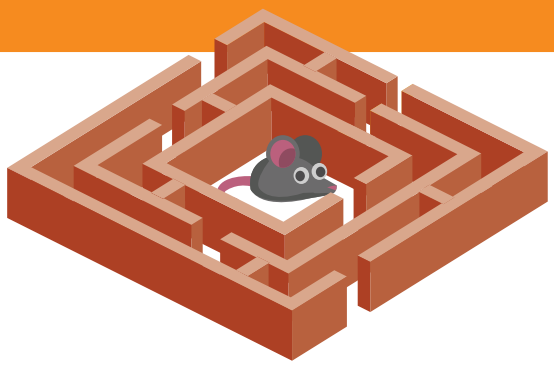
CIERRE

Reflexionamos sobre las posibilidades que brindan las distintas herramientas usadas en la actividad: procedimientos, repeticiones y funciones. Los procedimientos resultan de gran utilidad para dividir el problema original en otros más simples. Además, al usar nombres adecuados, el programa que se obtiene es fácil de entender. Por otra parte, las repeticiones permiten describir de modo sintético una serie de acciones que deben repetirse. Por último, las funciones nos permiten obtener valores que luego usamos para resolver problemas.

NOMBRE Y APELLIDO:

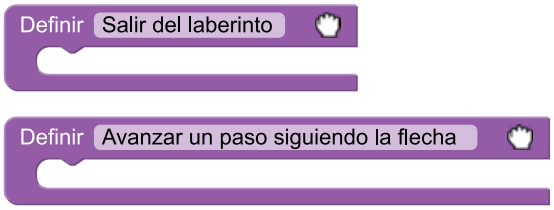
CURSO: FECHA:

LABERINTO

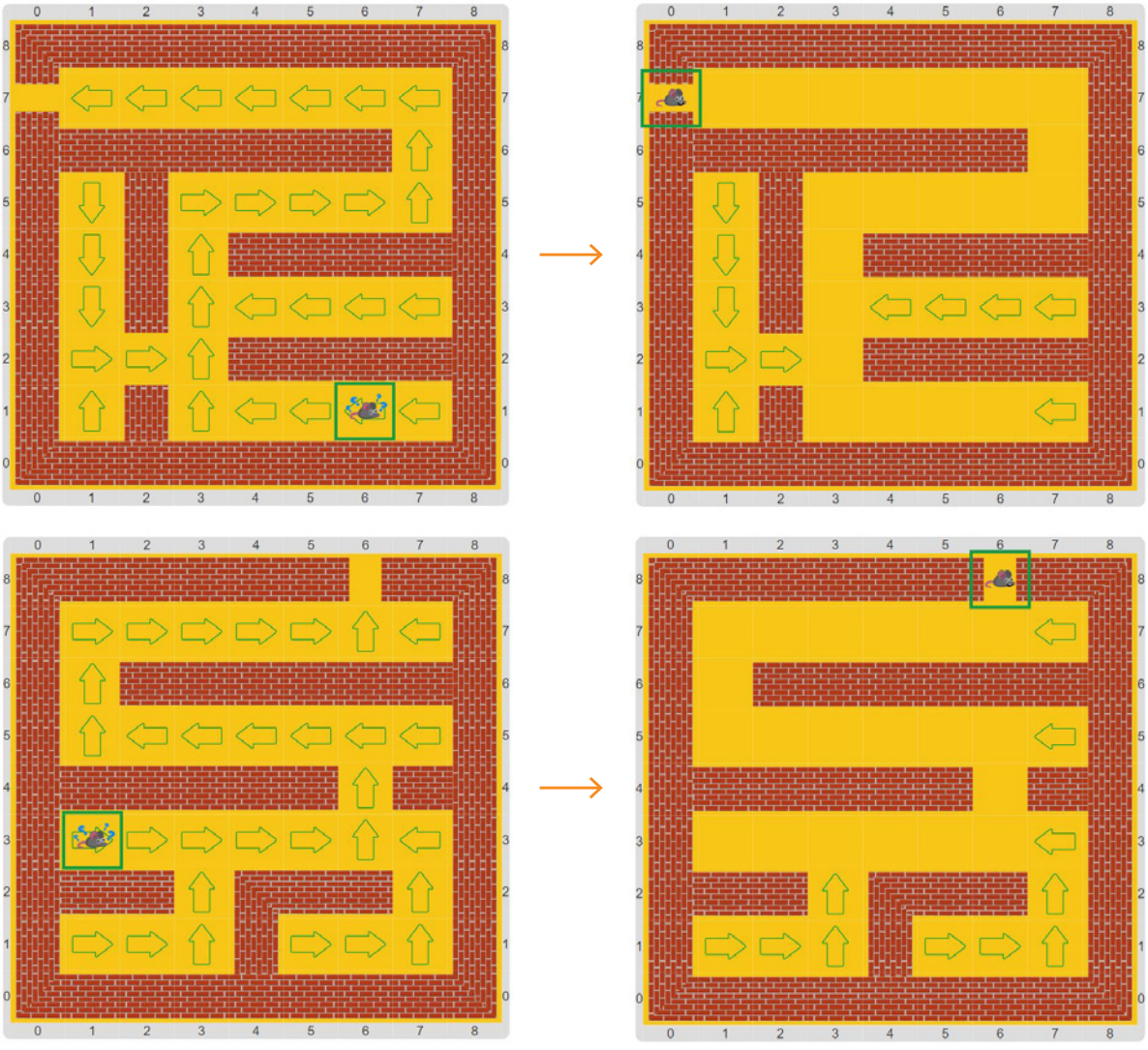


¡Un ratoncito está atrapado en un laberinto y necesita de tu ayuda para poder escapar! Por suerte, hay flechas que marcan el camino hacia la salida, que se encuentra a solo 20 pasos.

1. Abrí el proyecto "Laberinto", y completá los procedimientos `Salir del laberinto` y `Avanzar un paso siguiendo la flecha` de modo que le indiquen al ratón lo que debe hacer para escapar.

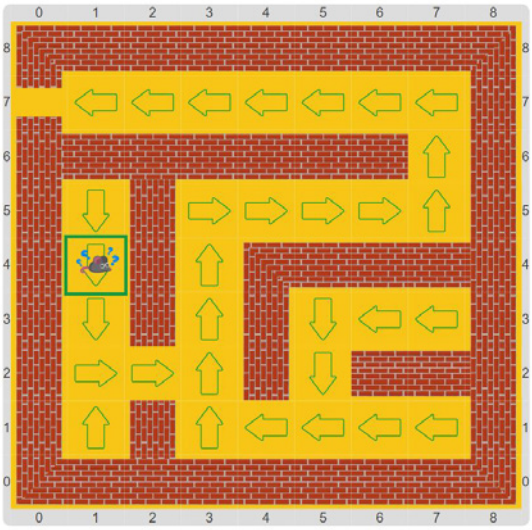


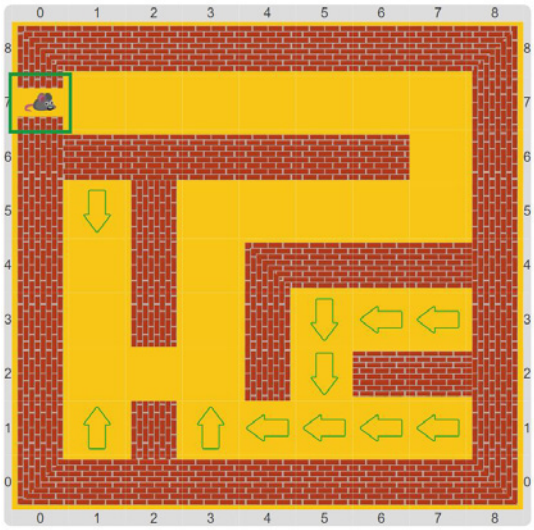
Te mostramos algunos laberintos y cómo tienen que quedar al terminar:



NOMBRE Y APELLIDO:

CURSO: FECHA:





Revisá los procedimientos y las funciones de la biblioteca, ya que pueden ayudarte a completar el desafío.

Copiá acá abajo cómo quedó tu programa.

Secuencia Didáctica 2

ALTERNATIVA CONDICIONAL

Las condiciones en las cuales se ejecutan los programas suelen ser diversas. Por lo tanto, al programar, debemos evaluar los distintos escenarios que pueden presentarse y las acciones adecuadas a realizar en cada caso. En esta secuencia didáctica se presenta la **alternativa condicional**, que es una herramienta de los lenguajes de programación para que un programa se comporte de uno u otro modo de acuerdo con ciertas condiciones de los datos. Además, durante la ejercitación, se presentarán otros sensores de Gobstones, el tipo de dato booleano (cuyos únicos dos posibles valores son *verdadero* y *falso*), el operador de negación y operadores que permiten comparar números.

OBJETIVOS

- Presentar la alternativa condicional.
- Introducir el tipo de datos booleano.

Actividad 1
Apagar la luz o prenderla

DE A DOS

OBJETIVOS

- Presentar el uso de la alternativa condicional en programas.
- Presentar el tipo de datos booleano.

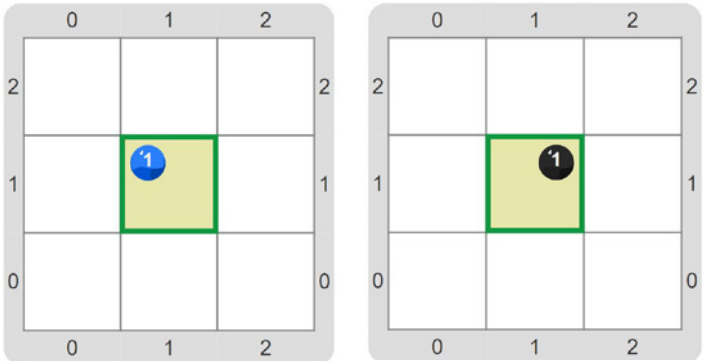
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

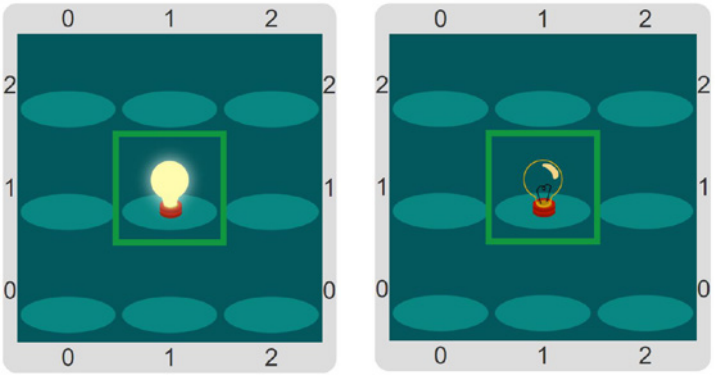
En esta actividad, los estudiantes se enfrentarán por primera vez a un problema cuya resolución requiere el uso de una alternativa condicional. Estas permiten que un programa evalúe una condición y, dependiendo de si es verdadera o falsa, realice ciertas acciones u otras.

Comenzamos pidiéndoles a los estudiantes que carguen el proyecto "Apagar la luz o prenderla". Encontrarán uno de dos posibles tableros iniciales. En uno de ellos hay una bolita azul en la celda (1,1) y, en el otro, una negra. Les sugerimos que presionen varias veces el botón *Ejecutar* para ver ambos.



Posibles tableros iniciales

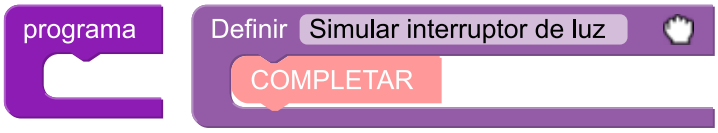
A continuación, les indicamos que activen la vestimenta –presionando el ícono del ojo– sobre cada uno de los tableros iniciales. Notarán entonces la asociación entre bolitas e imágenes que hace la vestimenta en esta actividad: una bolita azul representa una lamparita encendida y una negra, una lamparita apagada.



Posibles tableros iniciales, con vestimenta

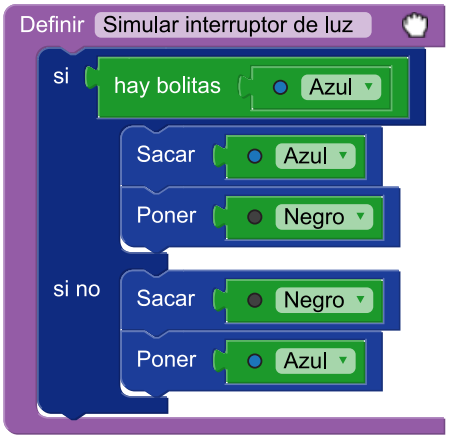
El objetivo es simular un interruptor de luz. Si en el tablero inicial hay una lamparita encendida, hay que apagarla; si hay una apagada, hay que encenderla. A diferencia de lo que ocurría en actividades anteriores, en esta no hay procedimientos en la biblioteca que simplifiquen la tarea. Por lo tanto, para resolver el desafío es necesario tener presente cómo la vestimenta representa cada estado de la lamparita: para pasar de uno a otro hay que poner y sacar bolitas.

En el espacio de programa se encontrarán con el procedimiento `Simular interruptor de luz`, que deberán completar. Invitamos a los estudiantes a resolver la consigna.



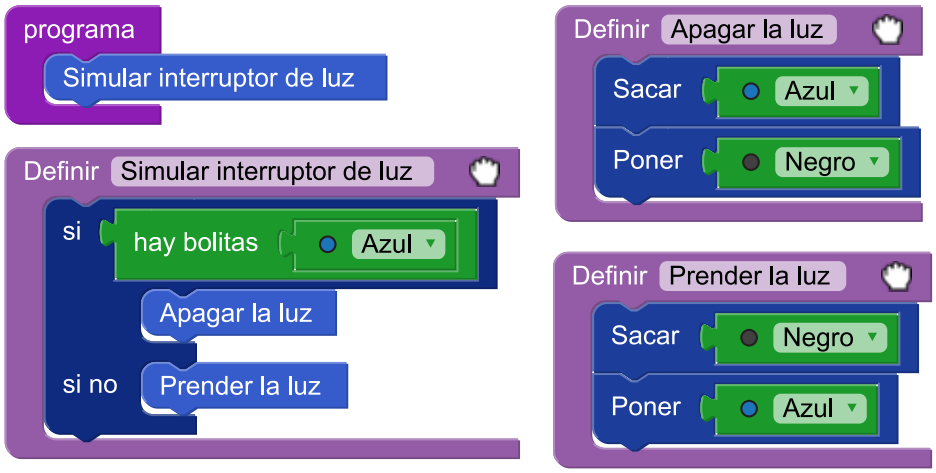
Espacio del programa al abrir el proyecto

Para saber qué acciones debe realizar el programa, hace falta evaluar el estado inicial de la lamparita, lo que no es posible con lo estudiado hasta el momento. Al explorar el entorno, en *Comandos* > *Alternativas* los estudiantes encontrarán el bloque `si [ ] / si no`. Este permite chequear una condición e indicar qué acciones seguir tanto si es verdadera como si es falsa. En este caso, la condición es "la lamparita está encendida". Recuperando la representación que define la vestimenta, en términos de bolitas se traduce en examinar si en la celda hay o no una bolita azul, para lo cual nos valdremos del sensor `hay bolitas [ ]` usando `Azul` como argumento. En caso afirmativo, hay que apagarla –sacar la bolita azul y colocar una negra–; caso contrario, encenderla –sacar la bolita negra y colocar una azul–.



Posible propuesta de estudiantes

Si bien `Simular interruptor de luz` modifica el estado de la lamparita, al leerlo no resulta claro qué es lo que hace. Una opción superadora consiste en definir dos procedimientos, `Apagar la luz` y `Prender la luz`, que encapsulen cada una de estas dos acciones. Aquí puede verse una versión completa del programa.



Programa que resuelve el desafío

Booleanos

Los valores de verdad se llaman *booleanos* en honor a George Boole, matemático y filósofo inglés del siglo XIX que sentó las bases de la aritmética de computadoras. Se lo considera uno de los padres de la computación moderna.

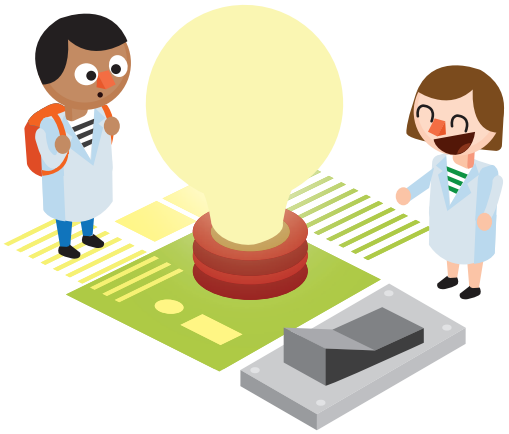
CIERRE

Como cierre, reflexionamos con los estudiantes sobre las posibilidades que brinda el uso de las alternativas condicionales. Por último, presentamos el tipo de datos booleano, cuyos únicos valores posibles son *verdadero* y *falso*. El sensor `hay bolitas [ ]` siempre arroja un resultado de este tipo: *verdadero* cuando bajo el cabezal hay una bolita del color usado como argumento y *falso* en caso contrario.

NOMBRE Y APELLIDO:

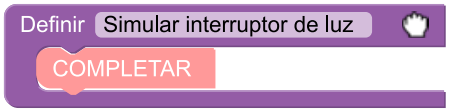
CURSO: FECHA:

APAGAR LA LUZ O PRENDERLA



¿Cómo funciona un interruptor de luz? Al accionarlo, cambia el estado de una lamparita: si está apagada, la prende; si está prendida, la apaga.

1. Abrí el proyecto "Apagar la luz o prenderla", y completá el procedimiento `Simular interruptor de luz`. Tené en cuenta que, *a priori*, no sabemos si la lamparita está encendida o apagada. ¡El programa tiene que funcionar en ambos casos!

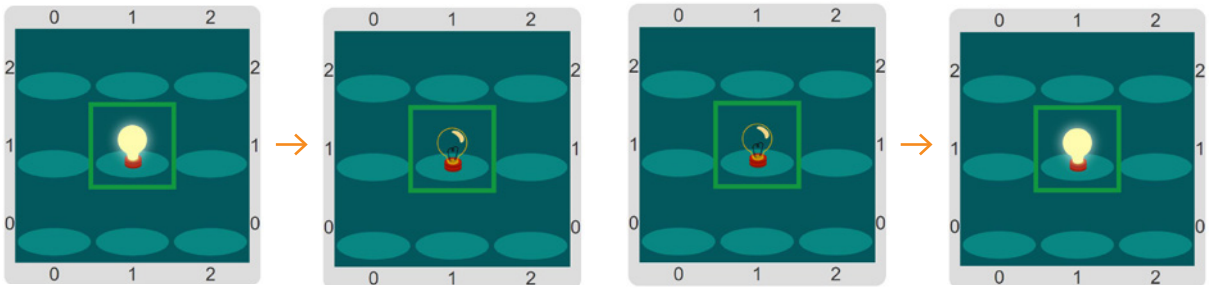


¡A TENER EN CUENTA LA VESTIMENTA!

No hay procedimientos en la biblioteca. Para resolver el desafío, hay que considerar la vestimenta de la actividad.

- → lamparita prendida
- → lamparita apagada

Los posibles tableros iniciales y sus correspondientes tableros finales son los siguientes:



Tablero inicial → Tablero final → Tablero inicial → Tablero final

Copiá acá abajo cómo quedó tu programa.

¿APAGAR O PRENDER?

¿Cómo sabemos si tenemos que prender o apagar la luz? Explorá el entorno para encontrar las herramientas que hacen falta. ¡Acordate de usar procedimientos!

Actividad 2
No me quiero caer

DE A DOS

OBJETIVOS

- Presentar el sensor de Gobstones `puede mover [ ]`.
- Ejercitar el uso de alternativas condicionales.

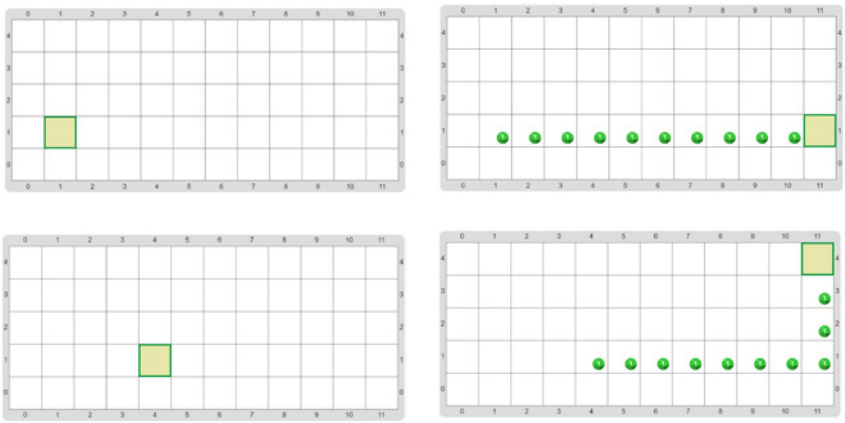
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

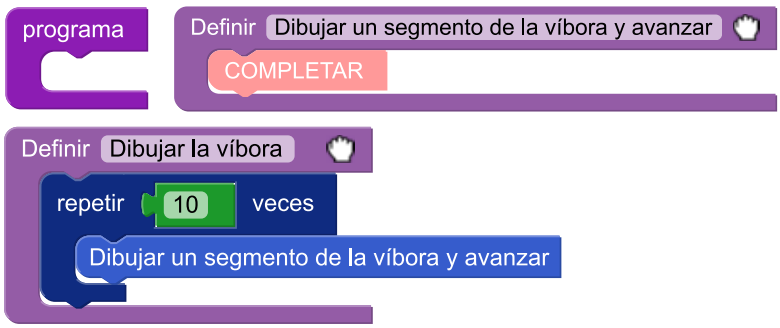
En esta actividad se seguirá trabajando el uso de alternativas condicionales. Además, se empleará el único sensor no utilizado hasta el momento: `puede mover [ ]`. Este sirve para determinar si el cabezal puede desplazarse en una cierta dirección sin caerse del tablero.

Comenzamos repartiendo la ficha y pidiéndoles a los estudiantes que abran el proyecto "No me quiero caer". Al hacerlo, encontrarán un tablero vacío. Les indicamos que presionen reiteradas veces el botón *Ejecutar* para observar que existen varios tableros iniciales: en algunos casos, varía la dimensión y, en otros, solo la posición inicial del cabezal. El objetivo es dibujar una víbora usando 10 bolitas verdes en dirección este y, en caso de alcanzar el borde, continuarla hacia el norte.



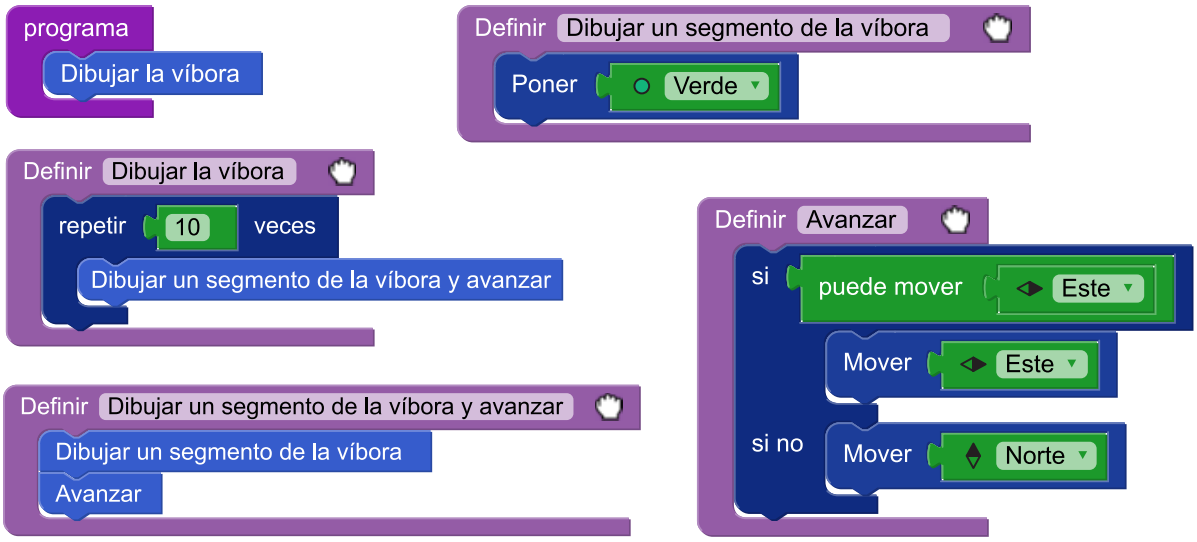
Dos tableros iniciales y sus correspondientes finales

Encontrarán un programa incompleto. El procedimiento `Dibujar la víbora` mediante una repetición simple, llama 10 veces a `Dibujar un segmento de la víbora y avanzar`, que en un principio se encuentra vacío.



Espacio del programa al abrir el proyecto

Para completar `Dibujar un segmento de la víbora y avanzar` se pueden crear dos procedimientos: `Dibujar un segmento de la víbora` y `Avanzar`. El primero, muy sencillo, se encarga de ubicar una bolita verde en la celda bajo el cabezal. En el segundo, como la dirección en la que hay que desplazar el cabezal depende de que haya o no una celda a la derecha de donde está posicionado, hay que usar una alternativa condicional. En caso de que sí haya una, hay que mover el cabezal hacia el este; en caso contrario, hacia el norte. Para poder determinar si tal celda existe, hay que usar el sensor `puede mover [ ]`. A continuación puede observarse una solución completa.



Programa que resuelve el desafío

CIERRE

Reflexionamos junto con los estudiantes sobre la utilidad de los sensores. En este caso, uno de ellos nos permitió escribir un programa de forma tal que el cabezal no se cayera del tablero. En general, son útiles para obtener información del ambiente mientras un programa se ejecuta. Esa información se puede usar, por ejemplo, para que nuestros programas "decidan" qué hacer a continuación o para realizar algún cálculo.

NOMBRE Y APELLIDO:

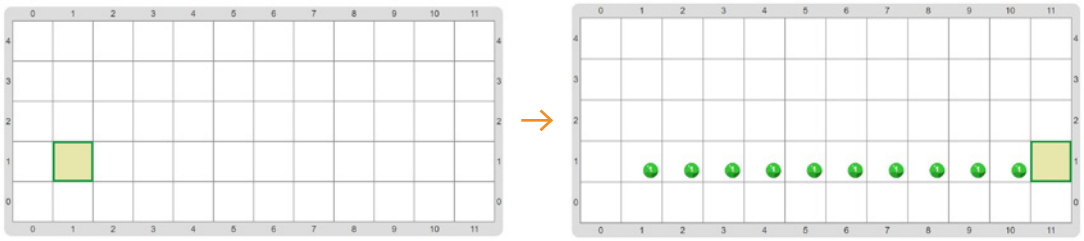
CURSO: FECHA:

NO ME QUIERO CAER

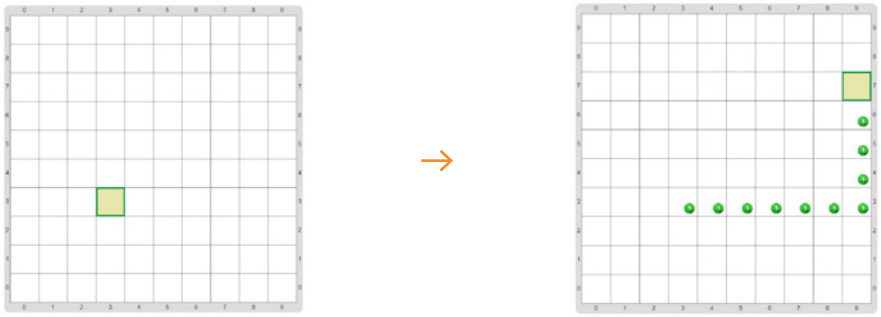
¡Las víboras vuelven más verdes que nunca!

1. Abrí el proyecto "No me quiero caer" y construí un programa para dibujar una víbora con 10 bolitas verdes. Tenés que colocar la primera bolita en la celda donde inicialmente se encuentra el cabezal y continuarla hacia el este. Si hay suficiente lugar, la víbora queda alineada; pero si en algún momento no puede moverse más hacia el este porque se acaba el tablero, dobla y sigue hacia el norte.

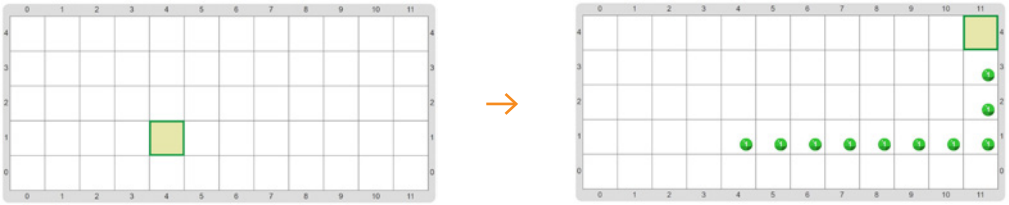
Las siguientes figuras muestran algunos tableros iniciales y los tableros finales correspondientes:



Tablero inicial (11 lugares al este) → Tablero final



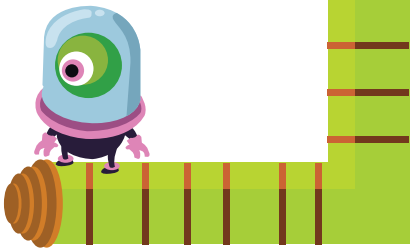
Tablero inicial (7 lugares al este) → Tablero final



Tablero inicial (8 lugares al este) → Tablero final

¿CÓMO EVITAR QUE EL CABEZAL HAGA BOOM?

Explorá el entorno en busca de algo que te permita determinar si el cabezal puede moverse en una dirección específica.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Copiá acá abajo cómo quedó tu programa.

Actividad 3
Que gane el mejor

DE A DOS

OBJETIVO

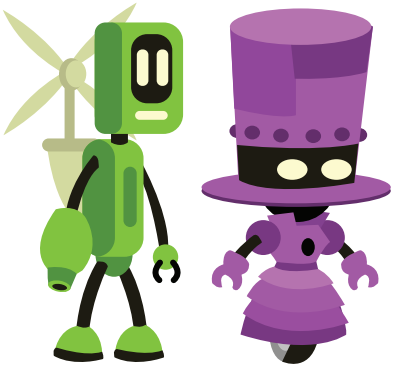
- Utilizar operadores relacionales en un programa.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

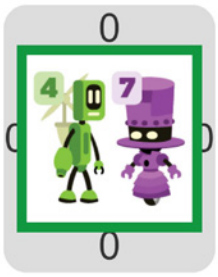
En esta actividad, los estudiantes usarán por primera vez operadores para comparar valores numéricos. De acuerdo al resultado, el programa se comportará de un modo u otro.



Verdolaga y Copa Alta

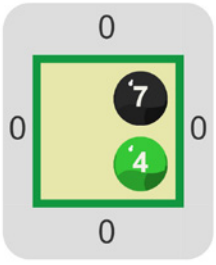
Comenzamos contándoles a los estudiantes sobre Verdolaga y Copa Alta, dos robots que fueron entrenados mediante técnicas de inteligencia artificial para enfrentarse en combate. Cuando luchan, el más entrenado gana. En esta actividad construirán un programa para simular un combate entre Verdolaga y Copa Alta.

Luego de presentar el universo de los androides, repartimos la ficha y les pedimos que carguen el proyecto "Que gane el mejor". Encontrarán un tablero de una única celda en la que aparecen Verdolaga y Copa Alta, cada uno con un número que indica su nivel de entrenamiento. Por ejemplo, si Verdolaga tiene nivel 4 y Copa Alta nivel 7, al enfrentarse Copa Alta vencerá.



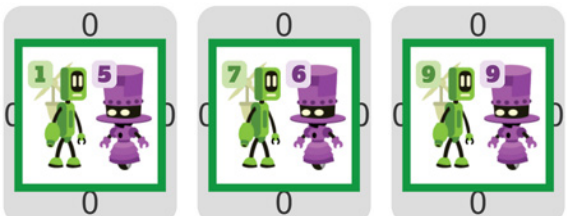
Posible tablero inicial

A continuación, les indicamos que presionen el ojo para ver cómo la vestimenta del proyecto representa a los robots y sus niveles de entrenamiento: la cantidad de bolitas verdes indica el nivel de Verdolaga y la cantidad de negras, el de Copa Alta.



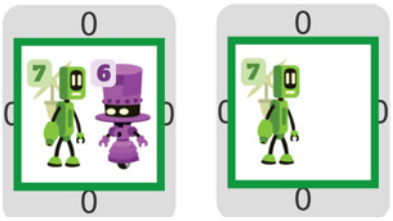
Tablero inicial, sin vestimenta

Además, les sugerimos que presionen varias veces el botón *Ejecutar* para ver los distintos tableros iniciales. En algunos, el robot Copa Alta está más entrenado; en otros, menos; y también hay tableros donde ambos están entrenados por igual.



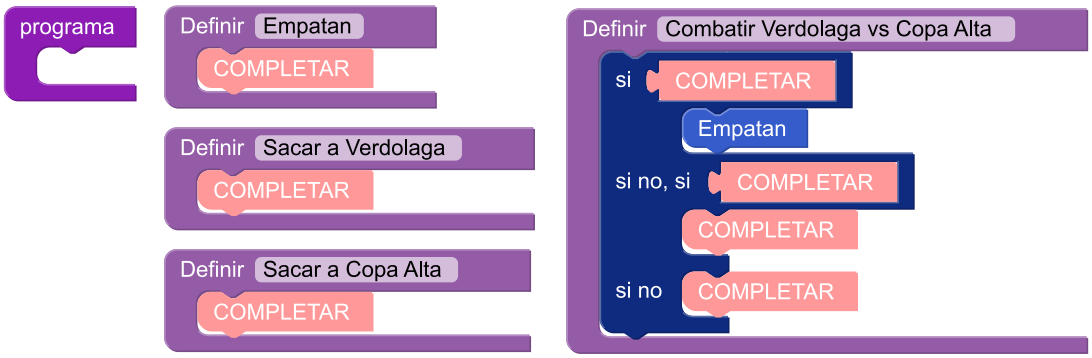
Posible tableros iniciales

El objetivo del programa es retirar del tablero al androide vencido en un combate. Por ejemplo, si en el tablero inicial Verdolaga tiene nivel 7 y Copa Alta nivel 6, al terminar la ejecución solo debe permanecer en el tablero Verdolaga, sin alterar su nivel de entrenamiento. En caso de que los dos estén igualmente entrenados, ambos deben retirarse.



Verdolaga vence a Copa alta

En el espacio del programa aparecerán 4 procedimientos: `Empatan`, `Sacar a Verdolaga`, `Sacar a Copa Alta` y `Combatir Verdolaga vs Copa Alta`. Los tres primeros están completamente vacíos, mientras que el último tiene una alternativa condicional que hay que completar.



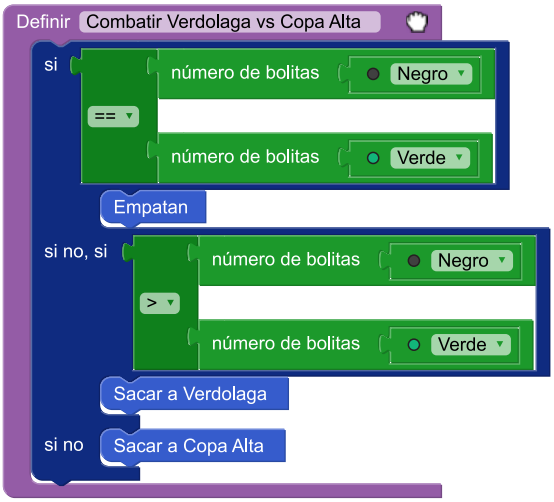
Espacio del programa al abrir el proyecto

En el procedimiento `Combatir Verdolaga vs Copa Alta` puede observarse un nuevo bloque que permite realizar una alternativa condicional: `si [ ] / si no, si [ ] / si no`. Se trata, en rigor, de una extensión de `si [ ] / si no` que permite evaluar varias condiciones –en lugar de una– para determinar qué acciones debe ejecutar un programa. En este caso resulta de utilidad debido a que un combate tiene tres posibles resultados: *(i)* empate, *(ii)* gana Verdolaga y *(iii)* gana Copa Alta.

El programa que se pide completar debe simular un combate entre Verdolaga y Copa Alta. Por lo tanto, el cuerpo principal del programa consiste en una invocación a `Combatir Verdolaga vs Copa Alta`.

Cuerpo principal del programa

Para completar `Combatir Verdolaga vs Copa Alta`, primero hay que incorporar la condición que determina que un combate resulta empatado. Esto ocurre cuando ambos tienen el mismo nivel de entrenamiento; en término de bolitas, cuando hay igual cantidad de verdes que de negras. Con el sensor `número de bolitas [ ]` se puede conocer el número de bolitas de cada color, pero ¿cómo se hace para ver si las cantidades coinciden? En *Expresiones > Operadores* se encuentra un bloque para realizar comparaciones entre números. Este permite determinar si dos números son iguales, si son distintos, si uno es mayor que otro, etc. Combinando este bloque con el sensor `número de bolitas [ ]` y los procedimientos que todavía falta terminar, se puede completar `Combatir Verdolaga vs Copa Alta`.



Procedimiento `Combatir Verdolaga vs Copa Alta`

En caso de empate, hay que retirar ambos robots del tablero. Por lo tanto, el procedimiento `Empatan` puede resolverse invocando `Sacar a Verdolaga` y `Sacar a Copa Alta`.

Procedimiento `Empatan`

Para completar el programa es conveniente usar el procedimiento `Sacar todas [ ]` disponible en *Biblioteca > Procedimientos*, que retira del tablero todas las bolitas de un color dado. Para sacar a Verdolaga retiramos todas las bolitas verdes; y para sacar a Copa Alta, las negras.

Procedimientos `Sacar a Verdolaga` y `Sacar a Copa Alta`

Una solución más declarativa consiste en agregar los procedimientos `Gana Verdolaga` y `Gana Copa Alta`. De este modo, queda mejor expresada la dinámica de los combates entre Verdolaga y Copa Alta.

Otro programa que resuelve el desafío

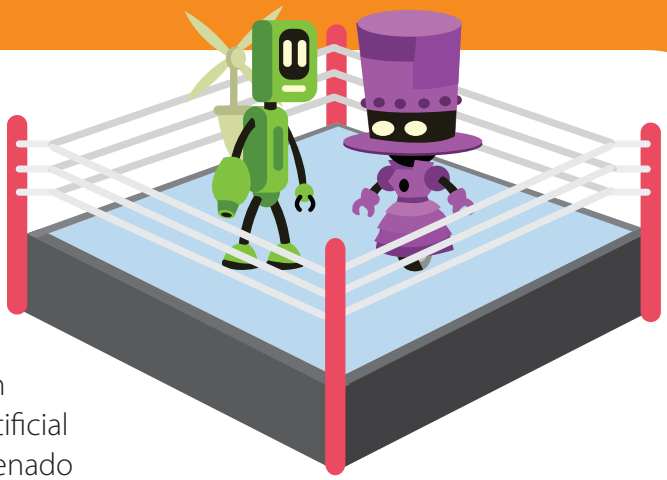
CIERRE

Para finalizar, reflexionamos con los estudiantes sobre los operadores de comparación de números. A diferencia de los usados previamente (como la suma o el producto, que dados dos números, arrojan otro número como resultado), al usar estos operadores se obtiene un valor booleano. El resultado de una comparación será o bien verdadero, o bien falso.

NOMBRE Y APELLIDO:

CURSO: FECHA:

QUE GANE EL MEJOR



Verdolaga y Copa Alta son dos robots que fueron entrenados mediante técnicas de inteligencia artificial para combatir. Cuando se enfrentan, el más entrenado gana. En esta actividad, tenés que construir un programa para simular un combate entre Verdolaga y Copa Alta.

1. Abrí el proyecto "Que gane el mejor" y construí un programa que retire del tablero al robot vencido en combate. Si los dos tienen el mismo nivel de entrenamiento, se produce un empate y ambos deben retirarse.

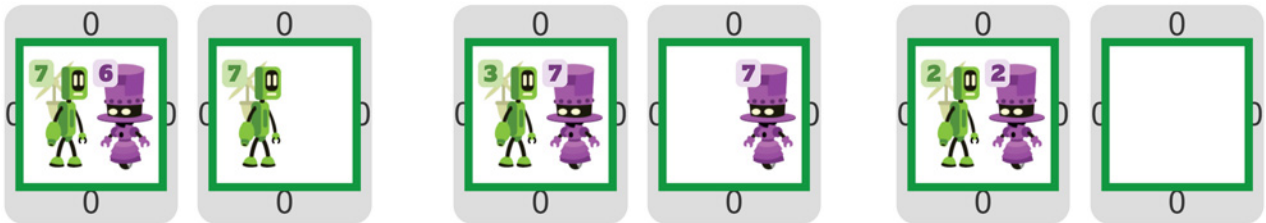
EL INICIAL DEL FINAL

Luego de ejecutar un programa, podés volver a ver el tablero inicial haciendo clic en Tablero inicial.

ALGUNAS AYUDITAS

- Activá y desactivá la vestimenta para ver cómo están representados los robots y sus niveles de entrenamiento.
- Buscá en el entorno algunos bloques que te permitan comparar el nivel de entrenamiento de los robots.
- ¡Inspeccioná la biblioteca! A lo mejor encontrás algo que te ayuda a construir tu programa.

Acá podés ver algunos tableros iniciales y cómo deben quedar al finalizar la ejecución del programa.



2. ¿Qué diferencia encontrás entre los bloques `si [ ] / si no` y `si [ ] / si no, si [ ] / si no`?

Actividad 4
Agregamos pelotas, ¿o no?

DE A DOS

OBJETIVOS

- Presentar el operador booleano de negación.
- Usar alternativas condicionales simples.

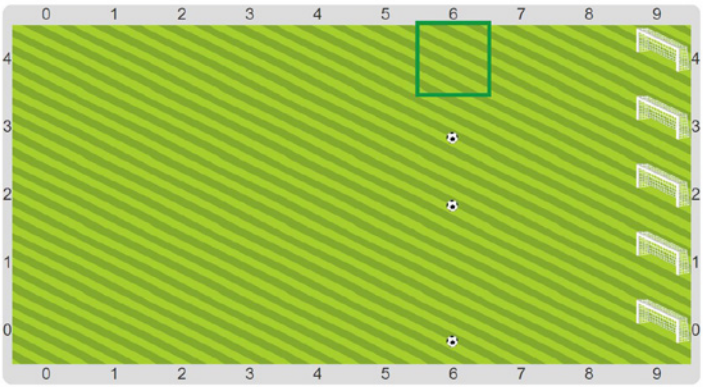
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

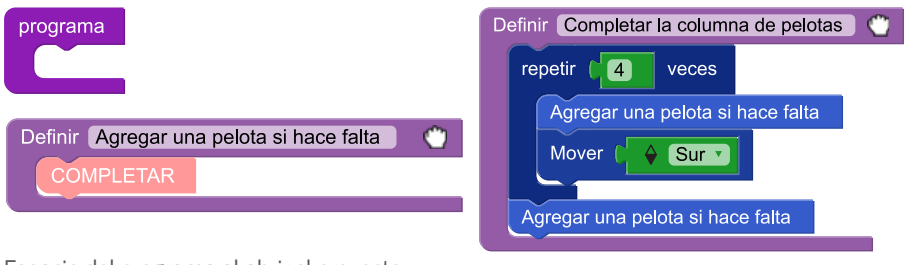
En esta actividad, los estudiantes utilizarán el operador `no [ ]`, que invierte un valor de verdad. Por ejemplo, si una condición `c` es verdadera, `no [c]` es falsa. Del mismo modo, si `c` es falsa, `no [c]` es verdadera. Además, usarán una alternativa condicional que no define ninguna acción en caso de que no se satisfaga la condición evaluada.

Comenzamos la actividad repartiendo la ficha y les pedimos a los estudiantes que abran el proyecto "Agregamos pelotas, ¿o no?". Se encontrarán con un tablero de 10 × 5 en el que hay un arco de fútbol en todas las celdas de la columna 9 y hay pelotas en algunas de la columna 6. El objetivo es que en todas las celdas de la columna 6 queden pelotas, de forma tal que el robot el Beto pueda entrenar.[1] Les indicamos que presionen varias veces el botón *Ejecutar* para que observen que hay distintos tableros iniciales. A continuación, les pedimos que resuelvan la actividad.



Uno de los tableros iniciales

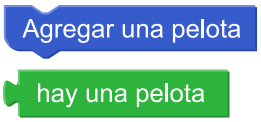
En el espacio del programa, encontrarán dos procedimientos: `Completar la columna de pelotas` y `Agregar una pelota si hace falta`. El primero, en el que se recorre cada una de las filas, ya está resuelto. Falta completar el segundo, que debe agregar una pelota solo si en la celda bajo el cabezal no hay ninguna.



Espacio del programa al abrir el proyecto

[1] Ver la actividad "El entrenamiento del robot goleador" en el capítulo 3.

En *Biblioteca* > *Procedimientos* se encuentra disponible el procedimiento `Agregar una pelota` y en *Biblioteca* > *Funciones*, la función `hay una pelota`, ambos de gran utilidad para resolver la actividad.

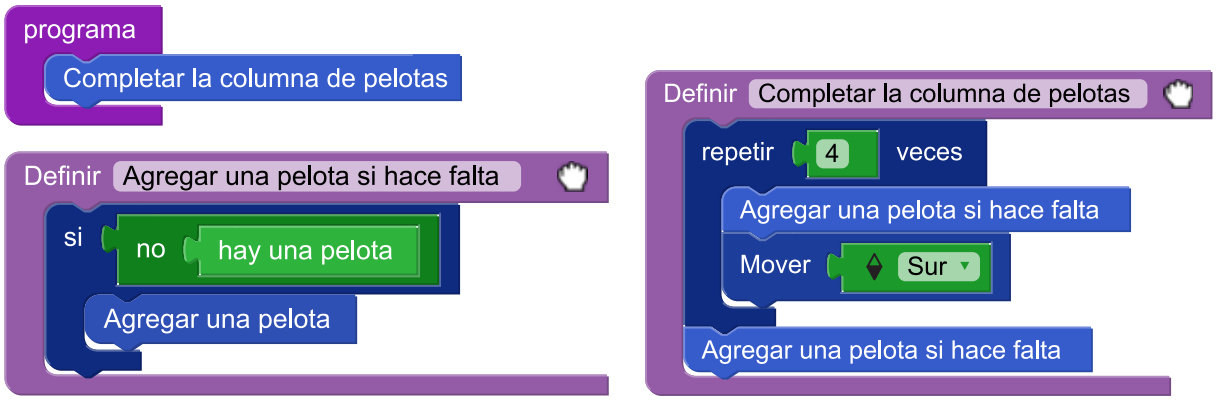


Procedimiento `Agregar una pelota` y función `hay una pelota` de la Biblioteca

En las actividades anteriores, frente a una determinada condición, se definía qué acciones realizar, tanto en el caso de que la condición se diera como si no. En esta oportunidad, hay que agregar una pelota si no hay ninguna; de lo contrario, no hay que hacer nada. En la categoría *Comandos* > *Alternativas* se encuentra el bloque `si [ ]`, que sirve específicamente para estas situaciones. Además, para poder verificar si hay que agregar una pelota, hay que combinar la función `hay una pelota` con el operador booleano `no [ ]`, de la categoría *Expresiones* > *Operadores*. La expresión `no [hay una pelota]` será verdadera si y solo si `hay una pelota` es falsa.

Alternativa condicional y operador de negación

A continuación, se muestra una versión completa del programa.



Programa que resuelve el desafío

CIERRE

Para concluir, explicamos el operador de negación y sus particularidades. Una diferencia entre este operador y los utilizados en actividades anteriores es que toma como argumento un valor de verdad... Es muy útil poder decir que no de vez en cuando.

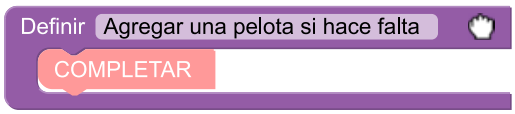
NOMBRE Y APELLIDO:

CURSO: FECHA:

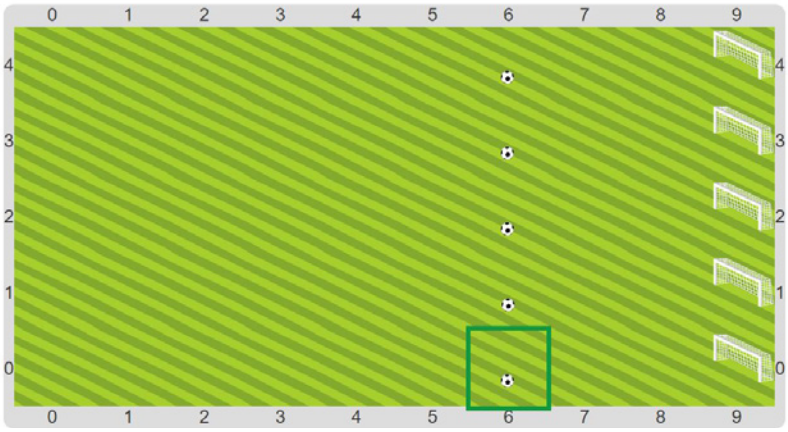
AGREGAMOS PELOTAS, ¿O NO?

¡Hay que preparar el campo de juego para que el Beto pueda entrenar!

1. Abrí el proyecto "Agregamos pelotas, ¿o no?" y completá el programa para que al Beto le queden pelotas por patear a todos los arcos. Ya viene resuelto el procedimiento `Completar la columna de pelotas`. Vos tenés que ocuparte de completar el cuerpo principal del programa y el procedimiento `Agregar una pelota si hace falta`, que lo único que tiene que hacer es depositar una pelota en la celda bajo el cabezal solo si allí no hay una previamente. ¡Cuidado! Si agregás una pelota donde ya había una… ¡BOOM!



Tenés que conseguir un tablero final como este:



Hay muchos tableros iniciales. ¡Tu programa tiene que funcionar en todos!

2. Copiá acá abajo cómo completaste `Agregar una pelota si hace falta`:

BIBLIOTECA

En la biblioteca vas a encontrar disponibles el procedimiento `Agregar una pelota` y la función `hay una pelota`. ¡Usalos para resolver el desafío!

Agregar una pelota

hay una pelota

Secuencia Didáctica 3

FUNCIONES

Cuando queremos realizar alguna acción compleja, los procedimientos nos permiten definir un nuevo comando y darle un nombre que represente esa tarea. Al trabajar con datos, también podemos necesitar construir algunos más complejos que provengan de, por ejemplo, combinar herramientas como los operadores y los sensores. Sin embargo, con lo visto hasta ahora, no tenemos forma de nombrarlos. Las **funciones** sirven justamente para esto. Permiten calcular valores y nombrarlos de un modo que describa lo que se está computando. Esta nueva herramienta completa el conjunto de herramientas básicas necesarias para programar. En esta secuencia presentamos las funciones y sus usos más frecuentes. Además, en los ejercicios que la componen, se presentan los operadores booleanos de conjunción y disyunción.

OBJETIVOS

- Presentar el uso de funciones en programación.
- Utilizar los operadores booleanos de conjunción y disyunción.

Actividad 1
Tomateras y luces

DE A DOS

OBJETIVOS

- Definir funciones.
- Presentar el operador booleano de conjunción.

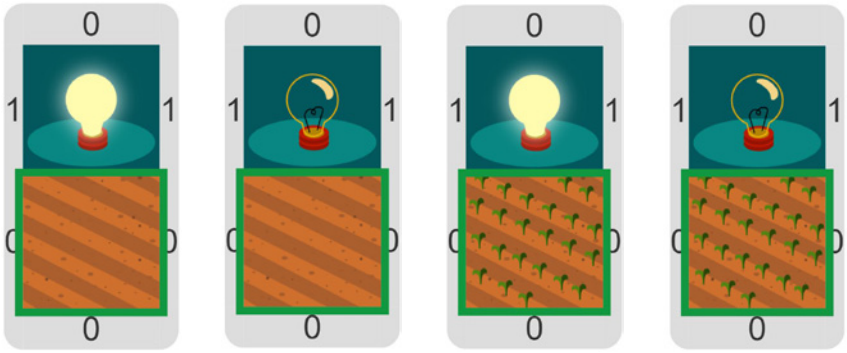
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

Esta actividad está dividida en dos partes. En la primera, los estudiantes se aproximarán a la creación de funciones. Comenzarán comparando dos procedimientos, uno que invoca a una función, y otro que no lo hace. Podrán observar, entonces, cómo el uso de funciones favorece la legibilidad de los programas. Como resultado de lo observado, definirán una función que mejora cualitativamente el programa analizado. En la segunda parte, usarán por primera vez el operador booleano de conjunción `[ ] y también [ ]`, que toma dos argumentos booleanos y da como resultado verdadero si y solamente si ambos valores también lo son.

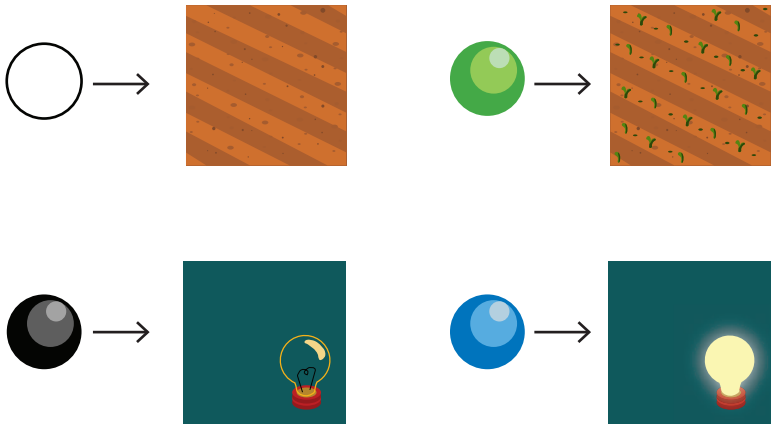
Comenzamos la actividad repartiendo la ficha y pidiéndoles a los estudiantes que abran el proyecto "La luz está prendida vs. hay bolitas: mi elección es clara". Encontrarán un tablero de 1 × 2, con el cabezal sobre la celda (0,0). Hay cuatro tableros iniciales posibles, que surgen de las siguientes combinaciones: en (0,0) puede haber una parcela de tierra sin plantar o una con tallos de plantas de tomate; y en (0,1), una lamparita prendida o una apagada. Para que puedan observarlos, les indicamos que desplieguen el menú que se encuentra arriba a la izquierda en el espacio del tablero y seleccionen cada uno.



Los cuatro tableros iniciales del proyecto

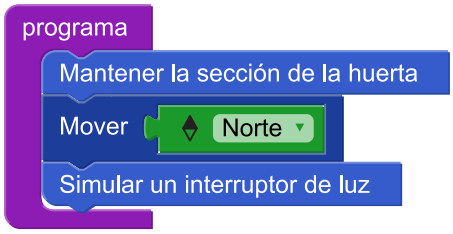
Menú para observar los tableros iniciales

A continuación, les sugerimos que activen y desactiven la vestimenta de cada tablero inicial para descubrir cómo es la asociación entre bolitas e imágenes. Una celda sin bolitas representa una parcela vacía; una bolita verde, una parcela con tallos de tomateras; una negra, una lamparita apagada; y una azul, una encendida.



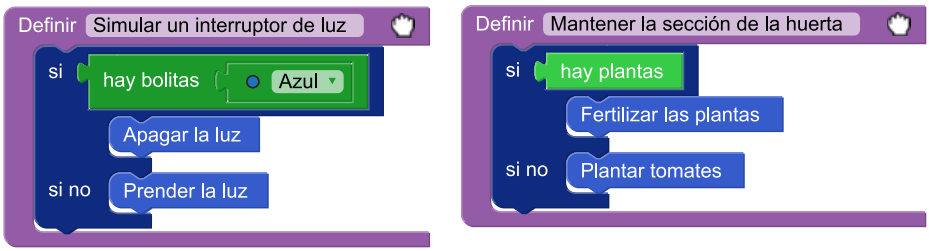
Vestimenta de la actividad

En el espacio del programa encontrarán el cuerpo principal del programa y los procedimentos que invoca, con los que deben familiarizarse. Si prestan atención, notarán que el propósito es, en primer lugar, plantar una tomatera si la parcela de tierra está vacía o, si ya hay una, fertilizarla; y a continuación cambiar el estado de la lamparita: encenderla si está apagada o apagarla si está encendida.



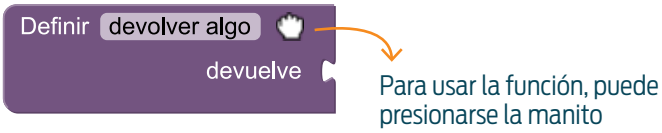
Cuerpo principal del programa

Les pedimos a los estudiantes que comparen los procedimientos `Simular un interruptor de luz` y `Mantener la sección de la huerta`. La diferencia principal radica en la condición que se usa en la alternativa condicional. Mientras que, en el primer caso, se utiliza un sensor y la condición queda expresada en términos de bolitas, en el segundo se usa una función cuyo nombre se inscribe dentro del dominio del problema que se está abordando. En caso de que no lo noten, podemos preguntar: "¿No les resulta raro decir 'si hay bolitas azules, apagar la luz'? ¿No sería más natural decirlo de otra forma? ¿Cómo?".



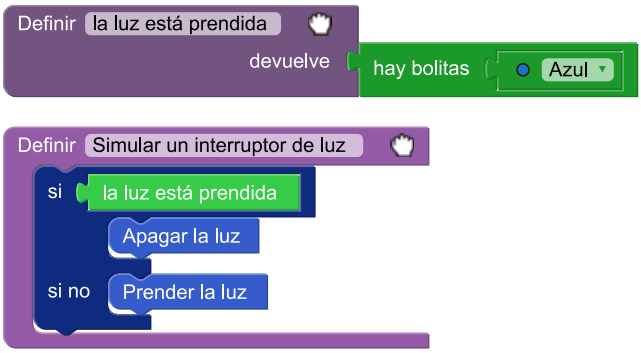
Procedimientos disímiles

Una vez que todos hayan identificado la diferencia, les indicamos que modifiquen el programa para que el procedimiento `Simular un interruptor de luz` también utilice un lenguaje inherente al dominio del problema. Al explorar el entorno, en la categoría *Definiciones* > *Definición de funciones*, encontrarán un bloque que les permite definir sus propias funciones. Al definir una nueva función, hay que especificar el valor que arrojará al ser invocada. Además, es importante elegir un nombre descriptivo que permita interpretar de forma sencilla su significado. Una vez definida la función, queda disponible para su uso en *Mis operaciones* > *Mis funciones*.



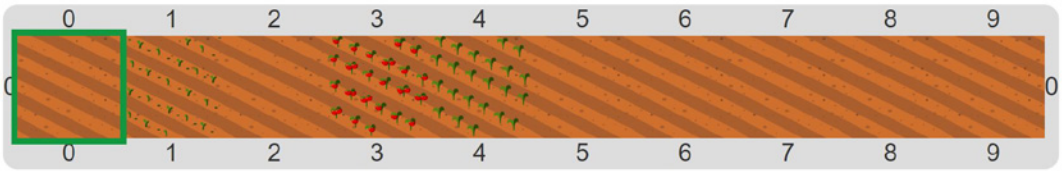
Bloque para definir una función

Un nombre adecuado para la función puede ser `la luz está prendida`. La definición de la función `hay plantas`, sumada a la asociación entre bolitas e imágenes de la vestimenta, debería guiarlos para llegar a una solución: la luz está encendida si y solamente si hay una bolita azul en la celda. Luego, para terminar de resolver la primera parte de la actividad, usamos la función en el cuerpo del procedimiento `Simular un interruptor de luz`.



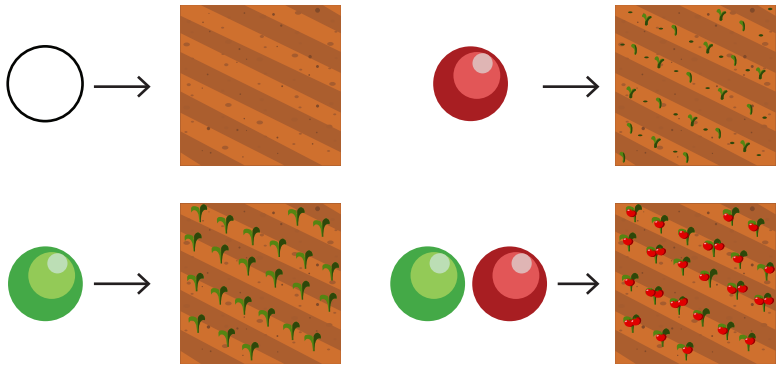
Definición y uso de la función `la luz está prendida`

Para la segunda parte, les pedimos a los estudiantes que carguen el proyecto “Y también...”. Al hacerlo, encontrarán un tablero de 10 × 1. En todas las celdas hay una parcela de tierra; en algunas hay brotes verdes; en otras, tallos de tomateras ; y, en algunas otras, plantas con tomates. El objetivo es que el programa recolecte todos los tomates que están listos para ser cosechados.

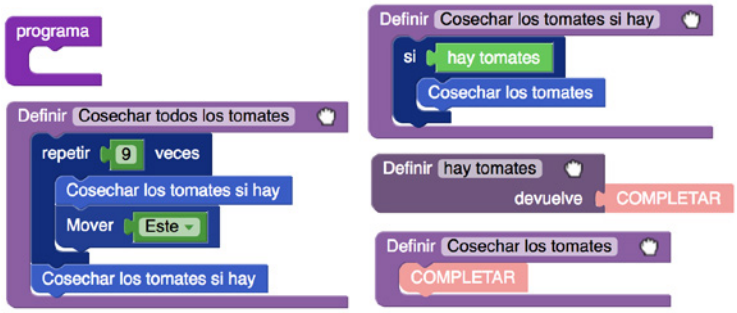


Un tablero inicial

Nuevamente, les pedimos que presionen reiteradas veces el botón *Ejecutar*, para observar que hay una diversidad de tableros iniciales. Al activar y desactivar la vestimenta, verán que en este proyecto *(i)* una parcela de tierra está representada por una celda vacía; *(ii)* una parcela con brotes, por una bolita roja; *(iii)* una con tallos altos, por una verde, y *(iv)* las plantas con tomates, por una bolita roja y una verde.

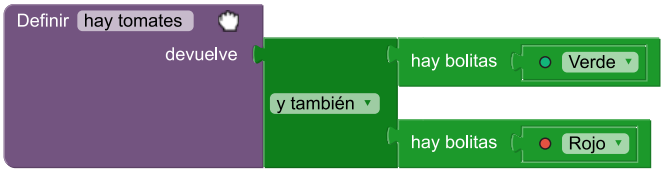


Parte del programa ya está construido en el proyecto. Se encuentran completos los procedimientos `Cosechar todos los tomates`, que recorre cada parcela de tierra, y `Cosechar los tomates si hay`, que cosecha tomates en caso de que haya una planta con frutos maduros.



Espacio del programa al abrir el proyecto

¿Qué sucede con la función `hay tomates`? Para determinar si hay tomates, debemos observar si en la celda que estamos inspeccionando hay una bolita roja y una verde. Para responder afirmativamente, las dos condiciones deben ser verdaderas. En *Expresiones* > *Operadores*, se encuentra el operador booleano `[ ] y también [ ]` que sirve para este propósito. El resultado será verdadero si y solamente si, al evaluarlo, los dos argumentos lo son. Combinando este bloque con el sensor `hay bolitas [ ]` podemos completar la función.

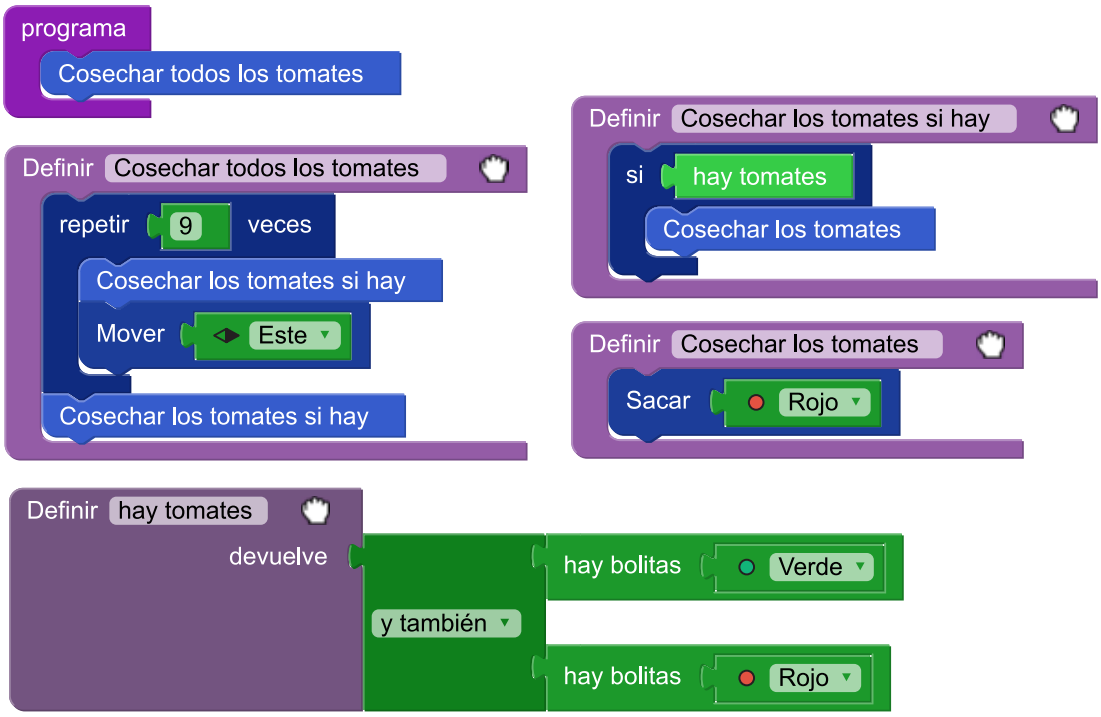


Función `hay tomates`

Para completar el procedimiento `Cosechar los tomates`, hay que tener presente que una parcela con plantas con tomates se representa con una bolita verde y una roja, y una parcela solo con los tallos se representa con una verde. Por lo tanto, hay que quitar la roja para retirar los frutos.

Procedimiento `Cosechar los tomates`

Finalmente, el cuerpo principal del programa consiste en una invocación a `Cosechar todos los tomates`. Una versión completa del programa se observa a continuación:



Programa que resuelve el desafío

CIERRE

Como cierre, repasamos con los estudiantes que en esta actividad definieron funciones por primera vez. Al igual que con los procedimientos, hay que escoger nombres adecuados y descriptivos de lo que representan. Sin embargo, mientras que los procedimientos encapsulan acciones y, por lo tanto, sus nombres comienzan con un verbo, el nombre de una función debe ser descriptivo del valor que arroja como resultado. Por último, reflexionamos sobre el operador lógico de conjunción, que permite expresar condiciones booleanas complejas a partir de otras más simples.

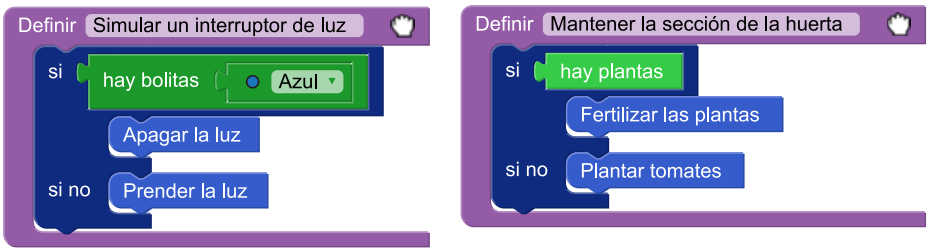
NOMBRE Y APELLIDO:

CURSO: FECHA:

TOMATERAS Y LUCES

¿Qué tienen que ver las bombitas de luz con las plantas de tomate? Es una excelente pregunta, ¿no?

1. Abrí el proyecto "La luz está prendida vs. hay bolitas: mi elección es clara" y compará los procedimientos `Simular un interruptor de luz` y `Mantener la sección de la huerta`. ¿En qué se diferencian?



Ahora tenés que conseguir que `Simular interruptor de luz` sea tan fácil de leer y entender como `Mantener la sección de la huerta`. ¡Inspeccioná el entorno!

2. Cargá el proyecto "Y también…". Tu trabajo es completar el programa para cosechar los tomates. Tené en cuenta que hay varios tableros iniciales.

ASÍ LOS PODEMOS VER

Hay varios tableros iniciales. Para verlos, usá el menú que está arriba a la izquierda del espacio del tablero.

¿Cómo te quedó la función `hay tomates`? Copiala acá abajo.

ALGUNAS PISTAS

- Activá y desactivá la vestimenta para ver cómo están representados los distintos elementos.
- Explorá el entorno en busca de algún bloque que te permita expresar condiciones compuestas.

Actividad 2
Laberinto con queso

DE A DOS

OBJETIVO

- Integrar el uso de funciones, procedimientos, sensores y operadores.

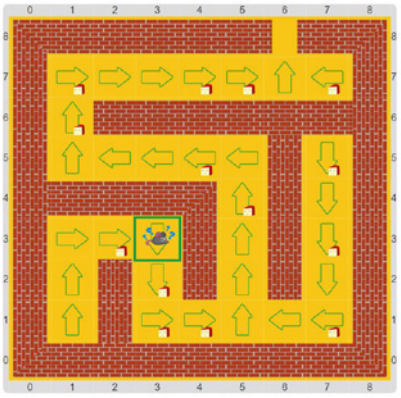
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

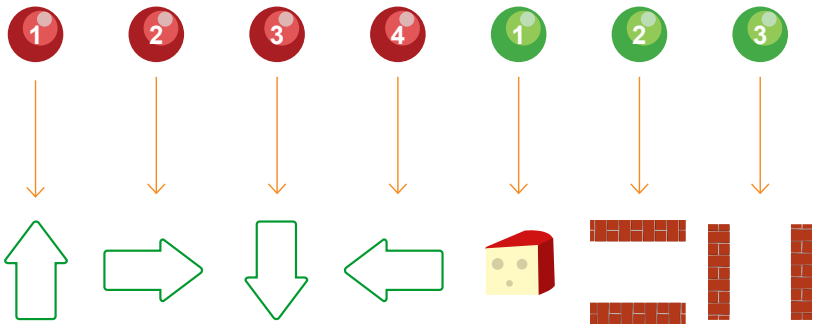
En esta actividad, los estudiantes integrarán distintas herramientas estudiadas, incluyendo el uso de sensores, operadores y la definición de procedimientos y funciones.

Comenzamos repartiendo la ficha a los estudiantes y les indicamos que abran el proyecto "Laberinto con queso". Se trata de un desafío muy similar al de la actividad "Laberinto". También en este caso hay un ratón que se encuentra a 20 celdas de la salida, y en cada celda hay una flecha que indica cómo desplazar al roedor para que pueda escapar. La diferencia es que en algunas celdas hay también un pedazo de queso. Además de salir del laberinto, en esta oportunidad el roedor tiene que comer todo el queso que se cruce en su camino. Nuevamente, hay varios tableros iniciales, que pueden observarse accediendo al menú ubicado en la parte superior izquierda del espacio del tablero.



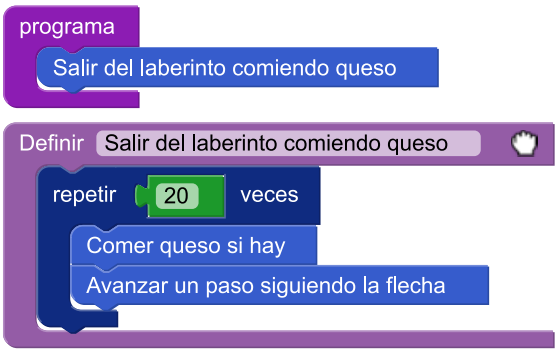
Uno de los tableros iniciales

A lo largo de la actividad, es importante tener presente la asociación entre bolitas y elementos que hace la vestimenta. Las flechas en dirección norte, este, sur y oeste se representan con 1, 2, 3 y 4 bolitas rojas respectivamente; el queso, con una verde; y las dos posibles salidas del laberinto, horizontal y vertical, con dos y tres bolitas verdes respectivamente.



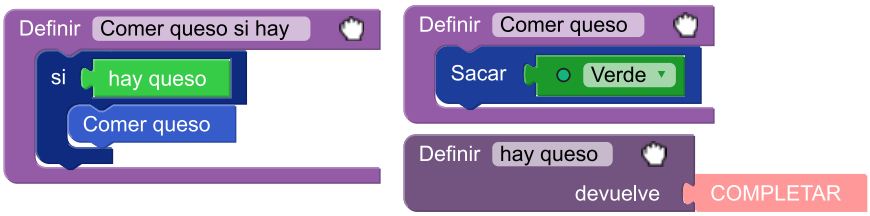
Vestimenta

Para resolver el desafío, los estudiantes deben completar el programa, que está parcialmente construido. Al observar el cuerpo principal del programa y el procedimiento `Salir del laberinto comiendo queso` –ambos están construidos–, queda clara la estrategia del programa: repite en 20 oportunidades la intención de comer un pedazo de queso y avanzar un paso hacia la salida.



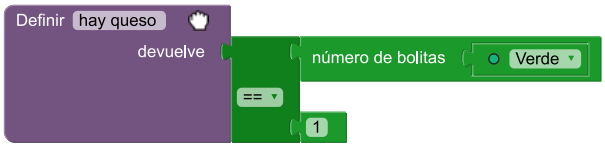
Cuerpo principal y procedimiento `Salir del laberinto comiendo queso`

Para completar el procedimiento `Comer queso si hay` –que al cargar el proyecto está vacío–, en primer lugar se debe chequear si hay un pedazo de queso usando la función `hay queso` –que está vacía– y, en caso de que así fuera, invocar `Comer queso` –que ya está construido–.



Procedimiento `Comer queso si hay`

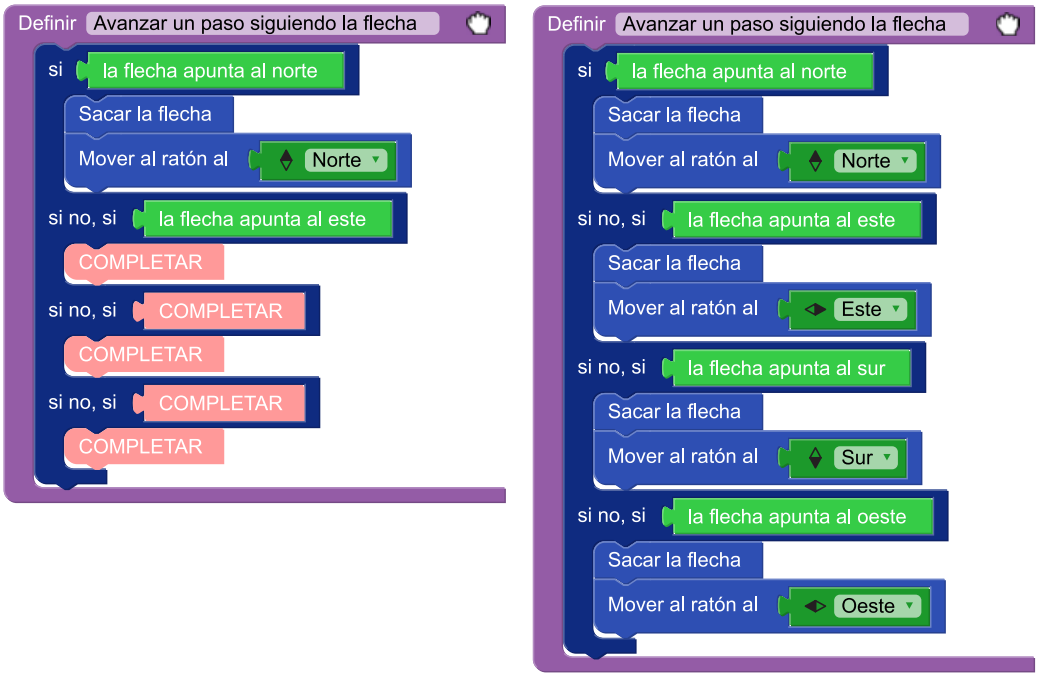
Teniendo en cuenta que el queso está representado por una bolita verde, la función `hay queso` tiene que dar como resultado verdadero si y solo si en la celda bajo el cabezal hay una bolita de dicho color. Un error que puede ocurrir es que algún estudiante construya esta función usando el sensor `hay bolitas [ ]` en lugar de chequear la cantidad. Como las salidas del laberinto se representan con dos o tres bolitas verdes (dependiendo de si es horizontal o vertical), si el cabezal se encontrase sobre algunas de ellas, `hay queso` daría verdadero, lo cual es incorrecto: en la salida nunca hay un trozo de queso.



Función `hay queso`

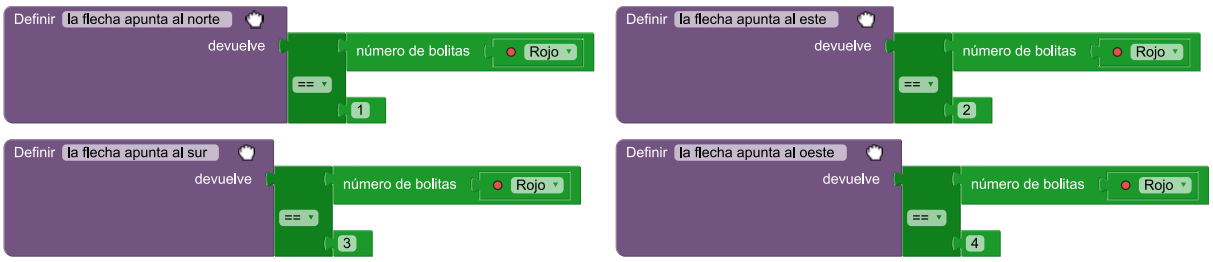
Versión errónea de `hay queso`

El procedimiento `Avanzar un paso siguiendo la flecha` está parcialmente resuelto. Para completarlo, se puede emular la parte ya provista (un movimiento cuando la flecha apunta al norte) para las restantes direcciones: este, sur y oeste.



Procedimiento `Avanzar un paso siguiendo la flecha`, como viene en el proyecto y una vez completado

Para terminar la actividad, hay que completar las funciones `la flecha apunta al este`, `la flecha apunta al sur` y `la flecha apunta al oeste`. Esto puede realizarse siguiendo el esquema de `la flecha apunta al norte`, que ya está dado.



Funciones para determinar a dónde apunta la flecha

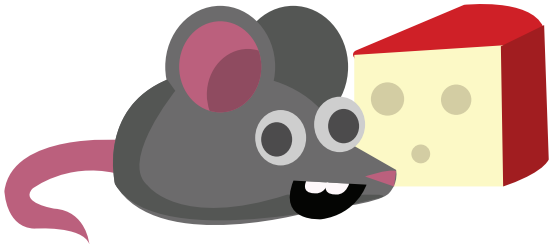
CIERRE

Para cerrar la actividad, señalamos que este es un programa complejo, pero que, gracias a la capacidad de definir procedimientos y funciones que expresan el vocabulario del problema y la estrategia de solución, se puede entender y completar con relativa facilidad, al menos con más facilidad que si todo estuviese escrito en términos de bolitas y movimientos del cabezal. Como se puede observar, utilizar procedimientos y funciones de esta forma es imprescindible para conseguir programas que no sean difíciles de entender.

NOMBRE Y APELLIDO:

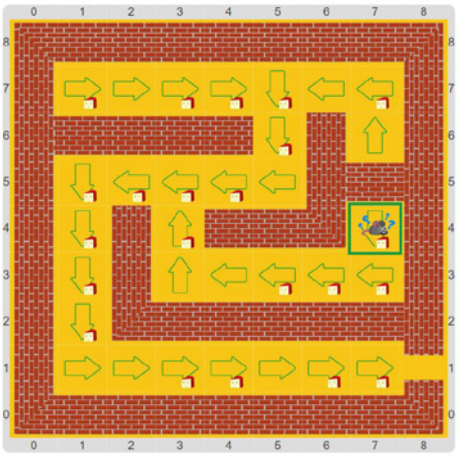
CURSO: FECHA:

LABERINTO CON QUESO

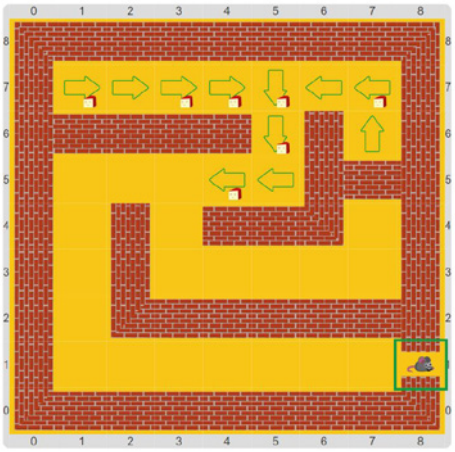


Una vez más el ratón está en el laberinto, a 20 pasos de distancia de la salida. Pero, esta vez, hay una diferencia: en algunos lugares del laberinto hay pedacitos de queso desperdigados.

1. Abrí el proyecto "Laberinto con queso" y completá el programa para que el ratón pueda salir del laberinto. Esta vez, además, tiene que comer todo el queso que se cruce en su camino. Acá te mostramos un tablero inicial y su correspondiente tablero final:

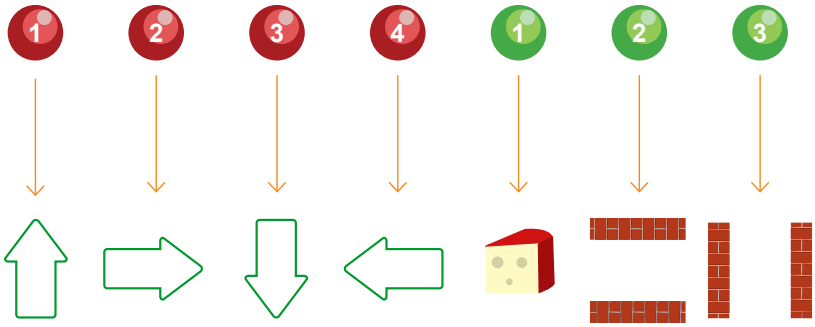


Tablero inicial



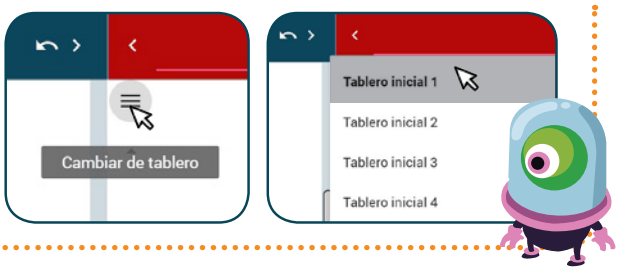
Tablero final

Para resolver la actividad, hay que tener presente cómo la vestimenta representa los elementos del problema.



PARA TENER EN CUENTA

Hay varios tableros iniciales. Para verlos, usá el menú que está arriba a la izquierda del espacio del tablero.

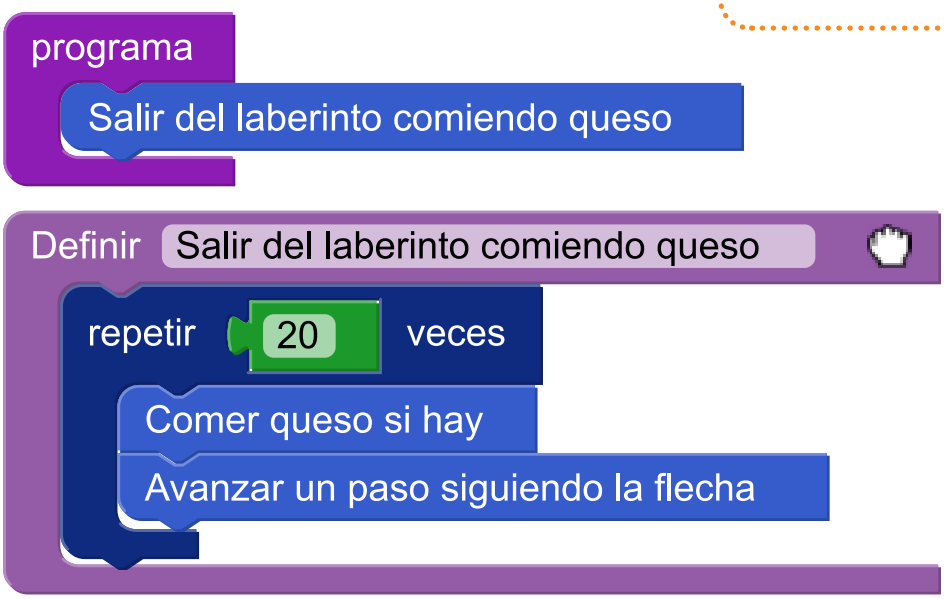


NOMBRE Y APELLIDO:

CURSO: FECHA:

Hay partes del programa que ya están resueltas.
Vos solamente tenés que completarlo.

¡No te olvides de chequear qué hay en la biblioteca!



¡EMPEZÁ POR EL PRINCIPIO!

Comenzá mirando el cuerpo principal del programa y el procedimiento `Salir del laberinto comiendo queso`. Ahí está la clave para entender la estrategia de solución. Luego, completá los procedimientos y funciones a medida que los necesites.

Lucho recargado

DE A DOS

OBJETIVO

- Construir un programa que integre todos los temas estudiados.

MATERIALES

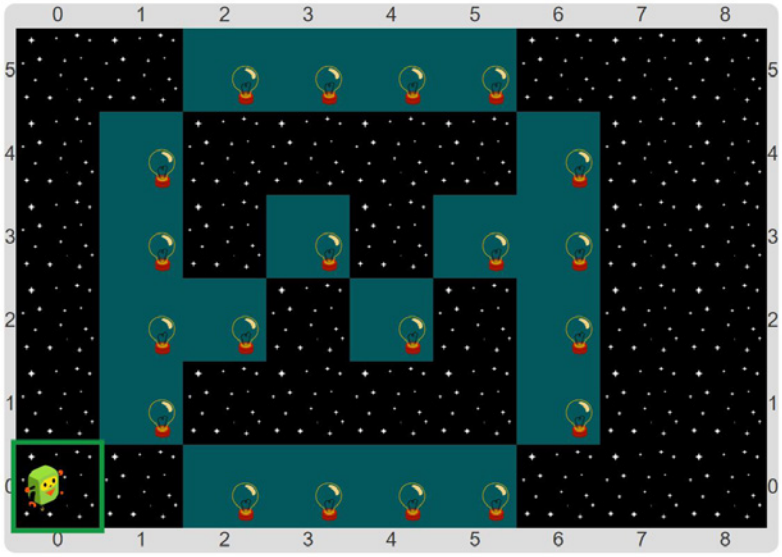
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes utilizarán todas las herramientas estudiadas hasta el momento: comandos básicos, procedimientos, funciones, operadores, repeticiones, alternativas condicionales y sensores.

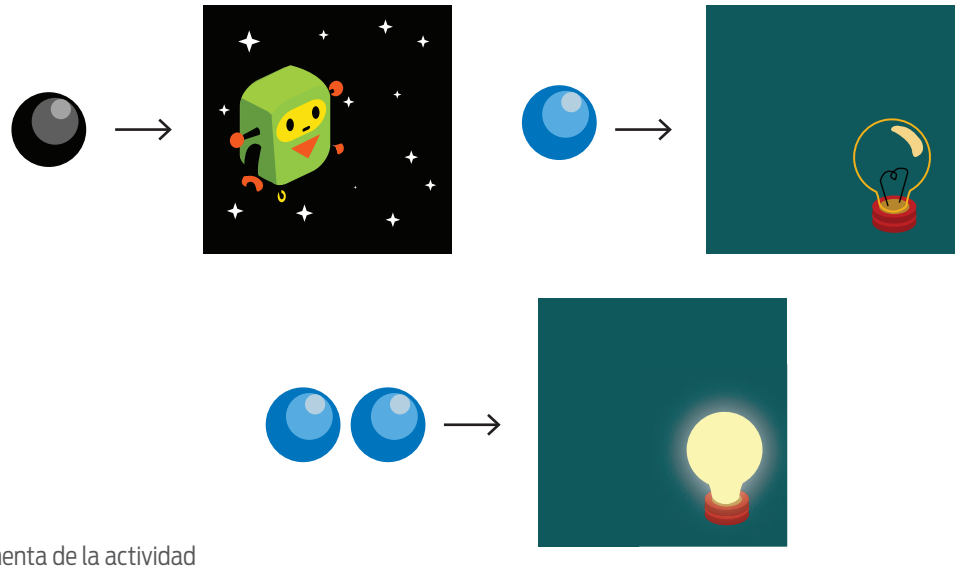
Comenzamos la actividad repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto "Lucho recargado". Al hacerlo, encontrarán un tablero de 9 × 6 en el que el robot Lucho se encuentra ubicado en la posición (0,0) y en algunas celdas hay una lamparita apagada. El objetivo es que el androide recorra el tablero y encienda todos los focos que se vaya cruzando. Presionando varias veces el botón *Ejecutar* podrán observar que hay muchos tableros iniciales diferentes, aunque todos de la misma dimensión.

Lucho



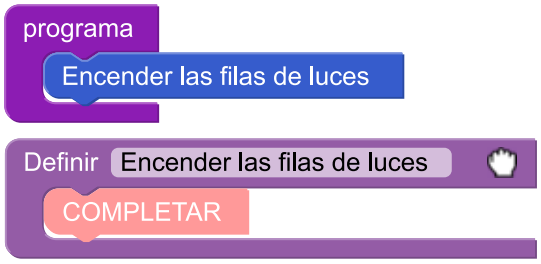
Un tablero inicial de "Lucho recargado"

Para resolver la actividad, es necesario tener en cuenta cómo la vestimenta representa los distintos elementos: el robot Lucho se representa con una bolita negra; una lamparita apagada, con una azul, y una encendida, con dos azules. Para observar esta asociación, se puede activar y desactivar la vestimenta.



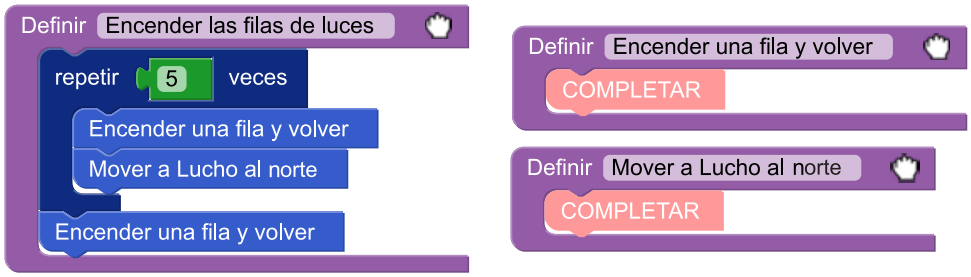
Vestimenta de la actividad

En el espacio de programa están todos los procedimientos y funciones que hay que completar para resolver el desafío. La forma más conveniente de abordar el problema es familiarizarse con los procedimientos propuestos para delinear una estrategia que permita alcanzar progresivamente una solución. De ese modo, puede observarse que el cuerpo principal del programa podría consistir solo en invocar el procedimiento `Encender las filas de luces`, aun cuando este último no esté resuelto.



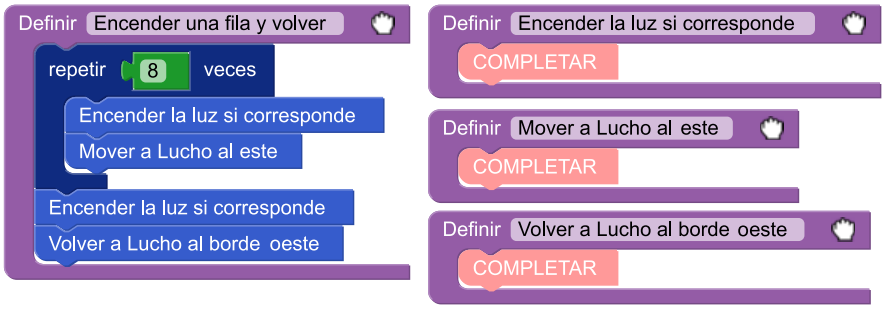
Cuerpo principal del programa.

Para encender las lamparitas de todas las filas, hay que pasar por cada una. Por lo tanto, para completar `Encender las filas de luces` se puede combinar el comando `repetir [ ] veces` con los procedimientos `Encender una fila y volver` y `Mover a Lucho al norte`. Es importante notar que mientras que el tablero tiene 6 filas, el argumento de `repetir [ ] veces` es `5`: hay que tratar de forma separada la última fila. Esto se debe a que es un caso de borde: de incluirse en la repetición, Lucho caería fuera del tablero.



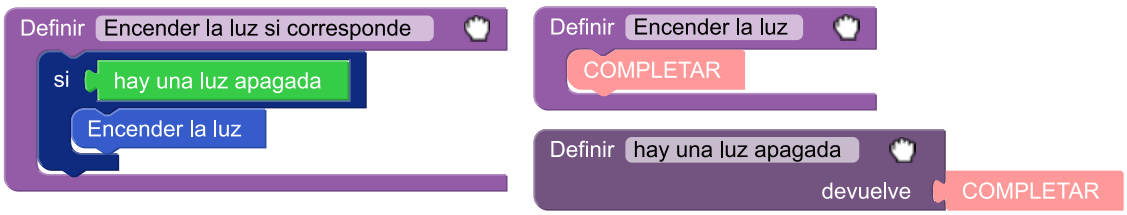
Procedimiento `Encender las filas de luces`

Para completar `Encender una fila y volver`, hay que recorrer las nueve celdas de una fila, encender cada lamparita con la que Lucho se topa y volver a posicionar al robot en la primera columna. Una forma de resolverlo consiste, en primer lugar, en combinar `repetir [ ] veces` con los procedimientos `Encender la luz si corresponde` y `Mover a Lucho al este`, y a continuación tratar la última celda por separado y luego invocar `Volver a Lucho al borde oeste`.



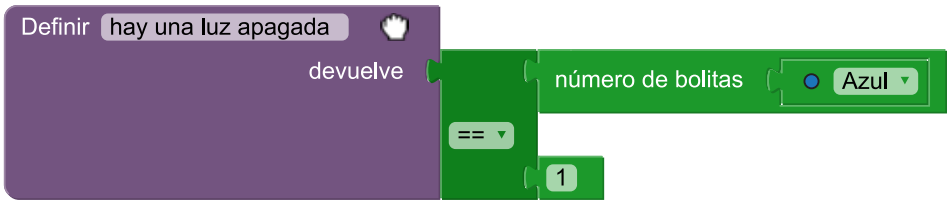
Procedimiento `Encender una fila y volver`

En `Encender luz si corresponde` hay que chequear si hay una luz apagada y, de ser así, encenderla.



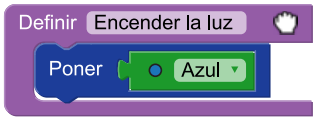
Procedimiento `Encender la luz si corresponde`

Para completar la definición de la función `hay una luz apagada` hay que tener presente la representación que define la vestimenta: un lamparita apagada se representa con una bolita azul.



Función `hay una luz apagada`

Al programar `Encender la luz`, nuevamente debe tenerse presente la vestimenta. Como una lamparita apagada se representa con una bolita azul y una encendida con dos azules, solo hay que agregar una azul.

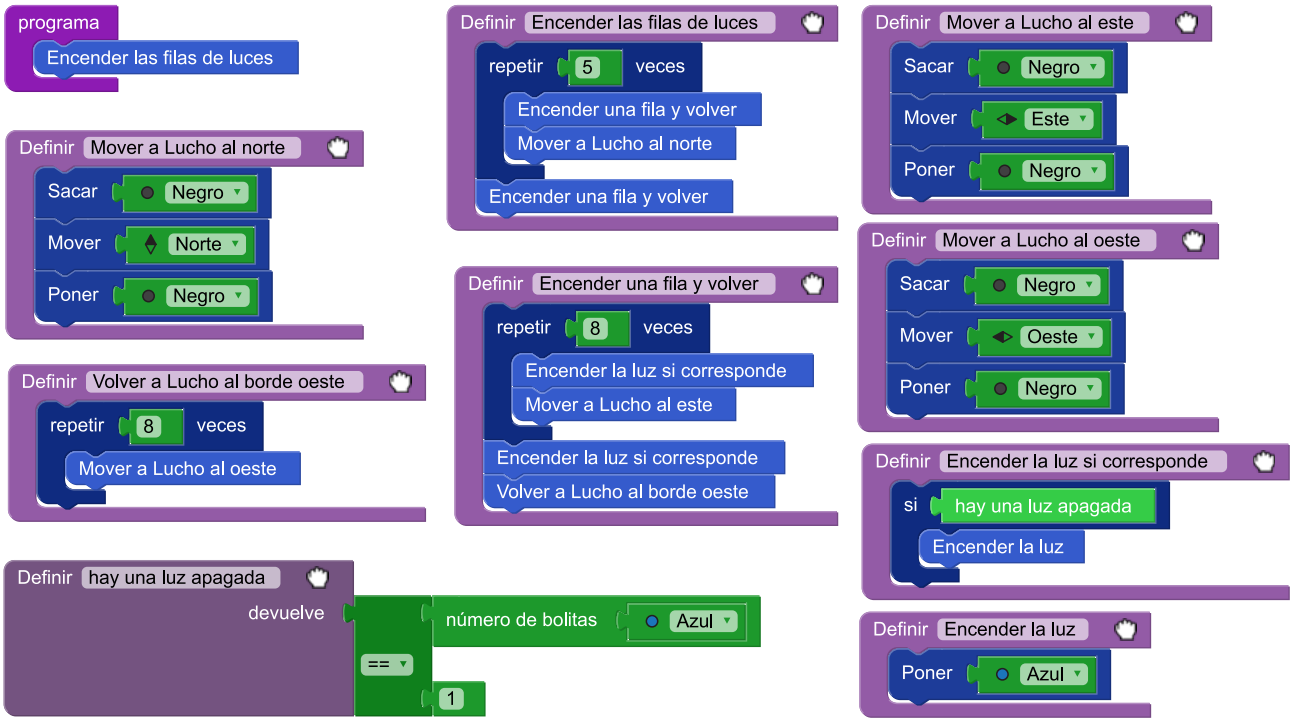


Procedimiento `Encender la luz`

Falta completar los procedimientos para mover al robot. Como Lucho está representado por una bolita negra, en `Mover a Lucho al este`, `Mover a Lucho al norte` y `Mover a Lucho al oeste` hay que sacar la bolita negra de la celda bajo el cabezal, mover el cabezal una posición en la dirección adecuada y depositar allí una negra. Por último, como `Volver a Lucho al borde oeste` se invoca solo cuando el androide está en el extremo este –8 celdas a la derecha–, se resuelve combinado `repetir [ ] veces` con `Mover a Lucho al oeste`.

Procedimientos para desplazar a Lucho

Se muestra a continuación el programa completo:



Programa que resuelve el desafío

CIERRE

Para concluir, reflexionamos junto con los estudiantes sobre el hecho de que, en esta ocasión, los procedimientos y las funciones que expresan una estrategia y representan información estaban incluidos en el proyecto. Sin embargo, en general, una parte importante de la tarea al programar consiste en dividir el problema en partes y definir los procedimientos y funciones que expresan esas partes.

NOMBRE Y APELLIDO:

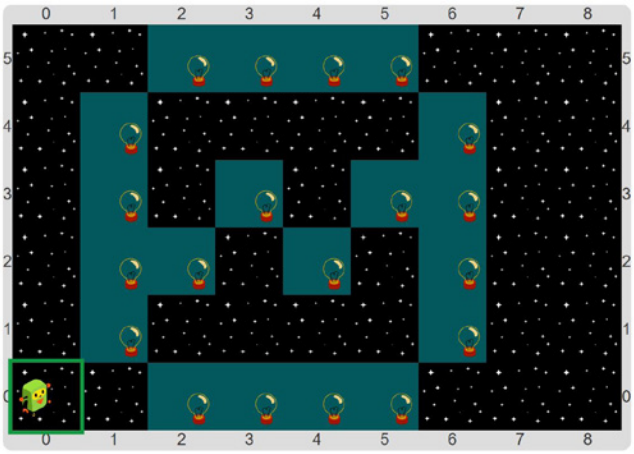
CURSO: FECHA:

LUCHO RECARGADO

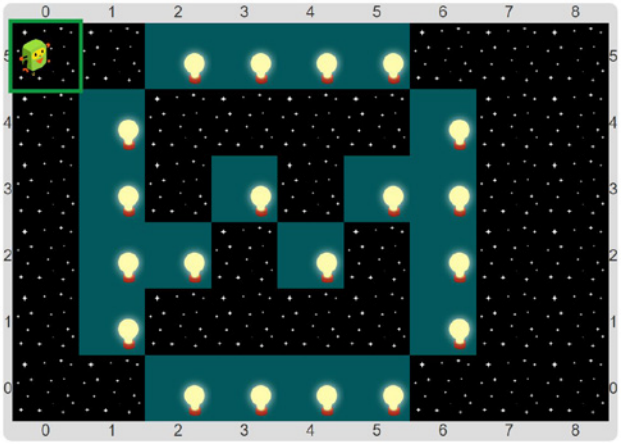
Al robot Lucho no le gusta la oscuridad.
¡Tenés que ayudarlo a encender las luces!

1. Abrí el proyecto "Lucho recargado" y completá el programa para ayudar a Lucho con su tarea.

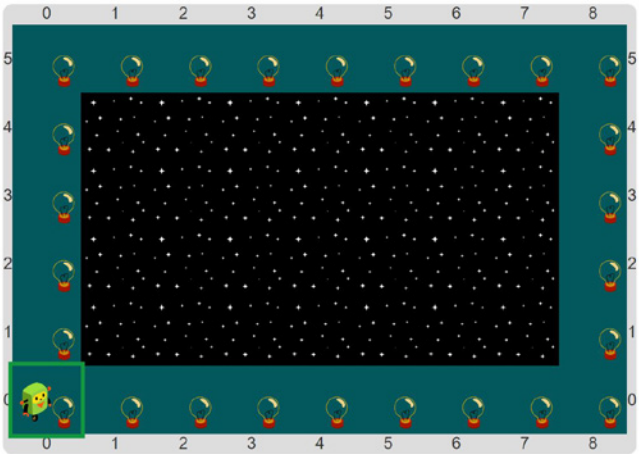
Hay muchos tableros iniciales distintos. Acá te mostramos dos y sus correspondientes tableros finales:



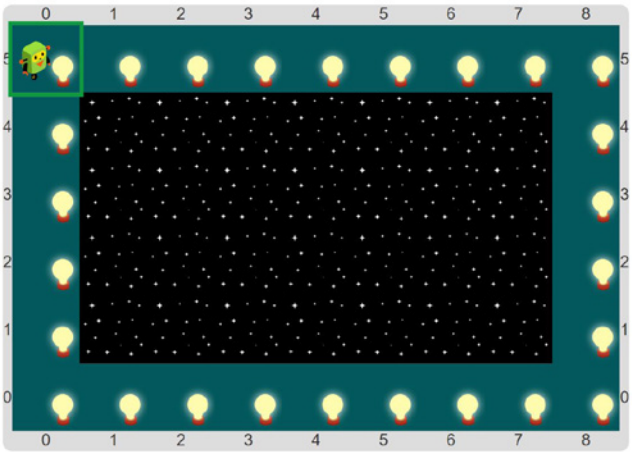
Tablero inicial



Tablero final



Tablero inicial

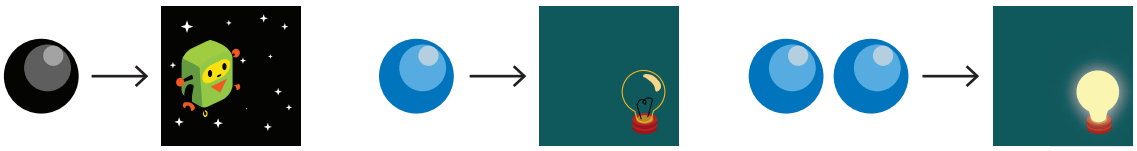


Tablero final

NOMBRE Y APELLIDO:

CURSO: FECHA:

¡TENÉ EN CUENTA LA VESTIMENTA!



2. Copiá acá abajo cómo armaste el procedimiento `encender una fila y volver`.

Cuando completes los procedimientos para expresar la estrategia, no te olvides de considerar los casos de borde en las repeticiones.

05

REPRESENTACIÓN DE LA INFORMACIÓN

Al utilizar una computadora, interactuamos con información de muy variado tipo: texto, números, imágenes, música, video, etc. Como usuarios, recibimos esa información a través de una pantalla o parlantes, por ejemplo. Sin embargo, internamente, la computadora la almacena utilizando solamente secuencias de bits, es decir, secuencias de ceros y unos.

Este capítulo se centra en cómo **representar** tres tipos diferentes de información: **números**, **imágenes** y **texto**. Comenzamos abordando la representación binaria de los números y analizando cómo se determina cuánto ocupa o "pesa" la información. A continuación, se trabaja sobre la representación de imágenes a color, imágenes en blanco y negro e imágenes comprimidas. Por último, incursionamos en la representación de texto y trabajamos sobre estrategias para enviar mensajes de forma secreta con un método de cifrado simétrico clásico.

Secuencia Didáctica 1

INFORMACIÓN EN BINARIO

Con frecuencia, trabajamos con números: cantidades, precios, distancias, etc. En la vida cotidiana lo hacemos utilizando diez símbolos: los dígitos del 0 al 9. Sin embargo, en una computadora, los números se representan con bits, que admiten solo dos valores: 0 y 1.

En esta secuencia didáctica abordaremos el sistema de numeración binario. Indagaremos cómo las computadoras almacenan y codifican información, cómo representan números, y estimaremos el espacio que ocupa la información en un medio físico.

OBJETIVOS

- Representar información usando un sistema binario.
- Comprender cómo representan los números las computadoras.
- Familiarizarse con las magnitudes más utilizadas para medir la información.

Actividad 1
Desempatamos el torneo

GRUPAL (4)

OBJETIVOS

- Evaluar distintas formas de codificar la misma información.
- Codificar información en forma binaria.

MATERIALES

Hoja de papel

Monedas

DESARROLLO

Esta actividad tiene por objetivo ensayar diferentes formas en que se puede representar la información, para luego poner el foco sobre una forma en particular: la binaria.

Les presentamos a los alumnos el siguiente problema: "Hay 4 equipos de fútbol que, al finalizar las fechas del torneo, están empatados en todas las categorías: puntos, goles, etc. Como no hay tiempo para desempatar jugando, se decidió hacerlo al azar, tirando monedas. Se dispone de una moneda y se necesita sortear al ganador. Los organizadores del torneo nos encargaron diseñar un mecanismo para decidir cuál es el equipo que se consagrará campeón mediante tiradas de monedas. Además, nos pidieron tener en cuenta que el mecanismo debe ser justo, es decir, todos los equipos tienen que tener la misma chance de ser elegidos ganadores. ¿Cómo podemos hacer?".

Equipo 1 Equipo 2 Equipo 3 Equipo 4

Los cuatro equipos empatados

Antes de abordar el problema con los 4 equipos, les sugerimos que primero piensen cómo resolverían el desempate si se tratase solamente de 2 equipos. Esto nos permite considerar que la moneda tiene dos estados posibles, cara y ceca, y que cada lado puede utilizarse para representar una de dos opciones. Por ejemplo, podemos decir que si sale cara gana el equipo 1 y si sale ceca gana el equipo 2 (o viceversa). Sin embargo, el problema de los 4 equipos de fútbol va a requerir representar más de dos opciones. ¿Cómo podemos hacerlo con una única moneda?

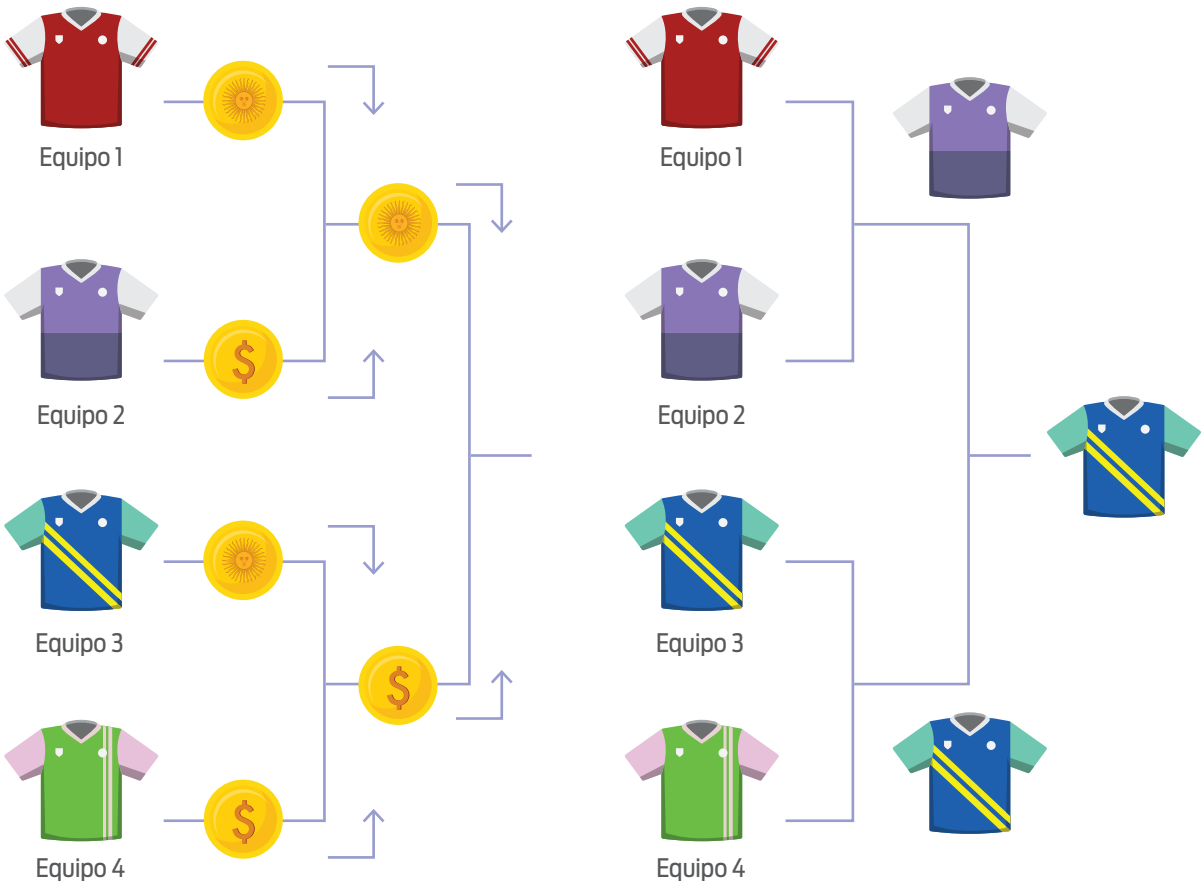
Dos equipos: con cara gana el equipo 1 y con ceca el equipo 2

Les damos un rato a los grupos para que puedan plantear, discutir y probar distintas estrategias. A los que hayan presentado soluciones que requieran tres tiradas de la moneda o más, los invitamos a pensar si se podría hacer en menos cantidad de tiradas. Cuando todos los grupos tengan una propuesta de desempate, les contarán al resto cómo es la estrategia que pensaron y cuántas tiradas requiere para determinar el equipo ganador.

Algunas de las propuestas que pueden surgir son las siguientes:

ELIMINACIÓN POR LLAVE

Dados los equipos 1, 2, 3 y 4, primero se sortea al ganador entre 1 y 2, luego al ganador entre 3 y 4, y finalmente al ganador entre los ganadores de los dos sorteos anteriores. En cada enfrentamiento, si sale cara resulta ganador el equipo que se encuentra más arriba en el cuadro y, si sale seca, el que está más abajo (o viceversa). Esto requiere tres lanzamientos de la moneda. Por ejemplo, si las tiradas salen ceca - cara - ceca, pasarán a la final el equipo 2 –por la primera tirada ceca– y el equipo 3 –por la tirada cara-, y además este último se coronará campeón –por la última ceca–.



Sistema de eliminación por llave y ejemplo de tirada ceca - cara - ceca

ELIMINACIÓN ORDENADA

Primero se sortea al ganador entre los equipos 1 y 2; luego al ganador de la tirada anterior con el 3; y finalmente al ganador de la tirada anterior con el 4. Esta estrategia no otorga igual probabilidad de ganar a todos los equipos, por lo que no es una opción correcta. El equipo 4 tiene un 50% de probabilidades de ganar, el equipo 3 tiene el 25% y los equipos 1 y 2 tienen un 12,5%. Una forma sencilla de verlo sin tener que hacer las cuentas consiste en notar que para consagrarse campeón, al equipo 4 le alcanza con ganar un partido, mientras que el equipo 3 debe ganar dos y, los equipos 1 y 2, tres. Por ejemplo, las tiradas cara - cara - ceca consagran al equipo 4, aun cuando participe en una sola tirada.

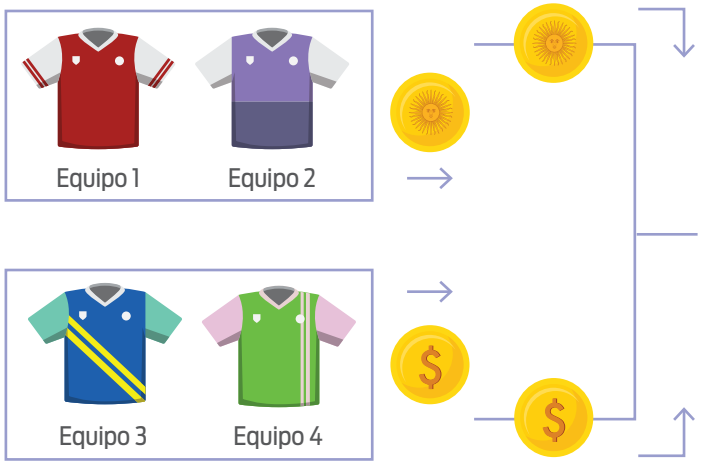
Sistema de eliminación ordenada

Ejemplo de tirada cara - cara - ceca

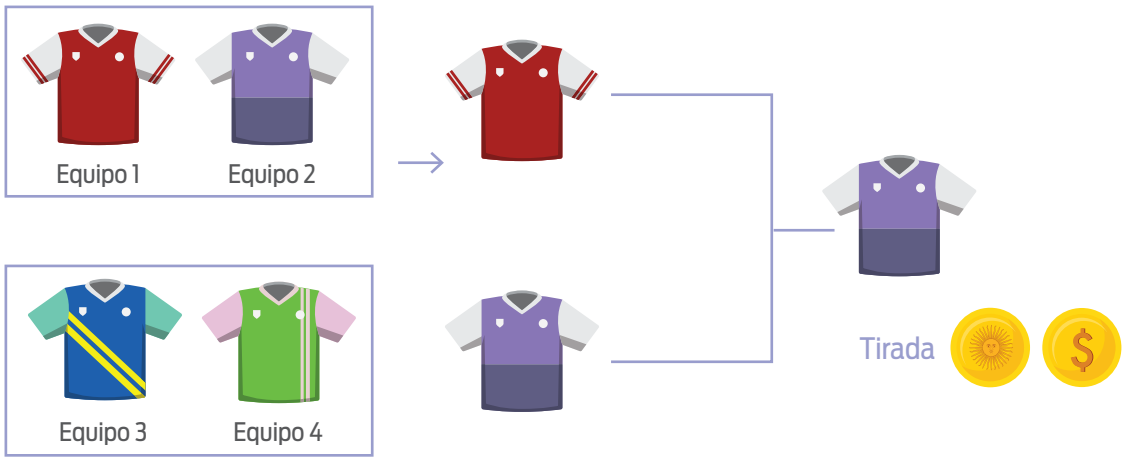
ELIMINACIÓN BINARIA

Se separa a los equipos en dos grupos. Por ejemplo, por un lado, agrupamos el 1 y el 2, y por otro lado, el 3 y el 4. En un primer sorteo, se elige cuál es el grupo ganador y con un segundo sorteo se elige el equipo ganador dentro de ese grupo. Este mecanismo requiere solo dos lanzamientos de la moneda.

De las 3 propuestas que enumeramos, solo la primera y la tercera son estrategias válidas, ya que dan a los 4 equipos igual chance de ganar. Mientras que para la primera se requieren 3 lanzamientos, para la tercera basta con 2 tiradas. Con este último método, estamos asociando cada equipo con un posible resultado de dos tiradas.



Sistema de eliminación binaria



Ejemplo de tirada cara - ceca

TIRADAS DE MONEDAS	EQUIPO CAMPEÓN
	Equipo 1 
	Equipo 2 
	Equipo 3 
	Equipo 4 

Secuencia de tiradas que consagra campeón a cada equipo

Lo que hicimos fue utilizar los resultados del lanzamiento de la moneda para codificar un conjunto de datos. A los resultados de la tirada usualmente los llamamos cara y ceca, pero también los podríamos llamar ganar y perder, A y B o 0 y 1. La computadora utiliza una estrategia similar para representar información: como en la memoria la información se representa usando dos niveles de voltaje (bajo y alto), decimos que trabaja en binario y, para simplificar, abstraemos la noción de voltaje bajo como 0 y la de voltaje alto como 1. A la menor medida de información que permite identificar una de 2 alternativas la llamamos **bit**. Un sistema binario consiste, básicamente, en elegir cómo codificar datos con secuencias de bits.

En esta actividad vimos que, para representar 2 resultados, alcanzó con un solo bit y, para 4, bastó con una secuencia de 2 bits. Podemos plantear las siguientes situaciones: "¿Cuántas tiradas necesitaríamos para desempatar entre 8 equipos? ¿Y entre 16?".

CIERRE

A modo de conclusión, reflexionamos con los alumnos sobre la codificación binaria. Así como en la actividad pudimos representar de manera unívoca cada posible resultado del torneo con combinaciones de caras y cecas, la computadora representa cualquier tipo de información utilizando secuencias de bits. Cuantos más sean los bits de los que dispongamos, mayor cantidad de información podremos representar.

Actividad 2
Golosinas binarias

GRUPAL (4)

OBJETIVOS

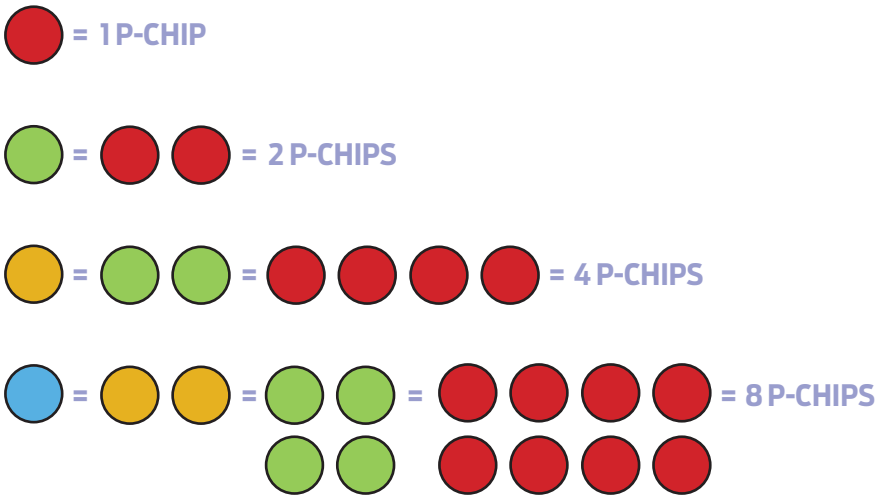
- Presentar la representación de números en un sistema binario.
- Establecer relaciones entre el sistema de numeración decimal y el binario.

MATERIALES

Ficha para estudiantes

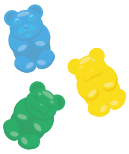
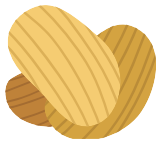
DESARROLLO

En esta actividad buscaremos abordar las ideas principales de la codificación de los números en un sistema binario. Para eso, utilizaremos golosinas de diferentes precios, expresados en una unidad de moneda inventada, los p-chips, que representamos con fichas de 4 colores: rojas, verdes, amarillas y azules. Las fichas rojas valen 1 p-chip, las fichas verdes tienen el mismo valor que 2 fichas rojas, las amarillas valen 2 fichas verdes, y las azules valen 2 amarillas. Una máquina expendedora de golosinas requiere que se introduzca el precio exacto de una golosina en fichas de colores, pero no acepta más de una ficha de cada color.



Valores de las fichas de colores expresados en p-chips

Repartimos la ficha a los estudiantes. En la primera consigna se encontrarán con golosinas de distintos valores y deberán decidir qué fichas usar para obtener cada una. Las golosinas, sus precios en p-chips y las fichas que deben emplear en cada caso se muestran a continuación:

GOLOSINA	PRECIO (EN P-CHIPS)	FICHAS
	1	
	2	
	3	
	5	
	7	
	10	
	15	

Precio de las golosinas en p-chips

Una vez que todos hayan concluido, hacemos una puesta en común. En caso de que algunos no hubiesen respondido correctamente, repasamos los valores de cada ficha y analizamos por qué las combinaciones propuestas no son adecuadas.

Luego, razonamos entre todos sobre cuál es el valor máximo que puede tener una golosina expedida por esta máquina, que es lo que pide investigar la segunda consigna. Como no puede usarse más de una ficha de cada color, el precio más alto es el que corresponde a usar exactamente una ficha de cada color, es decir, 15 p-chips. Para representar montos más grandes, harían falta más colores.

15 P-CHIPS

Valor máximo de una golosina de la máquina expendedora

Les pedimos que resuelvan la tercera consigna. Allí se les pide que representen los valores de 0 a 15 usando p-chips, manteniendo la restricción de no usar más de una ficha de cada color para representar cada número. Cada vez que necesiten una ficha de un color deben escribir un uno y, cuando no, un cero. A continuación se exhibe la solución esperada:

NÚMERO				
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Solución de la tercera consigna

Comentamos a los estudiantes: "Hemos representado cada número como una secuencia de dos símbolos: 0 y 1. Es decir, usando un sistema binario".

Continuamos: "Al trabajar con un sistema de numeración decimal utilizamos potencias de 10 para cada uno de los dígitos. Por ejemplo, si queremos representar el número 123 en el sistema decimal, lo que hacemos es $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$, que por simplicidad escribimos como 123". Para fijar lo enunciado, copiamos la figura en el pizarrón.

CENTENA **1**	1×10^2	=	1×100 =	**100**
DECENA **2**	2×10^1	=	2×10 =	**20**
UNIDADES **3**	3×10^0	=	3×1 =	**3**
				123

El número 123 como suma de potencias de 10

Continuamos: "Este mismo razonamiento se aplica a los sistemas de numeración binarios –que usan solo dos símbolos–, que son los utilizados por computadoras, pero usando potencias de 2 en lugar de potencias de 10. Si queremos representar el 123 en binario hacemos:
$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$, que para simplificar escribimos como 1111011".

1	1×2^6	=	1×64 =	**64**
1	1×2^5	=	1×32 =	**32**
1	1×2^4	=	1×16 =	**16**
1	1×2^3	=	1×8 =	**8**
0	0×2^2	=	0×4 =	**0**
1	1×2^1	=	1×2 =	**2**
1	1×2^0	=	1×1 =	**1**
				123

El número 123 como suma de potencias de 2

CIERRE

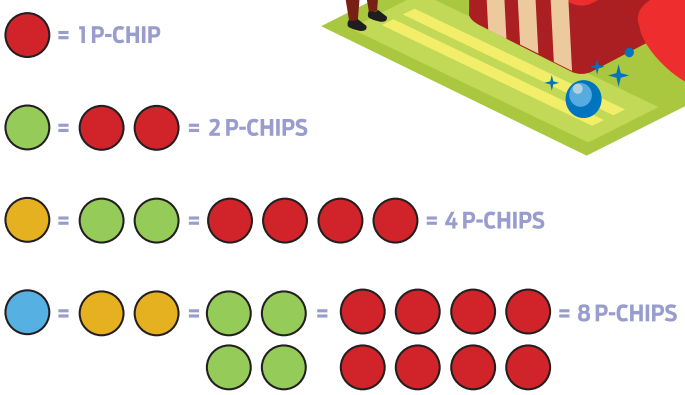
Concluimos que, cuando se representan números de esta forma, mediante dos símbolos –en este caso 0 y 1–, decimos que usamos un **sistema de numeración binario**. Por ejemplo, el número 9 se representa en binario como 1001, porque usa una ficha de valor 8 y una de valor 1, y ninguna de valor 4 y 2 (9 = 8 + 1). Es interesante destacar que esta forma de representar números es similar a la decimal, pero en lugar de usar unidades, decenas, centenas, etc. –que son todas potencias de 10–, utiliza potencias de 2. También es importante mostrar que en binario se requieren secuencias de dígitos más largas para representar los mismos números, dado que se dispone de menos símbolos.

NOMBRE Y APELLIDO:

CURSO: FECHA:

GOLOSINAS BINARIAS

¿Alguna vez escuchaste hablar del p-chip? Es una moneda que sirve para ¡comprar golosinas! Una ficha roja equivale a 1 p-chip. Y hay otras con los siguientes valores:



La máquina expendedora de golosinas que acepta p-chips requiere que se introduzca el monto exacto en fichas de colores, pero no está preparada para aceptar más de una ficha de cada color.

1. Indicá en la siguiente tabla qué fichas tenés que usar para comprar estas golosinas:

GOLOSINA	PRECIO (EN P-CHIPS)	FICHAS
	1	
	2	
	3	
	5	
	7	
	10	
	15	

NOMBRE Y APELLIDO:

CURSO: FECHA:

2. ¿Cuál es el monto más alto en p-chips que se puede pagar por una golosina en esta máquina?

3. Completá la siguiente tabla para expresar los números de la primera columna. En cada fila, si necesitás la ficha de ese color, escribí un 1; si no, un 0.

NÚMERO				
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

¿CEROS Y UNOS?

Las computadoras no usan los símbolos 0 y 1 para representar información. Internamente, la información contenida en la memoria se codifica usando dos niveles de voltaje: bajo y alto.

Actividad 3
¿Cuánto pesa la información?

DE A DOS

OBJETIVO

- Comprender las unidades de almacenamiento que usan las computadoras.

MATERIALES

- Calculadora
- Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es que los estudiantes se familiaricen con las unidades de medida de información más usadas y sus magnitudes. Además, la dimensión medida en esas unidades se corresponde con el espacio que ocupa la información en medios físicos, como los discos, y el tiempo necesario para procesarla. El poder de cómputo de una computadora es una medida relativa a la cantidad de bits de información que puede procesar por unidad de tiempo.

Comenzamos repartiendo las fichas a los alumnos y les pedimos que analicen el cuadro que muestra la relación entre las unidades de medida de información.

UNIDAD	Byte	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)
TAMAÑO	8 bits	1024 bytes	1024 KB	1024 MB	1024 GB

Unidades de medida de Información

Se puede ver aquí que un byte es una agrupación de 8 bits. Y que, luego, cada unidad equivale a 1024 veces la unidad anterior.[1] Por ejemplo, en un terabyte hay:

1 TB	
1024 × 1 =	1024 GB
1024 × 1024 =	1.048.576 MB (más de 1 millón de MB)
1024 × 1.048.576 =	1.073.741.824 KB (más de 1000 millones de KB)
1024 × 1.073.741.824 =	1.099.511.627.776 Bytes (más de 1 billón de bytes)
8 × 1.099.511.627.77680 =	8.796.093.022.208 bits (más de 8 billones de bits)

La información se almacena en la computadora como una secuencia de bits. Por lo tanto, cuando se mide el espacio que ocupa la información, se utilizan los bits y sus múltiplos (bytes, MB, KB, etc.) como unidades. Es posible que los alumnos hayan escuchado términos como terabyte, gigabyte o megabyte, pero quizás no sepan con exactitud qué valores representan.

[1] La razón de que el número sea 1024 es que es una potencia de 2, lo cual lo convierte en un número sencillo de manejar para una computadora; sin embargo, algunas personas utilizan potencias de 10 en lugar de potencias de 2, y entonces consideran que 1 GB son 1000 MB. Por eso, por ejemplo, un pendrive que se vende como de 2 GB tiene usualmente capacidad para solo 1.86 GB de información.

Por lo tanto, antes de abordar el problema, podemos presentar la siguiente analogía: "Supongamos que cada letra de un libro ocupa 1 byte.[1] Entonces, una página de ese libro podría tener 1 KB de información, y un libro de 1024 páginas –un libro bastante gordo– sería el equivalente de 1 MB. Una habitación en la que haya 1024 libros distribuidos en grandes bibliotecas, contendría 1 GB de información. ¿Cuánto sería entonces 1 TB? Serían unas 1024 habitaciones... Si pensamos en un edificio de 10 pisos con 4 departamentos por piso y 4 habitaciones por departamento, en cada edificio hay 160 habitaciones. O sea, se precisan 6 edificios de 10 pisos y uno de 5 pisos, cada uno lleno de libros desde el piso hasta el techo de todas las habitaciones para tener 1 TB de información. Y todo eso lo podemos guardar en un disco rígido que cabe en la mochila. Impresionante, ¿no?". Para clarificar lo expuesto, dibujamos en el pizarrón la siguiente tabla:

1 byte		Cada letra de un libro
1 Kb		Página de un libro
1 MB		Libro de 1024 páginas
1 GB		Una habitación llena de libros
1 TB		6 edificios de 10 pisos y uno de 5 pisos

Esto entra en un disco de un Terabyte.

Analogía para dimensionar volúmenes de información

Luego de esta breve introducción, presentamos el problema a resolver: determinar cuánto tiempo lleva procesar toda la información de un disco rígido de 1 TB, suponiendo que *(i)* contamos con 16 GB de memoria RAM; *(ii)* cargar 1 GB en la RAM requiere 1 segundo; y *(iii)* analizar 100 MB previamente cargados en la memoria también toma 1 segundo.

[1] Como se verá más adelante, esto no siempre es así, pero hacemos una simplificación para que se comprenda la analogía.

Si tuviesen dificultades para resolver la consigna, podemos plantear las siguientes preguntas para orientarlos:

¿Cuánto se tarda en llenar la memoria RAM de la computadora con datos del disco?
Como cargar 1 GB tarda un segundo y la RAM tiene 16 GB, se requieren 16 segundos para llenarla.

¿Cuánto se tarda en procesar los datos ya cargados en la RAM?
La RAM tiene una capacidad de 16 GB = 16 × 1024 MB = 16.384 MB. Como procesar 100 MB requiere 1 segundo, hacer lo propio con todo el contenido de la RAM toma 16.384 / 100 = 163,84 segundos.

¿Cuántas veces hay que repetir el procedimiento de carga de datos y procesamiento?
Como la capacidad del disco rígido es de 1 TB = 1024 GB, y estos se procesan en bloques de 16 GB, el procedimiento se repite 1024 / 16 = 64 veces.

Con lo recién expuesto, resulta sencillo llegar a la respuesta. Hay que repetir 64 veces la carga de información (que demora 16 segundos), y el procesamiento de cada carga (que requiere 163.84 segundos): 64 × (16 + 163,84) = 64 x 179,84 = 11.509,76 segundos. Casi 192 minutos; es decir, más de 3 horas.

Una vez que todos hayan concluido, hacemos una puesta en común para que los estudiantes compartan las distintas formas en las que llegaron al resultado.

Retomando la analogía, podríamos ver que el procesador tarda 1 segundo en entrar a una habitación, y 1 segundo en verificar cada libro, lo cual podría considerarse muy rápido. Aun así, demoraría más de 3 horas en verificar que todos los libros estuvieran bien. En este caso, las cuentas son exactamente las mismas, pero la analogía ayuda a mostrar de forma más concreta la cantidad de información involucrada. Es importante notar que, a pesar de que estamos procesando información a una velocidad de más de 800 millones de bits por segundo, procesar todo un terabyte, una cantidad estándar para los discos de hoy en día, demanda mucho tiempo.

Los términos que usamos usualmente para describir la capacidad de almacenamiento de una computadora representan una escala precisa de la cantidad de bits que se puede guardar en cada tipo de dispositivo. Las computadoras poseen una capacidad acotada de almacenamiento, lo que restringe la cantidad de programas y archivos que podemos tener guardados. Si bien actualmente los dispositivos tienen cada vez mayor capacidad de almacenamiento, de todas maneras hay limitaciones en cuanto al tiempo que lleva procesar esos volúmenes de datos.

CIERRE

Por último, mencionamos cuánta información contienen algunos de los archivos que se utilizan de manera cotidiana. Comentamos que un disco de música se puede estimar en alrededor de 700 MB y una película en 4,7 GB. Además, hoy en día es habitual que una foto tomada con un teléfono inteligente ocupe alrededor de 5 MB.

¿CUÁNTO PESA LA INFORMACIÓN?

¿Sabías que el espacio que ocupa la información se puede medir? La unidad más chica es un bit, que almacena uno de dos posibles valores. Además, hay otras unidades de medida:

UNIDAD	Byte	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)
TAMAÑO	8 bits	1024 bytes	1024 KB	1024 MB	1024 GB

1. ¡Se cortó la luz mientras usábamos la computadora! Al reiniciarla, aparece un mensaje que dice que debemos esperar a que verifique la integridad de los datos en nuestro disco rígido, pero no nos dice cuánto va a tardar. ¡Buscá la calculadora!

Sabiendo que tenemos un disco rígido de 1 TB y 16 GB de memoria RAM, queremos conocer cuánto tarda la verificación del disco completo. Para ello, debemos tener en cuenta que el proceso de verificación realiza los siguientes pasos:

- Se cargan datos no verificados del disco a la memoria RAM hasta que esté llena. Esto tarda 1 segundo por cada GB de datos.
- La verificación de la integridad de los datos tarda 1 segundo por cada 100 MB de datos cargados en la RAM.
- Se repiten los dos puntos anteriores hasta que no queden datos sin verificar en el disco.

¿Cuántos segundos hacen falta? Y eso, ¿cuántas horas son?

_____

_____

¿SABÍAS QUE...

… la expresión integridad de los datos hace referencia a que los datos guardados en el disco no hayan sufrido alteraciones? Por ejemplo, si se estaba escribiendo en el disco en el momento en que se cortó la luz, existe la posibilidad de que los datos estén incompletos, o que incluso se haya dañado la superficie del disco. Por eso hace falta verificarlos.

EL DATO

¿Sabés qué tamaño tiene un disco de música? ¿Y el archivo de una película? Un álbum en CD puede ocupar 700 MB, y el DVD de una película, 4,7 GB. ¡Eso quiere decir que, en un disco de 1 TB, entran casi 1500 discos de música o 220 películas!

Secuencia Didáctica 2

FORMATOS DE IMÁGENES

Estamos muy habituados a manipular imágenes digitales: sacamos fotos con nuestros teléfonos, mandamos gráficos por servicios de mensajería instantánea y miramos películas en televisores inteligentes. ¿Cómo hacen nuestros dispositivos para representar, transformar y manipular las imágenes?

Esta secuencia didáctica aborda la representación de imágenes digitales y algunos temas relacionados: la codificación de sus colores en RGB, la aplicación de filtros para transformarlas y la posibilidad de comprimirlas para que ocupen menor cantidad de espacio en un medio físico. Todas las actividades se trabajan con computadora usando proyectos de Gobstones.

OBJETIVOS

- Mostrar modos de representar imágenes.
- Construir filtros de transformación de imágenes.
- Explicar que la información se puede comprimir y descomprimir.

Actividad 1
Pintamos la camiseta de colores

DE A DOS

OBJETIVOS

- Presentar una forma de representar imágenes en color usada en los sistemas digitales.
- Comprender la noción de píxel.

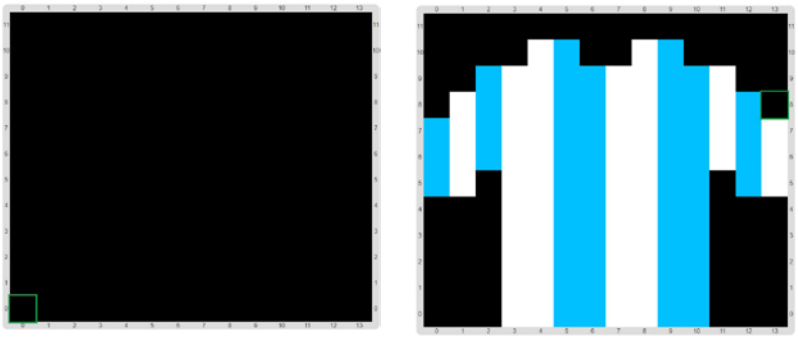
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

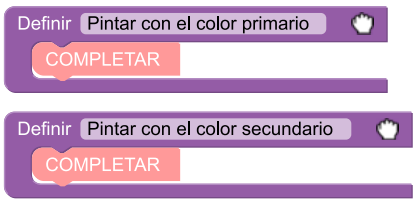
Existen diversas maneras de representar una imagen en la computadora. La más común es tener una grilla de *n* filas por *m* columnas, en donde para cada celda o píxel se almacenan tres valores: una cantidad de rojo, una cantidad de verde y una cantidad de azul. La mezcla de estos tres valores produce un color determinado para cada punto de la imagen. En esta actividad proponemos indagar sobre este sistema de representación usando un proyecto de Gobstones.

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto "Pintamos la camiseta de colores". El objetivo de la actividad es pintar una camiseta con 2 colores distintos. En la imagen se observa que el tablero inicial está totalmente pintado de negro mientras que en el tablero final se ha logrado representar una camiseta de Argentina.



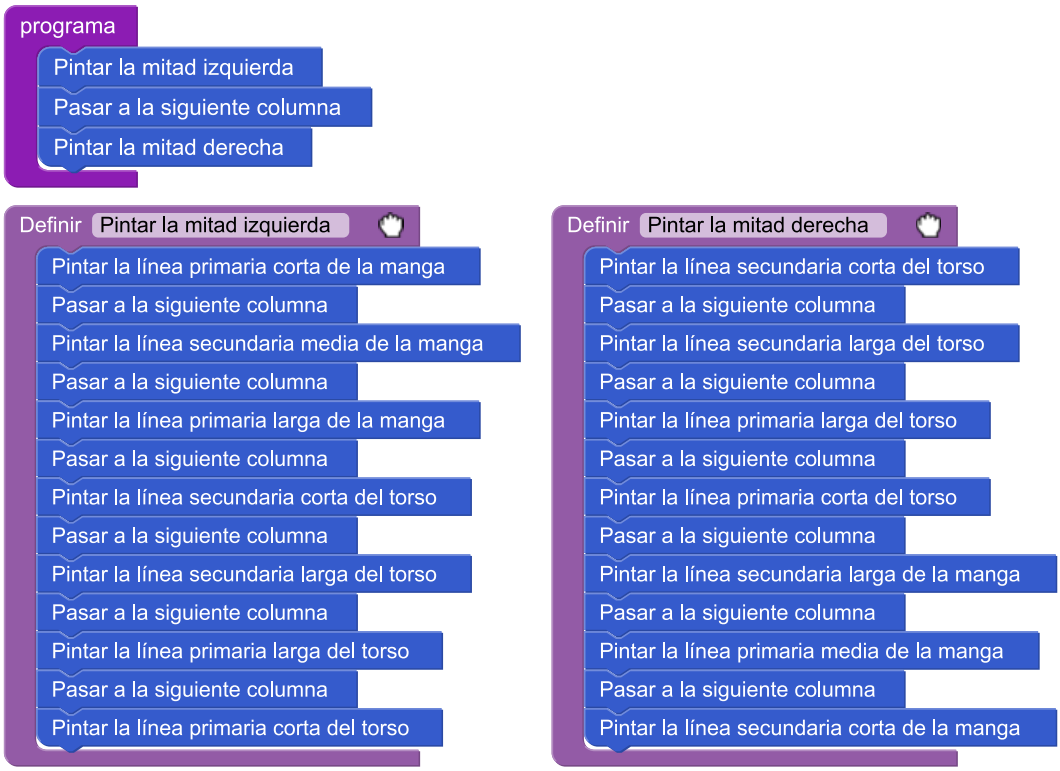
Tablero inicial y posible tablero final

El cuerpo del programa principal ya está resuelto, pero para ejecutarlo falta completar dos procedimientos: `Pintar con el color primario` y `Pintar con el color secundario`.



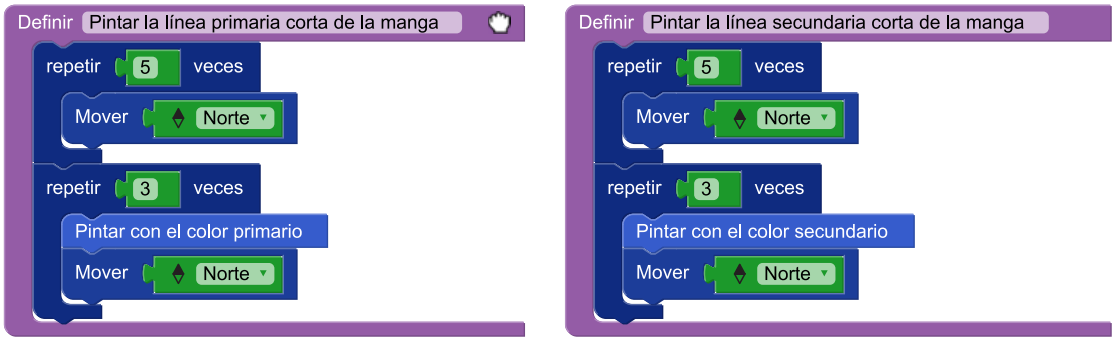
Procedimientos incompletos

La estrategia de solución está plasmada en el cuerpo principal del programa: pintar primero la mitad izquierda de la camiseta y, luego, la derecha. Al analizar cómo se colorea cada mitad, vemos que el enfoque adoptado es pintar una columna, pasar a la siguiente, pintar otra, pasar a la siguiente y así hasta colorear la última columna.



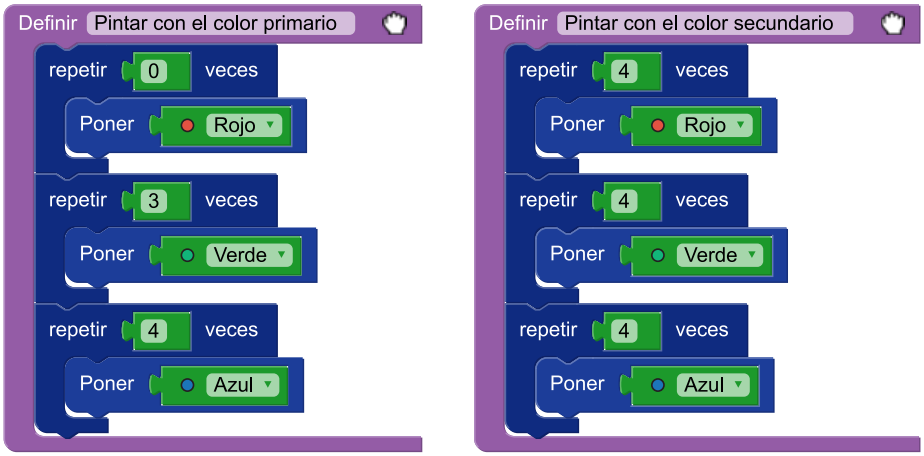
Cuerpo principal del programa y procedimientos `Pintar la mitad izquierda` y `Pintar la mitad derecha`

Al observar cómo están resueltos los procedimientos que forman parte de `Pintar la mitad izquierda` y `Pintar la mitad derecha`, notamos que todos los procedimientos *Pintar la línea...* son similares. A modo de ejemplo, se muestran `Pintar la línea primaria corta de la manga` y `Pintar la línea secundaria corta de la manga`, en los que se pone en evidencia que es necesario completar los procedimientos `Pintar con el color primario` y `Pintar con el color secundario` debido a que son invocados.



Procedimientos `Pintar la línea primaria corta de la manga` y `Pintar la línea secundaria corta de la manga`

Para poder pintar una celda de un color en particular, se deben colocar entre 0 y 4 bolitas de los colores rojo, verde y azul. Al habilitar la vestimenta, el color resultante será el que se obtenga de la mezcla aditiva de los colores de las bolitas; además, la cantidad de bolitas de un color indica la intensidad de dicho color en la mezcla. Es probable que en el transcurso de la actividad los alumnos se pregunten qué combinación de bolitas genera un color determinado. Se espera que exploren varias elecciones de colores hasta encontrar el diseño que más les guste. En la solución que proponemos, se muestran las combinaciones de colores que generan el celeste y el blanco de la camiseta de Argentina.



Procedimientos `Pintar con en el color primario` y `Pintar con el color secundario`

La estrategia de combinar distintas cantidades de rojo, verde y azul permite generar todos los colores; esto se conoce como codificación RGB por *red*, *green* y *blue* (rojo, verde y azul en inglés). En esta actividad solamente podemos diferenciar 5 niveles de intensidad de cada color, de 0 a 4. Así que, en este caso, se pueden representar un total de 5 × 5 × 5 = 125 colores distintos. Sin embargo, se puede pensar en sistemas que permiten más niveles de cada uno de los 3 colores, con lo que se logra una paleta más amplia y una mayor variedad de tonalidades.

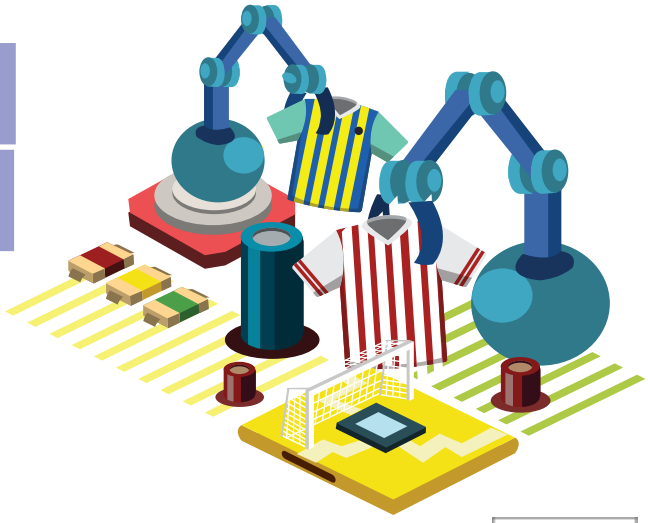
CIERRE

Para concluir, les comentamos a los estudiantes que en los monitores, por ejemplo, cada píxel o punto de la pantalla, está compuesto por 3 pequeñas luces: una roja, una verde y una azul. Al aumentar o disminuir la intensidad de cada luz, se generan distintos colores. Pero, en vez de tener 5 valores para cada una de las tres luces, se suele tener 256, lo que permite representar un total de 256 × 256 × 256 = 16.777.216 colores (más de 16 millones). A esto se lo suele conocer como *color real*, ya que se aproxima a lo que el ojo humano puede diferenciar.

NOMBRE Y APELLIDO:

CURSO: FECHA:

PINTAMOS LA CAMISETA DE COLORES

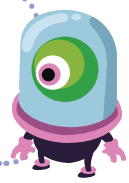


¿De qué cuadro sos hincha? ¡Ahora vas a poder pintar la camiseta de tu club!

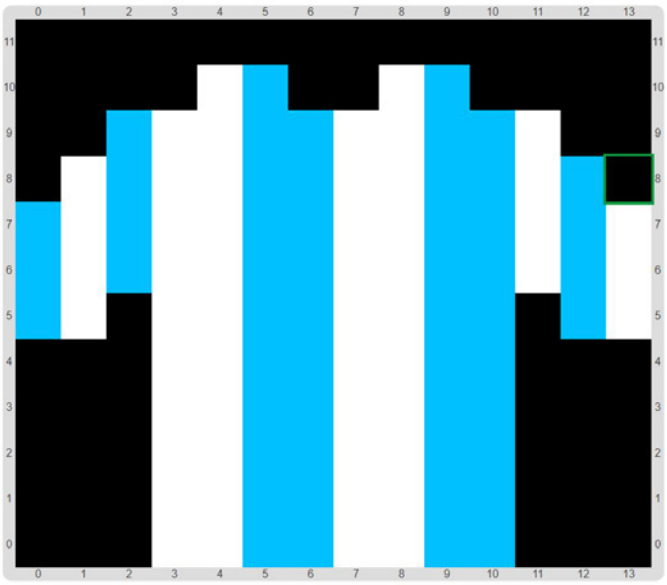
1. Abrí el proyecto "Pintamos la camiseta de colores". Vas a encontrar un programa que dibuja camisetas con dos colores, pero hay un problema: falta completar los procedimientos `Pintar con el color primario` y `Pintar con el color secundario`.

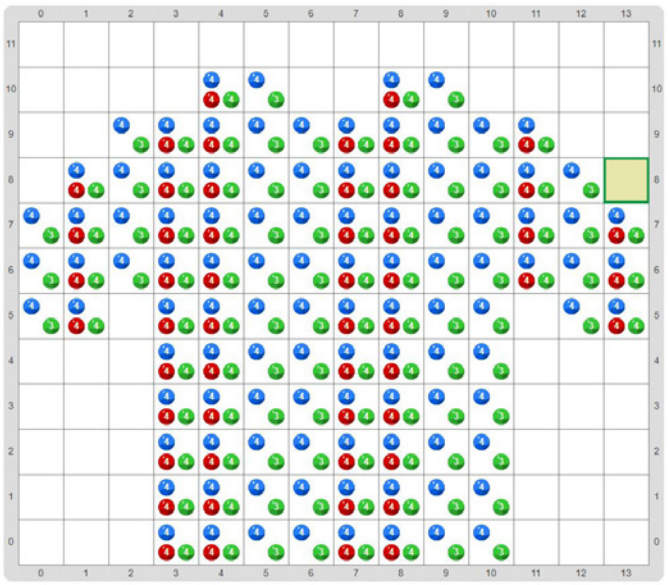
PARA TENER EN CUENTA

Para formar colores, estos procedimientos deben poner entre 0 y 4 bolitas de los colores `Rojo`, `Verde` y `Azul`.



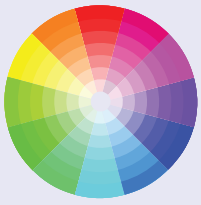
Experimentá con diferentes cantidades de bolitas rojas, verdes y azules para obtener distintas camisetas. Te mostramos cómo quedaría una posible combinación de colores con y sin vestimenta:





EL DATO

La codificación de colores con rojo, verde y azul se conoce como RGB (del inglés, *red*, *green*, *blue*). Usa 3 números –uno para cada color– entre 0 y 255, que marcan el nivel de intensidad de cada color que se utilizará al mezclarlos. De este modo se pueden diferenciar ¡más de 16 millones de colores! Acá, en cambio, utilizamos números entre 0 y 4, que permiten representar $5 \times 5 \times 5 = 125$ colores diferentes.



Actividad 2
Escala de grises

DE A DOS

OBJETIVOS

- Conocer formas alternativas de representación de imágenes.
- Comprender cómo se puede transformar una imagen a color a escala de grises.

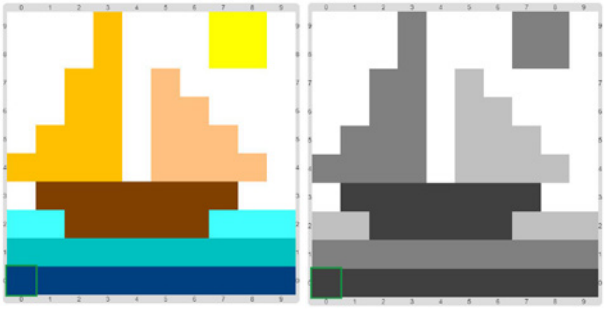
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

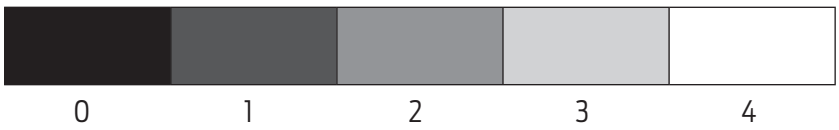
En esta actividad, los alumnos transformarán una imagen a color en una en escala de grises.

Luego de repartir la ficha, los estudiantes deben abrir en las computadoras el proyecto "Dibujamos en escala de grises". El objetivo es completar el programa que transforma a escala de grises una imagen de 10 píxeles de ancho por 10 de alto.



Tableros inicial y final

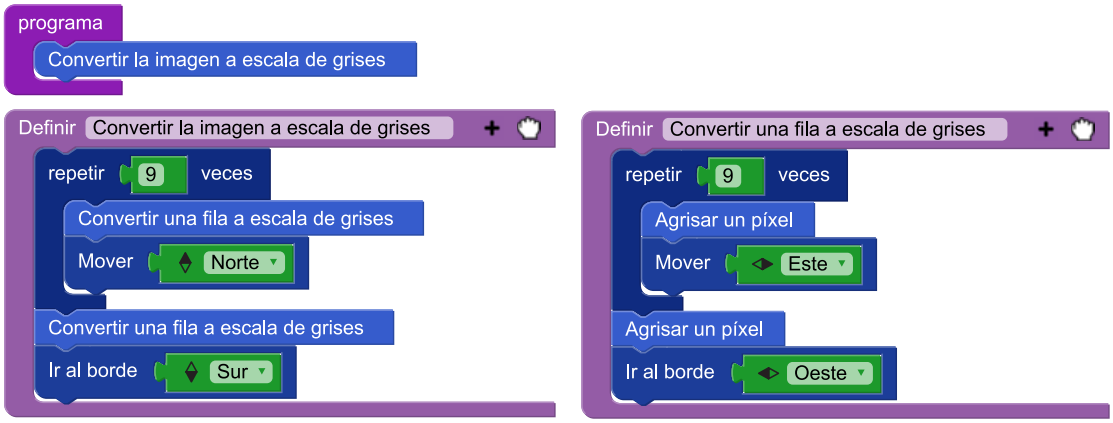
Les contamos que una imagen en escala de grises no contiene información de color, sino que está formada únicamente por distintos tonos de gris. Se trata de lo que habitualmente se conoce como "imágenes en blanco y negro". Preguntamos: "¿Cómo creen que se podría representar una imagen en blanco y negro?". Hay varias respuestas posibles. En particular, en la computadora, para cada píxel se utiliza un único valor numérico en un rango dado que indica en qué posición de la escala se encuentra: usualmente 0 es negro y el máximo valor del rango es blanco; cualquier valor intermedio representa algún tono de gris.



Una escala de grises

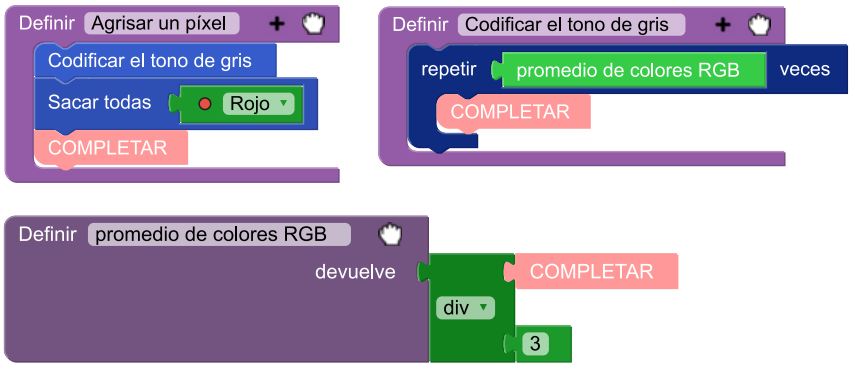
Continuamos: "¿Cómo podemos hacer para convertir una imagen a color codificada en RGB a escala de grises?". Damos unos minutos para que piensen en parejas cómo se podría realizar la conversión y luego hacemos una puesta en común para que expongan sus ideas. Para pasar la imagen a escala de grises, transformamos de a un píxel por vez. En cada celda, hay que cambiar uno de los 125 posibles colores en RGB por una de las 5 posibles tonalidades de grises, ya que usaremos entre 0 y 4 bolitas negras para representar el gris. Para cada píxel podemos calcular el promedio entre las cantidades de bolitas de los 3 colores. Este valor nos permite obtener la intensidad media del color que representa un píxel RGB, que normalmente es muy similar a la posición correspondiente para un color dado en la escala de grises.

En el espacio del programa encontrarán una parte del programa ya resuelta y otras que falta terminar. Ya están completos el cuerpo principal del programa y los procedimientos `Convertir la imagen a escala de grises` y `Convertir una fila a escala de grises`. Al verlos, se observa la estrategia de solución: la imagen se transforma a escala de grises de a una fila por vez, y cada fila se transforma de a un píxel por vez.



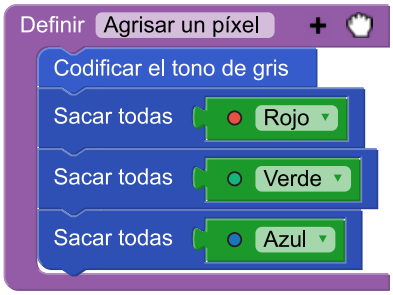
Parte del programa que está resuelta

Falta completar los procedimientos `Agrisar un píxel` y `Codificar el tono de gris` y la función `promedio de colores RGB`.



Parte incompleta del programa

El procedimiento `Agrisar un píxel`, en primer lugar, se encarga de incorporar las bolitas negras que hacen falta para obtener el tono de gris buscado –invocando al por ahora incompleto `Codificar el tono de gris`– y retira luego todas las bolitas rojas, verdes y azules usando `Sacar todas [ ]`, disponible en *Biblioteca* > *Procedimientos*.

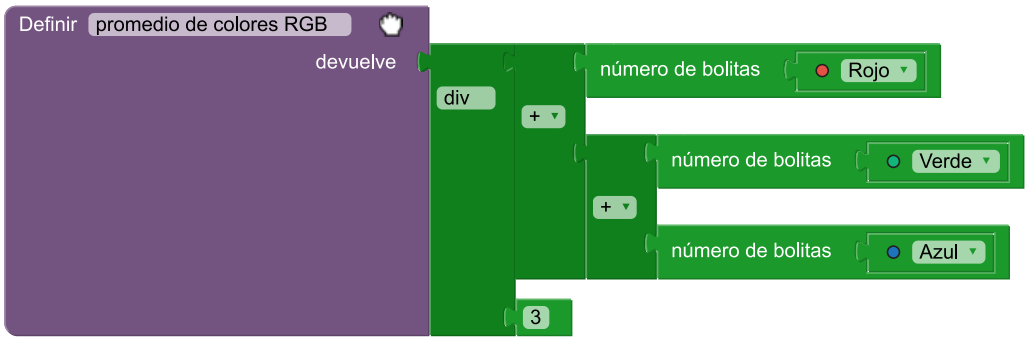


Procedimiento `Agrisar un píxel`

Para completar el procedimiento `Codificar el tono de gris` solamente agregamos `Poner [Negro]` dentro del cuerpo de `repetir [promedio de colores RGB] veces`. De esta forma, colocamos la cantidad de bolitas negras necesarias según el promedio obtenido entre las cantidades de bolitas rojas, verdes y azules.

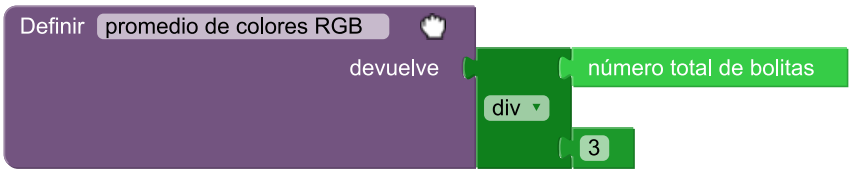
Procedimiento `Codificar el tono de gris`

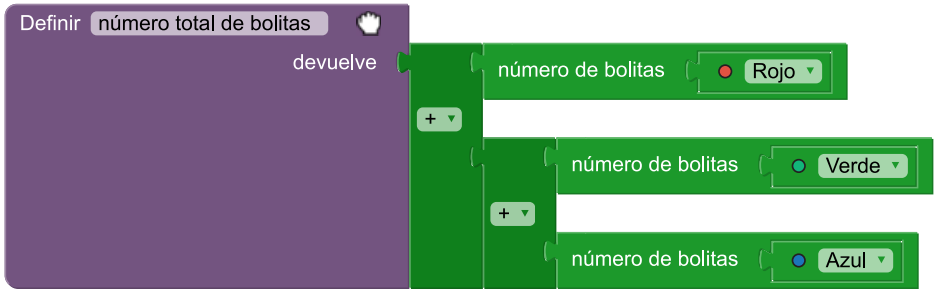
Por último, falta terminar la función `promedio de colores RGB` para que sume la cantidad de bolitas rojas, verdes y azules que hay en un píxel y divida ese valor por 3. Para hacerlo, usaremos el operador de suma, el de división y el sensor `número de bolitas [ ]`.



Función `promedio de colores RGB`

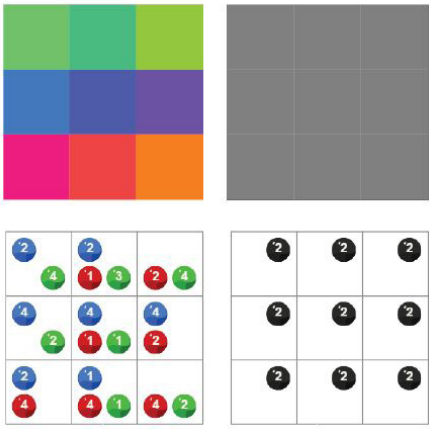
Para que `promedio de colores RGB` resulte más claro, podemos crear y programar una función `número total de bolitas` que sume la cantidad de bolitas rojas, verdes y azules que hay en una celda.





Funciones `promedio de colores RGB` y `número total de bolitas`

Una vez que todos hayan completado sus programas preguntamos: "El procedimiento inverso, que parte de la imagen en escala de grises y recupera la imagen original, ¿es posible?". Guiamos la discusión para llegar a la conclusión de que no lo es: cuando calculamos el promedio, perdimos la información original de los colores. Varios colores dan como resultado el mismo tono de gris, como, por ejemplo, 4 bolitas rojas y 2 azules, 2 verdes y 4 azules, o cualquier combinación de 6 bolitas entre rojas, verdes y azules. La representación de la imagen en gris requiere menos bits porque efectivamente contiene menos información.



A partir de una imagen en escala de grises no se puede reconstruir la imagen original

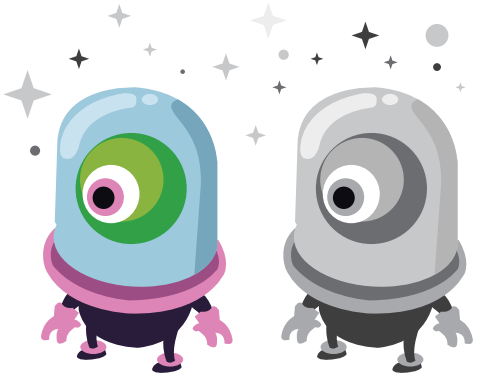
CIERRE

Como reflexión final, mencionamos que las operaciones de filtro comunes en la mayoría de las cámaras digitales modernas y también en muchas aplicaciones que permiten editar fotos (como las que suelen tener los teléfonos inteligentes) efectúan procesamientos sobre los píxeles de forma similar a como hicimos el pasaje a escala de grises: realizan una operación por cada píxel de manera independiente.

NOMBRE Y APELLIDO:

CURSO: FECHA:

ESCALA DE GRISES



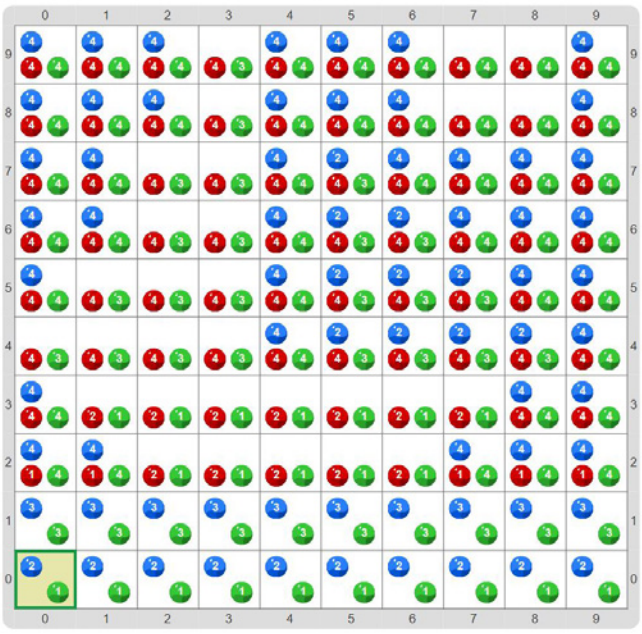
¿Sabés cómo hace una computadora para convertir una imagen a color en una en blanco y negro?

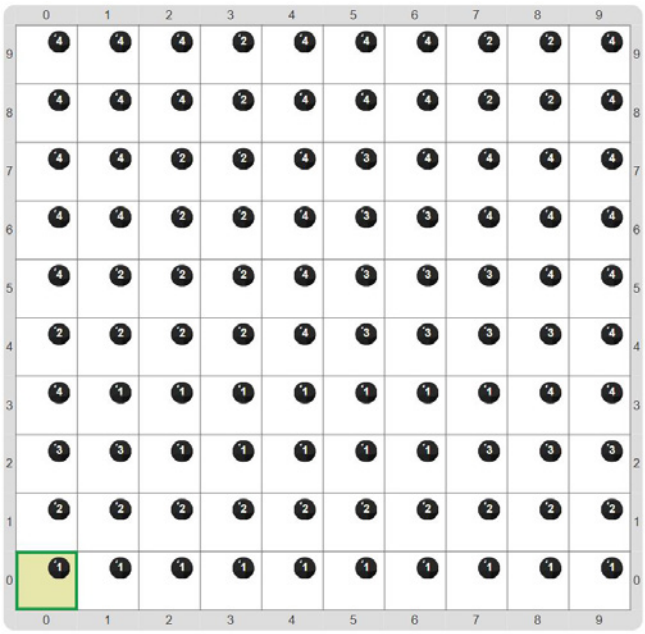
1. Abrí el proyecto "Dibujamos en escala de grises". Vas a encontrar una imagen a color hecha con la codificación RGB: entre 0 y 4 bolitas rojas, verdes y azules en cada celda. Tenés que completar el programa para que la imagen se transforme en una en escala de grises.

Mirá el tablero inicial y el tablero final, sin y con vestimenta:

¿CÓMO REPRESENTAMOS LOS GRISES?

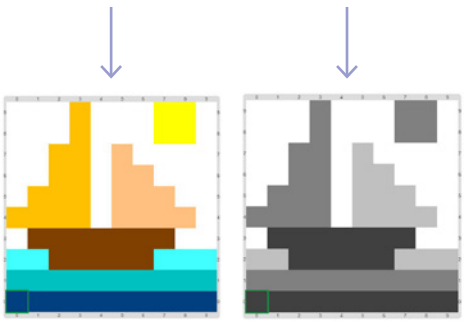
Vamos a usar también entre 0 y 4 bolitas de color negro. El tono de gris está determinado por la cantidad de bolitas que hay en una celda.





VOCABULARIO

El término *monocromático* significa 'de un solo color', es decir que se utiliza, por ejemplo, solamente el color gris, en diferentes intensidades, para representar una imagen en escala de grises.



PROMEDIAMOS LOS COLORES

Asegurate de que la función `promedio de colores RGB` informe el promedio de las cantidades de bolitas rojas, verdes y azules que hay en una celda. Vas a tener que retirar las bolitas de colores y colocar en su lugar esa cantidad de negras.

Actividad 3
Compresión de imágenes

DE A DOS

OBJETIVOS

- Presentar la idea de compresión de datos.
- Que los estudiantes se familiaricen con representaciones comprimidas de imágenes.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

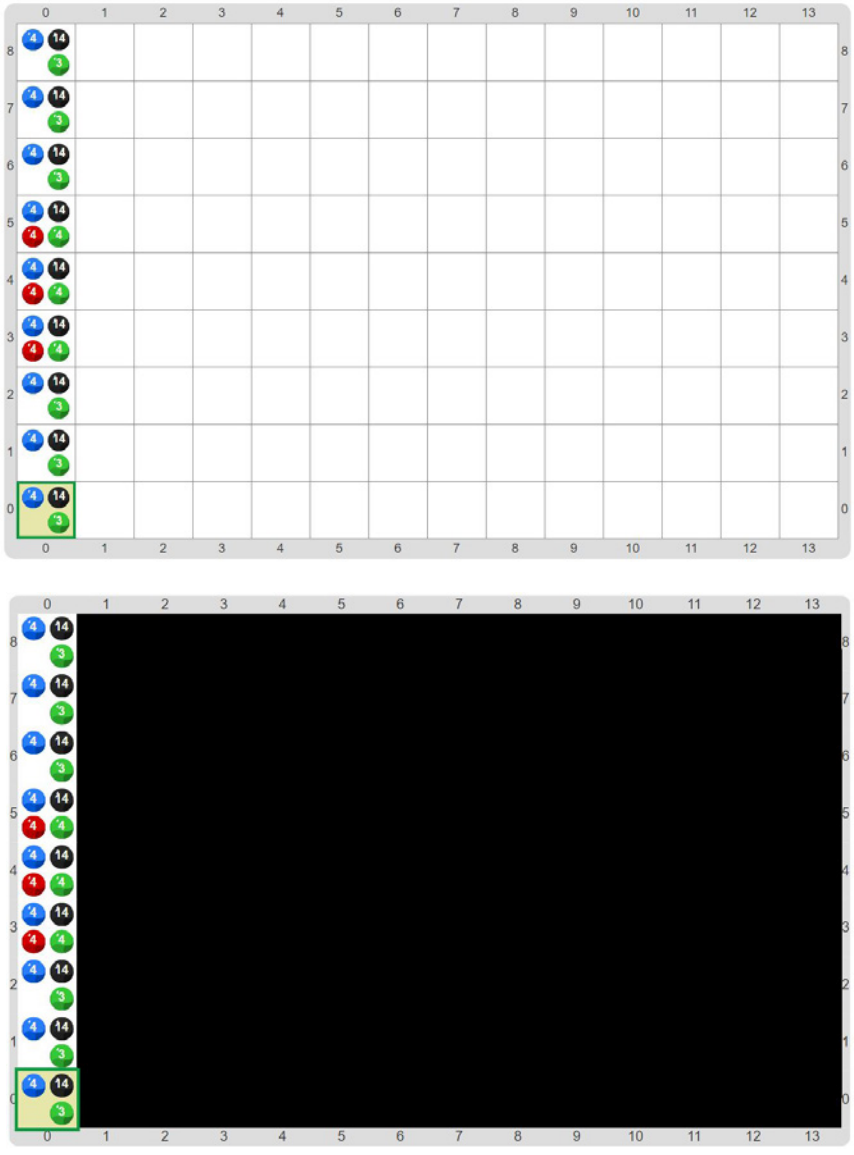
AYUDAMEMORIA

Codificamos colores en RGB usando entre 0 y 4 bolitas rojas, verdes y azules. De esta forma podemos representar hasta 125 colores distintos.

DESARROLLO

La compresión de datos representa un área importante en las Ciencias de la Computación. En esta actividad, abordaremos una posible forma de comprimir una imagen.

Les repartimos la ficha a los estudiantes y les pedimos que abran el proyecto "Compresión de imágenes". Encontrarán un tablero que tiene bolitas únicamente en la primera columna; el resto de las celdas están vacías. Si habilitan la vestimenta, verán que la primera columna continúa mostrando las bolitas, mientras que el resto de la grilla se tiñe de negro. Esto se debe a que la vestimenta está basada en la codificación RGB trabajada en actividades anteriores.



Tablero inicial, sin y con vestimenta

Comentamos: "Muchas veces manejamos volúmenes de datos tan grandes que resultan poco prácticos para ciertos usos. Por ejemplo, hoy en día no es extraño que una foto sacada con un teléfono inteligente pese 5 o 10 MB. ¡Esto es muchísimo si queremos mandarla por un servicio de mensajería instantánea!". Para poner en evidencia la existencia de métodos de compresión, vamos a observar la diferencia de tamaños de archivos entre una imagen en su formato original (tal como fue registrada después de la toma) y una versión comprimida por un programa de mensajería instantánea.

Para esto, le indicaremos a algún estudiante que saque una foto con su teléfono y que nos diga cuánto pesa el archivo (en general, se pueden conocer datos técnicos de una imagen accediendo a *información*). Le pediremos que se la mande a alguno de sus compañeros y, cuando este la reciba, deberá decir cuánto pesa el archivo que recibió. Observaremos una diferencia significativa. En un caso, el peso estará en el orden de los MB y, en el otro, de los KB. Preguntamos: "¿Qué habrá pasado en el medio?". Guiamos la conversación para que surja la palabra *compresión*, que, concluimos, consiste en conseguir una nueva representación que requiera menor cantidad de espacio -o de bits-.[1]

Continuamos: "Hasta ahora hemos representado el espacio que ocupa la información con bits, bytes, kilobytes, etc., es decir, con unidades de medida. Sin embargo, en distintos contextos lo podemos calcular de otras maneras. Por ejemplo, en Gobstones, podemos medir la cantidad de espacio que ocupan ciertos datos a partir de la cantidad de bolitas que haya en el tablero".

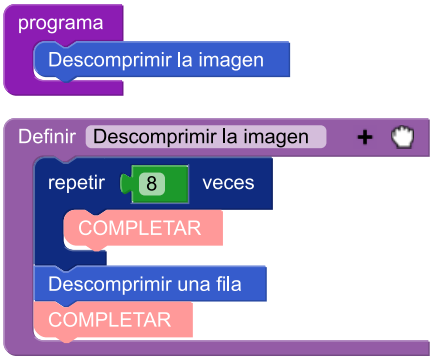
A continuación, les indicamos que activen y desactiven la vestimenta. "Aunque no parezca, lo que observan en el tablero es una codificación comprimida de la bandera argentina. Ustedes no pueden verla porque la vestimenta interpreta combinaciones de entre cero y cuatro bolitas rojas, verdes y azules como 125 colores distintos, pero la información está ahí. Su tarea es descomprimir los datos del tablero para que finalmente se pueda ver nuestra bandera. Además, al concluir la ejecución del programa, el cabezal tiene que quedar posicionado sobre la celda (0,0) del tablero".

Tablero final esperado

Preguntamos: "¿Cómo podemos hacer para pasar del tablero inicial al tablero final? ¿Qué información nos brindan las bolitas de la primera columna? ¿Hay algún patrón que se repita? ¿Cuál es la codificación del celeste y cuál la del blanco? ¿Qué papel juegan las bolitas negras?". Estas preguntas son útiles para orientar a los estudiantes de modo que descubran que 4 bolitas rojas y 3 verdes codifican el celeste, 4 de rojo, 4 de verde y 4 de azul codifican el blanco, y que la cantidad de bolitas negras indica la cantidad de columnas del tablero. Por lo tanto, las bolitas negras señalan la longitud de la fila y las bolitas rojas, verdes y azules el color que debemos usar para pintarla.

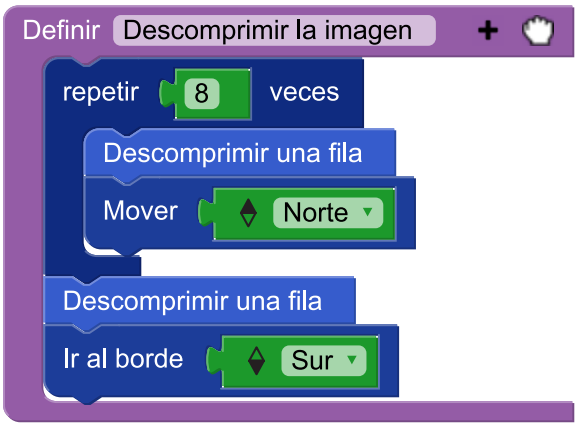
[1] También podemos observar esta diferencia en un solo teléfono si accedemos a la fotografía en la aplicación de galería o cámara y a la misma foto en la conversación del mensajero instantáneo en la que fue enviada.

En el espacio del programa encontrarán una parte que está construida y otra que tienen que programar. El cuerpo principal consiste en una única invocación al procedimiento `Descomprimir la imagen`, que deben completar.



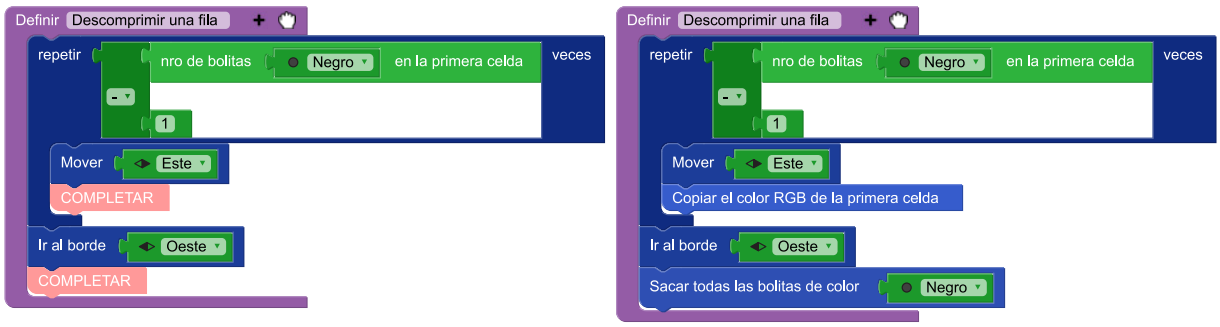
Cuerpo principal del programa y procedimiento `Descomprimir la imagen` sin completar

Teniendo en cuenta que *(i)* el proyecto tiene un procedimiento llamado `Descomprimir una fila` y *(ii)* que el tablero tiene 9 filas, se puede inferir que hay que completar el cuerpo de `repetir [8] veces` de `Descomprimir la imagen` con los bloques `Descomprimir una fila` y `Mover [Norte]` –este último para posicionar el cabezal en la siguiente fila–. La descompresión de la novena y última fila se hace fuera de `repetir [8] veces` porque al tratarse de un caso de borde, hay que evitar que el cabezal caiga fuera del tablero. Además, una vez finalizada la descompresión de todas las filas, para que el cabezal retorne al origen alcanza con invocar `Ir al borde [Sur]`.



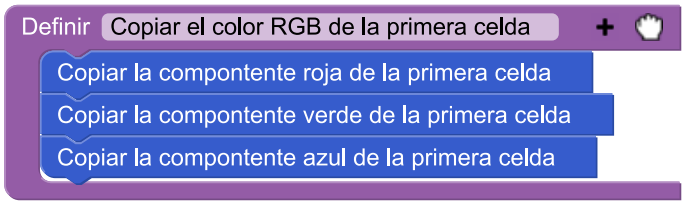
Procedimiento `Descomprimir la imagen`

Para descomprimir una fila, hay que hacer avanzar el cabezal hacia el este hasta que se acaba la fila e ir colocando en cada celda tantas bolitas rojas, verdes y azules como las que hay en la primera celda de la fila en cuestión. Como se mencionó, la longitud de la fila –o sea, la cantidad de columnas– está codificada por la cantidad de bolitas negras de la primera celda. Por lo tanto, la cantidad de veces que se desplaza el cabezal hacia el este es `nro de bolitas [Negro] en la primera celda` - 1. Además, para colocar las bolitas de colores que representan el color de la celda, se debe usar el procedimiento `Copiar el color RGB de la primera celda` (aún no completado), que ya está en el proyecto. A continuación, hay que volver al borde oeste y quitar todas las bolitas de color negro para que en la primera celda de la fila también se codifique el color adecuado. Para hacerlo, usamos el procedimiento `Sacar todas las bolitas de color [ ]` disponible en *Biblioteca* > *Procedimientos*. ¿Qué pasa si no quitamos las bolitas negras? La primera celda quedará con bolitas que no codifican ningún color en RGB, por lo que no se va a mostrar ningún color al activar la vestimenta.



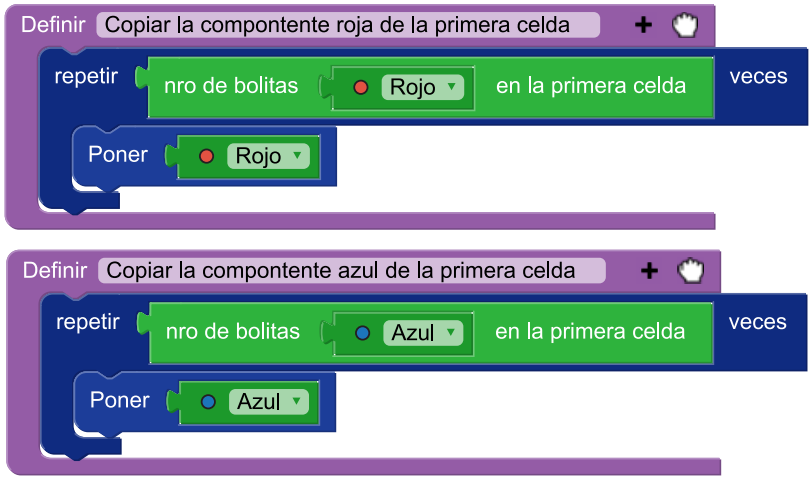
Procedimiento `Descomprimir una fila` antes y después de completarlo

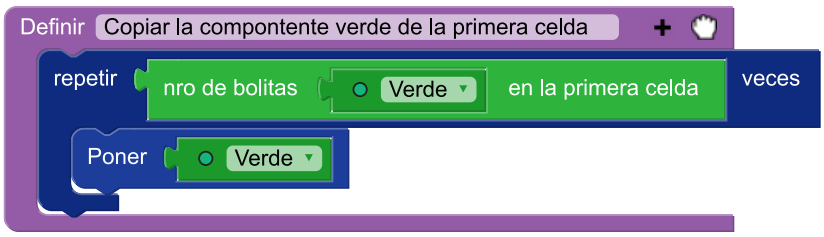
El procedimiento `Copiar el color RGB de la primera celda` ya está resuelto y consiste solo en tres invocaciones para copiar las componentes roja, verde y azul del color buscado.



Procedimiento `Copiar el color RGB de la primera celda`

Para completar el proyecto, falta programar los procedimientos `Copiar la componente roja de la primera celda`, `Copiar la componente verde de la primera celda` y `Copiar la componente azul de la primera celda`. Los tres pueden completarse de forma similar, usando la función disponible en la biblioteca `nro de bolitas [ ] en la primera celda` y variando solo el color que se usa como argumento.





Procedimientos para copiar las componentes roja, verde y azul

Una vez que todos hayan concluido sus programas, preguntamos: "¿El tablero inicial representa la misma imagen que el tablero final?". Sí. "En términos de bolitas, ¿ocupan la misma cantidad de espacio?". Es importante que los estudiantes comprendan que, si bien los dos tableros representan la misma bandera, la versión comprimida lo hace utilizando menos bolitas. Aprovechando que las filas de la bandera comparten una estructura común –todas las celdas de una fila son del mismo color–, se comprime en una celda la información sobre el color de las celdas de la fila y su longitud. Muchos de los métodos de compresión se basan en esta idea: representar de manera más sintética la información repetida.

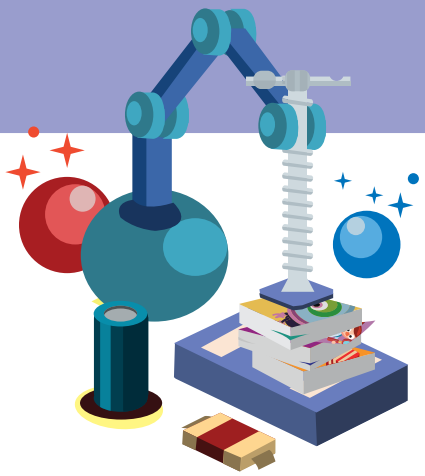
Resulta interesante notar que, a pesar de que las dos codificaciones representan la misma imagen, en la versión comprimida no podemos ver la imagen tal cual es. El motivo es que la vestimenta solo está preparada para mostrar imágenes codificadas según la representación RGB. Por lo tanto, para poder ver la imagen, es necesario decodificar esa codificación. Esto sucede muchas veces cuando se trabaja con información comprimida: aunque ocupa menos espacio, ya no puede utilizarse directamente, sino que debe pasar por un proceso de reconstrucción, usualmente llamado *descompresión*.

CIERRE

Terminamos la actividad diciéndoles a los alumnos que la compresión trabajada permite codificar información que luego podemos reconstruir. A los métodos que poseen esta característica se los llama *sin pérdida de información*. Al guardar imágenes en PNG se usa este tipo de compresión. Además, existen otros métodos con pérdida de información, es decir que una vez comprimidos los archivos, no permiten reconstruir la información original. Las imágenes guardadas en formato JPG tienen esta característica. Si bien degradan la calidad de la imagen, son muy útiles en ciertos contextos, por ejemplo, para mandar una foto originalmente muy pesada por un canal de mensajería instantánea.

NOMBRE Y APELLIDO:

CURSO: FECHA:

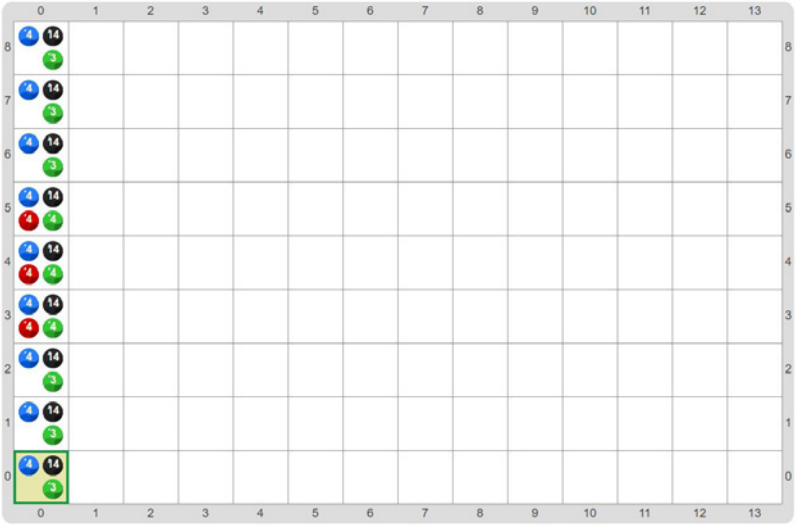


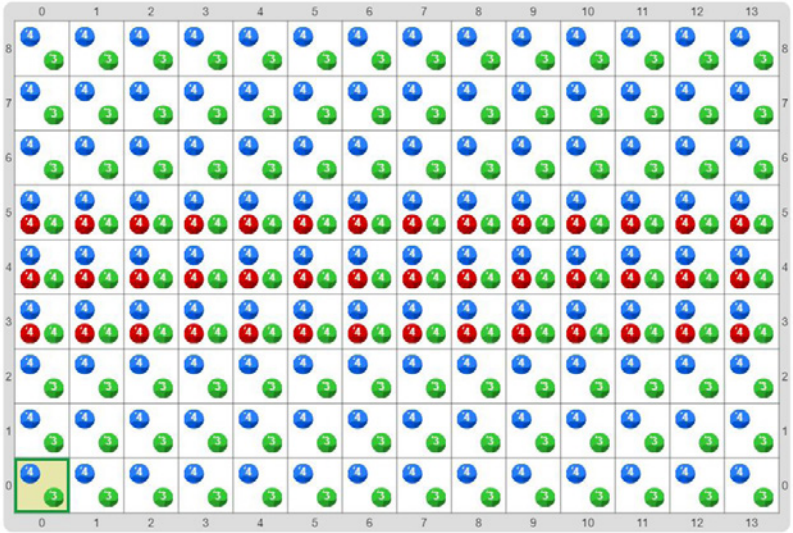
COMPRESIÓN DE IMÁGENES

¿Alguna vez te pasó que quisiste mandar una foto por correo electrónico y no pudiste porque era demasiado pesada? ¿Sabés cómo hacen las computadoras cuando tienen que transmitir enormes volúmenes de datos?

Abrí el proyecto "Compresión de imágenes". Las bolitas en el tablero codifican un dibujo de la bandera argentina, pero la información está codificada de una forma que ocupa menos espacio que en el formato RGB. Tu trabajo consiste en completar el programa para que cambie la codificación a la forma RGB. Al finalizar una ejecución del programa, el cabezal debe quedar posicionado sobre la celda (0,0).

La codificación comprimida solamente tiene bolitas en la primera celda de cada fila. Te mostramos un posible tablero inicial y su correspondiente tablero final, con y sin vestimenta.





AYUDAMEMORIA

Codificamos colores en RGB usando entre 0 y 4 bolitas rojas, verdes y azules. De esta forma, podemos representar hasta 125 colores distintos.

PARA TENER EN CUENTA

En lugar de usar bits, bytes, etc., en esta actividad calculamos el espacio que ocupa la información como la cantidad de bolitas dispuestas sobre el tablero.

¿SABÍAS QUE...

… la compresión de datos puede provocar la pérdida de información? Si no se pierde información, podemos reconstruir la información original. Esto sucede, por ejemplo, al guardar una imagen como PNG. En caso contrario, para ahorrar espacio, sacrificamos ciertos datos en el camino; por ejemplo, al guardar una imagen como JPG degradamos la imagen, es decir, reducimos el espacio que ocupa, pero ya no podremos reconstruir la imagen original.

Secuencia Didáctica 3

REPRESENTACIÓN DE TEXTO

Todos los días usamos textos en nuestros dispositivos digitales. Utilizamos servicios de mensajería instantánea, hacemos comentarios en las redes sociales, leemos páginas web, entre muchas otras actividades que involucran palabras. ¿Cómo hacen las computadoras para guardar, transmitir y mostrar estos textos?

Esta secuencia didáctica consiste en dos actividades en las que se usan proyectos de Gobstones: la primera presenta la noción de codificación de caracteres mediante la asociación de letras con las bolitas que hay en una celda; la segunda explica el cifrado César, un antiguo método de encriptación para la transmisión de mensajes cifrados.

OBJETIVOS

- Comprender cómo se representa texto en una computadora.
- Presentar un método simple de cifrado de texto.

Actividad 1
Representamos caracteres

DE A DOS

OBJETIVO

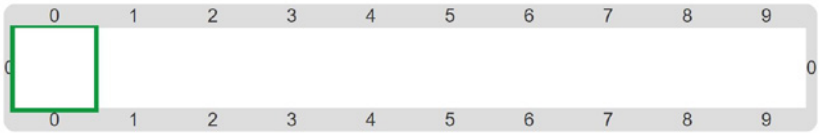
- Comprender cómo se representan los caracteres en la computadora.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

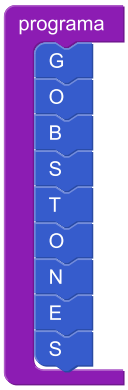
DESARROLLO

El objetivo de esta actividad es que los estudiantes comprendan que el texto puede codificarse con números. En este caso, trabajarán con el alfabeto latino y verán una forma de codificar cada letra.



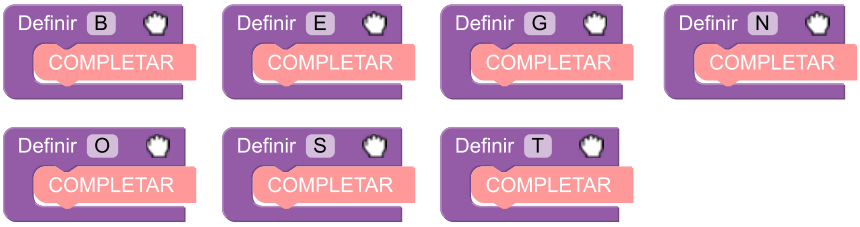
Tablero inicial

Comenzamos la actividad repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto "Representamos caracteres". Encontrarán un tablero unidimensional vacío de 10 celdas que va a hacer las veces de pizarra. Además, en el espacio del programa, observarán que el cuerpo principal ya está resuelto. Se trata de varias invocaciones a distintos procedimientos, uno a continuación del otro: `G`, `O`, `B`, `S`, `T`, `O`, `N`, `E` y `S`.



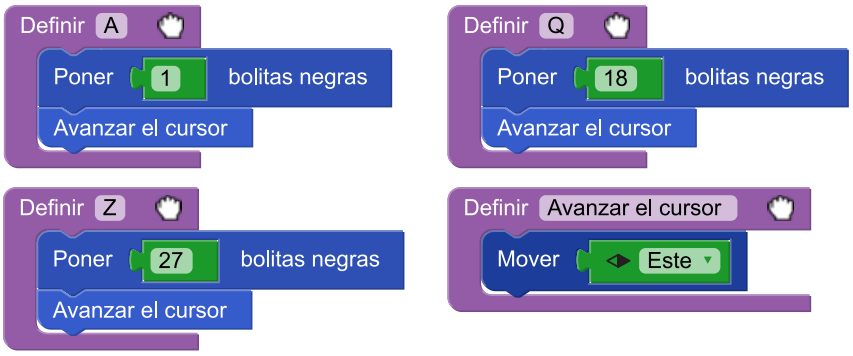
Cuerpo principal del programa

Les contamos que el objetivo del programa es escribir sobre el tablero la palabra *GOBSTONES*. A continuación, les pedimos que ejecuten el programa, que fallará porque ninguno de los procedimientos invocados está resuelto. Deben, entonces, completarlos.



Procedimientos no programados

Al observar el panel del programa, notarán que hay un procedimiento por cada letra del abecedario. A diferencia de los invocados desde el cuerpo principal del programa, los restantes ya están resueltos. Por ejemplo, el procedimiento `A` pone una bolita negra, `Q` pone 18 y `Z` 27, usando `Poner [ ] bolitas negras` (que está disponible en *Biblioteca* > *Procedimientos*). A continuación, todos ellos invocan `Avanzar el cursor`, que ya está programado y solo mueve el cabezal una posición hacia el este.



Procedimientos `A`, `Q`, `Z` y `Avanzar el cursor`

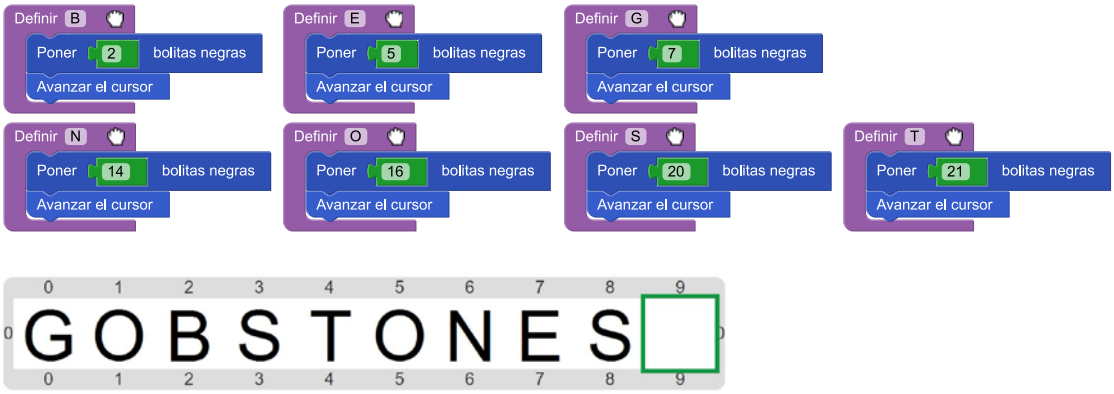
Sin nuestra ayuda, los estudiantes tienen que descubrir que cada procedimiento pone una cantidad de bolitas negras que corresponde a la posición que ocupa cada letra en el alfabeto. Por eso, `A` pone 1, `Q` 18 y `Z` 27. Además, la vestimenta asocia las distintas cantidades de bolitas negras con una imagen de la letra correspondiente. Por último, tal como se observa en el tablero inicial al desactivar la vestimenta, el espacio está representado por una celda sin bolitas.

	A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12	13

N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26	27

Letras del alfabeto latino y su representación numérica

Una vez que hayan reconocido cómo se codifican las letras con bolitas negras, no tendrán inconvenientes para completar el programa. Ahora sí, al ejecutarlo, sobre el tablero aparecerá la palabra buscada.



Procedimientos una vez completados y tablero final

Después de que hayan completado el programa, les proponemos que piensen distintas palabras y escriban los programas para que esas palabras aparezcan en el tablero. Deben tener cuidado de elegir palabras de 9 caracteres o menos, para evitar que el cabezal se caiga del tablero.

Luego de unos minutos les hacemos notar que, si bien en general es recomendable que los nombres de los procedimientos comiencen con un verbo, hacerlo como lo hicimos en este caso no altera la claridad y legibilidad del programa. Alternativamente, podrían haberse llamado `Escribir la letra A`, `Escribir la letra B`, etc.

CÓDIGO ASCII

El código ASCII es una codificación muy usada por las computadoras para representar los símbolos del alfabeto latino. Utiliza 8 bits para representar cada símbolo, con lo que puede codificar $2^8 = 256$ caracteres distintos. Hay otros alfabetos que están compuestos por muchísimos más símbolos –como el del chino mandarín, por ejemplo, que tiene más de 56.000–, por lo que este sistema de codificación no es apto para cualquier alfabeto. Para resolver este problema se crearon otras codificaciones que usan más bits para representar los caracteres; UNICODE es uno de los más difundidos.

CIERRE

Para concluir, reflexionamos con los estudiantes acerca de que el texto, como todo otro dato almacenado en una computadora –ya sean números, imágenes, etc.–, se representa con bits. No obstante, los programas rara vez se piensan en términos de bits, sino que se basan en otros elementos a su vez representados con bits, como son los números. Así, en esta actividad representamos caracteres con cantidades de bolitas que pueden estar entre 0 y 27. Estas cantidades, a su vez, la computadora las representa internamente con bits. Sin embargo, nosotros nos abstraemos de este hecho: al representar caracteres solo trabajamos con las bolitas, y no pensamos nunca en cuáles son los bits que se están usando para almacenarlos.

NOMBRE Y APELLIDO:

CURSO: FECHA:

REPRESENTAMOS CARACTERES

Todos los días usamos textos en nuestros dispositivos digitales. Utilizamos servicios de mensajería instantánea, hacemos comentarios en las redes sociales, leemos páginas web y muchas cosas más. ¿Cómo hacen las computadoras para guardar, transmitir y mostrar estos textos si solo trabajan con números?

1. Abrí el proyecto "Representamos caracteres" y ejecutá el programa. ¿Falló? ¡Ups! Arreglalo para que escriba la palabra *GOBSTONES*.

¿Cómo completaste los procedimientos?

_____

_____

_____

¡PRESTÁ ATENCIÓN!
Para completar los procedimientos que faltan, fijate en cómo están hechos los que ya están resueltos.

2. Elegí otras palabras y armá el programa para que aparezcan en el tablero. ¿Qué pasa con las palabras de más de nueve letras? ¿Por qué?

_____

_____

¿SABÍAS QUE...
… en las computadoras se utiliza un código llamado ASCII para representar las letras del alfabeto latino? Utiliza 8 bits para representar cada símbolo, con lo que puede codificar $2^8 = 256$ caracteres distintos. Hay otros alfabetos que están compuestos por muchísimos más símbolos –como el chino mandarín, por ejemplo, que tiene más de 56.000–, por lo que este sistema de codificación no es apto para cualquier alfabeto. Para resolver este problema se crearon otras codificaciones que usan más bits para representar los caracteres; UNICODE es uno de los más difundidos.

Actividad 2
Mensajes secretos

DE A DOS

OBJETIVO

- Presentar un método de cifrado.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

Mucho antes de la aparición de las primeras computadoras, la privacidad de la información ya era un problema esencial en distintos ámbitos. Hace más de 2000 años, el líder político y militar de Roma, Julio César, enviaba mensajes confidenciales a sus generales mediante cartas. Para evitar que los enemigos descubrieran sus secretos si capturaban a uno de los mensajeros, ideó un mecanismo para ocultar el significado del texto. Reemplazaba cada letra por otra siguiendo un número determinado. Si ese número era 1, cambiaba cada letra por la siguiente del alfabeto: la *A* por la *B*, la *B* por la *C*, y así hasta la *Z* por la *A*. Él ideó lo que hoy se conoce como métodos de cifrado de información. En esta actividad abordaremos uno de los métodos más conocidos, que supuestamente utilizaron las tropas romanas: el cifrado César.

Para comenzar, les preguntamos a los estudiantes si conocen alguna forma de encriptar o cifrar la información para que, al enviar un mensaje, solo el destinatario sea capaz de comprenderlo. Si no conocen ninguno, los invitamos a que se tomen unos minutos para pensar cómo lo harían. Es probable que sus propuestas consistan en métodos de cifrado simétrico, es decir, que usan la misma clave para cifrar y descifrar el mensaje.[1]

Si no surge en esta etapa, presentamos el cifrado César, que consiste en realizar un desplazamiento constante –también llamado *número del César*– de cada una de las letras de un mensaje. En su forma más simple, el desplazamiento es de una letra. De este modo, al escribir un mensaje, la letra *A* se reemplaza por una *B*, la *B* por una *C*, etc. La *Z*, que es la última letra del abecedario, se reemplaza por la *A*, la primera. Por ejemplo, la palabra *GOBSTONES* se escribiría *HPCTUPÑFT*. El receptor del mensaje, si sabe que se utilizó un desplazamiento de una letra, puede reconstruir el mensaje original reemplazando cada *B* por una *A*, cada *C* por un *B*, etc. La clave de este método de cifrado es el valor del desplazamiento.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
O	P	Q	R	S	T	U	V	W	X	Y	Z	A

GOBSTONES ⟷ HPCTUPÑFT

Desplazamiento de un carácter y cifrado y descifrado de la palabra *GOBSTONES*

[1] Los métodos de cifrado asimétrico usan una clave para cifrar y otra distinta para descifrar. La criptografía moderna se basa en esta clase de métodos.

Podría usarse un desplazamiento diferente, como por ejemplo de 3 letras; en ese caso, *GOBSTONES*, cifrado, es *JREVWRPHV.*

A	B	C	D	E	F	G	H	I	J	K	L	M	N
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

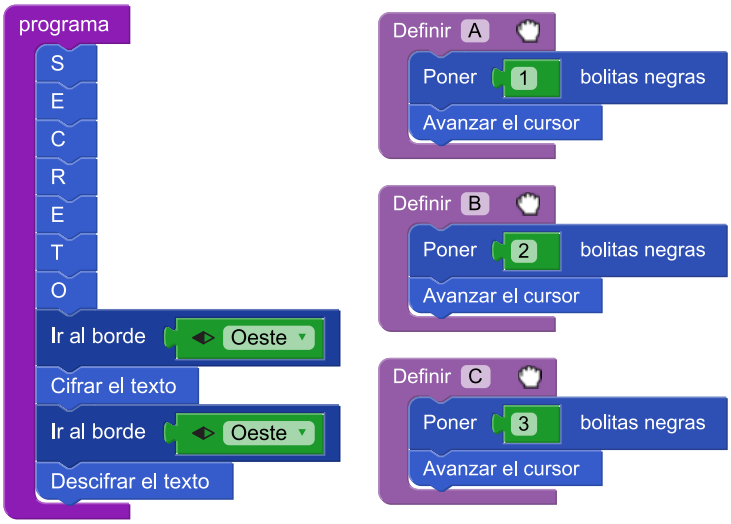
GOBSTONES ⟷ JREVWRPHV

Desplazamiento de tres caracteres y cifrado y descifrado de la palabra *GOBSTONES*

Repartimos las fichas a los estudiantes y les pedimos que abran el proyecto de Gobstones "Mensajes secretos". El objetivo es que programen procedimientos para cifrar y descifrar mensajes usando el método César.

En el panel del programa observarán que tanto los procedimientos para codificar letras y mover el cursor, como el cuerpo principal del programa, ya están resueltos. Los procedimientos `A`, `B`, `C`, etc. colocan una cantidad de bolitas negras que corresponde a la posición de la letra en el abecedario.

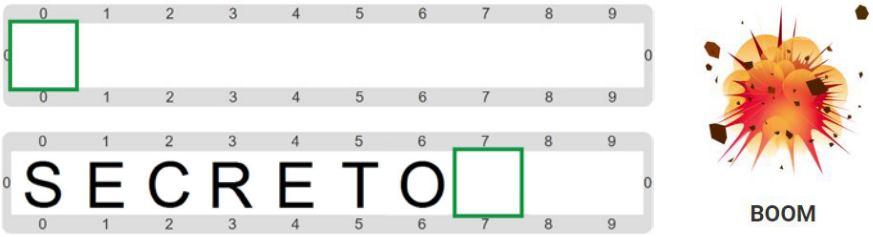
El programa principal, por su parte, invoca los procedimientos `S`, `E`, `C`, `R`, `E`, `T`, `O`, `Cifrar el texto` y `Descifrar el texto`, y el comando básico `Ir al borde [ ]`.



Cuerpo principal del programa y procedimientos `A`, `B` y `C`

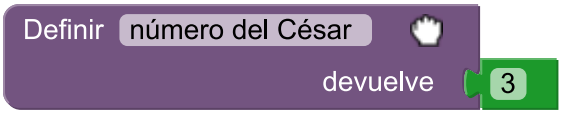
Les indicamos que ejecuten el programa. Verán que, partiendo de un tablero vacío, primero se escribirá la palabra *SECRETO*, y luego el tablero explotará debido a que los procedimientos `Cifrar el texto` y `Descifrar el texto`, no están completos.

Procedimientos `Cifrar el texto` y `Descifrar el texto`, inicialmente vacíos



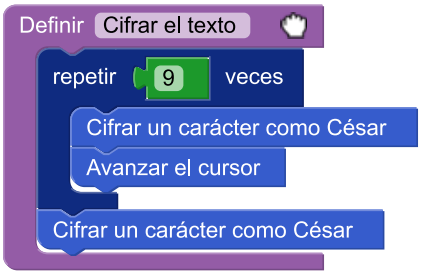
Ejecución del programa tal como viene dado

En primer lugar tienen que decidir el número del César que utilizarán para cifrar los mensajes. El proyecto provee la función `número del César` que lo establece en 3, pero los estudiantes pueden cambiarlo si así lo desean. A modo de ejemplo, continuaremos con el valor 3 en lo que resta de la explicación de la actividad.



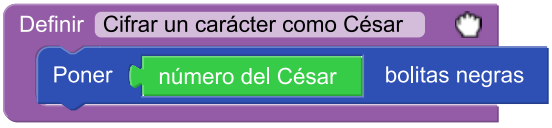
Función `número del César`

Para completar `Cifrar el texto` hay que cifrar los caracteres que aparecen en cada una de las celdas del tablero, para lo que se pueden usar los procedimientos `Cifrar un carácter como César` –que está dado, pero hay que completar– y `Avanzar el cursor` –que ya está resuelto–.



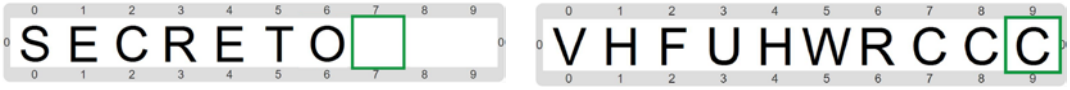
Procedimiento `Cifrar el texto`

Es probable que, como una primera aproximación, los estudiantes propongan completar `Cifrar un carácter como César` agregando tantas bolitas negras como el número del César elegido, usando `Poner [ ] bolitas negras`, disponible en *Biblioteca > Procedimientos*.



Posible propuesta para `Cifrar un carácter como César`

Si ejecutan el programa notarán que, si bien la palabra *SECRETO* se cifra correctamente, en las últimas 3 celdas del tablero aparece una *C*. El cifrado César reemplaza cada letra por otra, pero no debería tener efecto sobre las celdas vacías.

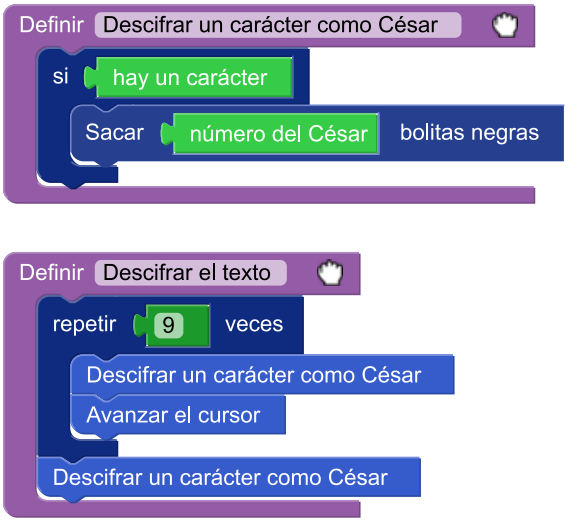


Efecto de la ejecución de una versión errónea de `Cifrar el texto`

En el proyecto ya está resuelta la función `hay un carácter`, que sirve para determinar si hay una letra en una celda chequeando la presencia (o ausencia) de bolitas negras. La usamos, entonces, para corregir `Cifrar un carácter como César`. Al hacerlo de este modo, no se agregan bolitas en celdas vacías.

El programa no realiza cambios en celdas que tienen un espacio en blanco

Es probable que, para descifrar el texto y recuperar el texto original, los alumnos propongan realizar el proceso inverso: sacar tres bolitas de cada celda en la que haya una letra. Para hacerlo, hay que completar los procedimientos `Descifrar el texto` y `Descifrar un carácter como César`; para completar este último, es necesario usar `Sacar [ ] bolitas negras`, disponible en *Biblioteca > Procedimientos.*

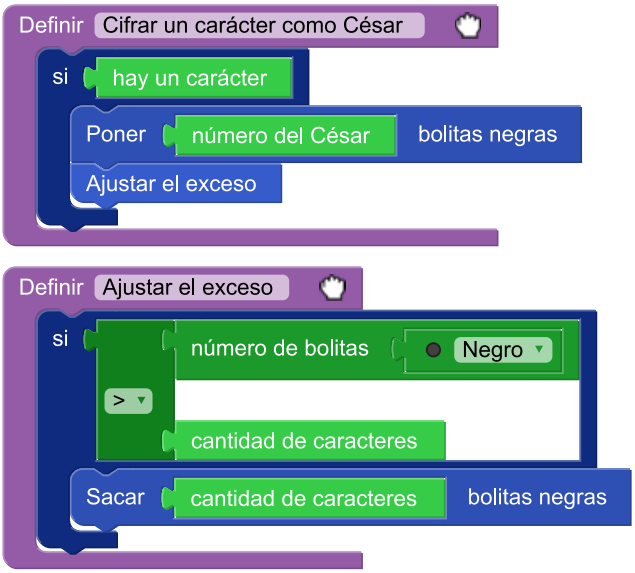


Procedimientos `Descifrar el texto` y `Descifrar un carácter como César`

A continuación, les indicamos que cambien el cuerpo principal del programa de forma tal que la palabra a cifrar y descifrar sea *CARNAZA*, y que ejecuten el programa. Verán entonces que al cifrar la letra *Z* el programa, en lugar de reemplazarla por la *C*, muestra 30 bolitas negras.

No se cifra correctamente la palabra *CARNAZA*

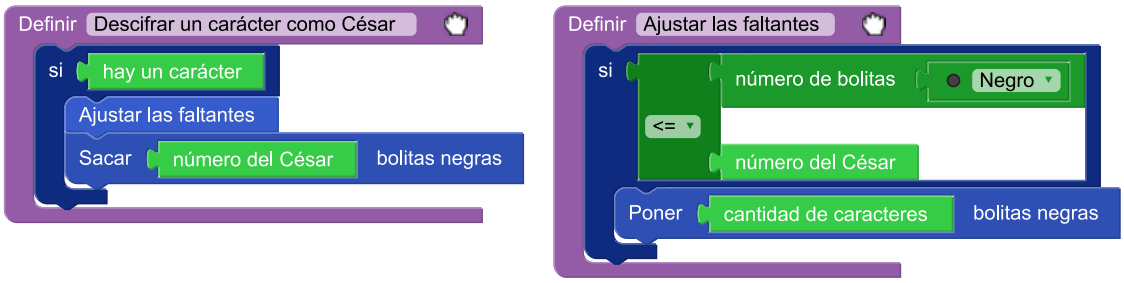
De acuerdo con el cifrado del César, en la celda de la *Z* debería aparecer una *C*, que se representa con 3 bolitas negras y no con 30, que no codifica ninguna letra. Sin embargo, en el procedimiento `Cifrar un carácter como César`, en caso de que nos encontremos con un carácter, siempre estamos agregando bolitas, aun cuando en ciertas ocasiones debemos sacar bolitas.[1] De algún modo, cada vez que lleguemos a tener 27 bolitas negras, para seguir avanzando hay que volver a contar desde 1. Es lo que se conoce como **ajustar el exceso**, y puede hacerse restando 27 a todos aquellos valores mayores a este número. En nuestro ejemplo, 30 - 27 = 3. El proyecto trae un procedimiento ya resuelto que sirve para este propósito: `Ajustar el exceso`.



Procedimientos `Cifrar un carácter como César` y `Ajustar el exceso`

[1] Al usar 3 como número del César, algo similar sucede con las letras *X* –que se reemplaza por una *A*– e *Y* –que se reemplaza por una *B*–. En el caso general, al usar *n* desplazamientos, esto ocurrirá con las últimas *n* letras del abecedario.

Del mismo modo, al descifrar cualquiera de las tres primeras letras –*A*, *B* o *C*–, hay que tener cuidado de realizar lo que se conoce como **ajustar las faltantes**. Es decir, antes de retirar bolitas, en primer lugar sumar 27. Siguiendo con el ejemplo de *CARNAZA*, , que cifrada es *FDUPDCD*, al descifrar la *C* -que se representa con 3 bolitas negras-, se obtiene (3 + 27) - 3 = 30 - 3 = 27, lo que permite restablecer la letra *Z*. Si esto no se hace, en la celda no quedarán bolitas, y no se mostrará ninguna letra.[1] En el proyecto ya está resuelto el procedimiento `Ajustar las faltantes` que utilizamos para corregir `Descifrar un carácter como César`. De este modo, completamos el programa.



Procedimientos `Descifrar un carácter como César` y `Ajustar las faltantes`

Una vez que los estudiantes concluyan sus programas, relacionamos la idea de cifrado con la de representación de la información: toda forma de cifrado es una manera de representar información. La manera de codificar datos, ya se trate de texto, números o imágenes, es arbitraria y cada programador puede elegir la que más le convenga según su propósito. Sin embargo, como vimos en esta actividad, cuando se trabaja con este tipo de mensajes es necesario que la persona que debe leerlos conozca con qué método se cifraron para poder descifrarlos. Esto, en realidad, es así en todas las formas de codificación de la información: para poder comprender la información es necesario saber cómo se codificó. Por eso, tener codificaciones estandarizadas permite que diversos programas puedan interpretar los mismos datos. Así, es posible encontrar distintos programas que abren los mismos archivos de texto, imágenes, música y video, entre otros.

CIERRE

Para concluir la actividad, mencionamos que los métodos de cifrado se utilizan para proteger la privacidad de los datos. En esencia, plantean una representación alternativa de la información que solo puede ser interpretada correctamente por el destinatario deseado, puesto que está basada en codificaciones en las que algún dato no se conoce, ya sea el método de cifrado o la clave. Por ejemplo, gran parte de la información privada que enviamos por Internet (como números de tarjetas de crédito, contraseñas, etc.) viaja en forma cifrada para evitar que otras personas, si la interceptan, puedan comprenderla.

[1] Más aún, si en lugar de la *C*, la letra una vez cifrada fuese *A* o *B* -representadas por 1 y 2 bolitas negras respectivamente- se intentaría sacar más bolitas de las que hay en la celda bajo el cabezal, lo que producirá una falla durante la ejecución del programa.

NOMBRE Y APELLIDO:

CURSO: FECHA:

MENSAJES SECRETOS

Corría el año 50 a. C. Julio César, líder político y militar de Roma, enviaba mensajes secretos a sus generales mediante cartas. Para evitar que sus enemigos descubrieran sus secretos si capturaban a uno de los mensajeros, ideó un mecanismo para ocultar el significado del texto. Reemplazaba cada letra por otra a partir de un número que se llamó "el número del César". Si ese número era 1, reemplazaba cada letra por la inmediatamente posterior en el alfabeto: la ***A*** por la ***B***, la ***B*** por la ***C***, y así hasta llegar a la ***Z,*** que reemplazaba por la ***A***:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
O	P	Q	R	S	T	U	V	W	X	Y	Z	A

De este modo, la palabra SECRETO se transformaba en *TFDSFUP*. Tomando 3 como número del César, se reemplazaba cada letra por la que estaba tres lugares más adelante: la *A* por la *D*, la *B* por la *E*,..., la *W* por la *Z*, la *X* por la *A*, la *Y* por la *B* y la *Z* por la *C*. En este caso, *SECRETO* se transformaba en *VHFUHWR*.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Para poder comunicarse, él y sus generales se ponían de acuerdo en el número del César. Al enviar un mensaje lo transformaban de este modo y el destinatario, que conocía este número, reconstruía el mensaje original.

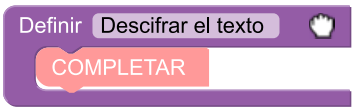
NOMBRE Y APELLIDO:

CURSO: FECHA:

1. Abrí el proyecto "Mensajes secretos", elegí un número del César y completá estos dos procedimientos:

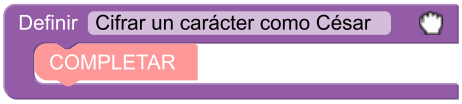
`Cifrar el texto`: cifra un texto de a lo sumo 10 caracteres usando el cifrado de César con el número elegido.

`Descifrar el texto`: descrifra un texto de a lo sumo 10 caracteres usando el mecanismo inverso al cifrado de César. Tenés que usar el mismo número del César que usaste al cifrar el mensaje.

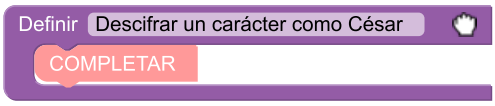


Para completarlos exitosamente, también tenés que programar estos:

`Cifrar un carácter como César`: cifra una letra usando el número elegido.



`Descifrar un carácter como César`: descifra una letra usando el número elegido.



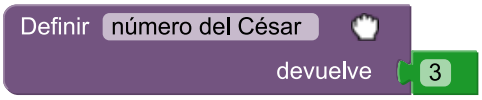
PARA TENER EN CUENTA

- Para cambiar el número del César, modificá el valor que arroja la función `número del César`, que por defecto es 3.
- La codificación de letras ya está dada: la *A* se representa con una bolita negra, la *B* con dos, etc.

PARA AJUSTAR EL EXCESO Y LAS BOLITAS FALTANTES

Para no pasarnos más allá de 27 bolitas negras o ir por debajo de 1, el proyecto trae resueltos dos procedimientos que podés usar: `Ajustar el exceso` y `Ajustar las faltantes`.

Al cifrar una letra, hay veces que tenés que empezar a contar desde 1 otra vez. Por ejemplo, si el número del César es 3, tenés que hacer estos reemplazos:



LETRA ORIGINAL	REPRESENTACIÓN	TRANSFORMACIÓN	LETRA CIFRADA
A	1	4	D
B	2	5	E
...			
W	24	27	Z
X	25	1	A
Y	26	2	B
Z	27	3	C

Además, al descifrar, tenés que hacer el proceso inverso.

Texto en colores

DE A DOS

OBJETIVO

- Comprender cómo codificar un texto con atributos.

MATERIALES

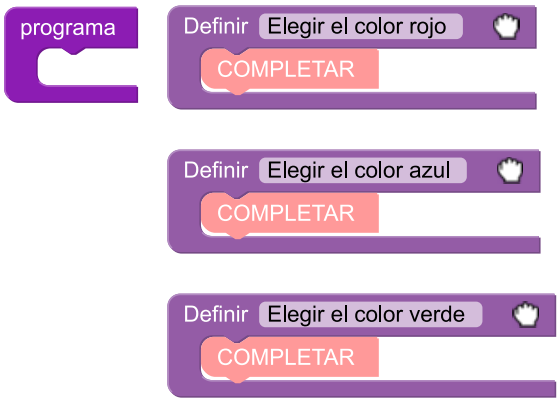
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En muchas aplicaciones, como procesadores de texto, sistemas de correo electrónico, mensajería, etc., se le brinda al usuario la posibilidad de personalizar el texto que escribe: ponerle un color, cambiarle la fuente o el tamaño, subrayar, tachar, poner en negrita, etc. A estas características se las conoce como **atributos del texto** y es necesario codificarlas para luego poder mostrarlas en la pantalla. En esta actividad, veremos cómo se puede codificar un texto con letras en colores.

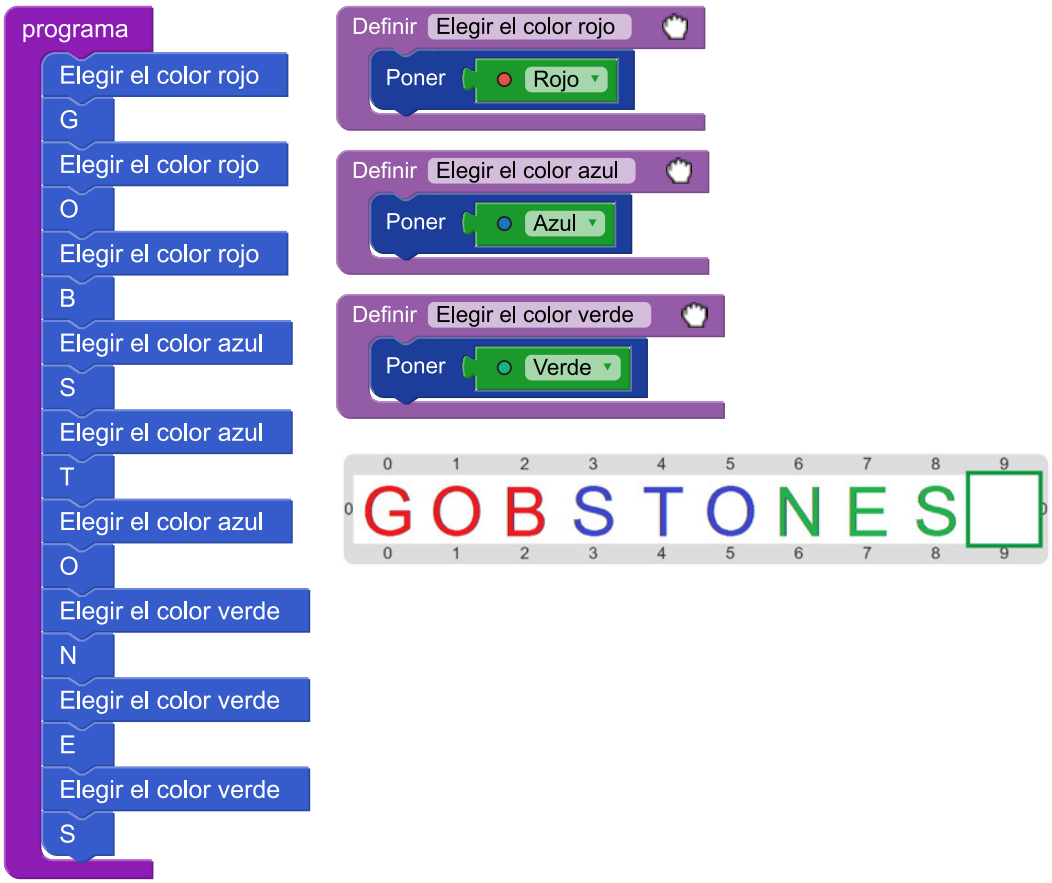
Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Texto en colores". El objetivo de la actividad es lograr escribir un texto en el que se pueda cambiar el color de cada letra. Para ello deben utilizar una codificación que contenga: *(i)* la codificación de letras usando bolitas negras y *(ii)* la codificación del color de cada letra. Las opciones de colores para las letras son negro, rojo –codificado por una bolita roja–, verde –codificado por una bolita verde– y azul –codificado por una bolita azul–. De este modo, una celda con una bolita negra y una roja codifica una *A* roja, una con tres negras y una verde una *C* verde, una con 26 negras y una azul, una *Y* azul, y una con 27 negras, una *Z* negra.

En el panel del programa, ya se encuentra implementada la mayoría de los procedimientos necesarios como, por ejemplo, `Avanzar el cursor`, `Espacio` y los que sirven para escribir cada una de las 27 letras. Lo que falta programar es el cuerpo principal del programa y los procedimientos que colorean cada carácter: `Elegir el color rojo`, `Elegir el color verde` y `Elegir el color azul`.



Procedimientos a completar

Les proponemos, entonces, que seleccionen una palabra para escribir en el tablero eligiendo el color de cada letra. Deben tener presente que, de no indicar el color, la letra se visualizará en negro (pues no se estará agregando una bolita que indique alguno de los otros colores disponibles).



Gobstones en colores

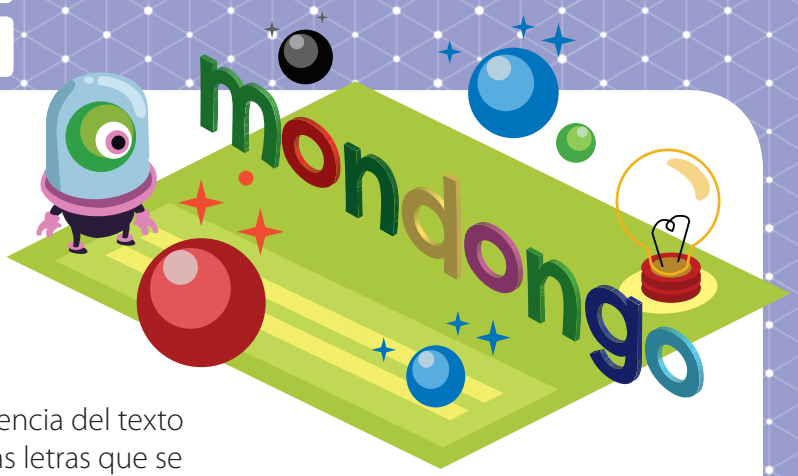
CIERRE

Para concluir esta actividad, reflexionamos sobre otros atributos posibles para el formato de los caracteres. Si los estudiantes han utilizado alguna vez un procesador de textos, seguramente podrán mencionar algunos de ellos, como, por ejemplo, texto en negrita, itálica y subrayado, color de fondo, tipo de letra, tamaño, etc. Se puede llegar a la conclusión de que, para poder almacenar estas características, es necesario utilizar una codificación apropiada que permita representarlas. Además, cada uno de estos atributos es información adicional que debe almacenarse junto con el contenido del texto. Por este motivo, los archivos que almacenan características de formato son habitualmente de mayor tamaño que los de texto plano, es decir, los que contienen únicamente los caracteres.

NOMBRE Y APELLIDO:

CURSO: FECHA:

TEXTO EN COLORES

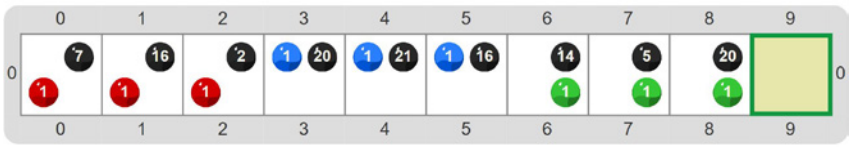


El texto con estilo o texto enriquecido, a diferencia del texto plano, tiene información que va más allá de las letras que se imprimen. Podemos indicar colores, **negrita**, *cursiva* y taMañoS, entre otras características a las que llamamos atributos. En esta actividad, vamos a agregarles color a los textos en Gobstones.

1. Abrí el proyecto "Texto en colores". Tenés que completar el programa para poder incluir colores en los textos.

Los procedimientos para codificar letras ya están resueltos. ¡Es tu tarea darles color! Mirá el resultado de ejecutar este programa, con y sin vestimenta.

```
programa
  Elegir el color rojo
  G
  Elegir el color rojo
  O
  Elegir el color rojo
  B
  Elegir el color azul
  S
  Elegir el color azul
  T
  Elegir el color azul
  O
  Elegir el color verde
  N
  Elegir el color verde
  E
  Elegir el color verde
  S
```



CODIFICACIÓN DE LETRAS A TODO COLOR

Codificamos las letras con la cantidad de bolitas negras que corresponde a su posición en el alfabeto español. Los colores de las letras –rojo, verde y azul– se codifican agregando una bolita del color correspondiente.

06

PARÁMETROS, REPETICIÓN CONDICIONAL Y VARIABLES

Muchos de los comandos tienen un "agujero" que debe completarse con un valor al invocarlos, como *Poner [ ]*, que requiere un color. A este "agujero" se lo denomina **parámetro**, y al valor que se usa en una invocación, **argumento**. La primera secuencia didáctica se centra en la noción de parámetro.

La segunda secuencia aborda la repetición condicional, que, a diferencia de la repetición simple, permite efectuar repeticiones hasta que se cumpla una cierta condición en lugar de una cantidad de veces conocida *a priori*.

Por último, en muchas ocasiones hace falta recordar y modificar valores a lo largo de un programa; por ejemplo, al programar un juego, hay que llevar registro de los puntos alcanzados por un jugador. La tercera y última secuencia didáctica presenta las variables, que permiten almacenar valores en la memoria de la computadora, leerlos y modificarlos.

Secuencia Didáctica 1

PROCEDIMIENTOS CON PARÁMETROS

Muchas veces los procedimientos que escribimos son extremadamente parecidos y solo difieren en algún valor. Un **parámetro** es un "agujero" que se deja en un procedimiento y al usarlo debe completarse con un valor o **argumento**. Por ejemplo, al usar el comando `Poner [ ]` debemos indicar un valor de tipo color (por ejemplo `Rojo`). Desde ya, un procedimiento puede tener varios parámetros.

A lo largo de esta secuencia didáctica se presentarán los motivos por los cuales se utilizan procedimientos con parámetros y en qué casos conviene hacerlo, y se practicará cómo utilizarlos en programas de Gobstones de complejidad incremental.

OBJETIVOS

- Comprender las nociones de parámetro y argumento.
- Utilizar procedimientos con distintas cantidades de parámetros en un programa.

Actividad 1
Un elefante se balanceaba

TODA LA CLASE

OBJETIVO

- Presentar las nociones de parámetro y argumento.

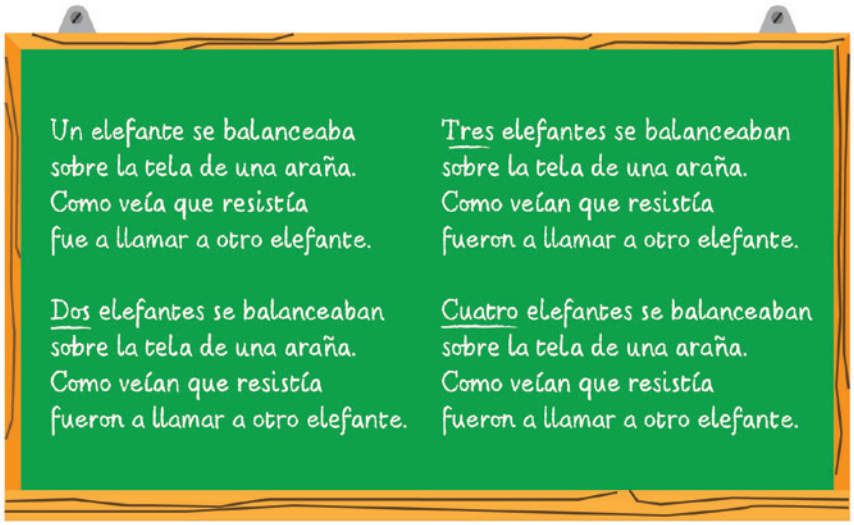
MATERIALES

- Papel
- Lápiz
- Reproductor de audio

DESARROLLO

El objetivo de esta actividad es presentar las nociones de **parámetro** y **argumento**. Comenzamos escuchando con toda la clase la canción *Un elefante se balanceaba*[1] y les pedimos a los estudiantes que transcriban en un papel las cuatro primeras estrofas de la manera más concisa posible.

Después de que ensayen posibles escrituras, hacemos una puesta en común. Preguntamos: "¿Cómo transcribieron la canción? ¿Escribieron cada estrofa por separado?". Si bien son muy parecidas, las distintas estrofas no son iguales. En cada una, la cantidad de elefantes que se balancean es distinta. Escribimos la canción en el pizarrón y subrayamos las diferencias de la segunda estrofa en adelante.[2]

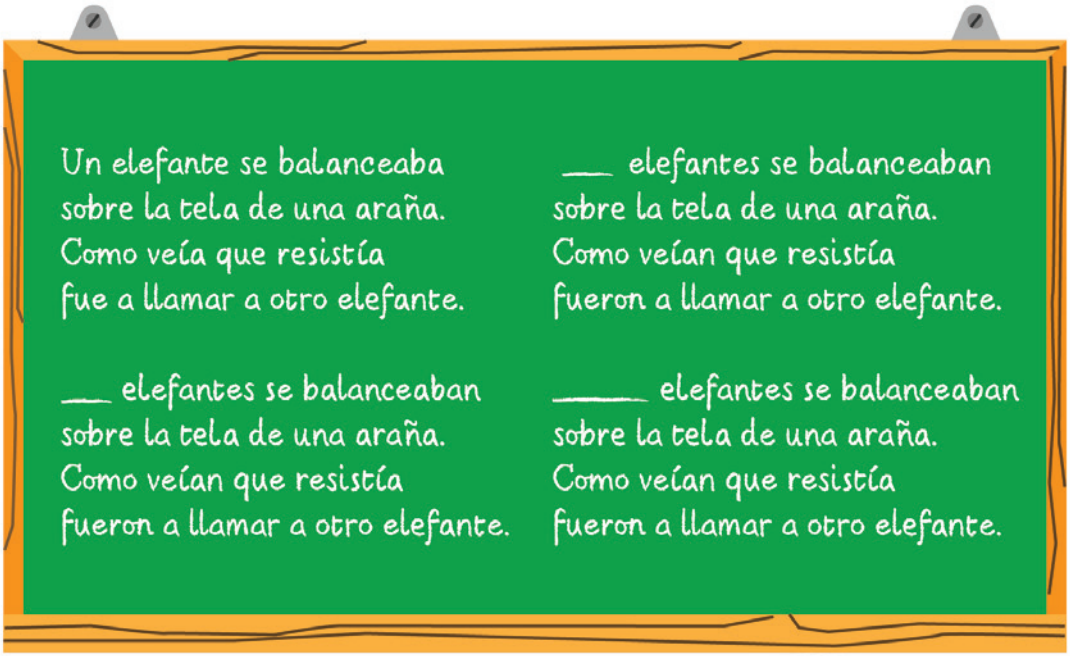


Estrofas de *Un elefante se balanceaba*

[1] Un videoclip animado de la canción puede verse en https://goo.gl/MhkcGL.

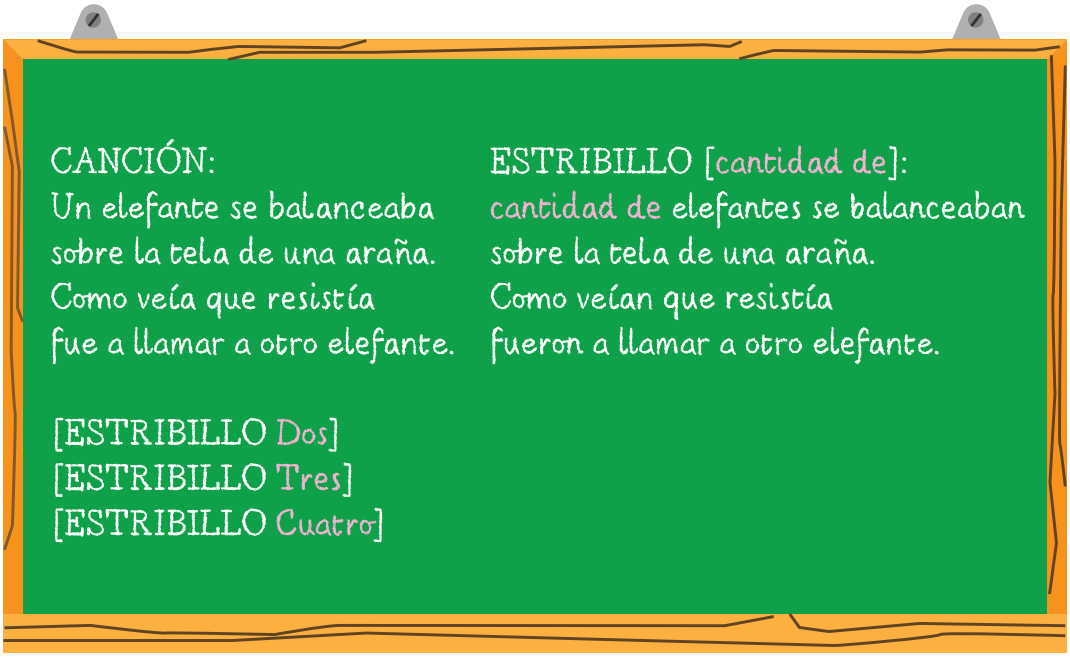
[2] Se deja fuera de la comparación entre estrofas la primera porque es la única con el sujeto y el predicado en singular (*elefante* en vez de *elefantes*, por ejemplo).

A continuación, borramos las palabras subrayadas para que noten que, sin ellas, las estrofas son exactamente iguales.



Segunda, tercera y cuarta estrofa de la canción sin las cantidades de elefantes

Luego, dejando de lado la primera estrofa, hacemos notar que, al eliminar la mención al número de elefantes, las estrofas siguientes funcionan como una suerte de estribillo. Por lo tanto, pueden quitarse dos de estas estrofas y transformar la restante en un estribillo con un "agujero" o "hueco" en el lugar donde aparecía antes el número de elefantes. A este agujero le ponemos un nombre que describa cuál es el dato faltante; en este caso podría llamarse, *cantidad de*. De este modo, queda definido un estribillo que admite distintas cantidades de elefantes. Por último, escribimos la canción, remitiendo el estribillo así formulado y completando el agujero con el valor que corresponda en cada caso. Señalamos que el agujero se denomina **parámetro** y el valor usado, **argumento**. En la canción, *cantidad de* es el nombre del parámetro y *Dos*, *Tres* y *Cuatro*, argumentos.



Otra forma de escribir la canción

CIERRE

Reflexionamos con los estudiantes sobre la utilidad de los parámetros y los argumentos. En este caso, nos permitieron expresar la variación entre versos muy similares dejando un agujero (el parámetro), que luego completamos con un valor específico (el argumento). Al parámetro le pusimos un nombre descriptivo (*cantidad de*) que se reemplaza por el valor del argumento correspondiente cada vez que se invoca el estribillo. En el ejemplo, *cantidad de* se reemplaza por *Dos*, *Tres* y *Cuatro*. De este modo, el uso de parámetros y argumentos nos permitió escribir la canción de manera concisa.

Actividad 2
Cuadrados de colores

DE A DOS

OBJETIVOS

- Comprender la conveniencia de usar parámetros al programar.
- Definir y utilizar procedimientos con parámetros.

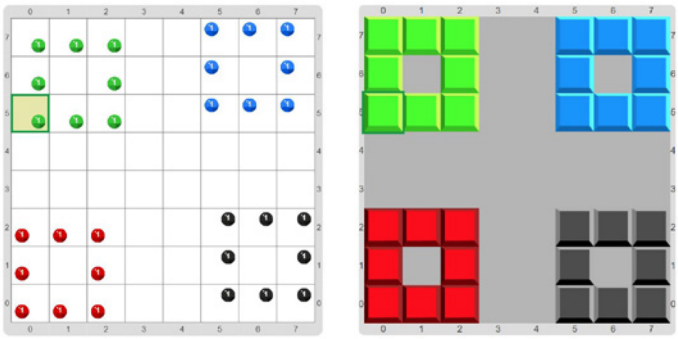
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

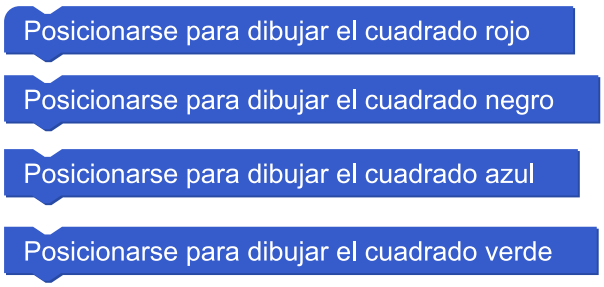
Esta actividad tiene como objetivo poner de manifiesto la utilidad e importancia del uso de parámetros en programación. En ejercicios previos, los estudiantes ya usaron comandos con un parámetro, como `Poner [ ]`, `Mover [ ]` e `Ir al borde [ ]`. Cada uno tiene un parámetro que adquiere un valor, como `Rojo` o `Este`, al ser invocados desde un programa. En esta actividad, veremos cómo definir procedimientos que tengan valores parametrizados.

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto "Cuadrados de colores , toma 1". Encontrarán un tablero vacío de 8 × 8. El objetivo es dibujar cuatro cuadrados de 3 × 3, cada uno de un color distinto: rojo en la esquina suroeste, negro en la esquina sureste, azul en la esquina noreste y verde en la esquina noroeste.



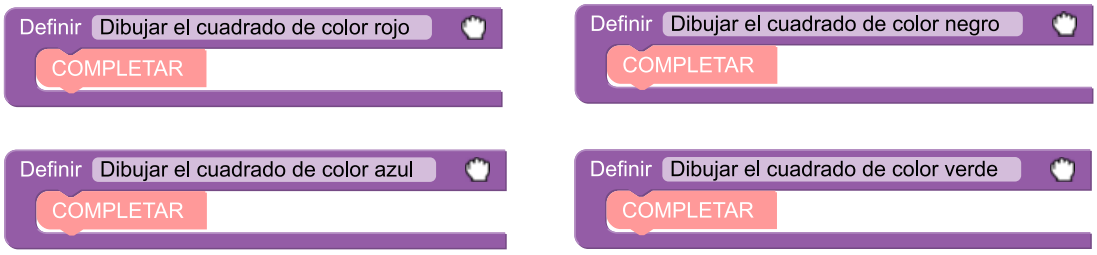
Tablero final sin y con vestimenta

En el panel del programa verán 8 procedimientos: 4 ya resueltos y 4 que hay que completar. Los procedimientos resueltos son los que posicionan al cabezal en el lugar apropiado para dibujar cada cuadrado: `Posicionarse para dibujar el cuadrado rojo`, `Posicionarse para dibujar el cuadrado negro`, `Posicionarse para dibujar el cuadrado azul` y `Posicionarse para dibujar el cuadrado verde`.



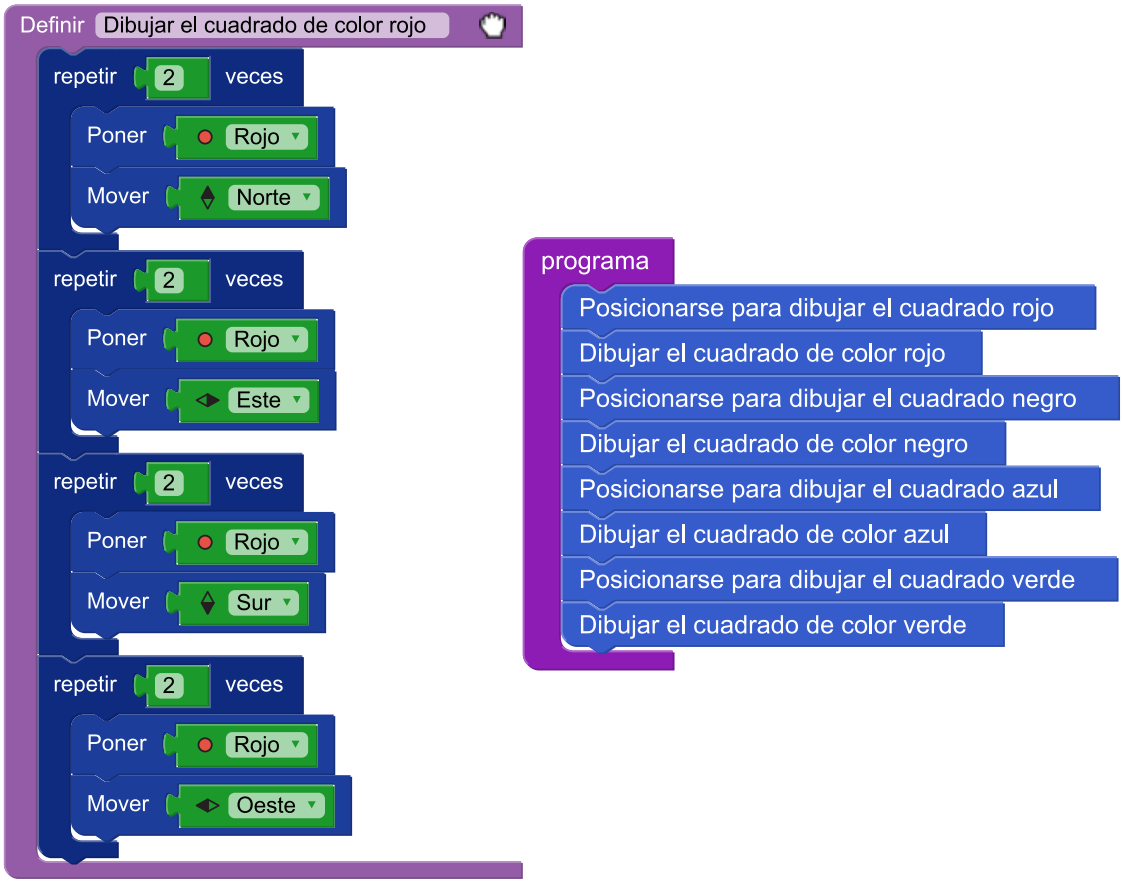
Procedimientos del proyecto ya resueltos

Los procedimientos que falta completar son los que dibujan los cuadrados: `Dibujar el cuadrado de color rojo`, `Dibujar el cuadrado de color negro`, `Dibujar el cuadrado de color azul` y `Dibujar el cuadrado de color verde`.



Procedimientos del proyecto que deben completarse

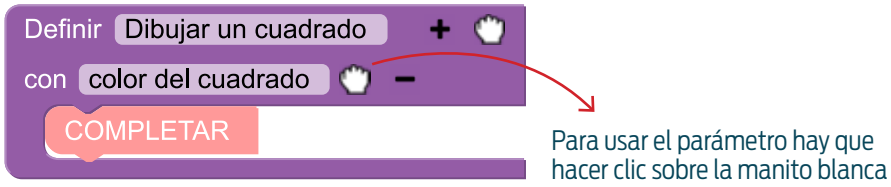
Mientras los estudiantes resuelven la actividad, les recordamos la conveniencia de usar el comando `repetir [ ] veces`. En la figura se observan posibles soluciones para el cuerpo principal del programa y para uno de los procedimientos que dibujan un cuadrado.



Procedimiento `Dibujar el cuadrado de color rojo` y cuerpo principal del programa

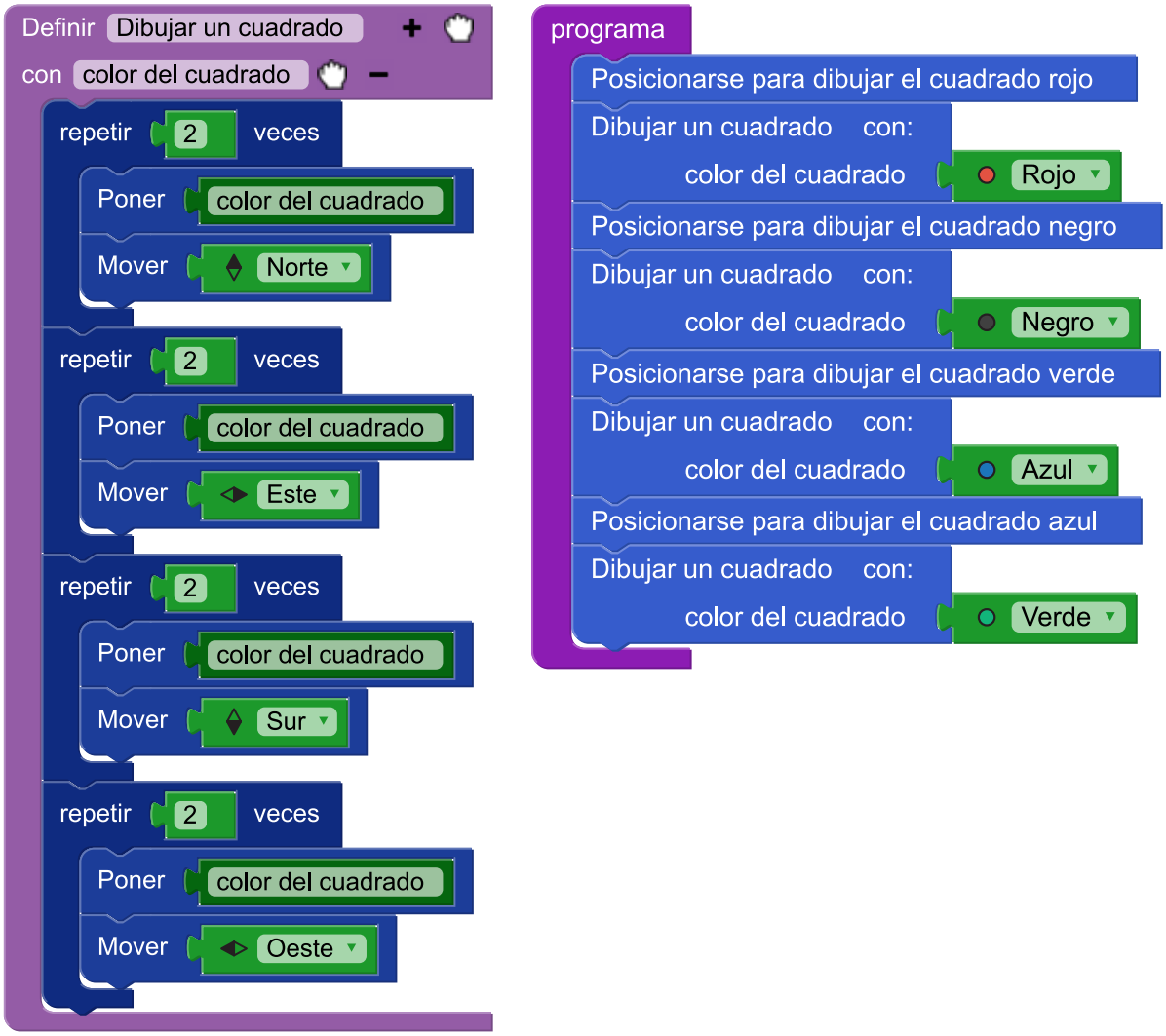
Probablemente, los estudiantes completen de forma casi igual los cuatro procedimientos que dibujan un cuadrado, variando únicamente el color de las bolitas que depositan sobre las celdas. Preguntamos: "¿Se les ocurre cómo hacer para que el programa resulte más conciso?". ¡Con un parámetro! Si los estudiantes advierten esto antes de concluir los cuatro procedimientos, no es necesario esperar a que terminen de escribirlos: directamente se puede continuar con lo que resta de la actividad.

A continuación, los invitamos a abrir el proyecto "Cuadrados de colores, toma 2". Tanto el tablero como los procedimientos ya resueltos son los mismos que los de la toma 1. Sin embargo, en lugar de tener 4 procedimientos vacíos para dibujar cada uno de los cuadrados, en este caso hay solamente uno, `Dibujar un cuadrado [ ]`, con un parámetro, `color del cuadrado`.



Procedimiento con un parámetro

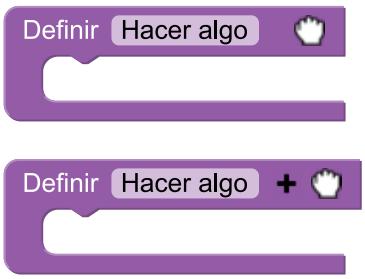
Para completar `Dibujar un cuadrado [ ]` podemos utilizar la misma estrategia que en el programa anterior, pero reemplazando el color específico por el parámetro `color del cuadrado` todos los usos del comando `Poner [ ]`. En el programa principal, cuando invoquemos `Dibujar un cuadrado [ ]`, debemos darle un valor concreto como argumento: `Rojo`, `Negro`, `Azul` y `Verde` respectivamente.



Posible solución para el proyecto "Cuadrados de colores, toma 2"

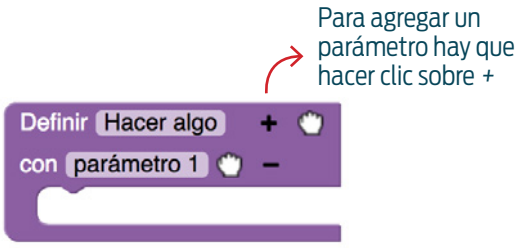
Una vez que los estudiantes completen sus programas, les preguntamos: "¿Qué desventaja tiene la estrategia empleada al comienzo, en el primer proyecto? ¿Y, si en vez de haber 4 colores en Gobstones, hubiera mil?". Los parámetros —se hará notar— nos dan la posibilidad de generalizar un comportamiento: en lugar de crear un procedimiento para cada color, hicimos un único procedimiento que abstrae la idea de color. Si tuviéramos mil colores y no contáramos con parámetros, deberíamos crear mil procedimientos.

A continuación, indicamos: "Observen el procedimiento `Dibujar un cuadrado [ ]`. ¿Notan alguna instrucción que aparezca muchas veces?". Efectivamente, para dibujar los lados del cuadrado, se utiliza cuatro veces `repetir [2] veces`; en el cuerpo de cada una de esas cuatro veces, solo cambia la dirección en la que se desplaza el cabezal. "¿Qué se puede hacer para seguir simplificando el programa?", preguntamos entonces. Se puede crear un nuevo procedimiento que dibuje un lado en el que la dirección sea un parámetro. En *Definiciones* > *Definición de procedimientos* encontraremos dos bloques: uno para crear procedimientos sin parámetros y otro para crear procedimientos con parámetros, que se distingue del primero por tener el signo + entre *Hacer algo* y la manito blanca.



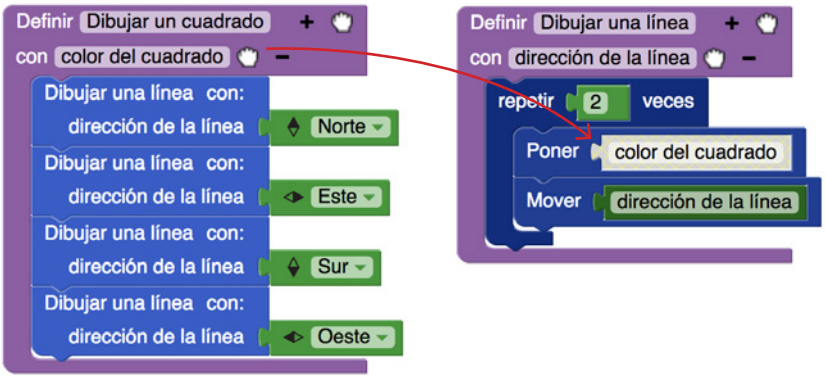
Bloques para crear procedimientos

Para agregar un parámetro hay que hacer clic sobre +. Por defecto, el nombre del parámetro es `parámetro 1`. Al igual que con los nombres de los procedimientos y las funciones, conviene cambiar este por uno que describa en forma clara su propósito.



Procedimiento con un parámetro

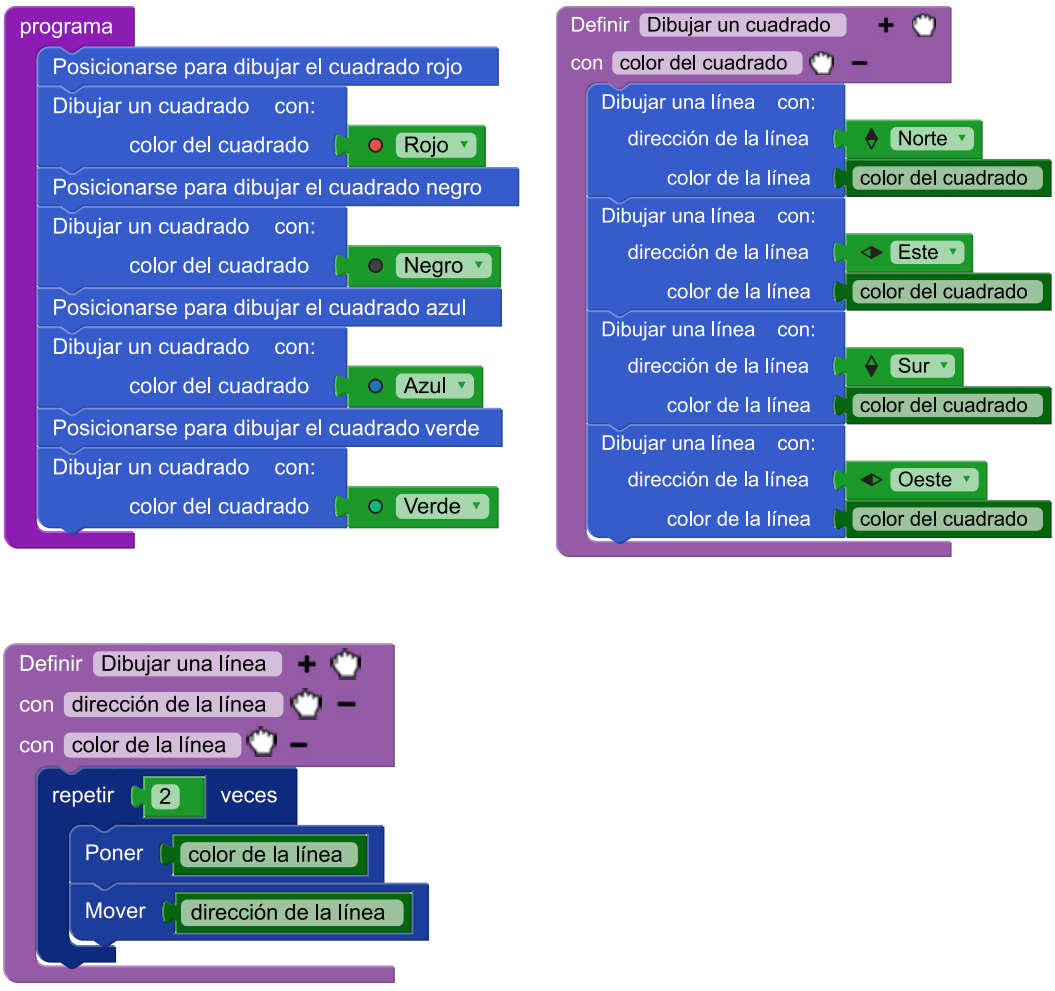
En este caso, un nombre adecuado para el procedimiento podría ser `Dibujar una línea`, y para el parámetro que indica la dirección, `dirección de la línea`. Ahora bien, ¿cómo indicar el color de la bolita que hay que poner en las celdas? Podría ocurrir que algún estudiante proponga usar el parámetro `color del cuadrado` del procedimiento `Dibujar un cuadrado [ ]`. En tal caso, al encastrarlo, notarán que el bloque aparece en gris, lo que indica que algo no está bien.



Un parámetro de un procedimiento no puede usarse en otro procedimiento

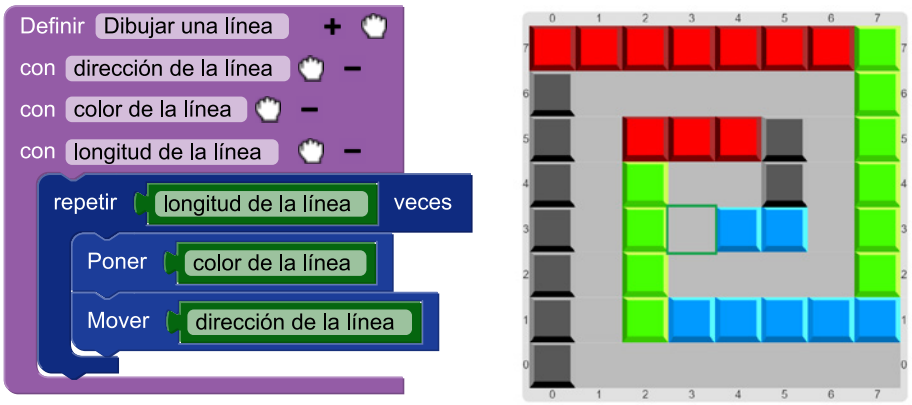
Como en casi todos los lenguajes de programación, el alcance de un parámetro está limitado al procedimiento en el que ese parámetro está declarado. Por lo tanto, en este caso, `color del cuadrado` solo puede usarse dentro del cuerpo de `Dibujar un cuadrado [ ]`.

Para corregir el programa, hay que agregar un nuevo parámetro a `Dibujar una línea [ ] [ ]`. Este nuevo parámetro se puede llamar `color de la línea`. Como el color del cuadrado coincide con el color de sus líneas, al invocar `Dibujar una línea [ ] [ ]` desde `Dibujar un cuadrado [ ]` hay que usar como argumento el parámetro `color del cuadrado`. Con esto completamos el programa. Cuando los estudiantes lleguen a esta instancia, les decimos: "El parámetro de un procedimiento puede usarse como argumento al invocar otro procedimiento". A continuación se observa una solución completa:



Solución completa del proyecto "Cuadrados de colores, toma 2" utilizando parámetros

Luego de hacer una puesta en común, reflexionamos con los estudiantes sobre las diferencias entre las soluciones alcanzadas durante el desarrollo de la actividad. En todos los casos dibujamos 4 cuadrados de distintos colores, pero: *(i)* en el primer caso programamos un procedimiento para dibujar cada cuadrado; *(ii)* en el segundo abordaje escribimos un procedimiento para dibujar cuadrados de cualquier color; y *(iii)* en el tercero conseguimos que porciones muy parecidas del programa no se repitieran una y otra vez. Finalmente, preguntamos: "Si quisiéramos dibujar figuras que no fueran cuadrados, e incluso dibujos en los que las líneas no tuviesen la misma longitud, ¿que tendríamos que hacer?". Alcanza, en este caso, con agregarle un parámetro `longitud de la línea` a `Dibujar una línea [ ] [ ] [ ]`.



Modificación del procedimiento para dibujar líneas de diferentes longitudes y un posible dibujo

CIERRE

Para cerrar, recordamos la diferencia entre parámetro y argumento: mientras que el primero es el "agujero" al que le ponemos un nombre cuando definimos un procedimiento, el segundo es un valor concreto que se le asigna al parámetro al invocar el procedimiento en el programa.

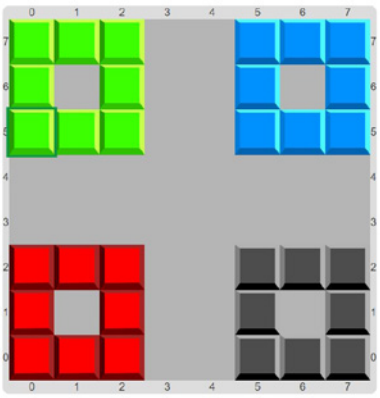
NOMBRE Y APELLIDO:

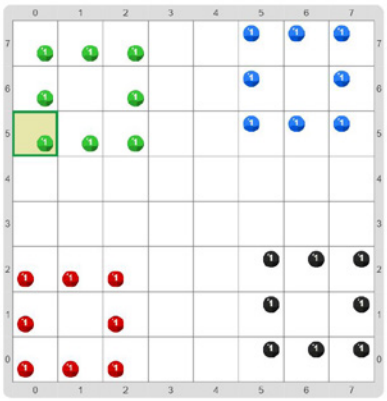
CURSO: FECHA:

CUADRADOS DE COLORES

Ahora vamos a construir un programa para dibujar cuadrados de colores en Gobstones.

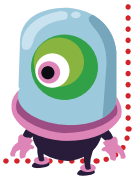
1. Abrí el proyecto "Cuadrados de colores, toma 1", y completá el programa para que dibuje cuatro cuadrados, uno en cada esquina, y cada uno de diferente color. El tablero final, con y sin vestimenta, debe ser como este:





SUBTAREAS Y PROCEDIMIENTOS

El proyecto ya viene con algunos procedimientos construidos, para que no te olvides que siempre es conveniente dividir un problema en subtareas



2. Abrí el proyecto "Cuadrado de colores, toma 2" y completá el programa para lograr el mismo tablero final que en la consigna anterior. Esta vez tenés que completar un único procedimiento para dibujar un cuadrado. ¿Cómo hacés para que el color no sea siempre el mismo?

3. Escribí un nuevo procedimiento `Dibujar una línea [ ] [ ]` para dibujar los lados del cuadrado. ¿Qué parámetros tiene el nuevo procedimiento? ¿Qué nombres les pusiste? ¿Qué argumentos usaste al invocarlo?

4. Modificá `Dibujar una línea [ ] [ ]` para cambiar el programa, de modo que pueda hacer distintos dibujos. A lo mejor necesitás agregar un nuevo parámetro para indicar la longitud de la línea. Acá hay un dibujo que te puede servir de inspiración.

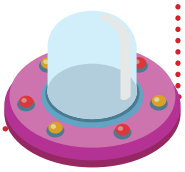
¿Cómo quedó el tuyo?

¡NO TE OLVIDES!

Guardá tu programa antes de modificarlo, así podés conservar las distintas soluciones que construyas.

BUSCÁ NUEVAS OPCIONES

Investigá cómo hacer para agregarle parámetros a los procedimientos.



Secuencia Didáctica 2

REPETICIÓN CONDICIONAL

En ocasiones debemos repetir una tarea, pero no sabemos de antemano la cantidad de veces que hay que hacerlo. En estos casos, no es posible utilizar una repetición simple para resolver el problema. Los lenguajes de programación proveen otra herramienta, llamada **repetición condicional**, que repite un bloque de instrucciones hasta que se cumpla una determinada condición.

Comenzaremos la secuencia presentando a los estudiantes la noción de repetición condicional a partir de recorridos de longitud desconocida. A continuación, ejercitaremos su uso y la combinaremos con otras herramientas ya trabajadas, como la alternativa condicional y la repetición simple.

OBJETIVOS

- Presentar la repetición condicional.
- Combinar la repetición condicional con otros comandos.

Actividad 1
Súper Lucho

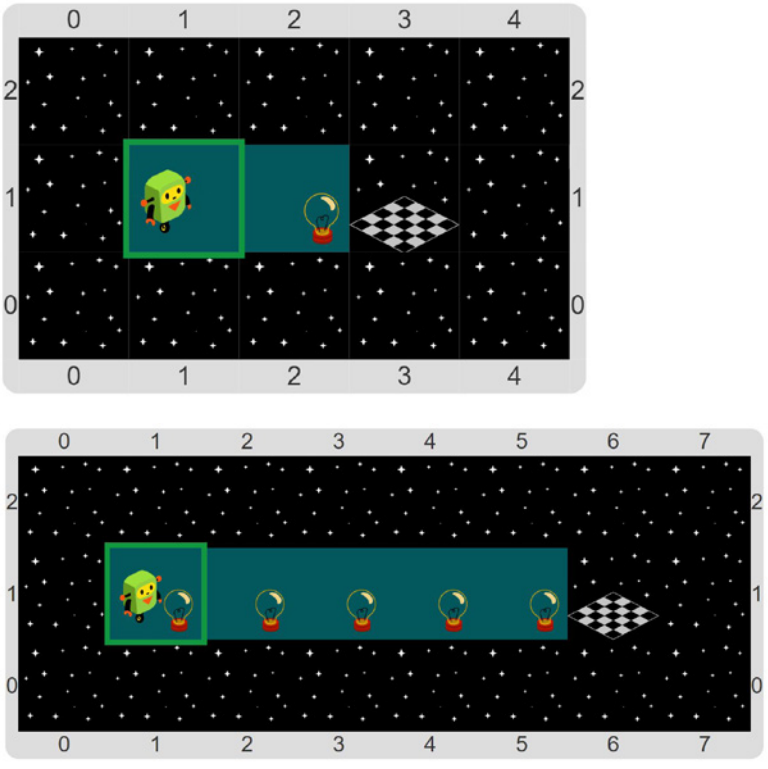
DE A DOS

OBJETIVOS
- Comprender qué es la repetición condicional.
- Combinar la repetición condicional con la alternativa condicional.

MATERIALES
- Computadoras
- Gobstones
- Ficha para estudiantes

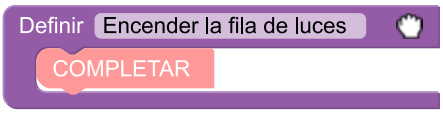
DESARROLLO
En esta actividad comenzaremos a trabajar con una herramienta muy útil en programación: la **repetición condicional**. La actividad tiene dos partes: en la primera hay que resolver un desafío en el que es necesario usar el comando `repetir hasta que [ ]`; en la segunda, sobre un escenario similar, hay que combinar la repetición condicional con la alternativa condicional.

Luego de repartirles la ficha a los estudiantes, les pedimos que abran el proyecto de Gobstones "Súper Lucho 1". En el tablero encontrarán al robot Lucho situado en la primera celda de un camino. En cada celda, salvo en la última -que tiene una bandera a cuadros-, hay una lamparita que está apagada. El objetivo es armar un programa para que Lucho prenda todas las luces. Hay varios tableros iniciales posibles con distintas cantidades de celdas y luces a encender. El programa debe funcionar en todos ellos.



Dos tableros iniciales

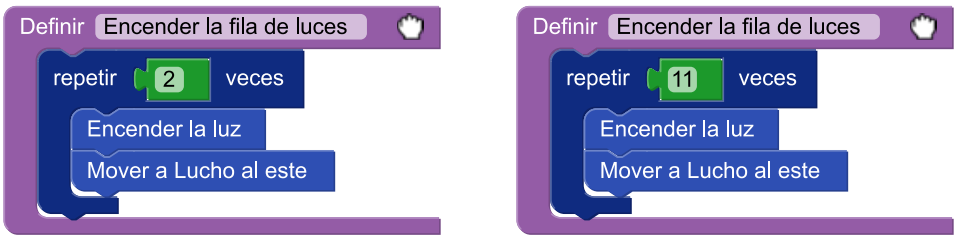
En el panel del programa verán que hay un procedimiento vacío: `Encender la fila de luces`. Los estudiantes van a tener que completar solamente ese procedimiento.



Procedimiento a completar

Les damos algunos minutos para que intenten resolverlo con las herramientas que ya conocen; se espera que descubran que con ellas no es posible cumplir el objetivo para todos los escenarios. Como en el primer escenario que encontrarán hay solo dos lamparitas, una solución que pueden proponer algunos estudiantes es utilizar una repetición simple con argumento `2` - esto es, `repetir [2] veces` - junto con los procedimientos de la biblioteca `Encender la luz` y `Mover a Lucho al Este`. Tal programa solo sirve para resolver el primer escenario: al ejecutar el programa en otro escenario, este no soluciona el problema.

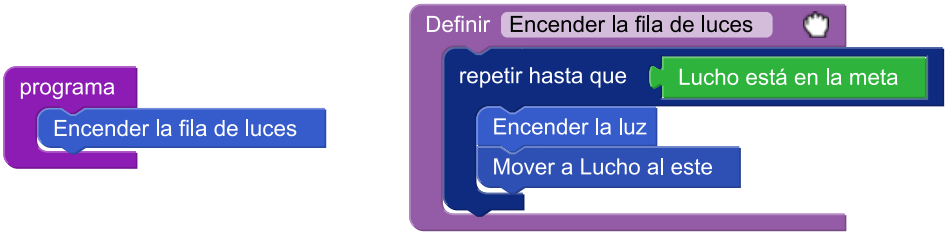
Otra solución que puede proponerse es utilizar la repetición simple con un argumento muy grande, de manera tal que sirva para todos los escenarios posibles. El problema es que, de este modo, Lucho se caerá del tablero en algunos escenarios y hará "¡Boom!".



Soluciones incorrectas para cubrir todos los escenarios del proyecto Súper Lucho 1

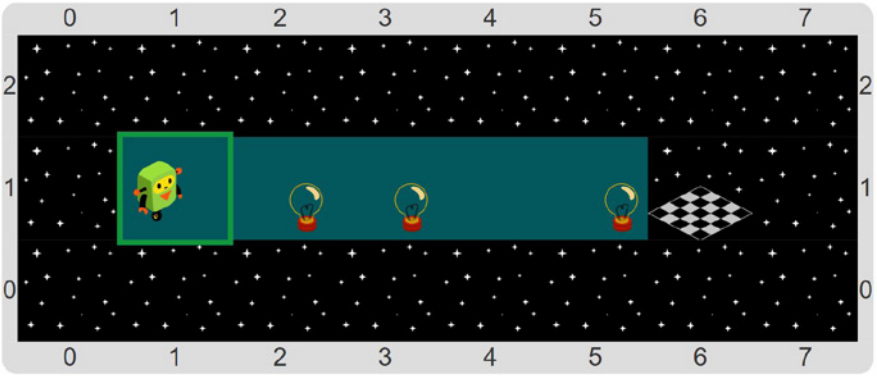
En esta etapa, les decimos a los estudiantes que, para los casos como este, en los que no se conoce de antemano la cantidad de repeticiones que hay que realizar, disponemos de una nueva herramienta de programación: la **repetición condicional**. En Gobstones, el comando correspondiente a una repetición condicional es `repetir hasta que [ ]`, que nos permite repetir una secuencia de instrucciones hasta que una condición sea verdadera.

Una solución posible para el desafío propuesto consiste en utilizar `repetir hasta que [ ]` junto con la función `Lucho está en la meta`, disponible en *Biblioteca* > *Funciones*. Esta devuelve verdadero si Lucho está ubicado sobre la bandera a cuadros y falso en caso contrario. Dentro de la repetición, entonces, hay que darle a Lucho las instrucciones para que encienda una luz y avance hacia el este, hasta que se encuentre parado en la meta.



Programa que resuelve el proyecto "Súper Lucho 1"

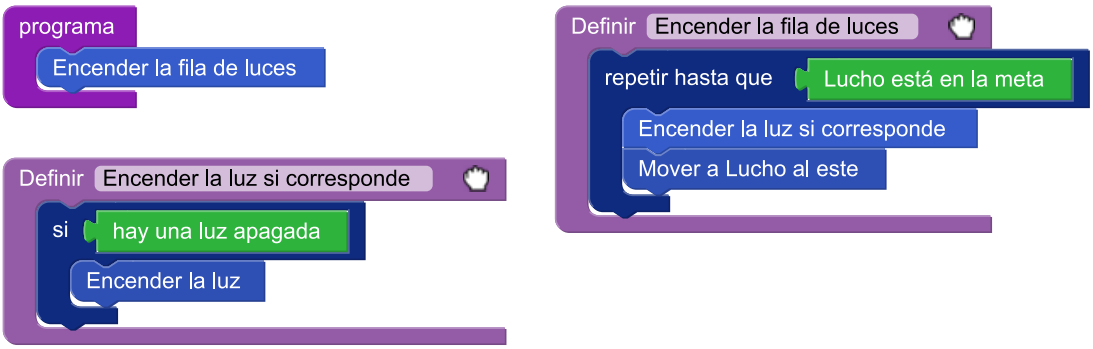
A continuación, les pedimos a los estudiantes que abran el proyecto de Gobstones "Súper Lucho 2". El objetivo es similar al de la actividad anterior: Lucho tiene que prender todas las luces hasta llegar al final del tablero. Sin embargo, no todas las celdas tienen una luz.



Un escenario inicial de "Súper Lucho 2"

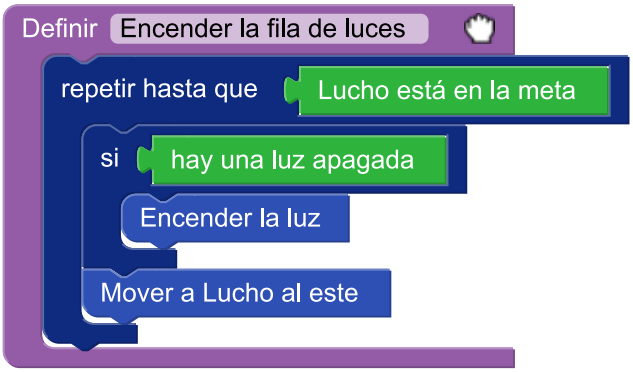
Antes de que los estudiantes se dispongan a resolver este nuevo problema, les preguntamos: "¿Les parece que el programa anterior sirve para resolver este desafío? ¿Por qué? ¿Se le podría hacer algún cambio? ¿Cuál? ". A partir de este intercambio se espera que detecten que, en este caso, solamente hay que darle a Lucho la instrucción de encender una luz cuando esté parado sobre una celda en la que haya una.

Continuamos: "¿Y cómo hacemos para prender la luz a veces sí y a veces no?". La alternativa condicional —con la que los estudiantes ya trabajaron en la secuencia 2 del capítulo 4— es la herramienta que nos permite verificar si en una celda hay o no una lamparita. Para que el programa sea lo más claro posible conviene crear un nuevo procedimiento que se ocupe de encender la luz si corresponde. De este modo, se evita anidar la alternativa condicional dentro de la repetición.[1] Para completar este procedimiento, al que podemos llamar `Encender la luz si corresponde`, resulta útil la función `hay una luz apagada`, disponible en *Biblioteca* > *Funciones*.



Solución de "Súper Lucho 2"

Puede ocurrir que algún estudiante proponga que la alternativa condicional esté dentro del mismo procedimiento que tiene la repetición condicional. En ese caso, es importante hacer notar que conviene dividirla en distintas subtareas e incentivar a que resuelvan cada subtarea mediante un procedimiento diferente. De este modo, se garantiza que cada uno de los procedimientos resulte sencillo de entender. En general, y no solo en este ejercicio, intentar solucionar todo en el mismo procedimiento tiene por resultado un programa más difícil de entender y menos legible.



Posible propuesta de algún estudiante

[1] Decimos que un bloque está anidado dentro de otro cuando el primero es parte del cuerpo del segundo. En este caso, el bloque `si [ ]` aparecería dentro del cuerpo de `repetir hasta que [ ]`.

Una vez que los estudiantes hayan completado la actividad, les comentamos que lo que han hecho es un programa para realizar un **recorrido**; es decir, recorrer una secuencia de elementos, procesando cada uno de algún modo. Es una forma de dividir un problema de procesamiento de elementos en tres subtareas: una función que determina si finalizó el recorrido (`Lucho está en la meta`), un procedimiento de procesamiento de elementos (`Encender la luz si corresponde`) y un procedimiento para pasar al siguiente elemento (`Mover a Lucho al este`).

CIERRE

Al finalizar, reflexionamos con los estudiantes acerca de la repetición condicional. Al respecto, recordamos que la repetición condicional resulta útil cuando no sabemos de antemano la cantidad de veces que hace falta repetir algo, pero sí conocemos una condición que, cuando se cumple, implica que no hay que seguir repitiendo la tarea.

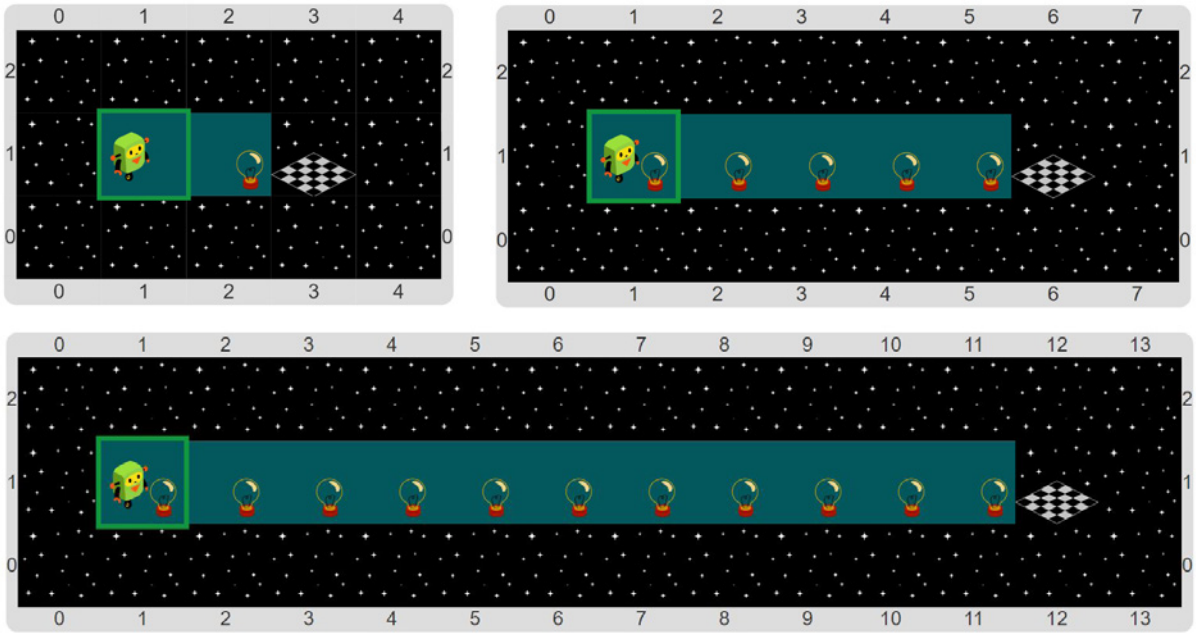
NOMBRE Y APELLIDO:

CURSO: FECHA:

SÚPER LUCHO

Otra vez el pobre Lucho tiene que ocuparse de que todo quede bien iluminado. Y, de nuevo, tu misión es darle instrucciones para que lo logre.

1. Abrí el proyecto "Súper Lucho 1". Lucho tiene que llegar a la meta habiendo encendido todas las luces que hay en el camino. ¡El problema es que no sabemos cuántas son! Algunos de los tableros iniciales son estos:



PARA TENER EN CUENTA

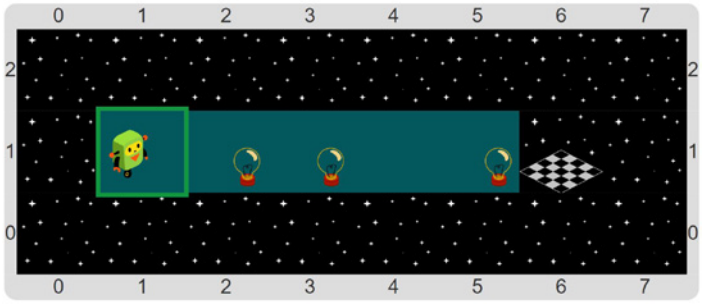
- Revisá en la biblioteca los procedimientos y las funciones que vienen en el proyecto.
- Buscá en el entorno algún comando que te permita que Lucho avance encendiendo foquitos hasta llegar a la meta.

¿Usaste por primera vez algún comando? ¿Cuál? ¿Qué te permitió hacer?

NOMBRE Y APELLIDO:

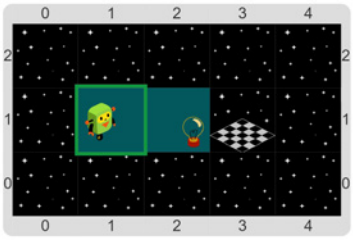
CURSO: FECHA:

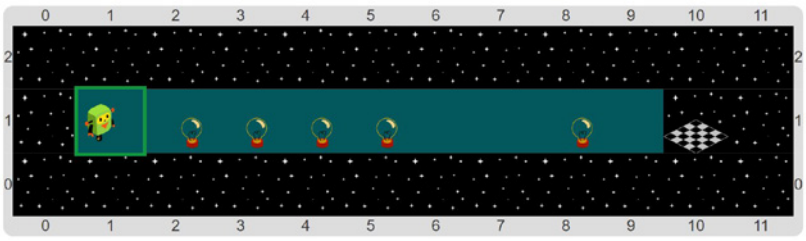
2. Abrí "Súper Lucho 2". Igual que antes, Lucho tiene que prender todas las lamparitas y no sabemos cuánto mide el sendero. Pero, además, ahora hay varias celdas sin lamparitas. Algunos de los tableros iniciales son los siguientes:



AYUDA

Combiná el nuevo comando que usaste en la consigna anterior con una alternativa condicional.





CONSEJO

Cuando tengas que combinar varias herramientas de programación (como repeticiones y alternativas condicionales), tené presente que conviene identificar el propósito de cada una y ubicarlas en procedimientos diferentes.

¿Qué dos herramientas son fundamentales para tu solución? ¿Cómo las combinaste?

Actividad 2
Laberinto con queso, recargado

DE A DOS

OBJETIVO

- Afianzar el uso combinado de la repetición condicional y la alternativa condicional.

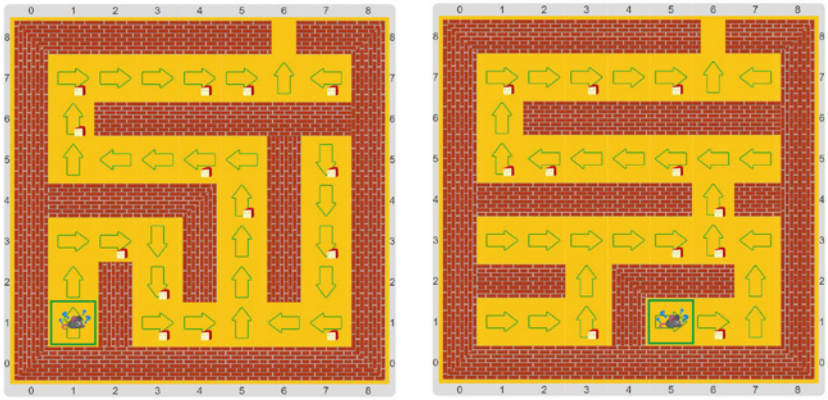
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

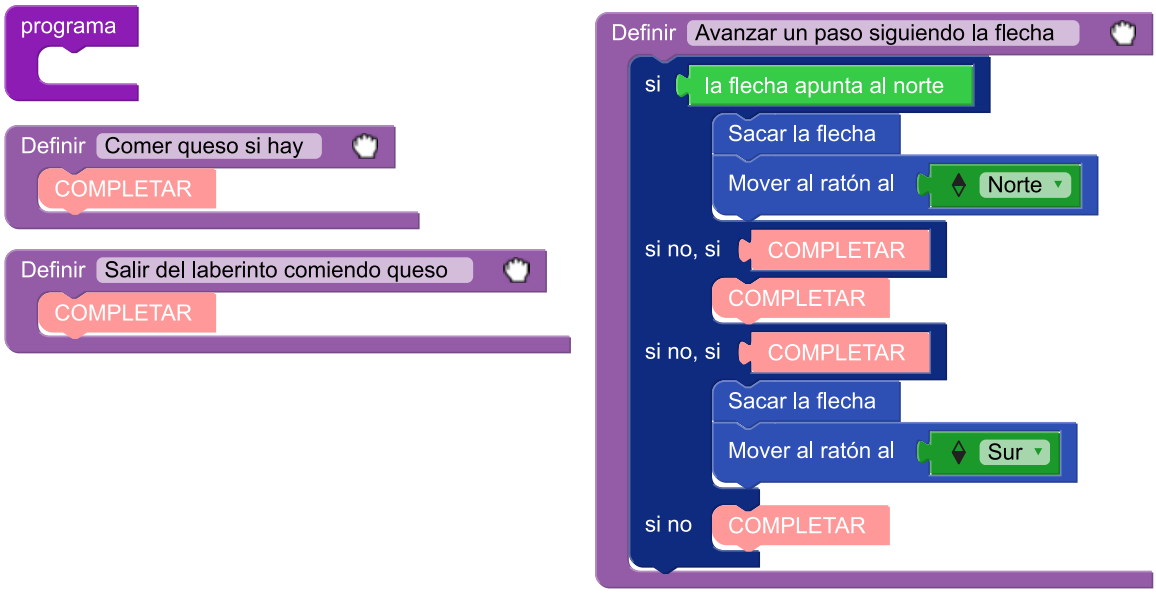
El objetivo de esta actividad es ejercitar el uso de la repetición condicional en combinación con la alternativa condicional.

Repartimos la ficha a los estudiantes y los invitamos a abrir el proyecto "Laberinto con queso, recargado". El objetivo es guiar a un ratón para que salga de un laberinto después de comer todos los pedazos de queso con los que se haya topado en el camino. Hay distintos tableros iniciales y la distancia —medida en celdas— que tiene que recorrer el roedor no es la misma en todos. Además, las celdas en las que hay trozos de queso tampoco coinciden en todos los tableros. Por último, en cada posición hay una flecha que indica en qué dirección el pequeño roedor tiene que dar el próximo paso.



Dos tableros iniciales de "Laberinto con queso, recargado"

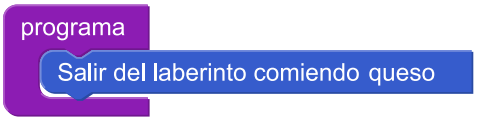
En el panel del programa observarán que hay que: *(i)* completar en su totalidad el cuerpo principal del programa y los procedimientos `Comer queso si hay` y `Salir del laberinto comiendo queso`; y *(ii)* completar `Avanzar un paso siguiendo la flecha`, que viene parcialmente construido.



Espacio del programa al cargar el proyecto

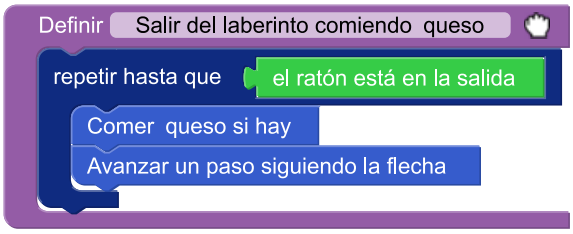
Al explorar el entorno encontrarán que en la biblioteca se encuentran disponibles los procedimientos y funciones que hacen falta para resolver el desafío: `Comer queso`, `Mover al ratón al [ ]`, `Sacar la flecha`, `el ratón está en la salida`, `hay queso`, `la flecha apunta al norte`, `la flecha apunta al este`, `la flecha apunta al sur` y `la flecha apunta al oeste`.

Para alcanzar el objetivo, desde el cuerpo principal del programa hay que invocar el procedimiento `Salir del laberinto comiendo queso`.



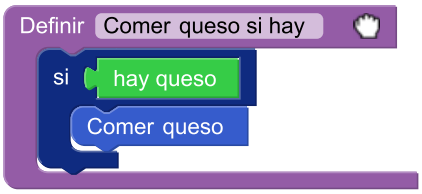
Cuerpo principal del programa

Para completar `Salir del laberinto comiendo queso` hay que hacer avanzar al ratón hasta que alcance la salida del laberinto y, a cada paso, si encuentra un trozo de queso, debe comerlo. Para ello, hay que combinar `repetir hasta que [ ]` con la función `estoy en la salida`, y usar los procedimientos `Comer queso si hay` y `Avanzar un paso siguiendo la flecha`.



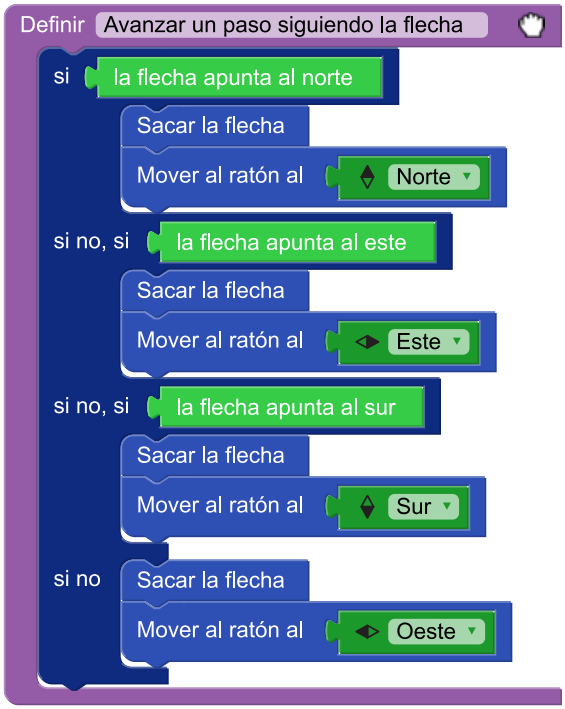
Procedimiento `Salir del laberinto comiendo queso`

El procedimiento `Comer queso si hay` se vale de la función `hay queso` y del procedimiento `Comer queso`, ambos disponibles en la biblioteca.



Procedimiento `Comer queso si hay`

Para completar `Avanzar un paso siguiendo la flecha` alcanza con reproducir para las restantes direcciones lo que se hace cuando el ratón se encuentra posicionado sobre una flecha que apunta al norte: sacar la flecha y mover al ratón en la dirección que corresponde. De este modo, completamos el programa.



Procedimiento `Avanzar un paso siguiendo la flecha`

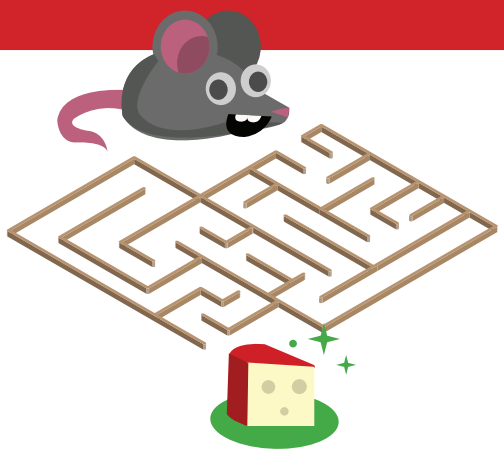
CIERRE

Para completar la actividad, reflexionamos sobre algunos casos en los que la repetición condicional resulta indispensable. Por ejemplo, cuando debemos recorrer un camino y no sabemos de antemano su longitud, aunque sí podemos preguntar si terminó. Recordamos, además, que la combinación de una repetición con una alternativa condicional nos permite tomar decisiones sobre qué hacer con los elementos en cada uno de los pasos del recorrido.

NOMBRE Y APELLIDO:

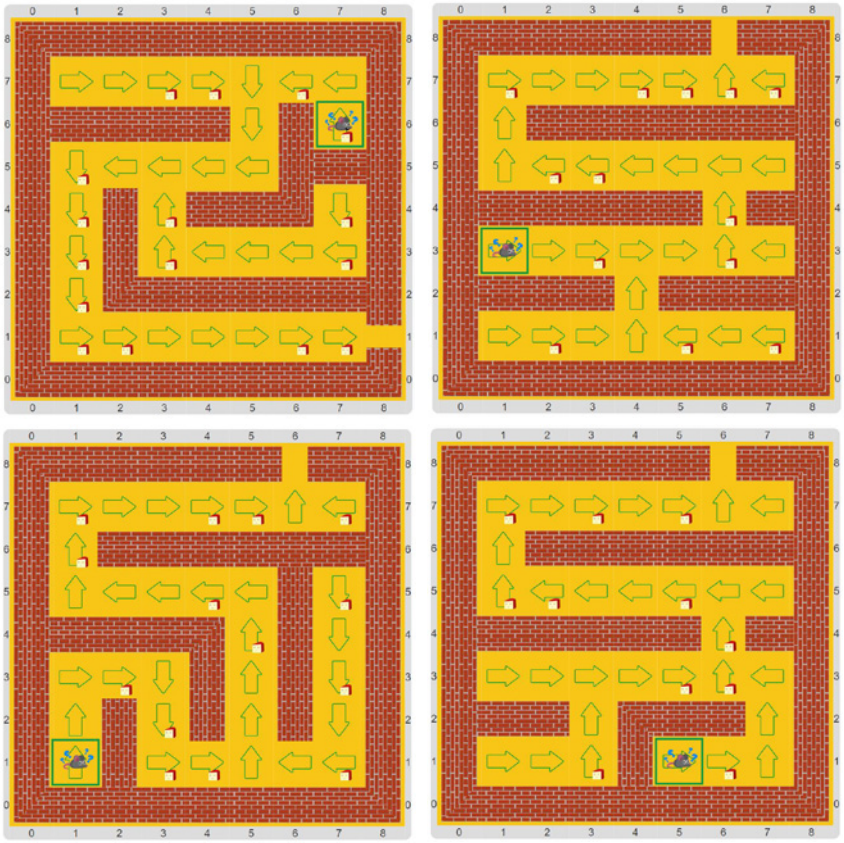
CURSO: FECHA:

LABERINTO CON QUESO, RECARGADO



El ratón está en un laberinto y para salir tiene que seguir las flechas. Además, debe comer los pedazos de queso desperdigados por el piso. Después de todo, el camino hacia la salida no está tan mal, ¿no?

1. Abrí el proyecto "Laberinto con queso, recargado" y completá el programa para que el ratón salga del laberinto. Te mostramos algunos tableros iniciales.



PARA TENER EN CUENTA

- Pensá en el problema que enfrenta el ratón y no te preocupes por la vestimenta: los procedimientos y las funciones que están en la biblioteca se encargan de ello.
- Asegurate de que tu programa resuelva el problema para todos los tableros iniciales.

2. ¿Qué herramienta de programación utilizaste en `Salir del laberinto comiendo queso`? ¿Y en `Comer queso si hay`?

Actividad 3
Más entrenamiento para el Beto

DE A DOS

OBJETIVO

- Combinar la repetición condicional con la repetición simple.

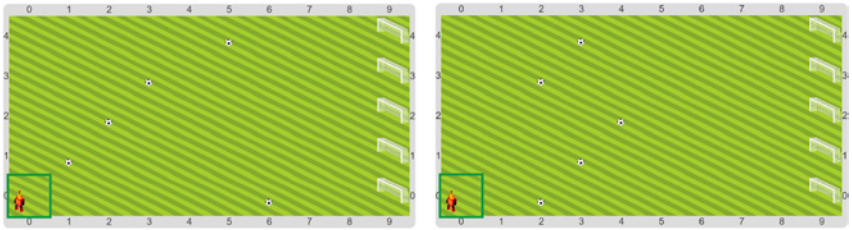
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

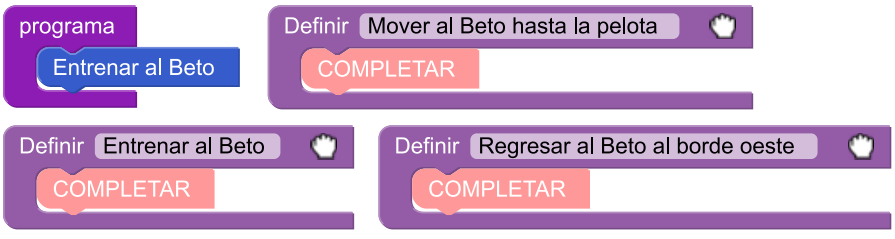
En esta actividad, se presenta una nueva combinación de comandos: la repetición simple con la repetición condicional. Se trabaja sobre un recorrido que se desarrolla en dos dimensiones; en ella, una de sus dimensiones es fija y la otra es variable. Mientras que la primera se resuelve con una repetición simple, la segunda requiere el uso de una repetición condicional.

Comenzamos repartiendo la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Más entrenamiento para el Beto" y que observen los distintos tableros iniciales. Verán que estos siempre tienen la misma cantidad de pelotas, una en cada fila. Sin embargo, su ubicación varía entre distintos escenarios. El objetivo es que el Beto meta los 5 goles, independientemente de dónde se encuentren ubicadas las pelotas en las filas.



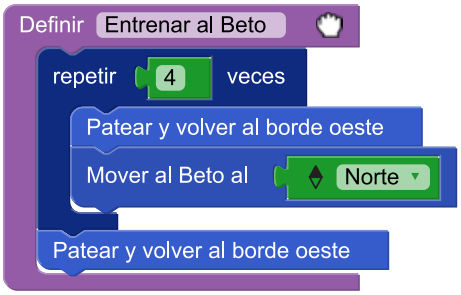
Dos escenarios iniciales de "Más entrenamiento para el Beto"

En el panel del programa, encontrarán procedimientos y funciones ya programados y otros que falta completar: `Entrenar al Beto`, `Mover al Beto hasta la pelota` y `Regresar al Beto al borde oeste`.



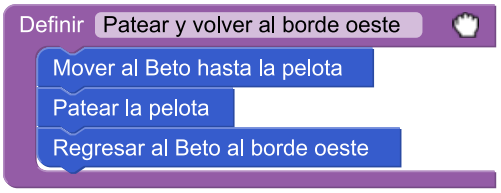
Cuerpo principal del programa y procedimientos que falta completar

Alentamos a los estudiantes a que piensen una estrategia de solución para resolver el problema, de forma tal de que puedan completar el procedimiento `Entrenar al Beto`. La más simple consiste en que el Beto patee la pelota de una fila, pase a la inmediatamente superior, patee la de esa fila y continúe hasta terminar de patear todas las pelotas. Como son cinco filas en total, alcanza con combinar una repetición simple con los procedimientos `Patear y volver al borde oeste`, que viene resuelto en el proyecto, y `Mover al Beto al [ ]`, que está disponible en la biblioteca. Nótese que el procesamiento de la fila superior queda fuera del cuerpo de `repetir [ ] veces`, por tratarse de un caso de borde. Si no, el Beto caería fuera del tablero.



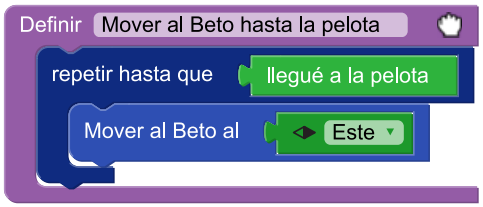
Procedimiento `Entrenar al Beto`

El procedimiento `Patear y volver al borde oeste` invoca `Mover al Beto hasta la pelota`, `Patear la pelota` y `Regresar al Beto al borde oeste`. Falta completar el primero para trasladar al robot hasta la pelota, y el tercero para volver a ubicar al Beto sobre el borde oeste de la fila.



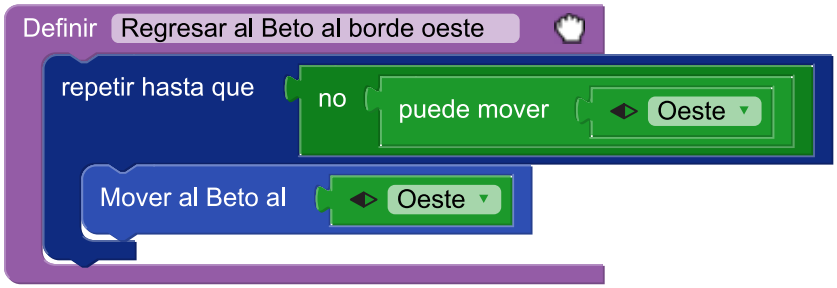
Procedimiento `Patear y volver al borde oeste`

Para que el Beto pueda ubicarse frente a la pelota hay que moverlo en dirección este hasta la celda donde está la pelota. Hay que combinar, entonces, una repetición condicional `repetir hasta que [ ]` con la función `llegué a la pelota` -disponible en la biblioteca- y el procedimiento `Mover al Beto al [ ]`.



Procedimiento `Mover al Beto hasta la pelota`

Finalmente, para completar `Regresar al Beto al borde oeste`, mientras no haya llegado al borde oeste, lo moveremos en esa dirección. En este caso, para expresar la condición de corte de la repetición hay que usar el operador de negación `no [ ]` y el sensor `puede mover [ ]`.



Procedimiento `Regresar al Beto al borde oeste`

CIERRE

Para terminar, destacamos que en un mismo programa pueden coexistir la repetición simple y la repetición condicional. La primera se utiliza en contextos donde se conoce de antemano la cantidad de veces que hay que repetir una secuencia de acciones; la segunda, cuando no se conoce esa cantidad, pero sí alguna condición que indique que la repetición no debe continuar.

NOMBRE Y APELLIDO:

CURSO: FECHA:

MÁS ENTRENAMIENTO PARA EL BETO

Cinco pelotas. Cinco arcos. El Beto no se conforma con meter goles. Quiere clavarlas todas en el ángulo. Para eso, no hay más remedio que seguir entrenando.

1. Abrí el proyecto "Más entrenamiento para el Beto" y completá el programa para que el Beto mande todas las pelotas al fondo del arco. Estos son algunos tableros iniciales posibles.

2. ¿Cómo hiciste para que el Beto no se caiga del tablero?

PARA TENER EN CUENTA

- En todos los escenarios hay 5 pelotas, pero no sabemos de antemano a qué distancia del arco está cada una. El Beto va a tener que moverse desde su posición hasta llegar a cada balón.
- Gran parte del programa ya está resuelto. Vos tenés que ocuparte de completar `Entrenar al Beto`, `Mover al Beto hasta la pelota` y `Regresar al Beto al borde oeste`.

Actividad 4
¿Dónde puse la llave?

DE A DOS

OBJETIVO

- Utilizar la repetición condicional para recorrer parcialmente una estructura.

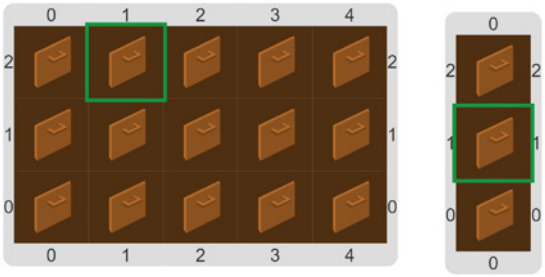
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

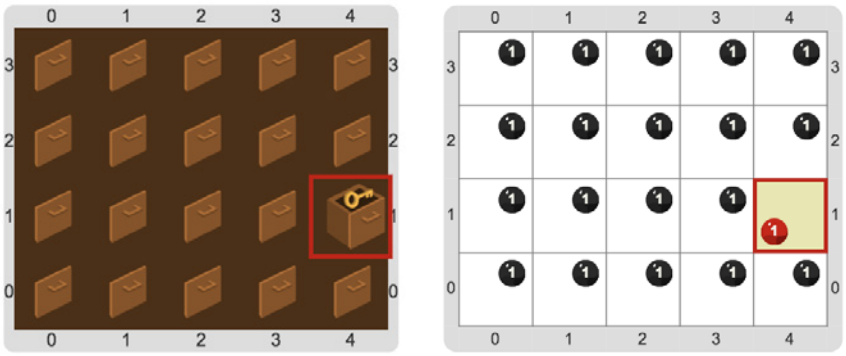
En esta actividad abordaremos un uso particular que se le puede dar a la repetición condicional: realizar un recorrido hasta encontrar un elemento, en lugar de recorrerlos todos. Solamente hay que verificar en cada posición si está el objeto buscado o si hay que seguir avanzando.

Les repartimos la ficha a los estudiantes y los invitamos a abrir el proyecto de Gobstones "¿Dónde puse la llave?". En el tablero van a notar que en cada celda hay un cajón. En uno de ellos hay una llave. El objetivo es construir un programa que vaya revisando los cajones hasta localizarla. Al hacer clic varias veces en el botón *Ejecutar* verán que hay distintos tableros iniciales con dimensiones diferentes. Además, también varía la celda sobre la que aparece inicialmente el cabezal.



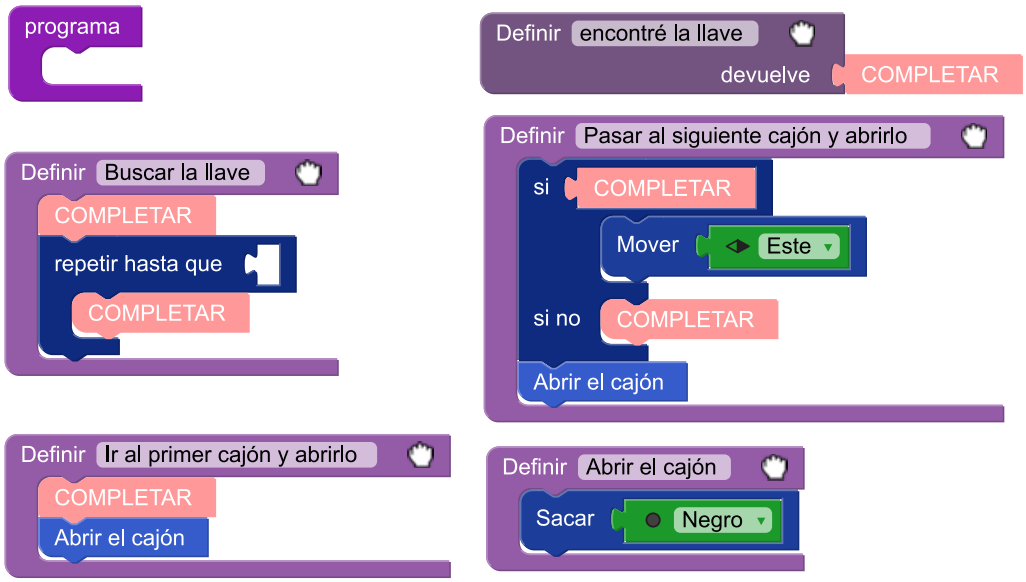
Dos tableros iniciales del proyecto "¿Dónde puse la llave?"

Para resolver el desafío deben tener presente que un cajón cerrado se representa con una bolita negra y, la llave, con una bolita roja.



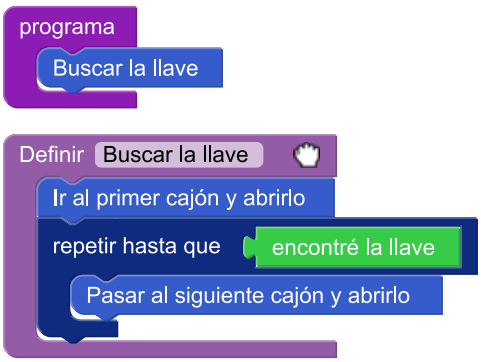
Vestimenta del proyecto

Al observar el panel del programa van a notar que se encuentra en gran parte incompleto. Antes de que los estudiantes comiencen a programar, discutimos entre todos la estrategia para resolver el problema. "¿Cómo podemos recorrer el tablero? ¿Es lo mismo hacerlo de oeste a este y de sur a norte que al revés? ¿Qué pista nos da al respecto la instrucción `Mover [Este]` de `Pasar al siguiente cajón y abrirlo`? En el procedimiento `Ir al primer cajón y abrirlo`, ¿cuál será el primer cajón?". Con estas preguntas buscamos ayudar a plantear una estrategia de solución.



Espacio del programa al cargar el proyecto

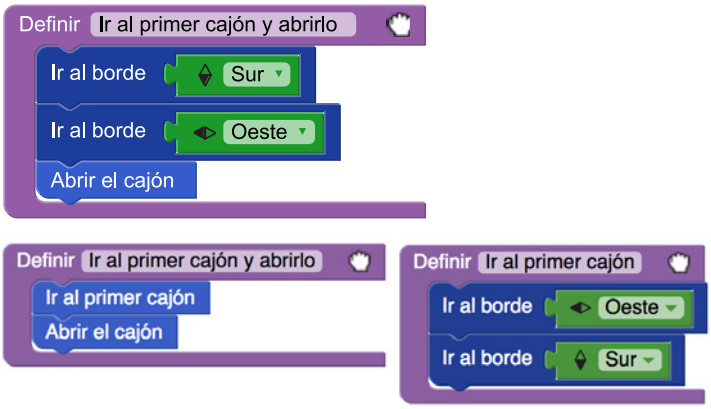
El cuerpo principal del programa consiste en solo una invocación a `Buscar la llave`. Este, por su parte, puede definirse como ir al primer cajón y fijarse si allí está la llave, luego pasar al siguiente y verificar si contiene la llave, y así ir pasando por los cajones hasta encontrar la bendita llave. Para hacerlo, hay que usar una repetición condicional, los procedimientos `Ir al primer cajón y abrirlo`, `Pasar al siguiente cajón y abrirlo` y la función `encontré la llave`. Hay que tener presente que, si no se incluyese la invocación `Ir al primer cajón y abrirlo`, el recorrido comenzaría en la posición en la que inicialmente aparece el cabezal y, por lo tanto, podría no inspeccionarse el cajón que contiene la llave.



Cuerpo principal del programa y procedimiento `Buscar la llave`

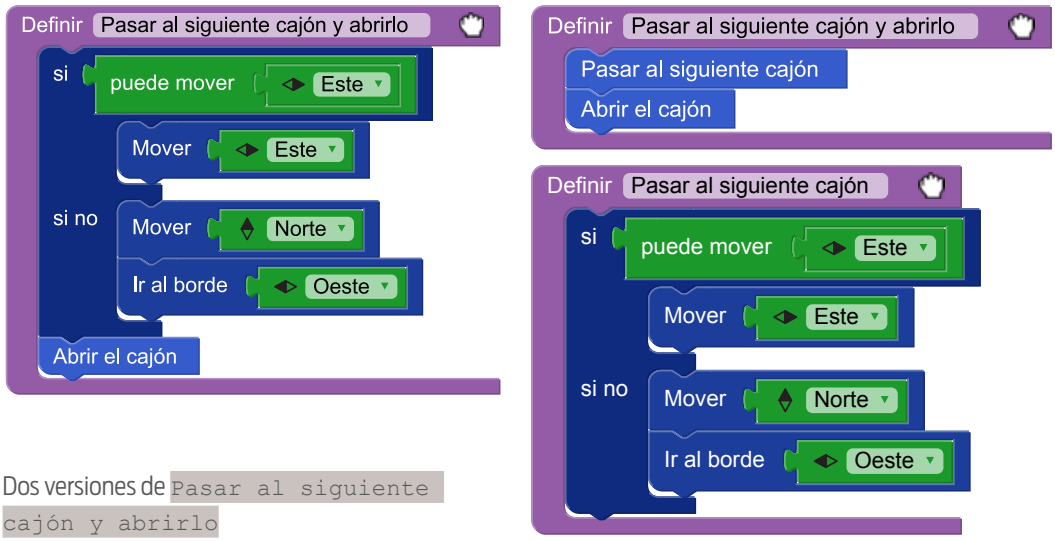
Llegados a este punto, los estudiantes deben ocuparse de definir cómo es el recorrido que realiza el cabezal sobre el tablero. Proponemos aquí una solución en la que se recorren los cajones de a una fila por vez, comenzando por la que está más al sur, en dirección oeste-este. Sin embargo, con pocos cambios se pueden obtener soluciones que recorren la estantería de otro modo.

Para completar `Ir al primer cajón y abrirlo` falta colocar el cabezal sobre el extremo suroeste del tablero, que puede hacerse usando dos veces el comando `Ir al borde [ ]`. Alternativamente, el posicionamiento del cabezal puede resolverse creando un nuevo procedimiento llamado, por ejemplo, `Ir al primer cajón`.



Dos versiones de `Ir al primer cajón y abrirlo`

Para completar `Pasar al siguiente cajón y abrirlo` hay que observar si el cabezal ha llegado a la última celda de una fila —y, por lo tanto, debe moverse al inicio de la siguiente— o, simplemente debe avanzar una posición hacia el este. Con tal propósito, usamos el sensor `puede mover [ ]`. Nuevamente, el movimiento del cabezal puede encapsularse en un procedimiento separado.



Dos versiones de `Pasar al siguiente cajón y abrirlo`

Para completar el proyecto solo falta definir la función `encontré la llave`. Como la vestimenta representa este elemento con una bolita roja, alcanza con chequear si en la celda sobre la que se encuentra el cabezal hay una bolita de ese color: si es así, devuelve verdadero y si no, devuelve falso.

Función `encontré la llave`

CIERRE

Para concluir la actividad, reflexionamos con los estudiantes sobre las posibilidades que nos brinda la repetición condicional: recorrer una estructura cuyo tamaño es desconocido de antemano y, además, recorrerla de manera parcial, hasta que se cumpla una determinada condición.

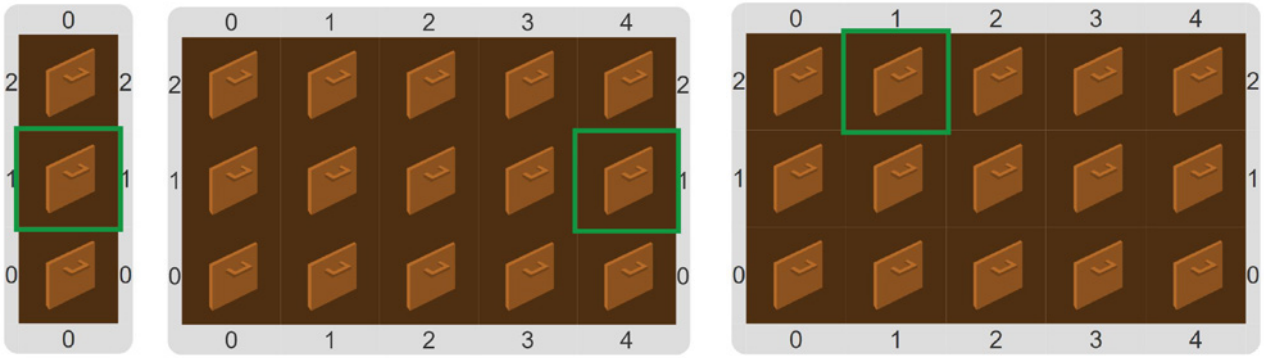
NOMBRE Y APELLIDO:

CURSO: FECHA:

¿DÓNDE PUSE LA LLAVE?

¿Nunca te pasó no poder irte de tu casa porque no encontrabas la llave cuando estabas apurado?

1. Abrí el proyecto "¿Dónde puse la llave?" y completá el programa para que revise los cajones en busca del objeto perdido. Hay varios tableros iniciales. Mirá algunos y prestá atención a la vestimenta.

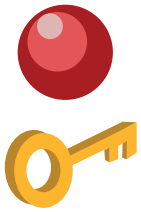


2. ¿Qué estrategia usaste en `Buscar la llave`?

3. ¿Qué pasaría con tu programa si las llaves no estuvieran en ningún cajón? ¿Por qué?

PARA TENER EN CUENTA

- El cabezal puede aparecer en cualquier posición del tablero.
- Vas a tener que usar sensores para determinar cuál es el próximo cajón que hay que inspeccionar y para saber si encontraste la llave.
- La llave está representada por una bolita roja.



Secuencia Didáctica 3

VARIABLES

Hasta aquí, siempre que precisamos que un programa recordase algún valor lo codificamos usando el tablero y algunas bolitas. Sin embargo, existen problemas en los que este enfoque puede volverse muy complejo. Los lenguajes de programación proveen una herramienta para recordar información: las **variables**.

En la primera actividad de la secuencia presentamos la noción de variables y explicamos cómo asignarles un valor. Luego, veremos uno de los usos más frecuentes: contar una cantidad que se va modificando durante la ejecución del programa. Finalmente, se usan variables en **funciones con procesamiento**, que permiten calcular valores utilizando información de todas las regiones del tablero.

OBJETIVOS

- Presentar la noción de variable.
- Combinar las variables con las herramientas aprendidas.
- Presentar las funciones con procesamiento.

Actividad 1
Otra vez el cabezal juega a ser mimo

DE A DOS

OBJETIVO

- Presentar el uso de variables en programación.

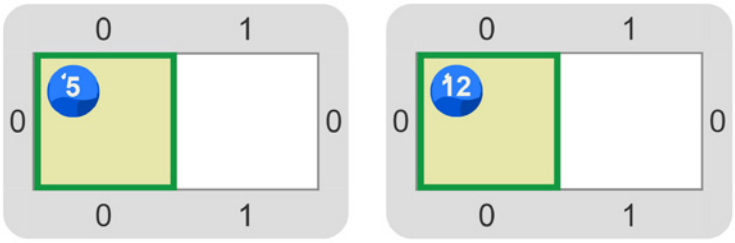
MATERIALES

- Computadoras
- Gob stones
- Ficha para estudiantes

DESARROLLO

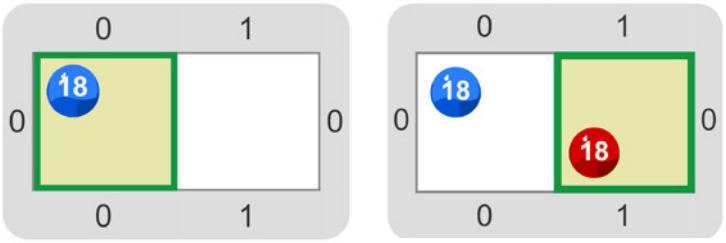
En esta actividad abordaremos la noción de **variable** e indagaremos en la relación entre las variables y la memoria de la computadora. Se trata de un desafío en Gobstones que, al usar una variable, admite una solución simple.

Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Otra vez el cabezal juega a ser mimo". Les indicamos que presionen varias veces el botón *Ejecutar* para que noten que los distintos tableros iniciales siempre tienen 2 celdas, que en la primera hay una cantidad de bolitas azules y que esta cantidad varía de una ejecución a otra.



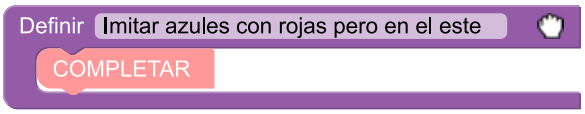
Dos tableros iniciales de "Otra vez el cabezal juega a ser mimo"

El objetivo de la actividad es construir un programa que ponga tantas bolitas rojas en la celda de la derecha, como azules haya en la celda de la izquierda. Por ejemplo, si el tablero inicial tuviera 18 bolitas azules en la primera celda, el final debería seguir teniendo esas mismas bolitas y, además, 18 bolitas rojas en la celda contigua.



Un tablero inicial y el tablero final esperado

En el panel del programa se puede observar el cuerpo principal del programa y un procedimiento que hay que completar: `Imitar azules con rojas pero en el este`.

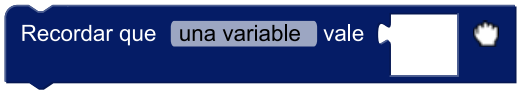


Porcedimiento a completar

Antes de que los estudiantes comiencen a trabajar en la computadora, proponemos un debate para que expongan sus ideas para resolver el problema, apelando tanto a lo que ya aprendieron como a las ideas nuevas que se les ocurran. En caso de que en el intercambio no surja ninguna propuesta de solución, podemos preguntar: "Sabemos que cuando el cabezal se mueve de una celda a otra ya no se puede usar un sensor para averiguar la cantidad de bolitas azules que había en la celda anterior. Supongamos que el programa tuviese la posibilidad de recordar ese valor, ¿nos sería útil? ¿Nos beneficiaría en algo? ¿Qué programa se imaginan que podríamos construir?".

Si bien al ubicarse en la primera celda, mediante el sensor `número de bolitas [ ]`, podemos detectar cuántas bolitas azules hay, no bien el cabezal se mueva hacia el este habremos perdido el registro de este valor. Que el programa lo recuerde es conveniente para resolver el problema. Para tal propósito, en programación existen las **variables**: un nombre con el que recordamos un valor que luego podemos consultar y modificar. Las variables permiten, por ejemplo, que los juegos recuerden la cantidad de puntos o de vidas de un jugador, personalizar la experiencia de uso de un programa mediante el registro de nuestro nombre al comienzo, etc.

En Gobstones, en *Comandos* > *Asignación* está el bloque `Recordar que [ ] vale [ ]`, que permite definir una variable y asignarle un valor.



Bloque para definir una variable

Al igual que con los procedimientos y las funciones, a las variables también hay que ponerles un nombre que describa su propósito. En este caso, un nombre adecuado es `cantidad de bolitas azules`. Además, para establecer el valor que almacena, hay que usar el sensor `número de bolitas [ ]`.

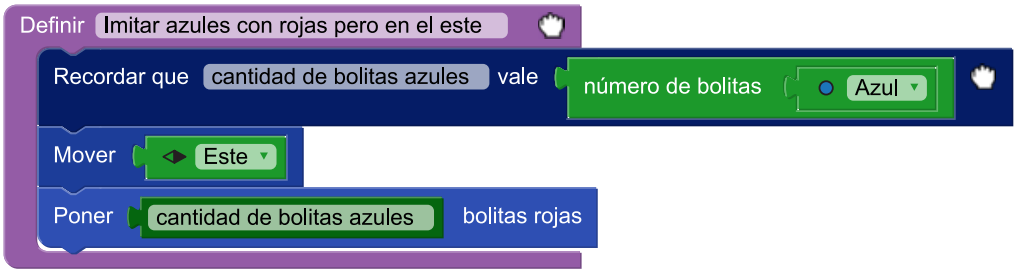
Variable que recuerda la cantidad de bolitas azules

Para usar la variable hay que hacer clic sobre la manito

Para usar la variable como argumento de otros comandos, procedimientos o funciones, hay que hacer clic sobre la manito que aparece a la derecha de la definición de la variable. Esta acción genera un nuevo bloque encastrable con nombre `cantidad de bolitas azules`.

Nuevo bloque para usar la variable

Ahora que ya sabemos que el programa puede recordar un valor, podemos completar el programa moviendo el cabezal al este y colocando la cantidad de bolitas rojas que indique la variable. Para realizar esto último, usamos la función `Poner [ ] bolitas rojas`, disponible en *Biblioteca > Procedimientos*.



Procedimiento `Imitar azules con rojas pero en el este`

CIERRE

Repasamos con los estudiantes las posibilidades que dan las variables. Hacemos hincapié en que las variables nos sirven para recordar un valor, y que, una vez definida, una variable puede leerse más adelante.

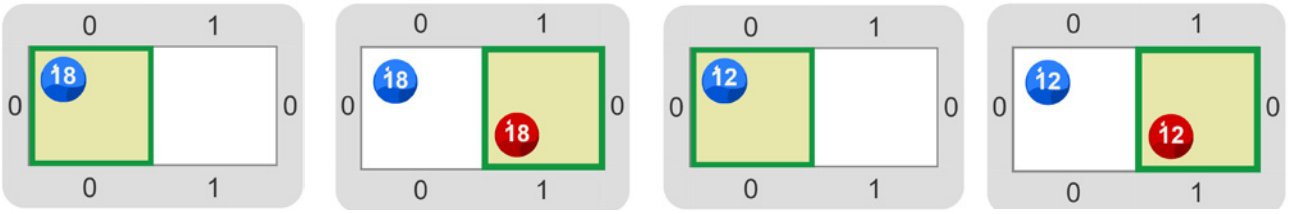
NOMBRE Y APELLIDO:

CURSO: FECHA:

OTRA VEZ EL CABEZAL JUEGA A SER MIMO

Las rojas no quieren ser menos que las azules. Sin embargo, tampoco quieren ser más.

1. Abrí el proyecto "Otra vez el cabezal juega a ser mimo" y construí un programa que ubique bolitas rojas en la celda del este, de manera que la cantidad de azules y rojas sea la misma. Acá podés ver dos tableros iniciales y los correspondientes tableros finales.



2. ¿Qué nueva herramienta usaste para resolver este problema? ¿Para qué la usaste?

3. Describí distintas situaciones en las que sea necesario que un programa recuerde información.

PARA TENER EN CUENTA

- Vas a precisar alguna herramienta para que el programa recuerde cuántas bolitas azules hay en la celda del oeste **antes** de moverte al este. ¡Explorá el entorno en busca de algo nuevo!
- Acordate de usar nombres descriptivos para las tareas que debe realizar el programa. ¿Qué te parece "cantidad de bolitas azules" como nombre del dato importante que debe recordar el programa?

Actividad 2
Súper Lucho también cuenta luces

DE A DOS

OBJETIVOS

- Comprender que se pueden usar variables para contar.
- Combinar las variables con la repetición condicional.

MATERIALES

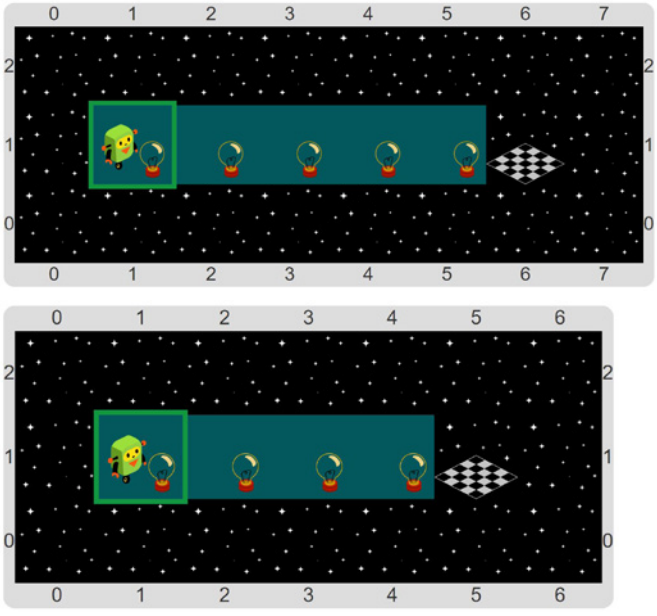
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es utilizar variables cuyo valor se modifica durante la ejecución de un programa.

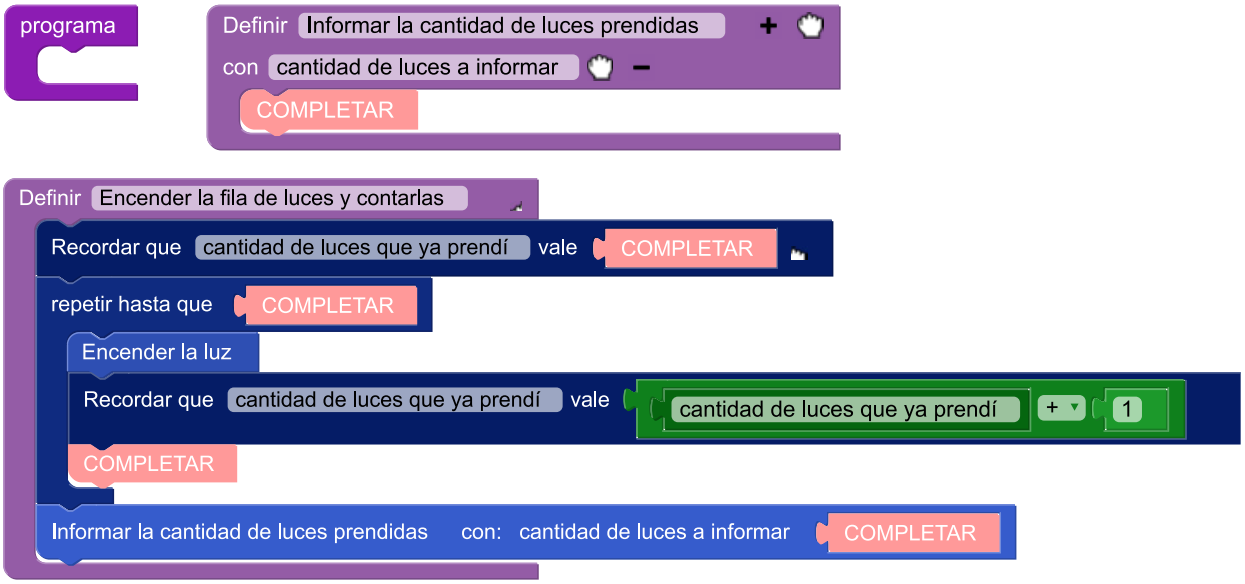
Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Súper Lucho también cuenta luces". Como hicimos al presentar la actividad anterior, les indicamos que presionen reiteradas veces el botón *Ejecutar* para observar que en todos los tableros iniciales Lucho se encuentra al inicio de un camino en el que hay luces en todas las celdas salvo en la última. Además, también notarán que la longitud del camino varía en los distintos escenarios.

El objetivo es que el robot prenda todas las luces y, al llegar a la meta, anote cuántas lamparitas encendió. Para indicar el número hay que poner tantas bolitas rojas como cantidad de lamparitas haya encendido en el camino.



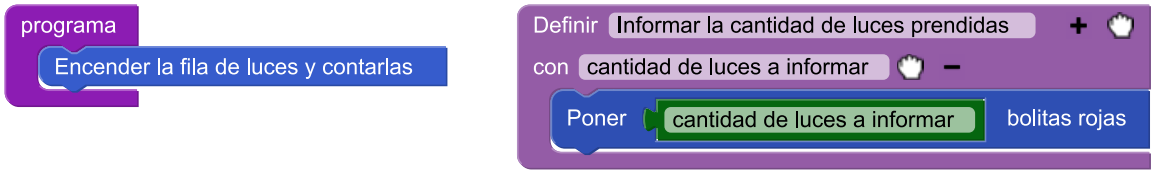
Dos tableros iniciales de "Súper Lucho también cuenta luces"

Al mirar el panel del programa, verán que el cuerpo principal del programa y los procedimientos `Encender la fila de luces y contarlas` e `Informar la cantidad de luces prendidas` están incompletos. Sin embargo, en la biblioteca disponemos de la función `Lucho está en la meta` y los procedimientos `Mover a Lucho al [ ]` y `Poner [ ] bolitas rojas`, que resultarán útiles para resolver el desafío.



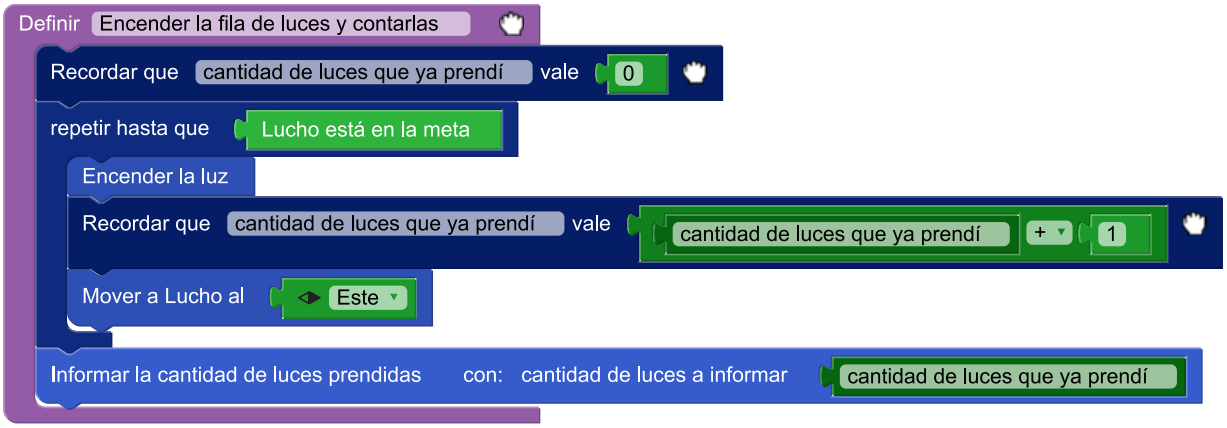
Espacio del programa al cargar el proyecto

El cuerpo principal del programa y el procedimiento `Informar la cantidad de luces prendidas` son los más fáciles de completar. El cuerpo principal del programa consiste en invocar el procedimiento `Encender la fila de luces y contarlas`. En cuanto al procedimiento `Informar la cantidad de luces prendidas`, consiste en poner tantas bolitas rojas como indique el valor recibido en el parámetro `cantidad de luces a informar`. Esto se realiza usando el procedimiento `Poner [ ] bolitas rojas` disponible en *Biblioteca* > *Procedimientos*.



Cuerpo principal del programa y procedimiento `Informar la cantidad de luces prendidas`

En `Encender la fila de luces y contarlas` puede observarse que hay una variable que funciona como contador (es decir que se usa para ir contando una determinada cantidad): `cantidad de luces que ya prendí`. Cada vez que Lucho prenda una luz, se va a incrementar en 1 su valor. Sin embargo, falta asignarle un valor inicial. Como al principio del programa todavía no se ha encendido ninguna luz, comienza con el valor 0. Esta misma variable es la que hay que usar como argumento cuando se invoca `Informar la cantidad de luces prendidas [ ]`, pues, al llegar a ese punto del programa, el valor de la variable será igual a la cantidad total de lamparitas que Lucho haya encendido hasta llegar a la meta. Por su parte, la repetición condicional se tiene que ejecutar hasta que Lucho alcance la bandera a cuadros, para lo que hay que usar la función de la biblioteca `Lucho está en la meta`. Finalmente, para completar el procedimiento, resta indicar qué sucede luego de que Lucho enciende una luz. Esto es, desplazarlo a la siguiente celda del camino invocando `Mover a Lucho al [Este]`.



Procedimiento `Encender la fila de luces y contarlas`

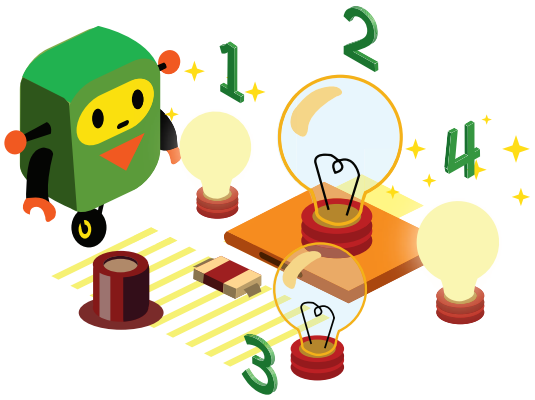
CIERRE

Para concluir, reflexionamos con los estudiantes sobre el uso que les dimos a la variable. En este caso, no usamos una variable solo para recordar un valor. También la fuimos modificando durante la ejecución del programa. Este uso es el que permite, por ejemplo, que en un juego se registre la cantidad de puntos que acumula un jugador, la cantidad de vidas que le quedan, etc.

NOMBRE Y APELLIDO:

CURSO: FECHA:

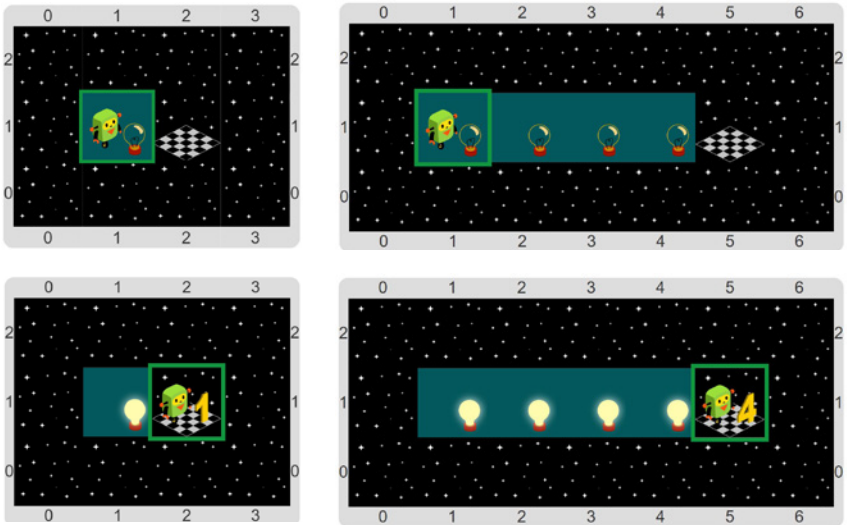
SÚPER LUCHO TAMBIÉN CUENTA LUCES



Ay Luchito, Luchito, ahora no solo tenés que encargarte de encender las luces. ¡También tenés que contarlas!

1. Abrí el proyecto "Súper Lucho también cuenta las luces" y completá el programa para que, una vez que Lucho haya encendido todos los foquitos, informe cuántos prendió.

Mirá dos tableros iniciales con sus correspondientes tableros finales:



NOMENCLATURA

Cuando hay una variable cuyo valor se incrementa durante un recorrido, recibe el nombre de **contador**, porque se usa para ir contando una determinada cantidad.

PARA TENER EN CUENTA

- Hay muchos tableros iniciales distintos. Presioná varias veces el botón *Ejecutar* para verlos.
- En la meta, tenés que poner tantas bolitas rojas como focos haya encendido Lucho. La vestimenta se va a encargar de mostrar el número.

UNA AYUDITA

La variable `cantidad de luces que ya prendí` se usa para ir contando las luces a medida que Lucho las va prendiendo. Pero antes de que el robot comience a encender las lamparitas, ¿cuál es ese valor? Acordate de completarlo.

Actividad 3
¿A qué distancia están los bordes?

DE A DOS

OBJETIVOS

- Presentar las funciones con procesamiento.
- Ejercitar el uso de variables.

MATERIALES

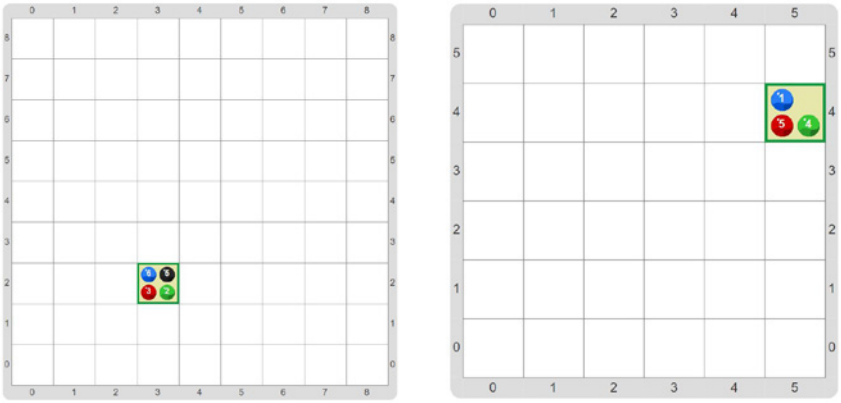
- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

Hasta ahora construimos funciones que nos permiten calcular nuevos valores usando operadores, valores y sensores. Sin embargo, existen contextos en los que hace falta trabajar con valores que son el resultado de acciones más complejas. En esta actividad se trabaja con **funciones con procesamiento**. Estas funciones habilitan la posibilidad de recabar información de cualquier celda del tablero para realizar cálculos.

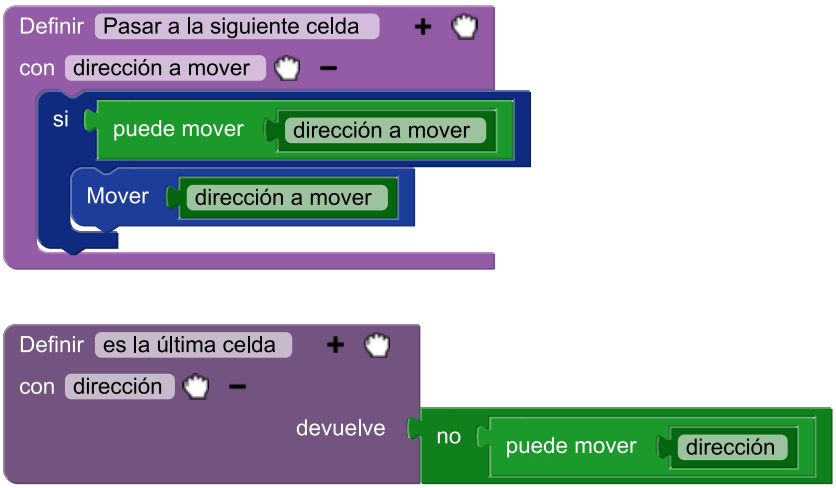
Repartimos la ficha a los estudiantes y los invitamos a abrir el proyecto de Gobstones "¿A qué distancia están los bordes?". Encontrarán un tablero vacío con el cabezal situado en alguna de las celdas. Al hacer clic varias veces sobre el botón *Ejecutar* notarán que en distintas ejecuciones cambian tanto las dimensiones del tablero como la posición inicial del cabezal.

El objetivo es calcular la distancia (medida en cantidad de celdas) a la que se encuentra el cabezal de cada uno de los bordes del tablero. Al finalizar la ejecución del programa, la cantidad de bolitas azules dispuestas en la celda en la que se encontraba el cabezal inicialmente, tiene que indicar la distancia que hay respecto del borde norte; la cantidad de bolitas negras, la distancia respecto del borde este; la cantidad de bolitas verdes, la distancia en relación con el borde sur; y la cantidad de bolitas rojas, la distancia respecto del borde oeste.

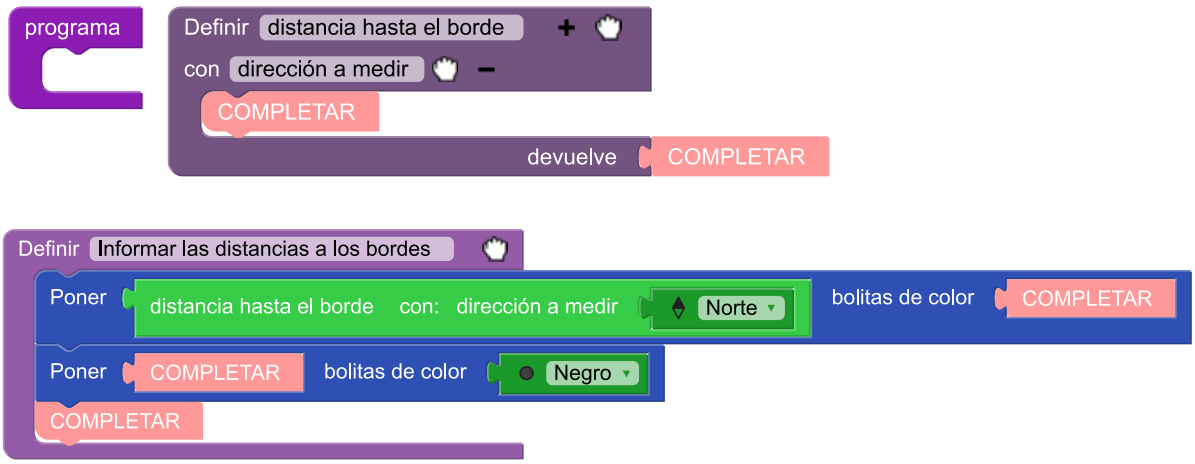


Dos tableros finales del proyecto "¿A qué distancia están los bordes?"

En el panel del programa se puede observar que ya están resueltos el procedimiento `Pasar a la siguiente celda [ ]` y la función `es la última celda`. Hay que completar el cuerpo principal del programa, el procedimiento `Informar las distancias a los bordes` y la función `distancia hasta el borde [ ]`.



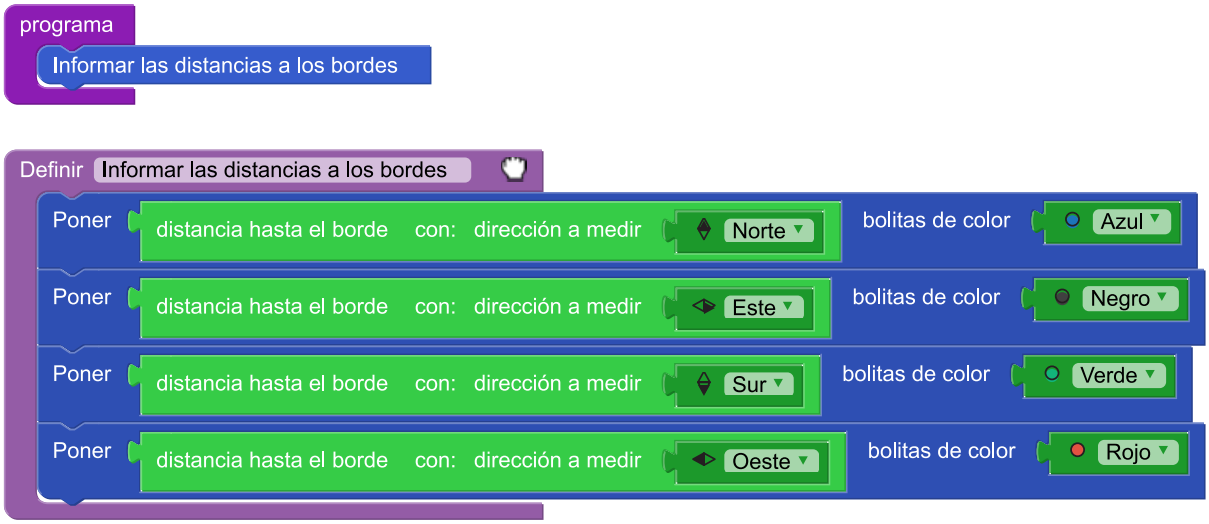
Porción del programa que vienen resuelta



Porción del programa a completar

El cuerpo principal del programa se completa invocando `Informar las distancias a los bordes`. Este último, tal como viene dado, invoca dos veces el procedimiento `Poner [ ] bolitas de color [ ]`, disponible en *Biblioteca* > *Procedimientos*, pero en cada invocación falta un argumento.

En la primera invocación, no está el color de las bolitas con las que se indica la distancia respecto del borde norte; y en la segunda falta la cantidad de bolitas negras que hay que poner. Como hay que indicar la distancia que existe hasta el borde norte con bolitas azules, la primera invocación se completa con el argumento `Azul`. Además, como la cantidad de bolitas negras corresponde a la distancia respecto del borde este, para completar la segunda invocación hay que invocar, a su vez, la función `distancia hasta el borde [ ]`, y usar como argumento `Este`. El procedimiento para comunicar las distancias respecto de los otros dos bordes se completa del mismo modo.

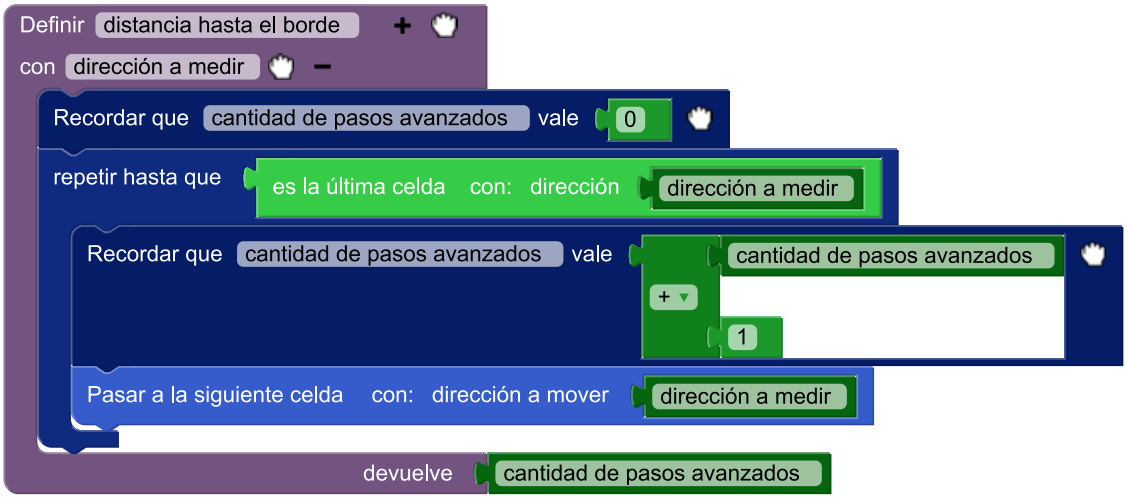


Cuerpo principal del programa y procedimiento `Informar las distancias a los bordes`

Para completar la función `distancia hasta el borde [ ]` los estudiantes van a necesitar nuestra orientación. Al prestar atención, se puede observar que la forma del bloque de esta función difiere de la de los bloques de las funciones usadas hasta ahora. Debajo del nombre hay una pequeña muesca en la que se pueden encastrar bloques, del mismo modo que se hace con, por ejemplo, los procedimientos. Estas funciones, llamadas **funciones con procesamiento**, no solo permiten calcular nuevos valores en los que se realicen cuentas usando operadores y sensores, sino que además posibilitan hacer cálculos inspeccionando otras regiones del tablero, más allá de la celda sobre la que se encuentra posicionado el cabezal.

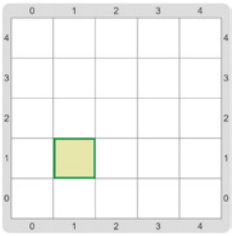
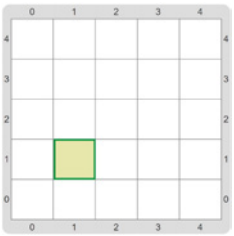
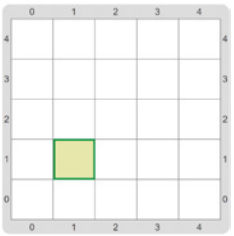
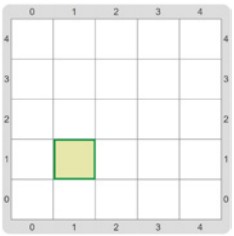
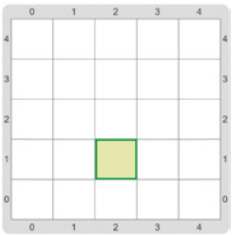
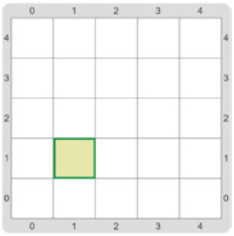
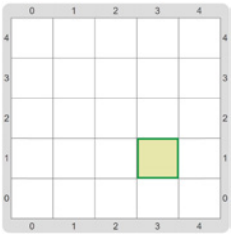
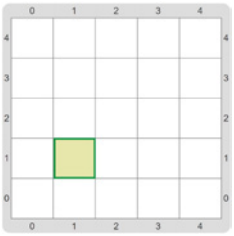
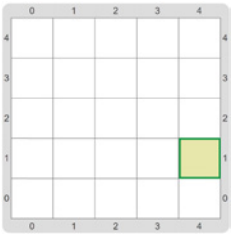
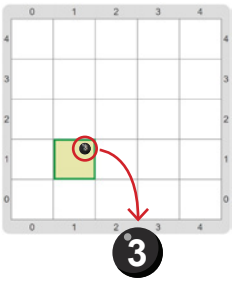
Bloques para definir funciones con y sin procesamientos

Una forma de completar la función con procesamiento `distancia hasta el borde [ ]` consiste en desplazar el cabezal en la dirección indicada por el parámetro `dirección a medir` hasta llegar al borde, e ir acumulando en una variable `cantidad de pasos avanzados` la cantidad de desplazamientos que se realizan. Inicialmente, su valor es cero; cada vez que se desplaza el cabezal, se incrementa en uno su valor. Por último, para determinar si se ha llegado al borde, se puede usar la función `es la última celda [ ]` y, para que avance el cabezal, el procedimiento `Pasar a la siguiente celda [ ]`.



Función con procesamiento `distancia hasta el borde [ ]`

Es importante que los estudiantes comprendan que las funciones con procesamiento no alteran el tablero. Aun cuando en apariencia se ha movido el cabezal, este movimiento no corresponde a un desplazamiento efectivo, sino solo a una operación destinada a realizar un cálculo. Al retornar de la función, el cabezal se encontrará posicionado sobre la misma celda en la que estaba antes de la invocación. Lo mismo sucedería, incluso, si durante la ejecución de la función se hubiesen agregado o quitado bolitas del tablero: esas transformaciones "viven" únicamente mientras la función se ejecuta. Estas funciones trabajan, de algún modo, sobre una copia del tablero que desaparece no bien termina su ejecución.
En la tabla siguiente puede observarse la ejecución paso a paso de una invocación de la función con procesamiento `distancia hasta el borde [ ]` con el argumento `Este`.

	TABLERO DEL PROGRAMA	COPIA DEL TABLERO DE LA FUNCIÓN DE PROCESAMIENTO	VARIABLE *cantidad de pasos avanzados*
Antes de la ejecución de la función como procesamiento		—	—
Durante la ejecución de la función con procesamiento			0
			1
			2
			3
Después de la ejecución de la función de procesamiento		—	—

Ejecución paso a paso de la función con procesamiento `distancia hasta el borde [ ]` con el argumento `Este`

CIERRE

Reflexionamos con los estudiantes sobre las posibilidades que brindan las funciones con procesamiento. Hasta ahora, las funciones calculaban valores a partir del uso de valores literales, operadores y sensores, con los que obteníamos información solo de la celda sobre la que estaba posicionado el cabezal. Las funciones con procesamiento agregan la posibilidad de realizar cómputos a partir de información presente en cualquier celda del tablero.

NOMBRE Y APELLIDO:

CURSO: FECHA:

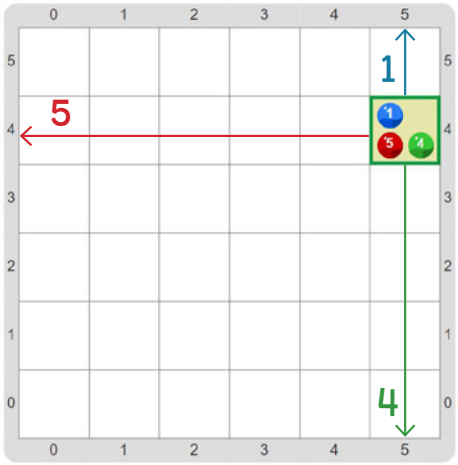
¿A QUÉ DISTANCIA ESTÁ EL BORDE?

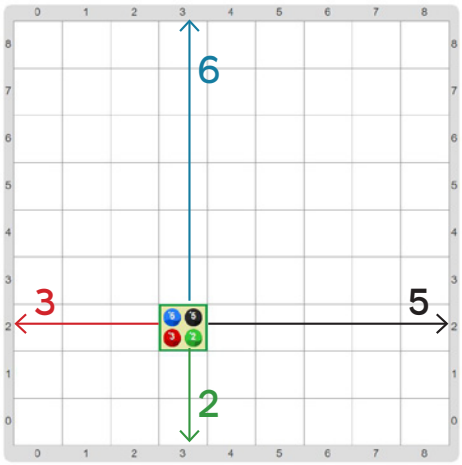
Un poco perdido, el cabezal se pregunta: "¿Dónde estoy?". Vamos a construir un programa para orientar al pobre despistado.

1. Abrí el proyecto "¿A qué distancia está el borde?" y completá el programa para que indique cuántas celdas hay desde donde está posicionado el cabezal hasta cada uno de los bordes. Expresalo con cantidad de bolitas, utilizando un color para cada punto cardinal, según la siguiente tabla.

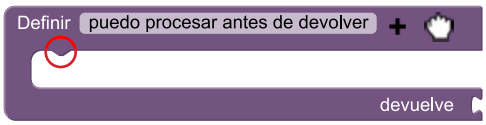
NORTE	ESTE	SUR	OESTE
			

Acá podés ver algunos tableros finales.





FUNCIONES CON PROCESAMIENTO



En Gobstones, tenemos la posibilidad de definir funciones que no solo calculan valores en relación con la celda sobre la que se encuentra el cabezal, sino que también pueden recorrer el tablero libremente para realizar un cómputo. Se llaman **funciones con procesamiento** y, a diferencia de las que usamos hasta ahora, tienen una muesca en la que podés encastrar bloques antes de devolver un valor.

PARA TENER EN CUENTA

- Hay muchos tableros iniciales diferentes.
- Para recordar valores se pueden usar variables.
- Las funciones con procesamiento no alteran el tablero del programa: antes de empezar su ejecución crean una copia del tablero, que deja de existir no bien la función termina.

Contamos las baldosas

DE A DOS

OBJETIVOS

- Ejercitar el uso de variables.
- Poner en práctica el uso de funciones con procesamiento.

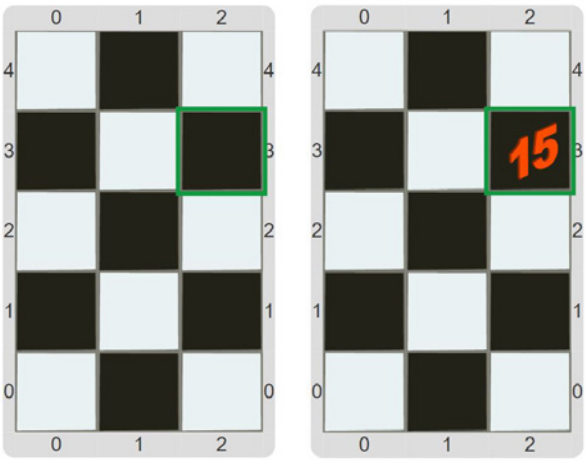
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

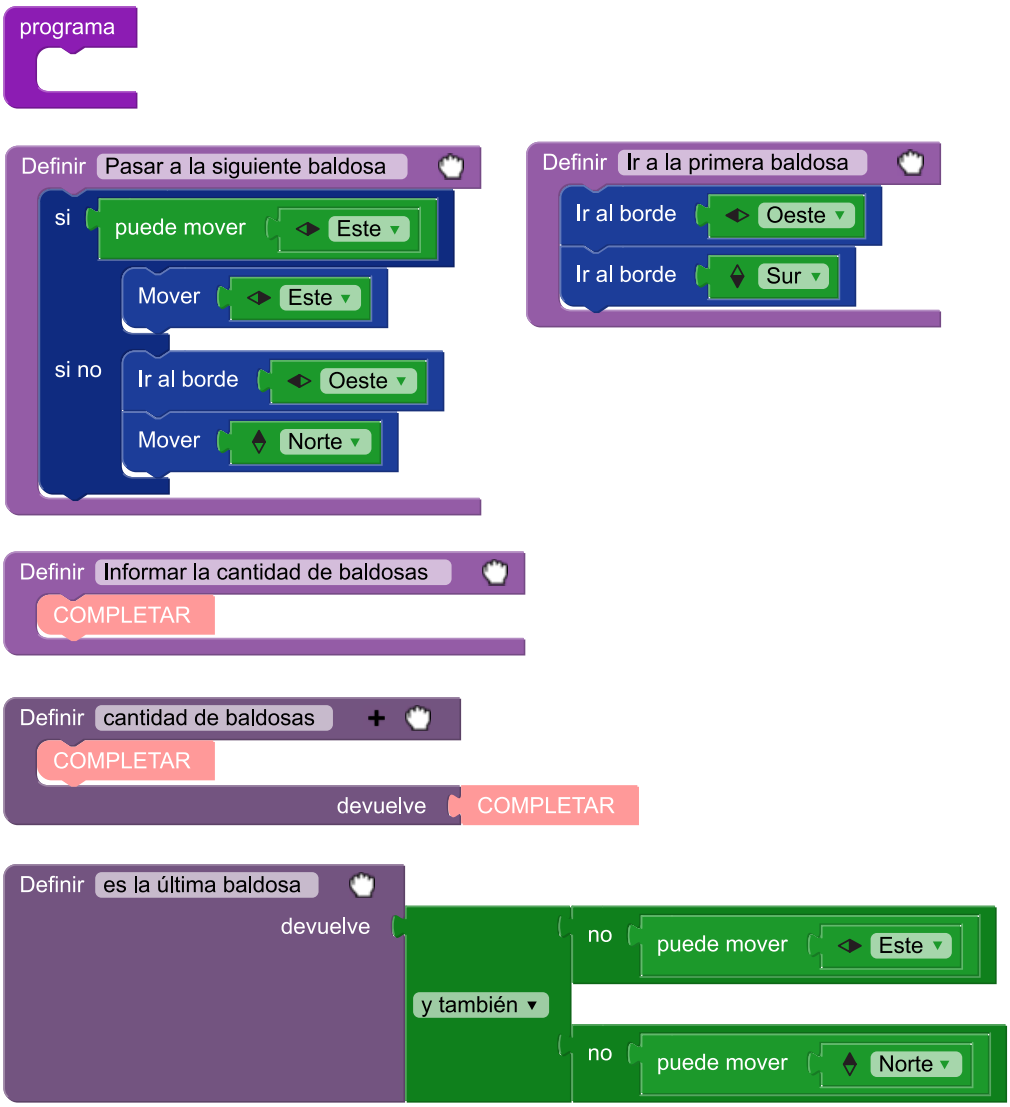
El objetivo de esta actividad es ejercitar el uso de variables y funciones con procesamiento. Tiene dos partes: en la primera deberán contar la cantidad de celdas o baldosas que tiene el tablero; en la segunda, se deberán contar solo las baldosas rajadas.

Les repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Contamos las baldosas". Se encontrarán con un tablero en el que en cada celda hay una baldosa. Al presionar el botón *Ejecutar* varias veces, notarán que hay muchos tableros iniciales que varían en cuanto a dimensiones y posición inicial del cabezal. El objetivo es construir un programa que informe la cantidad de baldosas dispuestas sobre el tablero, colocando esa cantidad de bolitas negras en la celda en la que se encuentra originalmente el cabezal.



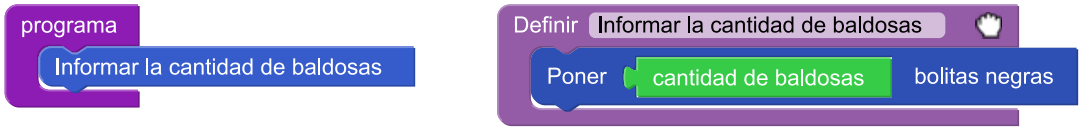
Un tablero inicial y su correspondiente tablero final del proyecto "Contamos las baldosas"

En el panel del programa se puede ver que los procedimientos `Ir a la primera baldosa`, `Pasar a la siguiente baldosa` y la función `es la última baldosa` ya están resueltos. Falta completar el cuerpo principal del programa, el procedimiento `Informar la cantidad de baldosas` y la función con procesamiento `cantidad de baldosas`.



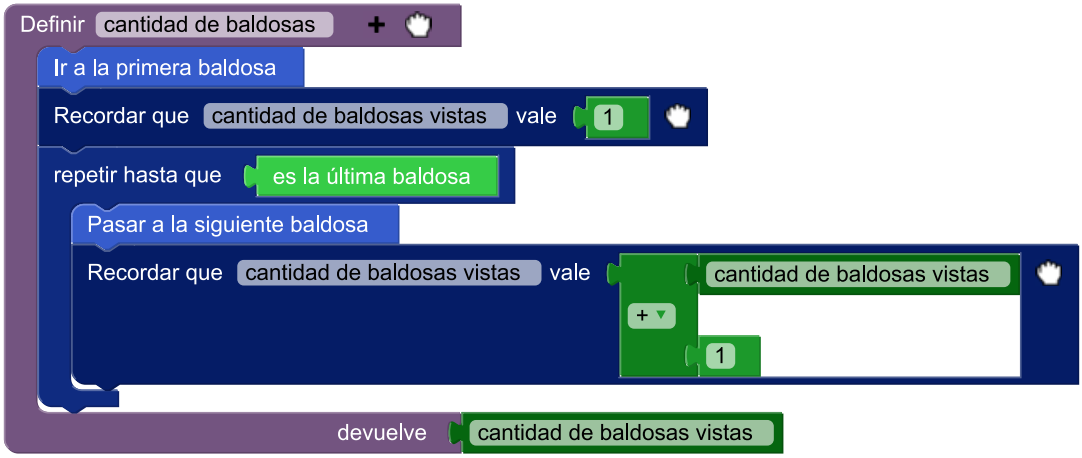
Espacio del programa al cargar el proyecto

El cuerpo principal del programa consiste en la invocación del procedimiento `Informar la cantidad de baldosas`. Este, por su parte, puede resolverse usando el procedimiento `Poner [ ] bolitas negras` (disponible en *Biblioteca* > *Procedimientos*), poniendo tantas bolitas negras como el valor devuelto por la función con procesamiento `cantidad de baldosas`. Por tratarse de una función con procesamiento, al retornar, el cabezal seguirá posicionado sobre la misma celda en la que estaba al comenzar la ejecución del programa.



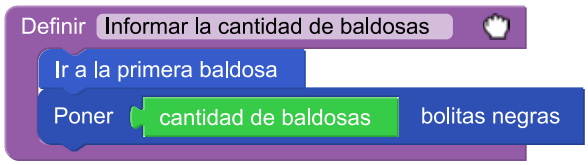
Cuerpo principal del programa y procedimiento `Informar la cantidad de baldosas`

Para completar `cantidad de baldosas` hay que recorrer todas las celdas del tablero e ir contabilizando cada baldosa visitada. Lo primero, entonces, es posicionar al cabezal sobre la primera baldosa del recorrido invocando el procedimiento `Ir a la primera baldosa`, que ya está resuelto. Para llevar la cuenta de las baldosas visitadas hace falta una variable, que puede llamarse `cantidad de baldosas vistas`. Antes de mover el cabezal por primera vez, solo habrá estado sobre el primer mosaico, por lo que hay que inicializar la variable con el valor `1`. Luego, se avanzará a la próxima baldosa (utilizando el procedimiento `Pasar a la siguiente baldosa`) y, dado que habremos visto una baldosa nueva, se deberá incrementar en 1 el valor de la variable `cantidad de baldosas vistas`. Para repetir esta tarea para todas las baldosas, colocamos estos bloques dentro de una repetición condicional utilizando como condición la función `es la última baldosa`.



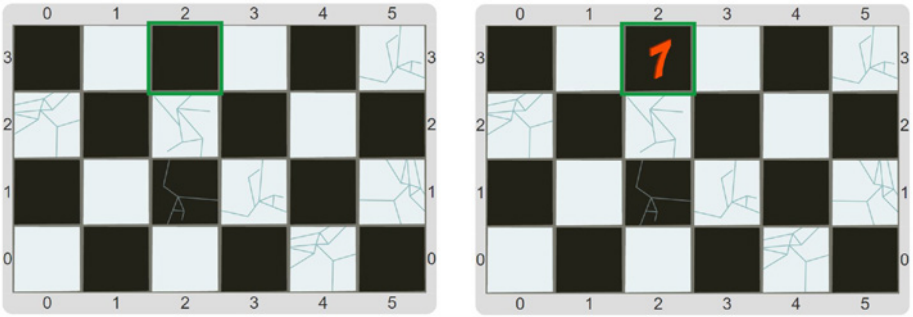
Función con procesamiento `cantidad de baldosas`

Puede ocurrir que algún estudiante proponga poner el cabezal sobre la primera baldosa al comienzo del procedimiento `Informar la cantidad de baldosas`. Sin embargo, a diferencia de lo que ocurre con las funciones con procesamiento que trabajan con una copia del tablero, en este caso se perdería la posición inicial del cabezal. Por lo tanto, la cantidad de baldosas se terminaría registrando sobre la primera baldosa del recorrido –la de la esquina suroeste, por el modo en que está programado `Ir a la primera baldosa`–, en lugar de donde comienza posicionado el cabezal.



Implementación errónea de `Informar la cantidad de baldosas`

Una vez que terminen sus programas, les indicamos a los estudiantes que abran el proyecto "Contamos baldosas rajadas". El objetivo es similar al del proyecto anterior, aunque en esta oportunidad no hay que contar todas las baldosas, sino solo aquellas que tienen rajaduras.



Un tablero inicial y su correspondiente tablero final del proyecto "Contamos baldosas rajadas"

Este proyecto trae en la biblioteca la función `es una baldosa rajada`, que devuelve verdadero si el cabezal está sobre una baldosa rajada y falso en caso contrario. En este desafío hay que aumentar el valor de la variable (que puede llamarse `cantidad de baldosas rajadas vistas`) solo en el caso de que el cabezal esté efectivamente sobre una baldosa rajada.

```
programa
    Informar la cantidad de baldosas rajadas

Definir Informar la cantidad de baldosas rajadas
    Poner cantidad de baldosas rajadas bolitas negras

Definir cantidad de baldosas rajadas
    Recordar que cantidad de baldosas rajadas vistas vale 0
    Ir a la primera baldosa
    repetir hasta que es la última baldosa
        si es una baldosa rajada
            Recordar que cantidad de baldosas rajadas vistas vale cantidad de baldosas rajadas vistas + 1
        Pasar a la siguiente baldosa
    si es una baldosa rajada
        Recordar que cantidad de baldosas rajadas vistas vale cantidad de baldosas rajadas vistas + 1
    devuelve cantidad de baldosas rajadas vistas
```

Programa para contar baldosas rotas

CIERRE

Al finalizar, discutimos con los estudiantes sobre las consecuencias de combinar el uso de variables con alternativas condicionales. Para iniciar la discusión, podemos plantearles: "Supongamos que estamos jugando a un juego en donde hay que embocar una pelota en un aro. ¿Qué pasaría con el puntaje si no verificamos que la pelota haya pasado por el aro? ¿Cómo distinguimos un acierto de una pifia?".

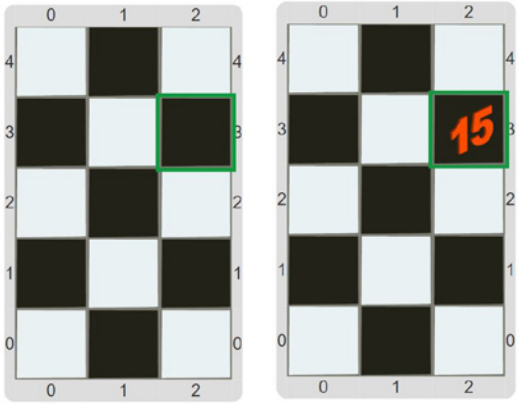
NOMBRE Y APELLIDO:

CURSO: FECHA:

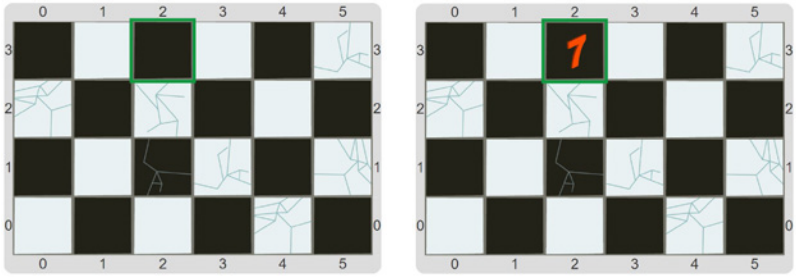
CONTAMOS BALDOSAS

¿Alguna vez te tropezaste con una baldosa rota? ¿Nunca te pasó que un día lluvioso pisaste una y te empapaste el pantalón? Es hora de que cuentes cuántas baldosas rotas hay y se lo reclames a la intendencia de tu municipio.

1. En un mundo ideal, no hay baldosas rotas. Abrí el proyecto "Contar las baldosas" y completá el programa para contar cuántas baldosas hay en el piso. Tenés que dejar el resultado escrito sobre el mosaico donde aparece inicialmente el cabezal. Acá te mostramos un tablero inicial y el tablero final correspondiente.



2. El mundo ideal no existe. A veces más, a veces menos, siempre hay baldosas rotas. Abrí el proyecto "Contamos baldosas rajadas". Ahora tenés que informar solo la cantidad de baldosas dañadas. Con el resto, no hay drama. ¿Cuál es la diferencia entre las soluciones de los dos proyectos?



_____

_____

PARA TENER EN CUENTA

- Hay muchos tableros iniciales. Podés ver algunos presionando varias veces el botón *Ejecutar*.
- Para escribir un número en una baldosa tenés que poner tantas bolitas negras como el número que quieras escribir.
- En la biblioteca vas a encontrar algo que te va a servir.

UNAS AYUDITAS

- Para contar vas a necesitar una variable.
- Pensá cómo hacer para recorrer todo el tablero sin perder registro de la posición inicial del cabezal.
- Gran parte del programa ya está resuelto. Fijate cómo podés usar los procedimientos `Ir a la primera baldosa`, `Pasar a la siguiente baldosa` y la función `es la última baldosa`.

07

INTERACTIVIDAD

SECUENCIA DIDÁCTICA 1:
PROGRAMAS INTERACTIVOS
Editor de tableros
Editor de textos simple

SECUENCIA DIDÁCTICA 2:
VIDEOJUEGOS
Definición por penales
Snake

En muchos de los programas que habitualmente utilizamos existe una interacción entre el programa y quien lo usa: presionamos las teclas y vemos que aparece texto en la pantalla, movemos el ratón y vemos que un puntero se desplaza, etc. En este capítulo se trabaja con los programas de esta clase, que se conocen como **programas interactivos**.

El capítulo consta de dos secuencias didácticas. La primera presenta la programación interactiva a través de dos proyectos simples. La segunda propone la programación de dos videojuegos que, además, permiten poner en práctica muchas de las herramientas estudiadas en capítulos anteriores.

Secuencia Didáctica 1

PROGRAMAS INTERACTIVOS

Muchos de los programas que se usan cotidianamente, como los procesadores de texto o los juegos, interactúan con el usuario. Por eso se los conoce como **programas interactivos**. En esta secuencia didáctica se presentan dos actividades para construir programas interactivos en Gobstones.

OBJETIVOS

- Comprender la noción de interactividad.
- Construir programas interactivos.

Actividad 1
Editor de tableros

DE A DOS

OBJETIVO

- Presentar programas interactivos.

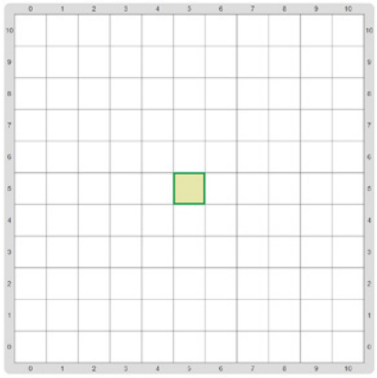
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

El objetivo de esta actividad es presentar los programas interactivos. En primer lugar, los estudiantes tienen que completar un programa en el que, al ejecutarlo, el usuario pueda desplazar el cabezal por el tablero usando las flechas del teclado. Luego, deben extenderlo para que, además, se pueda colocar y retirar bolitas de colores de las celdas.

Para comenzar, repartimos la ficha a los estudiantes, les pedimos que abran el proyecto de Gobstones "Editor de tableros" y resuelvan la primera consigna. Van a encontrar un tablero vacío de 11 × 11 con el cabezal ubicado en el centro. Los invitamos a que prueben el programa y descubran qué hace. Luego, realizamos una puesta en común para llegar a la conclusión de que, al ejecutar el programa, no sucede nada hasta que no se presionan las flechas derecha → o izquierda ← del teclado y que, al hacerlo, el cabezal se mueve en el sentido de las flechas.

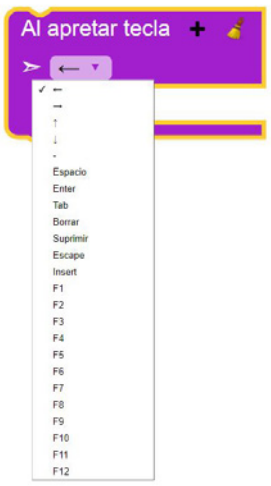


Tablero inicial de la actividad

En el panel del programa puede verse que, en lugar del bloque `programa`, hay uno llamado `programa interactivo`. Los bloques que se encastran dentro definen cómo reacciona el programa frente a ciertos eventos producidos por un usuario. En este caso, al combinar `Al apretar tecla ←` con `Mover [Oeste]`, se establece que cuando el usuario presiona la flecha izquierda, el cabezal se desplaza una celda en esa dirección; del mismo modo, al combinar `Al apretar tecla →` con `Mover [Este]`, se determina que, al apretar la flecha derecha, el cabezal se mueve hacia la derecha.

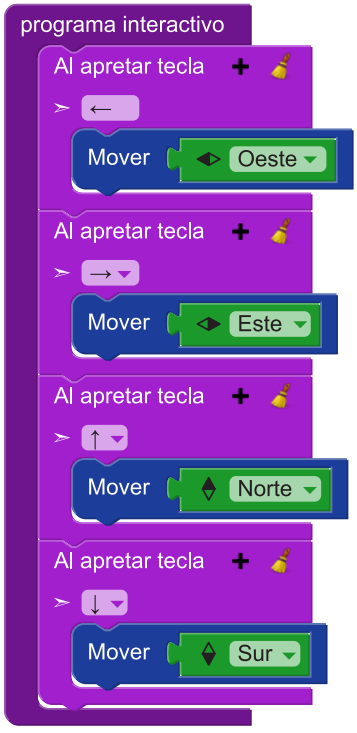
Programa interactivo al cargar el proyecto

Les pedimos que modifiquen el programa para que también sea posible desplazar el cabezal hacia arriba y hacia abajo con las flechas correspondientes (↑ y ↓, respectivamente). Para agregar instrucciones interactivas hay que ir a *Definiciones* > *Eventos* y elegir el bloque `Al apretar tecla [ ]`. Al usarlo, hay que indicar cuál es la tecla que, al ser presionada, genera una respuesta del programa. Además, los bloques que se encastren dentro especifican cómo debe ser la respuesta.



Bloque `Al apretar tecla [ ]`

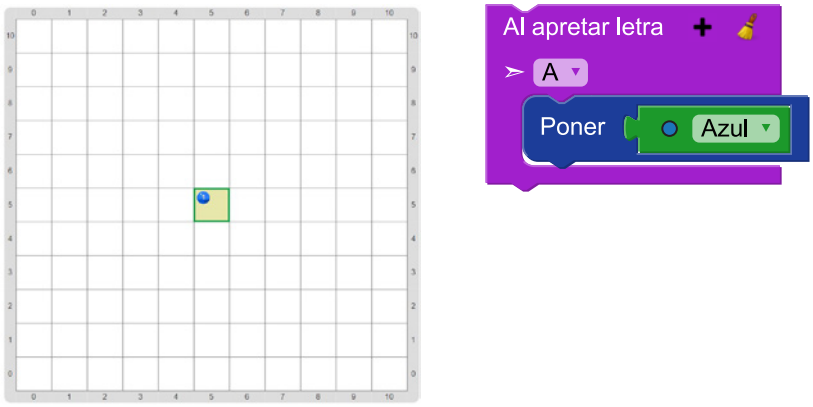
Para completar la consigna, hay que agregar dos bloques `Al apretar tecla [ ]`: uno para la flecha de arriba y otro para la de abajo. Luego, dentro de cada uno, hay que encastrar `Mover [Norte]` y `Mover [Sur]` respectivamente. De este modo, el usuario podrá mover al cabezal en las cuatro direcciones.



Programa para mover el cabezal en las cuatro direcciones

Una vez que todos hayan concluido, hacemos una puesta en común y les indicamos que resuelvan la segunda consigna. En este caso, tienen que extender el programa para que, además de mover el cabezal, sea posible poner bolitas de todos los colores usando las teclas *A*, *N*, *R* y *V* –por azul, negro, rojo y verde– y sacarlas usando las mismas teclas junto con *SHIFT* (en el teclado español corresponde a "Mayús.").

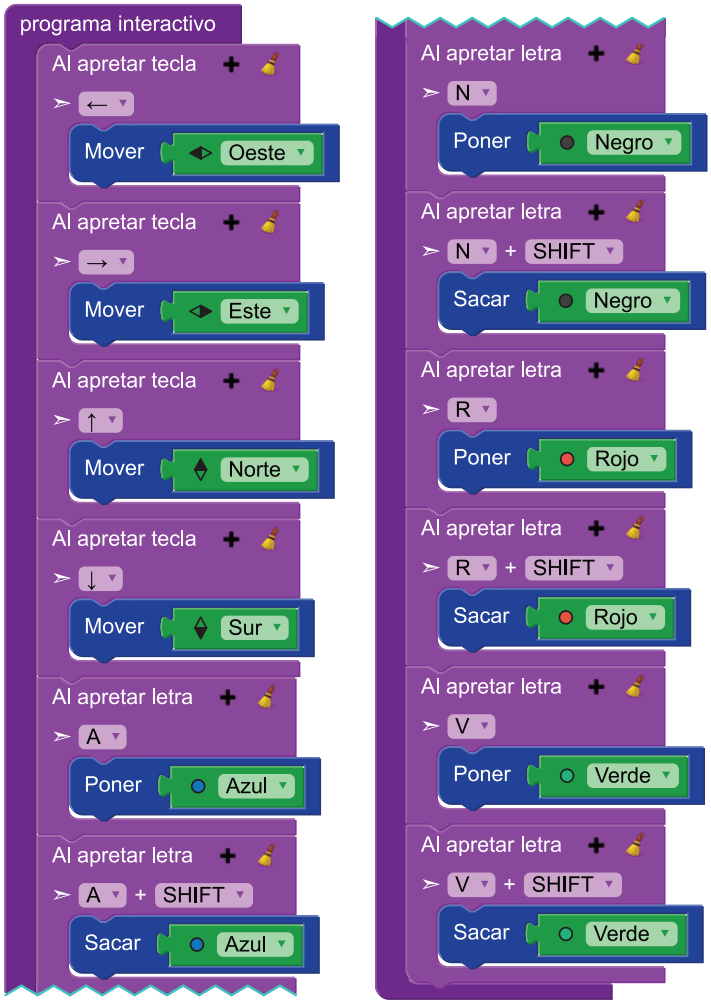
En *Definiciones* > *Eventos*, se encuentra el bloque `Al apretar letra [ ]`, que se diferencia de `Al apretar tecla [ ]` en las teclas que permite seleccionar: en este caso, hay que optar por una de las 27 letras del abecedario. Para resolver cómo poner bolitas azules, hay que encastrar `Poner [Azul]` dentro de `Al apretar letra [A]`. Para el resto de los colores, la solución sigue la misma lógica.



Al apretar la A se deposita una bolita azul sobre el tablero

Para poder sacar bolitas, tenemos que combinar las letras *A*, *N*, *R* y *V* con *SHIFT*. *SHIFT* no es una de las opciones para `Al apretar letra [ ]`. Para usarla, hay que hacer clic en el *+* que se encuentra justo a la derecha del texto *Al apretar letra*. Entonces, aparece la opción de seleccionar *SHIFT*, *CTRL* o *ALT* como tecla adicional que también hay que presionar para que el programa reaccione de algún modo. Siguiendo con el ejemplo, al encastrar `Sacar [Azul]` dentro de `Al apretar letra [A] + [SHIFT]`, se consigue el efecto buscado. Lo mismo debe hacerse con los demás colores

Al apretar *A* + *SHIFT* se retira una bolita azul del tablero



Programa que resuelve la actividad

CIERRE

Les pedimos a los alumnos que mencionen algunos programas interactivos que usan en su vida diaria. Es probable que nombren los juegos, los navegadores de Internet y los editores de texto como programas de esta clase. Para cada programa mencionado, les pedimos que comenten cuáles son las acciones del usuario que desencadenan respuestas del programa, y en qué consisten estas últimas. Por ejemplo, los juegos permiten controlar personajes que aparecen en la pantalla utilizando el ratón, el teclado o la pantalla táctil, y los procesadores de texto hacen aparecer letras en la pantalla cuando el usuario presiona las teclas.

NOMBRE Y APELLIDO:

CURSO: FECHA:

EDITOR DE TABLEROS

Al jugar en la compu, podemos mover un personaje por la pantalla usando un ratón, un teclado o una pantalla táctil; al escribir en un procesador de texto, cuando presionamos las teclas de las letras, las vemos aparecer en el monitor. ¿Notaste que en estos casos estamos interactuando con los programas? Nosotros hacemos algo y el programa responde.

1. Abrí el proyecto "Editor de tableros" y ejecutá el programa.
¿Qué diferencias observás con respecto a otros programas de Gobstones?

Extendé el programa de modo que, mediante el uso de las teclas ⬆ y ⬇, se pueda desplazar el cabezal hacia arriba y hacia abajo.

2. Extendé el programa de la consigna anterior para que, además, se pueda poner y sacar bolitas de las celdas. Tené en cuenta lo siguiente:

- que al presionar la letra *A* se agregue una bolita **azul** en la celda bajo el cabezal;
- que al presionar la *A* + *SHIFT* se retire una bolita **azul**;
- que al presionar la *N* se agregue una bolita **negra**;
- que al presionar la *N* + *SHIFT* se retire una bolita **negra**;
- que al presionar la *R* se agregue una bolita **roja**;
- que al presionar la *R* + *SHIFT* se retire una bolita **roja**;
- que al presionar la *V* se agregue una bolita **verde**; y
- que al presionar la *V* + *SHIFT* se retire una bolita **verde**.

¡Explorá el entorno en busca de nuevos bloques que te permitan resolver la consigna!

PROGRAMAS INTERACTIVOS

Los programas interactivos son aquellos en los que hay una interacción entre el programa y quien lo usa. Frente a ciertos eventos que genera el usuario, el programa reacciona. Casi todos los programas que usamos habitualmente entran en esta categoría.

Actividad 2
Editor de texto

DE A DOS

OBJETIVO

- Ejercitar la construcción de programas interactivos.

MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

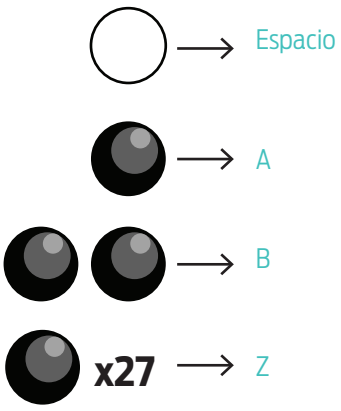
DESARROLLO

El objetivo de esta actividad es ejercitar la construcción de programas interactivos. Con ese fin, se programará un sencillo editor de textos recuperando algunas ideas sobre representación de la información.

Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Editor de texto". Se van a encontrar con un tablero vacío de 15 × 10, que hace las veces del documento en el que van a escribir texto. El propósito es que, en este editor, se pueda mover el cursor en las cuatro direcciones usando las flechas del teclado, escribir en mayúscula cualquiera de las 27 letras del abecedario, ingresar un espacio y un salto de línea (o *Enter*).

Tablero inicial del proyecto "Editor de texto"

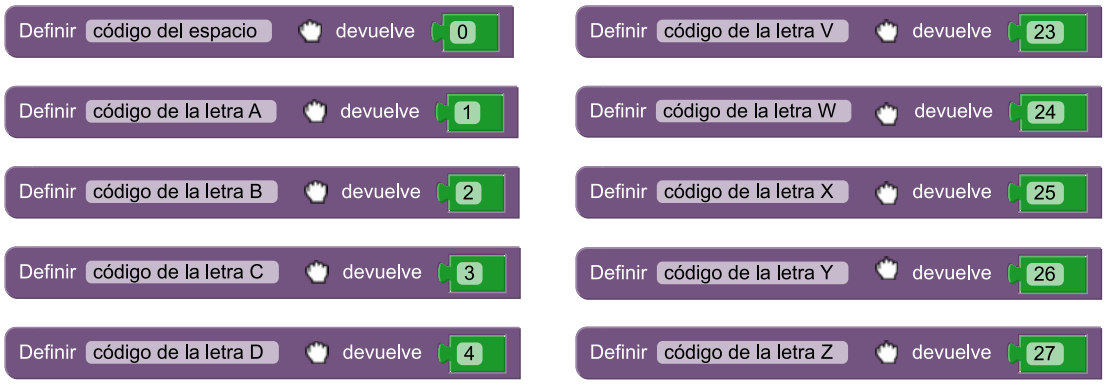
La vestimenta de la actividad usa bolitas negras para representar cada uno de los símbolos que se pueden imprimir en el documento: 0 negras representa el espacio, 1 la *A*, 2 la *B*, y así hasta la *Z*, que se representa con 27.



En el espacio del programa, notarán que gran parte del desafío ya está resuelto. En el cuerpo principal –todos los bloques encastrados dentro de `programa interactivo`–, ya están las vinculaciones de *(i)* las teclas →, ←, ↑ y ↓ con el desplazamiento del cursor hacia el este, el oeste, el norte y el sur respectivamente –invocando `Mover si se puede [ ]`, aún no implementada–; *(ii)* el botón *ENTER* con el salto a una nueva línea –invocando el procedimiento `Nueva línea`, aún no implementado–; y *(iii)* la barra espaciadora y las letras de la *A* a la *U* con la aparición en pantalla de estos caracteres –invocando el procedimiento `Poner el carácter [ ]` cuyo argumento, en cada caso, es una llamada a una función que devuelve la cantidad de bolitas que se emplean para representar el carácter en cuestión–. Para completar el cuerpo principal, falta agregar los casos de las letras de la *V* a la *Z* con el mismo criterio.

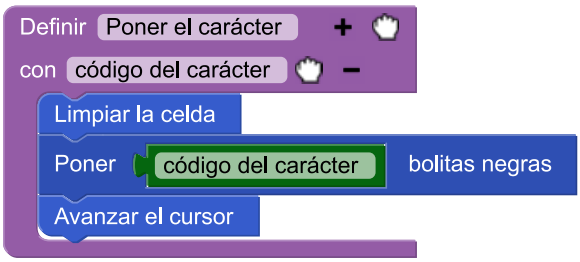
Porciones del cuerpo principal del programa interactivo

Las funciones `código del espacio`, `código de la letra A`, ... y `código de la letra U` ya están resueltas en el proyecto. Tomándolas como referencia y sabiendo cómo se representan las letras, resulta sencillo completar las funciones correspondientes a las letras de la *V* a la *Z*.



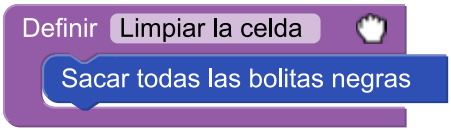
Funciones que devuelven el código de algunas letras del abecedario

Para completar `Poner el carácter [ ]` hay que *(i)* retirar las bolitas negras que pudiera haber en la celda bajo el cabezal –invocando el procedimiento `Limpiar la celda`, aún no implementado–, *(ii)* colocar las bolitas negras que representan la letra que se quiere escribir –usando el procedimiento de la biblioteca `Poner [ ] bolitas negras` junto con el parámetro `código del carácter`– y *(iii)* desplazar el cabezal a la siguiente posición –usando el procedimiento `Avanzar el cursor`, aún no implementado–.



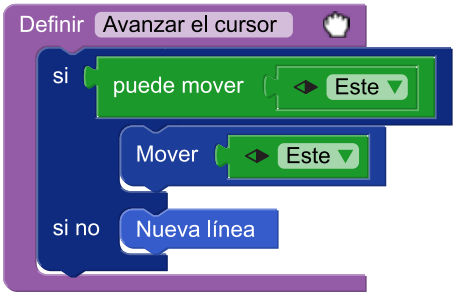
Procedimiento `Poner el carácter [ ]`

El procedimiento `Limpiar la celda` se resuelve retirando todas las bolitas negras que se encuentren bajo el cabezal. En este caso, hay que usar `Sacar todas las bolitas negras`, disponible en *Biblioteca > Procedimientos*.



Procedimiento `Limpiar la celda`

Para determinar dónde hay que ubicar el cabezal cuando se invoque `Avanzar el cursor`, es necesario chequear si hay celdas a la derecha del cabezal –en ese caso, hay que moverlo en dirección este– o no –en ese caso, hay que posicionarlo al comienzo de una nueva línea–. Esto se resuelve usando una alternativa condicional, el sensor `puede mover [ ]`, el comando básico `mover [ ]` y el procedimiento `Nueva línea`.

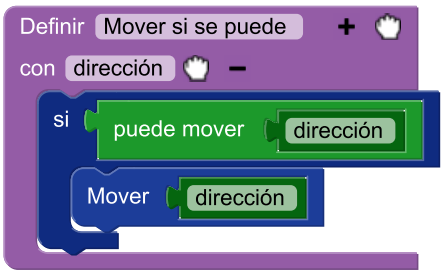


Procedimiento `Avanzar el cursor`

Para completar `Nueva línea`, si el cabezal no se encuentra en la última fila hay que desplazarlo hacia el sur y, en cualquier caso, moverlo hacia el borde oeste. Se resuelve usando el procedimiento `Mover si se puede [ ]` y el comando básico `Ir al borde [ ]`.

Procedimiento `Nueva línea`

Para completar el programa, solo falta programar `Mover si se puede [ ]`, que antes de mover el cabezal hacia la dirección indicada por el parámetro `dirección`, chequea que sea posible hacerlo.



Procedimiento `Mover si se puede [ ]`

CIERRE

Concluimos la actividad reflexionando junto con los estudiantes acerca del hecho de que muchos de los programas que utilizamos, como los editores de texto, son programas interactivos. En combinación con las herramientas adquiridas en los capítulos anteriores, la interactividad nos permite crear este tipo de programas. Si bien los editores reales manejan muchas más opciones, la manera en la que interactúan con los usuarios, asociando diferentes acciones a la pulsación de ciertas teclas, es similar a la del programa que realizaron en esta actividad.

NOMBRE Y APELLIDO:

CURSO: FECHA:

EDITOR DE TEXTO

Muchas veces escribiste textos en la computadora, ¿no? ¡Ahora vas a armar un programa para que otros puedan escribir en la computadora lo que más les guste!

1. Abrí el proyecto "Editor de texto" y completá el programa para transformar el tablero de Gobstones en un documento para escribir.

PARA TENER EN CUENTA

- El cursor se encuentra en la celda bajo el cabezal.
- El programa tiene que permitir:
  - Mover el cursor con las flechas.
  - Escribir las 27 letras del abecedario.
  - Dejar un espacio cuando se presiona la barra espaciadora.
  - Hacer un salto de línea cuando se aprieta ENTER.
  - Reemplazar un carácter por otro.
- Si se presiona la barra espaciadora o una letra, se tiene que reemplazar el contenido de la celda en la que está el cursor por el símbolo correspondiente a la tecla presionada.
- Cada vez que escribas una letra, el cursor tiene que desplazarse a la siguiente posición. Si no puede seguir hacia la derecha, debe pasar a la siguiente línea. Y si ya no hay una línea abajo… ¡Tené cuidado de que no se caiga del tablero!

Secuencia Didáctica 2

VIDEOJUEGOS

Los videojuegos son programas interactivos en los que uno o más jugadores controlan una parte del comportamiento de la computadora. Al mover un joystick o presionar teclas, desplazan personajes, efectúan disparos, mueven piezas de un juego de mesa, etc. Una acción de los jugadores genera una respuesta del programa.

En esta secuencia didáctica se propone la programación de juegos que integran las distintas herramientas estudiadas. El primero es un juego de fútbol en el que un jugador y un arquero se enfrentan en la definición de un partido por penales. El segundo es una versión en Gobstones del difundido *Snake*.

OBJETIVOS

- Desarrollar videojuegos sencillos que integren las herramientas aprendidas.
- Incorporar *timeout* en programas interactivos.

Actividad 1
Definición por penales

DE A DOS

OBJETIVO

- Programar un juego interactivo.

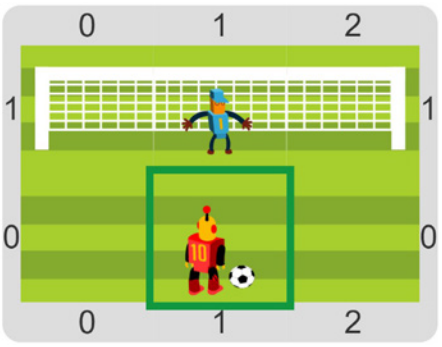
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

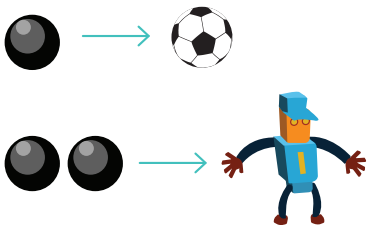
DESARROLLO

En esta actividad, los estudiantes ejercitarán la construcción de programas interactivos programando un juego de ejecución de penales.

Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Definición por penales". Encontrarán un tablero de 3 × 2. En la fila superior hay un arco y un arquero, y en la inferior, un jugador y una pelota. Al presionar varias veces el botón *Ejecutar* notarán que este es el único tablero inicial. A lo largo de la actividad, será importante que tengan presente que la vestimenta representa a la pelota con una bolita negra y al arquero dos bolitas negras.



Tablero inicial al cargar el proyecto



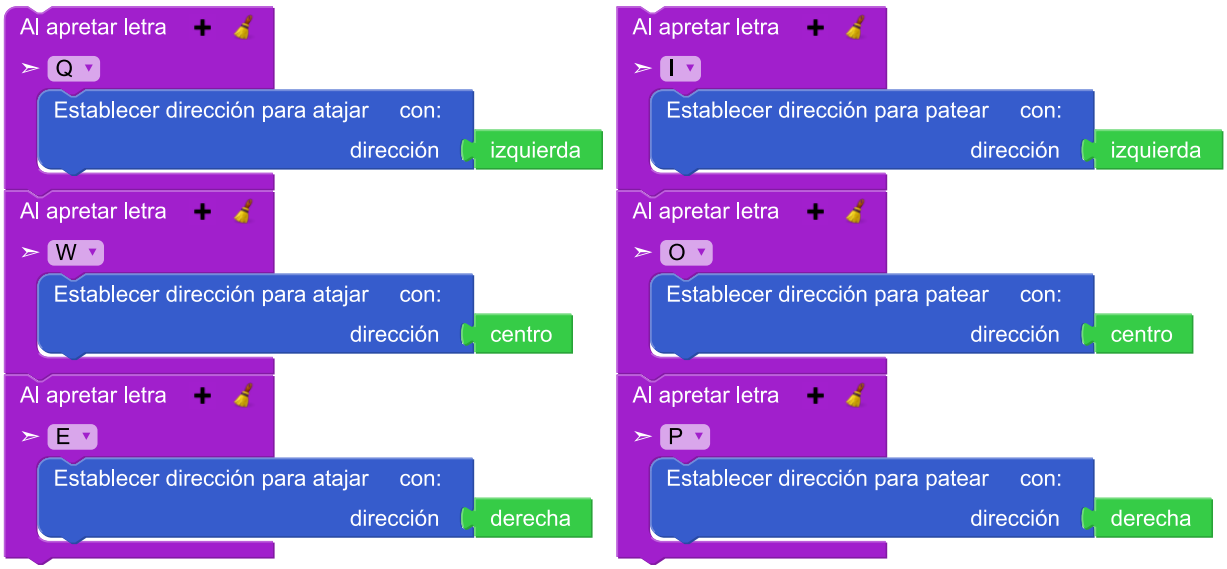
Vestimenta de la actividad

El objetivo es armar un juego para 2 participantes: un jugador será el goleador y el otro, el arquero. El jugador que ocupe el rol del goleador podrá elegir patear la pelota a la izquierda, al centro o a la derecha usando las teclas *I*, *O* y *P*, respectivamente. Por su parte, quien comande al arquero tendrá la posibilidad de hacerlo volar hacia la izquierda, de dejarlo parado o de hacerlo volar hacia la derecha usando las teclas *Q*, *W* y *E*. Cada jugador, al presionar una tecla, no debe ser visto por su contrincante.[1] Una vez que ambos hayan seleccionado una opción, se usará la barra espaciadora para ejecutar el penal. Además, la tecla *Enter* reiniciará el juego, de modo que sea posible patear muchos penales en una misma ejecución del programa.

Una forma de resolver el desafío consiste en comenzar programando el cuerpo principal, donde hay que indicar qué debe hacer el programa cuando se presionan determinadas teclas. Por ejemplo, si se presiona la *Q*, hay que registrar que el arquero se tirará hacia la izquierda cuando se ejecute el penal; si se aprieta la *O*, que el jugador intentará convertir el gol pateando al centro del arco. En el panel del programa encontrarán dos procedimientos (que hay que completar) que sirven

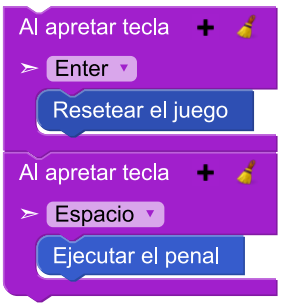
[1] Pueden, por ejemplo, ponerse de espaldas.

para este propósito: `Establecer dirección para atajar [ ]` y `Establecer dirección para patear [ ]`. El parámetro, en ambos casos, sirve para indicar una dirección. El proyecto tiene incorporadas las funciones `izquierda` (ya resuelta) y `centro` (que hay que completar). Siguiendo el mismo criterio, se puede crear una función `derecha` que permita definir todas las opciones para atajar y patear.



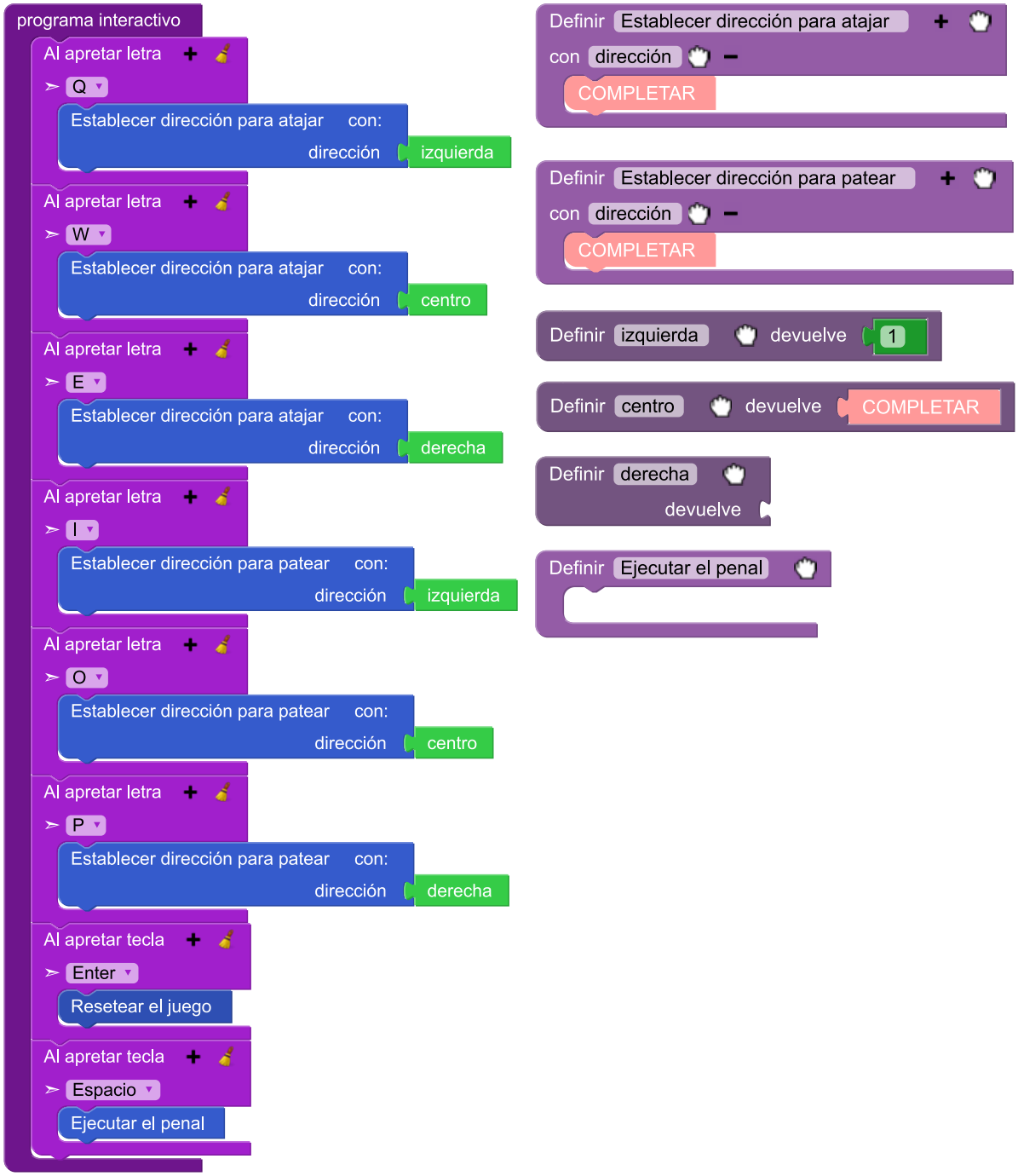
Opciones para atajar y patear

Para establecer los vínculos con *Enter* y la barra espaciadora, puede usarse el procedimiento `Resetear el juego` disponible en *Biblioteca* > *Procedimientos* y crear otro que se llame `Ejecutar el penal`.



Acciones para *Enter* y la barra espaciadora

A continuación puede observarse el cuerpo principal del programa y los procedimientos y funciones que invoca y usa: *(i)* los que están resueltos, *(ii)* los que hay que completar y *(iii)* los creados por nosotros.



Definición del cuerpo principal del programa interactivo

La opción elegida por cada jugador se codifica depositando bolitas verdes y azules en la celda (1,0), que es aquella en la que inicialmente se encuentra el cabezal. Una bolita azul indica que el jugador pateará a la izquierda; dos azules, que pateará al centro y tres azules, a la derecha; una bolita verde indica que el arquero volará a la izquierda; dos verdes, que se quedará en el medio del arco y tres verdes, que volará a la derecha.

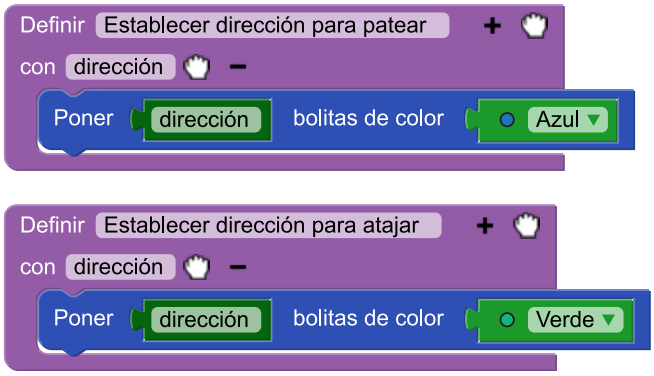
Codificación de acciones con bolitas

Personaje	Cantidad y color de bolitas	Acción
	●	Patear a la izquierda
	● ●	Patear al centro
	● ● ●	Patear a la derecha
	●	Volar a la izquierda
	● ●	Quedarse parado
	● ● ●	Volar a la derecha

Teniendo en cuenta la codificación de las acciones con bolitas y la función `izquierda` (ya resuelta en el proyecto), podemos completar las funciones `centro` y `derecha`. Cada una devuelve el número de bolitas que hay que colocar en (1,0) para indicar una dirección, tanto para que el jugador patee la pelota como para hacer volar al arquero.

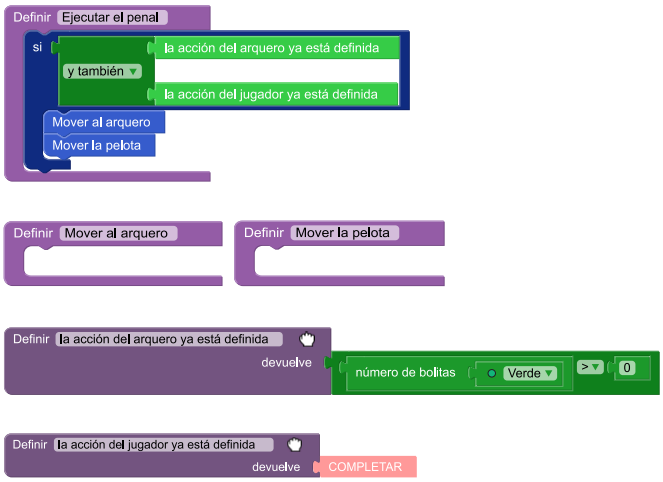
Funciones `izquierda`, `centro` y `derecha`

Para completar `Establecer dirección para atajar [ ]` y `Establecer dirección para patear [ ]`, es conveniente prestar atención al modo en que estos procedimientos se invocan desde el cuerpo principal del programa: al llamarlos, se usa como argumento el resultado de una invocación o bien a `izquierda`, o bien a `centro` o bien a `derecha` que, como se ha mencionado, devuelve la cantidad de bolitas que se usan para codificar cada dirección. De este modo, tanto `Establecer dirección para atajar [ ]` como `Establecer dirección para patear [ ]` tienen un parámetro que, a pesar de llamarse `dirección`, es numérico. Usando este parámetro, los colores `Azul` –que representa las opciones del jugador– y `Verde` –que representa las opciones del arquero– y el procedimiento de la biblioteca `Poner [ ] bolitas de color [ ]` podemos resolver ambos procedimientos.



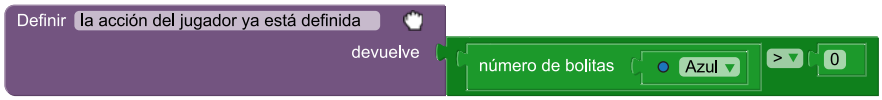
Procedimientos para establecer la dirección para patear y para atajar

El procedimiento `Ejecutar el penal`, en caso de que ambos jugadores hayan seleccionado una opción, tiene que encargarse tanto de hacer volar al arquero como de que el jugador patee la pelota. Para chequear si tanto el jugador como el arquero ya tienen establecidas sus opciones para patear y atajar, se puede usar una alternativa condicional y las funciones `la acción del arquero ya está definida` (que ya está resuelta) y `la acción del jugador ya está definida` (que todavía no está programada). Además, para mover al arquero y la pelota se pueden crear dos procedimientos: `Mover al arquero` y `Mover la pelota`.



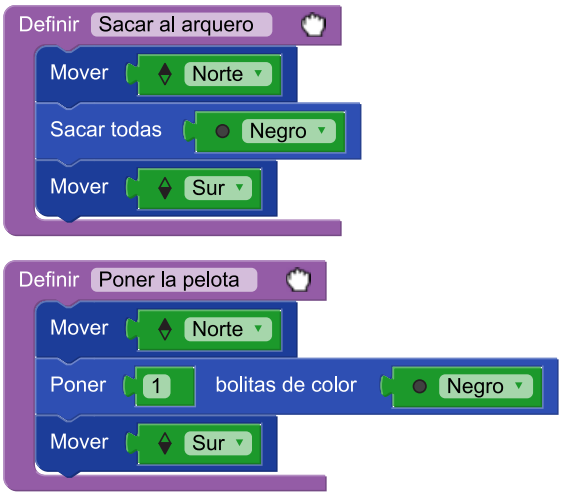
Procedimiento `Ejecutar el penal`

Tomando en cuenta cómo se codifican las opciones para patear y atajar, y observando cómo está programada la función `la acción del arquero ya está definida`, es fácil completar `la acción del jugador ya está definida`: tiene que devolver verdadero si bajo el cabezal hay bolitas azules y falso en caso contrario.



Función `la acción del jugador ya está definida`

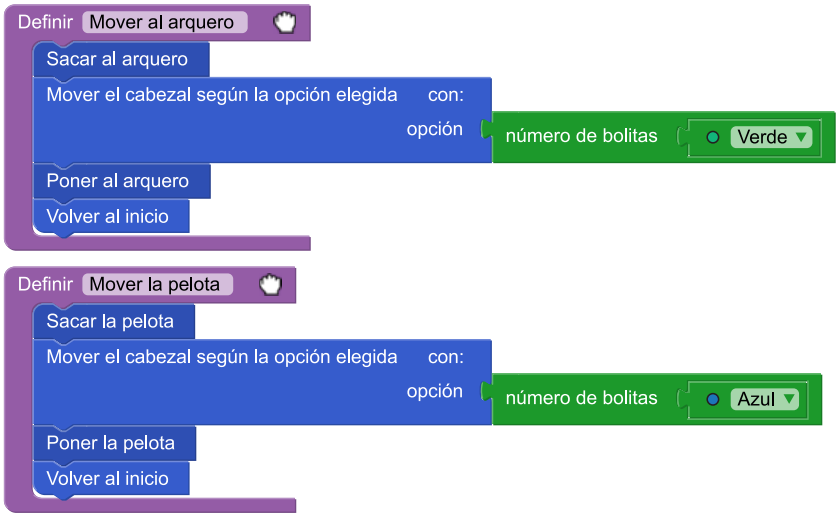
Antes de resolver `Mover al arquero` y `Mover la pelota`, es importante reparar en que los procedimientos que operan sobre el arquero o la pelota (como por ejemplo `Sacar al arquero` y `Poner la pelota`, que vienen resueltos en el proyecto), al finalizar su ejecución, dejan el cabezal posicionado sobre la misma celda en la que se encontraba antes de ser invocados. Esto permite que, cuando se los invoque, se tenga garantía sobre donde quedará el cabezal una vez que su ejecución finaliza. Al completar tanto `Mover al arquero` como `Mover la pelota`, deberá hacerse lo mismo; en estos casos, dejar al cabezal sobre la celda (1,0), la de inicio.



Procedimientos `Sacar al arquero` y `Poner la pelota`

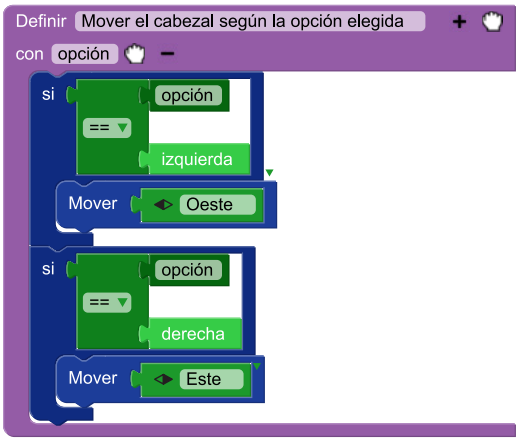
Para resolver `Mover al arquero` hay que: *(i)* sacar al arquero de su lugar; *(ii)* mover al cabezal según la opción que se haya elegido para el arquero; *(iii)* poner allí al arquero; y *(iv)* volver a posicionar

el cabezal en la celda inicial. Para ello, usamos los procedimientos `Sacar al arquero`, `Poner al arquero`, `Volver al inicio` (los tres vienen resueltos en el proyecto) y `Mover el cabezal según la opción elegida [ ]` (que hay que completar). Este último tiene un parámetro numérico para indicar hacia dónde mover el cabezal. En el caso del arquero, la posición está determinada por la cantidad de bolitas verdes que haya en (1,0). De forma muy parecida se resuelve el movimiento de la pelota, en este caso usando `Sacar la pelota` y `Poner la pelota`, y chequeando la cantidad de bolitas azules para definir dónde ubicarla.



Procedimientos `Mover el arquero` y `Mover la pelota`

Finalmente, para completar `Mover el cabezal según la opción elegida [ ]` hay que usar dos alternativas condicionales que, de acuerdo al argumento recibido, desplace el cabezal en la dirección adecuada. Es importante notar que, si `opción` es igual a `centro`, no hay que hacer nada, pues ahí es donde se encuentra el cabezal antes de comenzar la ejecución de este procedimiento.



Procedimiento `Mover el cabezal según la opción elegida [ ]`

CIERRE

Como cierre, les hacemos notar a los estudiantes que acaban de crear un juego utilizando muchas de las herramientas que fueron aprendiendo a lo largo del curso: la interactividad, la subdivisión en tareas, la creación de procedimientos y funciones, la representación de información para codificar los movimientos del arquero y la pelota, la alternativa condicional y los operadores lógicos, entre otras.

NOMBRE Y APELLIDO:

CURSO: FECHA:

DEFINICIÓN POR PENALES

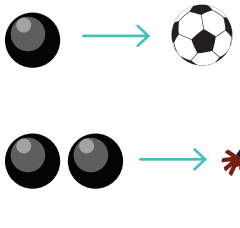
Un penal es, ante todo, una prueba de fortaleza psicológica para quien patea. ¿Hacia dónde se tirará el arquero? ¿Le pego fuerte? ¿A colocar? Estas son algunas preguntas que asaltan a un futbolista cuando se enfrenta a un arquero desde los doce pasos.

1. Abrí el proyecto "Definición por penales" y completá el programa para que dos jugadores se enfrenten en la ejecución de un penal. Uno, que comanda al goleador, elige dónde patear. El otro decide qué debe hacer el arquero. Una vez que ambos hayan definido las acciones de los futbolistas, presionando la barra espaciadora se tiene que ver en la pantalla la ejecución del penal. Es esta tabla te mostramos cómo hacen los jugadores para establecer qué van a hacer tanto el goleador como el arquero:

JUGADOR	AL PRESIONAR LA TECLA	SE DECIDE
	I	Patear a la izquierda
	O	Patear al centro
	P	Patear a la derecha
	Q	Volar a la izquierda
	W	Quedarse parado
	E	Volar a la derecha

VESTIMENTA

La vestimenta representa a la pelota y al arquero con una y dos bolitas negras respectivamente



NOMBRE Y APELLIDO:

CURSO: FECHA:

Cuando se presiona alguna de estas teclas, tenés que registrar la opción elegida. Para ello, vas a colocar bolitas verdes y azules en el tablero, sin mover el cabezal. Después, cuando se ejecute el penal, observando cuántas bolitas de cada color hay, podés saber a dónde mover tanto la pelota como al arquero. Así tenés que codificar cada opción:

PERSONAJE	CANTIDAD Y COLOR DE BOLITAS	ACCIÓN
	🔵	Patear a la izquierda
	🔵🔵	Patear al centro
	🔵🔵🔵	Patear a la derecha
	🟢	Volar a la izquierda
	🟢🟢	Quedarse parado
	🟢🟢🟢	Volar a la derecha

PISTA

Tanto cuando se define que la pelota vaya a la izquierda como cuando se decide que el arquero vuele hacia ese lado, hay que poner una bolita. Si mirás el panel de programa, vas a ver que hay una función `izquierda` que devuelve `1`. ¿Qué debería devolver la función `centro` que hay que completar? Además, ¿no sería conveniente también crear una función `derecha`?
A lo mejor, estas funciones las podés usar como argumento cuando invoques `Establecer dirección para atajar [ ]` y `Establecer dirección para patear [ ]`.

PARA PATEAR OTRA VEZ

Al presionar la tecla *Enter* hay que reiniciar el juego dejando todo como al comienzo. ¡Inspeccioná la biblioteca, para ver si encontrás algo que te ayude!

SUENA EL SILBATO DEL ÁRBITRO

Una vez que ambos jugadores hayan definido las acciones de los futbolistas, presionando la barra espaciadora se tiene que ver en la pantalla la ejecución del penal.

PARA TENER EN CUENTA

- ¡Acordate de dividir el problema en subtareas! Podrías tener procedimientos tales como `Ejecutar el penal`, `Mover al arquero`, `Mover la pelota`, etc.
- Para hacer las cosas más fáciles, al terminar cada procedimiento ubicá el cabezal sobre la celda en la que estaba al comienzo. De este modo, cuando programes procedimientos y funciones, podés tener la seguridad de que el cabezal comienza estando allí.

Actividad 2
Snake

DE A DOS

OBJETIVOS

- Ejercitar la construcción de programas interactivos.
- Introducir el uso de *timeout*.

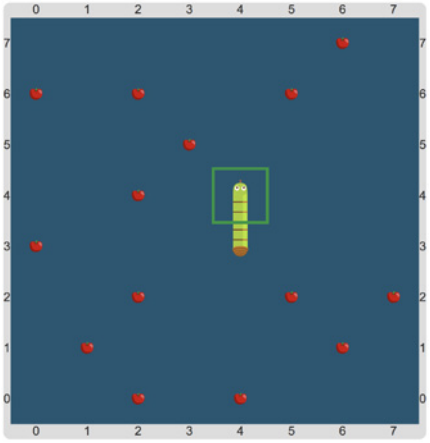
MATERIALES

- Computadoras
- Gobstones
- Ficha para estudiantes

DESARROLLO

En esta actividad, los estudiantes programarán el juego de la viborita, popularmente conocido como *Snake* por su nombre en inglés. Además de integrar y poner en práctica los conocimientos aprendidos, trabajarán con una herramienta frecuentemente usada en los programas interactivos: el *timeout*.

Repartimos la ficha a los estudiantes y les pedimos que abran el proyecto de Gobstones "Snake". Se encontrarán con un tablero de 8 × 8 que muestra una víbora de dos celdas de largo y varias manzanas distribuidas por el tablero. Si presionan en reiteradas oportunidades el botón *Ejecutar*, notarán que hay distintos tableros iniciales. Se trata de un juego para un solo jugador que, usando las 4 flechas del teclado, debe mover a la víbora por el tablero para que coma todas las manzanas sin chocarse con los bordes ni consigo misma. Además, cada vez que coma una manzana, aumentará su longitud en una celda.

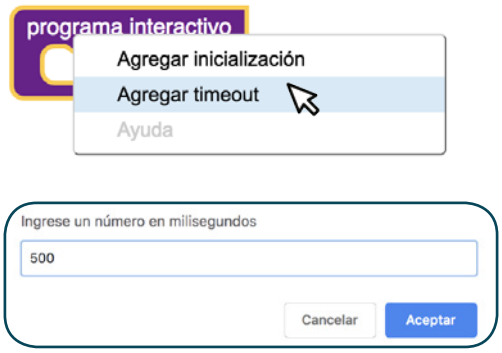


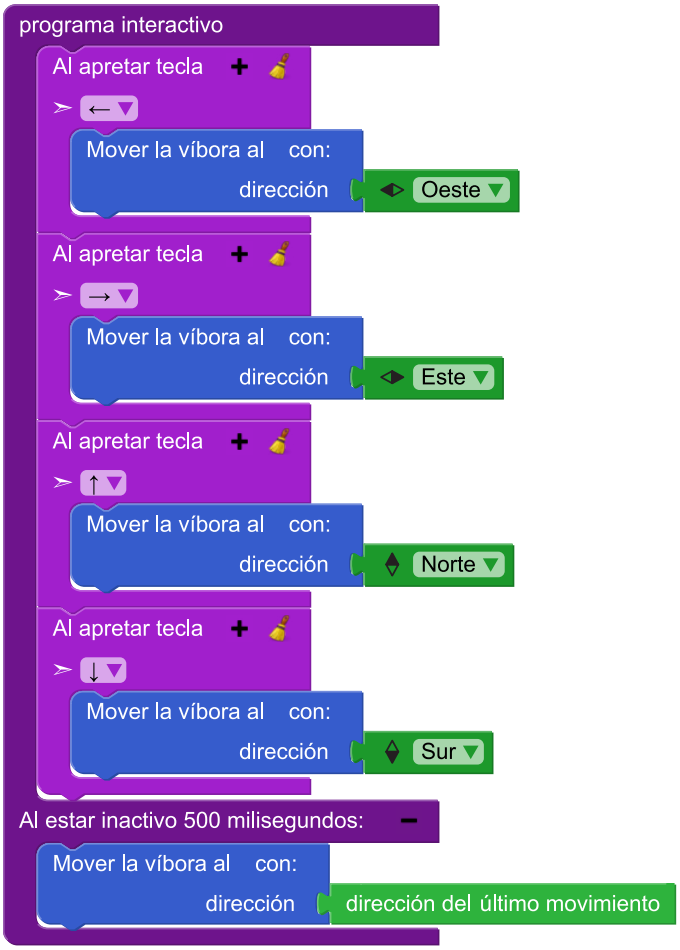
Un tablero inicial de Snake

Para armar el cuerpo principal del programa, en el que hay que establecer las vinculaciones entre las teclas y las acciones que estas desencadenan al ser presionadas, el proyecto incluye un procedimiento –aunque todavía no está programado– que es de suma utilidad: `Mover la víbora al [ ]`. Con él podemos indicar que: *(i)* cuando se presione la flecha arriba ↑ la víbora se moverá al norte; *(ii)* cuando se apriete la flecha derecha →, al este; *(iii)* al presionar la flecha abajo ↓, al sur; y *(iv)* al apretar la flecha izquierda ←, al oeste.

Además de mover a la víbora con las flechas, hay que tener en cuenta que, si el jugador no presiona ninguna tecla, después de un lapso de tiempo el reptil seguirá avanzando en la misma dirección. En los programas interactivos de Gobstones existe un evento especial, llamado *timeout*, que se activa cuando el usuario no toca ninguna tecla después de un cierto tiempo. Para incluirlo, hay que hacer clic con la tecla derecha del ratón sobre el bloque `programa interactivo` y elegir la opción *Agregar timeout*. Al seleccionarla, se desplegará un cuadro de texto en el que hay que ingresar la cantidad de milisegundos (ms) que tienen que pasar sin que se presione una tecla antes de que el programa reaccione del modo que indiquemos. En este caso, una vez que haya pasado medio segundo (o 500 ms),

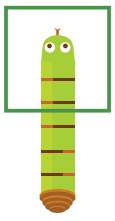
la víbora avanzará en la dirección de su último movimiento. Para esto último hay que usar el procedimiento `Mover la víbora al [ ]` y la función `dirección del último movimiento`, que está disponible en *Biblioteca* > *Funciones*.





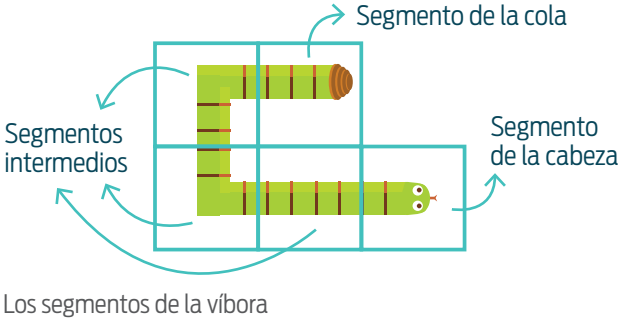
Incorporación de *timeout* en el cuerpo principal del programa

En todos los tableros iniciales, el cabezal aparece posicionado sobre la cabeza de la víbora. Si en cada procedimiento nos ocupamos de que también al finalizar el cabezal quede ubicado en esa parte del reptil, podremos asumir que, cuando una función o un procedimiento comiencen su ejecución, allí se encontrará el cabezal.



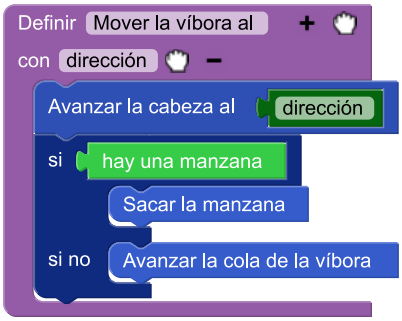
El cabezal sobre la cabeza de la víbora

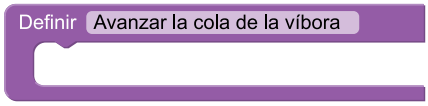
Es importante notar que la víbora está compuesta por una serie de segmentos consecutivos, cada uno ubicado en una celda distinta. Entre ellos, se pueden distinguir: *(i)* la cabeza, que no tiene un segmento anterior; *(ii)* la cola, que no tiene un segmento posterior; y *(iii)* los intermedios, que tienen tanto un segmento anterior como uno posterior.



Los segmentos de la víbora

Para completar el procedimiento `Mover la víbora al [ ]` hay que tener presente que, si al moverse come una manzana, se debe alargar su longitud en un segmento. Esto puede conseguirse desplazando la cabeza una celda en la dirección que corresponde y dejando el resto de la víbora intacta. En *Biblioteca > Procedimientos*, se encuentra disponible `Avanzar la cabeza al [ ]`, que extiende la longitud de la víbora un segmento haciendo avanzar la cabeza una celda en una dirección determinada. Pueden presentarse dos situaciones: *(i)* que al adelantarse haya una manzana, en cuyo caso habrá que retirarla del tablero; y *(ii)* que allí no haya una fruta, con lo cual la víbora tendrá que seguir midiendo la misma cantidad de segmentos, por lo que también habrá que hacer avanzar la cola una celda. Usando una alternativa condicional, la función `hay una manzana`, el procedimiento `Sacar la manzana` –ambos por completar– y creando un nuevo procedimiento `Avanzar la cola de la víbora`, podemos completar `Mover la víbora al [ ]`.

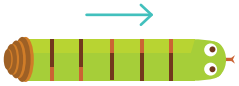
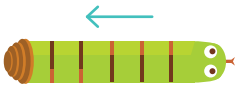




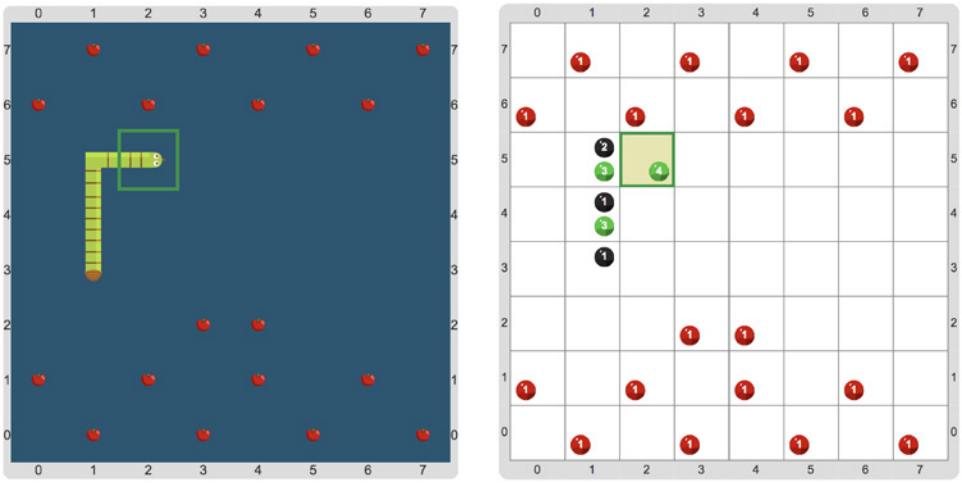
Procedimiento `Mover la víbora al [ ]`

Para completar los procedimientos y las funciones que faltan, es necesario tener en cuenta el modo en que la vestimenta representa los distintos elementos que forman parte del juego. Cada segmento de la víbora tiene codificado hacia dónde apuntan el segmento anterior (es decir, el que va hacia la cabeza) y el segmento posterior (el que va hacia la cola). El segmento anterior se codifica con bolitas negras y el posterior, con verdes. La dirección en la que se orientan los segmentos aledaños se codifica utilizando distintas cantidades de bolitas: 1 bolita para indicar dirección norte, 2 para indicar el este, 3 para el sur y 4 para el oeste.

Codificación de los segmentos anterior y posterior

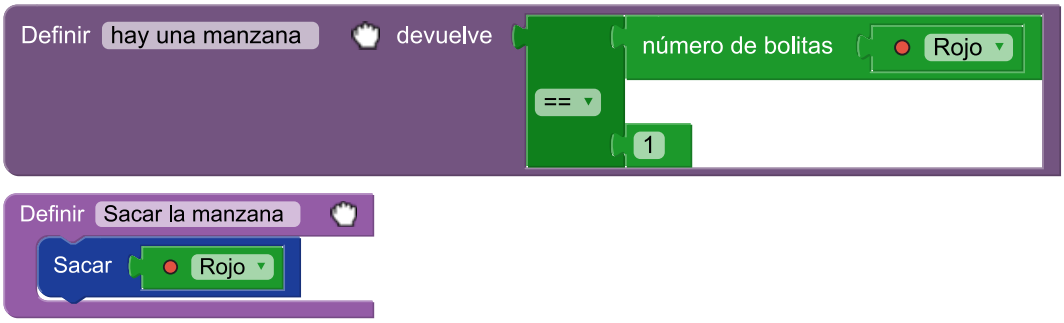
Segmento de la víbora	Cantidad y color de bolitas	Dirección hacia la que se orienta
 Segmento anterior	●	Hacia el norte
	● ●	Hacia el este
	● ● ●	Hacia el sur
	● ● ● ●	Hacia el oeste
 Segmento posterior	●	Hacia el norte
	● ●	Hacia el este
	● ● ●	Hacia el sur
	● ● ● ●	Hacia el oeste

Se debe notar que, como la cabeza no tiene un segmento anterior, en la celda en la que se encuentre habrá solo bolitas verdes. Del mismo modo, como la cola no tiene un segmento posterior, en dicha celda habrá solo bolitas negras. Además, cada manzana se codifica con una bolita roja.



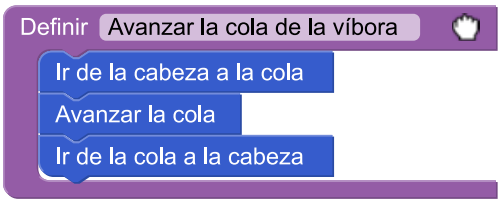
Tablero de la actividad con y sin vestimenta

Para completar la función `hay una manzana` se debe chequear si bajo el cabezal hay o no una bolita roja. En caso afirmativo, la función devuelve verdadero; en caso contrario, falso. Por su parte, el procedimiento `Sacar la manzana` consiste únicamente en retirar una bolita roja.



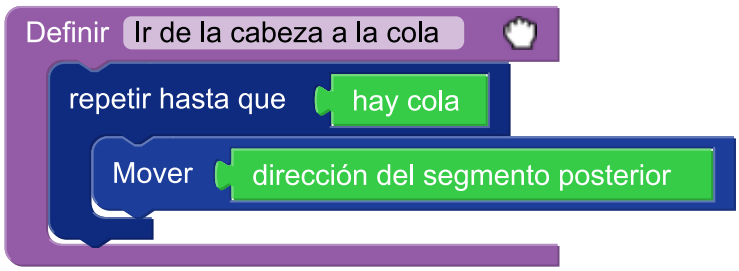
Función `hay una manzana` y procedimiento `Sacar la manzana`

Para resolver `Avanzar la cola de la víbora` hay que ocuparse de: *(i)* mover el cabezal desde la cabeza hasta la cola, *(ii)* hacer avanzar la cola y *(iii)* volver a posicionar el cabezal sobre la cabeza del reptil. Es conveniente, entonces, dividir este problema en tres partes y crear un procedimiento para cada una: `Ir de la cabeza a la cola`, `Avanzar la cola` e `Ir de la cola a la cabeza`.



Procedimiento `Avanzar la cola de la víbora`

El procedimiento `Ir de la cabeza a la cola` tiene que ocuparse de mover el cabezal desde la cabeza, que es donde está posicionado, hasta la cola, es decir, la celda en la que está el último segmento del reptil. Para llegar, hay que seguir las direcciones que señalan dónde está el segmento posterior hasta que finalmente se alcance la cola (sin indicación en tal sentido pues no tiene un segmento posterior). Usando una repetición condicional, el comando básico `Mover [ ]` y las funciones `hay cola` (que ya está en el proyecto, pero hay que completar) y `dirección del segmento posterior` (que ya está en el proyecto y también hay que terminar) se puede completar el procedimiento.

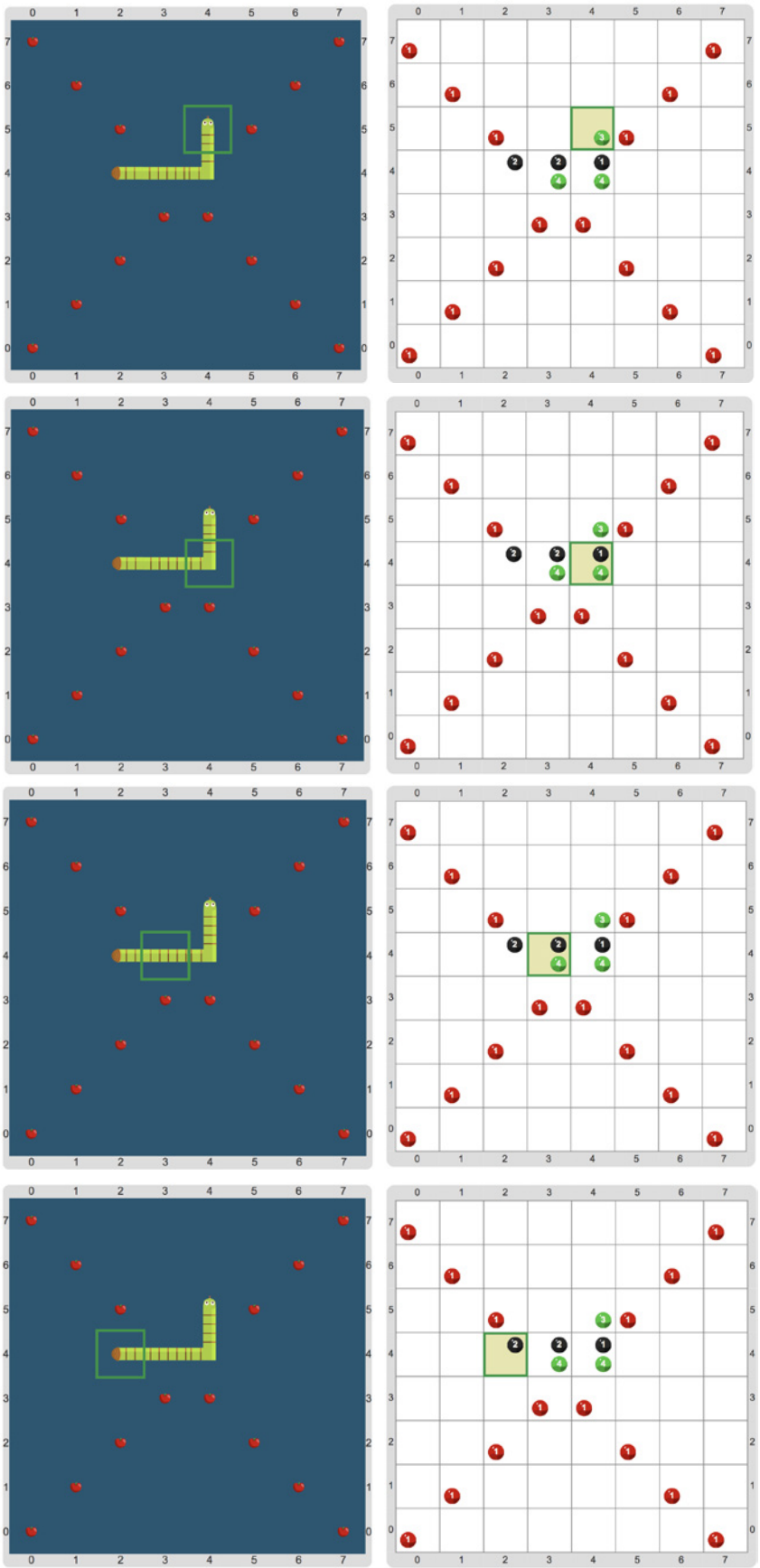


Procedimiento `Ir de la cabeza a la cola`

A continuación puede observarse una ejecución paso a paso de `Ir de la cabeza a la cola`, con y sin vestimenta.

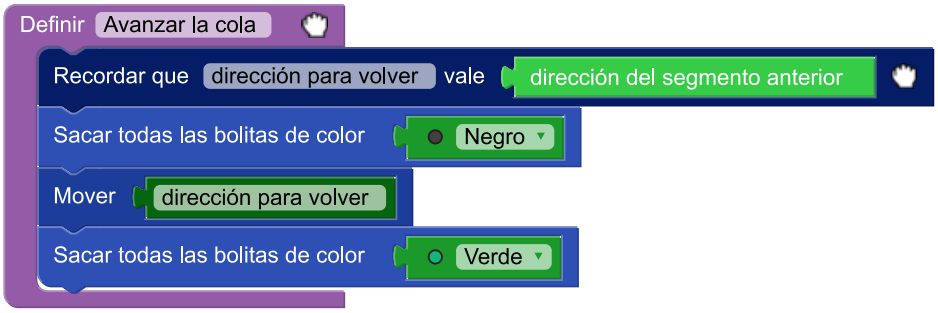
Con vestimenta

Sin vestimenta



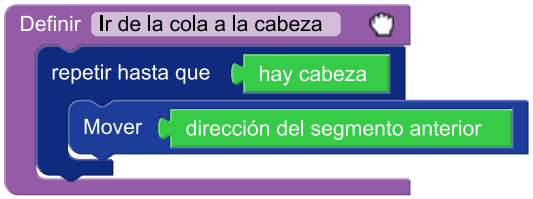
Ejecución paso por paso de `Ir de la cabeza a la cola`

El procedimiento `Avanzar la cola` tiene que sacar el último segmento de la víbora. Además, el que hasta ese momento era el penúltimo pasará a ser el último. Por lo tanto, hay que retirar de esa celda todas las bolitas verdes que codifican la dirección hacia donde se orienta el segmento posterior. Es importante tener en cuenta que, si directamente se sacan todas las bolitas negras dispuestas en la cola –el último segmento de la víbora–, se perderá la información que indica hacia dónde apunta el segmento anterior –penúltimo segmento hasta ese momento–. ¿Cómo sabremos, entonces, hacia dónde mover el cabezal para transformar el penúltimo segmento en el último? En general, para conservar información que se necesita más adelante hay que usar una variable. En este caso, podría llamarse `dirección para volver`. Una vez guardado ese valor, se puede *(i)* quitar todas las bolitas negras de la última celda usando el procedimiento de la biblioteca `sacar todas las bolitas de color [ ]`; *(ii)* mover el cabezal en la dirección que indique `dirección para volver`; y *(iii)* sacar de allí todas las bolitas verdes, para codificar que ya no hay un segmento posterior.



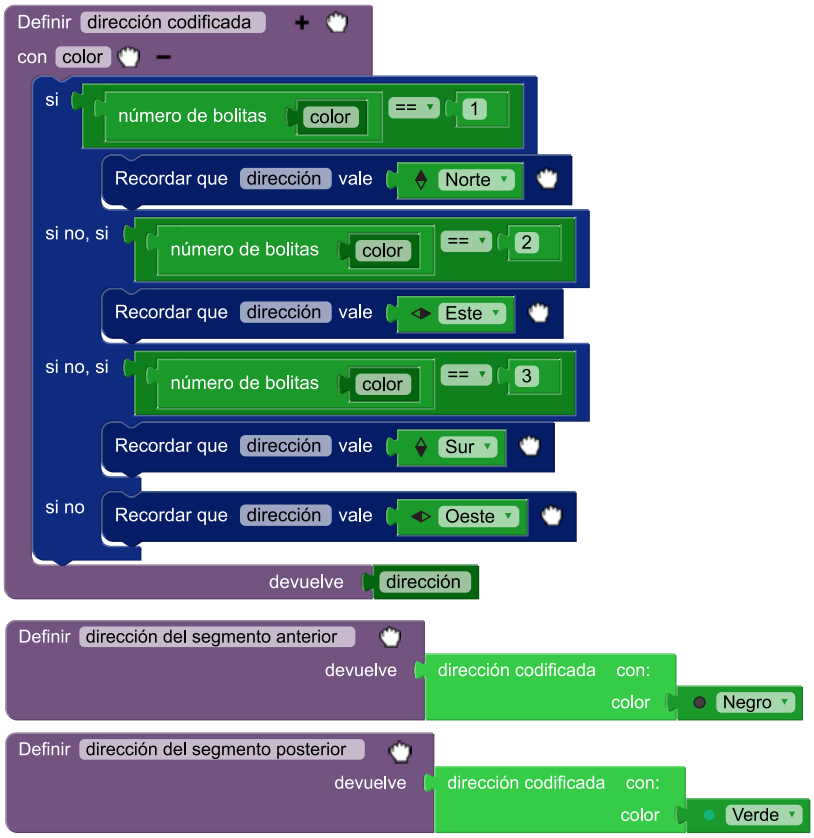
Procedimiento `Avanzar la cola`

Para volver a ubicar al cabezal sobre la cabeza de la víbora se puede usar la misma estrategia que en `Ir de la cola a la cabeza`: combinar una repetición condicional con el comando básico `Mover [ ]` y, en este caso, las funciones `hay cabeza` y `dirección del segmento anterior` (que están en el proyecto, pero hay que completar).



Procedimiento `Ir de la cola a la cabeza`

Teniendo en cuenta el modo en que la vestimenta representa los segmentos de la víbora, para resolver la función `dirección del segmento posterior` hay que ver cuántas bolitas verdes hay en la celda bajo el cabezal. Si hay una, el segmento posterior está orientado hacia al norte; si hay dos, hacia el este; si hay tres, hacia el sur; y si hay cuatro, hacia el oeste. Algo parecido hay que hacer para completar `dirección del segmento anterior`, salvo que en este hay que inspeccionar la cantidad de bolitas negras. Para no hacer dos funciones prácticamente iguales, una posibilidad es crear una función con procesamiento `dirección codificada` que tenga un parámetro que represente un color. Si al invocarla se usa `Verde` como argumento, el resultado será la dirección del segmento posterior; si se usa `Negro`, la dirección del segmento anterior.



Funciones `dirección codificada`, `dirección del segmento posterior` y `dirección del segmento anterior`

Para terminar el juego, falta completar las funciones `hay cola` y `hay cabeza`. La primera devolverá verdadero cuando bajo el cabezal haya bolitas negras pero no verdes; y la segunda, cuando haya verdes pero no negras.

Funciones `hay cola` y `hay cabeza`

CIERRE

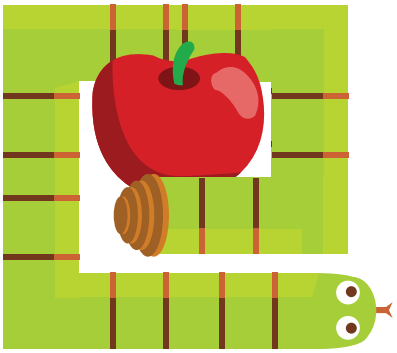
Reflexionamos con los estudiante acerca de que en este proyecto se incorporó el uso de *timeout* para indicar qué tiene que hacer el programa si durante un cierto lapso de tiempo el usuario no realiza ninguna acción. Los invitamos a que identifiquen otros programas que usen *timeout*, como por ejemplo los procesadores de texto que hacen titilar el cursor en la pantalla cuando el usuario no está escribiendo.

NOMBRE Y APELLIDO:

CURSO: FECHA:

SNAKE

¿Sabías que *Snake* es un juego de la década de 1970? En esta actividad vas a programar este juego en Gobstones.

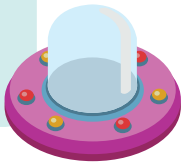


1. Abrí el proyecto "Snake" y completá el programa para poder jugar. Es un juego para un participante que, usando las 4 flechas del teclado, mueve por un tablero a una víbora que debe comer todas las manzanas sin chocarse con los bordes ni consigo misma. Cada vez que come una manzana, la víbora aumenta su longitud en un segmento. Entonces:

- Si al moverse come una manzana, alcanza con adelantar la cabeza.
- Si no come una manzana, tenés que ocuparte de que toda la víbora se desplace una posición.

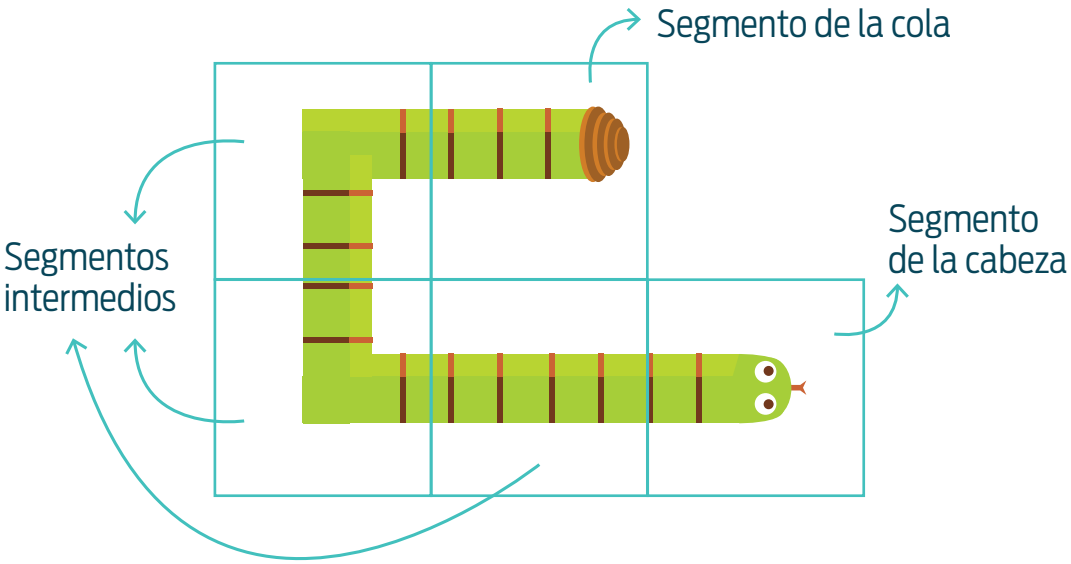
BIBLIOTECA

¡Fijate qué funciones y procedimientos están disponibles en la biblioteca!



LOS SEGMENTOS DEL REPTIL

La víbora está compuesta por una serie de segmentos consecutivos, cada uno ubicado en una celda distinta. Entre ellos, se pueden distinguir: *(i)* la cabeza, que no tiene un segmento anterior; *(ii)* la cola, que no tiene un segmento posterior; y *(iii)* los intermedios, que tienen tanto un segmento anterior como uno posterior.



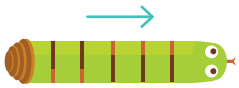
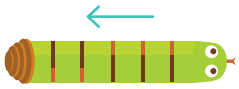
NOMBRE Y APELLIDO:

CURSO: FECHA:

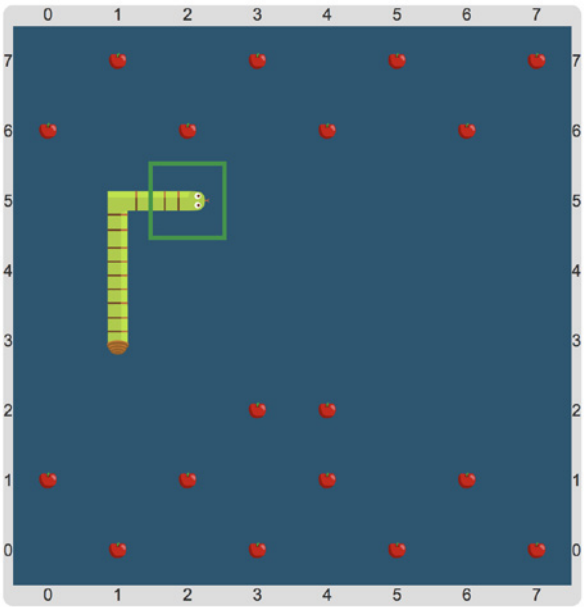
LA CODIFICACIÓN DE LOS SEGMENTOS DE LA VÍBORA

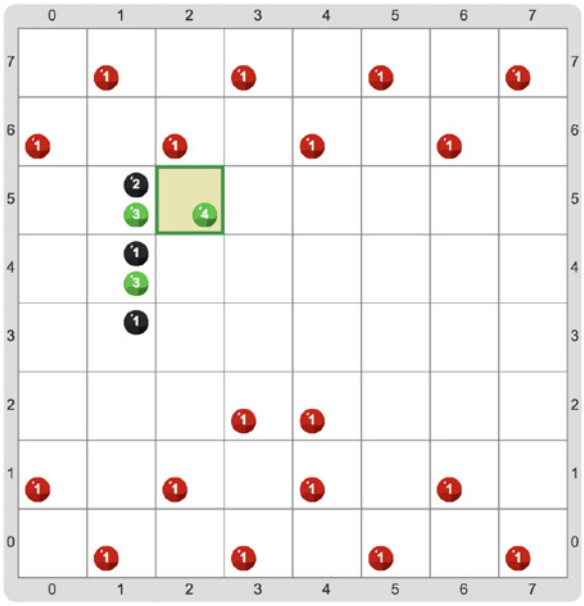
Cada segmento de la víbora tiene codificado dónde está el segmento anterior, es decir, el que va en dirección a la cabeza, y dónde está el segmento posterior, o sea, el que va en dirección a la cola.

- El segmento anterior se codifica con bolitas negras y el posterior, con verdes.
- La dirección de los segmentos aledaños se codifica usando distintas cantidades de bolitas: 1 bolita para indicar dirección norte, 2 para indicar el este, 3 para el sur y 4 para el oeste.

Segmento de la víbora	Cantidad y color de bolitas	Dirección hacia la que se orienta
 Segmento anterior	●	Hacia el norte
	● ●	Hacia el este
	● ● ●	Hacia el sur
	● ● ● ●	Hacia el oeste
 Segmento posterior	●	Hacia el norte
	● ●	Hacia el este
	● ● ●	Hacia el sur
	● ● ● ●	Hacia el oeste

- Como la cabeza no tiene un segmento anterior, en la celda en la que se encuentre habrá solo bolitas verdes. Del mismo modo, como la cola no tiene un segmento posterior, en dicha celda habrá solo bolitas negras.
- Cada manzana se representa con una bolita roja.
- Acá te mostramos un tablero, con y sin vestimenta:



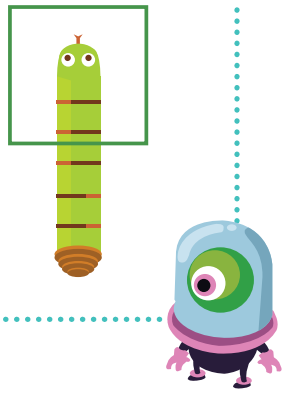


NOMBRE Y APELLIDO:

CURSO: FECHA:

UN CONSEJO

En todos los tableros iniciales, el cabezal comienza posicionado sobre la cabeza de la víbora. Conviene que en cada procedimiento nos ocupemos de que, también al finalizar, el cabezal quede ubicado sobre la cabeza del reptil. De esta forma, cuando una función o un procedimiento comiencen su ejecución, vas a poder asumir que el cabezal está ahí.



2. Extendé el juego para que, si pasa un determinado tiempo sin que el usuario presione una tecla, la víbora continúe avanzando en la misma dirección. ¡Explorá el entorno para ver cómo incorporar ***timeout***! Probá usar diferentes tiempos de espera y fijate cómo cambia la velocidad a la que se mueve la serpiente.

EPÍLOGO

¿CÓMO SEGUIMOS?

A lo largo del manual repasamos distintos aspectos de la computación. Marcamos la diferencia entre *hardware* y *software*, llenamos la caja de herramientas de programación, vimos cómo representar información y resolvimos varios problemas a través de la construcción de programas. Por supuesto, solo se trata de una introducción al mundo de la computación y hay mucho más para estudiar sobre estos y otros aspectos.

Una diferencia importante entre este manual y otras introducciones a la computación es que aquí se utiliza un lenguaje de programación basado en bloques. Si a los estudiantes les interesa seguir aprendiendo, podemos recomendarles que estudien otros lenguajes de programación. La mayoría utiliza una sintaxis textual en lugar de bloques; esto brinda una mayor flexibilidad, pero también hace que sea más fácil cometer errores al programar.

Hay que destacar que las herramientas conceptuales aprendidas en este manual pueden aplicarse directamente en un lenguaje de programación basado en texto. De hecho, lo estuvimos haciendo sin decirlo, pues Gobstones codifica internamente los bloques mediante texto. El cuadro al final de este epílogo muestra cómo traducir cada uno de los bloques a texto para que Gobstones lo pueda ejecutar. Intencionalmente ponemos los nombres en inglés porque así se los llama en el resto de los lenguajes de programación.

Cabe aclarar que en Gobstones los corchetes se usan para describir listas. Sin embargo, en este manual se utilizan para denotar la presencia de parámetros, tanto en procedimientos como en funciones. Esta decisión se tomó para tener una notación uniforme a lo largo de los cuatro volúmenes de la colección "Ciencias de la computación para el aula". En Gobstones la denotación de parámetros se indica entre paréntesis.

Menú para ver el programa textual

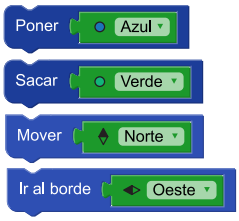
Siguiendo estas guías y a modo de cierre, podemos proponerles a los estudiantes que vuelvan a hacer algunas de las actividades de este manual, pero esta vez usando texto en lugar de bloques. Además, en el entorno hay una opción para que, dado un programa de bloques, se pueda obtener la versión de texto correspondiente. Para ello hay que hacer clic en las tres rayitas horizontales de la esquina superior izquierda y elegir la opción *Código generado*. Esta transición a la programación con texto permitirá aplicar los conocimientos adquiridos en lenguajes populares como Python o Java y continuar con una formación más especializada.

Por último, los contenidos que se desarrollan en este manual brindan una base para entender cómo funcionan los programas y, por lo tanto, permiten comprender mejor el mundo que nos rodea, donde la tecnología ha pasado a formar una parte fundamental de nuestra vida diaria.

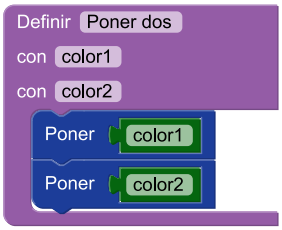
PARA PASAR DE BLOQUES A TEXTO

BLOQUES	TEXTO	CÓMO ESCRIBIRLOS
	`program { ... }`	Los comandos en texto se listan entre llaves.

	`Negro` `Norte`
	`Rojo` `Este`
	`Azul` `Sur`
	`Verde` `Oeste`
		Los colores y las direcciones se escriben con inicial mayúscula.

	`Poner(Azul)`
	`Sacar(Verde)`
	`Mover(Norte)`
	`IrAlBorde(Oeste)`
		Los comandos básicos se escriben sin espacios, empezando cada palabra con mayúscula y, a continuación, el argumento entre paréntesis.

	`hayBolitas(Negro)`
	`nroBolitas(Rojo)`
	`puedeMover(Este)`
		Los sensores se escriben sin espacios, empezando cada palabra con mayúscula salvo la primera y, a continuación, el argumento entre paréntesis.

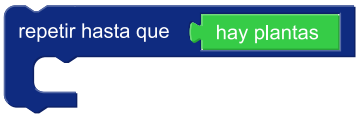
	`procedure PonerDos(color1, color2) {`
	`  Poner(color1)`
	`  Poner(color2)`
	`}`
		La definición de procedimientos se hace escribiendo la palabra ***procedure*** ('procedimiento' en español) seguida del nombre del procedimiento, que debe comenzar con mayúscula. A continuación, es necesaria una lista de parámetros entre paréntesis y, por último, el cuerpo del procedimiento entre llaves.

BLOQUES	TEXTO	CÓMO ESCRIBIRLOS

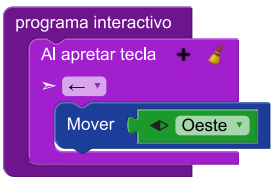
	`function hayPlantas() {`
	`  return (hayBolitas(Verde))`
	`}`
		La definición de funciones se hace con la palabra ***function*** ('función' en español) seguida del nombre de la función, que debe comenzar con minúscula. Luego, una lista de parámetros entre paréntesis y, finalmente, el cuerpo de la función que debe usar la palabra clave ***return*** ('devolver' en español) para indicar el valor que devuelve.

	`if (hayPlantas()) {`
	`  ...`
	`} else {`
	`  ...`
	`}`
		La alternativa condicional se hace con la palabra ***if*** ('si' en español) seguida de la condición a evaluar y los comandos a ejecutar si es verdadera. A continuación, se puede usar la palabra ***else*** (en español 'otro') y comandos a ejecutar si la condición es falsa.

	`repeat (10) {`
	`  ...`
	`}`
		La repetición simple se hace con la palabra ***repeat*** ('repetir' en español) y el número de veces a repetir entre paréntesis, seguido de los comandos a repetir.

	`while (not (hayPlantas())) {`
	`  ...`
	`}`
		La repetición condicional se hace usando la palabra clave ***while*** ('mientras' en español), seguida de una condición y los comandos a repetir. En lugar de repetir hasta que se dé una condición, repite mientras no se cumpla y por eso la condición del "repetir hasta" debe aparecer negada en el ***while***.

	`cantidad := 0`	Para la asignación de variables, se escribe el nombre de la variable a asignar seguida del operador de asignación (**:=**) y luego la expresión que denota el valor a guardar en la variable.

	`interactive program {`
	`  K_LEFT -> {`
	`  Mover(Oeste)`
	`  }`
	`}`
		Los programas interactivos se hacen con la palabra ***interactive*** ('interactivo' en español) seguida del programa. Las teclas son constantes que empiezan con K, van seguidas de una flecha y luego el bloque asociado a ellas.

GLOSARIO

activar una vestimenta. Indicación al entorno de Gobstones para que muestre el tablero con las imágenes provistas por una vestimenta en lugar de las bolitas.

acumulador. Variable que se va incrementado durante la ejecución de un programa, procedimiento o función y acumula una cantidad que se precisa contar. Sinónimo de **contador**.

alternativa condicional. Herramienta de un lenguaje de programación que permite elegir entre dos posibles bloques de comandos sobre la base de una condición (dada por una expresión de tipo booleano).

aplicación. Programa ejecutable por una computadora con un propósito específico, como navegar por Internet, escribir textos, organizar un álbum de fotos, etc.

argumento. Dato o valor con el que se completa un procedimiento o una función con uno o más parámetros al ser invocado.

arquitectura de von Neumman. Diseño conceptual y estructura operacional fundamental para computadoras digitales electrónicas, creados en 1945 por el matemático y físico John von Neumann. Incluye una unidad de procesamiento, una memoria principal y mecanismos de entrada y salida de información. En este diseño se basa la mayoría de las computadoras modernas.

asignación. Comando que permite vincular una variable con un valor.

autómata. Máquina que sigue al pie de la letra instrucciones.

Azul (expresión). Expresión literal de Gobstones que describe uno de los cuatro colores posibles de las bolitas.

bit. Unidad básica de información, que puede tomar solo dos valores, usualmente representados como 0 y 1.

booleano (tipo de datos). Tipo de datos que permite representar valores de verdad. Los valores de este tipo son dos: *verdadero* (que corresponde a responder "sí" a una pregunta) y *falso* (que corresponde responder "no" a una pregunta).

BOOM. Forma coloquial de nombrar una falla que se produce durante de la ejecución de un programa en Gobstones.

borde (del tablero). Cada una de las filas y columnas que delimitan el tablero de Gobstones.

cabezal. Componente activo de la máquina de Gobstones, que puede poner y sacar bolitas de una celda del tablero y desplazarse de una celda a otra celda vecina.

caso de borde. Conjunto de acciones que deben realizarse inmediatamente antes o después de una repetición, debido a alguna diferencia con los comandos que se repiten.

celda. Cada una de las regiones cuadradas en las que está dividido el tablero de Gobstones.

codificación binaria. Representación de información con secuencias de bits.

codificación RGB. Codificación de colores a partir de la combinación en distintas proporciones de rojo, verde y azul.

código ASCII. Código para representar caracteres en una computadora. Codifica 256 caracteres diferentes. Recibe su nombre de las siglas del nombre en inglés *American Standard Code for Information Interchange.*

código UNICODE. Código para representar caracteres en una computadora. Fue diseñado para facilitar el tratamiento informático, la transmisión y la visualización de textos de múltiples lenguajes y disciplinas técnicas, además de textos clásicos de lenguas muertas. Incluye caracteres de muchísimos idiomas, incluidos caracteres chinos, rusos, griegos, hebreos, etc.

color (tipo de datos). Tipo de datos de Gobstones que permiten representar diferentes colores. Los valores de este tipo son: Azul, Negro, Rojo y Verde.

columna. Serie de celdas que aparecen una a continuación de otra en forma vertical.

comando básico. Descripción de una acción básica que puede ejecutar una máquina. En Gobstones hay cuatro: Poner, Sacar, Mover, Ir al borde y Vaciar tablero.

compresión. Proceso por el cual se obtiene una representación alternativa para un dato (por ejemplo, una imagen), con el objetivo de utilizar una menor cantidad de bits.

computadora. Máquina con capacidad de almacenar y manipular distinto tipo de información utilizando programas.

conjunción lógica (operador de). Operador que permite combinar dos expresiones de tipo booleano para obtener una nueva expresión de tipo booleano que es verdadera cuando ambas condiciones son verdaderas.

contador. Ver **acumulador**.

CPU. Sigla que se refiere a la unidad central de procesamiento (del inglés, *central processing unit*).

cuerpo de una repetición. Instrucciones que se van a repetir. Ver **repetir (comando)**.

desactivar una vestimenta. Indicación al entorno de Gobstones para que muestre el tablero con bolitas y oculte las imágenes provistas por una vestimenta.

descompresión. Proceso inverso a la compresión, que permite recuperar un dato en su representación original.

desinstalar (un programa o aplicación). Eliminar una aplicación de una computadora.

dirección (tipo de datos). Tipo de datos de Gobstones que permite representar las direcciones de movimiento del cabezal. Los valores de este tipo son: Norte, Este, Sur y Oeste.

disco rígido. Ver **unidad de disco rígido**.

dispositivos de entrada. Dispositivos utilizados para ingresar información en una computadora, por ejemplo: teclado, ratón, micrófono, escáner.

dispositivos de salida. Dispositivos usados por la computadora para comunicar información al exterior, por ejemplo: monitor, parlante, impresora.

disyunción lógica (operador de). Operador que permite combinar dos expresiones de tipo booleano para obtener una nueva expresión de tipo booleano que es verdadera cuando al menos una de las condiciones es verdadera.

división en subtareas. Descomposición de una tarea en tareas más pequeñas que resuelven problemas más simples de modo que, al combinarlas, consiguen solucionar el problema original.

dominio (de un problema). Conjunto de elementos y operaciones específicos de un problema.

ejecutar un programa. Indicarle a la computadora que lleve a cabo las acciones que describe un programa.

ejecución. Proceso que sucede cuando se ejecuta un programa. En Gobstones, el resultado de la ejecución es un tablero final o un BOOM en caso de error. Ver **ejecutar un programa**.

elemento. Objeto o entidad que es parte del dominio de un problema.

entorno de programación. Programa de computadora diseñado para escribir programas y ejecutarlos.

error de ejecución. Situación anómala durante la ejecución de un programa, que le impide continuar. Por ejemplo, en Gobstones, mover el cabezal fuera del tablero o sacar una bolita que no existe.

esquina (del tablero). Unión de dos bordes del tablero.

Este (expresión). Expresión literal de Gobstones que designa una de las cuatro direcciones de movimiento del cabezal.

estrategia de solución. Forma específica de concebir la resolución de un problema. En general, existen varias estrategias diferentes para solucionar un problema.

evento. Suceso detectado por un programa interactivo que, en general, es causado por la interacción de un usuario del programa.

expresión (de un lenguaje de programación). Herramienta que describe información. Como regla general, los nombres de las expresiones son sustantivos o preguntas que se responden por sí o no.

expresión literal (de un lenguaje de programación). Expresión que designa un dato determinado de manera única y explícita (por ejemplo, un número fijo o un color conocido), literalmente.

fallar (un programa). Producirse un error durante la ejecución de un programa. En Gobstones, una falla produce un BOOM.

fila. Serie de celdas que aparecen una a continuación de otra en forma horizontal.

formato (de archivo). Codificación estandarizada para la información de un tipo de datos (imágenes, sonidos, videos, textos, etc.).

función. Herramienta de un lenguaje de programación que provee una forma mediante la cual se puede definir una expresión nueva y establecer cuál será su resultado.

función con procesamiento. Función que antes de devolver el valor que nombra puede realizar acciones para poder calcular ese valor. En Gobstones, las funciones con procesamiento no afectan el estado del tablero.

Gobstones. Lenguaje de programación que trabaja con un tablero, un cabezal y bolitas, diseñado para aprender a programar.

hardware. Componentes físicos de una computadora.

hay bolitas (sensor). Expresión de Gobstones referida a la presencia (o ausencia) en la celda bajo el cabezal de bolitas de un color indicado con un argumento.

igualdad (operador de). Operador que permite combinar dos expresiones del mismo tipo para obtener una nueva expresión de tipo booleano que es verdadera cuando ambos argumentos tienen un mismo valor.

información. Conjunto de datos que constituye un mensaje con la capacidad de alterar el estado del conocimiento.

instalar (un programa o aplicación). Incorporar una aplicación a una computadora para que pueda ejecutarse.

Ir al borde (comando). Comando básico de Gobstones que indica la acción del cabezal que consiste en moverse todo lo posible en una dirección señalada por un argumento. Esta acción hace que el cabezal se mueva hasta uno de los bordes del tablero.

legibilidad (de un programa). Capacidad de un programa de ser leído con claridad.

lenguaje de programación. Conjunto de reglas sintácticas (con una semántica inequívoca) para construir programas.

máquina. Dispositivo tecnológico que funciona de manera predefinida, ya sea mecánica o electrónicamente.

memoria principal. Ver **memoria RAM**.

memoria RAM. Memoria en la que una computadora almacena los programas y datos que usa en un determinado momento. Su nombre es sigla del nombre en inglés *Random Access Memory* (memoria de acceso aleatorio) debido a que, si bien la memoria está organizada como una tira de celdas una a continuación de la otra, puede accederse a todas las posiciones directamente y en un tiempo constante.

memoria secundaria. Ver **unidad de almacenamiento**.

Mover (comando). Comando básico de Gobstones que indica la acción del cabezal que consiste en moverse a la celda contigua en la dirección señalada por un argumento.

multiplicación (operador de). Operador que permite combinar dos expresiones de tipo número para obtener una nueva expresión de tipo número que describe la multiplicación de las cantidades involucradas.

negación (operador de). Operador que permite combinar una expresión de tipo booleano para obtener una nueva expresión de tipo booleano que es verdadera cuando la original es falsa, y viceversa.

Negro (expresión). Expresión literal de Gobstones que designa uno de los cuatro colores de las bolitas.

no (operador). Bloque que expresa el operador de negación en Gobstones.

nombrar (un procedimiento o una función). Elegir el nombre de un procedimiento o una función para poder reconocerlo al utilizarlo.

nombre adecuado. Nombre de un procedimiento o función que describe con claridad para qué sirve ese procedimiento o función, sin tener que leer la definición.

Norte (expresión). Expresión literal de Gobstones que se refiere a una de las cuatro direcciones de movimiento del cabezal.

número (tipos de datos). Tipo de datos de Gobstones que permite representar diferentes cantidades. Los valores de este tipo son los números naturales: 0, 1, 2, 3, etc.

número de bolitas (sensor). Sensor de Gobstones que permite obtener la cantidad de bolitas del color indicado por un argumento en la celda bajo el cabezal.

o bien (operador). Bloque que expresa el operador de disyunción lógica en Gobstones.

Oeste (expresión). Expresión literal de Gobstones que designa una de las cuatro direcciones de movimiento del cabezal.

operación (del dominio). Acción que se realiza sobre un elemento o el entorno y modifica algo de ellos.

operador. Herramienta de un lenguaje de programación que permite combinar expresiones cuyo valor resultante es también una expresión. Por ejemplo, el operador de suma (que describe la suma de dos números, como 4+3).

operador booleano. Operador cuyos argumentos y resultado son de tipo booleano. También llamado **operador lógico**.

operador de comparación. Operador cuyos argumentos son de cualquier tipo y el resultado es de tipo booleano. Los operadores de comparación son cuatro: la comparación por mayor, la comparación por menor, la comparación por mayor o igual y la comparación por menor o igual.

operador lógico. Ver **operador booleano**.

opuesto (operador). Operador que, dada una expresión de tipo dirección, da por resultado una nueva expresión de tipo dirección que designa la dirección opuesta. El opuesto de Norte es Sur, el de Este es Oeste, etc.

parámetro. "Agujero" en un procedimiento que, al ser invocado, debe completarse con un argumento. Al agujero se le pone un nombre para poder utilizarlo en el programa.

píxel. Unidad básica de una imagen digitalizada en pantalla. Una imagen se representa con una grilla de píxeles.

placa madre. Placa principal a la que se conectan los componentes principales de una computadora, como la unidad central de procesamiento y la memoria. Posee circuitos impresos que permiten la comunicación entre los componentes de *hardware* de una computadora.

Poner (comando). Comando básico de Gobstones que describe la acción del cabezal de poner una bolita del color de un argumento en la celda sobre la que se encuentra.

previo (operador). Operador que, dada una expresión de tipo dirección, da como resultado en una nueva expresión de tipo dirección que describe la dirección previa en el sentido de las agujas del reloj. El previo de Norte es Oeste, el de Este es Norte, etc.

procedimiento. Herramienta de un lenguaje de programación que permite definir un comando nuevo y establecer su funcionamiento.

procedimiento parametrizado. Procedimiento que tiene uno o más parámetros.

programa. Descripción de la solución de un problema computacional que puede ser ejecutado por una máquina.

programa interactivo. Programa que realiza distintos procesamientos según las acciones del usuario, como tocar teclas, mover el ratón, etc.

proyecto. Conjunto de componentes de Gobstones que permiten realizar una actividad. Un proyecto incluye un programa, uno o más tableros iniciales y cero, una o más vestimentas.

puede mover (sensor). Sensor de Gobstones que se usa para evaluar la posibilidad de que el cabezal se desplace en la dirección indicada por un argumento. Su resultado es de tipo booleano.

recorrido. Forma algorítmica básica que se utiliza para procesar los elementos de una secuencia. Un recorrido tiene como subtareas: ir al comienzo de la secuencia de elementos, procesar cada elemento, pasar de un elemento al siguiente y la condición de finalización.

repetición condicional. Herramienta de un lenguaje de programación que permite repetir comandos hasta que se cumpla una determinada condición.

repetición simple. Herramienta de un lenguaje de programación que provee una forma de combinación de comandos y permite establecer de manera fija la cantidad de veces que deben repetirse tales comandos.

repetir (comando). Comando para realizar la repetición simple de otros comandos. Consta de un número y un cuerpo (bloque de comandos que se van a repetir).

Rojo (expresión). Expresión literal de Gobstones que designa uno de los cuatro colores posibles de las bolitas.

Sacar (comando). Comando primitivo de Gobstones que indica la acción del cabezal que consiste en sacar una bolita del color del argumento en la celda sobre la que se encuentra.

secuencia (de comandos). Grupo de comandos organizados uno detrás del otro, que serán ejecutados en ese orden.

si ... entonces (comando). Variante de la alternativa condicional donde solo se indica un bloque de comandos cuando la condición es cierta.

si ... entonces / si no (comando). Comando que permite realizar la alternativa condicional entre dos bloques de comandos. Consta de una condición, un bloque de comandos para cuando la condición es cierta y un bloque de comandos para elegir cuando la condición es falsa.

siguiente (operador). Operador que, dada una expresión de tipo dirección, da como resultado una nueva expresión de tipo dirección que indica la dirección siguiente en el sentido de las agujas del reloj. El siguiente de Norte es Este, el de Este es Sur, etc.

sintaxis (textual) de un lenguaje de programación. Conjunto de símbolos, palabras y reglas para describir un programa en un lenguaje de programación.

***software*.** Componentes lógicos (datos y programas) necesarios para que una computadora realice tareas.

Sur (expresión). Expresión literal de Gobstones que designa una de las cuatro direcciones de movimiento del cabezal.

subtarea. Cada una de las partes en que se divide una tarea más grande para facilitar su resolución.

suma (operador de). Operador que permite combinar dos expresiones de tipo número para obtener una nueva expresión de tipo número que indica la suma de las cantidades involucradas.

tablero. Pieza rectangular dividida en casilleros denominados *celdas*.

tablero inicial. Tablero de Gobstones antes de comenzar la ejecución de un programa.

tablero final. Tablero de Gobstones al finalizar satisfactoriamente la ejecución de un programa.

***timeout*.** Tiempo que espera un programa para que ocurra un evento, antes de continuar la ejecución.

tipo de datos. Grupos de datos que comparten algunas características y se pueden usar para los mismos propósitos. En Gobstones hay cuatro tipos de datos: colores, direcciones, números y valores de verdad (o booleanos).

unidad central de procesamiento (CPU). Componente de *hardware* que se encarga de ejecutar una por una las instrucciones de un programa realizando operaciones aritméticas y lógicas. Además, controla el resto de los componentes de la computadora y dirige el flujo de datos entre ellos.

unidad de almacenamiento. Dispositivo que se integra a una computadora y que preserva información de manera prolongada, ya que no necesita suministro eléctrico.

unidad de disco rígido. Unidad de almacenamiento de datos que emplea un sistema de grabación magnética para almacenar archivos digitales. Se compone de uno o más platos o discos, unidos por un mismo eje que gira a gran velocidad dentro de una caja metálica sellada. Sobre cada plato y en cada una de sus caras se sitúa un cabezal de lectura y escritura, que flota sobre una delgada lámina de aire generada por la rotación de los discos.

variable (de un lenguaje de programación). Espacio reservado en la memoria en el que se puede almacenar un valor y al que se hace referencia con un nombre.

Verde (expresión). Expresión literal de Gobstones que designa uno de los cuatro colores posibles de las bolitas.

vestimenta. Conjunto de imágenes que reemplazan a las bolitas y permiten ver claramente distintos elementos representados mediante el uso de bolitas.

videojuego. Programa interactivo con fines lúdicos donde uno o más usuarios interactúan como jugadores.

y también (operador). Bloque que expresa el operador de conjunción lógica en Gobstones.

FOTOS E ILUSTRACIONES TÉCNICAS

Ada Lovelace: https://commons.wikimedia.org/wiki/File:Ada_Lovelace_Chalon_portrait.jpg. / Máquina tabuladora de Hollerith: https://commons.wikimedia.org/w/index.php?search=Hollerith&title=Special%3ASearch&go=Go#/media/File:Hollerith_census_machine.CHM.jpg / Computadora Colossus: https://commons.wikimedia.org/wiki/File:Colossus.jpg / EDVAC: https://commons.wikimedia.org/wiki/Category:EDVAC#/media/File:Edvac.jpg / Clementina: https://commons.wikimedia.org/wiki/File:Clementina.jpg / Commodore VIC-20: https://commons.wikimedia.org/wiki/Category:Commodore_VIC-20#/media/File:CBMVIC20P8.jpg / Impresora: https://www.freepik.es/foto-gratis/impresora-folios-blancos_977954.htm; Computadora Clementina - 01: Biblioteca Digital / Programa de Historia de la Facultad de Ciencias Exactas y Naturales, UBA.

Program.AR, Fundación Sadosky
Av. Córdoba 832, 5º piso.
Ciudad Autónoma de Buenos Aires, Argentina.

Ciencias de la computación para el aula : 1er. ciclo de secundaria / Pablo Ernesto Martínez López ... [et al.] ; compilado por Pablo Ernesto Martínez López ... [et al.] ; coordinación general de Vanina Klinkovich ; Hernán Czemerinski ; editado por Ignacio David Miller ; Alejandro Palermo ; editor literario Luz Luz María Rodríguez ; ilustrado por Luciano Andújar ... [et al.] ; prólogo de María Belén Bonello ; Fernando Pablo Schapachnik. - 1a ed . - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2019.
Libro digital, PDF - (Ciencias de la Computación para el aula / Klinkovich, Vanina; Czemerinski, Hernán; 3)

Archivo Digital: descarga
ISBN 978-987-27416-7-9

1. Informática. 2. Programación. I. Martínez López, Pablo Ernesto II. Martínez López, Pablo Ernesto, comp. III. Klinkovich, Vanina, coord. IV. Czemerinski, Hernán, coord. V. Miller, Ignacio David, ed. VI. Palermo, Alejandro, ed. VII. Luz María Rodríguez, Luz, ed. Lit. VIII. Andújar, Luciano, ilus. IX. Bonello, María Belén, prolog. X. Schapachnik, Fernando Pablo, prolog.
CDD 004.0712

Queda hecho el depósito que dispone la Ley 11.723
Ediciones Colihue.
Primera edición: Mayo de 2019.